CHALMERS

Modellera och implementera ett Business Intelligence system för analys av ekonomiska data

Modelling and implementing a Business Intelligence system for analysis of economic data

Master of Science Thesis in Software Engineering

JOHAN GUSTAVSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, November 2013

Modellera och implementera ett Business Intelligence system för analys av ekonomiska data

Johan Gustavsson

Examiner: Sven-Arne Andreasson

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden 2013

Sammanfattning

Syftet med denna rapport är att visa hur ett Business Intelligence system, för att analysera ekonomiska data, kan modelleras och implementeras. Rapportens problem utgår ifrån rapportens studerade företag, som är i behov att analysera sitt ekonomiska resultat baserat på konsult, kostnadsställe och bolagsnivå. Analysen av BI-systemets omgivande miljö visar att företaget inte självt förfogar över sina ekonomiska data utan dessa måste tillhandahållas av leverantören till det affärssystem bolaget använder sig av. Vidare begränsas datautbytet till ett antal XML-strömmar över https-protokollet, då leverantören tillhandahåller data i detta format. Av rapporten framgår slutligen att en stagedatabas, produktionsdatabas och datawarehouse implementerats för möjligöra de ekonomiska analyserna. Vidare har dataladdningsprocessen implementerats med hjälp av ett ETL verktyg och de nämnda XML-strömmarna över https-protokollet.

Abstract

The purpose with this report is to show how a Business Intelligence system can be modeled and implemented to support economic analysis. The starting point for the problem of the report is the studied company of the report, which needs to analyze its economical result based on consultant, department and company level. Analysis of the environment external to the Business Intelligence system shows that the studied company of the report does not govern its own data. Instead, the supplier of the ERP-system must supply these. The data exchange is also restricted to a number of XML-streams over the https-protocoll, since the supplier deliveries data in this format. The report shows that a stage database, production database and data warehouse were implemented in order to enable the economic analyses. Furthermore, the data loading process was implemented by means of an ETL tool and the mentioned XML-streams over https-protocoll.

Förord

Efter en lång process, som förlängts pågrund av några års arbete som verksam konsult, av läsande, skrivande, kodande, kravsammanställning och design är arbetet med detta examensarbete slut. Det har varit en lärorik och intressant tid där författaren skaffat sig flera nya insikter kring mjukvaruutveckling och hur den kan ge stöd åt utvecklingen av ett Business Intelligence system.

Det finns många personer som bidragit till att denna rapport ser dagens ljus. Författaren vill först börja med att tack Sven-Arne Andreasson, som bidragit med viktig och gedigen kunskap under arbetet med rapporten. Utan hans feedback och kunskap hade inte denna rapport varit möjlig. Författaren vill också tacka alla medarbetare på Know IT i Göteborg AB, som engagerat sig och alltid varit vänliga och tagit sig tid att ge viktiga bidrag till arbetet med denna rapport. Framförallt vill författaren tacka Per Wallentin, Fredrik Abrahamsson och Annika Karlund för deras värdefulla deltagande i och kring de aktiviteter som legat till grund för rapportens innehåll. Författaren vill också rikta ett stort tack till Marcus Andersson, som delat med sig av sin erfarenhet och sitt kunnande och därmed hjälpt till att föra arbetet med denna rapport i hamn.

Johan Gustavsson

Innehållsförteckning

Figurföreteckning

1 Inledning

I vardagen ställs människor ständigt inför situationer där de måste fatta olika beslut. Vissa utav dessa är triviala och ställer små krav på den kunskap som måste besittas för att genomtänkta och rationella beslut ska kunna tas, medan andra beslut rör frågor som till sin natur är komplexa och därmed ställer stora krav på kunskap hos beslutsfattaren. Detta tillsammans med att vi idag lever i ett informationssamhälle där vi ständigt mottar stora mängder information gör det svårt att överblicka och selektera den information som är relevant för olika beslut.

För att underlätta denna situation har olika teknologier tagits fram. Dessa underlättar beslutsprocesser genom att bland annat aggregera, selektera och visualisera sådan information beslutsfattaren anser vara viktig för att han eller hon skall kunna fatta underbyggda beslut. En kategori av applikationer och teknologier som kommit att vinna ett allt större intresse bland företag är Business Intelligence (BI) system. Denna samling av applikationer och teknologier, som ingår i konceptet Business Intelligence, ger beslutsstöd utifrån de data som ett företag har tillgång till genom exempelvis sitt affärssystem. Behovet av Business Intelligence har vuxit fram på grund av att företag i allt större utsträckning behöver få tillgång till tillförlitliga data som återspeglar dess ställning i realtid. Problemet har varit och är fortfarande att man inom en organisation har stora mängder data samlat i olika dokument, databaser och andra typer av datahållare, men man har inget bra sätt att göra alla dessa data tillgängliga för användarna.

Ett affärssystem utgör ett illustrativt exempel på de problem som visualiseringen av data utgör för användarna i en organisation. I detta system finns stora mängder finansiella data samlat, vilka är resultatet av input från olika affärsprocesser som är definierade och utförs i ett företag. Utifrån dessa kan sedan fakturering, beräkning av resultat och andra aktiviteter som är kritiska för ett företags verksamhet utföras. Problemet är dock att många finansiella tal inte beräknas utav affärssystemet utan att en användare själv måste beräkna dessa utifrån de data som finns samlade i systemet. Detta görs vanligtvis genom sammanställningar i någon form av kalkylark eller liknande beräkningsformulär. Arbetet är ofta mycket tidskrävande och ställer krav på vissa ekonomiska kunskaper hos den person som utför beräkningarna. Ett annat problem som detta förfarande medför är att man som beslutsfattare i en organisation inte får tillgång till ”färsk data”, det vill säga sådana data som beskriver verkligheten i realtid. Istället tenderar beräknade finansiella tal att beskriva historiska händelser, sådana händelser som exempelvis redan har skett den senaste månaden eller det senaste året, och vars värde inte är speciellt stort då en beslutsfattare behöver ha tillgång till data som ger en ögonblicksbild av verkligheten som den förhåller sig för tillfället.

Arbetet som ligger till grund för denna rapport utfördes våren 2007 vid företaget Know IT i Göteborg AB, vilket omfattade att utveckla en Business Intelligence lösning för analys av ekonomiska data. Några utav de funktionella krav som företaget ställde på lösningen var att systemet skulle kunna:

- Beräkna resultat per vecka, månad och år för respektive kostnadsställe inom företaget.

- Visualisera beräknade värden på en intranetportal för företagets anställda.

1.1 Förekommande begrepp och definitioner

Entitetstabell
Databastabell som inte innehåller främmande nycklar relaterade till andra databastabeller.

BI (Business intelligence)
BI är en samlingsterm för applikationer som hjälper till att visualisera och analysera affärsdata.

Beslutsfattare
En person som i sitt dagliga arbete behöver fatta beslut av något slag, vilka ofta baseras på olika typer av data. Exempel på sådana personer är Vd, mellanchefer och konsulter.

Datawarehouse
Central datalagringsplats där data ifrån flera olika verksamhetssystem sammanförs för att möjliggöra olika typer av analyser. Datawarehouset är ofta den centrala databasen i en BI-lösning.

Dimensionstabell
Databastabell som ingår i ett datawarehouse och innehåller information om de data som skall analyseras. En dimensionstabell kan exempelvis beskriva en kund, en anställd eller en avdelning.

ETL (Extract Transact Load) - verktyg
Applikation specialiserad på att extrahera data ifrån ett källsystem och föra över dessa till ett målsystem. Ett ETL-verktyg är framförallt specialiserat på snabba dataöverföringar och transformering av källsystemets dataformat till målsystemets dataformat.

Faktatabell
Databastabell som ingår i ett datawarehouse där de faktiska data som skall analyseras registreras. Exempel på data som registreras i en faktatabell kan exempelvis vara summan av en intäkt eller kostnad.

Normalisering
Tabellerna konstrueras så att dubbellagring av data undviks i en databas.

OLAP (Online Analysis Processing) kub
OLAP (Online Analysis Processing) är en viss typ av applikation som analyserar de data som finns sparad i ett datawarehouse.

Relationstabell
Databastabell som innehåller data i form av primärnycklar ifrån andra databastabeller.

Stagearea
En area, exempelvis en fil eller databas, i vilken data mellanlagars under en dataöverföringsprocess.

Snow-flake schema
Ett snow-flake schema är ett normaliserat star-schema på ett sådant vis att dimensionerna är normaliserade (dimensionerna är separerade i flera tabeller).

Star-Schema
En formation av en faktatabell och en eller flera dimensionstabeller som ofta återfinns i ett datawarehouse. Formationen liknar en stjärna vilket gett den dess namn.

1.2 Syfte

Syftet med detta examensarbete är att modellera och implementera ett BI system för analys av ekonomiska data.

1.3 Avgränsningar

Det föreliggande examensarbete begränsas till:

- Att beskriva den del av BI-lösningen som är hänförlig till datalagret, det vill säga den area där data sparas.

- Att beskriva den del av BI-lösningen som är hänförlig till extrahering, transformering och laddning av data mellan olika datakällor.

1.4 Problem och problemformulering

I takt med att antalet anställda på Know IT i Göteborg AB kommit att öka har arbetet med att överblicka den ekonomiska ställningen i företaget försvårats allt mer. I nuläget ansvarar den ekonomiansvarige i företaget för att de ekonomiska nyckeltal, såsom resultat, intäkter, kostnader etcetera, beräknas och kommer de beslutsfattare i företaget till handa som behöver dessa data för att kunna fatta olika beslut. Beräkningarna görs månadsvis och sker till stor del med manuella beräkningar utifrån de data som finns registrerad i det affärssystem som Know IT i Göteborg AB använder sig av. Denna process är mycket tidsödande, vilket får till följd att man enbart kan beräkna de ekonomiska nyckeltal som man inom företaget anser vara allra viktigast. Dock skulle man vilja kunna göra en än mer omfattande analys av de data som finns registrerat i affärssystemet. Då denna typ av analyser inte erbjuds av systemet vill man inom Know IT i Göteborg AB implementera ett Business Intelligence system där man kan läsa in data ifrån affärssystemet för att månadsvis analysera dessa i BI-systemet.

1.5 Uppgiften

Utifrån den ovan givna problemformuleringen innebär uppgiften som ligger till grund för detta examensarbete att ett BI system behöver konstrueras för analys av ekonomiska data. Konkret innebär detta att systemet behöver kommunicera med det affärssystem som Know IT i Göteborg AB använder sig av för att på så sätt få tillgång till de data som skall analyseras.

1.6 Beskrivning av Know IT i Göteborg AB

Know IT i Göteborg AB ingår som ett utav totalt 31 dotterbolag i Know IT-koncernen, vars huvudkontor är beläget i Stockholm. Know IT i Göteborg AB har sitt säte i Göteborg och erbjuder konsulttjänster inom områdena Enterprise Content Managament, Test & Quality Management samt Systemutveckling. Bland några utav bolagets kunder återfinns Västra Götalandsregionen, Västtrafik, Santa Maria och Svenska kyrkan (Know IT i Göteborg AB, 2010).

2 Vetenskaplig bakgrund

I det föreliggande kapitlet redogörs för examensarbetets vetenskapliga bakgrund. Områden som behandlas inom ramarna för denna framställning är Business Intelligence (BI), Extract, Transcat och Load (ETL) verktyg, Datawarehouse (DW) samt OLAP (Online Analysis Procesings) kuber.

2.1 Business Intelligence (BI)

Business Intelligence är ett samlingsnamn för en rad olika teknologier och applikationer som kan kombineras för att tillsammans hjälpa till att utforska data, relationer mellan data och olika trender. Det övergripande syftet med dessa teknologier är att förse beslutsfattare inom en organisation med information som är vital för att beslut skall kunna fattas på rationella grunder och inte utifrån mer eller mindre professionella gissningar. De värden som Business Intelligence kan erbjuda företag och organisationer är bland annat högre vinster och lägre kostnader samt bättre kontroll/styrning av affärsprocesser och resurser (Raisinghani, 2004).

Det finns idag ingen allmänt vedertagen definition för vad BI innebär. Mundey med flera (2006) ger dock egen definition av vad BI är som kan vara till hjälp för att illustrera vad BI och BI applikationer kan omfatta:

”BI applications are the delivery vehicle of business intelligence – the reports and analyses that provide usable information to the business users. BI applications provide a broad spectrum of reports and analyses, ranging from very simple fixed format reports to sophisticated analytic applications with complex embedded algorithms and domain expertise.”(Mundey m. fl, 2006)

Av denna definition framgår det att BI applikationer tillhandahåller rapporter och analyser som är användbara för brukarna av systemen. För att kunna presentera dessa rapporter och analyser krävs det dock att de produceras och sammanställs på något vis. I ett BI system finns det därför olika delar som har skilda typer av ansvar. I det följande kommer några vanligt förekommande delar i ett system av denna typ att presenteras, det vill säga ETL verktyg, datawarehouse samt OLAP kuber.

2.1.1 Extract, Transact and Load (ETL)

Ett vanligt förekommande verktyg för att extrahera, transformera och ladda data ifrån olika datakällor är ett så kallat ETL verktyg. Dessa verktyg är ofta en del av en BI lösning då de kan liknas vid en ”datapump”, som hämtar data ifrån ett system och uppdaterar ett annat system med dessa. Detta kan verka som en enkel process, men den kan försvåras av att grundsystemet, varifrån data hämtas, och målsystemet inte representerar information på samma sätt. En annan svårighet kan vara att data hämtas från en ostrukturerad källa, exempelvis ett kalkylark, medan denna skall läggas till i en strukturerad källa, som en databas. Dessa omständigheter ställer krav på att data representerad på en form kan

transformeras till en annan form, vilket ett ETL verktyg är specialiserat på. Genom att tillhandahålla komponenter för datatransformeringar av olika slag kan dataflöden konstrueras där eftersökt data först hämtas från ett grundsystem och transformeras till rätt form för att sedan läggas till i ett målsystem. Det som gör ETL verktygen speciellt lämpade för denna typ av problem är att de är optimerade för att hantera transformeringarna i minnet, vilket reducerar antalet I/O-operationer. Antalet I/O-operationer kan ha stor påverkan på systemets prestanda då det rör sig om stora datamängder, vilket ofta är fallet i en BI-lösning. (Kimball & Caserta, 2004)

2.1.2 Stagingarea

En stagingarea är vanligt förekommande i systemlösningar som involverar ett datawarehosue och ett ETL verktyg. Enligt Kimball & Caserta (2004) betyder staging i detta sammanhang att viss data skrivs till hårddisken under ETL-processens genomförande. Det kan finnas många anledningar till att detta sker, exempelvis kan data behöva sorteras, aggregeras och beräkningar sparas som behövs i senare steg av processen. En stagingarea kan också användas för att data behöver sparas ur ren säkerhetssynpunkt. (Kimball & Caserta, 2004)

Ett sätt att konstruera en stagingarea är som en databas, med tillhörande tabeller, där data kan sparas innan de laddas in i datawarehouset. En annan lösning är att bygga stagingareran med hjälp av vanliga filer för att uppnå motsvarande funktionalitet. Fördelen med att bygga stagingarean med hjälp av en databas är att denna tillhandahåller metadata per automatik medan sådana data måste definieras explicit då en stagingarea konstrueras med hjälp av filer. Nackdelen att konstruera arean som en databas är dock att prestanda blir lägre än då filsystemet används. Det beror på den extra overhead som tillkommer när databasen bland annat skall hantera insättning samt läsning av data. (Kimball & Caserta, 2004)

2.1.3 Datawarehouse (DW)

I de flesta av alla BI lösningar finns det någon slags central förvaringsplats för data. Utifrån dessa kan sedan de resultat som skall presenteras för användarna beräknas. Namnet på denna förvaringsplats är datawarehouse, vilket antyder dess roll i ett BI-system. Konceptuellt sett är ett datawaehouse en vanlig relationsdatabas, som är uppbyggd av tabeller innehållande kolumner och rader. Det finns dock en rad olika skillnader mellan ett datawarehouse och en vanlig transaktionsdatabas (OLTP databas). Skillnaderna grundar sig i de olika krav som ställs vid användandet av databaserna och beror främst på i vilket syfte de är tänkta att användas. En vanlig transaktionsdatabas ingår ofta i system där användaren behöver information om statusen i systemet såsom den är vid ett visst ögonblick i tiden. Ett datawarehouse å andra sidan syftar främst till att hantera stora datamängder och ge en historisk bild av data. Därmed skiljer sig användarnas krav åt beträffande den funktionalitet som de olika databaserna förväntas tillhandahålla.

Ytterligare skillnader föreligger mellan en transaktionsdatabas och ett datawarehouse. Då det finns många användare eller processer som samtidigt gör förfrågningar mot en transaktionsdatabas är det ofta ett krav att den är bra på att hantera en hög grad av samtidiga användare, det vill säga att många förfrågningar av data kan göras samtidigt och ske snabbt. I transaktionsdatabaser tillämpas oftast en hög grad av normalisering för att tillgodose dessa krav. Detta är dock oftast inte fallet i ett datawarehouse där antalet användare oftast är få (används oftast av ett fåtal personer i en organisation, exempelvis chefer), varvid en hög grad av samtidiga användare inte måste hanteras. En låg grad av normalisering leder också till att antalet JOIN operationer som måste göras mellan olika tabeller i ett datawarehouse är få relativt en transaktionsdatabas där normaliseringen är hög. Denna egenskap är viktig i ett datawarehouse där datamängden kan vara väldigt stor och röra sig om miljontals rader. I fallet med en hög grad av normalisering och en i databassammanhang dyr operation som JOIN skulle detta kunna få till följd att en databasfråga skulle kunna ta flera timmar, dagar eller veckor att slutföra. (Powell, 2006)

I det följande redogörs för två olika typer av databasdesign vilka ofta förespråkas i ett datawarehouse, nämligen star-schema och snow-flakeschema. Dessa scheman beskriver hur tabeller kan designas och knytas till varandra genom relationer.

2.1.3.1 Star-schema

Tabeller i ett datawarehouse brukar antingen benämnas som dimensionstabeller eller faktatabeller. Dimensionstabeller innehåller ofta data som sällan förändras, exempelvis information om kunder, produkter etcetera. Denna information är av beskrivande natur, det vill säga den innehåller exempelvis en kunds namn, adress etcetera. En faktatabell däremot innehåller ofta någon sorts fakta, exempelvis kvantiteten av sålda produkter, men också foreign keys från dimensionstabellerna. På så sätt binds dimensionstabeller och faktatabeller samman.

I ett star-schema ingår en faktatabell och en eller flera dimensionstabeller, vilka är sammankopplade som en en-till-många (one-to-many) relation. Denna går från en dimensionstabell till faktatabellen, se Figur 1. Namnet star-schema kommer av faktatabellens och dimensiontabellernas förhållande till varandra. Tillsammans bildar ett schema eller en figur, som kan liknas vid utseendet av en stjärna. Av Figur 1 framgår ett exempel på hur ett star-schema kan se ut (Powell, 2006)

Figur 1 – Schematisk bild av ett star-schema (Powell, 2006)**.**

Star-schemat är det mest effektiva databasschemat vid användande av faktatabeller och dimensionstabeller. Det kommer sig av att antalet JOIN-operationer mellan faktatabellen och dimensionstabellerna blir få då endast en JOIN-operation behöver utföras per dimensionstabell. Med ett star-schema kan beteendet i faktatabellerna karaktäriseras som dynamiskt, där nya data ofta tillkommer, medan beteendet i dimensiontabellerna är av mer statiskt karaktär, vilket innebär att ny data sällan tillkommer. På så sätt kommer storleken på tabellerna variera stort där faktatabellerna kan innehålla miljontals rader medan dimensionstabellerna endast innehåller några hundra. (Powell, 2006)

2.1.3.2 Snowflake-schema

Powell (2006) beskriver ett snowflake-schema på följande vis:

A snowflake schema is a normalized star schema, such that dimension entities are normalized (dimensions are separated into multiple tables) (Powell, 2006)

Ett snowflakeschema är alltså ett normaliserat starschema. Avsikten med denna design av ett datawarehouse är att reducera eventuell duplicerad information, som kan förekomma i dimensionstabellerna. Det leder till att vissa av de dimensionstabeller som är relaterade direkt till en faktatabell, då ett star-schema används, istället kommer att relateras till andra dimensionstabeller. I Figur 2 visas ett snowflake-schema där vissa dimensionstabeller är kopplade till andra dimensionstabeller, vilket leder till en högre grad av normalisering än vad som är fallet i ett starschema. Grundprincipen är dock densamma i de båda typerna av scheman, det vill säga att relatera dimensionerna till en faktatabell för att på så sätt möjliggöra flerdimensionella analyser av data.

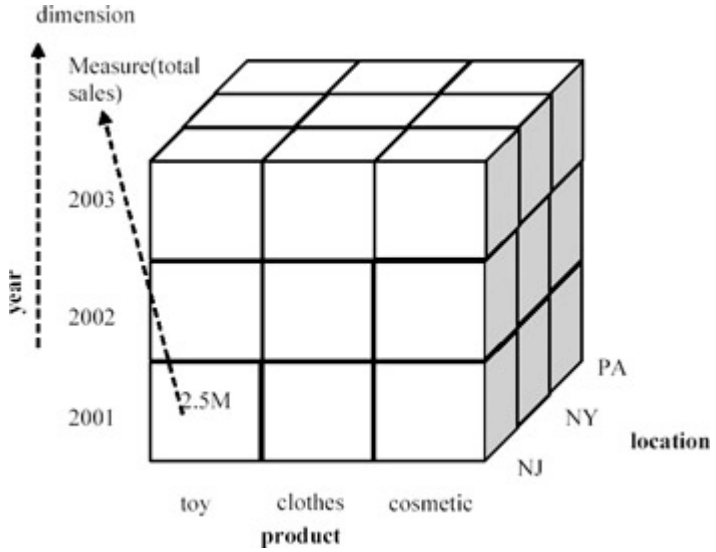
Figur 2 – Schematisk bild av ett snowfalke schema (Powell, 2006).

En nackdel med snowflake-schemat är dock att normaliseringen leder till att fler JOIN operationer måste utföras för att returnera resultatet av olika SQL-frågor. Som framgår av diskussionen ovan kan JOIN operationer ha en negativ inverkan på databasens performance om antalet rader i en tabell är många, vilka de ofta är i en faktatabell (Powell, 2006).

2.1.4 OLAP kub

OLAP (Online Analysis Processing) är en viss typ av applikationer vilka analyserar de data som finns sparade i datawarehouset (Robert Wrembel och Christian Koncilia, 2007). Fokus för denna typ av applikationer är att göra analyser av data (exempelvis försäljning) utifrån olika dimensioner (exempelvis kund, produkt och geografisk lokalisering). Analyserna kan användas till att exempelvis finna avvikelser och mönster ibland all data. För att göra analyseren används en datastruktur vilken kan liknas vid en flerdimensionell kub och därför kallas för OLAP kub (Computation of OLAP Cubes, 2006). OLAP kuber är lämpliga att använda då stora datamängder skall analyseras och snabbt visas för användaren. Det är möjligt genom att alla eller stora delar av värdena i kuben beräknats på förhand varvid de snabbt kan hämtas och visas för användaren.

I en *n-dimensionell* OLAP kub kan data analyseras ur *n* dimensioner där varje cell i kuben innehåller ett mätetal (exempelvis försäljning), vilket utgör skärningspunkten mellan dimensionerna. Därmed kan exempelvis försäljningen för leksaker (eng. toy), som finns belägen i New York (NY) år 2003 representeras som skärningspunkten, det vill säga cellvärdet, mellan dessa dimensioner i kuben, se Figur 3 nedan.



Figur 3 - Olap kub innehållande försäljning per produkt, år och geografisk placering (Computation of OLAP Cubes, 2006).

Styrkan med OLAP kuber är att analyser möjliggörs både på detaljerad och aggregerad nivå. Detta möjliggörs genom operationer såsom summering, subtrahering och division av cellvärden utmed en eller flera värden av de olika dimensionerna. Ur samma kub som försäljningen för leksaker i New York (NY) år 2007 kan beräknas går det lika enkelt att få fram den totala försäljningen för leksaker i New York för åren 2001, 2002 och 2003. Detta sker genom att summera alla celler vilka utgör skärningspunkterna mellan dimensionerna år, produkt och placering för dessa värden. Aggregatet görs genom att med SQL låta gruppera data i datawarehouset utmed dimensionerna år och produkt och summera de grupperingar som innehåller värdet New York för placeringsdimensionen samt leksaker för produktdimensionen.

3 Analys

I föreliggande kapitel analyseras användarkrav och dess möjliga effekter på BI systemets uppbyggnad. Analysen syftar sålunda till att utreda möjliga lösningar av systemet utifrån de användarkrav som föreligger.

3.1 Analys av användarkrav

Av intervjuer och workshops genomförda med Vd:n på Know IT i Göteborg AB, ekonomiansvarig samt ansvariga för respektive kostnadsställe, det vill säga den grupp av användare vars behov systemet skall tillgodose, har följande användarkrav identifierats:

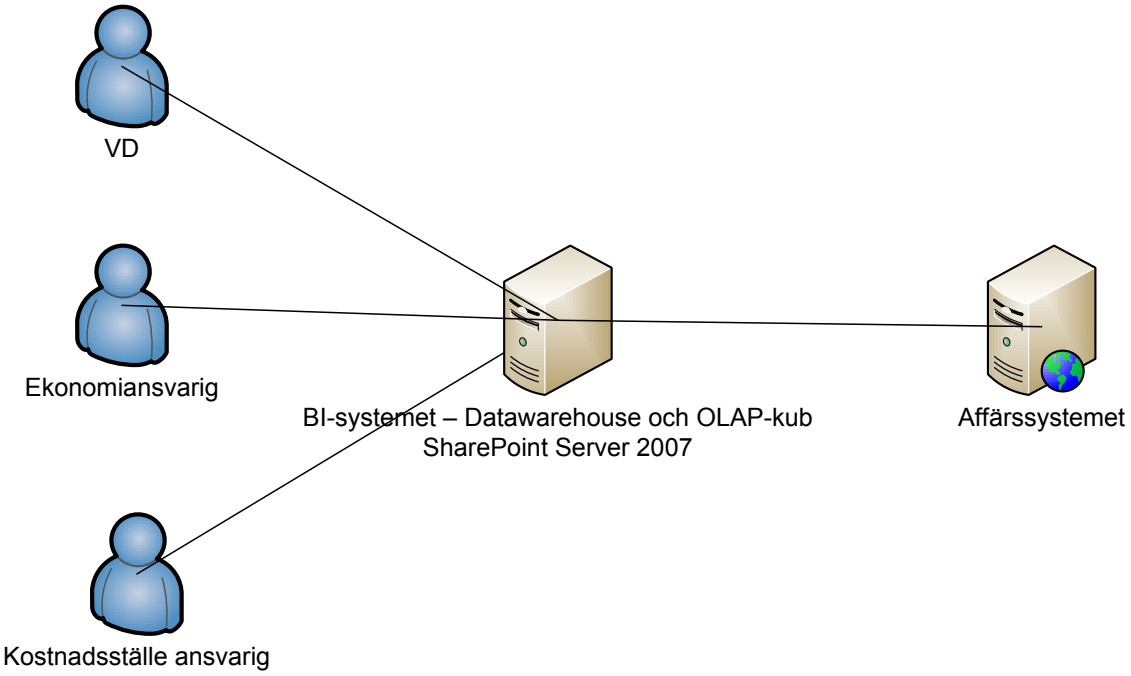
ID	Benämning	Beskrivning
F-01	Resultat per konsult	Det skall vara möjligt att se det ekonomiska resultatet för en konsult per: • Vecka • Månad • År Resultatet skall beräknas genom att subtrahera alla intäkter med alla kostnader som konsulten genererat. Vidare skall alla overheadkostnader för ett kostnadsställe fördelas så att konsulten erhåller en kostnad motsvarande kostnadsställets overheadkostnad dividerat med det totala antalet konsulter som arbetar inom kostnadsstället. Respektive konsult skall också påföras en andel utav bolagets overheadkostnader som motsvarar bolagets overheadkostnad dividerat med det totala antalet konsulter som arbetar inom bolaget.

		Overheadkostnaden för en konsult skall beräknas på följande vis: Overheadkostnad per konsult = (Overheadkostnad för kostnadsstället där konsulten arbetar / antal konsulter anställda vid kostnadsstället där konsulten arbetar) + (Overheadkostnad för bolaget / antal konsulter anställda vid bolaget)
F-02	Resultat per kostnadsställe	Det skall vara möjligt att se det ekonomiska resultatet för ett kostnadsställe per: • Vecka • Månad • År Resultatet för ett kostnadsställe skall beräknas som summan av resultatet för alla konsulter som arbetar inom kostnadsstället och därtill skall intäkter genererade av projekt inom kostnadsstället, men som inte kan härledas till en viss konsult, adderas . Resultatet för ett kostnadsställe beräknas på följande vis: Resultat för kostnadsställe = (∑ Resultatet per konsult som arbetar inom kostnadsstället) + Intäkter registrerade på projekt inom kostnadsstället, som inte kan härledas till en viss konsult.
F-03	Resultat för bolaget	Det skall vara möjligt att se det ekonomiska resultatet för

		bolaget per: • Vecka • Månad • År Resultatet för bolaget som helhet skall beräknas som summan av resultatet för alla konsulter som arbetar inom bolaget. Resultatet för bolaget beräknas på följande vis: Resultat för bolaget = ∑ Resultatet per konsult som arbetar inom bolaget.
F-04	Systemåtkomst	Ansvarig för ett kostnadsställe skall endast få tillgång till det ekonomiska resultatet för detta kostnadsställe samt för respektive konsult som arbetar inom kostnadsstället. Ekonomiansvarig och VD skall få se resultatet för alla kostnadsställen, bolaget som helhet samt för respektive konsult som arbetar inom bolaget.
F-05	Plattform	Systemet skall baseras på Microsoft SharePoint Server 2007 och Microsoft Sql Server 2005.
F-06	Utbyggbarhet	Systemet skall designas så att det enkelt i framtiden går att bygga ut de ekonomiska analyserna för fler nyckeltal, budget och projekt.

3.2 Analys av systemmiljö

En av förutsättningarna för att kunna analysera de ekonomiska data som det slutgiltiga systemet skall göra är att systemet får tillgång till dessa. Därför behöver den omgivande systemmiljön klarläggas på ett tidigt stadium av utvecklingsarbetet för att på så sätt ge förutsättningarna för det kommande design- och implementationsarbetet vad beträffar åtkomst av data. Figur 4 visar hur BI-systemet och dess omgivande miljö ser ut. Av figuren framgår att BI-systemet måste kommunicera med Know IT I Göteborgs AB:s affärssystem där ekonomiska data finns samlade. En viktig sak att notera här är dock att data inte står under Know IT i Göteborg AB:s kontroll, utan handhas av leverantören till affärssystemet. Vidare framgår av Figur 4 relationen mellan BI-systemet och dess användare. Av användarkraven framgår att åtkomsten till de data som finns i BI-systemet skall ske genom en intranätportal baserad på SharePoint Server 2007.



Figur 4 - BI-systemet och dess omgivande miljö.

3.3 Analys av data till grund för de ekonomiska analyserna

Av användarkraven i stycke ”3.1 Analys av användarkrav” framgår att BI-systemet skall göra det möjligt att analyser det ekonomiska resultatet för:

- En konsult per vecka, månad och år.
- Ett kostnadsställe per vecka, månad och år.
- Hela bolaget per vecka, månad och år.

För att uppfylla dessa krav behöver följande data utbytas mellan affärssystemet och BI-systemet:

- **Personer/Konsulter:** Alla personer/konsuter som finns registrerade i affärssystemet för Know IT i Göteborg AB.
- **Konton**: Alla konton som finns registrerade i affärssystemet för Know IT i Göteborg AB.
- **Intäkter:** Alla intäkter, för alla intäktskonton, med transaktionsdatum som finns registrerade i affärssystemet för Know IT i Göteborg AB. Dessa behövs för att kunna beräkna det ekonomiska resultatet.
- **Kostnader:** Alla kostnader, för alla kostnadskonton, med transaktionsdatum som finns registrerade i affärssystemet för Know IT i Göteborg AB. Dessa behövs för att kunna beräkna det ekonomiska resultatet.
- **Kostnadsställe:** Alla kostnadsställen, bolag är ett utav dessa, som finns registrerade i affärssystemet för Know IT i Göteborg AB.
- **Justeringar:** Alla justeringar som görs för ett projekt och konsult. Om det visar sig att det inte går att fakturera en debiterad timme görs en nedskrivning, det vill säga justering, vilken påverkar det ekonomiska resultatet.
- **Tidkod:** Alla tidkoder som används när en konsult rapporterar sin arbetade tid i affärssystemet. När konsulterna registrera tid som skall faktureras, det vill säga intäkter, registreras dessa för olika tidkoder. Tidkoder används för att göra en kategorisering av den debiterbara tiden utifrån vilken typ av arbete som utförts. Exempelvis finns en tidkod för projektledning, vilken utförs av projektledare, och en tidkod vanligt konsultarbete.
- **Övriga koder:** Alla koder som används för intäkter vilka inte direkt kan kopplas till en konsults nedlagda tid, exempelvis inköp.

3.4 Datautbyte med affärssystemet

All ekonomisk data som tillhör Know IT i Göteborg AB är idag samlad i ett affärssystem, vilket administreras av leverantören till systemet. Det innebär i sin tur att databasen inte står under Know IT i Göteborg AB:s kontroll varvid ett datautbyte med systemet på något sätt måste ske. När de anställda arbetar mot affärssystemet görs detta genom Java baserade klienter som kommunicera med en serverapplikation hos leverantören.

Det faktum att affärssystemet bygger på en distribuerad systemlösning eliminerar möjligheten för BI-systemet att hämta ekonomiska data direkt ifrån affärssystemets databas. Detta hade varit en möjlig lösning om Know IT i Göteborg AB hade administrerat systemet själva och sålunda haft en egen server vilken systemet och databasen exekverat på. Eftersom denna möjlighet inte längre finns för datautbyte mellan affärssystemet och BI-systemet är det främst två andra designlösningar som varit föremål för den fortsatta analysen kring utbytet av ekonomiska data:

1. Datautbyte genom kommunikation över SOAP-protokollet, det vill säga med hjälp av webbtjänster (webbservice).

2. Datautbyte av XML-strömmar över https-protokollet.

3.4.1 Datautbyte genom kommunikation över SOAP-protokollet

En möjlig lösning för utbyte av data mellan affärssystemet och BI-systemet skulle vara att låta systemen kommunicera med hjälp av webbtjänster (eng. webservice). Denna lösning bygger på att leverantören av affärssystemet erbjuder ett eller flera interface vilka BI-systemet i sin tur kan anropa med olika parametrar för att specificera de data som efterfrågas. Fördelen med att använda webbtjänster för att utbyta data med affärssystemet är att webbtjänster bygger på en specificerad standard vilket gör det möjligt för system att kommunicera med varandra oavsett om de är skrivna i samma programmeringsspråk eller inte. Exempelvis kan en applikation som är skriven i .Net utan problem kommunicera med en applikation skriven i Java om denna baseras på SOAP protokollet. Det beror på att kommunikation sker i ett XML baserat format och sålunda är inga datatyper, vilka är specifika för de olika språken, inblandade i kommunikationen. En annan fördel då .Net används vid kommunikationen är att en proxyklass med tillhörande metoder genereras automatiskt (Troelsen, 2007). För BI-systemet innebär det att ett datautbyte enkelt kan initieras med affärssystemet då det enda som krävs är anrop av olika metoder på det objekt som skapats utifrån proxyklassen. Kommunikationen med affärssystemet blir därmed transparent för BI-systemet då ett anrop till webbtjänsten ser ut som ett vanligt metodanrop på ett objekt.

Ett exempel på hur en webbtjänst skulle kunna användas för att hämta alla anställda ifrån affärssystemet framgår av Figur 5. Först skapas ett objekt ifrån den automatiskt genererade proxyklassen Employees. Sedan exekveras metoden getAllEmployees för objektet och returnerar en kollektion innehållande alla anställda.

```
EmployeeService employeeService = new EmployeeService();
List<Employee> employees = employeeService.getAllEmployees();
```

Figur 5 - Anrop av webbtjänst.

3.4.2 Datautbyte av XML-ström över https-protokollet

En annan möjlig lösning på problemet med att utbyta data mellan BI-systemet och affärssystemet är att paketera dessa i en eller flera XML-strömmar. Konceptuellt sett liknar denna lösning den föregående. Båda lösningarna förlitar sig på http/https protokollet för transport av data mellan de olika systemen. Skillnaden är dock att i denna lösning kommer data att utbytas med affärssystemet genom att BI-systemet först initierar ett anrop mot en specifik https adress, vilken utgör affärssystemets interface gentemot omvärlden. Anropet innehåller olika parametrar som avgör vilka data som skall utbytas. Användarnamn och lösenord måste också skickas med för att BI-systemet skall kunna autentiseras som en behörig användare.

Av Figur 6 framgår ett exempel på hur BI-systemet kan anropa affärssystemet över https protokollet. Anropet innehåller förutom https-adressen också parametrarna Användarnamn, Lösenord samt Anställda. Den sistnämnda parametern anger för affärssystemet att data innehållande alla anställda skall returneras, vilket sker i form av en

XML-ström. Ett exempel på hur denna ström kan se ut framgår av bilaga ”7.2 Bilaga 2 – Stagedatabasens uppbyggnad”. Det enda som skiljer en XML-ström från en vanlig XML-fil är att data här kommer som en dataström, och inte en fil, vilken sätts samman till ett XML-dokument i BI-systemet. Sålunda kommer XML-dokumentet inte att vara lagrat fysiskt på den server där BI-systemet exekveras. Fördelen med detta är att inga XML-filer måste skyddas på en fysisk enhet (i detta fall den server BI-systemet exekverar på) från att obehöriga personer kommer åt dessa och de data filerna innehåller. Ytterligare en fördel med denna lösning i det specifika fallet är att det redan idag finns ett datautbyte mellan ett annat system som Know IT i Göteborg AB använder och affärssystemet. Därmed har denna lösning redan visat sig fungera bra. Nackdelen är dock att ett anrop till affärssystemet måste göras om varje gång en viss data söks, även om denna tidigare efterfrågats av BI-systemet.

https://*addresstillaffärssystemet*?Username=*användarnamn*&Password=lösenord
&Employees=getallemployess
Figur 6 - Exempel på anrop över https-protokollet för att hämta alla anställda.

3.4.3 Karaktäristiken hos utbytt data

En vidare analys av data som utbyts vid ett datautbyte visar att de varierar i storlek. Viss data förändras sällan och kan anses vara relativt konstant i storlek sett över tiden, det vill säga ny data tillkommer sällan. Data som tillhör denna kategori är:

- Kostnadsställen
- Personer
- Konton
- Tidkoder
- Övriga koder
- Budget kategorier
- Justeringar
- Projekt

Övriga data kan däremot klassificeras som dynamiska, det vill säga ny data tillkommer ofta, vilket innebär att mängden data som behöver utbytas tenderar att vara stor. Data som tillhör denna kategori är:

- Intäkter
- Kostnader

Det är viktigt att beakta detta då stora datamängder kan påverka datautbytesprocessen negativt. Exempelvis kan en stor datamängd få ett anrop med https protokollet att resultera i en ”request timeout”, vilket innebär att anropet tar för lång tid att genomföra och därmed avslutas utan att några data utbytts.

3.5 Analys av datalagret

I detta stycke redogörs för val och överväganden avseende designen av BI-systemets datalager. Diskussionen tar först utgångspunkt i mer generella överväganden vid design av ett datawarehouse och en lösning som centreras kring denna typ av datalagringsplats. Därefter riktas fokus mer mot den specifika implementationen av datalagret i BI-systemet som är i fokus för denna rapport.

3.5.1 Generella designöverväganden

Då syftet med ett datawarehouse oftast är att ge en historisk bild av de data som finns spridda i olika system runt om i ett företag kommer insättning av nya data vara den vanligaste transaktionen som sker i datawarehouset. Insättning sker genom att data hämtas ifrån kringliggande system och sedan läggs till i datawarehouset. Processen att hämta och sätta in ny data är dock ofta inte så enkel som det kan verka vid en första anblick. Data i skilda system är ofta representerad på olika sätt. Ett exempel på detta kan vara att alla primärnycklar i ett system A är heltal (int) medan alla primärnycklar i ett system B är representerade som bokstäver (char). I datawarehouset har man bestämt att alla primärnycklar skall vara av typen heltal, vilket gör att nycklarna i system B måste transformeras till denna representation innan data ifrån detta system kan läggas till i datawarehouset. Primärnycklarna i ett datawarehouse brukar vara specifika för denna typ av databas och behöver därmed ersätta eventuella primärnycklar som förekommer i ursprungssystemen. I ett datawarehouse brukar denna typ av nycklar därför kallas för surrogatnycklar (eng. surrogate keys).

En annan viktig fråga som behöver beaktas vid en lösning som centreras kring ett datawarehouse är i vilka steg transformering och insättning av ny data skall ske. Kan systemet som hämtar data ifrån kringliggande system till datawarehouset få tillgång till all nödvändig data på en gång eller måste en mellanlagring och processning av data ske innan den slutligen kan läggas till i datawarehouset. Det kan också vara så att denna mellanlagring av data fungerar som input till efterkommande hämtningar av ny data ifrån källsystemen.

Ovanstående resonemang och frågeställningar är några av de aspekter som enligt författaren behöver beaktas då man designar en lösning baserad på ett datawarehouse. Den fortsatta analysen fokuserar på olika designval som är möjliga för systemet till grund för denna rapport.

3.5.2 Stagingarea

Av analysen i avsnitt ”3.4 Datautbyte med affärssystemet” framgår att Know IT i Göteborg AB inte själva förfogar över sina ekonomiska data. Geografisk finns dessa lagrade i en databas som leverantören till systemet äger och administrerar. Därför måste ett datautbyte på något sätt ske mellan BI-systemet och affärssystemet innan det är möjligt att göra de ekonomiska analyser som eftersträvas.

Ett första designövervägande är då var de erhållna data skall sparas innan analyserna kan genomföras. Ett första designval är om dessa skall sparas på ner på hårddisken eller om de skall sparas internt i datorns arbetsminne under själva datautbytesprocessen. För att göra datautbytesprocessen så snabb som möjlig, med hänsyn till att det kan finnas felaktigheter i de data som utbytts, fokuserar den fortsatta diskussionen på en lösning där data sparas på hårddisken. Här finns främst två möjliga lösningar, fil eller databas, att beakta i enlighet med det som Kimball och Ciestra (2004) anger för en stagingarea.

Den första lösningen innebär att data sparas i en eller flera filer. En sådan lösning ger också upphov till frågan om vilket filformat som skall användas. I enlighet med diskussionen i stycke ”3.4.2 Datautbyte av XML-ström över https-protokollet”, är en möjlighet att spara ner data i en eller flera XML-filer. En annan möjlig lösning skulle vara att spara data i en eller flera kommaseparerade (”,”) filer. Fördelen med att använda filer är att det går snabbt att spara och hämta de data som utbytts. Nackdelen är dock att det inte med enkelhet går att analysera de data som finns sparade i filerna som om de fanns sparade i en databas då det går att ställa frågor med hjälp av SQL-syntax.

Den andra lösningen bygger istället på att spara de data som utbytts i en databas. Istället för att använd sig av elementen i en XML-fil eller kommaseparering i en textfil används då tabeller för att spara data. Fördelarna med denna lösning är som framgår ovan att det är enkelt att analysera relationerna mellan data, vilket förenklar processen med att finna eventuella felaktigheter. Nackdelen är dock, som Kimbal och Ciestra (2004) påpekar att prestandan i datautbytesprocessen kan påverkas negativt på grund av den extra overhead som tillkommer när databasen bland annat skall hantera insättning samt läsning av data.

3.5.3 Produktionsdata

En stagearea löser problematiken med att få tillgång till de data som affärssystemet förfogar över. Här finns dock inga krav på att data är vare sig konsekvent eller fullständighet, vilket måste garanteras för att de ekonomiska analyserna skall kunna anses vara korrekta. Hade det varit möjligt att hämta data direkt ifrån affärssystemet hade detta krav kunnat garanteras på grund av att all data här måste vara fullständig och logiskt relaterad till varandra, ett krav affärssystemet måste uppfylla för att respresentera Know IT i Göteborg AB:s ekonomiska ställning. En area med motsvarande funktionalitet behövs därför i BI-systemet för att garantera att data är fullständig samt att det finns logiska relationer mellan de data systemet förfogar över.

En möjlig lösning för att tillgodose detta krav är att skapa en databas som använder sig av foreign-keys för att modellera relationerna mellan de data som är logiskt relaterade till varandra utifrån Know IT i Göteborg AB:s ekonomiska förutsättningar. Därmed kan denna databas anses vara en spegling utav affärssystemet. Om all data som utbytts mellan BI-systemet och affärssystemet kan relateras till varandra i denna databas utan att bryta mot kraven i foreign-keys relationerna kan kraven på fullständighet och konsistens anses vara uppfyllda. Nackdelen med denna lösning är dock att den riskerar att sänka prestandan på dataladdningsprocessen som helhet på grund av den extra overhead som tillkommer när databasen bland annat skall hantera insättning samt läsning av data.

3.6 Analys av datatransformering

När väl data utbytts med affärssystemet och sparats i stagedatabasen behöver denna gå igenom ett antal transformeringssteg för att garantera att den är konsistent, det vill säga säkerställa att data inte gått förlorad eller modifierats i datautbytet. Dessutom måste data laddas ifrån stagedatabasen till dess slutgiltiga destination som utgörs av datawarehouset. Passande för denna typ av uppgifter är ett ETL-verktyg, vilket redogörs för ovan. Dock finns det en rad överväganden att göra beträffande hur ETL-verktyget skall strukturera processen för att garantera att data är konsistent samt optimera dataladdningsprocessen mellan de olika databaserna. Extraherings och dataladdningsprocessen behöver därmed beakta följande:

1. Att data hämtat ifrån affärssystemet är konsistent med avseende på fullständighet, det vill säga att inga data gått förlorad, samt integritet, det vill säga att inga data av misstag modifierats under datautbytet.

2. Optimera dataladdningsprocessen för att minimera den tid dataladdningsprocessen tar att genomföra.

För att garantera att inga av de data som hämtas under datautbytesprocessen med affärssystemet gått förlorad måste någon form av verifiering genomföras. En rad potentiella felkällor föreligger vid ett datautbyte, vilka behöver motverkas för att inte introduceras i datawarehouset. Några exempel på sådana följer nedan:

- Anslutningen till affärssystemet avbryts, vilket göra att all data inte hämtas under ett datautbyte. Därmed förloras data vilket gör att den befinner sig i ett ofullständigt tillstånd. En möjlig orsak till detta tillstånd är att internetanslutningen bryts och processen inte kan fullgöras.

- En fil kan ha blivit korrupt, som exempelvis ett resultat av att den inte generats på ett korrekt vis, och därmed gör data ofullständig.

- Data som redan hämtats vid ett tidigare datautbyte kan komma att hämtas igen, exempelvis uppgifter om anställda, vilka inte skall läggas till i datawarehouset igen såvida befintliga uppgifter inte skall uppdateras.

- Data som tillåts att inte ha något värde, exempelvis Null-värden, i affärssystemet måste ha ett värde i datawarehouset. Dessa data måste korrigeras innan de laddas in i datawarehouset för att inte göra systemet inkonsistent.

Ett viktigt krav i många BI-lösningar är att optimera dataladdningsprocessen ifrån källsystemet till datawarehouset. Det på grund av att det ofta rör sig om mycket stora datamängder som skall utbytas mellan systemen (Kimball 2004). En strävan för BI-systemet som är i fokus för denna rapport är därför också att designa dataladdningsprocessen så att den går snabbt att genomföra. De designval som spelar in

här är först och främst valet av den teknik eller verktyg som används för att genomföra dataladdningen. En möjlig lösning är att bygga en applikation i exempelvis .Net som innehåller den funktionalitet som behövs för att ladda data ifrån affärssystemet till datawarehouset. Applikationen behöver i sin tur innehålla logik för att läsa data ifrån datakällan och sedan uppdatera datawarehouset med denna. En sådan lösning kommer att behöva innehålla både logik för att garantera att data är konsistent samt logik i form av SQL-syntax för att uppdatera datawarehouset. Ett avvägande här är vart logiken skall placeras, antingen kan mycket av logiken för att garantera kvalitén på data görs i själva applikationskoden genom exempelvis IF och ELSE satser, ett exempel på detta framgår av Figur 7 nedan, eller så kan denna logik hanteras av själva databasen vid en uppläsning av data ifrån källsystemet. Problemet med en lösning där konsistensen kontrolleras i databasen är dock att den endast är applicerbar när vi har att göra med ett system läser data ifrån en databas till en annan. Det i sin tur gör det svårt att låta dataladdningsprocessen hämta data ifrån andra källor om sådana krav skulle tillkomma.

```
public void CheckNull(object columnToCheck)
{
      if(columnToCheck is string && columnToCheck == DbNull)
      {
          columnToCheck = ”Empty”;
      }
}
```

Figur 7 – Metod i ett högnivåspråk, som exempelvis .Net, för att ersätta ett null-värde med dess representation i datawarehouset.

Ett annat viktig designövervägande är hur de olika stegen i dataladdningsprocessen skall genomföras. En möjlighet är att genomföra ett steg i taget, det vill säga sekventiellt, eller låta olika steg, vilka inte påverkas av varandra, löpa parallellt. En förutsättning för att kunna använda parallell exekvering för dataladdningsprocessen, som är i fokus för denna rapport, baseras på att separata steg kan urskiljas som är särskilda ifrån och inte påverkar varandra. Exempelvis kan alla konsulter laddas samtidigt som alla kostnadsställen laddas. Det är först i anställningsrelationen som en konsult relateras till ett kostnadsställe.

3.6.1 Datawarehouse

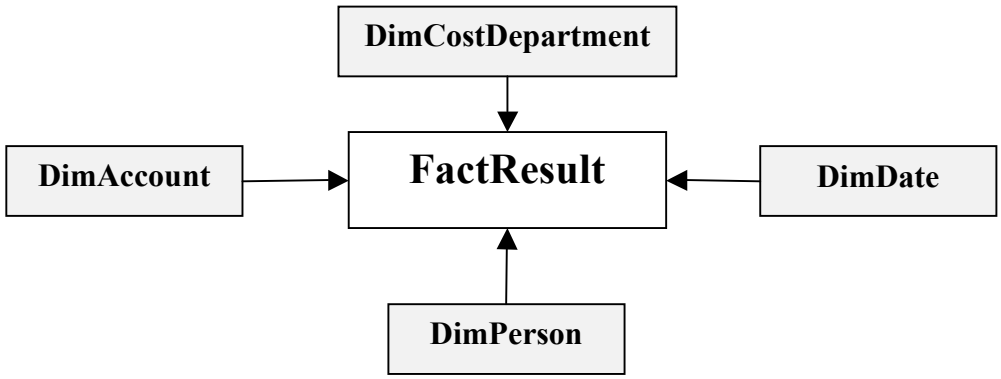
Ett datawarehouse kan designas på olika sätt. I avsnitt” 2.1.3 Datawarehouse (DW)” ovan framgår två möjliga scheman, det vill säga star-schema och snowflake-schema, att konceptuellt sett relatera dimensionstabeller och faktatabeller till varandra. I den BI-lösning som är i fokus för denna rapport utgör båda dessa scheman möjliga utgångspunkter för datawarehousets uppbyggnad.

Ovan framgår att ett datawarehouse ofta karaktäriseras av att innehålla ett stort antal rader i en eller flera faktatabeller, vilket är en följd av att den skall ge en historisk bild av olika data. Det faktum att JOIN-operationer är dyra i databassammanhang gör star-schemat fördelaktigt i BI-lösningen. Samtidigt leder detta schema till en lägre grad av normalisering än vad snowflake-schemat gör. Star-schemat skulle sålunda ge BI-systemet bra prestanda genom att reducera antalet JOIN-operationer som måste göras mellan olika tabeller. Samtidigt kan den lägre graden av normalisering, som star-schemat ger upphov

till, öka dubbellagringen av data då dimensionstabellerna här innehåller data som i ett snowflake-schema finns uppdelad i fler tabeller.

I Figur 8 återfinns ett exempel på hur ett star-schema kan användas för att designa en faktatabell med tillhörande dimensionstabeller som ligger till grund för beräkningar av Know IT i Göteborg AB:s ekonomiska resultat. Dimensionstabellerna innehåller information om kostnadsställe, konto, konsult och datum, vilket gör det möjligt att se det ekonomiska resultatet ur olika perspektiv. Ett möjligt perspektiv framgår av användarkrav F-01, vilket säger att det ekonomsiska resultatet skall kunna visas per konsult och månad.

Karaktäriserande för star-scheman är att det innehåller en låg grad av normalisering vilket får till följd att viss data återupprepas för ett flertal rader i en tabell. Av Figur 9 framgår ett exempel på hur den låga graden av normalisering får data i kolumnerna ”Month” och ”Year”, i tabellen DimDate, att återfinnas för flera rader.



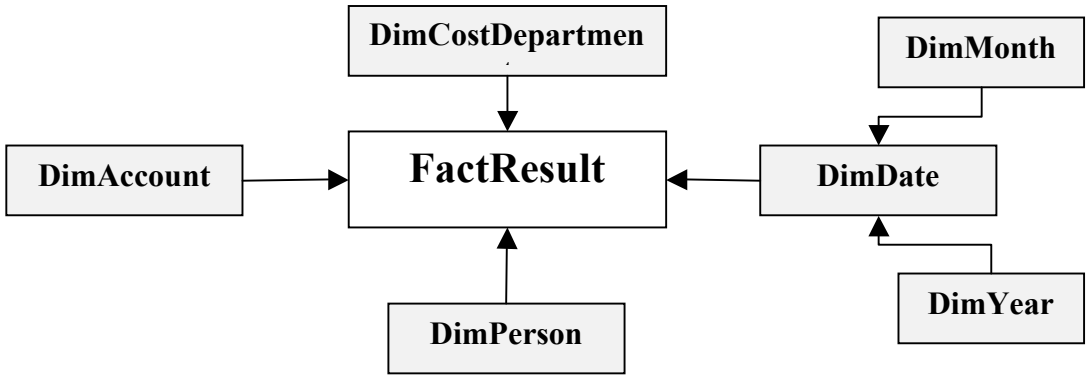
Figur 8 - Exempel på hur det ekonomiska resultet för Know IT i Göteborg AB kan representeras med hjälp av ett star-schema.

DateKey	Date	Month	Year
1	2009-05-11	May	2009
2	2009-05-12	May	2009
3	2009-05-13	May	2009

Figur 9 - Exempel på hur viss data upprepas i en dimensionstabell på grund av star-schemats låga grad av normalisering.

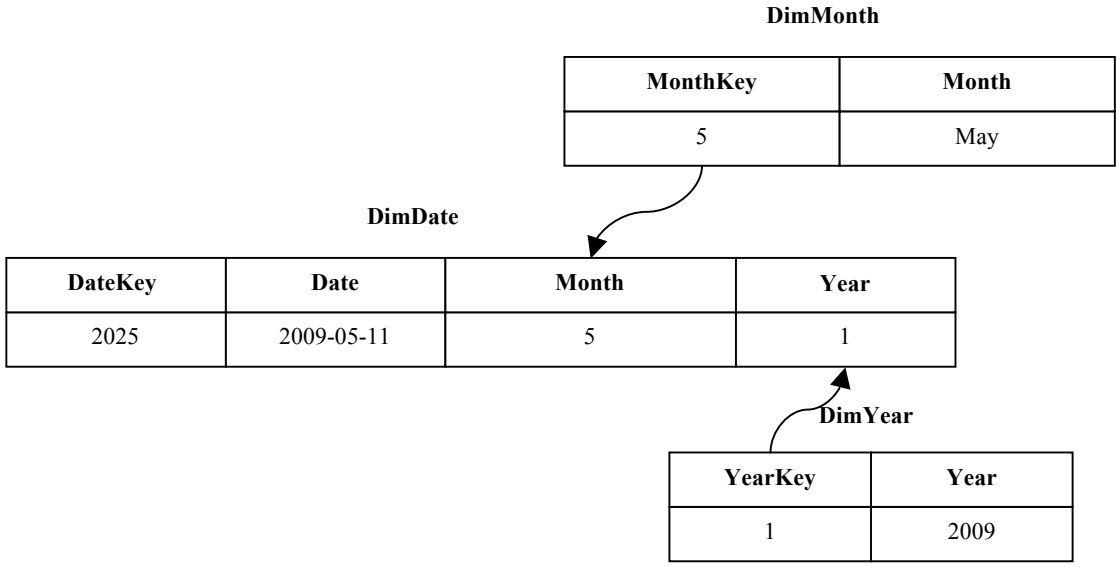
För att nå en högre grad av normalisering kan ett snowflake-schema istället användas. Figur 10 visar konceptuellt hur ett sådant kan se ut när det ekonomiska resultatet för Know IT i Göteborg AB modelleras med hjälp av dimensions- och faktatabeller i datawarehouset. Tabellen DimDate har här normaliserats med avsikt på attributen ”Month” och ”Year”, det vill säga dessa har brutits ut i fristående dimensionstabeller, DimYear och DimMonth, och relateras till DimDate genom en foreign-key relation.

Notera att det troligen finns fler attribut vilka kan normaliseras för dimensionstabellerna i Figur 10. Dessa utreds dock ej vidare på grund av att de följer samma resonemang som ovan och visualiseras konceptuellhet av Figur 10.



Figur 10 - Exempel på hur det ekonomiska resultet för Know IT i Göteborg AB kan representeras med hjälp av ett snowflake-schema.

Av Figur 11 framgår hur data skulle kunna representeras i dimensionstabellerna DimDate, DimMonth och DimYear när ett snowflake-schema används.



Figur 11 - Exempel på hur data kan representeras i dimensionstabellen DimDate när den normaliserats med hjälp av ett snowflake-schema.

4 Design

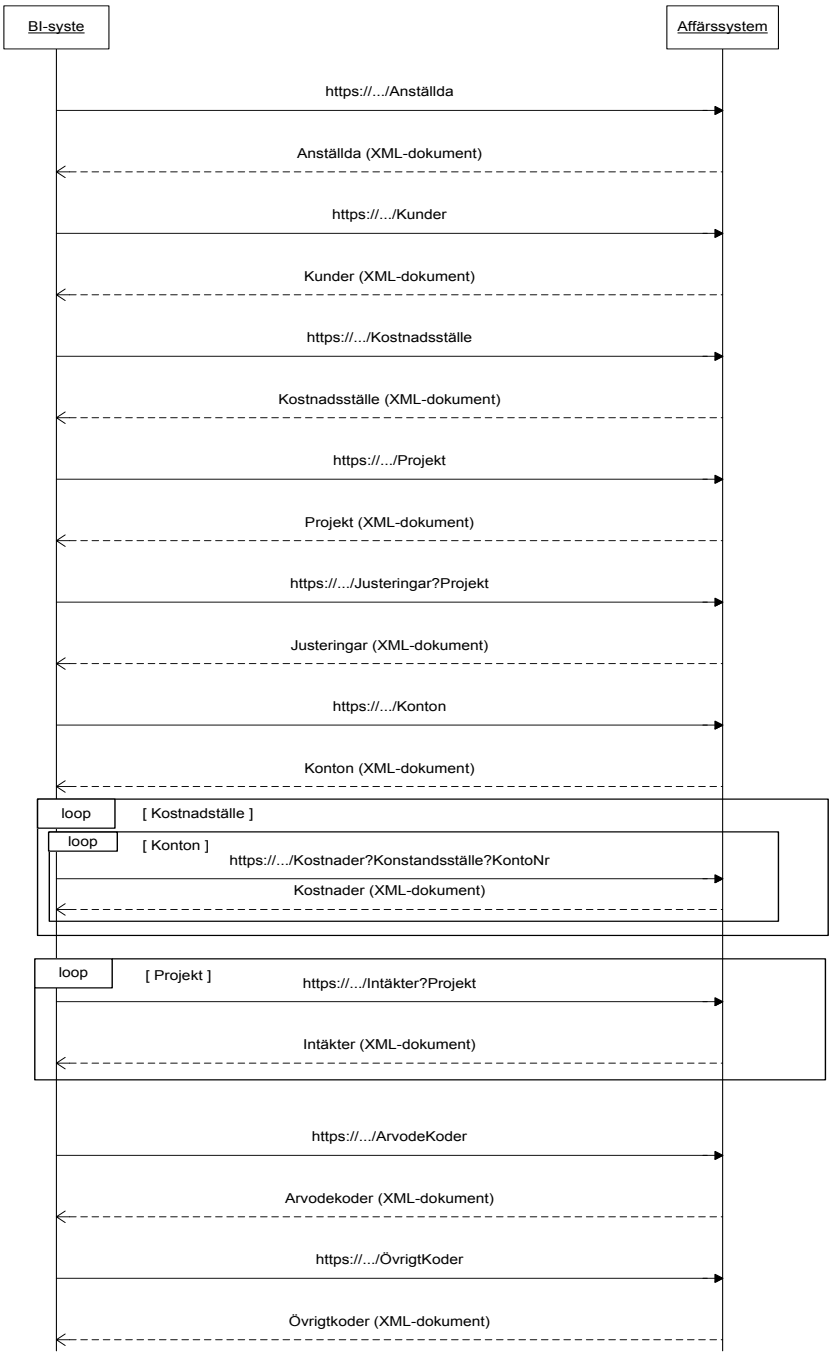
I det föreliggande kapitlet redogörs för implementationen av datalagret i det behandlade systemet som ligger till grund för denna rapport. Författaren argumenterar här för den valda lösningen och diskuterar för- och nackdelar med denna implementation.

4.1 Datautbyte med affärssystemet

Den lösning som slutligen implementerats för att hantera datautbytet mellan BI-systemet och affärssystemet följer det designförslag som diskuterats i stycke ”3.4.2 Datautbyte av XML-ström över https-protokollet”. Det bör tilläggas att detta designval inte var självklart. Det finns mycket som talar för att användandet av webbtjänster vid datautbytet skulle ha varit en bättre lösning än den valda. Det faktum att proxyklasser genereras automatiskt utifrån de data (baserat på SOAP protokollet) som utbyts mellan BI-systemet och affärssystemet förenklar i många avseenden implementeringen av de klasser som hanterar datautbytet. Istället för att implementera mer eller mindre avancerade klasser för att hämta data och läsa denna direkt ur ett XML-dokument har användandet av webbtjänster fördelen att det endast krävs ett enkelt metodanrop för att genomföra hela datautbytet mellan systemen. Webbtjänsten kan dessutom returnera en datatyp direkt (exempelvis en Array), vilken på ett naturligt sätt kan användas direkt tillsammans med övrig kod.

Den slutgiltiga lösningen innehåller tio definierade XML-strömmar (se Bilaga 1 - Xmlströmmar för datautbyte mellan BI - och affärsystemet). Av diskussionen som följer framgår det dock att antalet XML-strömmar som verkligen utbyts mellan BI-systemet och affärssystemet, varje gång datawarehouset skall uppdateras, kommer att variera beroende på beskaffenheten hos de data som utbyts. Varje XML-ström innehåller en viss avgränsad del av data, exempelvis innehåller en av XML-strömmarna alla anställda på Know IT i Göteborg AB medan en annan innehåller alla kunder som företaget har. Då några av XML-strömmarna är beroende av data ifrån andra XML-strömmar för att data skall kunna utbytas är den inbördes ordningen viktig i datautbytessprocessen. Exempelvis måste XML-strömmen innehållanden alla konton utbytas före XML-strömmen som innehåller alla kostnader. Det på grund av att kostnader hämtas med avseende på det konto som de finns registrerade på i affärssystemet. Därmed måste BI-systemet först avgöra vilka konton som är kostnadskonton och sedan göra en förfrågan per sådant konto för att alla kostnader skall kunna utbytas. Det är enkelt att inse att antalet XML-strömmar som måste utbytas mellan de båda systemen kommer att vara fler till antalet än de tio som finns definierade i bilaga ”7.1 Bilaga 1 - Xmlströmmar för datautbyte mellan BI - och affärsystemet”, vilket beror på att utbytet av XML-strömmarna ”7.1.5 XML innehållande information om alla kostnader” och ”7.1.11 XML innehållande alla övriga intäkter” kommer att vara beroende på tidigare utbytt data.

Figur 12 visar konceptuellt sett hur ett datautbyte mellan de båda systemen ser ut.



Figur 12 – Datautbyte mellan BI-systemet och Affärssystemet.

4.2 Databasdesign

4.2.1 Stageingdatabas

För att försäkra att de data som utbyts med affärssystemet är konsistenta och inte blivit korrupta i datautbytesprocessen med BI-systemet används en stagedatabas. I denna

laddas all data som hämtats ifrån affärssystemet utan krav på att de är vare sig fullständiga eller konsistenta. En sådan kontroll av hämtad data sker då de laddas från stagedatabasen till produktionsdatabasen. Stagedatabasen fyller i den implementerade lösningen främst tre behov:

1. Den agerar som en mellanlagringsplats av de data som finns i affärssystemet.

2. Möjliggöra hämtning av data som är beroende av tidigare hämtad data.

3. Möjliggöra felsökning av hämtad data.

Stagedatabasen fungerar här främst som en mellanlagringsplats i enlighet med det som Kimball och Caserta (2004) beskriver. Sålunda utgör den en spegling av de data som återfinns i affärssystemet. Vid en första anblick kan det verka onödigt att spara samma data som finns i affärssystemet i stagedatabasen. Valet att göra detta har dock sin förklaring i punkterna 1, 2 och 3 ovan.

En bidragande orsak till att all data som hämtats ifrån affärssystemet sparas ner i stagedatabasen är att många utav https anropen är beroende av tidigare hämtad data för att kunna hämta ytterligare data. Exempelvis måste alla konton först hämtas ifrån affärssystemet innan alla kostnader kan hämtas, vilket beror på kostnader är bundna till olika konton. Stagedatabasen möjliggör därmed att XML-strömmar, vilka är beroende av data ifrån andra XML-strömmar i datautbytesprocessen, kan hämtas ifrån affärssystemet. Samma funktionalitet som stagedatabasen erbjuder för att kunna hämta data hade också gått att uppnå genom att spara data i filer, exempelvis XML-filer, eller genom att använda sig av datatabellklasser i .Net med vars hjälp man kan skapa tabeller i arbetsminnet på webbservern med motsvarande funktionalitet som tabellerna i stagedatabasen. Problemet med denna lösning är dock att de data som utbytts med affärssystemet inte fysiskt finns sparade på hårddisken, vilket försvårar arbetet med att låta ETL verktyget föra över data till produktionsdatabasen.

En annan fördel med att låta spara ner all data som hämtats ifrån affärssystemet i stagedatabasen är att det möjliggör felsökning utav data om något gått fel i ett datautbyte. Eftersom data finns sparade i stagedatabasens tabeller kan denna inspekteras om det exempelvis visar sig att data inte är konsistent då de laddas över till produktionsdatabasen. Därmed blir det enklare att administrerar BI-systemet. Hade data bara sparats i exempelvis webbserverns arbetsminne i form av datatabeller hade en sådan felsökning inte varit lika enkel. Det på grund av att data försvinner ur tabellerna då programmet inte längre exekverar. Dock hade loggfiler kunnat vara till viss hjälp i en sådan situation, men att kunna ställa relevanta SQL frågor gentemot stagedatabasen ger större möjligheter att kunna finna orsaken till varför ett fel uppstått i form av exempelvis förlorade data.

Stagedatabasens uppbyggnad
Stagedatabasen uppbyggnad framgår av ” Bilaga 2 – Stagedatabasens uppbyggnad”. För att göra det möjligt att ladda in all data i databasen som utbytts med affärssystemet, även om det skulle saknas några data eller data inte är konsistent, finns det i denna databas inga restriktioner såsom foreign keys mellan olika tabeller.

4.2.2 Produktionsdatabas

Produktionsdatabasen innehåller data vilka kontrollerats med avseende på konsistens och fullständighet. Med konsistens avses här att data i olika tabeller relaterats till varandra med hjälp av foreign keys samt att inga data försvunnit i denna konsistenskontroll och sålunda kan anses vara fullständiga. Produktionsdatabasen kan därmed anses vara en lokal spegling av affärssystemet, vilken gör det enkelt att låta ETL verktyget för över data till datawarehouset.

Användandet av denna databas gör att inga felaktiga data introduceras i datawarehouset. Detta på grund av eventuella felaktigheter identifieras redan då data laddas in i produktionsdatabasen. Upptäcks några fel vid denna laddning avslutas den och all data som hunnit laddas in i databasen tas bort genom en rollback för att inte lämna den i ett inkonsistent tillstånd.

Produktionsdatabasens uppbyggnad
Produktionsdatabasen består av tabellerna nedan. För att garantera att data är konsistent och fullständig används foreign keys för att logiskt relatera data i olika tabeller med varandra. Nedan framgår Produktionsdatabasens uppbyggnad, som bland annat visar hur olika tabeller är relaterade till varandra.

Adjustments
Tabell som innehåller information om alla justeringar. En justering anger om det skett en uppskrivning eller nedskrivning för ett. En justering anges i det monetära värdet av upp- eller nedskrivningen.

Följande attribut beskriver en justering.

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
AdjustmentID	Genererad primärnyckel.	int	Ja	Nej	Nej
ProjectID	Projektets primärnyckel.	int	Nej	Ja	Nej
Amount	Storleken på justeringen	float	Nej	Nej	Nej
Date	Datumet då justeringen gjordes.	smalldatetime	Nej	Nej	Nej
Type	Typ av justering.	nchar(10)	Nej	Nej	Ja

Adjustment_OtherCode
Tabell som relaterar information om en justering till en övrig kod. Summan av justeringen härleds sålunda till en övrig kod för det avsedda projektet.

Följande attribut beskriver för justerings- och övrig kod tabellen:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
AdjustmentID	Justeringens primärnyckel.	int	Ja	Ja	Nej
OtherCodeID	Övrig kodens primärnyckel.	int	Ja	Ja	Nej

BudgetCategory
Tabell som innehåller information om budgetkategorier.

Följande attribut beskriver en budgetkategori:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
BudgetCategoryID	Genererad primärnyckel.	int	Ja	Nej	Nej
BudgetCategoryName	Namnet på budgetkategorin	nvarchar(50)	Nej	Nej	Nej

BudgetCategory_Account
Tabell som relaterar information om en budgetkategori med ett eller flera konton.

Följande attribut beskriver för budgetkategori- och kontotabellen:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
BudgetCategoryID	Budgetkategorins primärnyckel.	int	Ja	Ja	Nej
AccountID	Kontots primärnyckel	nchar(10)	Ja	Ja	Nej

Cost
Tabell som innehåller information om alla kostnader som hör samman med ett visst projekt samt det kostnadsställe som orsakat den.

Följande attribut beskriver en kostnad:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
CostID	Genererad primärnyckel.	int	Ja	Nej	Nej
CostDepartmentID	Kostnadsställe som kostnaden skall föras på.	int	Nej	Ja	Nej
Amount	Kostnadens storlek.	float	Nej	Nej	Nej
Date	Datumet då kostnaden uppstod.	smalldatetime	Nej	Nej	Nej
AccountID	Kontot kostnaden förs på.	nchar(10)	Nej	Ja	Nej

Cost_Project
Tabell som relaterar en kostnad till ett specifikt projekt. Därmed erhålls information om de kostnader ett visst projekt orsakat.

Följande attribut beskriver kostnads- och projkettabellen:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
CostID	Kostnadens primärnyckel.	int	Ja	Ja	Nej
ProjectID	Projektets primärnyckel.	int	Ja	Ja	Nej

Cost_Employee
Tabell som relaterar en kostnad till en specifik konsult. Därmed erhålls information om de kostnader en viss konsult orsakat.

Följande attribut beskriver kostnads- och konsulttabellen:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
CostID	Kostnadens primärnyckel.	int	Ja	Ja	Nej
Initials	Den anställdes primärnyckel.	char(3)	Ja	Ja	Nej

CostDepartment
Tabell som innehåller information om alla kostnadsställen. Ett kostnadsställe beskriver en avdelning inom Know IT i Göteborg AB.

Följande attribut beskriver ett kostnadsställe:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
CostDepartmentId	Generad primärnyckel.	int	Ja	Nej	Nej
Name	Kostnadsställets namn.	varchar (50)	Nej	Nej	Nej

Customer
Tabell som innehåller information om alla kunder.

Följande attribut beskriver en kund:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
CustomerID	Genererad primärnyckel.	int	Ja	Nej	Nej
CustomerName	Kundens namn	nvarchar (50)	Nej	Nej	Nej

CustomerCodes
Tabell som innehåller information om alla kundkoder. Kundkoderna används i affärssystemet för att identifiera en specifik kund.

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
CustomerCodeID	Genererad primärnyckel.	int	Ja	Nej	Nej
CustomerCode	Namnet på kundkoden	nchar(10)	Nej	Nej	Nej
CustomerID	Kundens Primärnyckel.	int	Ja	Ja	Nej

Employee
Tabell som innehåller information om alla anställda.

Följande attribut beskriver en anställd:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
Initials	Den anställdes initialer.	nchar (10)	Ja	Nej	Nej
FirstName	Den anställdes förnamn.	varchar (50)	Nej	Nej	Nej
LastName	Den anställdes efternamn.	varchar (50)	Nej	Nej	Nej

Employee_Adjustment
Tabell som relaterar information om en justering till en viss konsult. Summan av justeringen härleds sålunda till den konsult som orsakat den.

Följande attribut beskriver justerings- och anställdtabellen.

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
AdjustmentID	Justerings primärnyckel	int	Ja	Ja	Nej
Initials	Den anställdes primärnyckel.	char(3)	Ja	Ja	Nej
TimeCode	Tidkodens primärnyckel	nchar(10)	Ja	Ja	Nej

Employee_CostDepartment
Tabell som innehåller information om vilket kostnadsställe en konsult är anställd vid.

Följande attribut beskriver en anställning:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
Initials	Den anställdes initialer.	int	Ja	Ja	Nej
Name	Kostnadsställets namn.	varchar (50)	Ja	Ja	Nej
EmployementStartDate	Datum då anställningen påbörjades.	smallDateTime	Nej	Nej	Nej
EmployementEndDate	Datum då anställningen avslutades.	smallDateTime	Nej	Nej	Ja

Other_Codes
Tabell som innehåller information om alla övriga koder, det vill säga koder som inte utgör någon form av tidkod. Ett exempel på en sådan kod är ”3010 Inköp”.

Följande attribut beskriver en övrigkod:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
OtherCode	Kod som beskriver en övrigkod.	nchar (10)	Ja	Nej	Nej
OtherCodeName	Övrigkodens namn	varchar (50)	Nej	Nej	Nej

Project
Tabell som innehåller information om alla Know IT i Göteborg AB:s projekt.

Följande attribut beskriver ett projekt:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
ProjectID	Genererad primärnyckel.	int	Ja	Nej	Nej
ProjectName	Projektets namn	nvarchar(60)	Nej	Nej	Nej
ProjectCode	Projektets kod	nchar(10)	Nej	Nej	Nej
TypeOfPrice	Typ av pris för projektet.	char(1)	Nej	Nej	Nej
ProjectStartDate	Startdatum för projektet.	smalldatetime	Nej	Nej	Nej
ProjectEndDate	Slutdatum för projektet.	smalldatetime	Nej	Nej	Ja
Responsible	Ansvarig för projektet.	char(3)	Nej	Ja	Nej
CustomerID	Kunden projektet avser.	nchar(10)	Nej	Ja	Nej

ReportedTime
Tabell som innehåller information om hur och när en konsult lagt ner sin arbetade tid.

Följande attribut beskriver en konsults rapporterade tid:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
ReportedTimeId	Generad primärnyckel.	int	Ja	Nej	Nej
Initials	Den anställdes initialer.	nchar (10)	Nej	Ja	Nej
Date	Datum då arbetet utfördes.	smallDateTime	Nej	Nej	Nej
TimeAmount	Mängden arbetad tid.	float	Nej	Nej	Nej
Price	Timpriset för den arbetade tiden.	Float	Nej	Nej	Nej
TimeCode	Kod som beskriver en tidskod.	nchar (10)	Nej	Ja	Nej
ProjectID	Projektets primärnyckel.	int	Nej	Ja	Nej

Revenue
Tabell som innehåller information om alla intäkter som hör samman med ett visst projekt och kostnadsställe. Intäkter hör inte direkt samman med den tid en konsult lagt ner utan kan exempelvis vara intäkter från försäljning av licenser.

Följande attribut beskriver en intäkt:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
RevenueID	Genererad primärnyckel.	int	Ja	Ja	Nej
ProjectID	Projektets primärnyckel.	int	Nej	Ja	Nej
CostDepartmentID	Kostnadsställets primärnyckel.	int	Nej	Ja	Nej
Amount	Intäktens storlek.	float	Nej	Nej	Nej
Date	Datumet då intäkten uppstod.	smalldatetime	Nej	Nej	Nej
OtherCode	Övrigkodens primärnyckel.	nchar(10)	Nej	Ja	Nej
Type	Typ av intäkt	nchar(10)	Nej	Nej	Ja

Account
Tabell som innehåller information om alla konton. Här avses konton vilka återfinns i kontoplanen för Know IT i Göteborg AB.

Följande attribut beskriver ett konto:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
AccountCode	Kontots primärnyckel.	int	Ja	Nej	Nej
AccountDescription	Kontots namn	nvarchar(50)	Nej	Nej	Nej
ParrentAccountCode	Kontokoden för kontots föräldrakonto.	nchar(10)	Nej	Ja	Nej
AccountType	Kontots typ (Intäktskonto eller kostnadskonto)	char(1)	Nej	Nej	Nej
Operator	Anger vilken	char(1)	Nej	Nej	Nej

	operator som skall användas vid summeringar utav konton (exempelvis + eller -.)				
ValuType	Anger vilken värdetyp som skall anges vid summering i DW.	char(10)	Nej	Nej	Nej

Time_Codes
Tabell som innehåller information om alla tidkoder. En tidkod anger vad en konsult arbetat med under en viss tid, exempel på en tidkod är ”030 Konsult”, som innebär att konsulten lagt ner tid för en viss kund.

Följande attribut beskriver en tidkod:

Attribut	**Beskrivning**	**Datatyp**	**Primary key**	**Foreign key**	**Tillåt Null**
TimeCode	Kod som beskriver en tidskod.	nchar (10)	Ja	Nej	Nej
TimeCodeName	Tidkodens namn	varchar (50)	Nej	Nej	Nej

4.2.3 Datawarehouset

Av diskussionen ovan framgår att ett datawarehouse kan designas på olika sätt, det vill säga som ett starschema eller ett snowflakeschema. Det som skiljer dessa scheman åt är främst graden av normalisering, men som Powell (2006) beskriver innebär en högre grad av normalisering också att fler Join-operationer måste utföras mellan dimensions- och faktatabellerna. Av denna anledning har datawarehouset i BI-systemet implementerats som ett starschema för att på så sätt minimera påverkan på systemets prestanda då OLAP-kuberna skapas ifrån data i datawarehouset. Nackdelen med denna lösning är dock att det kan uppstå en viss grad av redundans, det vill säga att data som innehåller samma typ av information sparas flera gånger i datawarehouset. Denna företeelse återfinns främst i dimensionstabellerna. Det kan dock anses godtagbart eftersom de oftast innehåller ganska lite data och är statiska till sin natur, det vill säga de uppdateras sällan med ny data. Det gör att datamängden i datawarehouset inte påverkas nämnvärt av detta designval. Dock betyder det mycket för datawarehousets performance att antalet JOIN operationer minimeras då OLAP-kuberna beräknas.

Datawarehousets konstruktion
Datawarehouset består av dimensions- och faktatabellerna nedan.

DimAccount
Dimensionstabell som innehåller information om alla konton i datawarehouset. Kontona följer den kontoplan som Know IT i Göteborg använder för redovisning av det ekonomiska resultatet. Utifrån tabellen går det att utläsa all information om ett konto såsom kontokod och namnet på kontot. Vidare sammankopplar tabellen också alla konton i en hierarki där ett konto kan vara barn eller förälder till ett annat konto. Eftersom redovisning med konton bygger på att breda klasser av kontobenämningar, exempelvis kontot med koden 30 och namnet ”Intäkter”, kan förfinas med hjälp av underliggande klasser, exempelvis kontot med koden 3011 och namnet ”Intäkter konsult”, behöver tabellen DimAccount modellera detta förhållande för att det skall gå att analysera det ekonomiska resultatet på ett korrekt vis.

Sammankopplingen av barn- och föräldrakonton görs med hjälp av en foreignkey-relation mellan kolumnerna AccountKey och ParentAccountKey. Därmed återfinns för varje konto i tabellen en primärnyckel för kontot, AccountKey, och en primärnyckel för föräldrakontot, ParentAccountKey. Genom denna relation modelleras en hierarkisk struktur i tabellen. Noterbart är dock att Null värden tillåts för ParentAccountKey, vilket beror på att rotnoden, det vill säga det ekonomiska resultatet, inte har något föräldrakonto och därmed måste tillåtas vara Null i foreingkey-relationen.

Det ekonomiska resultatet utgörs i sin tur av alla intäkter minus alla kostnader. För att modellera detta förhållande behöver alla konton klassificeras antingen som en intäkt eller kostnad, vilket görs genom att kolumnen Operator för respektive konto ges värdet ”+” för en intäkt och ”-” för en kostnad. När en OLAP kub analyserar de registrerade transaktionerna kan den med hjälp av denna information avgöra om en transaktion skall hanteras som en intäkt eller som en kostnad.

Med hjälp av DimAccount tabellen är det enkelt att i en OLAP kub se resultatet för respektive konto när den relateras till faktatabellen som registrerar intäkter och kostnader.

Följande attribut beskriver ett konto i datawarehouset:

Attribut	**Beskrivning**	**Datatyp**	**Primary key**	**Foreign key**	**Tillåt Null**
AccountKey	Genererad primärnyckel.	int	Ja	Nej	Nej
ParentAccountKey	Primärnyckeln för kontots föräldrakonto.	int	Nej	Ja	Ja

AccountCodeAlternateKey	Kontokoden för kontot (ex 3011).	int	Nej	Nej	Ja
ParentAccountCodeAlternateKey	Kontokoden för föräldrakontot (ex 30).	int	Nej	Nej	Ja
AccountDescription	Namnet på kontot (ex Intäkter Konsult).	nvarchar (50)	Nej	Nej	Ja
AccountType	Kontotypen för kontot (anges som antingen Intäkt eller Kostnad).	nvarchar (50)	Nej	Nej	Ja
Operator	Anger vilken operator som skall användas vid summeringar utav konton (antingen + eller -.)	nchar (1)	Nej	Nej	Ja
ValueType	Värdetyp som skall användas av ett analysverktyg vid tolkning av kontosumman (anges som Currency för respektive konto).	nchar (10)	Nej	Nej	Ja

DimCostDepartment

Dimensionstabell som innehåller information om alla kostnadsställen i datawarehouset. Kostnadsställena utgör de olika resultatenheterna, det vill säga avdelningarna, inom Know IT i Göteborg AB. För dessa sammanställs i slutet av varje månad resultatet för verksamheten som bedrivits inom enheten samt för bolaget som helhet. Tabellen DimCostAccount måste därför innehålla information om respektive kostnadsställe och modellera relationerna mellan dem. Eftersom bolaget är den övergripande

kostnadsenheten behöver en hierarkisk relation modelleras där bolaget utgör rotnod medan övriga kostnadsställen är underordnade denna. I tabellen modelleras detta med en foreignkey-relation mellan kolumnerna ParentCostDepartmentKey, vilken innehåller primärnyckeln för kostnadsstället bolag, och CostDepartmentKey, vilken utgör primärnyckel för respektive kostnadsställe. Det är dock viktigt att notera att relationen inte på något sätt begränsas till att endast relatera kostnadsstället bolag till respektive underliggande kostnadsenhet. Skulle det exempelvis tillkomma ett kostnadsställe som ligger under ett av de övriga kostnadsställena relateras de på samma vis där det primärnyckeln för det överliggande kostnadsstället då anges i fältet ParentCostDepartmentKey för det nytillkomna kostnadsstället.

Med hjälp av denna dimensionstabell är det enkelt att i en OLAP kub se resultatet för respektive kostnadsenhet när den relateras till faktatabellen som registrerar intäkter och kostnader.

Följande attribut beskriver ett kostnadsställe i datawarehouset:

Attribut	**Beskrivning**	**Datatyp**	**Primary key**	**Foreign key**	**Tillåt Null**
CostDepartmentKey	Genererad primärnyckel.	int	Ja	Nej	Nej
ParentCostDepartmentKey	Primärnyckeln för kostndsställets föräldrakonto.	int	Nej	Ja	Ja
CostDepartmentName	Namnet på kostnadsstället (ex Bolag).	int	Nej	Nej	Ja

DimPerson
Dimensionstabell som innehåller information om alla personer i datawarehouset. Personerna utgörs främst av konsulter inom Know IT Göteborg AB, vilka alla genererar intäkter och orsakar kostnader, exempelvis i form av löner. Då alla konsulter hanteras utifrån deras initialer i affärssystemet är det den enda information denna tabell innehåller.

Med hjälp av denna dimensionstabell är det enkelt att i en OLAP kub se resultatet för respektive konsult när den relateras till faktatabellen som registrerar intäkter och kostnader.

Följande attribut beskriver en konsult i datawarehouset:

Attribut	**Beskrivning**	**Datatyp**	**Primary key**	**Foreign key**	**Tillåt Null**
PersonKey	Genererad primärnyckel.	int	Ja	Nej	Nej
Initials	Den anställdes	nvarchar	Nej	Nej	Ja

	initialer.	(50)			

DimTime
Dimensionstabell som innehåller information om alla datum i datawarehouset. Tabellen används för att ange datumet då en intäkt eller kostnad uppstod i datawarehouset. För att möjliggöra analyser av resultat, intäkter och kostnader över olika tidsperioder innehåller tabellen kolumner för exempelvis år, månad, vecka, dagens nummer inom veckan (exempelvis 1 för måndag), dagens nummer inom månaden och dagens nummer inom året (exempelvis 1 för den första januari).

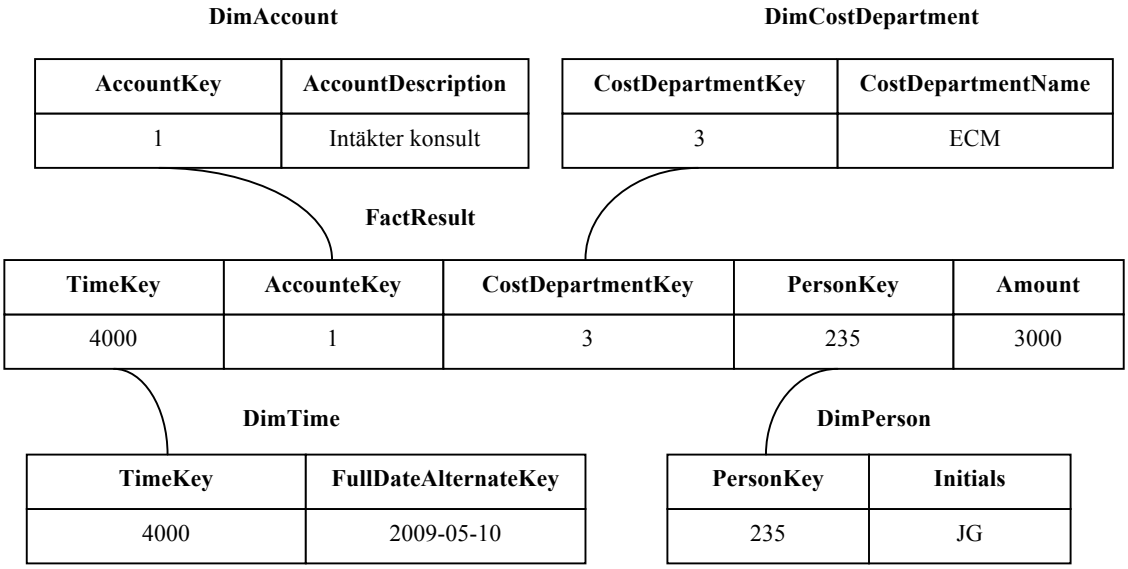
När ett analysverktyg gör analyser av de data som återfinns i en faktatabell och DimTime gör det summeringar utifrån värdena i de olika kolumnerna. Rader som exempelvis har värdet 2009 angett i kolumnen år kommer att summers tillsammans och därmed kan resultatet utläsas för detta år.

Med hjälp av denna information går det att i en OLAP kub se resultat, intäkter och kostnader ur olika tidsaspekter.

Följande attribut beskriver ett datum i datawarehouset:

Attribut	**Beskrivning**	**Datatyp**	**Primary key**	**Foreign key**	**Tillåt Null**
PersonKey	Genererad primärnyckel.	Int	Ja	Nej	Nej
Initials	Den anställdes initialer.	nvarchar (50)	Nej	Nej	Ja

FactResult
Faktatabell som innehåller information om det ekonomiska resultatet i datawarehouset. I denna tabell registreras alla intäkter och kostnader som kopplas samman med ett kostnadsställe, konto, datum och konsult. För varje rad i tabellen anges storleken på intäkten eller kostnaden och primärnycklarna för de rader i dimensionstabellerna DimAccount, DimCostDepartment, DimPerson och DimTime som transaktionen skall registreras för. Relationen mellan FactResult och respektive dimensionstabeller utgörs av en foreignkey-relation. Av Figur 13 framgår ett exempel på hur en intäkt på 3000 SEK, som uppstod 2009-05-10, har registrerats i tabellen FactResult för konsulten ”JG”, kostnadsställe ”ECM” och kontot ”3011 Intäkter konsult”.



Figur 13 - Exempel på relationen mellan FactResult och dimesionstabellerna DimTime, DimAccount, DimCostDepartment och DimPersion.

Följande attribut beskriver en intäkt eller kostnad som registrerats i datawarehouset:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
TimeKey	Datumets primärnyckel.	int	Nej	Ja	Nej
AccountKey	Kontots primärnyckel.	int	Nej	Ja	Nej
PersonKey	Personens primärnyckel.	int	Nej	Ja	Nej
CostDepartmentKey	Kostnadsställets primärnyckel.	int	Nej	Ja	Nej
Amount	Intäkten eller kostnadens storlek	float	Nej	Nej	Nej

Uppbyggnaden av tabellen FactResult ger möjlighet till flerdimensionell analys av Knowit i Göteborgs ekonomiska resultat där det är möjligt att analysera intäkter, kostnader och resultat ur olika perspektiv. Möjliga analyser är exempelvis:

- Det ekonomiska resultatet för en konsult en viss månad.
- Det ekonomiska resultatet för alla konsulter en viss vecka.
- Det ekonomiska resultatet, intäkter och kostnader för ett kostnadsställe ett visst år.
- Det ekonomiska resultatet för bolaget som helhet under en dag, vecka, månad eller år.

Användarna ges på det här sättet möjligheter att själva analysera data ur de perspektiv de är intresserade av. Denna möjlighet ges när en OLAP kub genererats utifrån de data som återfinns i faktatabellen och tillhörande dimensionstabeller.

4.3 Design av datatransformering

Den implementerade lösningen för att föra över data ifrån stagedatabasen till produktionsdatabasen och sedan vidare ifrån denna till datawarehouset bygger på ett ETL verktyg. Valet att basera lösningen på detta verktyg bygger på att det är optimerat för hantering av datatransformeringar i minnet, vilket reducerar antalet I/O operationer, samt att det är enkelt att förändra och bygga ut om nya krav tillkommer i framtiden. Lösningen bygger också i vissa delar på parallellism, det vill säga att vissa delar utav dataflödet sker parallellt, vilket ETL verktyget också ger stöd för.

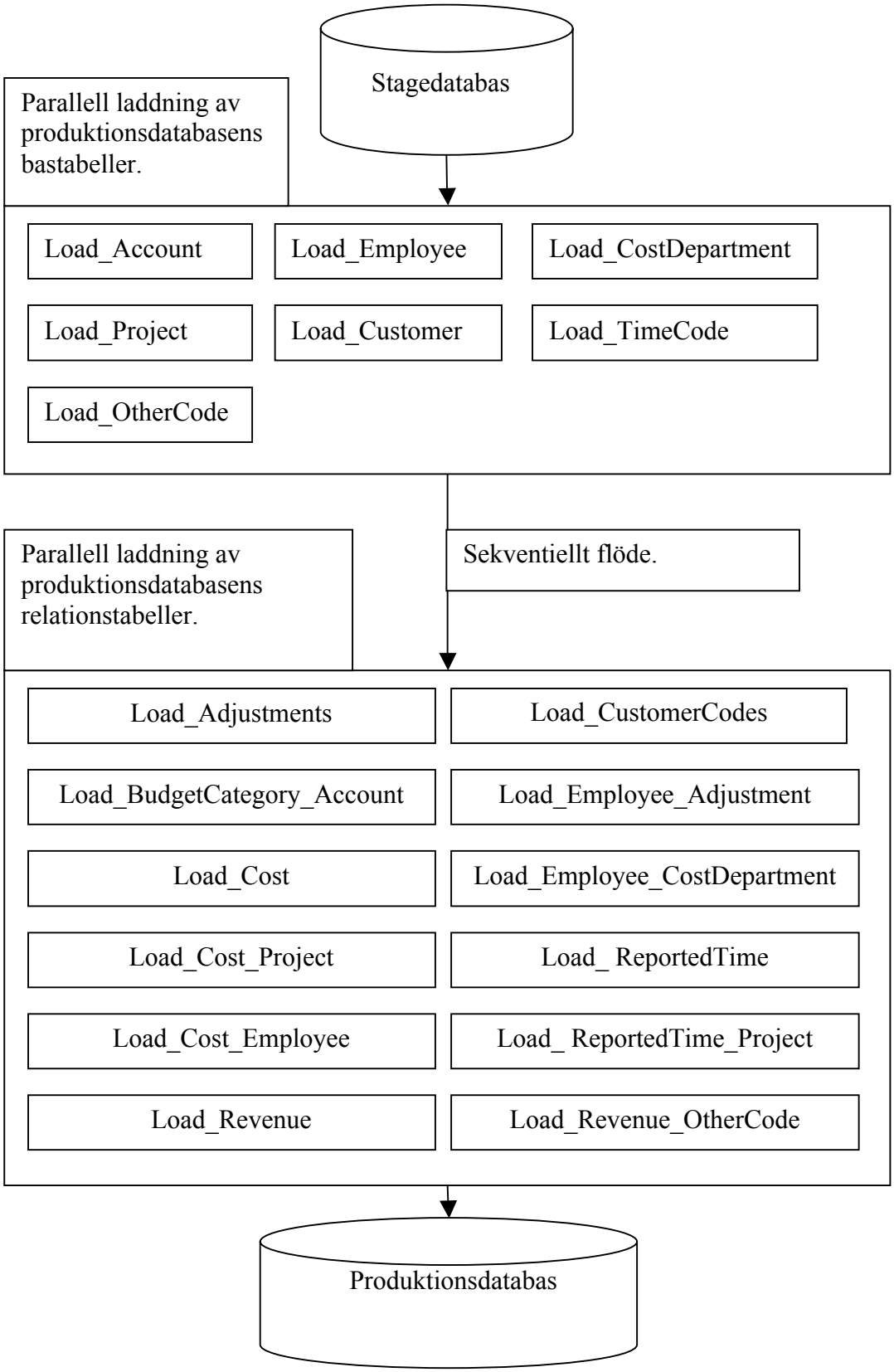
Av analysen i stycke ”3.6 Analys av datatransformering” framgår av datatransformeringslösningen bör beakta två mål:

1. Att data hämtat ifrån affärssystemet är konsistent med avseende på fullständighet, det vill säga att inga data gått förlorad, samt integritet, det vill säga att inga data av misstag modifierats under datautbytet.

2. Optimera dataladdningsprocessen för att minimera den tid dataladdningsprocessen tar att genomföra.

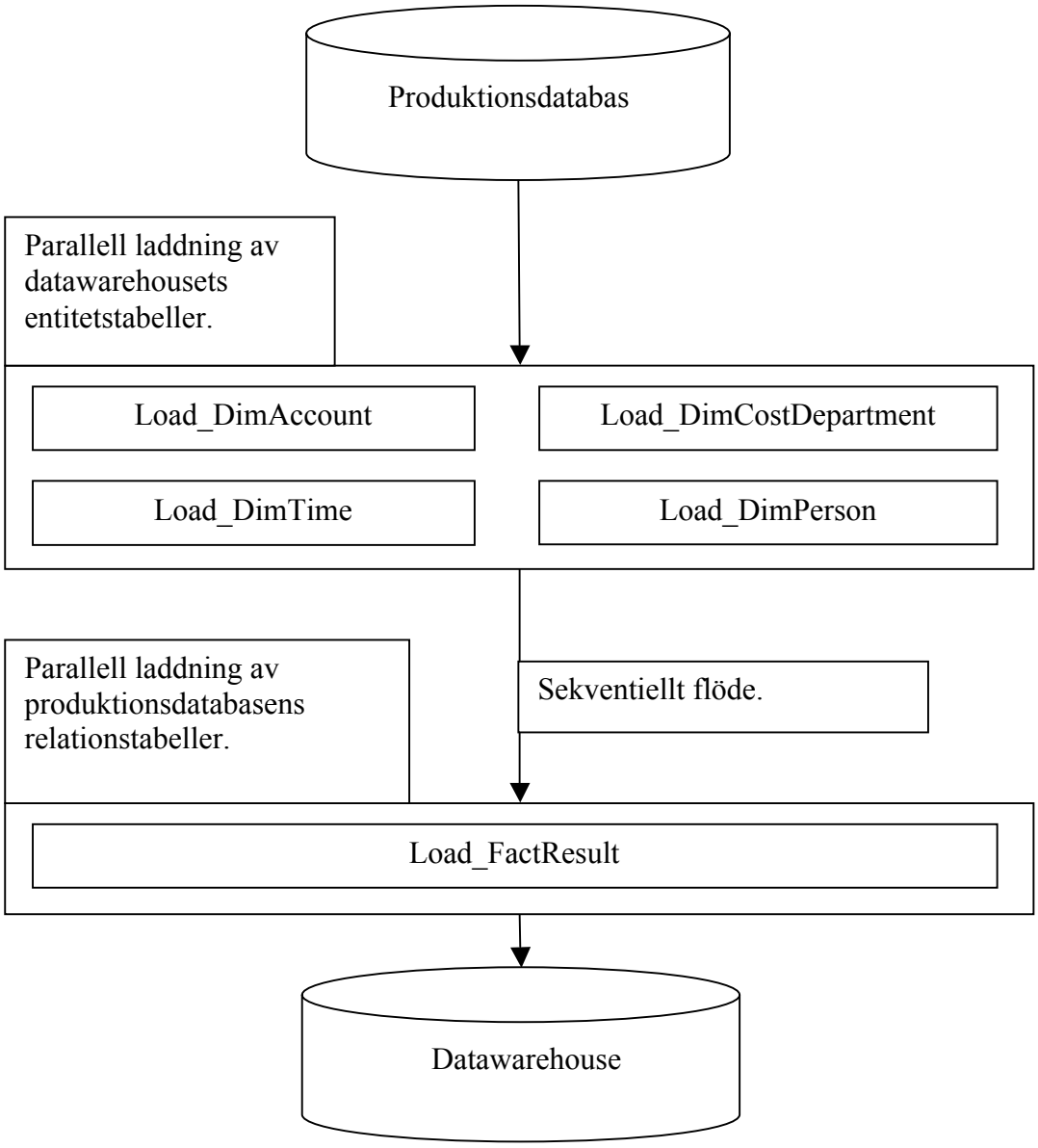
Den valda lösningen syftar till att svara upp mot mål 1 genom att inte ladda in data i vare sig produktionsdatabasen eller datawarehouset om data strider mot relationerna, som utgörs av foreign-keys, i dessa databaser. Om ett sådant tillstånd skulle uppstå avbryts dataladdningsprocessen då ett tillstånd av inkonsistens eller ofullständighet, det vill säga data har förändrats eller saknas, kan antas ha uppstått.

För att uppnå mål 2 utförs flera utav stegen i respektive dataflöde parallellt, vilket innebär att dataladdningstiden kan förkortas tillskillnad mot om detta hade skett helt sekventiellt. I lösningen innebär det att alla entitetstabeller kan laddas parallellt och sedan kan alla relationstabeller laddas parallellt på samma sätt. Eftersom relationstabellerna är beroende av att data i entitetstabellerna kan dataladdningen dock inte helt parallelliseras utan ett sekventiellt flöde uppstår mellan laddningen av entitetstabellerna och relationstabellerna. Av Figur 14 framgår en schematisk bild av dataladdningsprocessen som laddar produktionsdatabasen med data ifrån Stagedatabasen.



Figur 14 - Schematisk bild av dataladdningsprocessen från Stage databasen till produktionsdatabasen.

Laddningen av datawarehouset med data ifrån produktionsdatabasen sker enligt samma sätt där entitetstabellerna, i detta fall dimensionstabellerna, laddas först och sedan relationstabellerna, i detta fall faktatabellerna. Dimensionstabellerna kan laddas parallellt eftersom de inte är beroende av varandra. Därefter kan faktatabellerna laddas eftersom de ingår i foreign-key relationer med dimensionstabellerna och sålunda är beroende av att data laddats in i dessa först. Ett sekventiellt flöde uppstår sålunda mellan laddningen av entitetsbastabellerna och relationstabellerna. Av Figur 15 framgår en schematisk bild av dataladdningsprocessen som laddar datawarehouset med data ifrån produktionsdatabasen.



Figur 15 - Schematisk bild av dataladdningsprocessen från produktionsdatabasen till datawarehouset.

4.4 Diskussion

Arbetet som ligger till grund för denna rapport är nu färdigt. Författarens förståelse för den problematik som omgärdar arbetet med att modellera och implementera ett BI-system för analys av ekonomiska data är förhoppningsvis större nu än när arbetet påbörjades. Kunskaper som erhållits är bland annat att det inte är helt trivialt hur dataladdningsprocessen och datalagret skall designas för att nå den funktion som systemet skall tillhandahålla. Designvalen som står till buds har inte varit helt klara ifrån arbetets början utan har vuxit fram under dess genomförande. Det i sin tur medför att vissa val kunde ha gjorts annorlunda om denna kunskap redan fanns på plats då arbetet påbörjades.

Ett sådant designval skulle vara att använda sig utav webbtjänster istället för den valda lösningen baserad på https protokollet för datautbytet mellan BI-systemet och affärssystemet. Som framgår ovan begränsades dock detta val av att leverantören till affärssystemet inte tillhandahöll några webbtjänster vilket gjorde en sådan lösning omöjlig. Användandet av webbtjänster hade dock haft potential att underlätta implementeringen utav BI-systemet då många högnivåspråk gör sådana metodanrop transparenta, det vill säga de beter sig som vanlig metodanrop.

Ett annat designval som troligen skulle kunna lösas på andra sätt än den valda är introduktionen av produktionsdatabasen. I BI-systemet fungerar denna som en spegling utav affärssystemet vars mål är att de data som återfinns här skall vara konsistenta och fullständiga. Detta mål borde dock gå att nå genom att tillämpa andra designval. En möjlighet skulle exempelvis kunna vara att bygga in mer logik i dataladdningsprocessen, vilket gör att samma resultat kan uppnås, samtidigt som produktionsdatabasen kan elimineras. Vinsten med detta tillvägagångssätt är att prestandan för dataladdningsprocessen troligen skulle kunna förbättras än mer samtidigt som mindre datalagringsutrymme behöver tas i anspråk av BI-systemet.

5 Rekommendationer

Av denna rapport framgår hur datalagret för ett BI-system fokuserat på analys av ekonomiska data kan modelleras och implementeras. För fortsatta studier av ett sådant system anser författaren dock att det skulle vara intressant att undersöka vidare om:

1. Designen av datautbytesprocessen med kringliggande system kan förenklas genom användande av webbtjänster eller någon annan teknik. Det i kontrats mot utbyte av XML-strömmar över https protokollet som varit föremål för denna rapport.
2. En stagearea är nödvändig eller om denna kan elimineras i sin helhet då en lösning av denna typ av system modelleras och implementeras.
3. Stagearean ej kan elimineras skulle det dock vara intressant att undersöka hur den skulle kunna modelleras och implementeras med hjälp av filer eller någon annan teknik för denna typ av system samt vilken påverkan det har dataladdningsprocessens prestanda.

6 Referenser

K

Kimball, R., Caserta, J., *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. John Wiley and Sons, USA

Know IT i Göteborg AB, 2010 [Elektronisk]. Tillgänglig; http://www.knowit.se/Om-knowit/Vara_kontor/Goteborg/Know-IT-Goteborg-AB/ [läst: 2010-03-28]

M

Mundey, J., Thornthwaite W., Kimball R., (2006). *The Microsoft Data Warehouse Toolkit*. Wiley Publishing Inc, USA

P

Powell, G., (2006). *Beginning Database Design and Implementation*. Wrox Press, USA

R

Raisinghani, M., (2004). *Business Intelligence in the Digital Economy: Opportunities, Limitations, and Risk*. Ide Group Publishing, USA

T

Troelsen A., (2007). *Pro C# with .NET 3.0, Special Edition*. Apress, USA

W

Wang, J., (2006). *Computation of OLAP Cubes - Encyclopedia of Data Warehousing and Mining*. Volume I, A-H, Idea Group Reference, USA

Wrembel, R., Koncilia, C., (2007). *Data Warehouses and OLAP: Concepts, Architectures and Solutions*. IRM Press, USA

7 Bilagor

7.1 Bilaga 1 - Xmlströmmar för datautbyte mellan BI - och affärsystemet

7.1.1 XML innehållande information om anställda

Beskrivning:
Denna XML innehåller alla anställda på Knowit i Göteborg AB. För varje anställd skall den anställdes **initialer**, **förnamn** och **efternamn** finnas med i XML strömmen.

XML definition:

- **employee:** Element som utgör en person.
- **initials:** Element som innehåller en persons initialer.
- **firstname:** Element som innehåller förnamnet på en anställd.
- **lastname:** Detta element ska innehålla efternamnet på en anställd.

XML struktur:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<businesssystem type="timesheet_get_response">
<status>OK</status>
    <employee>
          <initials>Initialier</initials>
          <firstname>Förnamn</firstname>
          <lastname>Efternamn</lastname>
       </employee>
     <employee>
          <initials>Initialer</initials>
          <firstname>Förnamn</firstname>
          <lastname>Efternamn</lastname>
       </employee>
  </businesssystem>
```

7.1.2 XML innehållande alla kunder

Beskrivning:
Denna XML ström innehåller alla Know IT i Göteborg AB:s kunder. Varje element i XML strömmen innehåller kundens **kod** samt **namn.**

XML definition:

- **customer:** Element som utgör en kund.

- **customer_id:** Element som innehåller en kundkod.
- **customer_name:** Element som innehåller kundens namn.

XML struktur:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<businesssystem type="timesheet_get_response">
   <status>OK</status>
   <customer>
       <customer_id>Kund id</customer_id>
       <customer_name>Kundnamn</customer_name>
   </customer>
  <customer>
       <customer_id>Kund id</customer_id>
       <customer_name>Kundnamn</customer_name>
   </customer>
 </businesssystem>
```

7.1.3 XML innehållande information om alla projekt som tillhör en specifik kund

Beskrivning:
Denna XML innehåller information om alla projekt som tillhör en kund. XML strömmen innehåller endast en kund åt gången med alla dennes tillhörande projekt.

För en specifik **kund** innehåller XML strömmen **kundens id**, **kundens namn** samt **initialerna** på den person som är ansvarig för kunden ifråga.

För ett projekt ska XML strömmen innehålla projektets **id/kod,** projektets **start –och slut datum, initialerna** för den person som är ansvarig för projektet samt om det för projektet är satt ett **fast pris** eller om projektet **debiteras löpande**.

XML definition:

- **customer_id:** Element som innehåller kundens ID.
- **customer_name:** Element som innehåller kundens namn.
- **customer_responsible:** Element som innehåller namnet på den person som är ansvarig för en kund
- **project:** Element som utgör ett projekt.
- **project_name:** Element som innehåller projektets namn.
- **project_id:** Element som innehåller ett projekts id.
- **project_startdate:** Element som innehåller datumet då projektet startade.
- **project_stopdate:** Element som innehåller datumet då projektet slutade. Är projektet pågående, d.v.s ej avslutat, innehåller detta element inga data.
- **project_responsible:** Element som innehåller namnet på den person som är ansvarig för ett projekt.
- **type:** Element som innehåller information ifall det för projektet är satt ett fast pris eller om det debiteras löpande.

XML struktur:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<businesssystem type="timesheet_get_response">
  <status>OK</status>
  <customer_id>Kund id</customer_id>
  <customer_name>Kundnamn</customer_name>
  <customer_responsible>Initialer</customer_responsible>
  <project>
      <project_id>Projekt id</project_id>
      <project_name>Projektnamn</project_name>
      <project_startdate>Startdatum(yyyy-mm-dd)</project_startdate>
      <project_stopdate>Slutdatum(yyyy-mm-dd)</project_stopdate>
      <project_responsible>Initialer</project_responsible>
      <types>
          <type>Typ av pris</type>
      </types>
  </project>
  <project>
      <project_id>Projekt id</project_id>
      <project_name>Projektnamn</project_name>
      <project_startdate>Startdatum(yyyy-mm-dd)</project_startdate>
      <project_stopdate>Slutdatum(yyyy-mm-dd)</project_stopdate>
      <project_responsible>Initialer</project_responsible>
      <types>
          <type>Typ av pris</type>
      </types>
  </project>
</businesssystem>
```

7.1.4 XML innehållande alla kostnadsställen

Beskrivning:
Denna XML innehåller information om alla kostnadsställen. XML strömmen innehåller alla kostnadsställen och alla som är anställda på respektive kostnadsställe.

För ett kostnadsställe innehåller XML strömmen **namnet** på kostnadsstället.

För de anställda vid ett kostnadsställe skall XML strömmen innehålla **initialerna** för en person, **datumet** då en person **började** arbeta på kostnadsstället och **datumet** då en person **slutade** arbeta på kostnadsstället. Elementet som innehåller datumet då en person slutade arbeta på ett kostnadsställe innehåller ingen information om personen fortfarande arbetar, d.v.s ej slutat arbeta, på kostnadsstället.

XML definition:

- **costdepartment:** Element som utgör ett kostnadsställe.
- **costdepartment_name:** Element som innehåller namnet på kostnadsstället.

- **employee:** Element som utgör en anställd.
- **initials:** Element som innehåller initialerna på en anställd.
- **startdate:** Element som innehåller datumet då en anställd började på ett visst kostnadsställe.
- **stopdate:** Element som innehåller datumet då en person slutade på ett visst kostnadsställe. Är personen fortfarande anställd innehåller detta element inga data.

XML struktur:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<businesssystem type="timesheet_get_response">
<status>OK</status>
   <costdepartment>
    <costdepartment_name>Kostnadsställe</costdepartment_name>
        <employee>
            <initials>Initialer</initials>
            <startdate>Startdatum(yyyy-mm-dd)</startdate>
             <stopdate>Slutdatum(yyyy-mm-dd)</stopdate>
        </employee>
        <employee>
            <initials>Initialer</initials>
            <startdate>Startdatum(yyyy-mm-dd)</startdate>
            <stopdate>Slutdatum(yyyy-mm-dd)</stopdate>
        </employee>
   </costdepartment>
   <costdepartment>
     <costdepartment_name>Kostnadsställe</costdepartment_name>
        <employee>
            <initials>Initialer</initials>
            <startdate>Startdatum(yyyy-mm-dd)</startdate>
            <stopdate>Slutdatum(yyyy-mm-dd)</stopdate>
        </employee>
        <employee>
            <initials>Initialer</initials>
            <startdate>Startdatum(yyyy-mm-dd)</startdate>
            <stopdate>Slutdatum(yyyy-mm-dd)</stopdate>
        </employee>
   </costdepartment>
  </businesssystem>
```

7.1.5 XML innehållande information om alla kostnader

Beskrivning:
Denna XML innehåller information om alla kostnadsposter som registrerats i affärssystemet.

För varje kostnadspost finns **namnet** på det **kostnadsställe** som kostnaden registrerats på med. Vidare ingår kostnadens **storlek, datumet** då kostnaden uppstod, och det **konto** i kontoplanen som kostnaden registrerats i XML strömmen. I de fall kostnaden kan

hänföras, d.v.s är registrerad för en specifik person/anställd skall dennes initialer också finnas med.

XML definition:

- **cost:** Element som utgör en kostnad.
- **costdepartment_id:** Element som innehåller namnet på ett kostnadsställe.
- **amount:** Element som innehåller kostnadens storlek.
- **date:** Element som innehåller det datum då kostnaden uppstod.
- **account_id:** Element som innehåller kontonumret på det konto som kostnaden registrerats på, ex kontonummer 7010.
- **project_id:** Element som innehåller ett projekts id. Ifall kostnaden inte kan härledas till ett specifikt projekt innehåller detta element ingen information.
- **initials:** Element som innehåller initialerna på den person som orsakat kostnaden. Ifall kostnaden inte kan härledas till en specifik person innehåller detta element ingen information.

XML struktur:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<businesssystem type="timesheet_get_response">
 <status>OK</status>
     <cost>
          <costdepartment_id>Kostnadsställe</costdepartment_id>
          <amount>Kostnad</amount>
          <date>Datum(yyyy-mm-dd)</date>
          <account_id>Kontonummer</account_id>
          <project_id>Projekt id</project_id>
          <initials>Initialer</initials>
      </cost>
     <cost>
           <costdepartment_id>Kostnadsställe</costdepartment_id>
          <amount>Kostnad</amount>
          <date>Datum(yyyy-mm-dd)</date>
          <account_id>Kontonummer </account_id>
          <project_id >Projekt id<project_id />
          <initials>Initialer</initials>
      </cost>
 </businesssystem>
```

7.1.6 XML innehållande alla konton i affärssystemet

Beskrivning:
Denna XML innehåller alla konton i Knowit:s kontoplan som finns angivna i affärssystemet. För ett konto innehåller XML strömmen kontots **kontonummer** samt kontots **benämning/namn**.

XML definition:

- **account:** Element som utgör ett konto.

- **account_id:** Element som innehåller ett kontonummer.
- **account_name:** Element som innehåller namnet på kontot.

XML struktur:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<businesssystem type="timesheet_get_response">
<status>OK</status>
    <account>
        <account_id>Kontonummer</account_id>
        <account_name>Kontonamn</account_name>
    </account>
    <account>
        <account_id>Kontonummer</account_id>
        <account_name>Kontonamn</account_name>
    </account>
</businesssystem>
```

7.1.7 XML innehållande alla rapporterade timmar

Beskrivning:
Denna XML innehåller all tid som rapporterats av KnowIT:s anställda.

Denna XML ström innehåller, för varje element, **initialerna** för den anställde som rapporterat tid, **datumet** som tidsrapporteringen avser, **antalet** timmar som rapporterats, **pris** per timma, den arvode kod som den rapporterade tiden registrerats på och projektkoden för det projekt som den anställde rapporterat tiden för

XML definition:

- **initials:** Element som innehåller initialerna på den anställd som den rapporterade tiden avser.
- **date:** Element som innehåller datumet som tiden rapporterats för.
- **time_amount:** Element som innehåller det antal timmar som rapporterats.
- **price:** Element som innehåller timpriset för den rapporterade tiden.
- **time_code:** Element som innehåller den arvode kod som den rapporterade tiden registrerats på.
- **project_id:** Element som innehåller projektkoden för det projekt som den anställde rapporterat tid för.

XML struktur:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<businesssystem type="timesheet_get_response">
<status>OK</status>
    <reported_time>
        <initials>Initialer</initials>
        <date>Datum(yyyy-mm-dd)</date>
        <time_amount>Antal rapporterad timmar</ time_amount>
        <price>Timpris</price>
```

```
            <time_code>Arvodekod</ time_code>
            <project_id>Projekt id</project_id>
        </reported_time>
        <reported_time>
            <initials>Initialer</initials>
            <date>Datum(yyyy-mm-dd)</date>
            <time_amount>Antal rapporterade timmar</ time_amount>
            <price>Timpris</price>
            <time_code>Arvodekod</ time_code>
            <project_id>Projekt id</project_id>
        </reported_time>
</businesssystem>
```

7.1.8 XML innehållande alla Arvode koder

Beskrivning:
Denna XML innehåller alla de arvode koder vilka KnowIT:s anställda anger då de rapporterar tid.

För varje arvode kod ska XML strömmen innehålla **arvode koden** samt **benämningen/namnet** för arvode koden.

XML definition:

- **time_code:** Element som utgör ett element för en arvode kod.
- **code:** Element som innehåller en arvode kod.
- **name:** Element som innehåller en arvode kods benämning.

XML struktur:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<businesssystem type="timesheet_get_response">
<status>OK</status>
    <time_code>
        <code>Arvodekod</code>
        <name>Arvodekodnamn</name>
    </time_code>
    <time_code>
        <code>Arvodekod</code>
        <name>Arvodekodnamn</name>
    </time_code>
    <time_code>
        <code>Arvodekod</code>
        <name>Arvodekodnamn</name>
    </time_code>
</businesssystem>
```

7.1.9 XML innehållande alla Övrigt koder

Beskrivning:
Denna XML innehåller alla övriga koder vilka Know IT i Göteborg AB:s anställda anger då de rapporterar tid. Utöver de övrig koder som finns innehåller denna XML ström

också de koder som återfinns under kategorin Inköp för varje projekt, ex ”200 Övriga inköp”.

För varje övrig kod innehåller XML strömmen **övrigt koden** samt **benämningen/namnet** för övrigt koden.

XML definition:

- **other_code:** Element som utgör ett element för en övrig kod/inköp kod.
- **code:** Element som innehåller en övrig kod/inköp kod.
- **name:** Element som innehåller en övrig kods/inköps kod benämning.

XML struktur:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<businesssystem type="timesheet_get_response">
<status>OK</status>
    <other_code>
        <code>Övrigkod</code>
        <name>Övrigkodnamn</name>
    </other_code>
    <other_code>
        <code>Övrigkod</code>
        <name>Övrigkodnamn</name>
    </other_code>
    <other_code>
        <code>Övrigkod</code>
        <name>Övrigkodnamn</name>
    </other_code>
</businesssystem>
```

7.1.10 XML innehållande alla justeringar

Beskrivning:
Denna XML innehåller alla justeringar som gjorts för ett projekt med avseende på Arvode (rapporterade tid), Inköp och Övrigt.

För varje justering innehåller XML strömmen s**torleken** på justeringen, vilket **datum** justeringen gjordes, **projektkoden** för det projekt som justeringen gjorts för, vilken **kategori** (Arvode, Inköp eller Övrigt) som justering avser, **intialerna** för den anställde som justeringen gjorts för (detta fält lämnas tomt om justeringen inte avser en justering för Arvode) och den **kod** (kod/övrig kod) som justeringen gjorts för.

XML definition:

- **adjustment:** Element som utgör ett element för en justering.
- **amount:** Element som innehåller juseteringens storlek.
- **date:** Element som innehåller datumet då justeringen gjordes.
- **project_id:** Element som innehåller projekt koden för det projekt justeringen hänförs till.

- **type:** Element som innehåller vilken utav kategorierna Arvode, Inköp och Övrigt som justreingen gjorts för.
- **initials:** Element som innehåller initialerna för den person som en justering kan hänföras till. Om elementet type inte har värdet Arvode innehåller elementet initials ingenting, d.v.s det är tomt.
- **code:** Element som innehåller den kod, ex 030 Konsult eller 801 Övertidsersättning100%, för vilken justeringen gjorts.

XML struktur:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<businesssystem type="timesheet_get_response">
<status>OK</status>
    <adjustment>
        <amount>Justering</amount>
        <date>Datum(yyyy-mm-dd)</date>
        <project_id>Projekt id</project_id>
        <type>Typ av kategori</type>
        <initials>Initialer</initials>
        <code>Justeringskod</code>
    </adjustment>
    <adjustment>
        <amount>Justering</amount>
        <date>Datum(yyyy-mm-dd)</date>
        <project_id>Projekt id</project_id>
        <type>Typ av kategori</type>
        <initials>Initialer</initials>
        <code>Justeringskod</code>
    </adjustment>
</businesssystem>
```

7.1.11 XML innehållande alla övriga intäkter

Beskrivning:
Denna XML innehåller alla övriga intäkter, d.v.s sådana intäkter som inte är av typen Arvode. Dessa intäkter kan hänföras till ett specifikt projekt samt kostnadsställe och finns inrapporterade i affärssystemet under Inköp och Övrigt för varje projekt.

För varje intäkt innehåller XML strömmen den **projekt kod** och **kostandsställe** som intäkten kan hänföras till. Den innehåller också intäktens **storlek**, **datumet** då intäkten uppstod samt det **konto** i kontoplanen som intäkten registrerats på. Vidare innehåller XML strömmen också den **kategori/typ** som intäkten är utav, d.v.s den kategori av Övrigt eller Inköp som intäkten är registrerad på för det specifika projektet samt vilken **kod** (Övrig kod eller Inköp kod) den har.

XML definition:

- **revenue:** Element som utgör ett element för en intäkt.
- **date:** Element som innehåller datumet då intäkten uppstod.
- **project_id:** Element som innehåller projekt koden för det projekt intäkten hänförs till.
- **account_id:** Element som innehåller kontonumret på det konto som intäkten registrerats på, ex kontonummer 4010.
- **costdepartment:** Element som innehåller namnet på det kostnadsställe som intäkten kan hänföras till.
- **type:** Element som innehåller vilken utav kategorierna Inköp och Övrigt som justreingen gjorts för.
- **code:** Element som innehåller den kod, ex 801 Övertidsersättning eller 200 Inköp, för vilken intäkten gjorts.

XML struktur:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<businesssystem type="timesheet_get_response">
<status>OK</status>
    <revenue>
        <amount>Intäkt</amount>
        <date>Datum(yyyy-mm-dd)</date>
        <project_id>Projekt id</project_id>
        <account>Kontonummer</account>
        <costdepartment>Kostnadsställe</costdepartment>
        <type>Typ av kategori</type>
        <code>Intäktskod</code>
    </revenue>
    <revenue>
        <amount>Intäkt</amount>
        <date>Datum(yyyy-mm-dd)</</date>
        <project_id>Projekt id</project_id>
        <account> Kontonummer </account>
        <costdepartment>Kostnadsställe</costdepartment>
```

```
            <type>Typ av kategori</type>
            <code>Intäktskod</code>
        </revenue >
</businesssystem>
```

7.2 Bilaga 2 – Stagedatabasens uppbyggnad

Account
Tabell som innehåller information om alla konton. Här avses konton vilka återfinns i kontoplanen för Know IT i Göteborg AB.

Följande attribut beskriver ett konto:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
AccountNo	Kontots kontonummer.	nchar(10)	Nej	Nej	Ja
AccountName	Kontots namn	nvarchar(50)	Nej	Nej	Ja
ParrentAccountCode	Kontokoden för kontots föräldrakonto.	nvarchar(50)	Nej	Nej	Ja
AccountType	Kontots typ (Intäktskonto eller kostnadskonto)	char(1)	Nej	Nej	Ja

Adjustments
Tabell som innehåller information om alla justeringar. En justering anger om det skett en uppskrivning eller nedskrivning för ett projekt samt för den konsult som orsakat justeringen. En justering anges i det monetära värdet av upp- eller nedskrivningen.

Följande attribut beskriver en justering.

Attribut	**Beskrivning**	**Datatyp**	**Primary key**	**Foreign key**	**Tillåt Null**
Initials	Initialerna för personen som justeringen hänförs till.	char(3)	Nej	Nej	Ja
ProjectCode	Projektkoden för det projekt justeringen hänförs till.	nchar(10)	Nej	Nej	Ja
Code	Tid eller övrig koden för justeringen.	nchar(10)	Nej	Nej	Ja
Amount	Storleken på justeringen	float	Nej	Nej	Ja
Date	Datumet då justeringen gjordes.	smalldatetime	Nej	Nej	Ja
Type	Typ av justering, d.v.s Övrigt, Arvode eller Inköp.	nchar(10)	Nej	Nej	Ja

Cost
Tabell som innehåller information om alla kostnader som hör samman med ett visst projekt samt den konsult som orsakat den.

Följande attribut beskriver en kostnad:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
ProjectCode	Projektkoden som kostnaden skall föras på.	nchar(10)	Nej	Nej	Ja
CostDepartmentName	Kostnadsställe som kostnaden skall föras på.	nchar(50)	Nej	Nej	Ja
Amount	Kostnadens storlek.	float	Nej	Nej	Ja
Date	Datumet då kostnaden uppstod.	smalldatetime	Nej	Nej	Ja
AccountNo	Kontot kostnaden förs på.	nchar(10)	Nej	Nej	Ja
Initials	Initialerna för konsulten som kostnaden skall föras på.	char(3)	Nej	Nej	Ja

CostDepartment
Tabell som innehåller information om alla kostnadsställen. Ett kostnadsställe beskriver en avdelning inom Know IT i Göteborg AB.

Följande attribut beskriver ett kostnadsställe:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
Name	Kostnadsställets namn.	varchar (50)	Nej	Nej	Ja

Cost_Employee
Tabell som relaterar en kostnad till en specifik konsult. Därmed erhålls information vilka kostnader en viss konsult orsakat.

Följande attribut beskriver kostnads- och konsulttabellen:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
CostID	Kostnadens primärnyckel.	int	Ja	Ja	Nej
Initials	Den anställdes primärnyckel.	char(3)	Ja	Ja	Nej

Customer
Tabell som innehåller information om alla kunder. Varje kund beskrivs med dess kundkod samt namn.

Följande attribut beskriver en kund:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
CustomerCode	Namnet på kundkoden	nchar(10)	Nej	Nej	Ja
CustomerName	Kundens namn	nvarchar(50)	Nej	Nej	Ja

Employee
Tabell som innehåller information om alla anställda. En anställd beskrivs med hjälp av dess initialare samt förnamn och efternamn.

Följande attribut beskriver en anställd:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
Initials	Den anställdes initialer.	char (3)	Nej	Nej	Ja
FirstName	Den anställdes förnamn.	varchar (50)	Nej	Nej	Ja
LastName	Den anställdes efternamn.	varchar (50)	Nej	Nej	Ja

Employee_CostDepartment
Tabell som innehåller information om vilket kostnadsställe en konsult är anställd vid.

Följande attribut beskriver en anställning:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
Initials	Den anställdes initialer.	char (3)	Nej	Nej	Ja
CostDepartmentName	Kostnadsställets namn.	varchar (50)	Nej	Nej	Ja
EmployementStartDate	Datum då anställningen påbörjades.	smallDateTime	Nej	Nej	Ja
EmployementEndDate	Datum då anställningen avslutades.	smallDateTime	Nej	Nej	Ja

Other_Codes
Tabell som innehåller information om alla övriga koder, det vill säga koder som inte utgör någon form av tidkod. Ett exempel på en sådan kod är ”3010 Inköp”.

Följande attribut beskriver en övrigkod:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
OtherCode	Kod som beskriver en övrigkod.	nchar (10)	Nej	Nej	Ja
OtherCodeName	Övrigkodens namn	varchar (50)	Nej	Nej	Ja

Project
Tabell som innehåller information om alla Know IT i Göteborg AB.:s projekt.

Följande attribut beskriver ett projekt:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
ProjectName	Projektets namn	nvarchar(60)	Nej	Nej	Ja
ProjectCode	Projektets kod	nchar(10)	Nej	Nej	Ja
TypeOfPrice	Typ av pris för projektet.	char(1)	Nej	Nej	Ja
ProjectStartDate	Startdatum för projektet.	smalldatetime	Nej	Nej	Ja
ProjectEndDate	Slutdatum för projektet.	smalldatetime	Nej	Nej	Ja
Responsible	Ansvarig initialerna) för projektet.	char(3)	Nej	Nej	Ja
CustomerCode	Kundenkoden för kunden som projektet avser.	nchar(10)	Nej	Nej	Ja

ReportedTime
Tabell som innehåller information om hur och när en konsult lagt ner sin arbetade tid.

Följande attribut beskriver en konsults rapporterade tid:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
Initials	Den anställdes initialer.	nchar (10)	Nej	Nej	Ja
Date	Datum då arbetet utfördes.	smallDateTime	Nej	Nej	Ja
TimeAmount	Mängden arbetad tid.	float	Nej	Nej	Ja
Price	Timpriset för den arbetade tiden.	float	Nej	Nej	Ja
TimeCode	Kod som beskriver en tidskod.	nchar (10)	Nej	Nej	Ja
ProjectCode	Projektkoden för projektet den arbetade tiden avser.	nchar (10)	Nej	Nej	Ja

Revenue
Tabell som innehåller information om alla intäkter som hör samman med ett visst projekt och kostnadsställe. Intäkter hör inte direkt samman med den tid en konsult lagt ner utan kan exempelvis vara intäkter från försäljning av licenser.

Följande attribut beskriver en intäkt:

Attribut	Beskrivning	Datatyp	Primary key	Foreign key	Tillåt Null
AccountNo	Kontonumret för kontot som intäkten registrerats på.	nchar (10)	Nej	Nej	Ja
ProjectCode	Projektkoden för projektet som intäkten hänförs till.	nchar (10)	Nej	Nej	Ja
CostDepartmentName	Namnet på kostnadsstället som intäkten hänförs till.	nchar (50)	Nej	Nej	Ja
Amount	Intäktens storlek.	float	Nej	Nej	Ja
Date	Datumet då intäkten uppstod.	smalldatetime	Nej	Nej	Ja
OtherCode	Övrigkodens som intäkten registrerats på.	nchar(10)	Nej	Nej	Ja
Type	Typ av intäkt (Inköp eller Övrigt).	nchar(10)	Nej	Nej	Ja

Time_Codes
Tabell som innehåller information om alla tidkoder. En tidkod anger vad en konsult arbetat med under en viss tid, exempel på en tidkod är ”030 Konsult”, som innebär att konsulten lagt ner tid för en viss kund.

Följande attribut beskriver en tidkod:

Attribut	**Beskrivning**	**Datatyp**	**Primary key**	**Foreign key**	**Tillåt Null**
TimeCode	Kod som beskriver en tidskod.	nchar (10)	Nej	Nej	Ja
TimeCodeName	Tidkodens namn	varchar (50)	Nej	Nej	Ja