IT University of Göteborg

CHALMERS | GÖTEBORG UNIVERSITY

Magisteruppsats i Informatik
Master thesis in applied Information Technology

REPORT NO. 2008:012
ISSN: 1651-4769

Department of Applied Information Technology

Hantering av viktiga framgångsfaktorer vid utveckling av ett Data Warehouse

En jämförelse av Inmons och Kimballs modeller och metoder

Managing of important success factors when developing a Data Warehouse

A comparing study of Inmon and Kimballs models and methods

Daniel Larsson
Marek Niemiec
Pierre Wessman

CHALMERS

UNIVERSITY OF GOTHENBURG

IT Universtiy of Göteborg
Chalmers University of Technology and Universtiy of Gothenburg
Göteborg, Sweden 2008

Hantering av viktiga framgångsfaktorer vid utveckling av ett Data Warehouse

En jämförelse av Inmons och Kimballs modeller och metoder

DANIEL LARSSON, MAREK NIEMIEC, PIERRE WESSMAN

Institutionen för tillämpad informationsteknologi

IT Universitetet i Göteborg

Göteborgs Universitet och Chalmers Tekniska Högskola

Abstrakt

Business Intelligence kan hjälpa organisationer att fatta rätt beslut genom att ta fram data som analyseras och presenteras för beslutsfattare. Data Warehouse som är en central del inom Business Intelligence går att bygga på flera sätt men där Ralph Kimballs och Bill Inmons metoder är de mest tongivande. Bill Inmons lösning fokuserar på att integrera hela organisationens data i en enda stor databas genom att modellera arkitekturen med hjälp av relationsdatabaser. Kimball däremot förespråkar dimensionsdatabasmodellering där man bygger en eller flera Data Marts och integrerar dessa via en databus. Kvaliteten för ett Data Warehouse går enligt tidigare forskning att mäta utifrån kritiska faktorer som datakvalitet och systemkvalitet. I denna uppsats undersöks via enkät och intervjuer hur Inmons och Kimballs metoder hanterar dessa kritiska faktorer. Studien granskar också hur organisatoriska faktorer kan påverkas beroende på vilken lösning man väljer. Dessa faktorer har tagits fram via en fokusgrupp bestående av representanter från WM-Data.

Nyckelord: Inmon, Kimball, Data Warehouse, Business Intelligence

Managing important success factors for development of a Data Warehouse

A comparing study of the methods and models by Inmon and Kimball

DANIEL LARSSON, MAREK NIEMIEC, PIERRE WESSMAN

Department of Informatics in Applied Information Technology

IT University of Göteborg

Göteborg University and Chalmers University of Technolgy

SUMMARY

Business Intelligence may help organizations in making the right decisions by providing data for analysis which can be presented to decision makers. Data Warehouse which is a central part of Business Intelligence can be built in several ways but the methods presented by Ralph Kimball and Bill Inmons are considered the two main methods. The solution provided by Inmon focuses on integrating the entire business into one large database using relational database development. Kimball however proposes dimensional database modeling by creating one or more data marts and integrating these using a data bus. Previous research has shown that the quality of a Data Warehouse can be measured through critical factors such as data quality and system quality. In this study we examine how the methods presented by Inmon and Kimball handle these critical factors. The study also examines how organizational factors are affected by the chosen method. These factors have been identified through a focus group comprised of representatives from WM-Data.

The report is written in Swedish.

Keywords: Inmon, Kimball, Data Warehouse, Business Intelligence

Tack till...

Vi vill tacka Business Intelligence -gruppen på WM- Data i Göteborg och Malmö som har ställt upp med mycket tid och resurser för att göra vårt examensarbete möjligt. De har välkomnat oss varmt och vi har känt att vi har fått vara ”en i gänget” under de månader vi varit där. Vi vill rikta ett särskilt tack till vår handledare Roger Casserdahl, som har lagt ner mycket tid åt att introducera oss till ämnet och bidragit till extern handledning under hela examensarbetet.

Innehållsförteckning

1 Introduktion

Många företag har idag problem med att få fram rätt typ av data för att kunna göra verksamhetsövergripande analyser och rapporter. Man har ofta flera olika källsystem som exempelvis ERP (Enterprise Resource Planning) och CRM- system (Customer Relation Management) men lyckas inte generera data ur dessa för att göra djupare analyser (Soejarto, 2003). Ett sätt att hantera detta problem är med hjälp av Business Intelligence. Med hjälp av Business Intelligence kan man ta fram data ur olika källsystemen för att generera verksamhetsövergripande analyser. Detta gör man genom att sammanföra och integrera data från olika källsystemen in i ett Data Warehouse, för att sedan med hjälp av analysverktyg kunna göra verksamhetsövergripande analyser och rapporter som ligger till grund för olika typer av beslutsfattande i en organisation (Breslin, 2002).

1.2 Bakgrund

I början på 1990 talet beskrevs Bill Inmon som fadern till Data Warehouse genom sitt nyskapande projekt ”Building the Data Warehouse” och Inmons vision om ett samlat datalager började implementeras med varierande grad av framgång (Breslin, 2007). Inmons utvecklingsmetod formgavs av ett ”uppifrån och ner” baserat synsätt där all data från de olika operationella systemens databaser skulle tidstämplas och samlas in till en större integrerad databas. Visionen är att sammanföra data genom att omvandla och integrera begrepp för att sedan kunna användas till olika analyser och rapporter.

Efter publicerandet av Inmons bok började flera experter inom databaser att ha synpunkter och olika visioner om Data Warehouse. Ralph Kimball presenterade sin syn på Data Warehouse 1996 med sin bok ”The Data Warehouse Toolkit”. Kimball till skillnad från Inmon ser utvecklingsmetoden i ett ”nerifrån och upp” orienterat synsätt vilket innebär att man utgår från det data man i slutändan vill ha (Breslin, 2007).

Gartner (Whitson, 2005) spådde att 50 % av Data Warehouse -lösningarna år 2007 kommer att misslyckas eller inte accepteras på grund av för dålig datakvalitet. De menar att fokus legat för mycket på inhämtning av data, göra den likformig samt att ladda in datan till ett Data Warehouse. Detta har gjorts utan att ta hänsyn till hur problemen med datakvalitet ska hanteras. Enligt Wixom och Watson (2001) menar man att datakvalitet och systemkvalitet är viktiga faktorer vid införandet av ett Data Warehouse och att det finns omfattande forskning som backar upp detta påstående (e.g Wand och Wang, 1996; Wang och Strong, 1996).

1.3 Problemområde

Idag finns främst två sätt att bygga ett Data Warehouse på nämligen Kimballs lösning och Inmons lösning (Lawyer & Chowdhury, 2004; Breslin, 2007). Dessa båda sätten skiljer sig åt på många sätt vad gäller arkitektur, databasmodellering, samt syn på Data Warehouse etc. För att lyckas med ett Data Warehouse -projekt är det mycket viktigt att känna till skillnaderna mellan Kimballs och Inmons modeller eftersom det

kan vara avgörande för om ett Data Warehouse -projekt kommer att lyckas eller inte (Jukic, 2006).

Enligt Hayen et al (2007) har man kommit fram till att ett Data Warehouse med hög datakvalitet och systemkvalitet förbättrar sättet som data distribueras till beslutstödsystemen och till slutanvändarna. Vidare har man kommit fram till att datakvalitet och systemkvalitet påverkar nettovinsten och ROI.

1.4 Syfte och Frågeställning

Syftet med uppsatsen är att undersöka Inmons och Kimballs lösning för ett Data Warehouse. Detta för att se hur viktiga faktorer hanteras av respektive lösning. Viktiga framgångsfaktorer som identifierats av tidigare forskning är exempelvis datakvalitet och systemkvalitet, men vi har kompletterat dessa genom en fokusgrupp med ytterligare några faktorer.

Hur hanteras kritiska framgångsfaktorer av Inmons och Kimballs Data Warehouse -lösningar?

1.5 Avgränsningar

Vi kommer i vår studie om Data Warehouse -lösningar och egenskaper för dessa, avgränsa oss till att enbart studera processen ifrån inhämtandet av data ifrån källsystemen fram till genereringen av data till rapport och analysverktygen. Denna process kallas även ”Back End” och är också som namnet säger den bakomliggande processen för datagenereringen. Behovet och nyttan av BI – system kommer inte heller att undersökas.

2. Metod

2.1 Inledning

Vi har under arbetets gång gått igenom fem faser från (1) val av ämne fram till (5) analys och diskussion. Efter val av ämne gjorde vi en litteraturgranskning (2) som delvis har varit en iterativ process under hela arbetet. Under litteraturgranskningen hittade vi framgångsfaktorer som legat till grund för vårt ramverk. Vi utvecklade sen ramverket genom en fokusgrupp där vi kom fram till ytterligare faktorer(3), och säkerställde sedan resultatet med hjälp av enkät och intervjumaterial (4).

Anledningen till att WM- Data också var intresserade av ämnet och problemet var att det idag finns två olika typer av framstående Data Warehouse -lösningar varav den ena kostar mycket mer att utveckla enligt WM- Data. WM- Data använder idag båda metoderna för olika typer av projekt, men framförallt är det regionala skillnader på vilken modell man förespråkar. Vi har under arbetets gång diskuterat mycket med gruppchefen i Göteborg och fått extern handledning och feedback. Vi har även haft diskussioner genom en fokusgrupp där vi gått igenom problemområde samt utvecklingen av ramverket. I fokusgruppen har vi, gruppchefen för Göteborg samt gruppchefen för Malmö varit med. Vi har både träffats och haft telefonkonferens vilket har varit till stor hjälp för utvecklingen av ramverket.

Vi valde att studera befintlig teori och därifrån dra slutsatser, vilket innebär att vi har valt en deduktiv forskningsansats (Andersen, 1998). Det deduktiva angreppssättet börjar i den teoretiska referensramen där vi har tagit upp relevant teorier för vårt problemområde, för att sedan tolka dem i resultatkapitlet där data sammanställts.

Det som passade vår studie bäst var en kvalitativ forskningsmetodik kompletterad med en kvantitativ undersökning i form av en enkät. Utifrån teorier har vi skapat ett ramverk med kritiska faktorer som sedan Inmons och Kimballs lösningar har ställts emot. Hur väl dessa faktorer stöds har undersökts via enkäter och intervjuer med BI-konsulter som har stor praktisk kunskap.

2.2 Tillvägagångssätt

För att undersöka hur de faktorer vi tagit upp i ramverket överensstämmer med verkligheten och hur väl Inmons och Kimballs lösningar hanterar dessa, valde vi att göra undersökningar i form av enkäter och intervjuer. Till en början mailade vi ut enkäter till samtliga anställda på BI-avdelningarna i Göteborg och Malmö, (ca 10 i Göteborg och ca 20 i Malmö). Eftersom svarsfrekvensen blev relativt låg valde vi att också komplettera med intervjuer. Att svarsfrekvensen blev låg beror på att många av de anställda inte har den övergripande kompetensen för teorierna för att kunna svara

Enkät
Vi utformade en enkät med 11 frågor som skickades till respektive stad. Enkäten distribuerades via Internet i form utav en webbenkät och länken skickades till alla respondenter via e- post. I enkäten kunde respondenterna rangordna svaret från 1-5 beroende på hur väl de tyckte att respektive metod kunde hantera framgångsfaktorerna. Vi valde att ställa frågor med tillhörande exempel för att undvika att respondenterna skulle feltolka frågorna. Utöver rangordningen fick också konsulterna frivilligt fylla i ett fält med kommentar till respektive fråga. Frågorna som ställdes var kopplade till hur väl Inmons respektive Kimballs lösning fungerade för att hantera ett antal viktiga faktorer som vi berättar mer om i teorin. Eftersom vi endast fick in 6 svar ansåg vi att detta inte räckte för att få ett tillräckligt bra underlag för resultatet och beslöt därför att komplettera med intervjuer.

Intervju
Efter enkätunderökningen gjordes fyra djupare intervjuer för att ytterligare ge bra validitet till resultatet då vi endast hade fått in 6 svar från enkäten. Vi valde att Intervjua fyra personer varav två av dessa hade svarat kortfattat på enkäten. Två personer från Göteborgskontoret samt två personer från Malmökontoret intervjuades för att ge mer tyngd i undersökningen. Respondenterna från Göteborg intervjuades på plats och respondententerna från Malmö intervjuades per telefon. De fyra personerna som valdes ut ansåg vi vara bland de mest kompetenta för att ge djupare svar på våra frågor och intervjuerna tog mellan 30-60 minuter. Dessa fyra respondeter fick svara på frågorna utifrån både Inmons och Kimballs perspektiv.

Fokusgrupp
För att bearbeta problemområdet till att både bli ett akademiskt intressant och korrekt arbete samt innehållsrikt för WM-Data diskuterades och problematiserades innehållet genom en fokusgrupp. Fokusgruppen bestod av gruppcheferna för respektive kontor i Malmö och Göteborg. Vi presenterade upplägget och vår syn på innehållet och med det startades sedan en diskussion om deras syn på innehållet. Under fokusgruppens gång framkom att organisatoriska aspekter som acceptans, ansvar och kompetens var viktiga.

Respondent presentation
HB arbetar i Göteborg och har varit verksam inom Business Intelligence under mer än 10 år. HB har mångårig erfarenhet inom projektledning, förstudier, behovsanalyser, utrednings- och utvecklingsarbete samt informations och utbildningsinsatser med specialisering på verksamhetsanalyser. HB arbetar inte med datamodellering eller ETL- processer (Extract Transform and Load) och har därför begränsad kunskap om Inmon och Kimballs datamodeller. HB har endast svarat på enkäten.

JJ har flerårig erfarenhet av utveckling i Microsoft miljö och SQL Server och god kännedom om utveckling av Data Warehouse och Business Intelligence lösningar. JJ arbetar i Göteborg och enligt Kimballs metoder med i huvudsak modellering av OLAP- kuber (Multidimensionella kuber). JJ har endast svarat på enkäten.

PW arbetar på Göteborgskontoret och är en erfaren arkitekt, analytiker och utvecklare av Data Warehouse, med erfarenhet inom flera olika affärsområden. PW arbetar till största delen enligt Kimballs sätt men har även god kunskap om Inmon metoder. PW

har även utmärkt kunskap inom ETL och OLAP- design. PW har endast blivit intervjuad.

RC arbetar i Göteborg som gruppchef för BI-avdelningen. Han har även bred kunskap och erfarenhet inom roller för teknisk arkitektur, utvecklare, ETL, samt som affärsbehovsanalytiker. RC arbetar utifrån Kimball men har även god kunskap om Inmons metoder. RC har både svarat på enkäten och blivit intervjuad.

KG är en erfaren arkitekt, modellerare och utvecklare av Business Intelligence – lösningar. KG arbetar i Malmö med ETL lösningar som följer Inmons arkitektur och använder dimensionsmodellering enligt Kimball. KG har endast blivit intervjuad.

PN arbetar med utveckling och förvaltning av BI-lösningar med huvudområde på ETL- processen. PN arbetar på Malmökontoret och till största del enligt Inmons lösning. PN har endast svarat på enkäten.

UW arbetar med utveckling och förvaltning av BI-lösningar med ETL processen som huvudområde. Han har tidigare även arbetat med utveckling, konvertering och implementation av relationsdatabassystem, samt konvertering/uppgradering av existerande databaser. UW arbetar i Malmö och till största del med Inmons systemlösningar. UW har endast svarat på enkäten.

JL är en erfaren projektledare och affärsbehovsanalytiker samt Data Warehouse arkitekt. Hans aktuella fokus ligger på Corporate Performance Management och affärsbehovsanalytiker. JL arbetar i Malmö och har kunskap om både Inmons och Kimballs systemlösningar. JL har både svarat på enkäten och blivit intervjuad.

En sammanfattning av deltagarnas kunskap presenteras nedan i Tabell nr1. De mörka fälten indikerar att respondenten blivit intervjuad.

Respondenter	**Kunskap inom Inmon**	**Kunskap inom Kimball**	**Arbetar utifrån Inmon**	**Arbetar utifrån Kimball**
HB		X		X
JJ		X		X
PW	X	X		X
RC	X	X		X
KG	X	X	X	
PN	X		X	
UW	X		X	
JL	X	X	X	

Tabell nr1: Visar respondenternas kunskap samt arbetsområde vad gäller Inmons och Kimballs teorier.

2.3 Källkritik

Att jämföra Inmons och Kimballs metoder utifrån vårt ramverk har inte varit helt enkelt, detta på grund av att den mesta litteraturen om dessa respektive lösningar har skrivits av dem själva. Vi har hittat en jämförande artikel där Inmon och Kimball ställs emot varandra av Breslin (2007) samt en undersökning och artikel av Watson och Wixom (2001) om framgångsfaktorer för Data Warehouse som vi använt oss till stor del utav.

För att få validitet i vårt arbete har vi genomför enkätundersökningar och intervjuer med medarbetare på WM- Data, men även detta kan kritiseras på grund av svarsfrekvensen. Vi tycker att de som svarat ändå har haft god kunskap och svarsfrekvensen får därav vara acceptabel för detta syfte.

3 Teori

Teoriavsnittet är uppdelat i fyra delar där varje del är viktig för förståelsen av resultatet. Först kommer en generell beskrivning av Data Warehouse där vi förklarat sambandet med Business Intelligence och vad som är viktigt för ett Data Warehouse. Vi har sedan beskrivit Inmon och Kimballs lösningar som presenteras var för sig för att avslutningsvis beskriva det ramverk av viktiga faktorer som resultatet presenteras utifrån.

3.1 Data Warehouse

Inmon et al (2005) definierar Data Warehouse med följande citat:

> *"A data warehouse is a subject oriented, integrated, non-volatile, and time variant collection of data in support of management's decisions"*(Inmon et al, 2005, s.29).

Inmon menar alltså att ett Data Warehouse är ett ämnesorienterat datalager av sammansatt och tidstämplad data som skall vara tillgängligt för trendvärde och beslutstöd. Kimballs definition av Data Warehouse är något annorlunda och han menar att ett Data Warehouse är:

> *"The conglomeration of an organization's data warehouse staging and presentation areas, where operational data is specifically structured for query and analysis performance and ease-of-use."(* Kimball, 2002, s.397).

Kimball menar alltså att ett Data Warehouse är ett sätt att lagra och presentera en organisations sammansatta data av en organisations operationella data. Denna data skall vara strukturerad utifrån ett bestämt sätt för att kunna ställa frågor och generera analyser och samtidigt vara lättanvänt. Både Inmon och Kimball anser därmed att ett Data Warehouse är en sammanfogning av en organisations data som är till för att generera rapporter och analysunderlag.

Data Warehouse som används för att kunna göra analyser och rapporter med hjälp av beslutstödssystem även kallat Business Intelligence är en central roll inom beslutstöd. Business Intelligence möjliggör att man från givna domäner och applikationer kan hämta information för att generera analytiska modeller och genom tillgången till databaserna stödja beslutsfattare att effektivt fatta beslut i komplexa och svårstrukturerade organisationer (Klein och Methlie, 1990).

Gartner Group definierar Business Intelligence enligt följande:

> *"The key to thriving in a competitive marketplace is staying ahead of the competition. Making sound business decisions based on accurate and current information takes more than intuition. Data analysis, reporting and query tools can help business users wade through a sea of data to synthesize valuable information from it—today these tools collectively fall into a category called "Business Intelligence."* Gartner (1996, http://www.b-eye-network.com/view/1119).

Business Intelligence handlar alltså om att använda information för att generera underlag för bättre och säkrare beslut. Att utifrån exakt och aktuell information skapa beslutsunderlag är hela tanken med BI och detta möjliggörs genom att lagra data i olika typer av databaser och Data Warehouse. Att lagra data och göra den konsistent möjliggör alltså att man utifrån historisk och aktuell data kan skapa trendvärden, analyser och rapporter med hjälp av olika typer av applikationer och verktyg. En annan typ av definition av BI där man istället vill lyfta fram varför BI är bra, samt nyttan av BI kan vara intressant att titta på. Inmon, Terdeman och Imhoff definierar nyttan av BI enligt följande:

> *"The discipline of understanding the business abstractly and often from a distance. With business intelligence, you can see the forest and the trees*." (Inmon et. al, 2000, http://www.b-eye-network.com/view/1119)

English (2005) menar att definitionerna ovan inte tar upp de mänskliga kraven eller kontexten för BI, utan endast talar ur mjukvara och tekniska aspekter. Han refererar till ett citat från Data Warehouse Review som också definierar nyttan samt menar att BI är följande:

> *"Business intelligence is actually an environment in which business users receive data that is reliable, consistent, understandable, easily manipulated and timely. With this data, business users are able to conduct analyses that yield overall understanding of where the business has been, where it is now and where it will be in the near future. Business intelligence serves two main purposes. It monitors the financial and operational health of the organization (reports, alerts, alarms, analysis tools, key performance indicators and dashboards). It also regulates the operation of the organization providing two- way integration with operational systems and information feedback analysis*." (English, 2005)

Time Variant och Non Volatile

All transaktionsbaserad data måste lagras med information om datan (exempelvis datum), och innehåller därmed tidsinformation. När data sparas är det för att kunna användas till ett historiskt perspektiv. Nya versioner av data får inte skrivas över eller uppdateras utan endast tidstämplas och läggas till. Att den inte får uppdateras och skrivas över är beskrivningen på Non Volatile.

Skillnad i tidsperspektiv

Enligt Inmon et al (2005) existerar det tydliga skillnader av tidsperspektivet i ett Data Warehouse i jämförelse med en transaktionsbaserad databas. Tidshorisonten för ett Data Warehouse är betydligt längre än ett operativt system enligt Inmon et al (2005). För ett operativt system är tidshorisonten ungefär 60 till 90 dagar medens ett Data Warehouse har en tidshorisont på 5 till 10 år. Operativa databaser innehåller aktuella värden, alltså värden som är aktuella just nu. Om ett värde i verkligheten ändras, så ändras även det motsvarande värdet i databasen. I ett Data Warehouse är värdet tidsstämplat, vilket innebär att det är kopplat till en tidsdimension. Därav sker inga uppdateringar i ett Data Warehouse. Nyckelstrukturen i en operativ databas kan

innehålla ett tidselement, men kan lika gärna inte göra det medans det alltid gör det i ett Data Warehouse.

Granularitet

Inmon et al (2005) menar att den absolut viktigaste aspekten i ett Data Warehouse är granularitet. Granularitet innebär detaljrikedomen eller summeringsnivån på datan i ett Data Warehouse. Hög detaljrikedom innebär låg granularitet. I en operativ databas tas låg granularitet för givet, till skillnad från ett Data Warehouse där det är allt annat än självklart hävdar Inmon et al (2005). Anledningen till att granulariteten är den viktigaste aspekten i ett Data Warehouse är att det påverkar databasens storlek samtidigt som det påverkar vilka typer av frågor man kan ställa till databasen. Detta skapar en balansgång mellan datavolym och detaljrikedom.

För att illustrera ett exempel kan vi använda ett telefonbolag. Vi fokuserar på databasen för telefonsamtal. I ena fallet lagrar vi varje samtal som görs under en månad (låg granularitet) och i andra fallet antal samtal som är gjorda summerat månadsvis (hög granularitet). Om vi väljer att i ena fallet ställa en fråga när Pelle ringde ett utlandssamtal i 3 timmar, så går detta endast att genomföra om vi väljer tabellstrukturen med låg granularitet. Även att det går kommer det att ta väldigt lång tid att svara på frågan eftersom enormt många rader behöver jämföras innan vi hittar Pelles samtal. Låt oss anta att vi vill ställa en fråga om hur många samtal som ringdes från Sverige till USA förra kvartalet. Det går att göra i båda fallen av granularitet, men det tar enormt mycket tid och resurser för att svara på frågan om vi väljer strukturen med låg granularitet, medans det går snabbt och smärtfritt om vi väljer hög granularitet.

För att både undvika att vissa frågor exempelvis över lång tid tar enormt mycket resurser och tid att ställa och samtidigt undvika att vissa frågor inte går att ställa på grund av för hög granularitet bör en organisation använda sig av flera nivåer av granularitet enligt Inmon et al (2005). Detta gör det möjligt att både ställa detaljerade frågor som när ringde Pelle ett utlandssamtal i 3 timmar förra månaden samtidigt som det går att ställa sammanfattande frågor som hur många samtal från Sverige till USA gjordes det under februari månad.

Partitionering av data

Inmon (1996) hävdar vidare att en annan viktig aspekt är partitionering av datan i ett Data Warehouse. Vad som menas med detta är att man bryter ner data i separata fysiska delar som kan hanteras oberoende av varandra. För att belysa vikten av partitionering och granularitet skriver Inmon (1996) att om dessa två begrepp utförs ordentligt kommer de övriga aspekterna av design och implementation att bli lättare att hantera. Frågan är enligt Inmon (1996) inte om man ska partitionera datan utan snarare hur man ska partitionera datan. Anledningen till att man vill bryta ner datan i mindre fysiska enheter är att det ger mer flexibilitet. Fördelar med detta är enligt Inmon (1996) att datan kan smidigare omstruktureras, indexeras fritt, Sekventiellt skannat, omorganiserat, återhämtas samt övervakas.

Vad som exakt menas med att partitionera data är att man delar in data med liknande struktur i mer än en fysisk enhet. Exempel på kriterier man kan dela in datan i är datum, affärsinriktning, geografi samt organisatoriska enheter.

Man kan välja att partionera datan på systemnivå eller på applikationsnivå. Om man väljer att partitionera datan på systemnivå innebär det att man uppnår detta med hjälp av databasen och i viss mån även operativsystemet. Om man istället väljer att partitionera datan på applikationsnivå utförs partitioneringen med hjälp av applikationskod och är därför kontrollerad helt och hållet av programmeraren. Inmon (1996) hävdar att det i regel är rimligt att partitionera datan på applikationsnivå. Anledningarna till detta är bland annat att definitionen av datan kan ändras för varje år om man partitionerar datan på applikationsnivå. När data lagras i långa tidsperioder kommer definitionen av datan som lagras att ändras under tiden och för att möjliggöra detta måste man partitionera datan på applikationsnivå. Ytterligare en viktig anledning till varför man bör partitionera datan på applikationsnivå är att datan kan flyttas från ett processkomplex till ett annat. När belastningen och volymen av datan blir för stor är det fördelaktigt att kunna göra detta menar Inmon (1996).

3.1.1 Relationsmodellering

År 1976 skrev amerikanen Peter Pin-Shaen Chen en artikel vid namn ”*The Entity Relationship Model – Toward a Unified View of Data”*, som lade grunden till vad vi idag kallar för ER-modellering och ERD (Entity Relationship Diagram). Modellen skapar ett standardiserat sätt att beskriva en datamodell i form av entiteter, attribut och relationer. Vi väljer att berätta mer om relationsmodellering eftersom Inmons lösning bygger på ett relationsmodellerat och normalfördelat Data Warehouse. Entiteter är något som vi vill ha information om, det kan t.ex. vara en person, en sak, ett tillstånd, en plats osv. En entitet kan vara något som existerar, har existerat eller kommer att existera (Andersen, 1994). Exempel på entiteter är Carl XVI Gustaf, Drottningholms slott, SM-finalen i Elitserien i hockey 2005 osv. I datamodelleringssammanhang ordnar man entiteterna i entitetstyper för att kunna modellera entiteter och dess attribut och relationer. En entitetstyp kan vara Person, Bostad eller Tillställning där Carl XVI Gustaf är en Person, Drottningholms slott är en bostad och SM-finalen en tillställning. För varje entitet skall en primärnyckel utses. Syftet med en primärnyckel är att ha ett attribut i entiteten som är unikt, som bland annat används för att länka relationer mellan entiteter. Primärnyckeln måste vara ett unikt attribut för just den entiteten, exempelvis personnumret för en person eller chassinumret för en bil. Det är inte alltid odiskutabelt vilket attribut som skall vara primärnyckel, dessa kallas kandidatnycklar vilka en primärnyckel väljs utifrån.

En relation är ett samband mellan två entiteter, ett sätt två entiteter förhåller sig på (Andersen, 1994). Exempelvis kan en relation mellan Person och Bostad vara att Carl XVI Gustaf bor i Drottningsholms slott. Det finns tre olika relationstyper: 1:1, 1:N och N:N. 1:1, eller en till en, som det uttalas innebär att det finns en förekomst av entitet A per förekomst av entitet B. Exempelvis om man kopplar tabellen Bil till tabellen Motor resulterar det i en 1:1-relation då en bil endast har en motor i taget och en motor kan endast sitta i en bil åt gången. Ett exempel på en 1:N-relation är Pappa och Barn, en pappa kan ha flera barn men ett barn kan endast ha en pappa. En N:N-relation skulle kunna vara Lärare och Elev, en lärare kan ha flera elever och en elev kan ha flera lärare.

Ett attribut är vanligtvis en egenskap hos en entitet. Man knyter ett antal attribut till varje entitet (Andersen, 1994). Entiteten Person kan t.ex. innehålla attribut som personnummer, namn, adress, telefonnummer, e- post, ålder, längd osv..

Inmon menar att ett relationsmodellerat Data Warehouse innebär större flexibilitet eftersom man i efterhand kan ändra datastrukturer samt vidareutveckla och bygga ut. Eftersom relationsmodellering bygger på en starkt normaliserat datastruktur blir det också fler tabeller men med mindre redundant data. Inmon menar att det är en fördel att använda relationsmodellering i ett Data Warehouse, men att man mycket väl kan använda dimensionsmodellering i Data Marts (Inmon et al, 2005).

3.1.2 Dimensionsmodellering

Till skillnad från relationsmodellering som Inmon förespråkar så använder Kimball en arkitektur som baseras på dimensionsmodellering. Inmon et al (2005) använder sig också av dimensionsmodellering, men då endast i sina Data Marts. Dimensionsmodellering är inget som Kimball uppfunnit utan det har sakta vuxit fram sedan början av 70-talet (Kimball, 2007). Dimensionsmodellering skiljer sig från traditionell relationsmodellering som många IT-experter använder sig av idag (Breslin, 2007). Inom dimensionsmodellering börjar man med tabeller och inte med relationsdatabasschema som ERD. Dimensionsmodellering är starkt denormaliserad och tabellerna är antingen faktatabeller eller dimensionstabeller, båda dessa beskrivs nedan.

Faktatabeller

En faktatabell innehåller ett numeriskt värde som på något sätt beskriver prestandan, som exempelvis försäljning i kronor. Enligt Kimball (2002) är tanken att varje affärsprocess ska ha ett tillhörande prestandamått som då läggs i en faktatabell. Detta resulterar i följande:

- En rad i en faktatabell motsvarar ett prestandamått
- Ett prestandamått är en rad i en faktatabell
- Alla prestandamått måste vara i samma grain

Faktatabeller ska innehålla information om exempelvis *hur mycket* något kostar och det viktigaste då är att faktatabellerna innehåller numeriska värden samt värden som är additiva (adderbara). Detta måste man göra eftersom en fråga mot en databas sällan returnerar enbart en enda rad och då är det önskvärt att kunna summera resultatet av flera rader. Det finns också delvis additiva värden samt icke additiva värden (Ballard et al, 2006). Delvis additiva värden går att summera ihop med vissa dimensioner medan icke additiva värden måste antingen göras om till additiva värden eller avläsas rad för rad. Detta är något man vill undvika eftersom det inte är särskilt effektivt då man måste behandla datan. Ett icke additivt värden skulle enligt Ballard et al (2006) kunna vara temperaturmätning per dag i grader. Det blir meningslöst att summera denna information, däremot skulle det kunna bli meningsfullt om man räknade ut ett medelvärde per dag (Tabell nr2).

Icke additivt värde

Dag	Temperatur
Måndag	24C
Tisdag	30C
Onsdag	27C
Total	81C OBS!

Exempel 1

Semiadditivt värde

Datum	Transaktionsstorlek	Saldo SEK
Januari, 3e	+500 (Insättning)	4000
Januari, 7e	-800 (Uttag)	3200
Januari, 14e	+400 (Insättning)	3600
Januari	**Total**	10800 OBS!

Exempel 2

Tabell nr2: Exempel på hur en tabell med icke additiva värden kan se ut samt en tabell på med semiadditiva värden.

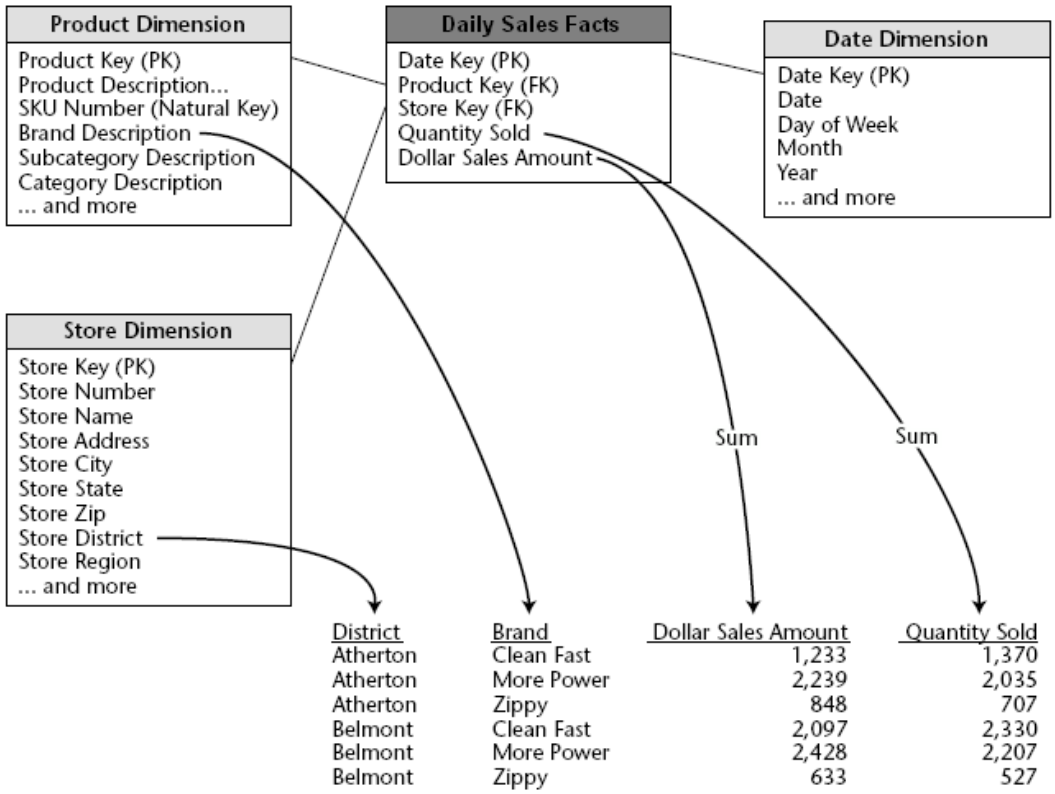
I Exempel 2 (Tabell nr2) visas exempel på ett semiadditivt värde. Det är inte intressant att veta summan av saldo i januari såvida man inte delar den på antalet dagar. Däremot är det ett semiadditivt fakta eftersom man skulle kunna ställa en fråga som är: ”*Vad är det totala saldot för alla konton i en viss bank vid på en viss dag?*” vilket innebär att datan är intressant bara vid en viss tidpunkt. Delvis additiva och icke-additiva värden kallas för *snapshot* eller *periodic snapshot.* Additiva värden kallas för *Cumulative* eftersom dessa beräknar summan av ett visst värde under en viss period.

Teoretiskt sett skulle man också kunna ha värden i faktatabellerna som är beskrivna med text, exempelvis ”mycket” eller ”lite” (Kimball, 2002). Detta är inte önskvärt för precis som med icke additiva värden så går inte värdet av en text att värdera och summera på ett enkelt sätt utan vad man då måste göra är att omvandla informationen till additivt, numeriskt värde. Innehåller en faktatabell ett textbaserat värde bör detta troligtvis ligga i en dimension istället såvida inte texten i varje rad i faktatabellen faktiskt är unik eller under ständig förändring. En viktig sak när det gäller numeriska värden i faktatabellen är att dessa tabeller inte bör fyllas ut med nollvärden om vi inte kan tilldela de något värde vid en viss tidpunkt. Exempelvis bör man inte skriva noll om ingen försäljning sker eftersom detta bara blir överflödig information i faktatabellen. Genom att bara inkludera aktiviteter som förekommer så kan man dra ner på storleken eftersom faktatabeller vanligtvis redan tar upp ca 90 % av utrymmet i en dimensionsdatabas (Kimball, 2007). Faktatabeller innehåller många rader och relativt få kolumner vilket förenklar användningen samt frågor mot databasen. Det gäller att hålla antalet kolumner nere. Ett exempel på en faktatabell med fem kolumner och millioner rader på 10GB kommer att öka med 2GB om man utökar databasen med bara en kolumn (Breslin, 2007).

Dimensionstabeller

En dimension kan vara en händelse eller ett objekt, som exempelvis en produktdimension. I dimensionstabellerna lagras beskrivningar olika delar som ingår i ett företag eller företagsverksamhet och varje dimension har sedan en primärnyckel som är kopplad till faktatabellen (Kimball, 2002). Exempel på dimensioner kan vara Tidsdimension eller Produktdimension. Dimensionstabellerna bör bestå av ett stort antal kolumner med tillhörande attribut som beskriver raderna i dimensionstabellen.

När man designar dimensionstabeller strävar man efter att ha många kolumner som istället resulterar i att man får betydligt färre rader. Antalet rader kan då ligga på hundra tusentals istället för miljoner. Dimensionstabeller kan däremot innehålla hundratals kolumner. Detta kan man göra eftersom alla attribut ligger i faktatabellen. (Breslin, 2007)

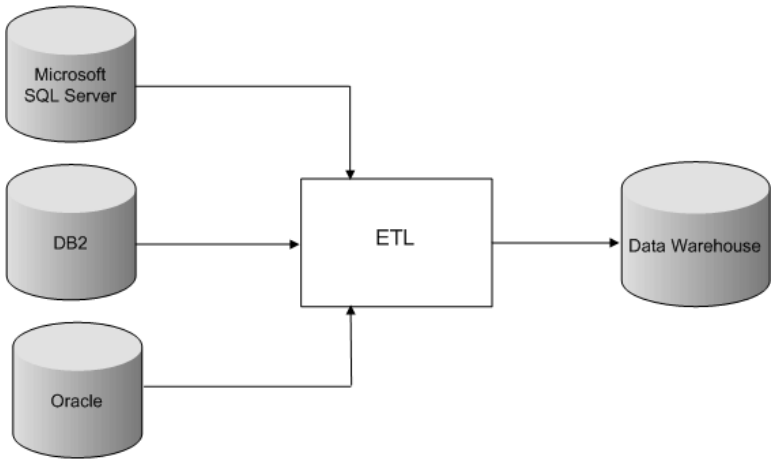


Figur nr1: I mitten finns en faktatabell och dimensionerna finns runtomkring. Detta kallas också för ett starschema.

Exempel på hur man kan dimensionsmodellera ses i Figur nr1. Det är viktigt att designa och namnge attributen på ett bra sätt eftersom man enligt dessa attribut sorterar resultatet efter när man kört frågor mot databasen. Det är detta som är intressant för slutanvändaren, att t.ex. kunna se antal sålda enheter per dag där dag är ett attribut i en datumdimension (Kimball, 2002). Attributen i dimensionstabellerna bör vara förklarande och inte med förkortningar. Det är vanligt att man har en längre och en kortare beskrivning. Om man vill använda förkortningar på exempelvis en produkt för att det är relevant så bör man utöka tabellen med ytterligare ett attribut som då innehåller förkortningen. När man ska hämta numerisk data från ett källsystem så är det inte alltid man vet om datan ska hamna i en faktatabell eller i en dimensionstabell. Vad man då bör göra det är att ta reda på om värdet är konstant eller om det på något sätt skulle kunna ingå i en relevant beräkning. Vi kan exempelvis fråga oss om det är relevant att kunna veta och summera vikten av produkt A. Om vi vill det så ska vikten hamna i en faktatabell och om det inte är intressant att mäta vikten så bör den hamna i en dimensionstabell (Kimball, 2002).

3.1.3 ETL

När man ska hämta data från olika källsystem eller filer för att sedan sammanföra den stöter man på problem med att datan är definierad på olika sätt eller innehåller olika datatyper fast man menar samma sak. Detta beror på att källsystemen inte är integrerade med varandra och databaserna skiljer sig åt. För att sammanföra dessa databaser så använder man sig av ett ETL-verktyg för att rensa datan. ETL är en förkortning för Extract, Transform och Load. ETL är egentligen flera verktyg eller subsystem som man använder för att hämta data, bearbeta den och sedan skicka den vidare. Enligt Kimball (2004) tar själva ETL-systemet eller ”the back room” som innefattar allt arbete som användarna inte ser, upp ca 70 % av tiden som det tar att bygga ett Data Warehouse. Även Inmon et al (2005) menar att detta arbete tar tid och att det inte är ovanligt att det tar 80 % av tiden. Nedan visas en förklarande bild på hur ETL-processen hänger samma med Data Warehouse (Figur nr2) och därefter följer en kort beskrivning med exempel på vad varje del i ETL gör.



Figur nr2: Illustrerad bild av ETL-processen och sambandet mellan de olika komponenterna.

Extract

Data hämtas från ett källsystem som skulle kunna vara Microsoft SQL Server men också t.ex. en fil eller ett excel-ark. När man hämtar datan från ett källsystem så gör man en databaskoppling till databasen för att på så sätt få ut informationen.

Transform

Datan ska sedan behandlas och bearbetas och under bearbetningen anpassar man datan och korrigerar den så att den blir anpassningsbar. Eftersom datan hämtas från flera olika källsystem kan den se olika ut och måste då integreras med data hämtad från andra källsystem. Saker som man måste ändra på kan vara felstavningar, delar som saknas, typomvandlingar etc. Verktyget som korrigerar felen går ofta att ställa in via parametrar. Exempel på en korrigering skulle kunna vara två olika databaser som benämner kön olika där den ena databasen använder sig av ”man/kvinna” medans den andra har ”1/0”. Verktyget hjälper då till med att sammanföra den heterogena information så att den blir homogen och kallas likadant.

Load

När datan har blivit transfomerad skickas sedan datan vidare till ett data warehouse för att användas mot slutanvändare eller för vidare behandling. Datan som hamnar i Data Warehouse't kan antingen skriva över gammal data varje gång eller någon gång i veckan eller så bygger man bara på med ny information så att man sparar historisk data. Detta beror helt på vad man har för behov.

ETL verktyg är idag ganska bra och enligt Inmon et al (2005) kan man skilja på två typer av verktyg, de som är kodproducerande och de som endast är run-timemoduler. Run-timemoduler kan ses som en motor som hämtar data och via ett antal parametrar skickar vidare (Load) datan till målsystemet. Kodproducerande ETL verktyg är effektivare eftersom de producerar fram kod, exempelvis PL-SQL kod, som man sedan kan ändra i om man vill innan man väljer att skicka datan.

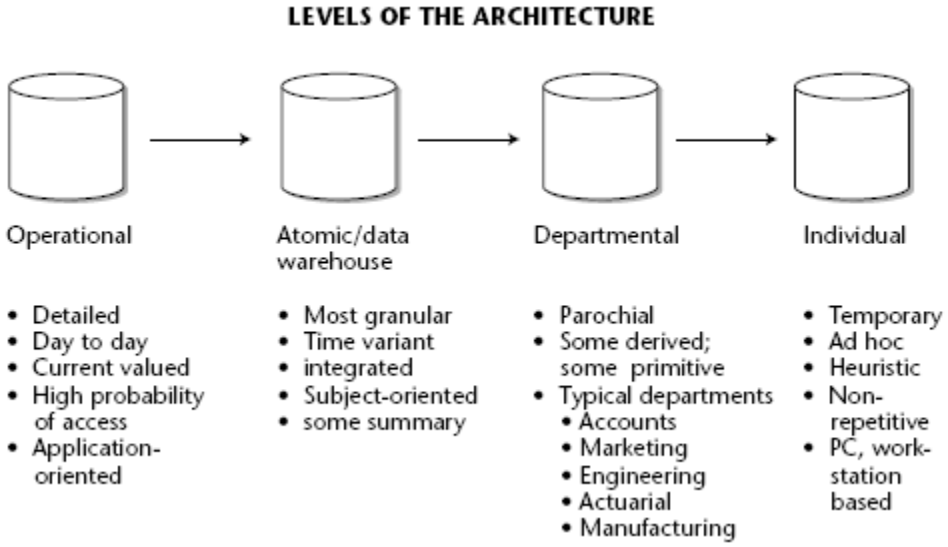
3.2 Inmons lösning

3.2.1 Filosofi

Inmon et al (2005) ser ett Data Warehouse som en del av sin organisationsomspännande modell CIF (Corporate Information Factory), vilket innebär att Data Warehouset och de operationella databaserna är del av en större modell. Hans metoder och modeller grundar sig i väl använda tillvägagångssätt som har funnits i årtionden innan begreppet Data Warehouse myntades. Detta är en av anledningarna till att man kallar hans filosofi evolutionär snarare än revolutionär. Han förespråkar även att utvecklingen av ett Data Warehouse ska ske gradvis. Med andra ord skapar man en kravspecifikation för ett delmål och utför detta, därefter återgår man sedan till kravspecifikationen igen och börjar om. Han tillämpar alltså ett iterativt sätt att utveckla ett Data Warehouse med små förändringar i taget, vilket är ytterligare ett kännetecken för ett evolutionärt synsätt. En bieffekt av detta evolutionära synsätt är att Inmons primära målgrupp är IT-proffs, då det krävs hög expertis för att förstå hans verktyg och utvecklingsmetodiker. Användaren har därmed en relativt passiv roll om man tillämpar Inmons synsätt.

3.2.2 Arkitektur

Inmon (1996) delar in omgivningen i fyra nivåer i den arkitektuella omgivningen – operationell nivå, atomisk nivå, avdelningsnivå och individuell nivå. Dessa nivåer utger basen för en större arkitektur som kallas för CIF (Corporate Information Factory).



Figur nr2: De fyra strukturerna i Inmons Data Warehouse-arkitektur.

Operationell nivå
Den operationella nivån innehåller detaljerad applikationsorienterad data, vilket innebär att det är data som är lagrad i källsystemet. Datan är aktuell, vilket betyder att alla poster representerar vad det motsvarande värdet är just nu. Exempelvis vad en produkt kostar just nu, eller hur mycket en person har lånat just nu.

Atomär nivå
Även kallad Data Warehouse-nivå. Den atomära nivån innehåller normaliserad data som är integrerad och innehåller historisk data. Datan har hög detaljrikedom och är inte är uppdateringsbar. Datan som existerar här är och i följande nivåer är historisk, vilket betyder att man kan se vad en produkt kostade för en vecka sedan eller hur mycket en person har lånat mellan 2001 till 2003. En viss summering av data förekommer här, exempelvis kanske försäljning summeras per dag istället för per transaktion.

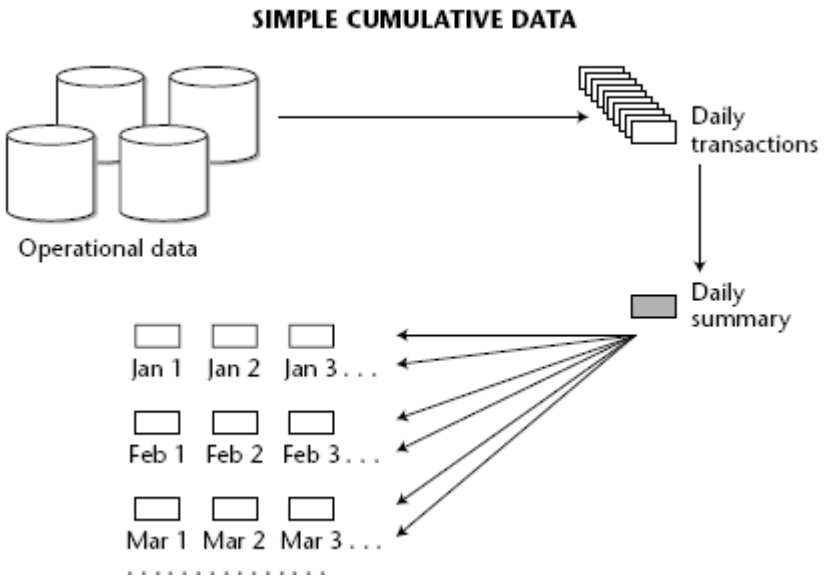
Avdelningsnivå
Denna nivå kan också kallas för Data Mart-nivå. Här summeras datan från lätt till hög summeringsnivå beroende på avdelningens behov. Denna nivå är utformad med hänsyn till slutanvändarnas behov och är specifikt anpassad till en viss avdelning.

Individuell nivå
Den individuella nivån innehåller temporär data baserad på frågor som ställs av enskilda användare. Datan här är heuristisk, är ad hoc-baserad och tenderar att existera på användarens egna dator. Exempelvis skulle en fråga kunna vara: *Hur många reklamationer gjordes i Sverige på Volvo V70 förra kvartalet?*.

3.2.2.1 Strukturering av data

Det finns många sätt att strukturera datan i ett Data Warehouse enligt Inmon (1996). Han tar upp de vanligast förekommande:

Simple cumulative data



Figur nr3: Den enklaste strukturingen av datan i Inmons modell: simple cumulative data.

Enligt Inmon (1996) är den en enklaste formen av data i ett Data Warehouse är när data har blivit samlat post för post, vilket kallas simple cumulative data. Figuren ovan

visar att dom dagliga transaktionerna transporteras från den operativa verksamheten och summeras därefter dagsvis i ett Data Warehouse.

Rolling summary data

En variant av simple cumulative data är strukturen rolling summary data. Skillnaden är att i rolling summary data summeras datan förutom dagligen även veckovis, månadsvis och årligen. När en månad har gått summeras månaden och veckosummeringen nollställs och börjar om, när ett år har gått summeras året och månadssummeringen nollställs och börjar om.

Simple Direct

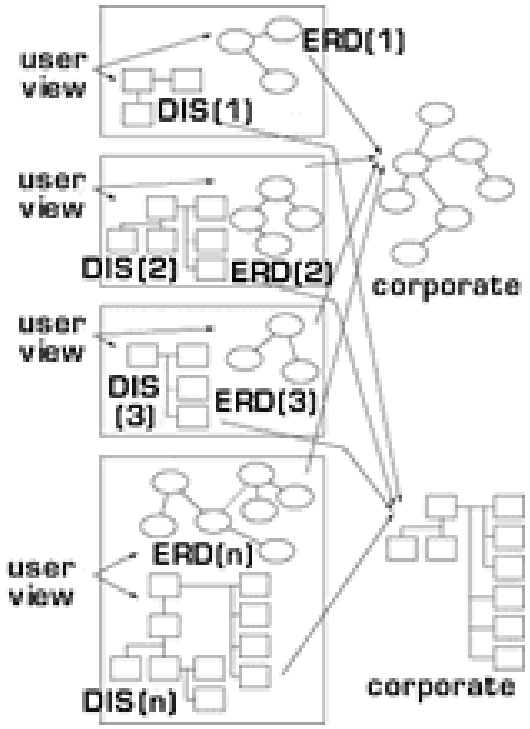
Ytterligare en möjlighet att strukturera datan på är modellen Simple Direct. Här samlar man in data och skickas helt och hållet in i ett Data Warehouse utan någon summering. Detta görs ej på daglig basis utan på en längre tidsintervall, exempelvis veckovis.

Continuous

När vecka 42 är klar kan den simpla filen (ovan) slås den ihop med den simpla filen för vecka 41 för att skapa en kontinuerlig fil.

3.2.2.2 Three Level Data Model

Inmon (1996) delar upp datamodellen i tre nivåer, ERD, DIS samt Fysisk modell. Nivåerna presenteras nedan i kronologisk ordning

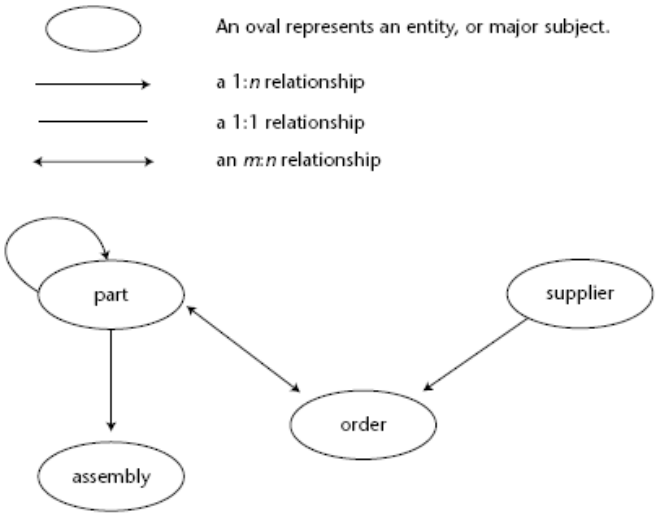


Figur nr4: En illustrerande bild av de tre nivåerna i Inmons datamodell, vilka är ERD, DIS och den fysiska modellen.

ERD (högnivå)

ERD (Entity Relationship Diagram) är den första nivån i datamodellen som Inmon

använder sig utav i Data Warehouset. ERD används för att hitta förfina entiteter, deras attribut och mellan dem. Relationerna mellan entiteterna är beskrivna med pilar, riktningen på pilarna indikerar hur relationen mellan entiteterna är.

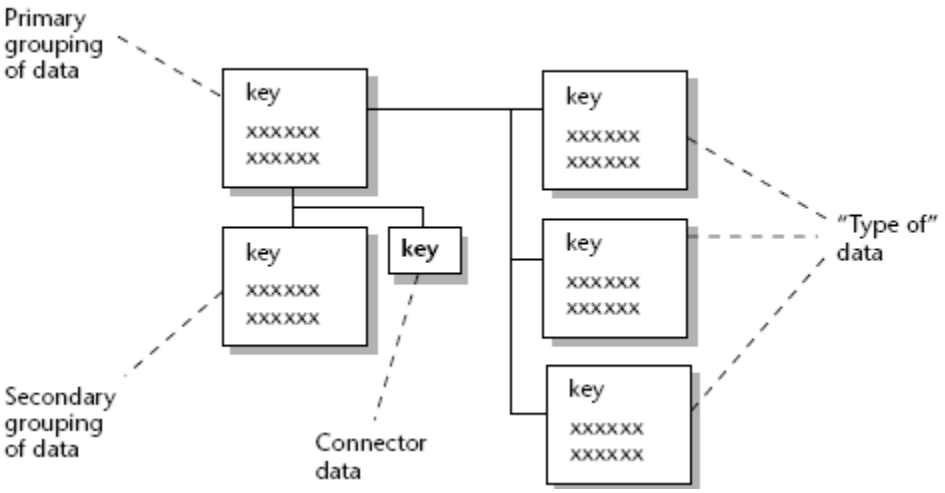


Figur nr5: Ett exempel på hur en ERD-modell kan se ut.

Vilka entiteter som skall vara med i modellen och inte bestäms av integrationsomfattningen (scope of integration) som Inmon kallar det. Integrationsomfattningen måste bestämmas innan man påbörjar modelleringsprocessen, och bestäms av modellerare, ledningen och slutanvändaren. Om omfattningen inte beslutas före modelleringen finns det enligt Inmon (1996) risk att modelleringsprocessen kommer att utmynna i en evighetsprocess. ERD-modellerna delas in i individuella användarvyer som sedan utgör en stor ERD för hela företaget som Inmon kallar för ”corporate ERD”.

DIS (mellannivå)
När relationerna mellan de övergripande entiteterna är modellerade kommer nästa steg i modellen, mellannivån eller DIS. För varje entitet som identifierades i ERD-modellen skapas en DIS-modell. En DIS-modell innehåller följande:

- En primär grupp av data
- En sekundär grupp av data
- En koppling
- “Typ av”-data



Figur nr6: Ett exempel på en DIS-modell.

Den primära gruppen av data innehåller attribut som endast existerar en gång per ämne, den sekundära gruppen av data innehåller attribut som kan existera flera gånger per ämne. Kopplingen fungerar som en främmande nyckel till de andra ämnena. ”Typ av”-datan är attribut som är bundet till ämnet. Inmon et al (2005) tar upp ett exempel som reder ut begreppen på ett bra sätt. För en bank genererar entiteten kund primära grupper av data, t.ex. konto. Då kan exempelvis lån, besparingar och bundet kapital bli den sekundära gruppen. Kopplingen visar att en kund kan ha flera konton på banken och slutligen genererar kontot data såsom uttag och insättningar som är ”typ av”-datan.

Fysisk modell (lågnivå)
Den sista nivån i modellen är den fysiska, som utökar mellannivån genom att beskriva nycklar och den fysiska karaktären. Detta inkluderar även optimeringsbeslut. Det första steget som omfattar optimering av prestandan är att besluta om granulariteten och partitioneringen. Efter detta sker ett antal andra optimeringsaktiviteter. Inmon et al (2005) tar upp följande: arrayer av data, sammanslagning av tabeller, selektiv redundans, fortsatt separation av data, ärvd data, preformatering, preallokering, relationsartifakt och för-joinade tabeller.

3.2.3 Utvecklingsmetodik

Inmon et al (2005) påpekar att när man utvecklar ett Data Warehouse bör inte allt skapas på en gång, utan gradvis. Detta kallar han för ett evolutionärt synsätt. Kostnaden att införa ett komplett Data Warehouse på en gång, kravet på resurser och avbrottet som sker i omgivningen på grund av utvecklingen är alla faktorer som ligger till grund för att utvecklingen skall ske iterativt och gradvis menar Inmon et al (2005). Han hävdar att man bör slutföra en iterativ process åt gången, för att minimera personalresurserna och störningen av den existerande applikationsmiljön. Både storleken och hastigheten är viktig eftersom resultaten måste komma snabbt menar Inmon.

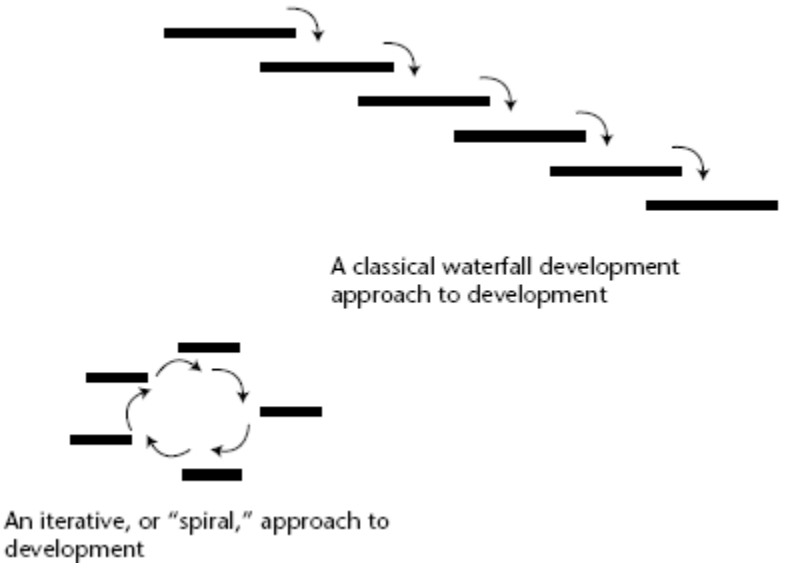
Migrationsplan
Grunden för en migrationsplan är organisationens datamodell, som representeras av informationsbehovet med betoning på vad som behövs, inte vad som finns. För att

belysa vikten av en datamodell jämför Inmon utvecklingen av ett Data Warehouse utan en klar organisatorisk datamodell med att navigera utan karta. I regel identifierar den organisatoriska datamodellen organisationen på hög nivå. Baserat på denna nivån modellerar man sedan en mer specifika modell (mitt-nivå). Denna modell är utformad ifrån ämnena som är framtagna i samband med den organisatoriska datamodellen. Båda dessa modeller fokuserar enbart på den atomära nivån (Data Warehouse) och har ingen ambition att beskriva data på avdelningsmål (Data Mart). När dessa två modeller är framtagna, är nästa steg att ta fram något som Inmon kallar för *the system of record*, vilket han refererar till företagets lämpligaste data. Med lämpligaste data menar han den data i källsystemen som lämpar sig bäst för att fylla den organisatoriska datamodellens behov. I vissa fall finns inte datan representerad i källsystemen, medans det i andra fall finns flera källor. Nästa fråga att besvara är vilka teknologiska utmaningar det innebär att hämta ut datan från källsystemet in i ett Data Warehouse. Följande aspekter tar Inmon upp: ändrad DBMS (databas) – datan skickas från en DBMS till en annan, ändrat operativsystem, behovet att förena datan från olika databaser och operativsystem och förändring i grundläggande dataformat – exempelvis från ASCII till EBCDIC. Ytterligare ett problem som Inmon tar upp är datavolymen. I vissa fall genererar källsystemen enorma mängder data. Därför kan det bli nödvändigt att behandla datan från källsystemen på olika sätt innan den kan läggas in i ett Data Warehouse. Detta kan t.ex. vara summering eller rengörning av datan. Nästa steg är att Data Warehouse-designen. Om modelleringen är utförd ordentligt kommer designen att bli relativt enkel att utföra menar Inmon. Endast ett fåtal element behöver ändras för att göra den organisatoriska datamodellen och mitt-nivå-modellen till en Data Warehouse-design. Principiellt behövs följande steg utföras: ett tidselement måste läggas till i nyckelstrukturen om det inte redan är gjort, all ren operationell data måste elimineras, referensiella integritetsrelationer måste göras till artefakter, ursprunglig data som används ofta läggs till i designen. Inmon hävdar även att man dessutom bör beakta datastabiliteten. Data som ändras ofta isoleras ifrån data som är stabil och inte ändras ofta, t.ex. ett bankkonto kan ändras flera gånger om dagen medens en kundadress inte ändras speciellt ofta. Data som förekommer väldigt ofta (stor datavolym) kommer att designas annorlunda än data som inte gör det. Metoder för att hantera data med stor volym är summering, aggregering och/eller partitionering. När Data Warehouse-designen är klar kommer nästa steg som är design och utveckling av gränssnittet mellan källsystemen och Data Warehouset. Detta gränssnitt är mer känt som ETL-processen. Inmon hävdar att upp till 80% av den totala utvecklingstiden av ett Data Warehouse går åt till denna process. Aktviteten efter ETL-processen är att befolka ämnesområdena med data (kund, produkt, försäljning osv.). Det går till enligt följande: först hämtas datan upp ifrån källsystemen till Data Warehouse-miljön, därefter skapas metadata och index och kataloger uppdateras. Den första iterationen i Data Warehouse-utvecklingen är nu klar för analys (Inmon et al, 2005).

Spiral utvecklingsmetodik

Metodiken för utvecklingen av ett Data Warehouse som Inmon et al (2005) förespråkar kallas för *spiral development methodology*, eller spiral utvecklingsmetodik översatt till svenska. Den spirala utvecklingsmetodiken sträcker sig längre än bara utvecklingsmetodik på så sätt att det beskriver inte bara hur man utvecklar ett Data Warehouse, utan även hur man använder det. Den spirala utvecklingsmetoden skiljer sig ifrån migrationsplanen på flera sätt. Migrationsplanen beskriver generella aktiviteter medans den spirala utvecklingsmetoden beskriver

specifika aktiviteter och ordningen på aktiviteterna. Tillsammans innehåller de en komplett bild av vad som behövs för att skapa Data Warehouset. Inmon skriver att det klassiska tillvägagångssättet vid denna typ av utveckling är en sekventiell metod, även kallad vattenfallsmodellen. Den går ut på att man gör klart varje del för sig och när ett delmoment är utfört avslutar man arbetet med det och påbörjar nästa. Kravspecifikationen görs klart fullständigt innan man går vidare till analysen, som också görs klart fullständigt innan man går vidare till designfasen osv. Metoden som Inmon förespråkar innebär att man delar in utvecklingen i små iterativa processer, när en iterativ process är klar påbörjar man nästa (Inmon et al, 2005).



Figur nr7: En illustrerande bild som jämför Inmons spirala approach mot den klassiska vattenfallsmetoden.

Inmon hävdar att iterativ utveckling är uteslutande det bästa tillvägagångssättet vid utveckling av ett Data Warehouse. Anledningarna till detta är enligt Inmon följande: industrins "best practices" rekommenderar detta starkt, slutanvändaren är oförmögen att artikulera behov tills iterationen först är gjord, ledningen engagerar sig inte fullt ut förrens resultaten är påtagliga och uppenbara och det finns ett behov att se synliga resultat snabbt (Inmon et al, 2005).

Datadriven metodik

Att en metodik är data-driven beskriver Inmon med två faktorer som särskiljer metodiken. En data-driven metodik har inte ett applikation-per-applikation-synsätt. I stället för att bygga systemet runt befintlig data, så bygger man på den befintliga datan. Den andra aspekten som skiljer en datadriven metodik ifrån ett vanligt applikationssynsätt är att fokusering på centraliseringen av organisationens data. Ett beslutstödssystem har en annorlunda utvecklingslivscykel än ett traditionellt operativt system. Utvecklingscykeln i ett operativt system börjar med behov och slutar med programkod, medans utvecklingscykeln i ett beslutstödssystem börjar med data och slutar med behov enligt Inmon et al (2005).

3.2.4 Användaren

Enligt Inmon et al (2005) är användaren av ett Data Warehouse i högsta grad beslutstödsanalytiker, men också först och främst en affärsman snarare än tekniker. Huvudsyftet för en BI-analytiker är enligt Inmon att definiera och upptäcka vad för information som används för beslutstöd av verksamheten. En BI-analytiker har enligt Inmon ett synsätt för tillvägagångssättet att ”Ge mig vad jag säger att jag vill ha, sen berättar jag vad jag verkligen vill ha.” Genom att titta på rapporter och analyser kan BI-analytikern undersöka möjligheterna för ytterligare information för beslutstöd. BI-analytikers attityd och inställning till vad för analyser och rapporter som bör göras är viktig, och bör tänkas på utifrån aspekter som ”befogad” och ”genomträngande” av datan. Hur noggrant detta görs har en stor effekt på hur Data Warehouset utvecklas och hur system som använder Data Warehouset arbetar (Inmon et al, 2005).

3.2.5 Datakvalitet

Trovärdighet

Inmon et al (2005) tar upp begreppet Living Sample Database, som är ett begrepp som egentligen har som syfte att snabba upp svarstiderna på statistiska frågor som ställs mot enormt många poster. Det går ut på att vid frågor som exempelvis ”Hur stor andel bilförare är män i Sverige?”, den frågan skulle då bli ställd mot cirka 9 miljoner poster. Lösningen på detta problem är att man i databasen innehåller ett slumpat urval, på kanske 100 000 poster som representerar de 9 miljoner posterna som befolkningen faktiskt är på. Tack vare detta snabbar man upp frågorna avsevärt, men sänker trovärdigheten något. Detta är en balansgång mellan svarstider och trovärdighet.

Tillgänglighet

Inmon menar att datans tillgänglighet har påverkan på datakvaliteten, anledningen till detta är att datan blir lättare att undersöka, övervaka och analysera efter fel. Övervakning är det bästa sättet att hitta vilket operativt system som producerar data med låg kvalitet (Inmon et al, 2005).

3.2.6 Systemkvalitet

Som vi har nämnt förespråkar Inmon ett Data Warehouse med en komplett datamodell för hela organisationen, även kallad den atomära nivån. Detta resulterar i en stor flexibilitet av dataomfattning. Datan finns integrerad och färdigbehandlad på den atomära nivån, det är bara att hämta in den till de Data Marts som behöver datan (Inmon et al 2005).

Förändring av data över tiden

Kimball har myntat ett välkänt begrepp kallat *Slowly Changing Dimensions*, vilket behandlar problemet med förändring av data över tiden i en dimensionsmodellerad databas. Inmon tar upp CIF´s motsvarighet till *Slowly Changing Dimensions* i en artikel i DM Review. Inmon tar upp ett antal sätt att hantera problemet, där det första och mest ingående sättet är att samla historisk data med hög detaljnivå. Man sparar helt enkelt så mycket information som möjligt, när väl en ändring sker så finns det redan information om hur tillståndet på datan var innan och efteråt och därav går det att se när ändringen skedde och vad den innebar. Inmon tar även upp en metod för att hantera förändring av referensdata över tiden (exempelvis på referensdata kan vara att

det står GM istället för General Motors). Hans förslag i artikeln på det lättaste sättet att hantera detta problem är att systematiskt spara ner referensdatatabellen med alla referensbeskrivningar i med jämna mellanrum. Ett problem med detta sätt är om en referens hinner ändras och ändras tillbaka mellan två sparade referenstabeller. Inmon föreslår då att man sparar referenstabellen exempelvis årligen och tillsammans med den sparar man ner alla ändringar som görs i referenstabellen. Den tredje och sista datan som ändras över tiden är metadata. Enligt Inmon är metadata den mest problematiska datan att hantera förändring för över tiden. Även här föreslår han att man bör spara metadata och dess förändringar över tiden för att kunna gå tillbaka och kolla hur den såg ut vid en tidigare tidpunkt (Inmon et al, 2005)

Omstrukturering

Inmon menar att en bieffekt av att flytta datan (och därmed eliminera bulkdata) från produktionsomgivningen till ett Data Warehous är att det underlättar korrekthet, restrukturering, övervakning och indexering. Resultatet av elimineringen av bulkdata i produktionsomgivningen är att datan blir mer formbar. Ytterligare en viktig aspekt med denna företeelse är att man flyttar den informella behandlingen av datan bort ifrån produktionsomgivning till Data Warehouset. Inmon menar att informationsbehandling överlag alltid utsätts för konstant förändring, vilket har effekt på den informella databehandlingen, mycket av vad som kallas underhåll och drift är just informell databehandling. Genom att flytta ut stora delar av den informella databehandlingen ut från produktionsomgivningen till Data Warehouset underlättar man driften av produktionsomgivningen enormt. Inmon hävdar att man underlättar även omstrukturering av produktionsomgivningen (och därmed underlättar förändring) tack vare att den är mindre, enklare och mer fokuserad (Inmon et al, 2005).

Förändrade användarbehov

Eftersom Inmon väljer att skapa en atomär nivå med en organisationsomspännande datamodell på detaljerad nivå kommer förändrade användarbehov att vara relativt smidigt att hantera. Exempelvis om ett nytt Data Mart skall utvecklas, finns all data redan tillgänglig i den form som behövs i den atomära nivån (Inmon et al, 2005).

Detaljerad och summerad data

Inmon löser problemet med kompromissen mellan detaljerad och summerad data genom att dela in ett DM i flera granulära nivåer. Exempelvis en summerad och en detaljerad, på så vis är det möjligt att ställa både detaljerade och omfattande frågor som kräver summerade poster. Anledningen till detta är att om man vill ställa en fråga baserad på enormt många poster, skulle det bli alldeles för resurskrävande att ställa en sådan frågan mot en databas med hög granularitet (Inmon et al, 2005).

3.3 Kimballs lösning

3.3.1 Filosofi

Redan från första kapitlet i boken ”The Data Warehouse Toolkit” pratar Kimball (2002) om affärsnyttan med Data Warehouse och att man måste se det ur företagets perspektiv. Data Warehouse handlar således om mycket mer än bara teknik och modellering. Kimballs mål med ett Data Warehouse är att:

Göra organisationens information lättillgänglig
Det som finns i databasen måste vara lätt att förstå för både utvecklaren och slutanvändaren och det åstadkommer man genom att namnge ett Data Warehouse på ett meningsfullt sätt. Det är vanligt att slutanvändare vill kombinera datan på olika sätt när man gör sökningar (slicing & dicing). Andra viktiga faktorer är att verktygen är enkla och att frågor returneras snabbt till användarna.

Presentera organisationens information på ett konsistent sätt
Datan som presenteras måste vara trovärdig. Informationen från en affärsprocess måste kunna matcha information från en annan. Exempelvis måste två mått betyda samma sak och göra samma sak annars ska de namnges på ett annat sätt. Konsistent information innebär att datan är av hög klass.

Låta Data Warehouse vara anpassningsbart
Förändringar i ett företag är oundvikligt och ett Data Warehouse måste då kunna klara av dessa förändringar. Det kan vara saker som att slutanvändarnas informationskrav ändras eller att ny teknik införs. Existerande data ska alltså inte förändras om man börjar ställa nya frågor eller om man lägger till ny data i ett Data Warehouse.

Skydda informationen
Ett Data Warehouse innehåller ofta information som är väldigt känslig som t.ex. vilket pris man tar och till vem man säljer. Det gäller att vara medveten om detta och försöka skydda denna information när den e lagrad i ett Data Warehouse.

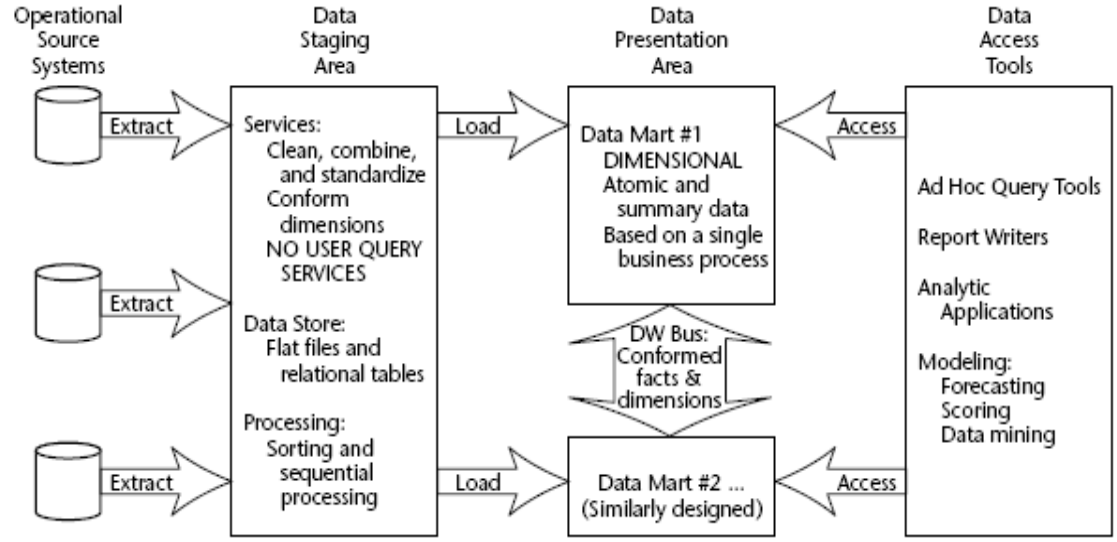
Data Warehouse måste vara bas för förbättrat beslutsfattande
Hela grundidén med att bygga ett Data Warehouse är att det ska underlätta beslutsfattande och det man då bygger är ju ett beslutstöd. Viktigast av allt är då att ett Data Warehouse innehåller rätt information.

Organisationen måste vara mottaglig för Data Warehouse om det ska fungera
Detta är en av de viktigaste punkterna som företag bör belysa. Det spelar ingen roll hur bra lösning man bygge rom man inte får organisationen att använda den. Till skillnad från exempelvis operationella system så är ett Data Warehouse frivilligt att använd.

Att lyckas med ett Data Warehouse-projekt innebär alltså att man både måste ha goda tekniska kunskaper såväl som organisatoriska och affärsmässiga kunskaper.

3.3.2 Arkitektur

Det finns olika sätt att bygga ett Data Warehouse på och nedan presenteras den modell som Kimball (2002) förespråkar. Modellen består av fyra stycken komponenter (Figur nr9) med start från vänster; Operational Source System, Data Staging Area, Data Presentation Area och Data Access Tools.



Figur nr9: Visar de olika komponenterna i ett Data Warehouse

Först hämtas informationen från Operational Source System (Extract) och lagras i Data Staging Area där den manipuleras och behandlas (Transform) för att sedan skickas till Data Presentation Area (Load). Datan som behandlas i Transformprocessen bör enligt Kimball modelleras enligt principerna för dimensionsmodellering till skillnad från traditionell relationsdatabasmodellering (ERD-modellering) som exempelvis Inmon et al (2005) förespråkar. Egentligen ingår även relationsdatabaser också i dimensionsdatabaser så en mer korrekt benämning (som också Kimball anser) det vore att säga att databaserna är relationella men starkt denormaliserade. När datan sedan laddas, skickas informationen till olika Data Marts. Varje Data Mart måste sedan indexera den data som tillkommit för att det ska gå att ställa frågor mot den.

I Data Presentation Area är all information lagrad i detaljerad och atomisk form som man sedan kan köra frågor mot. Att den är detaljerad gör att man kan ställa mer komplexa frågor. Det är det här som användarna av systemet ser och använder för att få fram olika typer av information via t.ex. Data Access Tools som beskrivs i nästa stycke. Data Presentation Area består egentligen av flera databaser eller flera *Data Marts*. Varje Data Mart representerar en affärsprocess i företaget. Detta medför att man kan uppnå en synergieffekt när man ställer frågor mot databasen då de olika Data Marts är kopplade till varandra via en Data Warehouse Bus som beskrivs senare. Om datan i Presentation Area är baserad på relationsdatabaser då är dimensionsmodellerna gjorda som ett *start schema* och om datan är baserad på multidimensionella databaser eller OLAP då är datan lagrad i en *kub.*

Data Access Tools är den sista komponenten i ett Data Warehouse. Data Access Tools är olika verktyg eller program som man använder för att ställa färdiga frågor mot Data

Staging Area'n. Dessa verktyg kan vara allt ifrån väldigt enkla till väldigt komplexa som endast används och förstås av ett fåtal användare.

The Data Warehouse Bus Architecture

Ett av huvudmålen med Kimballs syn är att man vill skapa integration mellan olika affärsprocesser eller Data Marts. För att uppnå detta presenterar Kimball The Data Warehouse Architecture Bus. En Bus kan ses som flera kanaler för att utbyta information men endast om datan är konsistent. Igrund och botten måste alltså nycklar, radnamn, attributdefinitioner och attribut värden vara angivna på samma sätt i alla Data Marts för att man ska kunna använda informationen. Ett exempel skulle kunna vara ett produktid för en produkt. Id't måste då vara likadant för de olika enheterna eller affärsprocesserna genom olika avdelningar. En produktutvecklare med produkt x som har produktid 123 måste referera till exakt samma produkt som en försäljare har som har produkt med produktid 123. När detta mål är uppnått kallar Kimball detta för *Conformed Dimensions* eller *Shared Dimensions*. Det kan vara svårt att göra denna data konsistent eftersom Data Martsen kan vara uppbyggda av olika utvecklare som namngivit informationen på olika sätt. För att lättare kunna dela in processer och dimensioner så använder man sig av ett verktyg som kallas för det Bus Matrix.

3.3.3 Utvecklingsmetodik

Utvecklingsprincipen går ut på att man bygger en Data Mart i taget. Den dimensionsbaserade designprocessen är enligt Kimball (2002):

Select Business Process

Först ska man välja en affärsprocess. En affärsprocess är en aktivitet på ett företag som på något sätt involverar en samling av data. Exempel på affärsprocesser skulle kunna vara inköpsprocessen eller lagerhantering. Det gäller att identifiera vilka processer som är beroende av varandra, exempelvis en beställning är kopplad till försäljning. För att åstadkomma detta är det är viktigt att lyssna på användarna. Det är viktigt att förstå att en affärsprocess inte är synonym med vad man gör på en viss avdelning på företaget utan det är processen som står i centrum. Genom att välja att dela upp det i processer istället för avdelningar så undviker vi problemet med inkonsistent data som kan uppstå. Som exempel kan man ta Försäljningsavdelningen och Marknadsföringsavdelningen som båda vill ha tillgång till ordrar, då är det bättre att modellera orderprocessen istället för Marknadsfäringsavdelningen och Försäljningsavdelningen som båda vill ha tillgång till orderinformation. Den första processen man bör identifiera är processen som är mest betydelsefull:

” The first dimensional model built should be the one with the most impact—it should answer the most pressing business questions and be readily accessible for data extraction.” (Kimball, 2002, s.33).

Declare Grain

Här väljer man ut data ur affärsprocessen för att sedan bestämma hur detaljerad beskrivningen av den ska vara. På den absolut längsta nivån är datan atomisk och kan inte längre delas in mer. Man strävar efter att dela in datan i anatomisk data då detta medför att man kan kombinera informationen bättre och få fram tydligare svar när man gör frågor. Exempelvis kan man få fram relationer mellan olika delar som man

tidigare inte visste att man hade. (Exempel blöjor och öl). Det handlar om att definiera vad varje faktarad ska innehålla för information.

Choose the Dimensions
Här gäller det att välja ut dimensionerna och det gör man genom att titta på vad man har för faktatabellrader. Vidare kan man fråga sig hur datan som returneras i en fråga ska beskrivas. Exempelvis att man vill sortera ett resultat efter produkt eller ett visst datum, då skapar man en produktdimension och en kunddimension (Kimball, 2002).

Identify the Facts
Här gäller det att ta reda på vilka numeriska fakta man ska lägga in i faktatabellerna. För att ta fram dessa så kan man fundera på vad det är man är intresserad av att mäta. Ofta vill företag på något sätt kunna mäta företagets prestanda, t.ex. antalet sålda produkter. Viktigt här är att man skapar fakta som stämmer överens med den ”grain” vi valde att använda oss av (Kimball, 2002).

3.3.4 Användaren

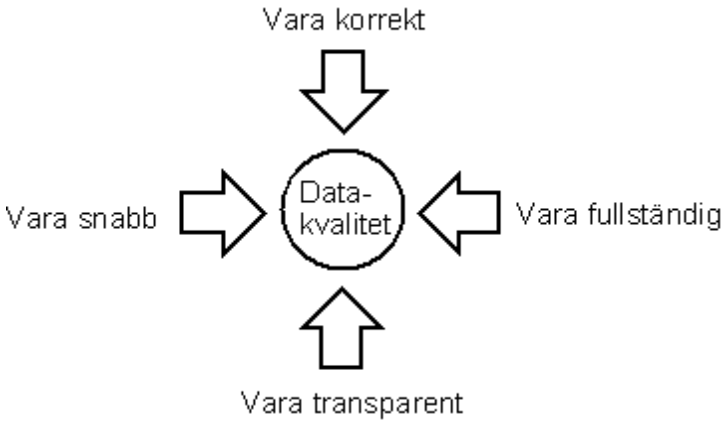
Kimball har en mycket klar syn på att användaren måste mycket enkelt och smidigt kunna använda verktyg som hämtar information och data ur ett Data Warehouse. Kimball pratar om ”Business Users” som den främsta användaren för att ta fram beslutsfattande data. Datan i ett Data Warehouse måste vara lätt att förstå, mycket intuitiv och tydlig för användaren och inte bara för utvecklaren menar Kimball. Med begreppen lätt att förstå och tydlighet menar Kimball att datan måste vara strukturerad på ett sådant sätt att användaren kan se en tydlig logik, samt att beteckningen av datan är tydlig. Som Data Warehouse Manager är det viktigaste ansvaret enligt Kimball att datan ska vara konsistent och innehållsrik (Kimball, 2002).

3.3.5 Datakvalitet

Kimballs sätt att nå hög datakvalitet går ut på att dels ta itu med designmål för hur datakvalitet ska hanteras samt hur man ska tvätta datan i ETL-processen. För att tvätta datan presenterar Kimball ett antal subsystem som ska hantera hanteringen av datakvaliteten (Becker, 2007). Dessa subsystem ska hjälpa till att skapa metadata för att diagnostisera källsystemproblem. Detta menar Kimball ska leda till att datakvaliteten blir högre. Nedan beskrivs först designmål och därefter subsystemen.

Designmål

Hur bra kvalitet man vill ha på datan beror bl.a. på hur man balanserar fyra styrande faktorer (Figur nr10). De fyra olika faktorerna är *fullständighet, snabbhet, korrekthet, transparens. Fullständighet* har att göra med hur omfattande man kan detektera, tvätta och dokumentera fel som kan uppstå. *Snabbhet* har att göra med hur snabbt ETL processen klarar av att bearbeta datan. *Korrekthet* nås genom dels ETL verktyg och dels genom att man går in och ändrar i befintliga källsystem. *Transparens* har att göra med hur enkelt det är att ändra i befintliga källsystem eftersom det kräver en hel del engangemang och insikt.



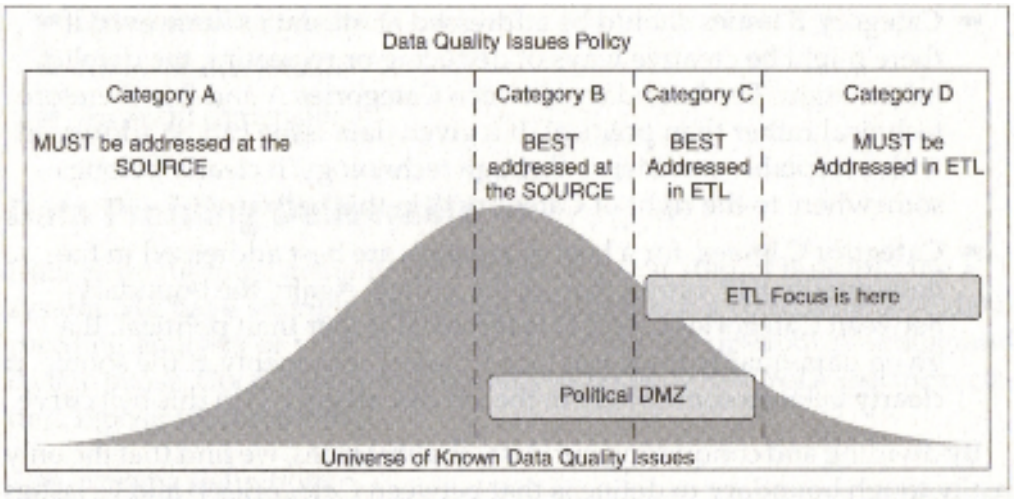
Figur nr 10: Fyra faktorer som ligger i konflikt med varandra och som avgör kvaliten på datan.

Fullständighet kontra Hastighet
Som det ser ut idag tar ETL-processen lång tid eftersom man bearbetar enorma mängder data och Kimball menar att man måste göra en slags ”*trade-off*” mellan tid och datakvaliteten där man själv efter företagets behov måste avgöra vad som är viktigt (Kimball, 2004).

Korrekthet kontra Transparens
Korrekthet slöar ner verksamheten eftersom man måste sätta sig in verksamheten och anpassa källsystemen. Om man däremot inte kommer åt källsystemen och ändrar datafelen där så går det inte att få fram bra data i Data Warehouset. Lösningen på detta problemet är att dokumentera förändringar, standarder etc. som görs i källsystemen (Kimball, 2004).

Roller
Dessa fyra olika faktorer bör analyseras samt utvärderas och enligt Kimball måste man först bestämma vem som ska styra och kontrollera datakvaliteten. Detta gör man genom att tilldela personer olika roller. Den viktigaste rollen är *The Information Steward.* Den som har rollen som Information Steward är ansvarig för att bestämma vilken strategi man ska använda, vilka mål man har, vilka datakällor man ska välja, publicerar metadata etc. (Becker, 2007). Tanken är att Information Steward ska hålla koll på vilken data som är viktig för företagen att ta fram samt styra denna.

Ett sätt att ta sig an problemet med datakvalitet är enligt Kimball att kategorisera problem för datakvalitet på olika nivåer (Figur nr11). I kategori A har man problem som t.ex. felaktig data om en kund eller avsaknad av information. Detta är inget man kan lösa med ETL verktygen utan vad man måste göra är att gå till källsystemen eftersom det är där problemet ligger och det är under denna kategori som de flesta datakvalitetsproblem faller under. Iden sista kategorin D, har man problem som går att lösa rent praktiskt via ETL verktygen (Kimball, 2004).



Figur nr11: Figuren visar de olika kategorierna där man kan behandla datakvaliten.

Det enda svåra när det gäller kategoriseringen är att identifiera om problemet ligger i källan eller om det går att lösa via ETL-processen.

Subsystemen

Det finns 38 subsystem (Kimball, 2004) som Kimball förspråkar vid ETL processen och vi tar här upp sex av dessa som är en del av Transformdelen i ETL-processen.

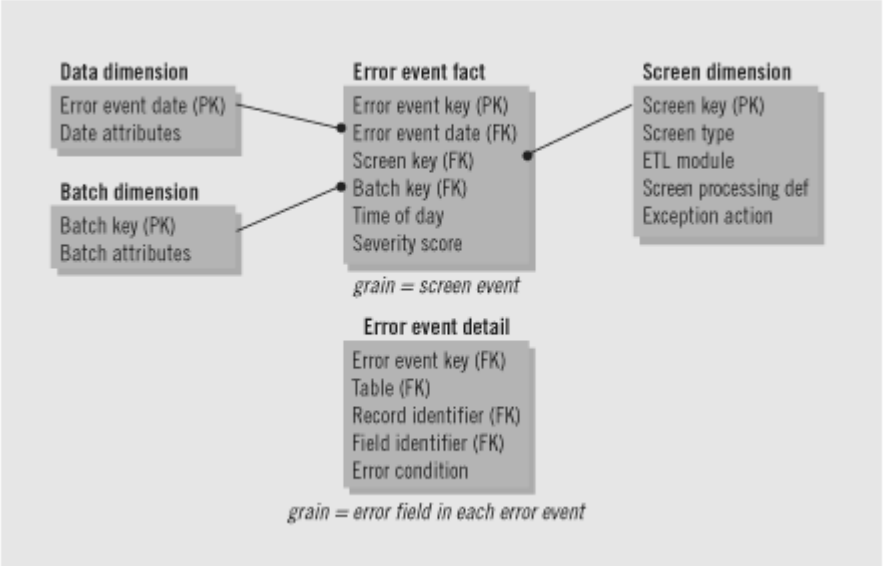
Data cleaning system (Subsystem 4)

Här visar Kimball (2004) metoder för att hålla kolla på på vad för typ av datakvalitetproblem man har, när man kollade på dem, vad man tittar på, samt resultatet. Vad man vill åstadkomma med detta är en kartläggning eller profilering av datakvaliteproblemen som man sen kan använda som underlag för hur man ska förbättra datakvaliten. Genom att göra en profilering av felen som uppkommer vid ETL-processen kan man identifiera exempelvis vilka källsystem som genererar flest fel eller om datakvaliten blir bättre eller sämre. Detta är frågor som Information Steward (som diskuterades tidigare) kommer att behöva titta på när han ska forma strategin för datakvalitetshantering. De tre system som ingår i Data cleaning systemet är Data Profiling System, Error event handler samt Audit dimension assembler. Dessa beskrivs nedan.

Data profiling system (Subsystem 3)

Kimball menar att man först bör göra en komplett kartläggning av datan man använder sig av i källsystemen. Det gäller att skapa metadataförråd som beskriver datakällorna, affärsmål, tabellnamn, domäner etc. Detta är något man bör göra innan själva ETL-processen påbörjas samt att profileringen ligger som grund för de två nedanstående subsystemen (Error event handler samt Audit dimension assembler). Dataprofileringen bör avgöra om källsystemet ska användas eller om man måste gå tillbaks och behandla felen i källan alternativt om man ska fixa problemen i ETL-processen (Kimball, 2007).

Error event handler (Subsystem 8)
I nästa steg bör man bygga en Error event table som är en dimensionstabell byggd som ett starschema (Figur nr12). Dimensionstabellen innehåller en faktatabell med olika errors och tillhörande dimensioner som innehåller information om dessa errors. Varje fel som bildas under datatvätten (data cleaning subsystem) läggs till som en rad i faktatabellen.



Figur nr12: Centralt i den här bilden är Errorfaktatabellen. Dimensionerna kommer sedan att säga något om själva felet som t.ex. när felet upptäcktes (Error event date), under vilket BATCH-jobb som felet påträffades och vilken screen som skapade felet.

Audit dimension assembler (subsystem 6)
Audit Dimension är en dimension som ger en övergripande beskrivning av datakvaliten i en faktatabell. När errorfaktatabellen beskriver vilka fel som uppträder under ETL-processen så beskriver Audit Dimension hur själva ETL-processen gått, exempelvis hur ofta errors inträffar eller hur många screens som misslyckats. Audit dimension innehåller också poäng på hur bra datakvaliten är eller snarare hur bra ETL-processen gått. Audit Dimensions är det sista man gör i varje process och där bör också stå beskrivningar av vilka ändringar man gjort i tabellen.

Quality screen handler (subsystem 7)
Screens är checkpoints och filter som används för att mäta datakvalitet. En screen är helt enkelt ett test som utförs under ETL-processen som raporterar in fel i error event tabellen (som beskrevs tidigare) och som beslutar om ETL processen ska fortsätta eller om felen är så omfattande att man måste stoppa ETL-processen.

Avvikelser
Under ETL-processen kommer det att förekomma avvikelseri databasen som kan vara svårt att se. Ett exempel skulle kunna vara att räkna antalet rader (Kimball, 2004). Ett annat exempel skulle kunna vara en tabell som visar alla de fulltidsanställdas löner (Tabell nr3) men där en av de anställda verkar tjäna en bårkdel av va de andra tjänar. Detta är något som enkelt går att upptäcka i en databas med hjälp av en enkel SQLsats.

Namn	Lön
Eva	5
Kalle	24000
Olof	22000
Sara	23000
Bengt	25000

Tabell nr3: Tabellen visar att datan förmodligen är fel i tabellen.

Man kan kategorisera dessa typer av avvikelser i tre olika typer av screens, Column screens, Structure Screens, Business value screens.:

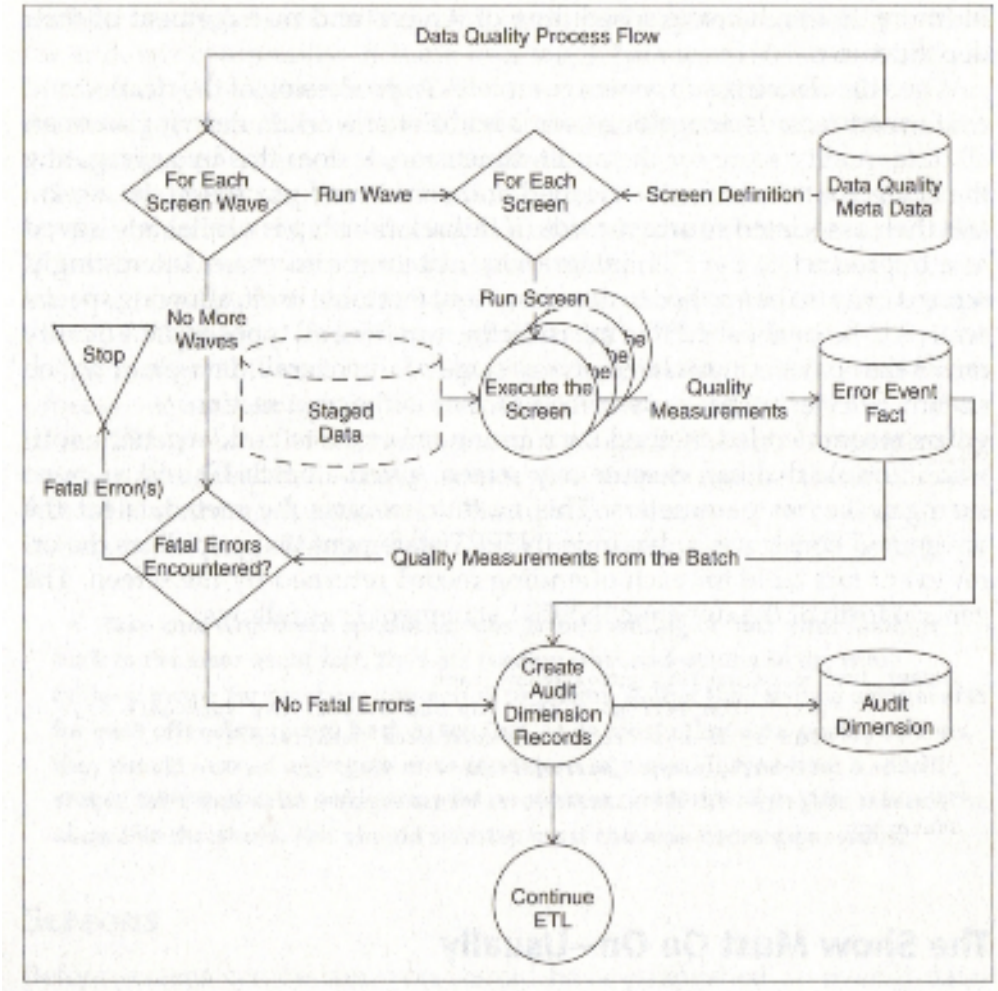
- **Column Property Enforcements**
  Det här testet kollar ett antal olika fel som exempelvis om tabellerna innehåller NULL-värden.

- **Structure Enforcement**
  Fokus i detta testet ligger i att titta på struktur och hur kolumner hänger samma med varandra via relationer. Man kan t.ex. testa två olika fält för att se om dessa bildar en en hierarki. Denna screen testar också primär/sekundärnycklar i två tabeller.

- **Data and Value Rule Enforcement**
  Detta är en mer komplex screen som testar datan efter olika förhållningssätt. Exempelvis kollar man och testar att en kund inte är både av typen Man och Kvinna på samma gång. Mer komplexa tester skulle kunna vara ett företag som enbart ger sina kunder VIP-status efter att dom köpt för en viss summa. Testet skulle då kunna vara att se om VIP-kunder verkligen betalat in den summa som krävs. Andra typer av tester skulle kunna vara att se om en pojke i databasen verkligen kan hta Lisa, vilket skulle kunna vara möjligt men detta borde i så fall generera någon form av varning.

Processen
Nedan (Figur nr13) visas hur flödet går genom processen (Kimball, 2004). Processen börjar med att ett antal screens eller ”error checks” ligger i en kö som väntar på att få köras. Varje screen står beskriven i screen-dimensionen som är en del av den error event faktatabellen som beskrevs tidigare. När en screens sedan körs och stöter på ett fel genererar detta en rad i error-eventtabellen. Metadatan om varje fel kommer sedan att beskriva hur alvarlit felet är. De alvarligaste felet kan vara så alvarliga att hela ETL-processen måste avbrytas och starta om. Exempel på sådana fel skulle kunna vara att hela delar av en viss tabell saknas, som exempelvis att data om dagens försäljning saknas. När varje data ”error check” har körts ställer man en fråga mot error event faktatabellen för att se hur alvarliga fel som inträffat. Finns det inga fel fortsätter processen och i annat fall om alvarliga fel påträffas stoppas processen och någon lämplig person kontaktas som exempelvis en kvalitetsansvarig eller Information Steward. För att processen ska bli optimal föreslår Kimball att man kör flera screens som kan köras parallellt. mer konkret menar Kimball att man bör köra en

”våg” av screens åt gången, dvs ett antal screens körs parallellt och när denna våg är klar då kommer nästa våg av screens. Ivilken ordning man tar screens'en bestäms av de denfinitioner man gjort i metadatan. När processen är klar och datan är tvättad och bekräftad gör subsystemet en beräkning av den totala datakvaliten. Det gör subsystemet genom att aggregera alla rader i error event tabellen.



Figur nr13: Bild som visar hur hela processen för att hantera datakvalitet ser ut.

Data conformer (subsystem 5)
För att integrera data på ett bra sätt använder sig Kimball av *Conformed dimensions* (Kimball, 2004). Conformed dimension innebär att man strukturerar upp data från olika källor och sammanför denna på en konsistent form, exempelvis att två olika system som benämner kön enligt (Kvinna, Man) och (K, M) går samman och enas om en enda benämning.

3.3.6 Systemkvalitet

Flexibilitet

Kimball (2004) visar på tre olika typer av tekniker för att hantera Slowly Changing Dimensions (SCD) samt ett par hybrider.

Type1: Skriva över värden

Med den här tekniken ersätter man helt enkelt det gamla värdet i en dimensionstabell med ett nytt värde. Det här sättet gör att man alltid har de senaste ändringarna i databasen men den stora nackdelen är att man inte kan se på historisk förändring över tiden. Det här sättet är snabbt att implementera och passar bra om exempelvis datan i databasen varit felaktig från början. Däremot passar tekniken mindre bra om man har exempelvis en produkt som tillhör en viss avdelning. En ändring i alla tabeller från Education till Strategy (Tabell nr4) kommer göra att man får en missvisande bild av vilken avdelning produkten tillhör eftersom det kommer se ut som produkten alltid tillhört Strategy. SCD av Typen2 och Typ3 hanterar denna problematik.

Produktnyckel (Surrogatnyckel)	Produktnamn	Avdelning	EAN-kod (Naturlig nyckel)
12345	IntelliKidz 1.0	Strategy	ABC922-Z

Tabell nr4: Visar en rad i en databastabell där en produkt bytt avdelning från Education till Strategy

> *"The type 1 response is easy to implement, but it does not maintain any history of prior attribute values".* (Kimball, 2002, s.97)

Type2: Lägga till en dimensionsrad

Den här typen av SCD hanterar historisk data genom att man lägger till en likadan dimensionstabellrad som man haft tidigare. Denna tabell (Tabell nr5) ska innehålla den nya informationen som förändrats samt ha en ny surrogatnyckel. Den naturliga nyckeln måste vara oberörd eftersom man måste kunna urskilja ny information från gammal även om det är tal om samma produkt.

Produktnyckel (Surrogatnyckel)	Produktnamn	Avdelning	EAN-kod (Naturlig nyckel)
12345	IntelliKidz 1.0	Education	ABC922-Z
54321	IntelliKidz 1.0	Strategy	ABC922-Z

Tabell nr5: Visar hur en databastabell kan se ut när man använt sig av SCD type 2.

> *"The type 2 response is the primary technique for accurately tracking slowly changing dimension attributes. It is extremely powerful because the new dimension row automatically partitions history in the fact table."* (Kimball, 2002, s.98)

Det kan tyckas vettigt att tidsstämpla dessa förändringar någonstans i dimensionstabellen för att veta vad som är historisk data och vad som är aktuell data. Kimball (2002) menar att detta inte behövs göra då datan redan blir tidsstämplad i

Staging Arean. En nackdel med Typ2 är att det blir en en ny rad i databastabellen för varje ändring som görs vilket förstora upp databasen.

Type3: Sparar förändringen
SCD av Typ2 är bra om man ska se på historisk förändring men det passar sig mindre bra att knyta ihop attribut med tiden. Gör man en sökning på Avdelning kommer man endast att få fram IntelliKidz efter att ändingen gjordes och all information när IntelliKidz tillhörde Education uteblir. För att lösa detta problemet skapar man istället ett extra attribut. Det extra attributet skulle då få namnet Tidigare Avdelning (Tabell nr6) som då skulle innehålla Education. Detta sätt att hantera SCD är bra ifall man vill ha en bra översikt av vad som har hänt. Har man information som ändras ofta med tiden blir emellertid Type3 dåligt eftersom varje ny förändring inte kommer att gå att spåra som man exempelvis kan med Typ2 samt att man måste namnge varje attribut som t.ex. Tidigare1, Tidigare2, Tidigare3 etc.

Produktnyckel (Surrogatnyckel)	Produktnamn	Avdelning	Tidigare Avdelning	EAN-kod (Naturlig nyckel)
12345	IntelliKidz 1.0	Strategy	Education	ABC922-Z

Tabell nr6: Visar hur en databastabell kan se ut när man använt sig av SCD type 3.

> *"The type 3 slowly changing dimension technique allows us to see new and historical fact data by either the new or prior attribute values."* (Kimball, 2002, s.101)

Enligt Kimball (2002) så är ingen av SCD typerna bäst eller sämst utan det är behovet som styr vilken typ man ska använda.

3.4 Ramverk för viktiga framgångsfaktorer

Vi har utifrån tidigare forskning gjord av Wixom och Watson (2001) konstaterat att datakvaliteten och systemkvaliteten är två mycket viktiga faktorer för huruvida ett Data Warehouse projekt kommer att lyckas att implementeras och utvecklas. Eftersom det idag finns i huvudsak två sätt att bygga Data Warehouse på, nämligen Inmons sätt och Kimballs sätt, har vi genom ett ramverk försökt att analysera skillnaden för hur de båda lösningarna ser på datakvalitet och systemkvalitet. Vi har tidigare redogjort för hur Inmon respektive Kimball ser på hur systemkvalitet och datakvallitet hanteras men vill gärna undersöka hur det upplevs av dem som också arbetar med deras metoder.

Vi har också genom en fokusgrupp tillsammans med medarbetare på WM-Data i Göteborg och Malmö utvecklat ramverket (Tabell nr7) med hjälp av ytterligare tre faktorer. Dessa faktorer är organisatoriska och hanterar acceptans, ansvar och kompetens vilket lyfter fram ytterligare ett perspektiv av de båda lösningarna. På WM-Data arbetar de i Göteborg till största delen utifrån Kimballs synsätt medens de i Malmö arbetar utifrån Inmons synsätt. Detta har gjort att vi med hjälp av de anställda i de respektive städerna har kunnat jämföra hur Inmons och Kimballs synsätt upplevs i verkligheten genom vårt ramverk.

Vårt ramverk beskrivs här nedan och följs av en beskrivande text för de tre perspektiven och undersökningsaspekterna.

Perspektiv	**Undersökningsaspekter**
Organisation	1. Acceptans
	2. Ansvar
	3. Kompetens
Systemkvalitet	4. Flexibilitet - Förändring av data över tiden (SCD)
	5. Flexibilitet - Förändrade användarbehov
	6. Flexibilitet - Förändrade affärsvillkor
	7. Flexibilitet - Hantering av tekniska villkor
	8. Integration
Datakvalitet	9. Korrekt data
	10. Konsistent data
	11. Fullständig data

Tabell nr7: Ramverk med framgångsfaktorer

3.4.1 Organisation

De organisatoriska faktorerna som vi kommit fram till via möten med representaner för WM-data är egenskaper som anses vara viktiga att titta på därför att de är knutna till om man väljer att bygga sitt Data Warehouse med Kimballs eller Inmons metod. Faktorerna är knutna till användarna samt driftpersonal och vad som krävs av dessa vilken i sin tur är en fråga om vilken arkitektur som ligger i botten.

1. Acceptans

Beskriver hur väl ett Data Warehouse accepteras av slutanvändaren (end-user) och power-user eller super-user (den som skapar kuber som ligger till underlag för rapporterna) samt hur väl man kan ta till sig systemet. Acceptans kan bero på struktur av datamodellering samt hur lätt det är att generera rapporter utifrån datastrukturen.

2. Ansvar

Intressekonflikter kan lätt uppstå vid införandet av nya arkitekturer. Vem som äger datan, och vem som skall administrera och äga ansvaret kommer vi ta upp här. Datan kan ibland lagras över flera avdelningar i dimensioner men också i avdelningsspecifika Data Marts. Inmon och Kimball har olika tekniska sätt att lagra data och detta kan möjligtvis bidra till ansvarskonflikter. Vem som skall stå för kostnaden av ett Data Warehouse kan också vara ett problem, om avdelningsspecifik data används av andra avdelningar.

3. Kompetens

Har att gör med vilken kompetens för drift och underhåll som krävs och om någon av metoderna kräver mer eller mindre underhåll samt om den ena eller andra metoden är mer komplex.

3.4.2 Systemkvalitet

Systemkvalitet fokuserar på själva systemet. DeLone och McLean (1992) hävdar att de mest använda prestandamåtten inom systemkvalitet inkluderar integration, flexibilitet, responstid och pålitlighet. För beslutstödsystem är integration och flexibilitet extra viktiga (Vandenbosch och Huff, 1997). Hög systemkvalitet är en av de största fördelarna med ett Data Warehouse eftersom det tillhandahåller infrastrukturen som integrerar data från olika källor och flexibilitet som stödjer aktuella och framtida användare och applikationer (Gray och Watson, 1998; Sakaguchi och Frolick, 1997).

4-7 Flexibilitet

Flexibilitet är en del av systemkvalitet och beskriver hur väl faktorer som påverkeras av inre och yttre omständigheter klarar av att hanteras av tekniken. Graden av flexibilitet kan variera och hanteras på lite olika sätt och det kommer vi under denna punkt att ta upp. Vandenbosch och Huff (1997) menar att flexibilitet ger beslutsfattare möjlighet att ändra i ett Data Warehouse när deras informationsbehov ändras. De menar också att flexibilitet existerar om datan inte är beroende av hur man har tänkt använda den. Externa faktorer som lagar och regler, samt politiska dimensioner kan bidra till att organisationer och verksamhetsområden måste förändras. Interna faktorer som att säljdistrikt slås ihop eller ändras, eller att en ny organisationsstruktur skapas kan också bidra till att man behöver ändra i datastrukturen.

Till flexibilitet hör faktorerna:
4. Förändring av data över tiden
5. Förändrade användarbehov,
6. Förändrade affärsvillkor,
7. Hantering av tekniska villkor som exempelvis inlåsningseffekter

8. Integration

System som integrerar data ifrån olika källor kan förbättra organisatoriska beslut (Wetherbe, 1991; Wyboand och Goodhue, 1995). Den önskade effekten av integrationen är att slå samman data ifrån källsystemen till ett Data Warehouse. Detta för att skapa konsekvent data för att möjliggöra för analysverktyg att ställa frågor mot datan.

3.4.3 Datakvalitet

Datakvalitet är kvaliteten på den data som finns lagrad i ett Data Warehouse (Wixom och Watson, 2001). Aspekter som påverkar kvalitén på datan kan vara hur korrekt den är, hur fullständig och hur konsistent datan är (Wixom och Watson, 2001 Citerar Lyon, 1998; Shanks och Darke, 1998). Anledningen till att vi väljer att titta på datakvaliteten är för att det finns forskning som har visat att datakvaliteten är en kritisk aspekt när man bygger ett Data Warehouse (e.g. Lyon 1998; Shanks och Darke, 1998). Kvalité är inget som är konstant utan beror på tolkning och är mätbart, därför har vi valt att använda oss av Juran's (1999) definition som säger att datan är av hög kvalité om den är anpassad för ändamålet vid beslutsfattande och planering. Nedan följer en kort beskrivning av de olika aspekterna som berör kvalitén på datan.

9. Korrekt data

Korrekt data har att göra med om det data som lagras stämmer överens med dess verkliga värde (Wang et al, 1995 Citerar Ballou et al, 1982). Med korrekt menar man att datan inte kan feltolkas samt att den är rätt. Ett exempel av data som kan feltolkas kan vara ett datum som är lagrat i en databas. I USA skriver man datum på formen MM-DD-YYYY och i Europa skriver man DD-MM-YYYY. Ett datum som är 12-10-2007 är alltså tvetydigt (svårt att skilja på dag och månad) och kan medföra problem när man ska integrera data som är skrivet på olika sätt om man inte känner till förutsättningarna. Inkorrekt data är data som innehåller fel information, exempelvis ett datum som borde vara 12-10-2007 är skrivet som 12-10-2006 i databasen.

10. Konsistent data

Det finns lägen där data som hämtas från olika system är korrekt men inkonsistent. Med inkonsistent menar man att datan inte är skriven på samma form vilket gör att den blir svår att sammanföra (Wang et al, 1995 Citerar Ballou et al, 1982). Exempel på inkonsistent data skulle kunna vara två olika system där det ena systemet använder ”Gbg” och där det andra systemet använder ”Göteborg”. Dessa problem ligger i att datakällan ser ut på ett visst.

11. Fullständig data

Datan anses ej vara fullständig om det t.ex. saknas en post eller rad i en databas (Wang et al, 1995 Citerar Ballou et al, 1982). Detta händer när man t.ex. felprogrammerat ett system som ska generera data automatiskt in i databasen. Ett annat tillfälle skulle kunna vara ett system som hämtar information från ett annat system där informationen inte finns tillgänglig

4 Resultat och Diskussion

För att öka läsbarheten i denna uppsats har varje framgångsfaktor behandlats var för sig, i den följden som presenterades i ramverket. Den undersökningen som gjordes med hjälp av intervju och enkät besvarades utifrån 11 frågor där varje fråga tog upp en utav ramverkets faktorer. I detta kapitel kommer för tydlighetens skull alltså varje faktor tas upp separat med ett tillhörande resultat och diskussion. Varje faktor kommer att inledas med en beskrivning av frågan, samt en kort sammanfattning av de slutsatser som har gjorts.

Respondenterna som svarat är från WM-Data i Göteborg och Malmö och har deras namn har förkortats till dess initialer. Varje respondents svar kommer redovisas separat för att i diskussionen dra paralleller och analyser. En mer ingående respondentpresentation finns i metodkapitlet.

4.1 Acceptans

Den fråga som ställdes för den organisatoriska faktorn ”acceptans” var menad att handla om hur systemet accepteras av slutanvändarna i form av ”power user” eller ”super user” där dessa är de användare som gör kuber mot Data Warehouset. Vår fråga till respondenterna var *”Hur väl hanteras systemet av slutanvändaren i form av power user eller super user?”*.

Resultat

JJ anser att användarna mycket lätt med hjälp av Kimballs metoder kan få fram rapporter. HB tycker också att det är lätt.

RC menar att acceptans för systemet till största delen handlar om hur väl användarna fått vara med i utvecklingsprocessen, och säger också att det skapar den nödvändiga utbildning och förståelse som krävs för att kunna använda systemet. RC menar att en mer IT-driven systemutvecklingsprocess som han anser att Inmon har, bidrar till att acceptansen för förankring av systemet kommer att vara svårare än för ett Kimballprojekt, som han anser är ett mer organisationspåverkande systemutvecklingsprojekt.

> *”Om man har en IT-driven systemutvecklingsprocess då kommer förankringsprocessen att vara jobbigare. Just eftersom användarna inte fått vart med i processen, utan skall bara ta ställning till ett färdigt system. I motsats till om användaren själva får vara med och ta fram kravspecifikationen och driva på utvecklingen. Det handlar om projektets förmåga, eller projektets förmåga att kommunicera vad det är man gör. Och där finns det skillnad mellan Inmon och Kimball. Inmon är ju ett mer IT-drivet systemutvecklingsprojekt, medens Kimball tar sitt utgångsläge i en mycket mer organisationspåverkande systemutveckling. Man frågar egentligen, man driver på mycket hårdare vad användaren vill ha, och så utvecklar man bara vad användaren vill ha. Inte vad de kanske skulle fråga efter sen.”(RC)*

RC tycker även att Kimballs dimensionsmodellering bidrar till en lättare förståelse om hur informationen i datalagret ser ut. Han menar att det till stor del gör att användare lättare kan ta till sig informationsmodellen och hur systemet kan användas.

> *”Man behöver inte snacka ett skit IT, utan man kan åskådliggöra väldigt tydligt även för icke IT-kunniga människor om hur informationen ser ut i datalagret” (*RC om varför dimensionsmodellering är bra).

En annan skillnad till varför RC anser att Kimball är enklare att förstå är Kimballs bus-matris. Han menar att detta är ett väldigt intuitivt och lätt sätt att förstå vilken data som finns, och vad som är gemensamma begrepp. RC menar vidare att Kimball har ett större fokus på användarvänlighet och förståelse, medans Inmon har ett mer datadrivet perspektiv.

> *”Så det är en kombination av starschema och projektet, eller systemutvecklingsmodellens förmåga att kommunicera vad man gör, vad systemet innehåller. Man ska inte underskatta detta, den är otroligt viktig egentligen. Annars har man byggt ett jättefint system, men ingen vet hur man ska använda det. Och har man fått med folk på den resan innan, så är det mycket lättare.”(RC)*

RC menar alltså att styrkan med Kimball är dels att det är star schema-modellerat samt att systemutvecklingsmodellen är mer kommunikativ och lätt att ta till sig för icke IT-kunniga.

PW anser även han att Kimballs fokus ligger mer på slutanvändarvänlighet. Han säger att Kimballs modell är väldigt lätt att förstå samt mycket tydlig. Anledningen är mycket tydliga dimensioner och starschema samt klara textbeskrivningar överallt. PW säger att starschema gör att det blir väldigt tydligt och att man inte behöver vara speciellt kunnig för att se datan på ett översiktligt sätt. Som slutanvändare, alltså användare som genererar rapporter, är det enklare för och de kommer också igång mycket snabbare säger PW.

KG svarar i enkäten enligt Inmons utvecklingsmetod (Itteration) och jämför det med Kimballs lösning med dimensionsmodellering.

> ***”****Iterations kan sägas stödja detta, även om dimensionsmodellering är mer rätt på sak.”(KG)*

JL anser att Kimball har bättre förutsättningar för slutanvändare som genererar rapporter, även kallade ”power-user”

> *”Om vi säger att slutanvändarna delas upp i två, ”power-user” och de som utvecklar modeller. Alltså de som skall bygga kuber. De som står och tittar rakt ner i databasen och bara ser massa tabeller och fält, då är Kimballs metod en fördel. Just eftersom starschema är mycket mer begripligt för människor att förstå än normaliserade databaser är.” (JL)*

UW anser inte det är någon skillnad mellan Inmon och Kimballs modeller i avseende av hur slutanvändaren uppfattar och accepterar systemet.

Diskussion
Vad man kan dra för slutsats av undersökningen är att Kimball har en klar fördel i denna fråga eftersom det är dimensionsmodellerat med starschema vilket gör datan mer överskådlig. Kimballs struktur där dimensionsmodellering används anser RC är enklare att ta till sig för icke IT-kunniga. Relationsmodellering som Inmon använder blir ofta mer komplexa i det perspektivet att man inte ser sambandet och översikten på lika tydligt sätt som för dimensionsmodellering. Kimball har där en fördel i och med att datan blir mer överskådlig. Hela Kimballs filosofi bygger på att det skall vara användarvänligt och ha en utvecklingsmetodik som innebär att användaren står i fokus, vilket också delas av både Inmon och Kimballs anhängare. JJ och HB har endast kunskap om Kimball men tycker att hans metod funkar bra. Att UW svarar att det inte är någon skillnad kan bero på att han uppfattat frågan fel. Det är mycket möjligt att han anset att vi menade slutanvändare som användaren av gränssnittet, vilket hanteras och ser likadant ut för både Inmon och Kimball.

4.2 Ansvar

Under denna rubrik är ansvarskonflikter i fokus, och med ansvarskonflikter menar vi konflikter som rör vem som äger datan, konflikter om vem som får stå för kostnaden för ett Data Warehouse o.s.v. Vår fråga till respondenterna var här: ”*Uppstår mycket ansvarskonflikter i samband med införandet och existensen av systemet? Och hur väl hanteras detta?”*.

Resultat
KG anser att Inmons Iteration innehåller hyfsade beskrivningar av olika roller för utvecklingsskedet av systemet. Roller för ansvar och förvaltning är det däremot inte så mycket fokus på menar han.

UW anser att metoden och modellen inte spelar så stor roll för ansvarskonflikter, utan att det snarare är andra faktorer som påverkar detta. Vilka faktorer har vi inte fått svar på.

JL menar att ansvarskonflikter vid införandet av systemet och ett Data Warehouse inte skiljer sig åt för Inmon och Kimball. Även om ett Data Warehouse delas in i avdelningsspecifika Data Marts, så används ändå data över avdelningar och ägandeskapet av viss data är därför ibland svår att bestämma anser JL.

> *”Men det är inte skillnad på modell. Just eftersom vi pratar om shared - dimensions, och i shared ligger då svårigheten med att definiera vem som har ansvaret över datan. Delat ansvar är ju inget ansvar, och det problemet finns i båda fallen.” (JL)*

JJ svarade 5 i enkäten och tycker därmed att Kimballs metod och modell sköter ansvarskonflikter på ett bra sätt.

RC säger att systemutvecklingsmodellen i sig inte påverkar ansvarskonflikter men anser att Kimballs metod är lättare att förstå och kommunicera. Han menar att acceptansen av införandet av ett Data Warehouse och därmed till viss del delad data hanteras lättare av Kimball just därför. Att bygga ett system där input från flera

verksamhetsprocesser behövs kräver bra kommunikation mellan avdelningar, och detta samarbete är också nödvändigt för att en implementation enligt Kimballs metod och modell skall bli lyckad.

> *”De är väl lite så att om man lyckas med ett Kimball projekt får man något som är väldigt väl förankrat i organisationen, hos användarna. Men det ställer större krav på kommunikationsfärdigheten, och det är vi väl kanske traditionellt sätt inte så duktiga på inom IT- branschen. Vi kan teknik, men är inte så duktiga på att förklara vad man skall ha den till. Och hela framgångsfaktorn, om man ska koka ner den med Kimball jämfört med Inmon är att i Inmons fall slipper vi kommunicera lite hårddraget sett. Medans i Kimballs tvingar oss att kommunicera med användarna. Och då står också projekten å faller på vår förmåga av det kan man väl säga.” (RC)*

RC menar att Kimballs systemutvecklingsprojekt är mer fokuserat på användarvänlighet, och kräver därför större förståelse av verksamhetens behov. Att få med folk utan att förklara varför och hur systemet skall användas menar RC är en svår resa, och säger också att det är en utav framgångsfaktorerna för ett Kimballsystem. Att bygga ett system där data från flera verksamhetsprocesser behövs utan att förklara vad man ska använda det till är svårt, och dessutom mer kostsam säger RC, det brukar inte heller vara gynnsamt för att få folk att bidra till projekt heller. RC menar alltså att med Kimball är det lättare att kommunicera vad som kommer att levereras och vad som inte kommer att levereras samt att kostnaden ligger generellt lägre för Kimballprojekt. Detta möjliggör också lättare acceptans och förståelse för systemet och därmed också mindre konflikter menar RC.

> *”Sen är det ju så att det finns en politisk dimension att, hur ska ja säga. Svårigheten med att driva ett användarfokuserat systemutvecklingsprojekt som Kimball gör, där den genensamma begreppsapparaten är den viktiga. Skärningspunkten är ju faktiskt att man riskerar att inte få någon gemensam begreppsapparat för det är ingen som kan komma överens om vilka begrepp som skall gälla. Så de politiska utmaningarna är ju nästan svårare i det fallet, i Kimballs värld. Medens man kan säga lite hårddraget att man skiter i frågeställningen och utvecklar istället i Inmons modell.” (RC)*

PW säger att ansvarsfördelningen mellan data är enklare i Kimballs struktur, eftersom det är uppdelat enligt processspecifika Data Marts. Det möjliggör en enklare ekonomisk indelning om kostnaden skall delas på respektive avdelningar. Inmons struktur som innebär mer sammanslagen lagring, i och med den atomära nivån gör att kostnaden blir svårare att dela på. Men PW säger också att även Inmon kan dela upp data över avdelningspecifika Data Marts. Ansvarskonflikter om vem som äger datan och vem som skall betala för drift och underhåll kan därmed bli större för en Inmon-struktur, även om den inte behöver bli det.

Diskussion

Ansvarskonflikter vid integrering av flera källsystem innebär att definitioner och attribut ibland behöver ändras för att integreringen till ett Data Warehouse skall bli korrekt. Ansvarskonflikter kan därmed uppstå mellan olika ansvariga över olika system och avdelningar, i den formen att alla vill ha sina egna begrepp igenom. För Inmons modell där man extraherar all data in i en mellanlagrad nivå (atomära nivån)

försvinner till viss del detta problemet. Transformerar man all data så den blir likformig där, slipper källsystemets struktur ändras, och därmed undviker man också konflikter. KG menar att Inmons metoder fungerar bra för att lösa ansvarskonflikter och detta är därmed inte så konstigt. JL tar upp kostnadsperspektivet för datan och påvisar att ägandeskap för data är svår att bestämma just eftersom data ofta används över avdelningar. Inmons modell är övergripande över alla avdelningar och data blir alltså inte avdelningsspecifik i den atomära nivån. Kostnaden för hela implementationen är därmed svår att dela på avdelningar, och till vilken grad dom använder Data Warehouse't. PW menar även han att just eftersom Kimball delar in datan i processspecifika Data Marts, så blir ansvarsindelningen enklare för datan. Varje process får stå för sin del av kostnaden, till skillnad från Inmon. Både JL och PW säger dock med visst stöd från RC att även Inmon kan delas upp i avdelningsspecifika Data Marts och därmed hanteras ansvaret bättre även med Inmons modell. Motstånd mot förändring kan alltid uppstå när nya system och strukturer införs och detta är också något man alltid skall ha i tankarna vid omstruktureringar. För Kimballs metod och modell som kräver mindre kostnad för utveckling kan möjligtvis mindre konflikter uppstå. Dessa typer av externa konflikter är inte beroende av funktion och modell för respektive teori och möjligtvis att det också var detta som UW hade i tankarna när han svarade att ansvarskonflikter ofta beror på andra faktorer, snarare än metod och modell. RC tar upp en annan diskussion nämligen att ansvarskonflikter hör ihop med acceptans för systemet. Har man inte användarna med sig och förstår de inte varför de ska införa ett system blir det också ansvarskonflikter menar han. Detta stöds också av teorin, att förändringsstrategin måste vara väl förankrad för att implementationen skall bli lyckad. RC förklarar att det är enklare att ta till sig ett Kimballsystem eftersom strukturen är mindre komplex samt att beskrivningsmodellen är tydligare än för Inmon. Han menar att en viktig dimension för att framgång är en tydlig kommunikation om vad man kommer att leverera och inte. Detta för att underbygga framtida konflikter vilket han anser stöds bättre med Kimball än med Inmon. Detta argument för Kimball är det ingen annan respondent som tar upp, men är väl så viktigt att beakta.

4.3 Kompetens

Kompetens är en viktig aspekt att titta på när det gäller drift och underhåll av ett Data Warehouse eftersom detta kan vara en stor konkurrensfördel för en enklare lösning. Vår fråga för den här aspekten var därför: ”*Hur stor kompetens krävs av driftpersonal som sköter underhåll?”*.

Resultat

KG svarar i enkäten att Inmons metod stödjer driftpersonalens förståelse av ETL-processen till 4, och anser därmed att Inmons metoder förstås ganska bra av driftpersonalen

UW svarar också 4a, och lägger även till att Inmons system kräver mer kompetens för drift, eftersom det byggs med fler lager/steg.

JL anser att Kimballs system får en dyrare driftkostnad, men säger samtidigt att det beror på hur tillvaron ändras. Han anser att om Kimballs metod och modell inte är genomtänkta från början och det kommer nya krav i efterhand, så finns det goda möjligheter att det blir dyrare med Kimballs modell och metod.

JJ svarade 3 på enkäten om kompetens för driftpersonal, och menar alltså att det krävs viss kompetens för drift av Kimballs metod och modell.

RC menar att de finns skillnad för drift mellan Inmon och Kimball. Han menar att eftersom Inmon lagrar data i flera steg blir de också mer kod som måste underhållas. Förändringar blir också svårare att spåra eftersom man får söka i flera steg. RC menar att Kimballs laddningsstruktur av databasen är enklare. Det är egentligen ett eller antal paket på en eller flera hierarkiska nivåer som sköter all laddning säger han, och detta bidrar till att det är lättare att hitta vad det gått fel och bidrar till en mer överskådlig kod, och mindre komplexa laddningar säger RC.

> *”Sen är det ju så, att det går att göra bra Inmon -modeller, och dåliga Kimball – modeller, så det är ju väldigt mycket upp till utvecklaren. Men följer man anvisningarna när de gäller systemutvecklingsmodellen så får man en mindre komplex lösning, tekniskt sett alltså. Och det borgar ju för att det är lättare att underhålla.” (RC)*

RC säger att laddningarna givetvis kan vara komplexa i en Kimballmodell också, men eftersom det blir färre laddsteg blir det även där mindre komplext, och därav också mer överskådligt. RC säger att krav på underhåll är synonymt med datakvalitet från källsystemen.

Han menar att krav på underhåll har mycket att göra med avvikande datakvalitet från källsystemen. Om datakvaliteten varierar över tid, kommer också ha liknande variation på underhåll. Det han menar är alltså att låg datakvalitet innebär mycket underhåll, och hög datakvalitet innebär mindre underhåll. Underhåll kan uppstå när man behöver ändra eller lägga till något menar han, och det är då lättare och kräver mindre arbete i en mindre komplex lösning eftersom beroendena är mindre då, säger RC.

> *”Ju mindre beroende, desto enklare kan man ju säga.” (RC)*

PW anser att Kimball är mycket enklare att hantera ur ett underhållsperspektiv och menar att det är mycket enklare att felsöka och spåra ändringar med Kimballs metod och modell. Han svarar även att datakvaliteten från källsystemen har en stor betydelse för underhåll.

> *”Inmon kräver så mycket mer underhåll det tar så mycket längre tid att felsöka. Jag har jobbat med den typen av modeller och jag har även suttit i förvaltning av sådana system och det var inte kul. Spåra i fem olika led och leta efter var felet var. Det är mycket felsökarspår så jag tycker ur ett kostnadsperspektiv så är det betydligt lättare med Kimball.” (PW)*

Diskussion

RC UW och PW anser att driftkostnader blir större för Inmons system eftersom ändringar måste göras i flera steg. Inmon lagrar mycket riktigt data i ett steg mer än Kimball och förändringar av data måste därmed göras i ett lager extra för Inmons modell. JL motsäger ju detta i sitt svar, men vi tror han svarat ur en annan synvinkel. JL tror vi menar att eftersom Inmon lagrar mer data, behövs inte ny data föras in vid mindre förändringar, och därmed blir det mindre driftkostnad för Inmon. JL anser att

Inmons modell lagrar mer data i sin modell, för att försäkra sig mot framtiden, och därav behövs det inte över tid lika mycket förändringar av modellen för Inmon. Detta framgår av intervjun, och måste gett grund till varför JL,s svar är motsägelsefullt.

KG och JJ svarade utifrån vardera modell, KG för Inmon och JJ för Kimball. De tycker båda att modellerna stödjer förändring, men KG tycker att Inmon är något bättre än JJ tycker att Kimball är. KG svarade 4 vilket innebär att han tycker att det nästan fungerar perfekt, medens JJ svarade 3 vilket tyder på att han inte är helt nöjd.

HB har inte den tekniska kompetensen och rollen, för att svara på frågan och avstod därmed. PN, s svar i intervjun har vi medvetet valt att inte ta med i resultatet eftersom vi ansåg att det inte svarade på frågan.

4.4 Flexibilitet – Förändring av data över tiden

Flexibilitet för bland annat organisationsförändringar eller nya typer av säljstrukturer kan vara en annan viktig konkurrensfördel mellan de olika lösningarna. Denna faktor kan lösas med hjälp av Slowly Changing Dimensions och vår fråga till respondenterna blev därför följande. ”Hur väl hanteras f*örändring av data över tiden med hjälp av SCD (Slowly Changing Dimensions)?”*

Resultat

Hanteringen av SCD (Slowly Changing Dimension) hanteras enligt Kimballs metoder bra anser RC. Han säger också att den dimensionsmodelleringsmetodik som Kimball föreslår för att hantera förändringar i dimensioner stödjer alla scenarier som han träffat på. RC ifrågasätter också behovet av SCD, och menar att kunden oftast inte är intresserade av att följa dimensionsförändringar och det implementeras väldigt sällan menar han. Ett exempel som RC tar upp där SCD kan vara användbart är vid organisationsförändringar, där försäljningsdistrikt eller liknande förändras. Statistik över sålda produkter är då intressant att se över tiden och kan därför vid sådana förändringar vara bra med SCD. Många kunder tycker inte det är relevant att införa SCD vid sådana exempel heller, eftersom resultat från en ny organisationsindelning inte går att jämföra med en gammal.

JJ och PW delar RC,s uppfattning att Kimballs metoder för hur SCD hanteras är mycket bra. PW menar också att SCD möjliggör lätt spårbarhet vid förändringar. PW menar också att både Inmon och Kimball har förutsättningarna för att hantera SCD på ett bra sätt, och att det endast är tekniken som skiljer hur det hanteras. PW säger också att de ibland använder sig av en hybrid lösning för att hantera SCD, där de inte använder Inmons normalisering, men ändå har ett arkiv för att lagra ursprungsdata i, detta för att ändringar och justeringar skall kunna återskapas ifrån ursprunget. PW menar att det brukar vara en rätt bra Trade-off som går relativt fort att implementera. Han delar också RCs uppfattning av att historisk förändring sällan efterfrågas.

> *”Det är otroligt sällan någon över huvud taget vill ha den historiska förändringen. Imånga fall är det en feature för de som gillar att följa Inmon. IPraktiken används det sällan.”*(PW)

KG anser att Kimball har mer fokus på SCD och mer väl beskriven metod, men delar även PWs åsikt om att en hybrid version ofta används.

PN säger att kunden ofta saknar kunskap om varför SCD behövs.

JL upplever att Inmons modell är bättre eftersom lagringsmodellen är normaliserad. Han anser också att det är lättare att använda SCD för Inmon eftersom det innebär mindre jobb vid förändringar.

UW säger att SCD används väldigt lite i det projekt han varit inblandad i.

Diskussion

Av intervjuerna och enkäterna vi har genomfört kan vi dra som slutsats att kunden väldigt sällan efterfrågar SCD. Om detta beror på att kunden inte förstår nyttan av det, eller att det faktiskt är så att funktionen inte anses tillräckligt funktionell vet vi inte. Ett antagande man kan göra är att analyser och rapporter inte är intressanta att ta fram utifrån inaktuell data, alltså med avseende på en gammal organisationsstruktur eller en säljstruktur som inte längre finns. Att jämföra aktuell data, med gammal data som exempelvis är organiserad annorlunda kan ses som att jämföra äpplen och päron, och detta kan vara ett skäl till varför kunden anser SCD vara en onödig funktion i vissa fall. De som arbetar med Kimball, med medhåll från KG tycker att Kimballs metoder för SCD fungerar bra, medens JL anser att Inmon fungerar bättre. HB har inte den detaljerade kunskapen om SCD som behövs för att ha en åsikt om det hanteras bra eller dåligt.

4.5 Flexibilitet - Förändrade användarbehov

Förändrade användarbehov som kräver nya typer av rapporter och analyser är för organisationer mycket viktigt att det kan hanteras. Frågan för denna faktor var därför följande. ”*Hur väl hanteras förändring av användarbehov?”*

Resultat

JJ svarade 4 och HB svarade 5 i enkäten för hur mycket väl Kimballs lösning stöder förändrade användarbehov. UW svarade 4 i samma fråga, fast med Inmons lösning som utgångspunkt. KG hade också Inmons lösning i åtanke när han skrev:

> *”Iterations ska i princip stödja detta (i flera iterationer). Ipraktiken är det alltid svårt att hantera ändrade behov och krav.”(KG)*

PN menar att det inte är speciellt komplext men att det är diskutabelt om hur lång tid det tar. Han säger att ”det är inget som ställer lösningen på kant så att säga”.

KG tycker också att förändring stöds av Iterations.

> *”Så principiellt är Inmon är dyrare men flexiblare, det är en försäkring man köper sig egentligen.” (JL)*

JL säger att om man skall vara ”renlärig” som han uttrycker det och följa Inmon fullt ut så är det större chans att man redan har datan som förändringen av användarbehoven resulterar i. JL liknar Inmons metod vid en försäkring inför framtiden, man bryr sig inte lika mycket om vad användarna ifrån början efterfrågar.

RC menar att Inmon har en fördel eftersom man lagrar större mängd grunddata, det möjliggör en lättare förändring av användarbehov. Med Kimball är det lättare att utveckla.

> *”Ja och svaret är det att de beror på. Inmon har styrkan att all data lagras. Och det är ju en sån teoretisk approach om vi lagrar allt kommer vi säkert hitta allt vi behöver efteråt. De är då till priset att det blir mer omsändigare och mer komplext att koda, men datan finns där, som regel iallafall. OM man har gjort en rejäl Inmon-insats medan Kimball är lättare att utveckla, men man inte får fatt på datan. Det finns fördelar med båda approacherna. Det är lättare att ändra i en Kimball-modell anser jag. Men det är till priset av inte får fatt på datan som annars hade lagrats i någon slags grundform om man hade kört Inmon. Är det snabba förändringar man vill ha, är Kimball att föredra. Men vill man garanterat ha all grunddata lagrad och tillgänglig i sin ursprungsform kanske Inmon är att föredra.” (RC)*

Hämtningen är dyr från källsystemen. När datan väl är hämtad är det billigt med förändring, säger RC.

PW säger att de använder sig av ett denormaliserat arkiv som är en form av ersättning av Inmons atomära nivå för att kunna hämta in eller spåra en ändring. Detta är dock vad vi vet inget som Kimball förespråkar utan snarare en form av hybrid som de själv har utvecklat, och kan därför inte vägas till fördel för Kimballs metod även att Kimballs metod inte omöjliggör att man har ett sådant arkiv. PW håller även med om att Inmons metod innebär att det finns större möjlighet att komma åt historisk data om data som inte var med i kravspecifikationen ifrån början, samtidigt som han ställer sig kritiskt till det här sättet eftersom det dels är svårt att få fram informationen för en power user, och dels så menar han att kunden oftast inte tycker att det är relevant att jämföra historiskt förändrad data.

Diskussion

Respondenterna från enkäten menar att sin respektive approach stöder förändrade användarbehov bra. Generellt sett kan man säga att alla intervjurespondenter är eniga om att Inmons metod borgar för en försäkring för framtiden vad gäller förändrade användarbehov. Om det är något som är positivt eller ej är man däremot oeniga om. Men slutsatsen är att man är enig om ambitionen med respektive approach, samtidigt som man inte är enig om resultatet av dem. Svaret på frågan om vilket resultat som är fördelaktigt är något vi inte har hittat något empiriskt stöd om, det är en svår fråga att svara på och den beror förmodligen på enormt många aspekter som inte går att förutsäga innan man väljer modell. Påståendet att Inmon är att föredra om man har ett växande scope och en stor förändring framför sig stöds även av Breslin (2007).

4.6 Flexibilitet - Förändrade affärsvillkor

Externa förändringar som lagar och regler kräver ibland förändring av datalagring. Detta är en annan viktig aspekt för systemkvalitet och bör beaktas vid införandet av ett Data Warehouse. Frågan på detta var därför: ”*Hur väl hanteras förändrade affärsvilkor (exempelvis sox-lag o.s.v)?”*.

Resultat

KG anser att hanteringen av förändrade affärsvillkor ska hanteras av ”Itterations” och att detta är ett bra sätt men att det i praktiken alltid är svårt att hantera ändrade behov och krav.

PN säger att han inte stött på några sådana problem och det enda han kunde komma på är när man lägger till nya attribut och tabeller. Detta tyckte han tog väldigt lång tid och krävde mer ansträngning än vad som var önskvärt. PN menar att man använder sig av flera olika areor i Inmons metod och får man önskemål om nya attribut som ska definieras så måste detta göras i alla areorna i relationsdatabaserna. Vidare menar PN att man måste gå in i ETL-processen och där definiera om. Dessutom måste man ändra i stored procedures (metoder i databasen) för lite mer komplex logik och det innebär att man måste in på ytterligare ett till ställe och ändra. Sedan måste menar PN att man måste se till att populera den här nya informationsmängden med historik också vilket kan va önskvärt i vissa sammanhang och att detta gör att det hela tar ännu länre tid.

JL upplever att svaret på denna fråga måste bli detsamma som i föregående fråga (förändring av användarbehov). Det är fortfarande en fråga om vilken information som ska finnas i modellen. Om man har väldigt mycket regler i modellen så tycker JL att det är lättare att lägga till detta i en normaliserad modell (som Inmon förespråkar) än i en denormaliserad modell eftersom man kan beskriva regler med mer än bara 1:M relationer mellan fakta och dimensioner.

> *”Du har ju i normaliseringen som princip att ta upp saker och ting i atomära delar, alltså det som återstår är att koppla ihop dom genom att beskriva regler för relationerna. Det är ju principen för en relationsdatabas, dom förutsättningarna har du tagit bort för dig själv rätt rejält när du använder en denormaliserad modell i form av ett Star Schema där jag ska hantera hela regelverket.” (JL)*

JL menar att man med Inmons metod som gör att man normaliserar, har relationerna som ett verktyg att jobba med där man kan tvinga 1:1, M:M eller 1:M för att beskriva verkliga relationer. Detta går enligt JL att lösa i Kimballs metod också men att enbart ett Starschema inte löser detta utan man måste skapa andra typer av lösningar. JL anser alltså att Inmons metod i grund och botten har bättre stöd för att förklara relationer eftersom metoden inte avgränsar lika hårt vilka typer av relationer som man får lova att använda. UW tycker att Inmons metoder passar bra för hanteringen av förändrade affärsvillkor. JJ har ingen uppfattning om Kimballs metod för att hantera förändrade affärsvillkor och HB tycker att det hanteras på ett mycket bra sätt.

RC påpekar att SOX är en ”udda fågel” som innebär att det finns ett tydligt krav på att man ska kunna spåra förändringar i i IT-systemen vilket innebär att man måste bygga loggningsmekanismer och att man inte får skriva över data utan endast tillföra ny data. Detta problem hanteras av Kimballs metod men även Inmons metoder hanterar detta problem och han ser ingen motsättning i uppläggen. Däremot tycker han att genom att man lagrar data i färre steg så är det lättare att implementera med Kimballs metod.

”Det blir tekniska problem på båda uppläggen beroende på hur datan ser ut. Men när det gäller allmänna förändringar eller krav på nya affärskrav på BI-lösningar så tycker jag att det är lättare att implementera med Kimballs perspektiv och det tycker jag är Kimballs styrka jämfört med Inmons modell. Kimball använder ”Good Enough-prinicpen” och då är det väldigt lätt att göra snabba förändringar för man behöver röra mindre av datan i datamodellen.” (RC)

Sammanfattningsvis tycker RC att det blir ett större systemutvecklingsprojekt av alla förändringar i en Inmon-metod.

PW pratar precis som RC om fler steg, att man i Inmons metod måste genomgå flera steg i ETL-processen. Han menar att varje steg innehåller viss logik medans det i Kimballs metod innebär att man gör en mer samlad och total laddning för hela transformationen vilket är lättare att underhålla. Eftersom affärsregler hanteras i ETL processen medför detta att man bara behöver ändra i ett enda steg.

”[..] vi tar en tabell i taget. För Inmon behöver tabellerna normaliseras i förmodligen flera måltabeller [...]” (PW)

Detta kan medföra att det blir svårt att hitta vad något görs, alltså att det är att svårt att veta vart de nya affärsreglerna ska implementeras i processen. IKimballs metod har man enligt PW bara ett ställe att leta i eller en central punkt.

Diskussion

Respondenterna som svarat utifrån både Inmons och Kimballs perspektiv anser att den modellen just dom använder är den smidigaste. Det finns argument åt båda hållen. JL (som använder Inmons metod) ser att hanteringen av förändrade affärsvillkor är bättre eftersom man normalisera och RC, PW, PN anser att Kimballs modell är enklare eftersom det är färre steg man måste ändra i när det uppkommer nya affärsvillkor. Enligt vår analys stämmer båda påståendena och det blir nästan en fråga om vilken approach man föredrar.

4.7 Flexibilitet - Hantering av tekniska villkor

Frågan för hantering av tekniska vilkor var: ”*Hur väl hanteras tekniska villkor, exempelvis inlåsningseffekter o.s.v.?”.*

Resultat

JJ svarade 5 på frågan hur väl Kimballs lösning hanterar förändrade tekniska villkor, medans HB valde att inte svara på frågan. Både KG och UW svarade 3 på samma fråga, fast med Inmon som utgångspunkt.

JL säger att inlåsningseffekterna är minimala med avseende på modellen. Han menar att inlåsningseffekterna är snarare beroende på databashanterare och val av tekniker snarare än modellen i sig. Han tar upp ett exempel med homegrown-produkter, där inlåsningseffekten är väldigt stor. JL säger vidare att om man tänker sig att lagringen (Data Warehouse) är samma som åtkomststeget (DM) i Kimballs modell och i Inmons fall är det normaliserat så tycker JL att det finns en större inlåsningseffekt i Inmons

modell, med antagandet att star schemat är det allmängiltiga formatet som de flesta front end-verktyg kan klara av att konsumera eftersom åtkomststeget hos Kimball är likadant som hos Inmon.

> *”Ett exempel: Om du använder dig av en parent-childrelation för att beskriva medlemmar av en dimensionshierarki, och så använder man sig av ett traditionellt standardrapportverktyg, så är stödet i standardrapportverktygen för att tolka en parent-childrelation då den ska i princip nysta upp hela strukturen, då den ser hur många nivåer finns där, dom är dåliga eller dom är närmast obefintliga och det gör ju då att parent-child som är ett tänkbart sätt att lagra saker och ting enligt Inmon om du har en ragged hiearki, har inget omedelbart stöd att bli tolkat att bli tolkat sen i standardrapportverktyget. Använder du däremot OLAP-motorer, så har dom stöd för att läsa ur en parent-childrelation, men inte standardrapportverktygen vanligtvis. Så då skulle man kunna säga att det blir en större inlåsningseffekt på Inmons spår att man behöver bygga starschemat för att komma vidare med vissa typer utav verktyg och det är precis det vi säger med/i dom modeller, ett starschema i alla fall men väljer lagra som Inmon.”(JL)*

JL säger att det finns en viss inlåsningseffekt i att använda Kimballs modell om källsystemen hanterar data med parent-child-relationer, eftersom man då måste ”platta ut” relationerna som han uttrycker det. Detta resulterar i att vissa relationsbeskrivningar blir lidande menar JL, och här har Inmon en fördel eftersom normaliseringen innebär att man får tillåtelse att använda parent-child-relationer. JL exemplifierar med att ett källsystem byts och man måste ta hänsyn till obalanserade träd, då menar han att man behöver bygga om mycket av lagringsstrukturerna i Kimballs fall medan Inmon är det bara att köra vidare. När det gäller modellens utformning bara köra vidare. Det är för att Kimball inte har stöd för att hantera obalanserade träd med automatik.

PN säger att han inte har tillräckligt med erfarenhet för att kommentera denna fråga, men frågar efter ett exempel. Vi gav han följande exempel:

> *”Det är väl kanske lite det som du nämnde innan att t.ex. Att har du storage arean (anm: Data Warehouset) så kanske QlikView vill ha en viss form på datan medans andra vill ha en parent child, det blir ju en viss form av inlåsningseffekt av att inte kunna erbjuda det.”(PN)*

PN håller med om att det kanske kan vara ett bra exempel och säger att de identifierade det problemet ganska tidigt i deras projekt, och därför skräddarsydde de då en access area (Data Mart) för de behoven.

RC tycker att Kimball och Inmons modeller är ganska likvärdiga vad gäller inlåsningseffekter, det beror snarare på vilken databashanterare man har valt menar han.

RC håller med om problematiken med parent-child för Kimball men menar att parent-child är en ganska speciell konstruktion där man vill göra lagringen oberoende av hur många nivåer man har i sin hierarki. Han säger även att parent-child är en ganska snygg teknisk lösning men det är väldigt få verktyg som klarar av att konsumera en

sådan struktur. RC menar att om klarar av det är det nästan alltid förknippat med begränsningar. Han säger att de som faktiskt gör det, gör det hyffsat bra men att det är en begränsande faktor i sig. RC hävdar att det är ganska komplext att lösa upp parent-child-strukturer, särskilt om hantering för Slowly Changing Dimensions är inblandat, det blir prestandaförlust när tabellerna innehåller många poster.

> *”Man har löst ett programmeringsproblem för utvecklarna, men man får istället ett prestandaproblem och ökar komplexiteten. Framförallt är det nästan omöjligt att utan verktyg läsa vad det faktiskt innehåller, man blir därmed beroende av verktyget för att tyda datan och det tycker jag är en begränsning i sig.”(RC)*

PW tycker att Kimball har ett större tekniskt oberoende, eftersom man i princip gör ett Data Mart per system och det som är gemensamt är bus-matriserna som är överskridande. Han menar att Inmon försöker klämma ihop allt till en förutbestämd datamodell vilket PW anser kräver betydligt mer mappning och merarbete. Han anser därmed att Kimball har en enklare modell, eftersom det kan variera väldigt när det gäller källsystem.

På frågan om parent-child-strukturer blir smidigare att hantera med Inmons modell svarar PW att verktygen i praktiken inte gillar att jobba med parent-child över huvudtaget och han kan inte koma på något tillfälle där de har haft kvar parent-child-strukturen till ett rapportverktyg. Han håller med om att det är möjligt att det är något lättare att hantera strukturen med Inmons modell men han kan inte se den praktiska nyttan med det. Till sist säger han att det inte finns några hinder för en parent-child-struktur i Kimballs modell heller, och att det inte finns någon direkt motsättning men den praktiska nyttan är begränsad. PW menar att det brukar snarare uppstå problem.

Diskussion

Sammanfattningsvis kan man säga att man är eniga om problemet som i detta fallet är hanteringen av parent-child-relationer, men man är oense om betydelsen av problemet där RC och PW menar att så gott som inget front-end-verktyg använder sig utav dessa typer av relationer medans JL och PN menar att det finns vissa som gör det. Eftersom HB svarade i enkäten, blir svaren på enkäten lite svåra att jämföra mellan Kimball och Inmons approach utifrån enkäten.

4.8 Integration

Integration av data från källsystemen är en mycket viktig del av ett Data Warehouse eftersom detta är en utav grundbultarna för att få den datakvalitet som behövs för att göra analyser och rapporter. Frågan på denna faktor var: ”*Hur väl hanteras integration ifrån flera källsystem?”*.

Resultat

JJ svarade 3, och HB 5 på enkäten utifrån Kimballs lösning. UW svarade 5 och KG 4, utifrån Inmons lösning.

JL tycker att Inmons approach hanterar integration bättre med motiveringen

”Eftersom du inte redan har spikat upp en så hård modell som en sån med denormaliserad lagring är.”(JL)

Han menar att eftersom Inmons modell innehåller fler tabeller, blir det lättare att foga små bitar av informationen eller tillägg av informationen utan att göra en stor ombyggnad av modellen. JL menar att detta är en utav de stora skillnaderna mellan de två modellerna, och han påpekar att det ur ett systemutvecklingsperspektiv så är detta en stor fördel för Inmons modell. Han menar att den stora ”tradeoffen” mellan de båda approacherna är lättanvänt vs flexibilitet, där Kimball är lättanvänt och Inmon flexibelt. JL uttrycker det så här:

”Det betyder att, vet vi allt vi behöver kan vi bygga utefter Kimball direkt. Förändras världen så tar vi smällen och kostnaden då istället. Inmon å andra sidan menar att man försäkra sig mot förändringar direkt. Det blir mer komplext på den tekniska sidan, men man är mer förberedd på förändring med Inmons modell.” (JL)

PN nämner ett exempel på ett projekt han har varit inblandad i nyligen och menar att de i detta fall har haft en enkel resa, eftersom de har hämtat in 100 % av källdatan ifrån ett enda ERP-system (Navision). Han berättar även om ett annat exempel där de hämtar data både ifrån ekonomisystemet och personalsystemet. I detta fallet var det en ekonomisk enhet ett begrepp som kallades för ekonomisk enhet som hade olika korrifieringar i de olika källsystemen och där fick de problem då det tog flera månader att komma fram till en bra lösning att få en samsyn på källsystemens olika korrifieringar och då slutade det med en mindre bra lösning, att kunden lyckades inte lösa det på sin sida utan det blev en s.k. cross reference-tabell i BI-systemet som PN då uppfattade som en temporär lösning för att komma igång med BI-lösningen.

”Det står på sidan ett i hans böcker, det är viktiga grejer integration av data.” (RC)

RC anser att integrationsmöjligheterna är likvärdiga mellan Inmon och Kimball. Han säger att båda adresserar att man ska ha många källsystem. RC menar också att när det gäller integration så har Kimball tydligare fokus på att faktiskt integrera data med gemensamma dimensioner. Rent tekniskt är det ingen skillnad enligt RC, men han menar att det är större fokus på att data ska vara jämförbart via gemensamma dimensionsstrukturer. RC ställer sig kritisk till Inmons val att använda ett Data Warehouse som ett mellanlager mellan källsystemen och Data Martsen, vad han menar är att han ser inte nyttan i att mellanlagra datan i normalfördelad form. Han menar att man inte löser grundproblemet, med att data faktiskt inte är jämförbart. RC säger vidare att tekniskt sett går det att skjuta ut vad som helst från Data Warehouse till Data Marts, men har man inte gjort hemläxan med att ha gemensamma register och annat så spelar det ingen roll tekniskt om man säger så det är det större fokus i Kimballs approach att ha gemensamma register på ett bra sätt. RC menar att Kimballs approach har bättre stöd för integration om man *”implementerar det enligt konstens alla regler”*. Han tycker att Inmon har alltför tekniskt angreppssätt på problemet, man löser problemet rent tekniskt men det finns även politiska aspekter på problemet. RC tar upp ett exempel på ett politiskt problem och nämner frågeställningen *”Vad är en kund?”*.

”Det är inte ett tekniskt problem. Då är svaret tycker jag iallafall inte att då att man tekniskt sett kan man flytta kundposten A till B utan det handlar om den organisatoriska förståelsen, organisatoriska acceptansen av vissa begrepp och då kan man säga att Kimball driver den organisatoriska frågan hårdare.” (RC)

På frågan om man kan sammanfatta frågan med att genom valet av Inmons approach köper man sig en försäkring vad gäller flexibilitet inför framtiden svarar RC att det resonemanget kan man tillämpa på all systemutveckling överlag men det återstår att bevisa. Han säger att han inte har hittat något stöd för den ena eller andra inriktningen (anm: Inmon eller Kimball) annat än man vet att man tar en stor kostnad i början av ett projekt, men det är osäkert vad man kommer att få för utfall eller vilken besparing det kommer att innebära eftersom besparingspotentialen består av *”tusen olika variabler”*. RC säger att också att företag som lägger ner många miljoner på IT-underhåll har större villighet att satsa på större lösningar därför att man har råd med det helt enkelt. Han säger sig vara helt övertygad om att Inmons approach driver mer kostnader, men påpekar samtidigt att han inte kan säga om den totala kostnaden efter hela systemets livscykel är större eller mindre men gissar samtidigt på att Kimballs approach är billigare totalt sett.

”Man köper inga försäkringar utan man lagar där det har brunnit.” (RC)

RC hävdar att utgångspunkten med en Inmonmodell är att det är för kostsamt att hämta data ifrån källsystemen, för komplext att hämta data, och att den hämtningen i sig driver mycket kostnad. Han tror att anhängare av Inmons vision är att det blir mycket lättare att hantera datan när den väl är inne i Data Warehouset, och att hämta data ifrån realsystem och produktionssystem innebär väldigt höga kostnader och mycket *”krångel”* hävdar han. Har man ett sådant skulle det kanske kunna vara lättare med Inmons approach hävdar RC. Men han säger samtidigt att det är lika vanligt att man säger att -Vi har bytt lönesystem, vi måste ha en ny integration av vårt nya Lönesystem med vårt BI-system, eller nu har vi infört en ny process eller behöver vi komplettera vårt datalager med data från ytterligare en verksamhetsprocess. Då är det nog snarare mer flexibelt med en Kimballösning, eftersom den har ju bara fokus på *”Time to Market”* menar RC. Han hävdar vidare att det inte finns ett svar på om antagandet att den ena eller den andra modellen skulle garantera flexibilitet stämmer eller inte.

”Ett scenario där det är jättekrångligt att hämta datan och själva hämtningen av datan av olika källsystem är relativt statistik, då är det nog enklare med Inmons approach, eftersom då påverkar man bara slutanvändarresultatet, eller Data Marts och liknande. Själva lagringen påverkas inte.” (RC)

RC säger att när det handlar om att byta ut eller tillföra nya verksamhetsprocesser så tycker han att Kimballs modell är effektivare. Han säger vidare att man löser framförallt ett annat organisatoriskt problem, d.v.s. integrationen av data. Vidare säger han att om det bara handlar om att få ytterligare rapporter så kanske det hade varit lättare med Inmons approach. Avslutningsvis säger RC att om man vill runda ett politiskt problem så kan man använda Inmons approach, men om det är viktigt för organisationen att lösa integrationsproblem, integration mellan data i form av

gemensamma dimensioner så adresserar Kimball problemet och då får man ju en effekt man vill ha av IT-system.

PW anser att ur ett modelleringsperspektiv så är det betydligt enklare för Kimball-anhängare att integrera flera källsystem. Det motiverar han med att de inte behöver fokusera på att få in dom i samma modell. Samtidigt så säger han att han personligen har ”köpt” den modellen, och att det är en personlig reflektion. PW håller inte med om liknelsen om att Inmons approach kan ses som en försäkring inför förändringar i framtiden, eftersom det innebär för mycket underhåll att driva en Inmon-modell.

Diskussion

Även på denna fråga ser man den röda tråden genom respondenternas åsikter som är att Inmons approach innebär mer flexibilitet i tradeoff mot något annat, innan har man sagt att den tradeoffen är kostnad men nu säger JL att tradeoffen är lättanvändbarhet där Inmons approach är flexiblare och Kimballs modell är mer lättanvänd. Intressant är att RC tar upp Inmons approach som fördel när man behandlar relativ statistik, då det är något vi tog upp i teorin under rubriken ”Living Sample Data” för Inmon, men vi hittade ingen motsvarighet i Kimballs böcker. Generellt sätt är enkätrespondenterna rätt eniga, då det endast skiljer 1 ”poäng” mellan deras svar för hur respektive approach stöder integration.

4.9 Korrekt data

Att datan inte är tolkningsbar och stämmer överens med verkligheten kontrolleras med hjälp av denna faktor. Frågan till respondeterna var: *”Hur korrekt är/blir datan i ett Data Warehouse, och hur väl stämmer datan överens med verkligheten?”*.

Resultat

KG satte en 2a på den frågan. Han tycker att detta är en viktiga fråga där han anser att varken Kimballs eller Inmons lösning är framträdande. Han menar att denna fråga (om korrekthet) lätt skulle kunna testas genom att räkna exempelvis antalet rader eller att använda timestamp vid indexering.

UW satte en 3a på den frågan och skriver att det beror på vilken verklighet som avses. Datan i källsystemet kan vara verkligheten men inte tvunget korrekt.

PN pratar om att det ofta finns två sätt att föra över data från källsystemen. Antingen så hämtar man datan direkt eller så hämtar man den som flatfiler. Idet första fallet säger PN att man får datan i sin ursprungsform vilket därmed speglar källsystemet samt att man får rätt datatyper eftersom det ligger validerat i exempelvis affärssystemet. Idet andra alternativet berättar PN att man etablerar kontrakt med en systemägare på källdatasidan. Detta innebär att den systemansvarige tillhandahåller filer som sedan används i Data Warehouset och skulle det vara något fel i dessa då hänvisar man till den systemansvarige. Den systemansvariges applikationer som extraherar och skapar dessa filer skulle möjligtvis kunna innehålla buggar vilket skulle kunna göra att datan blir midre korrekt. Därmed menar PN att det är troligast att datan blir mest korrekt om man hämtar den direkt från källan men då till priset av att man är ansvarig för att innehållet är korrekt. Men vanligtvis är datan i Data Warehouset en representation av datan i källsystemet.

JL säger att man med Inmons metod får ett Data Warehouse som är en korrekt representation av källsystemet men han ser att det är fördel med Inmons metod eftersom man väljer att lagra datan i normaliserad form eftersom datan ofta uppstår i normaliserad form i källsystemen. Detta menar JL medför att man inte behöver översätta källsystemets modell lika hårt till en lagringsmodell. JL berättar vidare att Inmon kör med Iterations och att detta ökar chanserna till korrekt data.

HB sätter en 5a på frågan och JJ sätter en 5a på frågan.

RC svarar att det brukar finnas en förhoppning om att man ska kunna rätta dålig data i ett datalager.

> *”Man tror att genom att initiera ett datalager så får man automatiskt bättre datakvalitet. Jag anser att det fungerar inte på det sättet, får man skräp in kommer det ut skräp.” (RC)*

Vidare menar RC att det är rätt sällan man använder datalager för att lista ut stavningar och liknande och att man försöker rätta problemet där det uppstår istället, man rättar i rätt ände av kedjan. Vidare menar RC att man måste känna till affärslogiken till 100% för att kunna få perfekt datakvalitet. Han menar att det går att göra men att det förmodligen blir mycket dyrare och mer komplext att sitta och göra felkontroller istället för att gå direkt till datakällan

PW berättar att ambitionen är att spegla källsystemen till 100%. En viktig sak PW säger det är att datakvalitet helt och hållet beror på källsystemet och att det man kan göra på BI/Data Warehouse-sidan är att underlätta för ägarna av källsystemet att identifiera felen. När det gäller skillnaderna mellan Inmons och Kimballs metoder ser PW att det skulle kunna vara svårare att fixa dålig datakvalitet med Inmons metoder. Han menar att det kan vara svårare att få in en normaliserad struktur om datan från början inte är korrekt. Om man ser det på det sättet så tycker PW att man med Kimballs metoder rent praktiskt kan hantera dålig data på ett mer korrekt sätt.

Diskussion

PN's svar är intressant där han nämner att kunden ibland väljer att exportera informationen från källsystemen som flatfiler och att man där skulle kunna tappa korrekthet. Vi kan dock inte se att detta skulle ha någon koppling till Inmons eller Kimballs metod när det gäller den biten.

RC säger att man vanligtvis brukar ha en förhoppning om att man får bättre datakvalitet i Data Warehouset men att det inte alltid är så i verkligheten. Om det är så att Data Warehouset speglar källorna till 100% som respondenterna säger och att man inte rättar så mycket av datan i Data Warehouset då kommer problemet med datakvalitet aldrig att ligga i Data Warehouset i sig utan alltid på källsystemsidan. Detta ser vi som en viktig punkt om man ser till tidigare forskning som lägger så stor tonvikt vid att korrektheten ska vara hög i Data Warehouseet när detta i praktiken kommer att bero på källan.

KG som säger att detta egentligen är ett enkelt problem att hantera (genom att räkna antalet rader) men som inte adresseras särskilt bra av varken Inmons eller Kimballs approach håller vi inte med om eftersom Kimball har med just det här exemplet med att räkna antalet rader tillsammans med en massa andra metoder för att hitta fel (anm: se teoridelen i uppsatsen).

4.10 Konsistent data

I ett Data Warehouse är konsistentheten av data mycket viktig. Data måste deklareras på samma sätt för att kunna jämföras och frågan på denna faktor var: *”Hur konsistent är/blir datan? (Exempel är att data anges på samma sätt med hjälp av Metadata.)”*.

Resultat

KG satte en 4a på frågan och svarade att grundtanken med ett är att bygga upp konsistent data men att det i praktiken blir avvikelser då olika delar byggs upp vid olika tidpunkter.

UW satte en 2a på frågan och tycker att Inmons metod inte hanterar problemet så bra samt att det krävs disciplin för att få datan konsistent.

PN berättar att detta kan skilja sig åt från fall till fall men att konsistens är något svårt som enligt honom ofta beror på hur mycket ordning det finns hos kunden. Han nämner att det i ett projekt han jobbade i fanns ett ekonomisystem och ett personaladministrativsystem som skulle knytas samma i ett Data Warehouse där centrala begrepp inte hade samma kodbas eller domän (inkonsistent data). Detta var ett problem som inte kunde lösas på kundens sida utan PN berättar att man fick använda sig av en “cross reference-tabell”. Detta i sin tur skapade problem när man ibland inte fick signaler om att lägga in nya koder i cross reference-tabellen eftersom det då inte gick att matcha indata. Enligt PN hade man då byggt in ett underhållsproblem i en sådan lösning.

JL valde att inte svara på denna fråga eftersom det är alldeles för djupt ner i ETL-processen för hans kompetensområde.

HB satte en 5a på frågan och JJ satte en 4a på frågan.

RC säger att han egentligen inte har någon bra koll på detta utifrån Inmons och Kimballs metoder men utgångspunkten för Kimballs syn är att man inte ska behöva veta något om det underliggande systemet utan detta ska beskrivas i metadatan. Om man tittar i Inmons bok så menar RC att där står att man trycker väldigt mycket på metadataprojekt men att man sällan orkar fullfölja eftersom man inte vet hur man ska hantera problemet organisatoriskt. Kimballs metod enligt RC är att man ska hantera det separat i någon dokumentationsform eller webbsida för att det ska vara lätt att förstå.

RC ser att det finns viss tonviktsskillnad mellan Inmons och Kimballs approach. Han anser att man i Inmons approach inte är så intresserad av att jämföra data utan att man ser presentationen av data som isolerade öar. RC får uppfattningen att det i Inmons approach inte är så intressant att jämföra äpplen och päron (mellan olika källsystem), att detta inte är IT-avdelningens jobb. IKimballs approach är det tvärtom menar RC

där tycker man att integrationen är viktig. RC upplever att Kimballs approach trycker på Integration till den grad att man inte bör bygga ett datalager om man inte kan integrera datan.

> *”Det här med share- dimensions eller dimensionsmodellering, det är lite hela utgångspunkten i datalagret är just den gemensamma begreppsapparaten oavsett vad jag har för källdata. Den är ju helt central i Kimballs perspektiv, men jag upplever inte alls att den är lika tydlig i Inmons perspektiv.” (RC)*

Enligt PW är detta en verksamhetsproblem som egentligen inte har så mycket med Kimballs eller Inmons modell att göra som han ser det.

> *“Har man tex. olika benämningar fast man menar samma sak i två olika datakällor så kan man bara till begränsad grad lista ut om attributen matchar varandra.“ (PW)*

Här menar PW att det är viktigare med dokumentation från kundens sida. När man väl lyckats sammanföra datan tycker PW sen att man kan hantera denna integration på ett mycket bra sätt med Kimballs metoder. Ioch med att man har Data Mart-överskridande dimensioner så blir datan extremt konsistent.

Diskussion

Vi håller med RC om att integrera flera källsystem är utgångspunkten i Kimballs approach. Det märks i litteraturen och det står bra beskrivet hur man ska adressera problemen med inkonsistent data. Utöver det klara stödet i litteraturen för hanteringen av datakonsistens så fungerar detta enligt PW även utmärkt i praktiken. Stöd för detta fås också av HB samt JJ som rankade Kimballs metod högt. När det gäller Inmons approach håller vi med RC om att man i litteraturen inte trycker så mycket på integration av data. Stöd för detta fås också av UW som inte tycker att problemet hanteras bra.

PNs svar är intressant eftersom det har med hanteringen av inkonsistent data att göra men vi vill inte dra någon slutsats utav detta då vi inte kan konstatera om metoden han använde sig av för att hantera inkonsistent data var Kimball eller Inmon-specifik.

4.11 Fullständig data

Att datan är fullständig är viktigt för att kunna jämföra och generera analyser och rapporter. Detta är en viktig faktor för hur bra datakvalitet som finns och frågan var: *”Hur fullständig är/blir datan? (Finns all data med?)”*.

Resultat

KG svarade en 2:a och han tycker att detta är en viktiga fråga där han anser att varken Kimball eller Inmon är framträdande. Han menar att detta lätt skulle kunna testas genom att räkna exempelvis antalet rader eller att använda timestamp vid indexering.

UW satte en 4:a på frågan och svarade att han ansåg att datan blir komplett.

JL tycker att detta är en fråga om hur komplett datan är efter laddningen och ETL-processen samt hur många % fel man väljer att acceptera. Han menar vidare att detta i sig blir en fråga om hur komplett datan måste vara för att användaren ska kunna fatta beslut. Enligt JL finns det ingen motsättning mellan Inmons metoder och Kimballs metoder utan det handlar mer om hur man etablerar ETL-processen samt hur man signalerar ut mot slutanvändaren. JL påpekar dock att det Kimballs approach beskrivs tydligare hur ETL processen ska utformas och att det i Inmons approach bara står vad man ska hantera och varför. Metodmaterialet för Kimballs approach är mer konkret medans Metodmaterialet för Inmons approach ligger mer på en akademisk nivå. JL påpekar dock att det inte finns någon motsättning mellan metoderna i sig eftersom det fortfarande är samma ETL-process.

> *”Det är inte så att den ena säger att lite skit får man leva med. Där är dom helt överens (Inmon och Kimball). Det finns en absolut nivå av datakvalitet, det man kan säga är väl att Kimball är lite mer pragmatisk i sina resonemang. Han menar att datakvalitet definieras utifrån dess användbarhet, och jag gissar att Inmon är lite mer akademisk i sin definition.” (JL)*

JL ser ingen skillnad i Kimballs och Inmons metoder när det gäller varför och vad man bör göra i ETL-processen med däremot kan det bli mer eller midre komplext beroende på om man har en normaliserad lagring eller inte.

HB satte en 5a på frågan och JJ satte en 5a på frågan.

RC När det gäller ”fullständighet” pratar JC om att Kimball i sin approach menar att man måste förstå affärslogiken i ursprungssystemet (källsystemet) till 100% för att kunna uppnå perfekt datakvalitet. Att efterlikna alla regler som finns i ett affärssystem är ett affärssystemprojekt i sig menar RC.

> *”Det går, men det jobbet blir förmodligen mycket dyrare än att göra det och mycket mer komplext att sitta och kontrollera vad någon har gjort istället för att faktiskt rätta det i källan.” (RC)*

RC menar vidare att det förmodligen är mycket enklare att rätta saker i källan eftersom det är mycket svårare att göra det senare i datalagret. Han berättar att dom tidigare har försökt att göra detta (rätta felen i datalagret) framförallt när man haft nyinstallationer av affärssystem där affärssystemet inte riktigt kommit igång ordentligt. Då har man varit tvungen att kolla logiken och detta har tagit enormt mycket tid. 25-30% av projektbudgeten har ibland gått åt till att koda logik för att kontrollera så att t.ex. alla poster är ifyllda eller om X och Y finns att även Z finns.
Problemet med att rätta datan i datalagret eller källsystemet menar RC inte är något som skiljer. Inmons eller Kimballs metoder åt eftersom det är samma logik egentligen.

PW berättar att man kan fixa vissa systematiska fel med logik men om det på källsystemsidan saknas en kund på en rad i databasen så kan man inte bara hitta på en kund. Viss information går att få fram via analyser men saknas info så struntar man inte i att ta med den för då skulle totalbeloppet bli fel och det menar PW att man vill undvika. Alternativet hade varit att lagra raden i en slags tabell för alla felaktiga rader men ambitionerna är att inte neka några rader som är ofullständiga utan ta med allt.

PW tycker inte att ETL-processen som kan påverka datakvaliteten skiljer sig åt i Inmons metod eller Kimballs metod utan det är jobbet man lägger på att analysera, specificera och hur mycket tyngd man lägger på felkontroller i ETL-processen. Detta i sin tur varierar beroende på hur mycket tid man har i projektet. Enligt PW skulle man vinna mer på att låta felen rättas i källorna än att man på Data Warehouse sidan gör koncentrerade automatiska kontroller som hanterar problem med datakvaliteten.

Diskussion

ETL-processen verkar alltså inte skilja sig något åt vare sig man använder Inmons eller Kimballs metoder.

Tidigare forskning hade visat att hög datakvalitet/systemkvalitet var synonymt med ROI men det intressanta är att respondenterna har sagt att det är kostsamt att rätta datakvaliteten i Data Warehouset. En intressant slutsats är att ju mer pengar man lägger ner på att rätta datan i Data Warehouset ju mer sannolikt är det att ROI bli högre vilket verkar vara motsägelsefullt ur kostnadssynpunkt. Frågan är då hur man ska balansera dessa två motsättningar. Intressant är också PWs åsikt om att man vinner mer på att låta felen rättas i källorna. En annan viktig aspekt i denna diskussionen är JL svar där han säger att frågan i sig borde vara hur komplett datan måste vara eller hur många fel man väljer att acceptera för att användaren ska kunna använda systemet.

RC påstående att man måste ha god insikt i kundens verksamhet för att uppnå högre datakvalitet stämmer bra överens med de teorier som beskrivs i Kimballs metod där han (Kimball) nämner detta.

JL säger att det finns ett bättre material kring hur ETL-processen ska gå till i Kimballs approach samt att Inmon ligger på en mer akademisk nivå och det är precis på det sättet vi också uppfattat materialet kring ETL-processen när vi gjort våra litteraturstudier.

5 Slutsats/slutdiskussion

Vi identifierade två kritiska faktorer utifrån Wixom och Watsons artikel om datakvalitet och systemkvalitet samt de övriga faktorerna om organisation (acceptans, kompetens samt ansvar), som vi kom fram till gemensamt med WM- Data i form av en fokusgrupp som resulterade i ett ramverk. Faktorerna som vi undersökte skulle enligt Wixom och Watsons artikel vara kritiska och detta har vi fått fram delade meningar om. När det gäller flexibilitet och Slowly Changing Dimensions (SCD) så ser kunderna ingen nytta i att kunna hantera denna typen av flexibilitet. När det gäller datakvalitet så har det framkommit att det inte är vanligt att man rättar särskilt mycket av datan i Data Warehousesidan utan mest på källsystemsidan. Detta var inte vår avsikt att undersöka då uppsatsen handlade om hanteringen av dessa faktorer men det är ändå en aspekt som är värd att nämna. Vidare när det kommer till hanteringen av dessa kritiska faktorer så trycker Inmon och Kimball på olika saker i sina böcker och därav kan vi inte entydigt svara enligt samma struktur för Inmon och Kimball i hur dessa hanterar dom kritiska faktorerna. Resultatet av undersökningarna svarade på hur de kritiska faktorerna fungerade i praktiken och utifrån detta har vi lyckats få fram skillnader och likheter mellan Inmons och Kimballs metoder och hur de hanterar dessa kritiska faktorer.

När det gäller acceptans för systemet har vi kommit fram till att Kimballs metod har en klar fördel i det avseendet att man använder dimensionsmodellering med star - schema vilket ger en struktur med mer överskådlig data. Det resulterar i att Kimballs modell blir lättare att ta till sig för icke IT-kunniga till skillnad från Inmons modell som är mer komplex och därmed svårare att ta till sig. Man kan även uttyda utifrån intervjuerna att ansvarskonflikterna skulle kunna bli lättare att hantera med Kimballs processbaserade lösning. Vad man bör ha i tanken här är att det även för Inmons Data Marts går att dela in utifrån processer även att han inte förespråkar det i första hand.

Vad gäller systemkvalitet (flexibilitet och integration) har vi kommit fram till att Inmons modell representerar hög flexibilitet till kostnad av lättanvändbarhet och resurser i form av utvecklingskostnad och drift. Respondenterna menar att kunden väldigt sällan efterfrågar stöd för Slowly Changing Dimensions. Man menar att jämföra aktuell data med gammal data som exempelvis är organiserad annorlunda kan ses som att jämföra äpplen och päron. Inmon -anhängarna menar att Inmons lösning hanterar affärsvillkor bättre eftersom man normaliserar datan. Kimballs modell kan däremot ses som enklare eftersom det är färre steg man måste ändra i när det uppkommer nya affärsvillkor. Enligt vår analys stämmer båda påståendena och det blir nästan en fråga om vilken lösning man föredrar. Respondenterna är eniga om att Inmons modell klarar av att hantera ”parent-child” – relationer bättre men vissa respondenter är däremot oeniga om betydelsen av detta och menar att antalet rapportverktyg som konsumerar denna typen av datastruktur är minimal.

En slutsats kring datakvalitet som vi har kommit fram till är att man inte får bättre datakvalitet i ett Data Warehouse än den datakvaliteten som finns i källsystemen. Det vi också har märkt är att ETL -processen inte skiljer sig speciellt mycket mellan de båda lösningarna och att modellen inte resulterar i skilda angreppssätt i hur ETL - processen skall utföras. Vi har även kommit fram till att respondenterna anser att det

är kostsamt att hålla en hög datakvaliteten samtidigt som tidigare forskning menar att hög datakvalitet och systemkvalitet är synonymt med hög ROI.

Undersökningsaspekter	**Resultat**	**K**	**I**
1. Acceptans	Kimball har en klar fördel eftersom datan blir mer överskådlig vid dimensionsmodellerade starschema.	+	
2. Ansvar	Konflikter för ägandeskap av data samt attribut och definitioner försvinner delvis med Inmons lösning eftersom man lagrar in allt till en övergripande atomär nivå. Lättare att ta till sig ett Kimballsystem eftersom lösningen är mindre komplex och därav accepteras lättare.	 +	+
3. Kompetens	Driftkostnader blir större för Inmons lösning eftersom ändringar måste göras i flera steg.	+	
4. Flexibilitet - Förändring av data över tiden	Kunden efterfrågar sällan SCD, men båda lösningarna stödjer detta.		
5. Flexibilitet - Förändrade användarbehov	Inmon har en fördel eftersom man lagrar mer data.		+
6. Flexibilitet - Förändrade affärsvillkor	Fördel med Inmon eftersom man normaliserar datan. Fördel med Kimball eftersom det är färre steg att ändra i.	 +	+
7. Flexibilitet - Hantering av tekniska villkor	Inmon stödjer ”parent-child” relationer bättre, men respondenterna är oense om hur viktigt detta är.		+
8. Integration	Inmon mer flexibel och Kimball mer lättanvänd. Båda anses hantera integration bra.	+	+
9. Korrekt data	Datakvaliteten avgörs i källsystemen. Kommer det in dålig kvalitet, blir kvaliteten på datan i ett DW lika dålig.		
10. Konsistent data	Kimballs metod för hantering av inkonsistent data har mer tonvikt än Inmons metod. Kimball är bättre på att integrera data från olika typer av källsystem.	+	
11. Fullständig data	Kostsamt att korrigera felen i ett DW, bättre att angripa problemen i källsystemen.		

Tabell nr8: Tabell som visar resultatet av undersökningen

6 Källförteckning

Andersen, Erling S. (1994*) Systemutveckling – principer, metoder och tekniker*

Andersen, Ib. (1998) *Den uppenbara verkligheten - Val av samhällsvetenskaplig metod*. Studentlitteratur, Lund

Becker, Bob. (Jun, 2006) *Kimball University: Data Stewardship 101: First Step to Quality and Consistency*
http://www.intelligententerprise.com/showArticle.jhtml;jsessionid=0BRDR0PWMXSUGQSNDLRSKH0CJUNN2JVN?articleID=188101650&pgno=1 [2008-05-26]

Becker, Bob. (Sep, 2007) *Kimball University: The Subsystems of ETL Revisited*
http://www.intelligententerprise.com/channels/bi/showArticle.jhtml?articleID=202405400 [2008-05-26]

Breslin, Mary. (May, 2007) *Data Warehousing Battle of the Giants: Comparing the Basics of the Kimball and Inmon Models*
http://www.bi-bestpractices.com/view-articles/4768 [2008-05-26]

Chuck Ballard, Daniel M. Farrell, Amit Gupta, Carlos Mazuela, Stanislav Vohnik. (Mar, 2006) *Dimensional Modeling: In a Business Intelligence Environment*
http://www.redbooks.ibm.com/redbooks/pdfs/sg247138.pdf [2008-05-26]

Davis, F. D. (1989) *Perceived usefulness, perceived ease of use, and user acceptance of information technology*. MIS Quarterly, 13(3), 319-340.

English, Larry P. (2005) *Business Intelligence Defined*
http://www.b-eye-network.com/view/1119 [2008-05-26]

Gartner Group report. (Sep, 1996)
http://www.b-eye-network.com/view/1119 [2008-05-26]

Gilad, B., Gilad, T. (1986) *Business intelligence – the quiet revolution. Sloan Management Review*, Vol. 27 No.4, pp.53-60.

Gray, P., and Watson, H. J. (1998) *Decision Support in the Data Warehouse, Prentice Hall, Upper Saddle River.*

Halvorsen, K. (1992) *Samhällsvetenskaplig metod*

Hayen, Roger L. Rutashobya, Cedric D. Vetter, Daniel E. (2007) *An investigation of the factors affecting data warehousing success*

Holme, Magne & Solvang Krohn. (1997) *Forskningsmetodik - Om kvalitativa och kvantitativa metoder*. Studentlitteratur, Lund

Imhoff, Claudia. (Oct, 2007) *Operational Business Intelligence – A Prescription for Operational Success*

http://www.b-eye-network.com/view/6281 [2008-05-26]

Inmon, W.H. (1996) *Building the Data Warehouse – Second Edition, Inmon Information Systems Architecture, Wiley*

Inmon, W.H. (2003) *Slowly Changing Dimensions in the CIF*
http://www.dmreview.com/issues/20030601/6806-1.html [2008-05-26]

Inmon, W.H. Terdeman, R.H. Imhoff, Claudia. (2005) *Exploration Warehousing: Turning Business Information into Business Opportunity*

Jaobsen, Dag Ingvar & Thorsvik, Jan. (2002) *Hur moderna organisationer fungerar*

Jukic, Nenad. (2006*) Modeling strategies and alternatives for data warehousing projects.* Communications of the ACM. April 2006/Vol. 49, No. 4

Juran, Joseph M. and A. Blanton, Godfrey. (1999) *Juran's Quality Handbook, Fifth Edition*, p. 2.2, McGraw-Hill

Kimball, Ralph. (Dec, 2004) *The 38 Subsystems of ETL*

Kimball, Ralph. Caserta, Joe. (2004) *The Data Warehouse ETL Toolkit.*

Kimball, Ralph & Ross, Margy. (2002) *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition), John Wiley & Sons, ISBN 0-471-20024-7*

Kimball, Ralph. (Oct, 2007) *An Architecture for Data Quality*. DM Review Magazine, October 2007
http://www.dmreview.com/issues/20071001/1093610-1.html [2008-05-26]

Klein, M. Methlie, L.B. (1990) *Expert Systems: A Decision Support Approach*, Addison-Wesley, Reading, MA,

Lawyer, Jeff and Chowdhury, Shamsul. (2004) *Best Practices in Data Warehousing to Support Business Initiatives and Needs*

Sakaguchi, T. and Frolick, M. N. (1997) *A Review of the Data Warehousing Literature.* Journal of Data Warehousing (2:1), pp. 34-54.

Salvatore T. March a, Alan R. (2005) *Integrated decision support systems: A data warehousing perspective. ScienceDirect*

Soejarto, Alex. (Mar, 2003) *Tough Times Call for Business Intelligence Services. issue of VARBusiness.*
http://www.crn.com/government/18823134 [2008-05-26]

Thomsen, Erik. Spofford, George. Chase, Dick. (1999) *Microsoft OLAP Solutions.*

Vandenbosch, Betty. Huff , Sid L. (Mar, 1997) *Searching and Scanning: How Executives Obtain Information from Executive Information Systems,* MIS Quarterly, Vol. 21, No. 1., pp. 81-107.

Wang, Richard Y. Storey, Veda C. Christopher, P. Firth. (1995) *A Framework for Analysis of Data Quality Research*

Wixom, Barbara H. Watson, Hugh J. (Mar, 2001) *An Empirical Investigation of the Factors Affecting Data Warehousing Success.* MIS Quarterly, Vol. 25, No. 1, pp. 17-41. (2001)

Whitson, Wanda. (Feb, 2005) *Gartner Says More Than 50 Percent of Data Warehouse Projects Will Have Limited Acceptance or Will Be Failures Through 2007* http://www.gartner.com/press_releases/asset_121817_11.html [2008-05-26]

Whittington, Richard. (2002) *Vad är strategi – och spelar det någon roll?*