HÖGSKOLAN SKÖVDE

Institutionen för kommunikation och information
Examensarbete i datavetenskap 10p
C-nivå
Vårterminen 2006

Analys och utveckling av en metod för distribution av data

Göran Grahn

HS-IKI-EA-06-609

Analys och utveckling av en metod för distribution av data

Examensrapport inlämnad av Göran Grahn till Högskolan i Skövde, för Kandidatexamen (B.Sc.) vid Institutionen för kommunikation och information. Arbetet har handletts av Anders Dahlbom.

2006-09-27

Härmed intygas att allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och att inget material är inkluderat som tidigare använts för erhållande av annan examen.

Signerat: _____

Analys och utveckling av en metod för distribution av data

Göran Grahn

Sammanfattning

Detta examensarbete omfattar utveckling och test av en metod för distribution av data i ett multidatabassystem. Testen av metoden har gjorts genom att den har implementerats i en fallstudie. Fallstudien berör en organisation som har behov av att integrera tre av sina databassystem. Detta för att information ska kunna utbytas mellan dessa. Integrationen av databassystemen har gjorts på ett sådant sätt att de data som distribueras från dessa skickas till en central databas, med SQL Server som databashanterare.

Det verktyg som har valts att användas i examensarbetet för att lösa uppgiften är Data Transformation Services, DTS, i SQL Server.

Nyckelord: Databassystem, SQL Server, DB2, multidatabassystem

Innehållsförteckning

1 Introduktion

Användandet av databassystem är vanligt idag och ofta lagras data på flera olika platser i stora nätverk. Det är heller inte ovanligt att data lagras i olika former, allt från enkla filer till mer avancerade databaser. Samtidigt har även tillgängligheten till Internet givit större möjligheter till att distribuera data.

Detta kapitel pekar på de behov som finns för att distribuera data och dessutom på de behov som finns att hantera dess metadata.

1.1 Behovet av att distribuera data

En av de ursprungliga drömmarna med databassystem (Bell och Grimson 1992) var att samla samtliga data, som användare behövde, till en gemensam plats, d.v.s. en databas. Användarna skulle sedan kunna komma åt och/eller uppdatera dessa data i databasen med hjälp av ett antal verktyg. Dessa verktyg skulle innehålla språk för metadata och funktioner för åtkomst- och uppdatering av data. Den här drömmen har, genom erfarenhet, visat sig fungera så länge antalet användare av ett databassystem har hållits på en rimlig nivå, beroende på dess prestanda. Däremot har det också visat sig att organisationer som använder sig av databassystem kan få problem om antalet användare till dem ökar (Bell och Grimson 1992). Detta problem gör att en växande organisation kan förvänta sig att användarna blir frustrerade om de data, som användarna behöver, hålls centraliserade (Bell och Grimson 1992).

Det är inte bara prestandaproblem som är enda anledningen till att det finns ett behov av att distribuera data (Bell och Grimson 1992). Organisationer har ofta problem med att de har data utspridda på flera olika platser i sina nätverk. En anledning till att de har placerat data lokalt kan vara att användarna i organisationerna har flera olika synsätt på hur problemen angående placeringen av data ska lösas och inte kan enas om en gemensam lösning. När sedan ledningen till en organisation exempelvis ska sammanställa vissa data från sina lokala databaser, så uppstår behovet att kunna distribuera dessa data (Bell och Grimson 1992).

1.2 Problem med hantering av metadata vid distribution av data

Om det finns skillnader i datamodellerna på hur data är lagrat i de databaser som finns i ett nätverk, t.ex. mellan relationsdatabaser och objektdatabaser, ger det problem om data ska distribueras mellan dessa. De data som ska distribueras måste då beskrivas med hjälp av metadata innan de kan användas. Varför data inte har samma datamodell i olika databaser kan bl.a. bero på att de databashanterare som används till respektive databas skiljer sig åt. Om datamodeller skiljer sig åt mellan databaser, d.v.s. heterogena multidatabassystem, så blir lösningarna för hanteringen av metadata mer komplicerade än om de var lika (Piattini och Diaz 2000).

Att endast distribuera databasers scheman räcker normalt inte till idag vid distribution av data från databaser. Detta för att databasers scheman som bäst kan ge en syntaktisk beskrivning av de data som finns i databasen. Databasers scheman ger nästan inget stöd för betydelsen av dessa data. Genom detta har behoven lett fram till utvecklingen av datakatalogssystem, Data Dictionary Systems (Bell och Grimson 1992). Datakatalogsystem består av datakataloger och deras datakataloghanterare. Datakataloger innehåller metadata för databaser. Datakataloger, klassificeras antingen som aktiva eller passiva beroende på hur deras datakatalogssystem är uppbyggda och integrerade med respektive databashanterare. Datakataloger i

heterogena multidatabassystem är vanligtvis passiva men fördelarna om de kunde vara aktiva är många (Bell och Grimson 1992).

1.3 Vad rapporten behandlar

Den här rapporten behandlar ett fall inom problemområdet för distribution av data mellan databassystem i ett nätverk. Rapporten har som mål att ta fram en metod för att ge åtkomst till data från olika databassystem. Fallstudien i rapporten berör en organisation som använder sig av en central databas för att distribuera data mellan deras olika databassystem.

1.4 Översikt av rapporten

Kapitel 1 innehåller introducerande text om rapporten. Efter introduktionen, i kapitel 2, presenteras grundläggande begrepp på bl.a. databassystem och distribuerade databaser. Detta för att rapporten ska bli så självförklarande som möjligt. Kapitel 3 introducerar och beskriver problemet som behandlas samt vilket mål som avses att nå. Kapitel 4 beskriver olika problem och möjligheter som uppkommer i samband med att en central databas används för distribuera data i ett multidatabassystem. Kapitel 5 beskriver den metod som utvecklats och kapitel 6 dess resultat vid användning av den i en fallstudie. Kapitel 7 beskriver och diskuterar om slutsatser utifrån resultatet av examensarbetet.

2 Bakgrund

Detta kapitel beskriver översiktligt begrepp och grundläggande kunskap som används av de resterande kapitlen i rapporten. Det beskriver begrepp inom databassystem och arkitekturer som finns hos distribuerade databaser. Dessutom beskrivs översiktligt fallstudien och dess ingående system.

2.1 Grundläggande begrepp

Nedanstående begrepp används i rapporten och definieras baserat på (Padron-McCarthy, 2006) enligt följande:

- Databas
  En databas är en samling data vilka på något sätt har relation till varandra.
- Databashanterare
  Ett program eller ett system av program som kan hantera en eller flera databaser. Exempel på databashanterare är Microsoft SQL Server.
- Databassystem
  Detta definieras som kombinationen av en databas och en databashanterare.
- Multidatabas
  En multidatabas består av flera databaser som kopplats ihop på ett sådant sätt att det går att söka i deras data samtidigt.
- Distribuerad databas
  En distribuerad databas har data utspritt på flera datorer. Sökningen av data kan göras som om det vore en vanlig databas, d.v.s. den har placeringstransparens.
- Datamodell
  En datamodell är ett sätt att beskriva världen. Ett exempel är relationsdatamodellen som används i relationsdatabaser.
- Schema
  En beskrivning av de data som finns och komma att kan finnas i en databas. En relationsdatabas har till exempel schema för vilka tabeller som ingår i databasen och vilka kolumner tabellerna har.
- Metadata
  Data om data. Metadata används bl.a. i rapporten som begrepp för de data en databashanterare behöver för att kunna veta vad det är för data som hanteras i en databas. Exempel på metadata är databasens schema samt datatyper på kolumner i tabeller.

Nedanstående begrepp definieras baserat på (Piattini och Díaz 2000) enligt följande:

- Placeringstransparens
  Med placeringstransparens menas att ändring av den fysiska platsen där data lagras, i en distribuerad databas, ska kunna ske utan att applikationers kod behöver ändras.

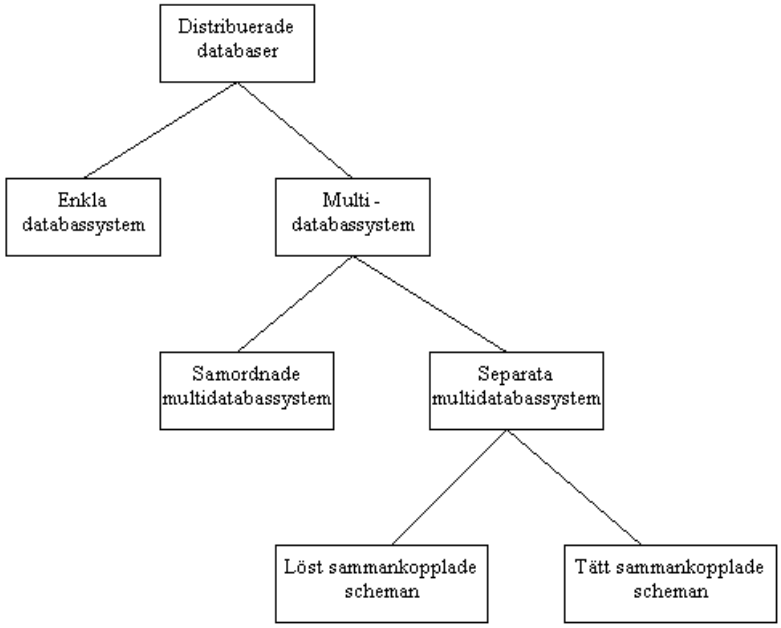
2.2 Distribuerade databaser

Det är många aspekter att ta hänsyn till gällande distribuerade databaser och lösningarna varierar ofta från fall till fall. Detta kapitel går igenom olika arkitekturer

som finns hos distribuerade databaser samt beskriver en modell för olika scheman i en distribuerad databas.

2.2.1 Arkitekturer hos distribuerade databaser

Det finns ett antal olika arkitekturer hos distribuerade databaser. Det finns också ett antal studier som delar in dessa distribuerade databaser i olika grupper beroende på vilken arkitektur de har. Detta kapitel beskriver den taxonomi som Piattini och Díaz (2000) använder. Beroende på vilka egenskaper en distribuerad databas har så delas den in i någon av dessa arkitekturer (se Figur 1).



Figur 1 Arkitekturer hos distribuerade databaser enligt Piattini och Díaz (2000)

Nedan ges en kort beskrivning av vad en distribuerad databas ska ha för egenskaper för att passa in i respektive arkitektur:

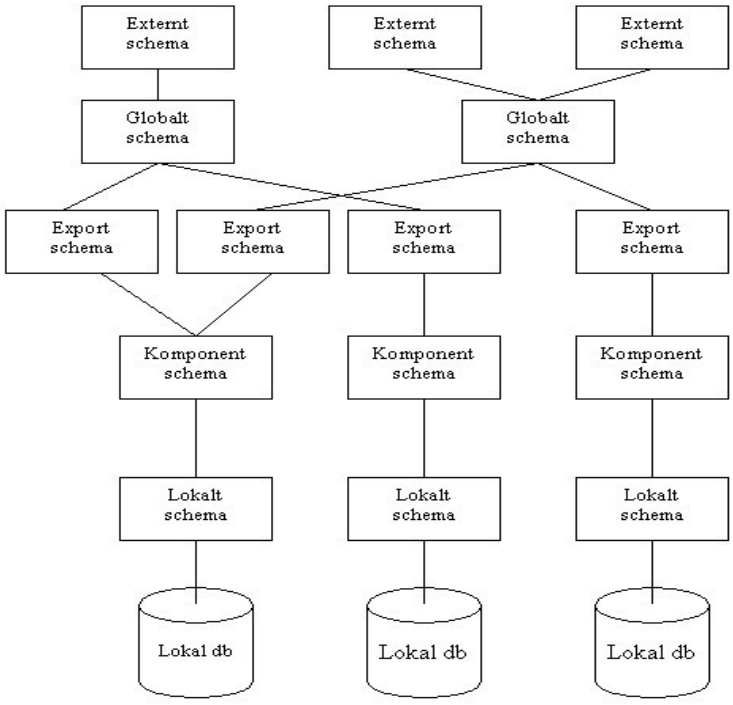
- Enkla databassystem
  Detta består av ett enda databassystem som styr flera databaser.
- Multidatabassystem
  Detta är uppbyggt av flera databassystem som var och ett har kontroll över sina egna databaser.
- Samordnade multidatabassystem
  Denna arkitektur har egenskaperna i multidatabassystem samt att de olika databasernas uppbyggnad, av t.ex. scheman, styrs av en gemensam databasadministratör.
- Separata multidatabassystem
  Denna arkitektur har egenskaperna i multidatabassystem fast de olika databasernas uppbyggnad styrs av flera lokala databasadministratörer.
- Löst sammankopplade scheman
  Denna arkitektur omfattar egenskaperna i separata multidatabassystem men har inga gemensamma scheman för att integrera databaserna till de olika databassystemen.

- Tätt sammankopplade scheman
  Denna arkitektur omfattar egenskaperna i separata multidatabassystem samt har gemensamma scheman för att integrera databaserna mellan de olika databassystemen.

Om samtliga databaser i en distribuerad databas har samma datamodell, t.ex. att samtliga databaser är relationsdatabaser, så är den distribuerade databasen homogen. Om de har olika datamodeller är den heterogen.

2.2.2 Modell med fem nivåer för heterogena multidatabassystem

Beroende på vilken arkitektur som en distribuerad databas har så finns det modeller med olika många nivåer för att integrera de ingående databassystemen i den. Heterogena multidatabassystem behöver fem nivåer (se Figur 2). Homogena multidatabassystem behöver bara fyra nivåer då komponentscheman inte behövs (Piattini och Díaz 2000). Detta kapitel beskriver den modell som finns i (Piattini och Díaz 2000).



Figur 2 Modell med fem nivåer för heterogena multidatabassystem enligt Piattini och Díaz (2000)

Varje nivå innehåller en typ av scheman:

1. Lokala scheman
   De scheman som finns för respektive lokal databas.
2. Komponentscheman
   Dessa används för att kunna skapa en gemensam datamodell mellan de olika lokala databaserna. De kan vara en variant av en ER-modell eller en objektorienterad datamodell.
3. Exportscheman
   Till varje komponentschema används ett eller flera exportscheman för att definiera vad som ska kunna nås i de lokala databaserna.

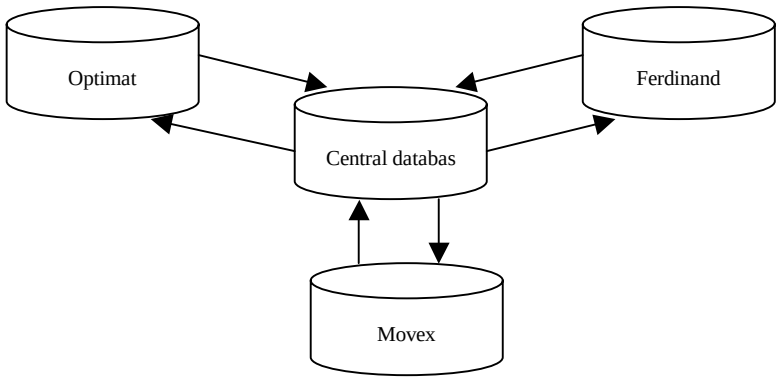
4. Globala scheman
   För att integrera ett eller flera exportscheman används ett globalt schema.
5. Externa scheman
   Externa scheman är vyer över globala scheman och innehåller information som en specifik applikation behöver från en distribuerad databas.

2.3 Fallstudie

Denna fallstudie har som syfte att ge ett exempel på hur metoden som utvecklas i rapporten fungerar i praktiken. Metoden är implementerad i det här fallet för att integrera en organisations tre olika databassystem så att de ska kunna utbyta information mellan varandra.

2.3.1 De ingående systemen

Data kommer att skickas genom en central databas. För att detta ska fungera måste databassystemet för den centrala databasen stödja flera olika typer av gränssnitt. Detta för att den ska kunna nå data från samtliga av de ingående databassystemen (se Figur 3).



Figur 3 De ingående databassystemen

2.3.1.1 Movex

Movex är ett affärssystem som används inom organisationer för att hantera stora delar av deras affärskritiska funktioner, t.ex. beslutsstöd, budgetering, EDI, order, lager och fakturering. Systemet är installerat och körs på en AS400 med databashanteraren DB2.

2.3.1.2 Optimat

Detta är ett receptsystem för chark och bageriprodukter. Det hanterar vissa data, såsom recept för tillverkning av produkter och innehållsförteckningar. Systemet har en databas med Microsoft SQL Server som databashanterare.

2.3.1.3 Ferdinand

Ett system för bl.a. expediering av tillverkningsordrar i produktion och märkning av produkter. Data för märkningen av produkterna kommer till stor del ifrån Optimat och Movex. Systemet har en databas med Microsoft SQL Server som databashanterare.

2.3.1.4 Central databas

Denna databas, som ska hantera data och metadata vid distribution av data från de olika systemen, byggs på en Microsoft SQL Server. De register som berörs är bl.a. artiklar, innehållsförteckningar, recept och receptrader

2.3.2 Databashanterare

De två ingående databashanterarna i fallstudien är:

- Microsoft SQL Server som är en produkt från Microsoft och är databashanterare för relationsdatabaser.
- DB2 som är en produkt från IBM och är även denna en databashanterare för relationsdatabaser.

2.3.3 Analys av systemarkitektur

Enligt taxonomin som används av (Piattini och Díaz 2000), se rubrik 2.2.1, hör de ingående databassystemen till gruppen samordnade multidatabassystem. Detta eftersom:

- Det är flera separata databashanterare som samspelar. Därför kan de ingående databassystemen inte hamna i gruppen för enkla databassystem.
- Det är en databasadministratör som ansvarar för samtliga databaser i den distribuerade databasen.

Eftersom de ingående databaserna i fallstudien är relationsdatabaser medför det att den distribuerade databasen är homogen. Av de olika arkitekturerna så är det heterogena multidatabassystem som är mest komplicerade att implementera (Piattini och Díaz 2000). För homogena blir lösningen inte lika komplicerad. För att lösa dessa används en modell med fyra nivåer för homogena multidatabassystem (se rubrik 2.2.2).

2.3.4 Krav vid distribution av data via den centrala databasen

För att effektivisera distributionen av data mellan databassystemen finns följande två krav på den centrala databasen:

1. De data som ska föras över från en databas till en annan ska hanteras i den centrala databasen. Data kan läggas till, tas bort eller uppdateras på valfritt sätt men måste vara kompletta innan mottagande databassystem får hämta data från den centrala databasen. För att garantera detta måste transaktionshantering användas av sändande system varje gång data uppdateras i den centrala databasen. Detta innebär att en transaktion måste startas i den centrala databasen innan data får uppdateras.
2. För att minimera nätverkstrafiken ska det undvikas att stora mängder av data skickas om på nytt varje gång vid uppdatering i den centrala databasen. Vid borttag, uppdatering eller tillägg bör endast berörda data hanteras.

3 Problembeskrivning

Detta kapitel beskriver målet med examensarbetet samt det problem som givit upphov till målet.

3.1 Åtkomst av data via en central databas i ett multidatabassystem

Mindre organisationer har normalt inte likadana behov som stora organisationer gällande distribution av data. Detta får betydelse vid valet av det/de verktyg som organisationerna sedan använder. För mindre organisationer kan kostnaden för licenser till avancerade verktyg för distribution av data bli för stora. Istället för detta kan då en alternativ lösning vara att välja något av de verktyg som ofta levereras med en databashanterare. Samtidigt är det fortfarande viktigt att de lösningar som tas fram, i möjligaste mån, uppfyller de krav som ställs på ett datakatalogsystem. Att därför försöka nå de krav som finns för datakatalogsystem med fungerande metoder passande de inkluderade verktygen hos de ingående databashanterarna i en organisation, kan ge kostnadsbesparingar för organisationen ifråga. Vilka databashanterare som respektive organisation använder sig av kan variera och måste därför analyseras från fall till fall. Detta fall inriktar sig på en organisation som har behov av att distribuera data mellan databaser med databashanterarna SQL Server och DB2. En möjlig väg för att kunna distribuera data mellan dessa databaser kan vara att skapa en central databas (Piattini och Díaz 2000). Denna ska ha som uppgift att hantera data och metadata mellan de olika databaserna i organisationen. Om tillvägagångssättet fungerar i det här fallet borde det även passa i andra fall där organisationer ska distribuera data från databaser.

3.2 Mål

Analysera och utveckla en metod för att distribuera data, via en central databas, i ett multidatabassystem bestående av SQL Server och DB2.

Problemet, som är beskrivet i kapitel 3.1, har lett fram ovanstående mål och med stöd av diskussionerna ovan kommer en central databas att användas, i det här fallet med en SQL Server som databashanterare.

3.3 Avgränsningar

Arbetet avgränsas till att utveckla en metod så att den uppfyller de behov som ställs för att få fram en lösning till fallstudien. Egenskaper som kan behövas för andra situationer, t.ex. skalbarhet och bättre plattformsoberoende, är inte nödvändiga vid utvecklandet av metoden, men önskvärda. Arbetet i rapporten avgränsas till att utveckla överföringar från lokala databaser med Microsoft SQL Server som databashanterare samt att anpassa den centrala databasen för att ta emot data som ska distribueras från lokala databaser med DB2 som databashanterare.

3.4 Delmål

De delmål som har satts upp för att nå det slutliga målet är följande

1. Val av verktyg och funktioner som är inkluderade i de två databashanterarna, för överföring av data till och från databaser.
2. Val av metod för distribution av data med de olika databashanterarna.
3. Utveckla metod för distribution av data mellan databaserna i fallstudien.

4. Implementera utvecklad metod i fallstudien
5. Analys och slutsats av resultat vid användandet av implementerad metod i fallstudien

4 Analys av problemställning

Detta kapitel beskriver olika problem och möjligheter som uppkommer i samband med att en central databas används för distribuera data i ett multidatabassystem. Kapitlet beskriver även ett antal olika möjligheter avseende tillgänglighet och distribution av data.

4.1 Jämförelse mellan ett multidatabassystem med en central databas och ett datalager

Ett datalager har som uppgift att hämta data från ett antal databaser för att sedan spara dessa data i en databas. Databasen kan sedan användas på olika sätt, allt från att vara en replika av databaserna till att visa mer avancerade sammanställningar av data från databaserna (Jarke et al., 2000). Den centrala databasen i multidatabassystemet i rapporten har däremot som uppgift att distribuera data från ett antal databaser i multidatabassytemet. Det närmaste sättet inom området dataintegration som inte tillhör området datalager är att använda sig av materialiserade vyer, d.v.s. lagrade vyer, (Jarke et al., 2000). Med utgångspunkt från detta blir den centrala databasen ett datalager om data måste replikeras från de lokala databaserna till den centrala databasen. SQL Server 2000 saknar stöd för materialiserade vyer vilket gör att data måste replikeras till den centrala databasen eller hämtas direkt ur de lokala databaserna genom, t.ex. vyer. Även databashanteraren DB2 saknar stöd för dessa vyer mot databashanteraren SQL Server, vilket gör även här att data måste replikeras till den centrala databasen eller hämtas direkt ur de lokala databaserna. Om data replikeras blir den centrala databasen ett datalager enligt definition, men det kan däremot i vissa fall bli nödvändigt ur prestandasynpunkt att replikera data till den centrala databasen. I senare versioner av dessa databashanterare kan det kanske bli möjligt att göra materialiserade vyer till den centrala databasen.

4.2 Egenskaper hos ett multidatabassystem med en central databas

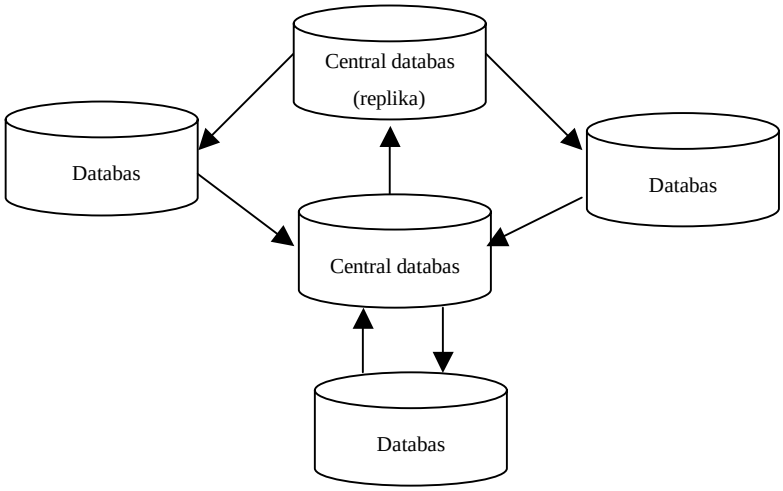
De egenskaper som uppkommer när en central databas används i ett multidatabassystem är både fördelaktiga och problematiska. En fördelaktig egenskap är att data kan hållas placeringstransparent (Piattini och Díaz 2000), se kapitel 4.2.1. Detta förenklar distribution av data från databaser i ett multidatabassystem. Ett problem är dock att fel eller krascher i den centrala databasen påverkar hela multidatabassystemet (Piattini och Díaz 2000). Det finns olika lösningar för att hantera detta problem, se kapitel 4.2.2.

4.2.1 Placeringstransparens

Placeringstransparens innebär, i fallet med ett multidatabassystem som har en central databas, att varje system måste fråga den centrala databasen efter data som är distribuerade från de lokala databaserna. Den centrala databasen har sedan i uppgift att avgöra hur dessa data ska ges åtkomst till för respektive system. Hur datat ges åtkomst till beror på de sätt de är distribuerade från de lokala databaserna.

4.2.2 Replikering

Att göra en replika av den centrala databasen (se Figur 4) minskar risken för krascher som kan påverka hela multidatabassystemet (Piattini och Díaz 2000). Replikering ger även möjlighet till lastbalansering vilket kan ge bättre möjligheter till skalbarhet hos den centrala databasen. Hur den centrala databasen väljs att utformas för att hållas tillgänglig, med t.ex. replika, blir en fråga om kostnad jämfört med tillgänglighet.



Figur 4 Exempel på replika av en central databas

4.3 Olika sätt att distribuera data med en central databas

Ett multidatabassystem med en central databas har möjlighet att distribuera data på olika sätt, vilka har både fördelar och nackdelar. Några av dessa sätt är att:

1. arbeta direkt med distribuerad data i lokala databaser, genom den centrala databasen.
   Detta förfarande innebär att den centrala databasen hämtar distribuerad data från respektive lokal databas till den centrala databasen först när den efterfrågas. Den hämtade datan förkastas sedan och får hämtas på nytt vid en ny förfrågan.
2. arbeta med replikerad data från lokala databaser, i en central databas. Med detta förfarande replikeras distribuerade data från de lokala databaserna till en central databas i förväg. Vid en förfrågan på dessa distribuerade data kan de hämtas direkt ur den centrala databasen.

Dessa två sätt har brister som kan göra ett multidatabassystem ineffektivt. Det första sättet ger stor belastning på bl.a. nätverk om det är stora datamängder som efterfrågas varje gång. Det andra sättet blir ineffektivt om det är stora datamängder som ska lagras i den centrala databasen. Om de kombineras, beroende på hur det distribuerade datat ska användas, kan prestandan förbättras jämfört med om de används var för sig.

4.4 Skillnader mellan scheman vid överföring av data till en central databas

Om data ska överföras från en lokal databas kan det bli det problem om det globala schemat skiljer sig från exportschemat (se Rubrik 2.2.2). Det kan t.ex. röra sig om att primärnycklar till tabeller är olika och att datatyper skiljer sig mellan dessa scheman:

1. Problem med att primärnycklar till tabeller är olika vid överföring av data. Om det globala schemat skiljer sig från exportschemat avseende nycklar till tabeller kan det vid överföring av data mellan dessa ge svårigheter som leder till risk för att det blir inkonsistens mellan tabellerna.

2. Datatyper
   Problem kan uppstå vid distribution av data om de olika databashanterarna i ett multidatabassystem inte hanterar varandras datatyper. Detta kan t.ex. gälla datatyper på kolumner i en tabell. I SQL Server finns datatypen float som inte stöds i DB2 (Knight, 2001). Detta gör att om data ska replikeras från en lokal databas, med DB2 som databashanterare, till den centrala databasen, med SQL Server som databashanterare, kan konvertering behöva göras på en del av dessa data. De data som blir berörda är de med datatyper som den mottagande databashanteraren inte har stöd för.

5 Utveckling av metod

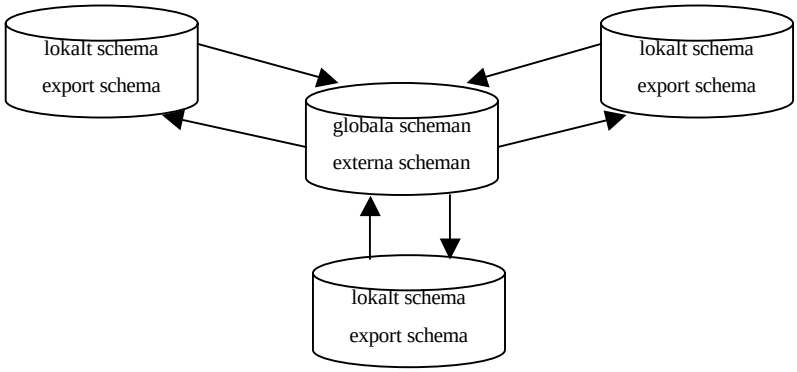
Av de två sätt som presenteras i kapitel 4.3 koncentreras arbetet på att utveckla en metod som distribuerar data från lokala databaser till en central databas genom replikering av data. För att det ska vara möjligt behöver metoden använda sig av ett eller flera verktyg för distribution och överföring av data till den centrala databasen. Dessa verktyg behöver ha stöd för databashanterarna SQL Server och DB2. Detta kapitel beskriver dels vilka verktyg som valts ut för metoden samt hur metoden är uppbyggd och implementerad.

5.1 Strukturering av scheman för distribution av data

Den centrala databasens uppgift är att ge åtkomst till distribuerad data från de lokala databaserna. För att detta ska fungera måste data distribueras från de lokala databaserna. Data distribueras från de lokala databaserna med hjälp av ett eller flera scheman för respektive lokal databas. Detta kapitel redogör vilka scheman som behövs i den distribuerade databasen och till hjälp finns en modell med ett antal nivåer beroende den distribuerade databasens arkitektur (se kapitel 2.2.2).

5.1.1 Placering av scheman

Då multidatabassystemet är homogent används en modell med fyra nivåer av scheman, d.v.s. lokala-, export-, globala- och externa scheman. Lokala- och exportscheman placeras i de lokala databaserna medan globala- och externa scheman placeras i den centrala databasen (se Figur 5)



Figur 5 Placering av scheman

Den här placeringen av scheman ger databasadministratören för respektive lokal databas möjligheten att själv bestämma vilka data som ska distribueras från databasen. De data som ska distribueras görs tillgängligt för den centrala databasen via exportscheman.

5.1.2 Struktur i den centrala databasen

Den centrala databasen består dels av de globala scheman som finns i multidatabassystemet och dels av de externa scheman som används för att olika system ska kunna nå de data som distribueras från de lokala databaserna i multidatabassystemet. Transformering av data från globala scheman till externa scheman kan göras genom t.ex. vyer i externa scheman eller aktiva regler i globala scheman.

5.2 Val av verktyg för överföring och åtkomst av data i databaser

Då den metod som utvecklas är anpassad för att replikera data, från lokala databaser till den centrala databasen, behöver lämpliga verktyg för denna uppgift väljas. De verktyg som är aktuella för att nå målet beskrivs i detta kapitel. Det kan i en del fall behövas olika verktyg för att kommunicera med DB2 jämfört med SQL Server. Verktygen är också olika effektiva beroende på vilket sätt som data ska distribueras från de lokala databaserna till den centrala databasen, i ett multidatabassystem. Då den centrala databasen har SQL Server som databashanterare koncentreras valet av verktyg till endast de som är inkluderade i denna databashanterare.

5.2.1 Verktyg i SQL Server

SQL Server har ett antal möjligheter för att flytta eller arbeta med data i databaser. Några av dessa, relevanta för lösningen, är:

1. DTS (Data Transformation Services)
   DTS är ett verktyg i SQL Server som har till syfte att flytta och arbeta med data. Verktyget rekommenderas till att skapa schemalagda förflyttningar av data, både inom och mellan databaser (Dalton och Whitehead 2001). Detta görs genom att skapa ett så kallat DTS-paket där funktionaliteten för överföringen av data finns. Därefter kan detta paket exekveras på schemalagda tider eller manuellt. Oberoende av databashanterare, SQL Server eller DB2, kan DTS användas. Detta om stöd för ODBC, Open Database Connectivity, finns för att nå data hos de olika databaserna.
2. Länkade datakällor
   Med länkade datakällor skapas en möjlighet att koppla sig till en annan databas och sedan få möjlighet att via SQL kunna hämta och arbeta med data i den databasen.
3. Replikering
   Detta verktyg i SQL Sever är anpassat för att kunna replikera data från databaser i SQL Server till andra databaser med SQL Server.

Dessa verktyg har olika egenskaper vilket medför att de endast lämpar sig till vissa av de sätt som data kan distribueras från de lokala databaserna.

5.2.2 Val av verktyg

DTS väljs som verktyg för att replikera data till den centrala databasen eftersom verktyget kan nå data från båda databashanterarna, SQL Server och DB2 (Dalton och Whitehead 2001). DTS-paket i SQL Server rekommenderas att användas vid överföringar av data mellan databaser (Dalton och Whitehead 2001).

Replikering väljs inte då det och är olika verktyg beroende på om det är SQL Server eller DB2.

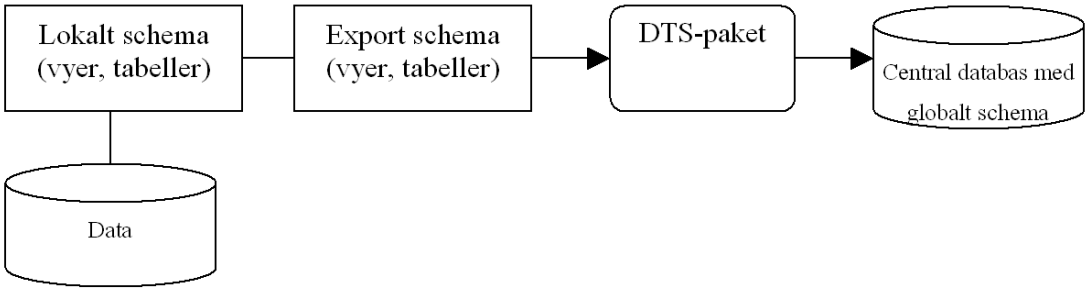
5.3 Beskrivning av metoden

Metoden har som syfte att överföra data från ett exportschema till ett globalt schema. Detta beroende på om databashanteraren för den lokala databasen är SQL Server eller DB2. Nedan ges några synpunkter på data överförda till den centrala databasen från en lokal databas med databashanteraren:

1. SQL Server.
   Då den centrala databasen har SQL Server som databashanterare kan detta

förenkla lösningen jämfört med andra databashanterare. I ett sådant fall kan metoden bestå av ett DTS-paket i SQL Server som replikerar data från exportschemat i den lokala databasen till det globala schemat i den centrala databasen (se Figur 6).

2. DB2.
   Med DB2 bör det finnas fler aspekter att ta hänsyn till än om databashanterarna hos den centrala- och den lokala databasen var lika. Då kan även hänsyn behöva tas till att hantera inkompabilitetsproblem mellan dessa. Även i detta fall kan metoden bestå av ett DTS-paket som replikerar data från exportschemat, i den lokala databasen, till det globala schemat, i den centrala databasen (se Figur 6).



Figur 6 Överföring från exportschema med DTS-paket till central databas

För att effektivisera replikeringen av de data som skickas till den centrala databasen och förhindra att omsändning av redan skickade data sker, behöver exportschemat förses med ytterligare funktionalitet. Exportschemat behöver kunna skilja data som har replikerats till den centrala databasen mot de data som inte har replikerats. Detta kan göras på olika sätt, t.ex. genom extra kolumner i tabeller. Hur detta löses kan skilja sig från exportschema till exportschema.

Om exportschemat och det globala schemat har konstruerats på ett sådant sätt att det är anpassat till att data enkelt ska kunna replikeras till den centrala databasen kan konstruktionen av DTS-paketen förenklas. Detta i jämförelse med att DTS-paketen måste transformera data som inte har likadant utseende mellan exportschema och globalt schema. Om DTS-paketen kan hållas enkla finns större möjligheter att göra dem mindre resurskrävande (Dalton och Whitehead 2001).

5.3.1 Fördelar med metoden

Metoden som tas fram är konstruerad för att replikera data från de lokala databaserna till den centrala databasen i ett multidatabassystem. Detta sätt att distribuera data är inte effektivt i alla situationer där data behöver distribueras i ett multidatabassystem. Det kan däremot i kombination med andra metoder för att distribuera data ge bra prestanda på multidatabassystemet (se kapitel 4.3). Metoden följer en modell med fyra nivåer för homogena multidatabassystem (se kapitel 2.2.2) (Piattini och Díaz 2000).

5.3.2 Nackdelar med metoden

Ett problem vid replikering av data är att det finns risk för att replikan inte är helt aktuellt uppdaterad med originalet. Denna risk finns också med den här metoden. Detta eftersom ändringar gjorda på data som ska distribueras från de lokala databaserna i multidatabassystem inte nödvändigtvis samtidigt replikeras till den centrala databasen. Detta sker asynkront.

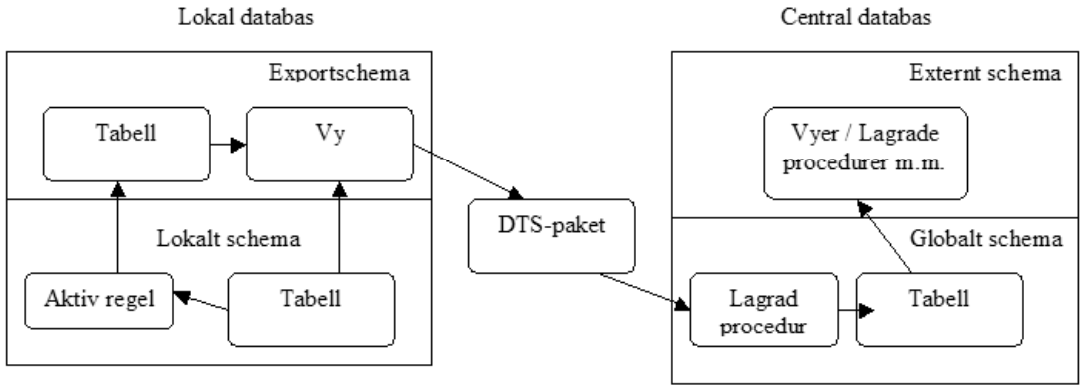
5.4 Implementation

Detta kapitel beskriver implementationen av metoden för att replikera data till den centrala databasen i multidatabassystemet. För att exemplifiera den implementerade metoden har ett schema använts som innehåller en tabell. I testfallet har däremot implementationen av metoden gjorts på tabeller i en av de lokala databaserna till den centrala databasen, båda med SQL Server som databashanterare.

5.4.1 Genomgång av implementerad metod

Ett problem vid distribution av data från en databas kan vara att den inte är anpassad för att distribuera data. Det är inte säkert att det lokala schemat i databasen har konstruerats för att distribuera data. Detta medför antingen att databashanteraren måste ha inbyggt stöd för att distribuera data eller att exportschemat och det lokala schemat måste anpassas så att det kan hantera distribution av data. I det här fallet skapas en aktiv regel i det lokala schemat på tabeller vars data ska distribueras till den centrala databasen.

För att öka effektiviteten vid replikering av data till den centrala databasen behöver exportschemat kunna skilja data som redan har replikerats till den centrala databasen mot data som inte har. Detta går att lösa genom att information sparas i den lokala databasen om vilka data som har replikerats eller ska replikeras. Nedanstående lösning har dessa egenskaper (se Figur 7)



Figur 7 Lösning med aktiv regel i lokal databas och DTS-paket

För denna metod att distribuera data, från en tabell i en lokal databas, till den centrala databasen i ett multidatabassystem används en aktiv regel. Den har som funktion att lagra information om de poster, som förändras i tabellen i det lokala schemat, till tabellen i exportschemat. En vy används sedan för att kunna få fram information om vilka poster som har förändrats samt vilka värden de har efter förändringen. Posterna tas bort ur tabellen i exportschemat efter det att de har replikerats till den centrala databasen. På detta sätt hålls antalet poster i tabellen, i exportschemat, på en låg nivå.

Den centrala databasen har en tabell i det globala schemat som data från tabellen i den lokala databasen kan replikeras till. Uppdateringar i tabellen i den centrala databasen sker genom en lagrad procedur.

Med t.ex. vyer eller lagrade procedurer, i externa scheman, kan sedan olika system ges åtkomst till data i den centrala databasen.

Transaktionshanteringen i DTS-paketet tar hand om och ser till att inga data lämnas ofullständigt uppdaterade i någon databas vid t.ex. fel i överföringen. Detta sker med hjälp av Microfts Distributed Transaction Coordinator, MS-DTC, som är inkluderad i

SQL Server. MS-DTC kan hantera transkationer för ett DTS-paket om arbetsuppgifterna i DTS-paktetet deltar i en transaktion (Dalton och Whitehead 2001). Nedan följer en beskrivning på de delar som ingår i implementationen.

5.4.1.1 Tabell i lokalt schema och exportschema samt vy

De tabeller som replikeras från de lokala databaserna i fallstudien, med SQL Server som databashanterare, replikeras på samma sätt som exempeltabellen nedan görs. Exempeltabellen, Tab_Objekt har följande utseende, i transact-SQL för SQL Server:

```
CREATE TABLE [Tab_Objekt] (
        [Id] [int] NOT NULL ,
        [Beskrivning] [varchar] (50) NULL ,
        [Vikt] [numeric](5, 1) NULL ,
        [Längd] [numeric](5, 1) NULL ,
        CONSTRAINT [PK_Tab_Objekt] PRIMARY KEY  CLUSTERED
        ( [Id] )  ON [PRIMARY]
) ON [PRIMARY]
```

Tabellen, Exp_Objekt, i exportschemat har likadan uppsättning av nyckelkolumner som tabellen Tab_Objekt. Dessutom har tabellen en kolumn för att kunna lagra information om raden ska läggas till/uppdateras, d.v.s. har värdet U i kolumnen DBOp, eller tas bort, d.v.s. har värdet D i kolumnen DBOp, i den centrala databasen. Exp_Objekt har följande utseende, i transact-SQL för SQL Server:

```
CREATE TABLE [Exp_Objekt] (
        [Id] [int] NOT NULL ,
        [DBOp] [char](1) NULL                   --Kolumn för information om tillägg eller borttag av respektive post
        CONSTRAINT [PK_Exp_Objekt] PRIMARY KEY  CLUSTERED
        ( [Id] )  ON [PRIMARY]
) ON [PRIMARY]
```

Vyn, Vw_Objekt, i exportschemat har som uppgift att korsa tabellen Tab_Objekt och Exp_Objekt med varandra. Anledningen till att en vy används i exportschemat istället för att bygga ut tabellen Exp_Objekt med samtliga kolumner är för att undvika redundans. Vyn har följande utseende, i transact-SQL för SQL Server:

```
CREATE VIEW [VW_Objekt]
As
        Select  t2.Id,t1.Beskrivning,t1.vikt,t1.Längd,t2.DBOp
        From    Tab_Objekt as t1
                Right join
                Exp_Objekt as t2
                On  (t1.id=t2.id)
```

5.4.1.2 Aktiv regel för uppdatering av data till exportschemat

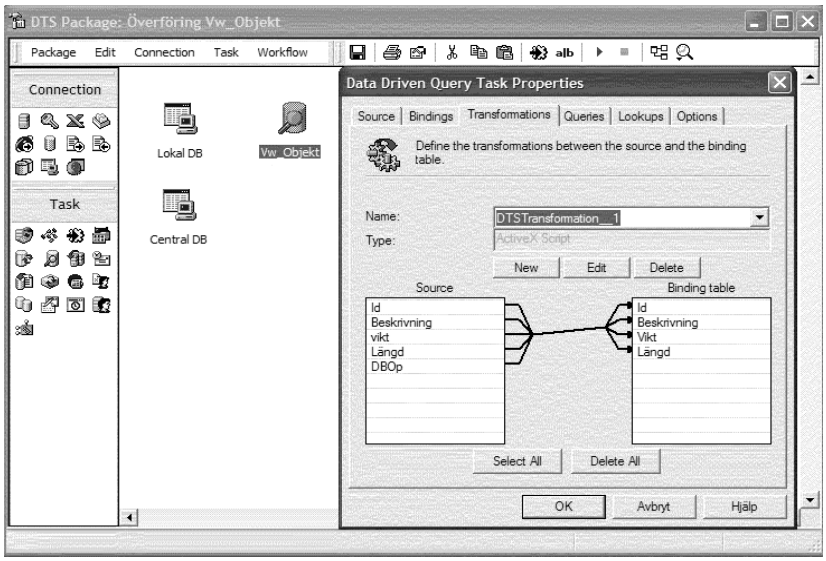
Denna aktiva regel lägger upp poster i tabellen Exp_Objekt när poster läggs till, ändras eller tas bort ur tabellen Tab_Objekt. De är enkelt konstruerade i rapporten för att förtydliga deras funktion:

```
CREATE TRIGGER Trg_Tab_ObjektIns ON [dbo].[Tab_Objekt]   --Aktiv regel för updaterade eller nyinstatta poster
FOR INSERT, UPDATE
```

```
AS
        Delete  Exp_Objekt              -- Ta bort existerande poster  i Exp_Objekt som i nästa sql-sats
        From    Exp_Objekt      t1      -- ska ersättas med nya uppdaterade poster från Deleted
                Join
                Deleted         t2
                on (t1.id=t2.id)

        insert  Exp_Objekt (Id, DBOp)   --Lägg upp nya uppdaterade poster
        select  Id, 'D'                 --från Deleted, dvs DBOp har värdet 'D'
        from    Deleted                 --på dessa poster

        Delete  Exp_Objekt              -- Ta bort existerande poster  i Exp_Objekt som i nästa sats
        From    Exp_Objekt  t1          -- ska ersättas med nya uppdaterade poster från inserted
                Join
                Inserted t2
                on (t1.id=t2.id)

        insert  Exp_Objekt (Id, DBOp)   --Lägg upp nya uppdaterade poster
        select  Id, 'U'                 --från Inserted, dvs DBOp har värdet 'U'
        from    Inserted                --på dessa poster
go

CREATE TRIGGER Trg_Tab_ObjektDel  ON [dbo].[Tab_Objekt]    --Aktiv regel för borttagna poster
FOR DELETE
AS
        Delete  Exp_Objekt              -- Ta bort existerande poster  i Exp_Objekt som ska
        From    Exp_Objekt      t1      -- ersättas med nya uppdaterade poster från Deleted
                Join
                Deleted         t2
                on (t1.id=t2.id)

        insert  Exp_Objekt (Id, DBOp)   --Lägg upp nya uppdaterade poster
        select  Id, 'D'                 --från Deleted, dvs DBOp har värdet 'D'
        from    Deleted                 --på dessa poster
go
```

5.4.1.3 DTS-paketet

DTS-paketet har som uppgift att flytta poster från vyn Vw_Objekt till tabellen Glb_Objekt och sedan ta bort de överförda posterna från Exp_Objekt. Detta kan lösas på olika sätt men den lösning som implementerats är genom en Datadriven Query, d.v.s. en datastyrd uppdatering med ett ActiveX-script. DTS-paketet ser ut enligt Figur 8 i programmet Enterprise Manager för SQL Server 2000. Komplett kod för DTS-paketet finns som appendix (se appendix 1) till rapporten, i Microsoft Visual Basic-kod.



Figur 8 DTS--Paket med datastyrd uppdatering

DTS-paketet hämtar samtliga poster ur vyn Vw_Objekt anropar sedan en lagrad procedur för att uppdatera poster i tabellen Glb_Objekt i det globala schemat. Den lagrade proceduren heter Prc_Update_Glb_Objekt. Proceduren anropas med olika värden i parametrarna beroende på hur datat i vyn Vw_Objekt ser ut, d.v.s. om det är uppdatering/insättning (U) eller borttag av rader (D) som ska utföras.

5.4.1.4 Den lagrade proceduren Prc_Update_Glb_Objekt

Denna lagrade procedur har som uppgift att förändra data i tabellen Glb_Objekt. Beroende på om det om parametern @DBOp har värdet U eller D görs antingen uppdateringar/nyinsättningar eller borttag av poster i denna tabell. Den lagrade proceduren är enkelt konstruerad för att förtydliga dess funktion och har följande utseende, i transact-SQL för SQL Server:

```
CREATE PROCEDURE Prc_Update_Glb_Objekt
        @Id             int,
        @Beskrivning    varchar(50),
        @Vikt           numeric(5,1),
        @Längd          numeric(5,1),
        @DBOp           char(1)
AS
if @DBOp='U'                                    -- Kontrollera om uppdatering/nyinsättning av post ska göras
Begin
        Update Glb_objekt                       --Uppdatera raden om den finns.
        set     Beskrivning=@Beskrivning,       -- Finns det ingen rad med detta id får denna sats ingen
                Vikt=@Vikt,                     -- effekt på datat i tabellen.
                Längd=@Längd
```

```
        where   Id=@Id

        insert Glb_Objekt (Id,Beskrivning,Vikt,Längd)       --Lägg till rad om den inte finns
        select  @Id,@Beskrivning,@Vikt,@Längd
        where   not exists ( select     *
                             from       Glb_Objekt
                             where      Id=@Id)
End
if @DBOp='D'                            -- Kontrollera om borttag av post ska göras
Begin
        Delete from Glb_Objekt
        where   Id=@Id
End
GO
```

6 Resultat

Detta kapitel jämför det utförda arbetet med examensarbetets mål. Kapitlet visar även resultatet vid användning av den implementerade metoden i fallstudien.

6.1 Jämförelse med examensarbetets mål

Det mål som sattes upp i exmensarbetet (se kapitel 3.2) och dess avgränsningar (se kapitel 3.3) har styrt arbetet till att skapa en metod som distribuerar data från de lokala databaserna till den centrala databasen genom replikering av data. Däremot har analysen av problemställningen visat på att det finns flera olika sätt att distribuera data via den centrala databasen (se kapitel 4.3). Då ett av dessa sätt har valts att utvecklas och implementeras i fallstudien finns det fler sätt kvar att utveckla. Detta för att förbättra prestanda på multidatabassystemet i sin helhet än om endast denna metod implementeras. Kraven på denna metod vid distribution av data i fallstudien (se kapitel 2.3.4) har lösts dels genom dess utformning och dels genom att använda inbyggd funktionalitet i DTS-paket. Detta medför att metoden är anpassad utifrån detta verktyg. Om metoden istället väljs att implementeras i ett annat verktyg än i DTS för SQL Server kan den komma behöva ändras. Hur metoden fungerar vid en replikering av den centrala databasen (se kapitel 4.2.2) har inte testats då fallstudien inte har en replikerad central databas. Att testa detta var dock inget mål i examensarbetet.

6.2 En jämförelse av den utvecklade metoden med Microsofts (1998) föreslagna metod

Den metod som föreslås implementeras vid datastyrda uppdateringar i DTS av Microsoft (Microsoft, 1998) är till stora delar lik den som presenteras i rapporten. Det som däremot skiljer dem åt, är t.ex.

- att ingen vy används utan istället hämtas data från en tabell där förändringar har registrerats som är gjorda i grundtabellen (se kapitel 5.4.1).
- att uppdateringar sker direkt mot en tabell, istället för att anrop görs till en lagrad procedur (se kapitel 5.4.1.4).

6.3 Resultat av implementering av metoden i fallstudien

Detta kapitel beskriver resultatet vid implementering av metoden i fallstudien. Det ger en inblick i hur metoden fungerar i praktiken och hur den bör fungera i andra liknande fall. Implementationen av metoden i fallstudien visade att den fungerade bra för distribution av data från de lokala databaserna, med SQL Server som databashanterare, till den centrala databasen. Nedan visas de fördelar och brister som upptäckts i samband med implementationen:

- Fördelar
    - Replikering av endast berörda data till central databas ger bättre prestanda. Detta eftersom att datamängder, som distribueras från de lokala databaserna, kan hållas på en liten nivå. Detta medför att bl.a. nätverkstrafik till den centrala databasen kan minskas.
    - Då replikeringen kan ske asynkront ger det möjlighet till bättre prestanda i de lokala databaserna. Om uppdateringar av data, som ska distribueras från de lokala databaserna till den centrala databasen, kan

ske utan att den centrala databasen behöver uppdateras direkt. Medför detta att prestandan kan förbättras vid förändringar av data i de lokala databaserna.

- Brister
    - Problem med initial replikering av data till den centrala databasen. Vid implementationen av metoden i fallstudien upptäcktes att en initial replikering behövdes första gången. Detta för att den centrala databasen ska bli konsistent med de lokala databaserna.

Fler fördelar och brister kan upptäckas om metoden implementeras i ytterligare fallstudier.

7 Slutsatser

Det viktigaste resultatet av detta examensarbete är metoden som utvecklats. Den visar att det är möjligt att distribuera data på det föreslagna sättet. Metoden uppfyller de krav som ursprungligen har ställts och också beskrivits i rapporten. Den har dessutom implementerats i en fallstudie för att det ska vara möjligt att se resultatet av dess användning i praktiken. I likhet med många andra utvecklingsprojekt kan antagligen även denna metod förbättras ytterligare eller förändras i efterhand. Under examensarbetets gång har det däremot framkommit att det är svårt att göra en generell metod som är bra för många olika fall, gällande distribution av data i ett multidatabassystem. Det verkar istället vara så att varje metod blir specialanpassad och optimal för just sina olika fall. Med olika fall avses här bl.a. hur stor mängd data som ska distribueras, hur dessa data relaterar till varandra och vilken arkitektur den distribuerade databasen har (se kapitel 2.2.1). Så att finna en generell metod som är optimal för samtliga nämnda fall är önskvärd fast antagligen mycket komplex.

Resultatet av jämförelsen av den utvecklade metoden som beskrivs i rapporten och den metod som föreslås av Microsoft (Microsoft, 1998) är intressant. Deras likheter ger stöd till att den utvecklade metoden lämpar sig till det valda verktyget, d.v.s. DTS.

En annan slutsats som kan dras är att när en central databas används i ett multidatabassystem så finns möjligheten att skapa lösningar liknande de som används för datalager. Detta för att den centrala databasen i vissa fall kan klassas som ett datalager (se kapitel 4.2).

7.1 Förslag till fortsatta arbeten

Förutom de brister som upptäckts vid test och implementation av den här metoden finns det även andra metoder som kan utvecklas för distribution av data från de lokala databaserna till den centrala databasen. Till detta bör det göras en analys av vilken kombination av metoder som ger bästa prestanda på ett multidatabassystem med en central databas.

Om detta görs, finns det därefter möjlighet att jämföra hur bra ett multidatabassystem, med en central databas, är i jämförelse med ett multidatabassystem, utan någon central databas.

.

8 Referenser

Elmasri Ramez, Navathe Shamkant B. (2000) *Fundamentals of Database Systems, third edition* Addison-Wesley books: U.S.A.

Bobak Angelo R. (1996) *Distributed and Multi-Database Systems* Artech House, Inc.: Norwood

Piattini Mario G., Díaz Oscar (2000) *Advanced Database Technology and Design* Artech House, Inc.: Norwood

Bell David., Grimson Jane (1992) *Distributed Database Systems* Artech House, Inc.: Norwood

Dalton Patrik, Whitehead Paul (2001) *SQL Server 2000™ Black Book* The Coriolis Group: Scottsdale

Jarke M., Lenzerini M., Yannis V., Panos V. (2000) *Fundamentals of Data Warehouses* Springer-Verlag Berlin Heidelberg 2000: Germany

Padron-McCarthy Thomas *Databaser ordlista* [online] available from: http://www.databasteknik.se/webbkursen/ordlista.html [accessed 3 juli 2006]

Knight Brian (2001) *Configuration of StarSQL 3.0* [online] available from: http://www.databasejournal.com/img/BK_starsql_code.html [accessed 6 augusti 2006]

Microsoft Corporation *SQL Server Books Online* [cdrom] available from: http://www.microsoft.com/ [accessed 9 augusti 2006]

Appendix 1 - DTS-paketets konstruktion

DTS-paktets konstruktion visas i form av källkod (Microsoft Visual Basic).

```
'************************************************************
'Microsoft SQL Server 2000
'Visual Basic file generated for DTS Package
'File Name: Överföring Vw_Objekt.bas
'Package Name: Överföring Vw_Objekt
'Package Description:
'Generated Date: 2006-08-17
'Generated Time: 08:46:46
'************************************************************

Option Explicit
Public goPackageOld As New DTS.Package
Public goPackage As DTS.Package2
Private Sub Main()
        set goPackage = goPackageOld

        goPackage.Name = "Överföring Vw_Objekt"
        goPackage.WriteCompletionStatusToNTEventLog = False
        goPackage.FailOnError = False
        goPackage.PackagePriorityClass = 2
        goPackage.MaxConcurrentSteps = 4
        goPackage.LineageOptions = 0
        goPackage.UseTransaction = True
        goPackage.TransactionIsolationLevel = 4096
        goPackage.AutoCommitTransaction = True
        goPackage.RepositoryMetadataOptions = 0
        goPackage.UseOLEDBServiceComponents = True
        goPackage.LogToSQLServer = False
        goPackage.LogServerFlags = 0
        goPackage.FailPackageOnLogFailure = False
        goPackage.ExplicitGlobalVariables = False
        goPackage.PackageType = 0

'---------------------------------------------------------------------------
' create package connection information
'---------------------------------------------------------------------------

Dim oConnection as DTS.Connection2
'------------- a new connection defined below.
'For security purposes, the password is never scripted

Set oConnection = goPackage.Connections.New("SQLOLEDB")
```

```
        oConnection.ConnectionProperties("Persist Security Info") = True
        oConnection.ConnectionProperties("User ID") = "sa"
        oConnection.ConnectionProperties("Initial Catalog") = "LokalDB"
        oConnection.ConnectionProperties("Data Source") = "zionlt"
        oConnection.ConnectionProperties("Locale Identifier") = 1053
        oConnection.ConnectionProperties("Prompt") = 4
        oConnection.ConnectionProperties("General Timeout") = 0
        oConnection.ConnectionProperties("Use Procedure for Prepare") = 1
        oConnection.ConnectionProperties("Auto Translate") = True
        oConnection.ConnectionProperties("Packet Size") = 4096
        oConnection.ConnectionProperties("Workstation ID") = "ZIONLT"
        oConnection.ConnectionProperties("Use Encryption for Data") = False
        oConnection.ConnectionProperties("Tag with column collation when possible") = False

        oConnection.Name = "Lokal DB"
        oConnection.ID = 1
        oConnection.Reusable = True
        oConnection.ConnectImmediate = False
        oConnection.DataSource = "zionlt"
        oConnection.UserID = "sa"
        oConnection.ConnectionTimeout = 0
        oConnection.Catalog = "LokalDB"
        oConnection.UseTrustedConnection = False
        oConnection.UseDSL = True

        'If you have a password for this connection, please uncomment and add your password below.
        'oConnection.Password = "<put the password here>"

goPackage.Connections.Add oConnection
Set oConnection = Nothing
'------------- a new connection defined below.
'For security purposes, the password is never scripted

Set oConnection = goPackage.Connections.New("SQLOLEDB")

        oConnection.ConnectionProperties("Persist Security Info") = True
        oConnection.ConnectionProperties("User ID") = "sa"
        oConnection.ConnectionProperties("Initial Catalog") = "CentralDB"
        oConnection.ConnectionProperties("Data Source") = "zionlt"
        oConnection.ConnectionProperties("Locale Identifier") = 1053
        oConnection.ConnectionProperties("Prompt") = 4
        oConnection.ConnectionProperties("General Timeout") = 0
        oConnection.ConnectionProperties("Use Procedure for Prepare") = 1
        oConnection.ConnectionProperties("Auto Translate") = True
        oConnection.ConnectionProperties("Packet Size") = 4096
        oConnection.ConnectionProperties("Workstation ID") = "ZIONLT"
        oConnection.ConnectionProperties("Use Encryption for Data") = False
        oConnection.ConnectionProperties("Tag with column collation when possible") = False
```

```
        oConnection.Name = "Central DB"
        oConnection.ID = 2
        oConnection.Reusable = True
        oConnection.ConnectImmediate = False
        oConnection.DataSource = "zionlt"
        oConnection.UserID = "sa"
        oConnection.ConnectionTimeout = 0
        oConnection.Catalog = "CentralDB"
        oConnection.UseTrustedConnection = False
        oConnection.UseDSL = True

        'If you have a password for this connection, please uncomment and add your password below.
        'oConnection.Password = "<put the password here>"

goPackage.Connections.Add oConnection
Set oConnection = Nothing
'------------------------------------------------------------------------
' create package steps information
'------------------------------------------------------------------------

Dim oStep as DTS.Step2
Dim oPrecConstraint as DTS.PrecedenceConstraint

'------------- a new step defined below

Set oStep = goPackage.Steps.New

        oStep.Name = "DTSStep_DTSDataDrivenQueryTask_1"
        oStep.Description = "Vw_Objekt"
        oStep.ExecutionStatus = 1
        oStep.TaskName = "DTSTask_DTSDataDrivenQueryTask_1"
        oStep.CommitSuccess = True
        oStep.RollbackFailure = True
        oStep.ScriptLanguage = "VBScript"
        oStep.AddGlobalVariables = True
        oStep.RelativePriority = 3
        oStep.CloseConnection = False
        oStep.ExecuteInMainThread = False
        oStep.IsPackageDSORowset = False
        oStep.JoinTransactionIfPresent = True
        oStep.DisableStep = False
        oStep.FailPackageOnError = False

goPackage.Steps.Add oStep
Set oStep = Nothing

'------------------------------------------------------------------------
' create package tasks information
'------------------------------------------------------------------------
```

```
'------------- call Task_Sub1 for task DTSTask_DTSDataDrivenQueryTask_1 (Vw_Objekt)
Call Task_Sub1( goPackage    )
'----------------------------------------------------------------------------
' Save or execute package
'----------------------------------------------------------------------------

'goPackage.SaveToSQLServer "(local)", "sa", ""
goPackage.Execute
goPackage.Uninitialize
'to save a package instead of executing it, comment out the executing package line above and uncomment the saving package line
set goPackage = Nothing

set goPackageOld = Nothing

End Sub

'------------- define Task_Sub1 for task DTSTask_DTSDataDrivenQueryTask_1 (Vw_Objekt)
Public Sub Task_Sub1(ByVal goPackage As Object)

Dim oTask As DTS.Task
Dim oLookup As DTS.Lookup

Dim oCustomTask1 As DTS.DataDrivenQueryTask2
Set oTask = goPackage.Tasks.New("DTSDataDrivenQueryTask")
Set oCustomTask1 = oTask.CustomTask

        oCustomTask1.Name = "DTSTask_DTSDataDrivenQueryTask_1"
        oCustomTask1.Description = "Vw_Objekt"
        oCustomTask1.SourceConnectionID = 1
        oCustomTask1.SourceSQLStatement = "SELECT     *" & vbCrLf
        oCustomTask1.SourceSQLStatement = oCustomTask1.SourceSQLStatement & "FROM         Vw_Objekt" & vbCrLf
        oCustomTask1.SourceSQLStatement = oCustomTask1.SourceSQLStatement & "Delete from Exp_Objekt"
        oCustomTask1.DestinationConnectionID = 2
        oCustomTask1.DestinationObjectName = """CentralDB"".""dbo"".""Glb_Objekt"""
        oCustomTask1.ProgressRowCount = 1000
        oCustomTask1.MaximumErrorCount = 0
        oCustomTask1.FetchBufferSize = 1
        oCustomTask1.InsertQuery = "Prc_Update_Glb_Objekt ?,?,?,?,'U'"
        oCustomTask1.DeleteQuery = "Prc_Update_Glb_Objekt ?,?,?,?,'D'"
        oCustomTask1.ExceptionFileColumnDelimiter = "|"
        oCustomTask1.ExceptionFileRowDelimiter = vbCrLf
        oCustomTask1.FirstRow = "0"
        oCustomTask1.LastRow = "0"
        oCustomTask1.ExceptionFileOptions = 1

Call oCustomTask1_Trans_Sub1( oCustomTask1  )
```

```
goPackage.Tasks.Add oTask
Set oCustomTask1 = Nothing
Set oTask = Nothing

End Sub

Public Sub oCustomTask1_Trans_Sub1(ByVal oCustomTask1 As Object)

        Dim oTransformation As DTS.Transformation2
        Dim oTransProps as DTS.Properties
        Dim oColumn As DTS.Column
        Set oTransformation = oCustomTask1.Transformations.New("DTSPump.DataPumpTransformScript")
                oTransformation.Name = "DTSTransformation__1"
                oTransformation.TransformFlags = 63
                oTransformation.ForceSourceBlobsBuffered = 0
                oTransformation.ForceBlobsInMemory = False
                oTransformation.InMemoryBlobSize = 1048576
                oTransformation.TransformPhases = 4

                Set oColumn = oTransformation.SourceColumns.New("DBOp" , 1)
                        oColumn.Name = "DBOp"
                        oColumn.Ordinal = 1
                        oColumn.Flags = 120
                        oColumn.Size = 1
                        oColumn.DataType = 129
                        oColumn.Precision = 0
                        oColumn.NumericScale = 0
                        oColumn.Nullable = True

                oTransformation.SourceColumns.Add oColumn
                Set oColumn = Nothing

                Set oColumn = oTransformation.SourceColumns.New("Id" , 2)
                        oColumn.Name = "Id"
                        oColumn.Ordinal = 2
                        oColumn.Flags = 32792
                        oColumn.Size = 0
                        oColumn.DataType = 3
                        oColumn.Precision = 0
                        oColumn.NumericScale = 0
                        oColumn.Nullable = False

                oTransformation.SourceColumns.Add oColumn
                Set oColumn = Nothing

                Set oColumn = oTransformation.SourceColumns.New("Beskrivning" , 3)
                        oColumn.Name = "Beskrivning"
                        oColumn.Ordinal = 3
                        oColumn.Flags = 104
```

```
        oColumn.Size = 50
        oColumn.DataType = 129
        oColumn.Precision = 0
        oColumn.NumericScale = 0
        oColumn.Nullable = True

oTransformation.SourceColumns.Add oColumn
Set oColumn = Nothing

Set oColumn = oTransformation.SourceColumns.New("vikt" , 4)
        oColumn.Name = "vikt"
        oColumn.Ordinal = 4
        oColumn.Flags = 120
        oColumn.Size = 0
        oColumn.DataType = 131
        oColumn.Precision = 5
        oColumn.NumericScale = 1
        oColumn.Nullable = True

oTransformation.SourceColumns.Add oColumn
Set oColumn = Nothing

Set oColumn = oTransformation.SourceColumns.New("Längd" , 5)
        oColumn.Name = "Längd"
        oColumn.Ordinal = 5
        oColumn.Flags = 120
        oColumn.Size = 0
        oColumn.DataType = 131
        oColumn.Precision = 5
        oColumn.NumericScale = 1
        oColumn.Nullable = True

oTransformation.SourceColumns.Add oColumn
Set oColumn = Nothing

Set oColumn = oTransformation.DestinationColumns.New("Id" , 1)
        oColumn.Name = "Id"
        oColumn.Ordinal = 1
        oColumn.Flags = 24
        oColumn.Size = 0
        oColumn.DataType = 3
        oColumn.Precision = 0
        oColumn.NumericScale = 0
        oColumn.Nullable = False

oTransformation.DestinationColumns.Add oColumn
Set oColumn = Nothing

Set oColumn = oTransformation.DestinationColumns.New("Beskrivning" , 2)
```

```
        oColumn.Name = "Beskrivning"
        oColumn.Ordinal = 2
        oColumn.Flags = 104
        oColumn.Size = 50
        oColumn.DataType = 129
        oColumn.Precision = 0
        oColumn.NumericScale = 0
        oColumn.Nullable = True

oTransformation.DestinationColumns.Add oColumn
Set oColumn = Nothing

Set oColumn = oTransformation.DestinationColumns.New("Vikt" , 3)
        oColumn.Name = "Vikt"
        oColumn.Ordinal = 3
        oColumn.Flags = 120
        oColumn.Size = 0
        oColumn.DataType = 131
        oColumn.Precision = 5
        oColumn.NumericScale = 1
        oColumn.Nullable = True

oTransformation.DestinationColumns.Add oColumn
Set oColumn = Nothing

Set oColumn = oTransformation.DestinationColumns.New("Längd" , 4)
        oColumn.Name = "Längd"
        oColumn.Ordinal = 4
        oColumn.Flags = 120
        oColumn.Size = 0
        oColumn.DataType = 131
        oColumn.Precision = 5
        oColumn.NumericScale = 1
        oColumn.Nullable = True

oTransformation.DestinationColumns.Add oColumn
Set oColumn = Nothing

Set oColumn = oCustomTask1.DeleteQueryColumns.New("Id" , 1)
        oColumn.Name = "Id"
        oColumn.Ordinal = 1
        oColumn.Flags = 0
        oColumn.Size = 0
        oColumn.DataType = 0
        oColumn.Precision = 0
        oColumn.NumericScale = 0
        oColumn.Nullable = True

oCustomTask1.DeleteQueryColumns.Add oColumn
```

```
Set oColumn = Nothing

Set oColumn = oCustomTask1.DeleteQueryColumns.New("Beskrivning" , 2)
        oColumn.Name = "Beskrivning"
        oColumn.Ordinal = 2
        oColumn.Flags = 0
        oColumn.Size = 0
        oColumn.DataType = 0
        oColumn.Precision = 0
        oColumn.NumericScale = 0
        oColumn.Nullable = True

oCustomTask1.DeleteQueryColumns.Add oColumn
Set oColumn = Nothing

Set oColumn = oCustomTask1.DeleteQueryColumns.New("Vikt" , 3)
        oColumn.Name = "Vikt"
        oColumn.Ordinal = 3
        oColumn.Flags = 0
        oColumn.Size = 0
        oColumn.DataType = 0
        oColumn.Precision = 0
        oColumn.NumericScale = 0
        oColumn.Nullable = True

oCustomTask1.DeleteQueryColumns.Add oColumn
Set oColumn = Nothing

Set oColumn = oCustomTask1.DeleteQueryColumns.New("Längd" , 4)
        oColumn.Name = "Längd"
        oColumn.Ordinal = 4
        oColumn.Flags = 0
        oColumn.Size = 0
        oColumn.DataType = 0
        oColumn.Precision = 0
        oColumn.NumericScale = 0
        oColumn.Nullable = True

oCustomTask1.DeleteQueryColumns.Add oColumn
Set oColumn = Nothing

Set oColumn = oCustomTask1.InsertQueryColumns.New("Id" , 1)
        oColumn.Name = "Id"
        oColumn.Ordinal = 1
        oColumn.Flags = 0
        oColumn.Size = 0
        oColumn.DataType = 0
        oColumn.Precision = 0
        oColumn.NumericScale = 0
```

```
		oColumn.Nullable = True

	oCustomTask1.InsertQueryColumns.Add oColumn
	Set oColumn = Nothing

	Set oColumn = oCustomTask1.InsertQueryColumns.New("Beskrivning" , 2)
		oColumn.Name = "Beskrivning"
		oColumn.Ordinal = 2
		oColumn.Flags = 0
		oColumn.Size = 0
		oColumn.DataType = 0
		oColumn.Precision = 0
		oColumn.NumericScale = 0
		oColumn.Nullable = True

	oCustomTask1.InsertQueryColumns.Add oColumn
	Set oColumn = Nothing

	Set oColumn = oCustomTask1.InsertQueryColumns.New("Vikt" , 3)
		oColumn.Name = "Vikt"
		oColumn.Ordinal = 3
		oColumn.Flags = 0
		oColumn.Size = 0
		oColumn.DataType = 0
		oColumn.Precision = 0
		oColumn.NumericScale = 0
		oColumn.Nullable = True

	oCustomTask1.InsertQueryColumns.Add oColumn
	Set oColumn = Nothing

	Set oColumn = oCustomTask1.InsertQueryColumns.New("Längd" , 4)
		oColumn.Name = "Längd"
		oColumn.Ordinal = 4
		oColumn.Flags = 0
		oColumn.Size = 0
		oColumn.DataType = 0
		oColumn.Precision = 0
		oColumn.NumericScale = 0
		oColumn.Nullable = True

	oCustomTask1.InsertQueryColumns.Add oColumn
	Set oColumn = Nothing

Set oTransProps = oTransformation.TransformServerProperties

	oTransProps("Text") = "'************************************************************************" & vbCrLf
	oTransProps("Text") = oTransProps("Text") & "'  Visual Basic Transformation Script" & vbCrLf
	oTransProps("Text") = oTransProps("Text") & "'************************************************************************" & vbCrLf
```

```
            oTransProps("Text") = oTransProps("Text") & "'  Copy each source column to the destination column" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "Function Main()" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "       DTSDestination(""Id"") = DTSSource(""Id"")" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "       DTSDestination(""Beskrivning"") = DTSSource(""Beskrivning"")" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "       DTSDestination(""Vikt"") = DTSSource(""vikt"")" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "       DTSDestination(""Längd"") = DTSSource(""Längd"")" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "       select case DTSSource(""DBOp"")" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "               case ""U""" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "                       Main = DTSTransformStat_InsertQuery" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "               case ""D""" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "                       Main = DTSTransformStat_DeleteQuery" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "               case else" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "                       Main=DTSTransformStat_SkipRow" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "       end select" & vbCrLf
            oTransProps("Text") = oTransProps("Text") & "End Function"
            oTransProps("Language") = "VBScript"
            oTransProps("FunctionEntry") = "Main"

        Set oTransProps = Nothing

        oCustomTask1.Transformations.Add oTransformation
        Set oTransformation = Nothing

End Sub
```