Documentation and evaluation of data environments for data warehouse

by

Jan Thomas Furset Urke

Master Thesis in

Information and Communication Technology

Agder University College

Faculty of Engineering and Science

Grimstad
Norway

May 2007

Abstract

In order to support Integrated Operations ConocoPhillips Norge (COPNO) has in cooperation with the "Good to Great" and "Data to Decisions" projects undertaken the task of creating a data warehouse system they have called "Knowledge Infrastructure". In conjunction with this project COPNO wishes to examine the system and give an overall description of the Knowledge Infrastructure. COPNO also wants to investigate into the possibilities of separating the development- and test environments as well as the chance of creating other environments.

In this thesis the system at COPNO is described and evaluated against current problems, especially the issue of separating user data marts from the batch environment and the limitation of only having two environments, namely one "DevTest" environment and a production environment. Further, possible solutions to the environment issue are discussed with focus on best practice from SAP.

A more complete solution implements a total separation of the production environment from any other environments. The production environment would then run on hardware separate from the rest of the system. Also, the thesis has arrived at the conclusion that there should exist separate development and test environments. Although bringing more complexity to the overall system, it is also the opinion of this thesis that creation of a sandbox environment could involve advantages and be favorable for the system.

Based on the needs and use of the system, several possible solutions to the addressed problem could be presented. An evaluation of costs versus funds and gains to the system needs to be performed in order to make a final decision as to which solution is favorable.

Further work consists in implementation of any adopted solutions based on a managerial decision as to whether they have any real value as against the current configuration of the system.

Preface

This thesis is the last in the study program for Master of Science in Information and Communication Technology (ICT) at Agder University College (AUC), Grimstad, Norway.

The thesis is written for ConocoPhillips Norge (COPNO), and part of the project was carried out in COPNO's offices in Tananger, Stavanger. I would like to thank COPNO for giving me the opportunity to write about a topic I find very interesting and future-oriented. I also deeply appreciate the premises, equipment and documentation placed at my disposal enabling me to base the research on relevant material, resulting in a relevant and adequate solution.

Additionally I would like to thank University College lecturer Jan Pettersen Nytun who was the formal supervisor for the project. His technical knowledge and help in managing the project in the right direction has been very important. My contact person for data warehouses, Christian S. Jensen at the University of Aalborg, Denmark, merits my thanks for his expertise and help in reviewing the thesis.

At ConocoPhillips Norge I would like to thank Snr. Specialist Integrated Operations Pål Navestad, who was the contact person for me in the company, for his ideas and comments about the work. At SAS Institute AS I would like to thank my contact person in the company, Professional Services Director Kåre Nygård, who helped me in finding relevant information and placing people of relevant knowledge at my disposal.

Finally I wish to direct my gratitude to all the people on the Data to Decisions team, in SAS Institute and in ConocoPhillips Norge incidentally who have put themselves at my disposal in every way to help me complete this thesis.

I am very satisfied with the final result of the thesis, and I am of the opinion that it has given me expertise that will be highly relevant now that I have completed my 5-year education at AUC, Grimstad.

Jan Thomas Furset Urke

Grimstad, 2007

Table of Figures

Table of Tables

Table of Contents

Acronyms and Abbreviations

AIX	IBM's UNIX Operating System
AUC	Agder University College
BCNF	Boyce-Codd Normal Form
BI	Business Intelligence
BSD	Berkeley Software Distribution
BW	Business Warehouse
COPNO	ConocoPhillips Norge
COP	ConocoPhillips
CP	ConocoPhillips
CP Norge	ConocoPhillips Norge
CPU	Central Processing Unit
D2D	Data to Decisions
DB	Database
DBA	Database Administrator
DBMS	Database Management System
DEn	Development Environment in the SAP system
DDL	Detail Data Layer
DDS	Detail Data Store
DEV	Development environment
DevTest	Combined Development and Test Environment in KI
DI	Data Integration
DIS	Data Integration Studio
DM	Data Mart
DR	Drilling
DS	Data Store
DSS	Decision Support System
DW	Data Warehouse
DWA	Data Warehouse Administrator
EG	Enterprise Guide
ER	Entity Relation
ERP	Enterprise Resource Planning
ETL	Extract Transform Load
E	Extraction job
T	Transformation job
L	Loading job
ETL APP	ETL Application Server in KI
BI APP	Business Intelligence Server in KI
FASMI	Fast Analysis Shared Multidimensional Information
FIT	Functional Integration Testing
G2G	Good to Great
GB	Gigabyte
GGRE	Geological, Geophysical, Reservoir and Exploration
GUI	Graphical User Interface
HOLAP	Hybrid OLAP
HW	Hardware
ICT	Information and Communication Technology
IO	Input/Output
IO	Integrated Operations
IPL	Integrated Planning
IS	Information System

JSP	Java Stored Procedure
KI	Knowledge Infrastructure
KPI	Key Performance Indicator
MB	Megabyte
MCRS	Management Control and Reporting System
MDDB	Multi-Dimensional Database
MOLAP	Multidimensional OLAP
MS	Microsoft
NF	Normal Form
NT	Windows NT
OE	Operations Excellence
OLAP	Online Analytical Processing
OLT	Online Transaction
OLTP	Online Transaction Processing
OOC	Onshore Operations Center
OS	Operating System
OSC	Onshore Support Center
PRD	Production Environment in the SAP system
Prod	Production Environment in KI
RDBMS	Relational Database Management System
ROLAP	Relational OLAP
SAS APP	SAS Application Server
SC	Service Control
SDDS	Stored Detail Data Store
SPM	Strategic Performance Management
SRP	Production Development Environment in the SAP system
SV	Specialized Maintenance (Spesialisert Vedlikehold)
SVG	Stavanger
SVGAIXSAS	SAS UNIX Server
SVGAPSAS	SAS Windows Application Server
T24	Educational environment in the SAP system
TB	Terabyte
TS2	Test Environment in the SAP system
TSQ	Test Environment in the SAP system
TST	Test Environment in the SAP system
TSZ	Sandbox Environment in the SAP system
WO	Work Order
WRS	Web Report Studio

1 Introduction

1.1 Background

Petroleum activity produces vast amounts of data from many different systems, including geological data, all sorts of measuring data from petroleum production, statistical data, equipment data and data about people: Location, transfers, work place and so on.

ConocoPhillips Norge is at the time undergoing a process where the Data to Decisions project is developing a data warehouse system to be used in e.g. automated reporting across all the departments in the Norwegian branch of the company. Integrating the operations in the company, ConocoPhillips aims at achieving better decision making, both within separate departments and across the domains. The data warehouse system is focused on turning operational data into information and knowledge to support Business Intelligence.

Combining all operational data into one data warehouse creates in the case of ConocoPhillips a data warehouse with over 200 operational systems, each system consisting of several databases and tables. Updating schedule in the data warehouse varies between hourly, daily and monthly for the different operational systems, tables and databases, but most of them run with a daily update schedule. Combined, this makes up for a data warehouse with some thousand transactions per day.

ConocoPhillips Norge commenced in cooperation with SAS Institute (hereby called SAS) the progress of establishing infrastructure and methodology etc. for the data warehouse system in 2005. Although a system like this always needs work like updates and changes, the project has a scope of a couple of years and is supposed to come to a completion during the spring or summer 2007. The basis of this thesis is to explore the data warehouse system at ConocoPhillips and potential possibilities of improvements to the setup of the system. Thoughts around the different environments within the data warehouse are particularly of interest concerning improvements to the system.

The thesis has been specified in collaboration between ConocoPhillips Norge and the student. It is nevertheless objective, and covers an area within data warehousing not very well explored or documented. The thesis may therefore also be of interest to others as well as ConocoPhillips, both with thought to theory around data warehouses, setup of a data warehouse in practice and best practices within such systems.

1.2 Thesis Definition

ConocoPhillips is one of the world's leading integrated energy companies. The Norwegian branch is the third largest energy company in Norway. The core activities of ConocoPhillips Norge are petroleum exploration and production.

In the process of supporting Integrated Operations, the need for large complex data warehouses is explicit; data-jobs like ETL jobs are made to do computations on and analyze data in the data warehouse. These jobs are created in a development environment, and have to be deployed into the production environment. In the production environment, the jobs have direct access to the most current data, or live-data, in the data warehouse.

Importing data into development and test environments locks the production system and must therefore be limited. The need for the test- and development environments to be separate from the production environment is essential, but this creates differences between test data and production data in the input systems. Considering this dilemma, how could such an information system best be designed?

The thesis will document parts of the existing system and identify existing problems, e.g. consistency problems between the different environments; in itself this is a complex task since documentation is scattered and faults might have been corrected in an ad hoc fashion.

Different models on how to set up development, test and production environments will be outlined. These models will be based on reported best practice.

1.3 Delimitations

Even though data warehousing is becoming a more commonly used concept in the industry, there are still mainly two authors standing out in the field of data warehousing. Although these will be the main authors used for theoretical research and literature study, the data warehouse system at ConocoPhillips does not necessarily strictly follow one or both of these schools. As the thesis main focus is on describing the system rather than comparing it to the literature, the system may in some ways differ from the presented theory, and these variations will not be regarded or discussed.

No two data warehouses are exactly alike. This project is focused on exploring ConocoPhillips's particular data warehouse and its related problems, excluding other systems and data sources. The thesis will in this study naturally research into the technology in use for the system as well as conduct a survey of best practice from an external system with similar problem areas. Results and conclusions reached by this thesis may still be applicable to other systems outside of the scope of this thesis, for instance as an example of best practice.

A very important field within a data warehouse system like this is security and restrictions of sensitive or personal data. Many of the security issues are imposed by law while others are restrictions required by company policies. In this thesis there is no room to elaborate on this type of security issues. Nevertheless it is important to know about these problems. Yet, other types of security issues like data integrity may be necessary to touch on as this is more closely connected to the nature of the project.

In the course of elaborating on the issues of the data warehouse system at ConocoPhillips, the Knowledge Infrastructure, this thesis will mainly restrict itself to regarding the two current main issues of the system: Separation of online users from the batch processes on the production server and the question respecting the number of environments in the system. It is likely possible to find other issues of interest as well, but the scope of this thesis allows for only a limited amount of research work to be done.

Nevertheless, some issues regarding general subjects will be outlined to give a more complete understanding of the implementation process of a system such as the one in regard.

Reflecting on best practice from other systems, this thesis will limit itself to examine one system – the SAP system at ConocoPhillips. Inquiring into too many systems would take too much time and steal the focus from the main task of the thesis – the Knowledge Infrastructure.

Suggestions and improvements recommended by the thesis will not be implemented, only explored theoretically for then to give ConocoPhillips and SAS the possibility to consider them. It will be up to ConocoPhillips and SAS to decide whether to test and implement any of the suggestions or not.

1.4 Previous Research and Status in the Field

Today there are many books and business intelligence systems based on data warehouses and the use of these. Mutually for these is that most of them are based on the work of either Kimball [4] or Inmon [2], two of the pioneers within data warehousing, or at most two or three other relatively big names in the field. Most data warehouses can be said to be based on either the Kimball approach or the Inmon approach. However most literature concentrates on how to build data warehouses with emphasis on how the data structures and tables are designed. One of the main issues for ConocoPhillips is however how to set up the logical infrastructure, with thought to how many environments which is favorable and the interaction between those environments. There are few if any articles concerning this area specifically.

Since most data companies building data warehouses are building custom warehouses for each one of their clients, it is hard to find any conclusive facts for how to solve problems in a system like this. Also, papers documenting best practices are nearly impossible to stumble across while these are mostly internal whitepapers if at all existing.

1.5 Literature Study

This study is tied to non-standard systems utilizing technology in constant change. For parts of the thesis it has therefore been difficult to find available literature of satisfactorily quality. In the work of documenting the system of object of survey and in the documentation of best practice, mainly proprietary documentation in combination with interviews with business- and professional personnel was used.

In the context of data warehousing many books describing different aspects of system design is on the market, but two authors stand out as the titans in the field of data warehousing, namely William Inmon [2], who is known as the father of data warehousing, and Ralph Kimball [4]. Their approaches to certain aspects of data warehousing differ greatly. The theory study of this thesis is mainly based on these two authors.

Other important authors concerning data warehouses include Poe [5] and McFadden [3], both with own definitions of a data warehouse or explanations of the key terms in the field. The majority of other existing literature uses the Inmon definition to define the data warehouse, although giving own explanations of the key terms.

2 What is a data warehouse?

This chapter will discuss the basics of data warehouses in order to establish what a data warehouse is and why it is needed in many modern companies. To a certain degree, this chapter will also deal with different aspects of data warehouses to consider when designing a data warehouse.

2.1 Data warehouse

2.1.1 Important concepts in data warehouse technology

Facts

A general sense of a fact is “something that has happened”. A data warehouse is built around these occurrences and saves information about its dimensions (see “Dimension” below). The time this occurrence took place can be a dimension, the place it happened another dimension.

In the literature two types of facts are mentioned. One being event facts and the other is state facts. The description of “something that has happened” could apply to the event fact type, but is not a clear description of a state fact. In the context of a fact table (see “Fact table” below), a fact is a row in the table. Thereupon the attributes of such a row can be described. Some are foreign keys while others are values of measures (see “Measures” below).

Fact table

Thinking graphically, a fact table is often located at the centre of a star schema surrounded by dimension tables. This fact table consists of the measurements (see Measures below), or facts, of a business process. Through these measurements the fact table provides the additive values by which dimensional attributes are analyzed. Analyzing dimensional attributes based on additive values is made feasible by the additive values acting as independent variables.

Fact tables are often defined by their grain (see granularity).

Measures

Measures are the values of a fact. These have a value either in form of a number or some other measurable format. A measure also has an additional element saying how to operate the value, like adding or counting the value over some period of time. To ascertain that no queries resulting in illogical or wrong answers are run, the measures are divided into three classes:

- Additive measures:
  - These can be combined along all dimensions. It may for instance be meaningful to look at the number of accidents on a road at a specific time, date, the age of the driver and the type of car.
- Semi-additive measures:
  - These have one or more dimensions for which they cannot be combined. As an example one can look at a road to find the number of cars on the road at a given time, but one cannot directly summarize the number of cars on that same road during a whole day. Doing this will count the same cars many times.
- Non-additive measures:
  - These cannot be combined along any dimensions. For instance a room-temperature measure cannot be used along dimensions without measuring the mean value instead of summarizing.

One should, as far as possible aim at getting the measures additive to increase the flexibility of the data warehouse. Sometimes it may seem necessary to use text in the fact table. Instead of having information as text in the fact table, one should try to generalize the content and put this in a dimension. If this is not possible, the value does probably not have much relevant information to add to the data warehouse.

Dimension

See section 2.5.2 about Dimension under 2.5 Dimensional modeling later in this chapter.

Granularity

In any general system granularity is a measure of the size of the components, or descriptions of components, that make up the system. Coarse-grained systems are systems of large components, while fine-grained systems consist of small components.

In data warehousing the grain of a fact table represents the most atomic level by which the facts may be defined. Said otherwise, granularity refers to the level of detail of the data stored in fact tables in a data warehouse. For example, one might have a date/time dimension which could be at the year, month, quarter, period, week, day, hour, minute, second, hundreds of seconds level of granularity. Most data warehouses do not go to the second or hundreds of seconds level, but it could be possible.

The granularity of a system would be the lowest level of depth of data; usually data that is at or near the transaction level. Data that is at the transaction level is generally referred to as atomic level data.

This means that by low granularity the registration of the data is very accurate, e.g. every second or so for each transaction. At higher granularity one can either aggregate or sum up the number of instances over time, or register the data more infrequently.

Redundancy

In general terms redundancy means duplication, or exceeding what is necessary or normal. Redundancy in a system may have both negative connotations and positive implications. On the negative side redundancy can represent repetitive and/or superfluous data. While this may result in computational overhead, one may want to introduce redundancy to serve as a duplicate for preventing failure of an entire system.

In a data warehouse redundancy can be a benefit in improving the response time for queries. Because the dimensions in a data warehouse generally takes a relatively small amount of space compared to the entire system (usually < 5 %), redundancy is normally not something one need to worry about, as long as it is kept in the back of ones mind. However, the update time can increase as a consequence of redundancy. For large systems where updates take much time, it may therefore be necessary to avoid redundancy because of the prolonged update time.

Metadata

Data about data is the simplest definition of metadata. Metadata is used to facilitate the understanding, use and management of data, and as such an item of metadata may describe an individual data item or a collection of data items.

In data warehousing metadata is used to describe how data is transformed from an external source into the warehouse. Metadata may be classified into two groups. One for the administrators, describing the data sources, attributes and definitions. The second for end users, explaining what the data represents.

Both of these types of metadata are stored together, but need not be stored at the same place as the data warehouse itself. This is nevertheless often the case since it maximizes the availability of the metadata. Because there is a need to control how the data is transformed while transmitted from the operational database to the data warehouse, metadata in a data warehouse is a very important component.

In a data warehouse context, metadata systems are sometimes separated into two sections:

1. Back room metadata - used for ETL (Extract Transform Load) functions to get OLTP (Online Transaction Processing) data into a data warehouse

2. Front room metadata – used to label screens and create reports

Kimball [4] divides the metadata in a data warehouse in three main categories:

- Source system metadata
  - Source specifications – such as repositories and source schemas
  - Source descriptive information – such as ownership descriptions, update frequencies, legal limitations and access methods
  - Process information – such as job schedules and extraction code
- Data staging metadata
  - Data acquisition information – such as data transmission scheduling and results, and file usage
  - Dimension table management – such as definitions of dimensions, and surrogate key assignments
  - Transformation and aggregation – such as data enhancement and mapping, DBMS load scripts, and aggregate definitions
  - Audit, job logs and documentation – such as data lineage records and data transform logs
- Database Management System (DBMS) metadata
  - DMBS system table contents

- Processing hints

Data marts

A data mart is an extract of a data warehouse, containing a sort of partial image of operational data. In an enterprise wide data warehouse containing all the data from every aspect of the enterprise's operations, all the company departments do not need all the data in the warehouse. Data marts are typically generated in such cases, where a data mart for one specific department only contains the data from the warehouse essential to this department.

This specialized version of a data warehouse does usually not contain all the data from the data warehouse. Its creation is predicated on a specific, predefined need for a certain grouping and configuration of select data. A data mart configuration emphasizes easy access to relevant information helping business people to strategize based on analysis of past trends and experiences.

Data marts may contain modifications on the data from the data warehouse to better handle the anticipated queries. As a data mart is a specialized version of the data warehouse, it is designed to be relevant to one or more business units. Like a corporation may have many business units there may be multiple data marts inside a single corporation's data warehouse.

2.1.2 Description and usage of a data warehouse

Simply put, a data warehouse is a central data storage, a place to gather all information for an enterprise to obtain a collective overview of all the data. Typically a data warehouse will collect data from many and input systems together with external data and combine this to a greater whole. Data warehouses are tailor-made for information purposes such as reporting, business analysis, statistical analysis and operation by objectives. Another important aspect of the data warehouse is to secure high data quality in such a way that the information present in the data warehouse is correct.

Furthermore, one will in a data warehouse save history of the data, information that in time often are lost in the different branch systems. Unlike the source systems of a business, the data warehouse is independent of organizational units and offers self-service solutions designed for end users to get the information they need when they need it.

Historical data and self-service solutions help end users to make decisions based on, among other things, trends instead of just having to rely on current data from the source systems. Data warehouses are accordingly examples of decision support systems (DSS). A DSS can be defined as "a computer-based information system whose primary purpose is to provide knowledge workers with information on which to base informed decisions" [1]. DSS can be divided into data-oriented DSS, model-oriented DSS and process-oriented DSS. A data-oriented DSS uses database systems as source of the decision support, in contrast to a model-oriented DSS which uses mathematical models to support business decisions and a process-oriented DSS which simulates human decision making processes [1]. Data warehouses are the primary example of data-oriented DSS today.

To obtain effective reporting & analysis and control of operations, this is often based on a data warehouse or a data mart. In other words, the data warehouse contains the raw material for management's decision support system. The critical factor leading to the use of a data warehouse is that a data analyst can perform complex queries and analysis on the information without slowing down the operational systems.

Data in data warehouses is frequently heavily denormalized, summarized and/or stored in a dimension-based model, making the data warehouse optimized for reporting and analysis (online analytical processing – OLAP). While data warehouses are optimized for OLAP, operational systems are optimized for simplicity and speed of modification (online transaction processing – OLTP). OLTP systems achieve this through heavy use of database normalization and an entity-relationship model.

2.1.3 Definition of data warehouse

There are two main authors in the field of data warehousing, namely William Inmon, who is known as the father of data warehousing, and Ralph Kimball. Their approaches to certain aspects of data warehousing differ greatly. Industry practitioners are aware of these authors and their differences. Practitioners mainly choose to follow either an Inmon approach, or a Kimball approach. Other data warehousing literature can easily be labeled as more towards Inmon's, or more towards Kimball's ideas. Some of these differences will be highlighted in this chapter. The literature study given in this chapter is mainly based on the work of these two authors.

Inmon [2] defines a data warehouse as a subject oriented integrated, non-volatile, and time variant collection of data in support of management decisions. McFadden et al [3] explain each of the parts of this definition:

1. "Subject oriented: A data warehouse is organized around the key subjects (or high level entities) of the enterprise. Major subjects may include customers, patients, students and products.

2. Integrated: The data housed in the data warehouse is defined using consistent naming conventions, formats, encoding structures, and related characteristics.

3. Time-variant: Data in the data warehouse contains a time dimension so that it may be used as a historical record of the business.

4. Non-volatile: Data in the data warehouse is loaded and refreshed from operational systems, but cannot be updated by end users."

Non-volatile: Data in the data warehouse is loaded and refreshed from operational systems, but cannot be updated by end users."

Kimball et al [4] simply define a data warehouse as "the queryable source of data in the enterprise."

Poe et al [5] define a data warehouse as "a read only analytical database *that is used as the foundation of a decision support system.*"

The majority of literature (excluding Kimball et al [4] and Poe et al [5]) use the Inmon definition to define a data warehouse, as well as their own explanation of the key terms, as for instance the above four clauses quoted from McFadden [3].

2.2 High level data warehouse architecture

The aim of the data warehouse is to give end users (mostly managers) easy access to data in the organization. Kimball et al [4] give a graphic representation of data warehouse architecture. To present organizational data to end users, it is necessary for the data warehouse to capture everyday operational data from the operational systems of the organization. Operational data come from transactional systems (OLTP) designed around relational databases, for instance point of sale systems. Systems like this become the source systems of the data warehouse. Figure 2-1 depicts the operation of the data warehouse in the organization.

Data from the source systems go through a staging area to the presentation servers [4]. This staging process involves four very important operations. Usually the data required for the data warehouse is distributed in various different source systems. Often these systems have different file formats running on different hardware and operating system platforms. The first step in data staging is thus to extract the data from the source systems. Secondly, the data is transformed to the data warehouse format. This phase also involves removing any errors and inconsistencies. Thirdly, the data is loaded into data marts in the presentation server. Scheduling this process is the final task of data staging.

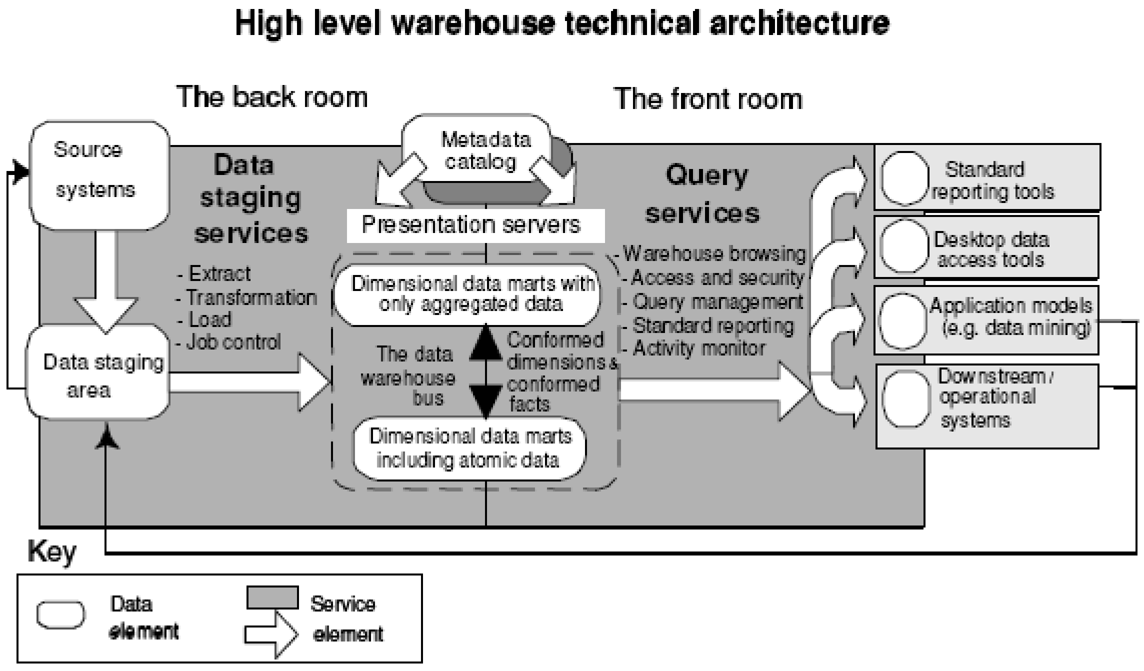


Figure 2-1 – High level data warehouse architecture [4]

2.3 Database normalization

Database normalization is a design technique widely used as a guide in designing relational databases, structuring them in a way that makes them invulnerable to certain types of logical inconsistencies and anomalies. Normalization is essentially a two step process that puts data into tabular form by removing repeating groups and then removes duplicated data from the relational tables.

Normalization theory is based on the concepts of normal forms, and tables can be normalized to varying degrees. A relational table is said to be a particular normal form if it satisfies a certain set of constraints. Relational database theory defines “normal forms” of successively higher degrees of stringency, so e.g. a table in third normal form is less open to logical inconsistencies and anomalies than a table that is only in second normal form.

The goal of normalization is to create a set of relational tables that are free of redundant data and that can be consistently and correctly modified. This means that all tables in a relational database should be in the third normal form (3NF). A relational table is in 3NF if and only if all nonkeyed records are (a) mutually independent and (b) fully dependent upon the primary key. Mutual independence means that no non-key column is dependent upon any combination of the other columns. The first two normal forms are intermediate steps to achieve the goal of having all tables in 3NF.

There are currently several normal forms that have been defined, the three first of which were defined by E. F. Codd. These first three normal forms will be shortly outlined below. The fourth and fifth normal forms (4NF and 5NF) deal specifically with the representation of many-to-many and one-to-many relationships among attributes. Sixth normal form (6NF) incorporates considerations relevant to temporal databases. In addition 3NF the Boyce-Codd Normal Form (BCNF) are the most important normal forms, and so the BCNF will also be briefly covered beneath.

First Normal Form (1NF)

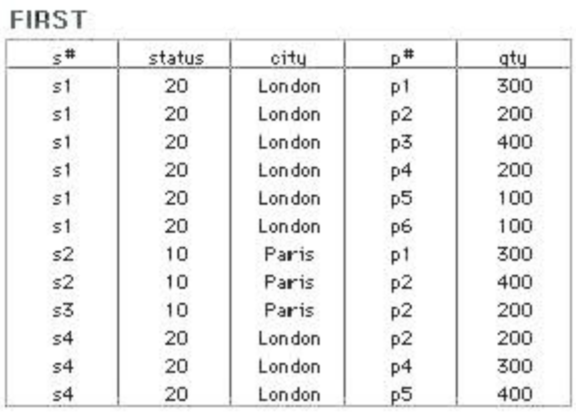
A relational table, by definition, is in 1NF. All values of the columns are atomic. That is, they contain no repeating values.

The criteria for 1NF are according to Codd [6]:

- A table must be guaranteed not to have any duplicate records; therefore it must have at least one candidate key.

- There must be no repeating groups, i.e. no attributes which occur a different number of times on different records. For instance, suppose that an employee can have multiple skills: A possible representation of the employee's skills is {Employee ID, Skill1, Skill2, Skill3 ...}, where {Employee ID} is the unique identifier for a record. This representation would not be in 1NF.

- Note that all relations are in 1NF. The question of whether a given representation is in 1NF is equivalent to the question of whether it is a relation.



FIRST

s#	status	city	p#	qty
s1	20	London	p1	300
s1	20	London	p2	200
s1	20	London	p3	400
s1	20	London	p4	200
s1	20	London	p5	100
s1	20	London	p6	100
s2	10	Paris	p1	300
s2	10	Paris	p2	400
s3	10	Paris	p2	200
s4	20	London	p2	200
s4	20	London	p4	300
s4	20	London	p5	400

Figure 2-2 – Table in 1NF [12]

Although the table in Figure 2-2 is in 1NF it contains redundant data. For example, information about the supplier's location and the company founder has to be repeated for every part supplied. Redundancy causes what are called update anomalies. Update anomalies are problems that arise when information is inserted, deleted, or updated. For example, the following anomalies could occur in the table in Figure 2-2:

- INSERT. The fact that a certain supplier is located in a particular city cannot be added until they supplied a part.

- DELETE. If a row is deleted, then not only is the information about quantity and part lost but information about the supplier as well.

- UPDATE. If supplier “Microflux” moved from Belgium to USA, then two rows would have to be updated with this new information.

Second Normal Form (2NF)

The definition of 2NF form states that only tables with composite primary keys can be in 1NF, but not in 2NF.

> A relational table is in 2NF if it is in 1NF and every non-key column is fully dependent upon the primary key. That is, every non-key column must be dependent upon the entire primary key [12].

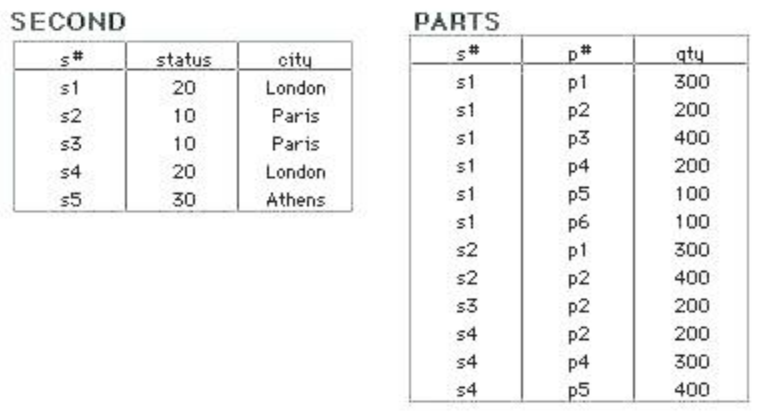
According to Codd [6] the criteria for 2NF are:

- The table must be in 1NF.

- None of the nonprime attributes of the table are functionally dependent on a part (proper subset) of a candidate key; in other words, all functional dependencies of nonprime attributes on candidate keys are fully functional dependencies. For example, consider a “Department Members” table whose attributes are Department ID, Employee ID, and Employee Date of Birth; and suppose that an employee works in one or more departments. The combination of Department ID and Employee ID uniquely identifies records within the table. Given that Employee Date of Birth depends on only one of those attributes – namely, Employee ID – the table is not in 2NF.

Note that if none of a 1NF table's candidate keys are composite – i.e. every candidate key consists of just one attribute – then we can say immediately that the table is in 2NF.

One problem with the design at this stage is that Company Founder and Company Logo details for a given company may appear redundantly on more than one record and so may the Supplier Countries and Continents for a given supplier as well. These phenomena arise from the part-key dependencies of a) the Company Founder and Company Logo attributes of Company, and b) the Supplier Country and Supplier Continent attributes of Supplier. 2NF does not permit part-key dependencies. The issue at hand is corrected by splitting out the Company Founder and Company Logo details into their own table, called Companies, as well as splitting out the Supplier Country and Supplier Continent details into their own table, called Suppliers.



SECOND

s#	status	city
s1	20	London
s2	10	Paris
s3	10	Paris
s4	20	London
s5	30	Athens

PARTS

s#	p#	qty
s1	p1	300
s1	p2	200
s1	p3	400
s1	p4	200
s1	p5	100
s1	p6	100
s2	p1	300
s2	p2	400
s3	p2	200
s4	p2	200
s4	p4	300
s4	p5	400

Figure 2-3 – Table in 2NF [12]

Tables in 2NF, but not in 3NF, like in the table in Figure 2-3 still contain modification anomalies:

- INSERT. The fact that a particular country belongs to a certain continent cannot be inserted until there is a supplier in the country.

- DELETE. Deleting any row in Suppliers destroys the continent information about the country as well as the association between supplier and country.

Third Normal Form (3NF)

The third normal (see Figure 2-4) form requires that all columns in a relational table are dependent only upon the primary key. A more formal definition is:

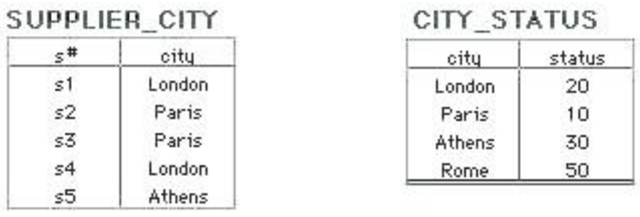
A relational table is in 3NF if it is already in 2NF and every non-key column is non-transitively dependent upon its primary key. In other words, all non-key attributes are functionally dependent only upon the primary key [12].

Codd [6]declares the criteria for 3NF to be:

- The table must be in 2NF.

- There are no non-trivial functional dependencies between non-prime attributes. A violation of 3NF would mean that at least one non-prime attribute is only indirectly dependent (transitively dependent) on a candidate key, by virtue of being functionally dependent on another non-prime attribute. For example, consider a "Departments" table whose attributes are Department ID, Department Name, Manager ID, and Manager Hire Date, and suppose that each manager can manage one or more departments. Department ID is a candidate key. Although Manager Hire Date is functionally dependent on Department ID, it is also functionally dependent on the non-prime attribute Manager ID. This means the table is not in 3NF.

There is still, however, redundancy in the design. The Supplier Continent for a given Supplier Country may appear redundantly on more than one record. This phenomenon arises from the dependency of non-key attribute Supplier Continent on non-key attribute Supplier Country, and means that the design does not conform to 3NF. To achieve 3NF (and, while we are at it, BCNF), we create a separate Countries table which tells us which continent a country belongs to.



SUPPLIER_CITY

s#	city
s1	London
s2	Paris
s3	Paris
s4	London
s5	Athens

CITY_STATUS

city	status
London	20
Paris	10
Athens	30
Rome	50

Figure 2-4 – Table in 3NF [12]

Boyce-Codd Normal Form (BCNF)

BCNF is named after Raymond F. Boyce and Edgar F. Codd, and is a more rigorous version of 3NF. A table is in BC normal form if and only if: [25]

- It is in 3NF, and

- For every one of its non-trivial functional dependencies “X -> Y”, “X” is a superset of candidate key.

Typically, any relation that is in 3NF is also in BCNF, only in rare cases does a 3NF table not meet the requirements of BCNF. However, a 3NF relation will not be in BCNF if: [25]

- There are multiple candidate keys
- The keys are composed of multiple attributes
- There are common attributes between the keys

A somewhat humorous way to remember BCNF is that all functional dependencies are:

“The key, the whole key, and nothing but the key, so help me Codd.” [25]

2.3.1 Denormalization

OLTP applications are characterized by a high volume of small transactions, and it is expected that the database remain in a consistent state after each transaction. OLAP databases are in contrast to databases intended for OLTP operations, primarily “read only” databases. Applications supporting OLAP operations often extract historical data that has accumulated over a long period of time. “Denormalized” or redundant data may for such databases facilitate Business Intelligence applications. Thus databases intended for Online Transaction Processing are typically more normalized than databases intended for Online Analytical Processing.

Denormalized data is often found contained within a star schema. Specifically, the dimensional tables in a star schema frequently contain denormalized data. During ETL processing the denormalized or redundant data must be carefully controlled, and users should not be permitted to see the data until it is in a consistent state. The normalized alternative to the star schema is the snowflake schema.

On smaller computers like in computerized cash-registers denormalization is also used to improve performance. In such systems no changes are to be made to the data as it is only used for lookup (e.g. price lookups), and swift response is crucial.

2.4 Data warehousing (OLAP) versus online transaction processing (OLTP)

Data warehouses are also known as online analytical processing (OLAP) systems because they serve managers and knowledge workers in the field of data analysis and decision making.

Online transaction processing (OLTP) systems, or operational systems, are those information systems that support the daily processing by an organization. OLTP systems' main purpose is to capture information about the economic activities of an organization. One might argue that the purpose of OLTP systems is to get data into computers, whereas the purpose of data warehouses is to get data or information out of computers.

2.4.1 OLTP[1]

"Online Transaction Processing (OLTP) is a class of programs that facilitate and manage transaction-oriented applications, typically for data entry and retrieval transaction processing.

The term Online Transaction Processing is somewhat ambiguous: Some understand "transaction" as a reference to computer or database transactions, while others define it in terms of *business* or *commercial* transactions."

"In large applications, efficient OLTP may depend on sophisticated transaction management software and/or database optimization tactics to facilitate the processing of large numbers of concurrent updates to an OLTP-oriented database."

In OLTP systems relational database design use the discipline of data modeling and generally follow the Codd rules of data normalization in order to ensure absolute data integrity. By using those rules complex information is broken down into its most simple structures (a table) where all of the individual atomic level elements relate to each other and satisfy the normalization rules. Codd defines 5 increasingly stringent rules of normalization, and typically OLTP systems achieve a 3rd level normalization. Fully normalized OLTP database designs often result in having information from a business transaction stored in dozens to hundreds of tables. Relational database managers are efficient at managing the relationships between tables and result in very fast insert/update performance because only a little bit of data is affected in each relational transaction.

OLTP technique is used in a number of industries, for instance banking, airlines, mail-order and supermarkets. OLTP has two key benefits: Simplicity and efficiency. One of the benefits of the OLTP may nevertheless impose a potential problem. The simplicity and the tendency to use the system as a world wide provider for data make it a valuable target for industrial espionage and hackers. The OLTP simplicity of an OLT system forces some databases to go down to complete certain steps of an individual process, thus missing out on some of the efficiency benefits that the system provides.

[1] Sections in quotation marks are taken from [13].

2.4.2 OLAP[2]

“Online Analytical Processing (OLAP) is an approach to quickly provide answers to analytical queries that are dimensional in nature. OLAP is part of the broader category business intelligence, which also includes Extract transform load (ETL), relational reporting and data mining. Databases that are configured for OLAP implement a multidimensional data model, allowing for complex analytical and ad hoc queries with a rapid execution time. The term OLAP was created as a slight modification of the traditional database term OLTP (Online Transaction Processing).”

“The output of an OLAP query is typically displayed in a matrix (or pivot) format. The dimensions form the row and column of the matrix; the measures form the values.”

“In the core of any OLAP system is a concept of an OLAP cube (also called multidimensional cube). It consists of numeric facts called measures which are categorized by dimensions. The cube metadata is typically created from a star schema or snowflake schema” (see sections “Star schema” and “Snowflake schema” under “Dimensional modeling” for a description of these) “of tables in a relational database. Measures are derived from the records in the fact table and dimensions are derived from the dimension tables.”

The following example in Figure 2-5 has only three dimensions so that one can represent it visually, but this is not a limit. It is an OLAP cube representing the oil and gas production of some company. The three axes represented are:

- Platform
- Products
- Time

[2] Sections in quotation marks are taken from [14].

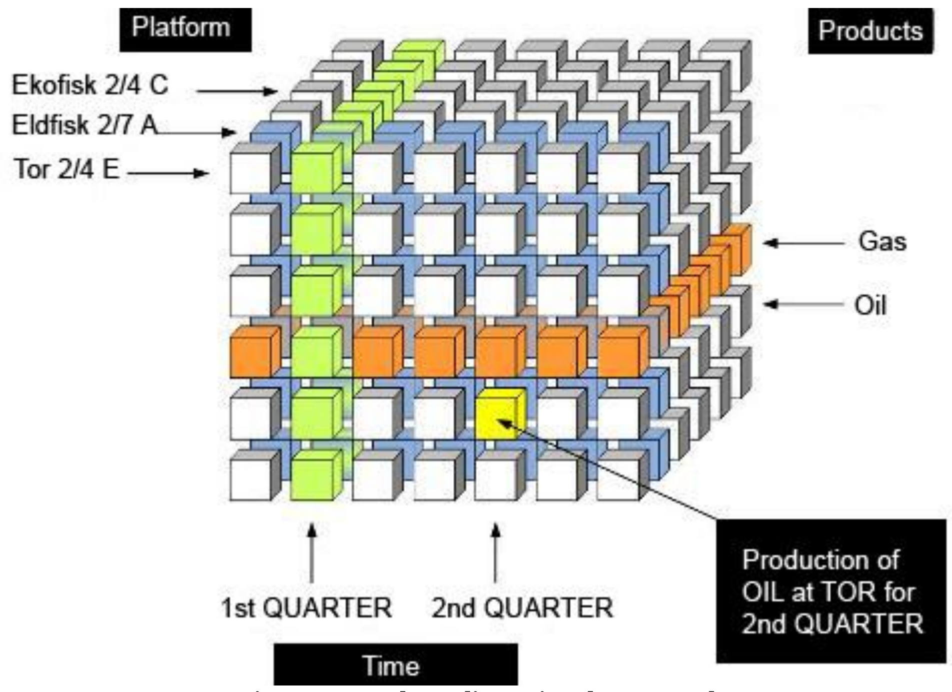


Figure 2-5 – Three dimensional OLAP cube

The crossing of the dimensions allows us to find the result immediately. In fact, as all the intersections of the cube are calculated, to reach desired information becomes an almost instantaneous operation.

“It has been claimed that for complex queries OLAP cubes can produce an answer in around 0.1% of the time for the same query on OLTP relational data. The single most important mechanism in OLAP which allows achieving such performance is use of aggregations. Aggregations are built from the fact table by changing the granularity on specific dimensions and aggregating up data along these dimensions. The number of possible aggregations is determined by every possible combination of dimension granularities.”

“The combination of all possible aggregations and the base data contain the answers to every query which can be answered from the data (as in Gray, Bosworth, Layman, and Pirahesh, 1997). Due to the potentially large number of aggregations to be calculated, often only a predetermined number are fully calculated while the remainder are solved on demand. The problem of deciding which aggregations (a.k.a. views) to calculate is known as the view selection problem. View selection can be constrained by the total size of the selected set of aggregations, the time to update them from changes in the base data, or both. The objective of view selection is typically to minimize the average time to answer OLAP queries, although some studies also minimize the update time as well.”

2.4.3 FASMI

Lately OLAP has become more of a hype-expression and consequently misused. Subsequently OLAP Report [15] suggested in 1995 an alternative and perhaps more descriptive term to describe the concept of OLAP: Fast Analysis of Shared Multidimensional Information, or FASMI. OLAP configured databases borrow aspects of both navigational databases and hierarchical databases which are speedier than their relational kin used for OLTP. FASMI tries describing these using key terms:

- *Fast:*

  The system should give a response to ad hoc queries within 1-20 seconds. Mean response time should be about 5 seconds. If the response time is too long, it is likely that the user will be disrupted in his train of thought and the quality of the analysis will be impaired. Having huge amounts of data, this speed is difficult to achieve - especially when the user's queries become complicated.

- *Analysis:*

  The system should be able to deal with any kind of business logic and statistical analysis relevant for the application and the user. At the same time it should be user friendly for the end user.

- *Shared:*

  Requirements for security must be implemented in the system. Possibility for several jobs to be able to work toward the same data at the same time, as well as write to the same data without encountering error conditions must exist. At this stage many of today's OLAP products fail, as they often assume that the application will only be readable – not writeable.

- *Multidimensional:*

  This is the key demand for an OLAP product. The system must arrange for a multidimensional view of the data, including support for hierarchies and multiple hierarchies as this is the most logical way of analyzing enterprises.

- *Information:*

  By information one means all the data and derivative data necessary to perform the necessary analysis. Relevant aspects in this context are how much information the products can process and how much storage space is needed to save the data.

Using these definitions OLAP is employed as a concept of tools collecting data to a repository, which further lets end users connect to the data to explore and transform this information into knowledge. In this repository the data is handled through cubes containing the aggregated computations of the data in the data warehouse. After defining one's queries, one can get the tool to show the result as graphs or 3-dimensional cubes, making it easier to find the important information.

2.4.4 OLAP structures

There are currently three different ways of structuring an OLAP system. These are ROLAP, MOLAP and HOLAP. Below is a short description of each of them.

R**elational **OLAP

Relational OLAP is the simplest OLAP structure in the sense that it utilizes the structure in the star schema of the data warehouse it works on top of. All calculations of measures happen in runtime leaving more strain on and demanding more of the server.

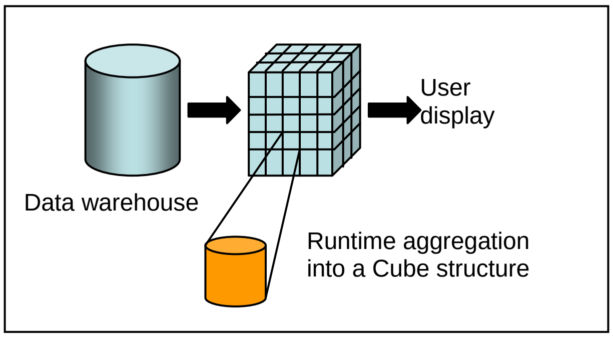


Figure 2-6 – ROLAP [11]

M**ultidimensional **OLAP

In Multidimensional OLAP aggregated data is saved in a multidimensional structure on the OLAP server, increasing performance in query-time considerably, but can require a lot of space because of data-explosion. Many OLAP tools have the possibility to i.a. find the queries that presumably give the best performance improvements, while keeping the volume of the returned values below a given size, e.g. 100 MB.

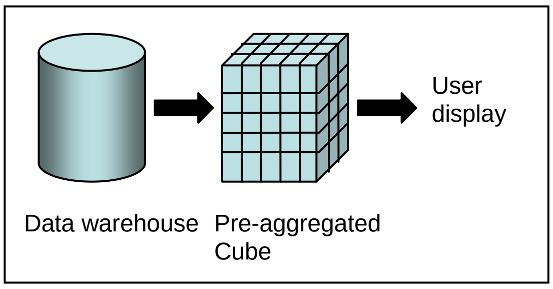


Figure 2-7 – MOLAP [11]

Hybrid OLAP

Hybrid OLAP has been introduced by Microsoft and is basically a mix of Multidimensional and Relational OLAP. Here the aggregated data are saved in a multidimensional structure on the OLAP server, while it keeps the source data in the existing relational structure.

When a request arrives on the database server, OLAP cubes are automatically generated and the restitution of the data is accelerated.

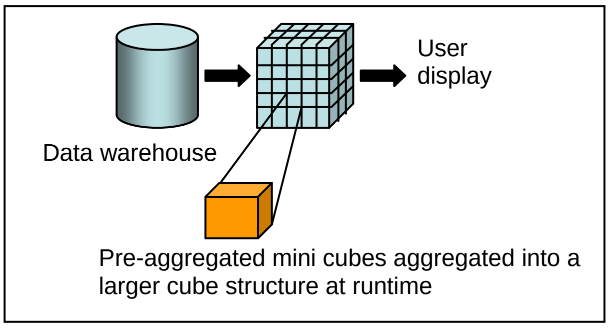

User
display
Data warehouse
Pre-aggregated mini cubes aggregated into a
larger cube structure at runtime

Figure 2-8 – HOLAP [11]

2.4.5 OLTP vs. OLAP

Comparing OLTP with OLAP, OLTP is the primary point of data entry for the business where operational databases help to control and run fundamental business tasks. OLTP contains a snapshot and/or ongoing status of business, whereas OLAP is a consolidation point of data, much of which is pulled from various OLTP databases. OLAP is the source of data and processes for planning and project/problem solving decision support, and shows trends over time and multiviews of the business.

Most OLTP tasks follow standard procedures and well-defined workflows having short, fast updates and queries of few records, with consistently fast, often real-time processing. While this is typical for OLTP, OLAP tasks and processing follow interesting leads and many refinements of models, having primary queries of many records and few updates. Processing time for OLAP is generally fast, but varies widely with task and process.

Table 2-1 shows a comparison of OLTP with OLAP on certain features:

Feature	**OLTP**	**OLAP**
Characteristic	Operational processing	Informational processing
Orientation	Transaction	Analysis
User	Clerk, database administrator (DBA), database professional	Knowledge worker (e.g. manager, executive, analyst)
Function	Day-to-day operations	Long-term informational requirements decision support
Database (DB) design	Entity relational (ER) based, application oriented.	Star / snowflake, subject oriented
Data	Current; guaranteed up-to-date	Historical; accuracy maintained over time
Summarization	Primitive highly detailed	Summarized, consolidated
View	Detailed, flat relational	Summarized, multidimensional
Unit of work	Short, simple transaction	Complex query
Access	Read / write	Mostly read
Focus	Data in	Information out
Operations	Index / hash on primary key	Lots of scans
Number of records accessed	Tens	Millions
Number of users	Thousands	Hundreds
DB size	100 MB to GB	100GB to TB
Priority	High performance, high availability	High flexibility, end user autonomy
Metric	Transaction throughput	Query throughput, response time

Table 2-1 – Comparison between OLTP and OLAP systems

It is difficult to combine data warehousing (OLAP) and OLTP capabilities in one system. The database structure of OLTP is highly normalized with many tables, while an OLAP structure contains fewer tables of simple star or snowflake design. The dimensional data design model used in data warehouses is much more effective for querying than the relational model used in OLTP systems. Furthermore, data warehouses may use more than one database as data source. The dimensional design of a data warehouse is not suitable for OLTP systems, mainly due to redundancy and the loss of referential integrity of the data.

Poe et al [5] stress the fact that analysis using OLAP systems, are primarily done through comparisons, or by analyzing patterns and trends. For instance, sales trends are analyzed along with marketing strategies to determine the relative success of specific marketing strategies with regard to sales patterns. Such analysis is difficult to perform with OLTP systems since the information accessed is stored in different systems across several departments in the organization.

Corey et al [7] highlight the fact that usage of OLTP systems is very predictable. For instance, a bank clerk always performs the same actions on the system. The usage of a data warehouse system on the other hand is very unpredictable. It is not possible to predict which trends will be analyzed by which managers during which time period.

Eckerson [16] argues that the most important difference between OLTP and OLAP systems is that an OLTP system forces business process structure that should not be changed, while OLAP systems need to be changed regularly. He argues that the more often business intelligence (BI) systems are changed, the better they become. They should change often to meet the ever changing needs of the business.

Kimball et al [4] highlight similar differences to those presented in Table 2-1 Inmon [2] presents a total different approach to the development of a data warehouse system. He argues that although OLTP are developed from requirements as a starting point, data warehousing starts at implementing the data warehouse and ends with a clear understanding of the requirements. The data warehouse development lifecycle is datadriven and OLTP are requirements driven.

2.5 Dimensional modeling

Dimensional modeling is somewhat different from its relational counterpart. The term dimensional modeling stems from the fact that dimensional databases are used to slice data along certain dimensions. For instance, a cake factory dimensional database could have dimensions of product, ingredient, shift, employee, supervisor, time, and so forth. Such a database could be used to generate reports that break down total production by each sort of cake, by date and time when cakes were baked, by shift that produced the goods, and so on.

In reporting and analysis, thousands to billions of transactions may need to be reassembled imposing a huge workload on the relational database. Given enough time the software can usually return the requested results, but because of the negative performance impact on the machine and all of its hosted applications, data warehousing professionals recommend that reporting databases be physically separated from the OLTP database.

In addition, data warehousing suggests that data be restructured and reformatted to facilitate query and analysis by novice users. OLTP databases, on the other hand, are designed to provide good performance by rigidly defined applications built by programmers fluent in the constraints and conventions of the technology. Add in frequent enhancements, and too many a database are just a collection of cryptic names, seemingly unrelated and obscure structures that store data using incomprehensible coding schemes – contrary to the suggested easy-to-understand formatting suggested by data warehousing. All factors, while improving performance of OLTP, complicate use by untrained people. Lastly, the data warehouse needs to support high volumes of data gathered over extended periods of time, it is subject to complex queries, and needs to accommodate formats and definitions inherited from independently designed package and legacy systems.

Responsibility for the design of data architecture synergy in the data warehouse is the realm of Data Warehouse Architects. The goal of a data warehouse is to support management and reporting needs by bringing data together from a variety of existing databases or source systems. The generally accepted principle is that data should be stored at its most elemental level because this provides for the most useful and flexible basis for use in reporting and information analysis. There can however be alternative methods for design and implementation of data warehouses due to different focus on specific requirements. There are two leading approaches to organizing the data in a data warehouse. The dimensional approach advocated by Ralph Kimball [4] and the normalized approach advocated by Bill Inmon [2].

Ralph Kimballs [4] “dimensional” approach partitions transaction data into either a measured “facts”, which are generally numeric data that capture specific values, or “dimensions” which contain the reference information that give each transaction its context. Oil extraction would as an example be broken up into facts such as the amount of oil extracted and the current price for crude oil, and dimensions such as date, geographical location, extraction plant and destination for the oil. Using the dimensional approach enables business staff with limited IT experience to easily understand and use the data warehouse. This is one of the main advantages of Kimballs dimensional approach. One of the other main advantages is that the data is pre-joined into the dimensional form, tending to result in the data warehouse to operate very quickly. The main disadvantage of the dimensional approach is that it is quite difficult to add or change later if the company changes the way in which it does business.

Dimensional models consist of one or several fact tables and many dimension tables. For this reason they are sometimes referred to as star schemas - one fact table surrounded by numerous dimension tables.

Bill Inmons [2] approach uses database normalization to store the data in the data warehouse in third normal form (see 2.3 Database normalization). Subject areas reflecting the general definition of the data (customer, product, finance, etc.) are used to group the tables together. This makes it quite straightforward to add new information to the database. While simplicity of adding new information is the main advantage of the normalized approach, the primary disadvantage is the number of tables involved. A large number of tables in the system may result in the production of information and reports to slow down. Furthermore, since the segregation of facts and dimensions in this type of data model is not explicit, it is difficult for users to join the required data elements into meaningful information without a precise understanding of the data structure.

Some data warehouses are implemented using the “snowflake” schema, which is a special case of the star schema. Snowflake simply normalizes one or multiple dimensions, or each dimension might be made up of more than one table. For example, a human resources data warehouse may contain a job dimension, which could be made up of job title and job category tables.

2.5.1 Dimensional and multidimensional database

A dimensional database is one which, rather than storing data in multiple two dimensional tables as a relational database does, represents key data entities as different dimensions.

Using relations in a relational database management system, one can achieve multi-dimensional data structures. Another way of implementing multi-dimensional data structures is by using multi-dimensional databases.

When attempting to store a multi-dimensional data structure in a two-dimensional relational database management system (RDBMS), the result is a star schema. A star schema model is a representation of a central fact table with foreign keys to many dimension tables, used as a means of storing data based on a set of known database dimensions. Adding foreign keys in the primary dimension tables, referencing additional dimensional data, the result is a snowflake schema. The snowflake schema is a normalized implementation of dimensional data, and does not increase the dimensionality of the model. Multi-dimensional databases also cannot increase the dimensionality of the data as the dimensionality is defined by the dimensional foreign keys in the fact table.

In a star schema all the (dimension) tables are referenced by one single fact table. This makes queries very fast, seeing that the only existing references are found in the central fact table. Since snowflakes add foreign keys in the primary dimension tables, often resulting in a significant impact on query performance, the use of snowflakes in a relational dimensional model is generally discouraged. Denormalizing the “outlying” dimensional data into a primary dimension table eliminates snowflakes.

When database administrators and designers need to store huge volumes of organizational data with very high transaction rates, the result is often the use of a relational DBMS. Structuring the data in normalized tables, and the use of entity-relationship (ER) modeling, have become a popular way of achieving such a system. Although simple to design and operate, accessing data from relational databases often requires complex join-operations almost impossible for untrained end users to perform. Thus, the simplicity of relational databases fall down when it comes to end users running queries.

In a multi-dimension database system each individual data value is contained within a cell accessible by multiple indexes. When presenting the data to the user in a way as to representing a multi-dimensional array, the result is a more readable view of the data because our perspectives of the data are more compactly represented in the report.

2.5.2 Dimension (data warehouse)

A dimension in data warehouse is a data element that categorizes each item in a data set into non-overlapping regions.

A great deal of duplication can be avoided if dimensions can be shared between multiple cubes. This is a keystone of the Kimball data warehouse methodology, and the process of sharing dimensions between multiple cubes is called “conformed dimensions”.

An example of a dimension in a data warehouse is when the data has a time stamp. If the data in the database has the attribute fields “day”, “month”, “week” and “year”, these could all be members of the Time dimension within a data warehouse. It would then be possible to categorize each report by either filtering based on the time dimension or displaying results broken out by the time.

A data warehouse cube is frequently composed of both dimensions and measures.

2.5.3 Star schema

Currently the most common data management system in organizations is relational databases. Using a star schema enables multi-dimensional database (MDDB) functionality using a traditional relational database. The star schema is also easily understandable. While most fact tables in star schema are in third normal form (3NF), dimensional tables are in de-normalized second normal form (2NF). Normalizing dimensional tables makes them look like snowflakes, and the same problems of relational databases arise. Complex queries are needed, making it difficult for untrained end users, or business users, to understand the meaning of the data. Highly normalized tables make reporting difficult and applications complex.

A star schema, Figure 2-9 and Figure 2-10, consists of fact tables with a compound primary key and dimension tables, and is regarded as the simplest data warehouse schema. The fact table has one segment for each "dimension", and contains the quantitative or factual data about a business. These facts are often numerical, additive measurements and can consist of many columns and millions or billions of rows. Dimension tables are usually smaller and hold descriptive data that reflect the dimensions, or attributes, of a business. The name star schema is derived from the fact that the schema is shaped like a star (see Figure 2-10 and Figure 2-12).

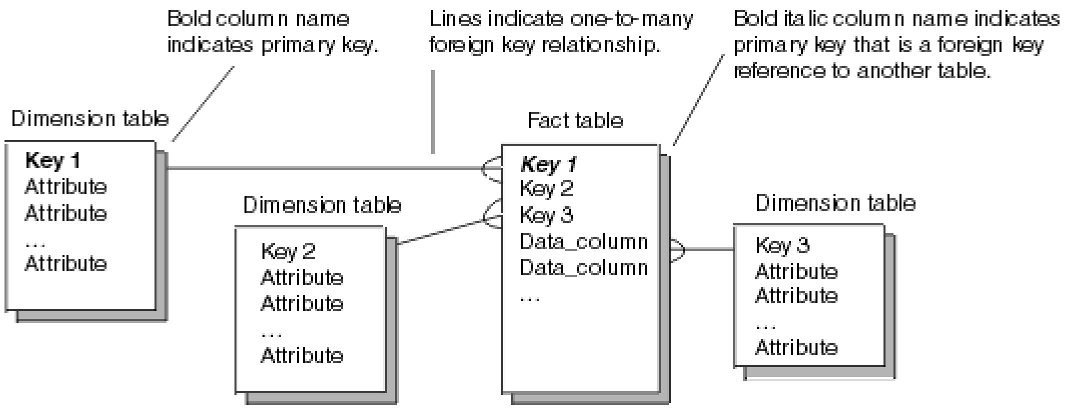


Figure 2-9 – Generic star schema layout [17]

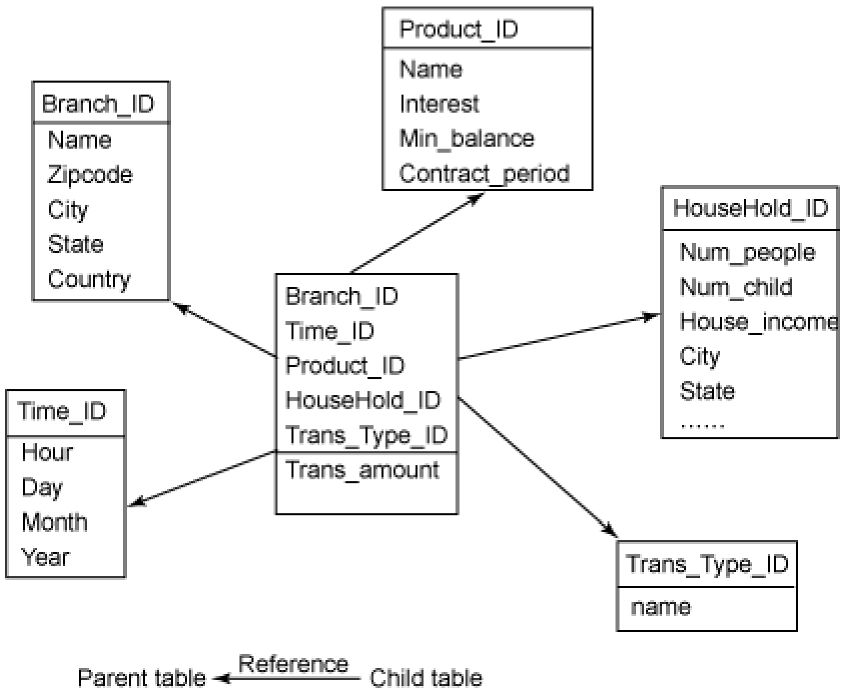


Figure 2-10 – Example of a star schema [18]

The physical structure of fact and dimension tables is the same, and they differ from each other only in their use within a schema. It is however important to understand the logical differences between fact and dimension tables. As explained by IBM Software information center [17], consider how an analyst looks at business performance:

- A salesperson analyzes revenue by customer, product, market, and time period.
- A financial analyst tracks actuals and budgets by line item, product, and time period.
- A marketing person reviews shipments by product, market, and time period.

"The facts - what is being analyzed in each case - are revenue, actuals and budgets, and shipments. These items belong in fact tables. The business dimensions - the *by* items - are product, market, time period, and line item. These items belong in dimension tables." [17]

"For example, a fact table in a sales database, implemented with a star schema, might contain the sales revenue for the products of the company from each customer in each geographic market over a period of time. The dimension tables in this database define the customers, products, markets, and time periods used in the fact table." [17]

“The terms fact table and dimension table represent the roles these objects play in the logical schema. In terms of the physical database, a fact table is a referencing table. That is, it has foreign key references to other tables. A dimension table is a referenced table. That is, it has a primary key that is a foreign key reference from one or more tables.” [17]

2.5.4 Snowflake schema

Snowflake schema, illustrated in Figure 2-11, Figure 2-13 and Figure 2-14, is an extension, or variation, of the star schema where each point of the star expands into more points. The term snowflake schema describes in other words a star schema structure normalized through the use of outrigger tables, where i.e. dimension table hierarchies are broken into simpler tables.

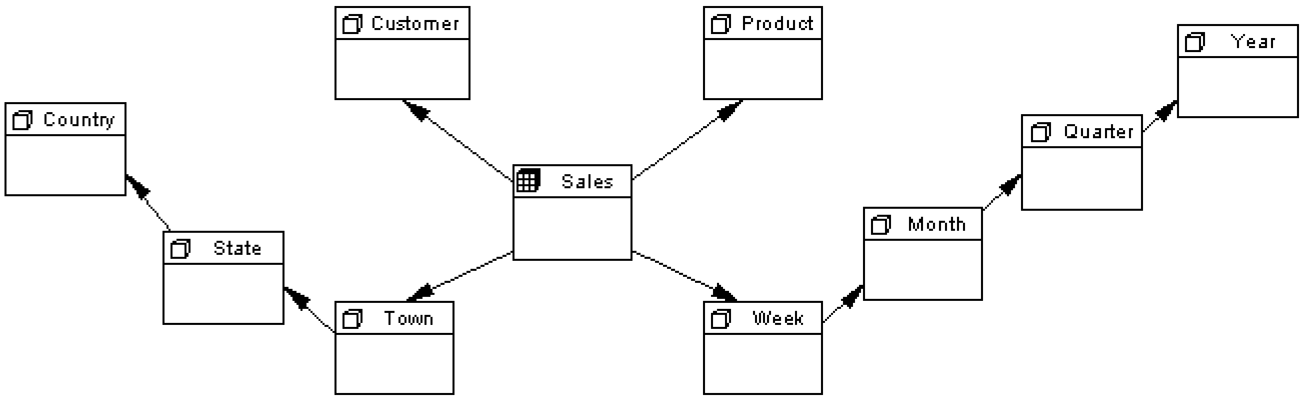


Figure 2-11– Snowflake schema: a central fact table connects a number of dimension tables, which in turn connect other dimension tables. A chain of connected dimension tables is called a dimension hierarchy [19]

The snowflake schema has received its name because it looks like a snowflake. Like the star schema it consists of a fact table connected to a number of dimension tables. The difference is that the dimension tables in a snowflake schema are in turn connected to other dimension tables. The graphics in Figure 2-12 and Figure 2-13 artistically illustrates the notion of a star schema versus a snowflake schema.

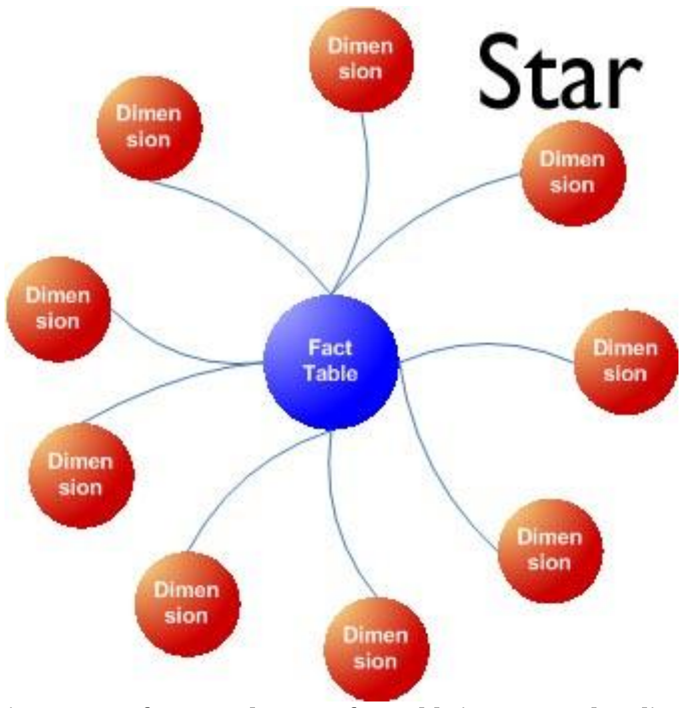


Figure 2-12 – Basic concept of a star schema: a fact table is connected to dimension tables [20]

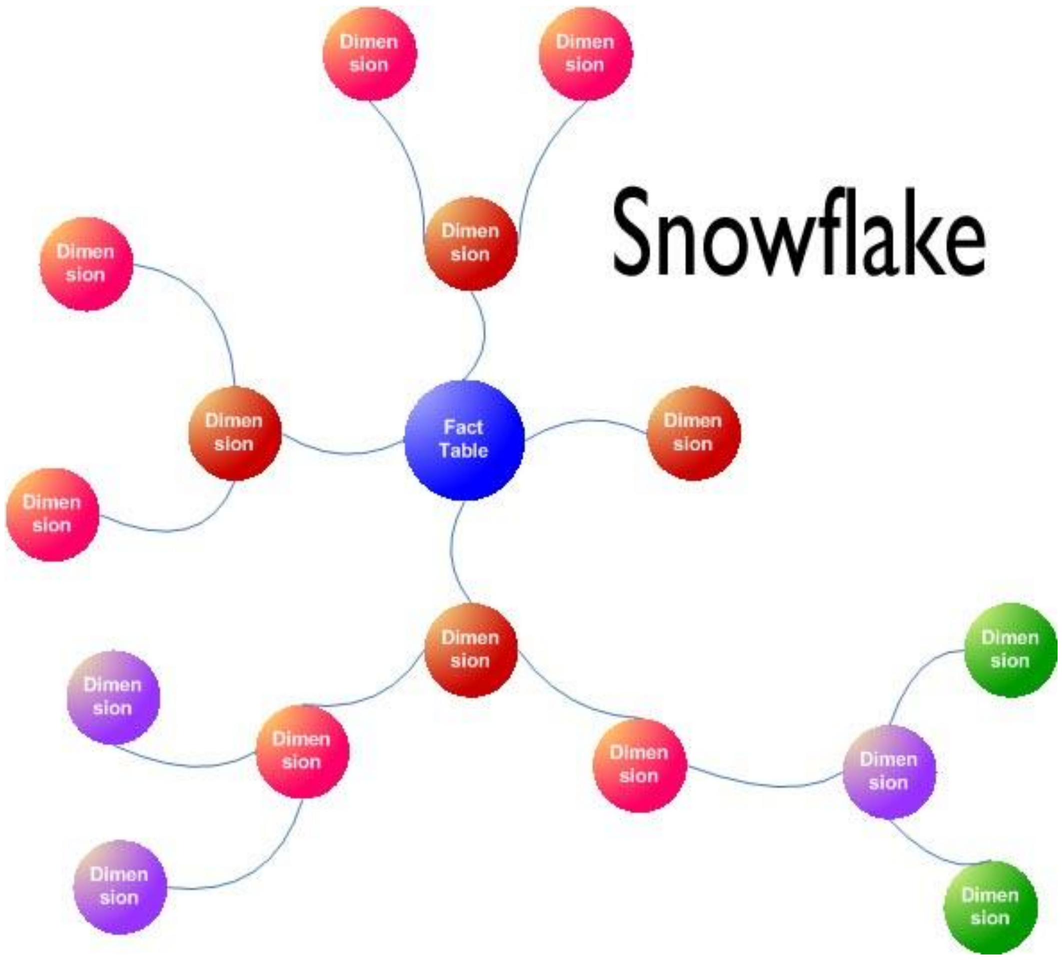


Figure 2-13 – Basic concept of snowflake schema: dimension tables are connected to other dimension tables [20]

Because the tables describing the dimensions are normalized the snowflake schema is a more complex schema than the star schema (see example of a snowflake designed database in Figure 2-14). While the snowflake schema may reduce the disk storage requirements and improve some queries by joining smaller lookup tables, the main disadvantage is the additional maintenance efforts needed due to the increased number of lookup tables. Many database designers keep away from snowflake schemas as a general rule of thumb, as they most likely will cost a lot also in terms of query times.

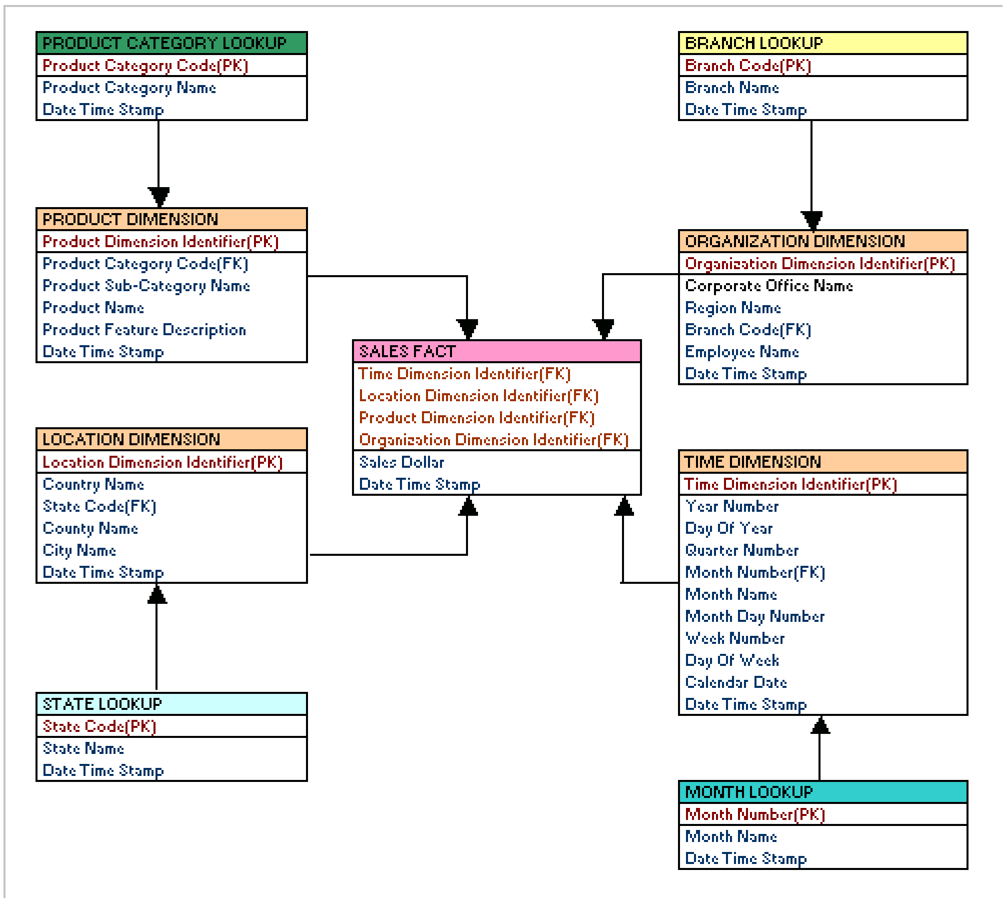


Figure 2-14 – Example of a sales database in snowflake design

3 Background for the system at ConocoPhillips

3.1 *About ConocoPhillips*

3.1.1 Global

ConocoPhillips is an international vertically integrated energy company and the third largest integrated energy company in USA, based on market capitalization, oil- and gas reserves and production.

By vertically integrated [21], in contrast with horizontal integrated [22] companies, one means companies that are united through a hierarchy and share a common owner. Usually each member of the hierarchy produces a different product or service, and the products combine to satisfy a common need.

One of the best examples of vertically integrated companies is in fact the oil industry. Oil companies, both multinational (such as CP) and national often adopt a vertically integrated structure, meaning that they are active all the way along the supply chain. This chain stretches from locating crude oil deposits, drilling and extracting crude, transporting it around the world, refining it into petroleum products such as Petrol, to distributing the fuel to company-owned retail stations where it is sold to consumers.

World-wide CP is the sixth largest publicly owned energy company based on oil- and gas reserves, and the fifth largest refining company. Headquarters are located in Houston, Texas, while other offices are located in around 20 countries on all continents except the Antarctic. The company was founded by the merger of Conoco Inc. and the Phillips Petroleum Company in 2002, and in Europe it operates Jet filling stations.

3.1.2 Norway

ConocoPhillips Norge (COPNO) is currently the third largest energy company in Norway, employing about 1770 people as of September 2006. The company’s headquarters in Norway is located in Tananger, just outside of Stavanger.

COPNO’s main activities consist in oil- and gas prospecting and production, and thus the company holds a strong position in vast oil- and gas fields on the Norwegian continental shelf. The foundation for ConocoPhillips’s (CP) activities in Norway is the Ekofisk field, for which the company is the operator.

Having an average production of 225.000 barrels oil equivalents per day in 2005, CP Norge stands for about 14% of the total oil- and gas production in this new, global company. CP Norge is also the largest company unit outside of USA measured in number of employees.

3.2 Project background

The data warehouse currently in development at COPNO is a result of the Data to Decisions (D2D) project. The data warehouse, or Knowledge Infrastructure provided by the D2D project is a means to supporting Integrated Operations (IO). Integrated Operations is a partial goal of the larger “From Good to Great” (G2G) project.

3.2.1 G2G

G2G is a project aiming at identifying all opportunities for work process enhancements, cost reduction, and other value adding opportunities in the Greater Ekofisk Area. According to the manager of the Ekofisk operating unit Brage Sandstad “the primary focus is to identify significant performance enhancements in operations and maintenance, drilling and well service, logistics and work related processes, rather than to redesign the organization”

Trond-Erik Johansen general manager Greater Ekofisk business centre, who is the champion of the G2G program emphasizes that “the primary focus of the program is to build on our previous and existing initiatives to improve the efficiency of our work processes, removing the frustrations which all of us experience in our daily work. Continuous improvement shall take us to new levels of operational effectiveness”.

With respect to IO the aim of G2G is to develop a suite of enhanced work processes and an organizational structure, with clear roles and responsibilities. The problem was that advanced technology was available in the Operations Centers but it was primarily used for communications with offshore installations rather than using the sources of data streaming into the centers to analyze, optimize and predict performance. An area of focus is cross-functional collaboration, providing visual information management systems that will give decision makers instant access to the information they require

3.2.2 D2D and IO

High performance requires more than data acquisition and control. Organizations need a more comprehensive approach to business intelligence that enables them to create value from data by providing timely, reliable and relevant information for making strategic, managerial and operational decisions at all levels.

The Data to Decisions project is to enable the Greater Ekofisk Operating Unit to take full use of the Onshore Operation Center (OOC) and ensure continuous improvement. One of the main hurdles is to combine and work with all data to produce relevant information. Needed is both real- and near-real-time data from various sources, information to do efficient integrated planning, and removing the gap between strategic/tactical plans and budgets and the individual activities. COPNO feel that they have a competitive advantage in having Onshore Support centers running because they have demonstrated the ability to combine existing data to create new information. This project will ensure that this advantage is being enhanced.

Prior to the project, a pilot was performed to verify that the software and methodology can create and support the sustainable knowledge environment they need. This pilot was successfully completed and found that the toolset can create the Knowledge Infrastructure needed. Also the toolset for management of change proved to be able to support a sustainable Knowledge Infrastructure with limited resources. Finally, the analytical, optimization, and governance tools have been verified both in the IS and business community to be very good for the tasks.

There is an inherent need to be able to combine nearly all CP’s data sources to create common information out of the data. This need has existed in the organization for a long time. It is especially important to develop a fully integrated plan for the Greater Ekofisk Area and be able to optimize production since the ability to combine data and produce information has become a necessity. They have chosen to call this Knowledge Infrastructure.

Their definition of Knowledge Infrastructure extends that of a standard data warehouse defined in literature by Kimball, Inmon and others. Later this thesis will try to explore some of these differences and what makes this Knowledge Infrastructure special.

Following are two examples of what the D2D project addresses, quoted from Pål Navestad, also explaining the need for Integrated Operations (IO) and Integrated Planning (IPL) – IPL being a part of or a tool used for IO.

"In December (2004) we had a full plant-shutdown at the Teesside oil receiving and processing plant. This was because of drilling mud being accidentally sent through the pipeline. After the incident it was found that 2 days prior the incident there were irregularities in the separator at EKOJ platform that is the main processing plant offshore. Further, we found that 2-3 days prior to this, well X-2 at EKOX (one of the wellhead platforms) showed strange behavior. In the investigation we found that the reason for the strange behavior was problems while drilling well X-42 at the same platform. Wells X-42 and X-2 are right next to each other. Well X-42 had lost circulation of drilling fluid (i.e. mud). With the proposed system in place, we would with 90 % certainty have been able to send an alarm to the engineers in the Onshore Operation Center that they should look into well X-2. Further, having had full access to Drilling plans and problems, OOC engineers would probably have seen the problem and could have taken appropriate steps to avoid the Teesside shutdown. The shutdown took about 12 hours; taking out all production from both Ekofisk and Eldfisk, as well as third party producing through Ekofisk and fields in the British North Sea sector such as the COP operated J-block. Total production loss to all parties was around 500,000 bbl of Crude Oil, and 200,000 bbl of crude to COP effectively deferring this production for 25 years and destroying almost 100% of its value through this deferral. The economic value of this amount of production is $10MM to COP. Although an example like this does not occur more than occasionally in a year there are minor incidents at weekly intervals, which can be prevented through the information improvements that this project will deliver.

A solution to avoid double entry of data and ensuring that everybody has the same information is in development. Without this system we had to build an integrated plan using meetings and manual input into Microsoft Project. Activities for the next 90 days are shown. Since everything has to be put in manually, one can't provide the level of detail to work off this plan, and when data is changed in the original systems these have to be re-entered, leading to irregularities. We have two people that, because of the inefficiency and productivity lost through the current processes, can't do any effective planning or optimization on the facilities, but mainly spend their available time finding relevant data and presenting it. In the system in development, the data is loaded automatically, the knowledge infrastructure would ensure that all that the data is consistent, appropriate to its intended purpose, and we could go down to the lowest level of detail when needed. The time horizon can be expanded, and we have the means to support full integration of strategic plans and budget down to the activity level. This would free up the time it takes for data entry and improve the quality of data being used. This gives the opportunity to improve plans and activities giving the whole organization a more efficient and better way of working. The ability to do better integrated planning is estimated to have at least a 10 % efficiency increase translating to $ 30 million (gross) or $10 million (net) operating cost improvement yearly.

From this high-level business need we have found that the key enabler is the development of a common knowledge infrastructure or metadata related to these areas of the business. Metadata consists of two parts; technical and business. The aim of the business metadata is to define the rules and terms making it possible for business people to reference all data consistently as if it were coming from the same system. The use of information is especially large in conjunction with production optimization in the Onshore Operations Center. Here the aim is to be able to take all available real time data and make it possible for business people to find the relevant data using only normal business terminology. It shall also be possible to combine these data with data at different levels of granularity for instance plan data. Rules for automatic checking of data quality and alerts for strange behavior shall be documented and used. The use of the knowledge infrastructure will not be limited to real time data but will encompass all data and transferring this into information."

Basically the Integrated Operations system at CP is the application of collaboration technologies, concepts and methodologies across the Operations Excellence[3] (OE) systems to raise asset awareness, improve decision making, and support the Business Units in the delivery of their goals.

IO means increased data capture (more data, with greater frequency) from all parts of oil and gas fields, together with the use of this data in real-time or near real-time modeling efforts to optimize the performance of the reservoir, wells, and facilities. One of the most important features of IO is the integration of analytical tools facilitating understanding of how each part of the system effect each other, and allows asset teams to optimize reservoirs, wells, and facility performance, as well as better manage performance of the overall system.

Integrated operations results in operations improvement across the enterprise yielding better data quality and validation, centralized monitoring and diagnosis and improved operations through the use of prediction and optimization tools. The ability to project data and information wherever and whenever it is needed gives better off-site monitoring and geographic transparency, removing distance barriers. Other benefits of IO includes more reliable and capable remote actuation systems extending the potential for distant/centralized monitoring and control, and potential for new business models to emerge to take advantage of the data and operational flexibility afforded by IO.

[3] The Good to Great project is one of the aspects of Operations Excellence.

3.3 Introduction to ConocoPhillips's Data Warehouse

3.3.1 Background

ConocoPhillips (COP) has through the "Good to Great Program" and the "Data to Decisions" project put a lot of effort into creating an environment where information is proactively used to improve operations. The environment consists of a "Knowledge Infrastructure" which is used to collect, integrate and organize data from a variety of sources and "smart applications" which leverages the information in application areas, e.g. Integrated Planning.

ConocoPhillips (COP) has identified a need for supporting its operations with better information and better analysis. The business benefits COP wants to achieve are related to:

- Avoiding unwanted incidents
- Maximizing production
    - Reducing unplanned loss of production
    - Reducing planned loss of production
- Efficient use of resources

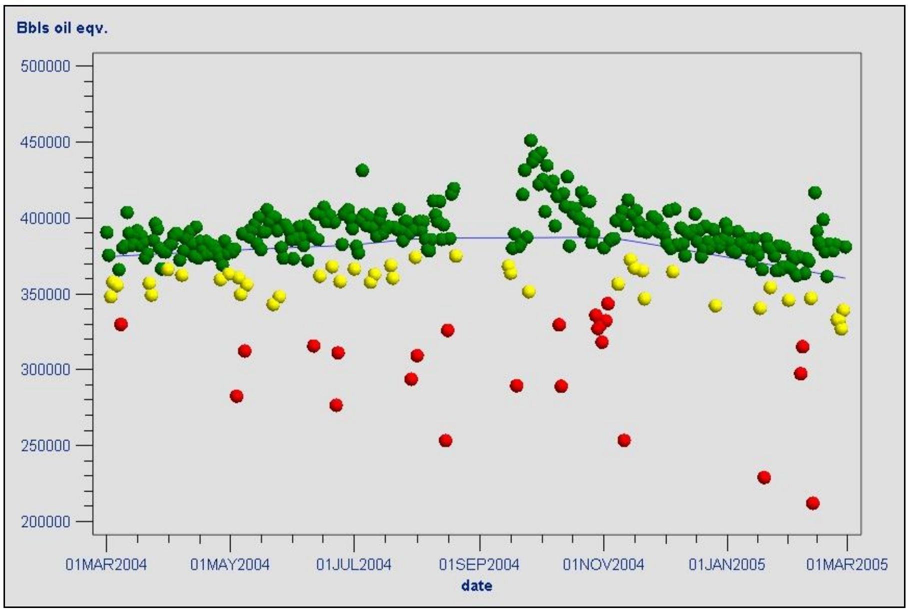


Figure 3-1 – Potential benefits of reducing loss of production [8]

Figure 3-1 indicates some of the potential if one is able to reduce loss of production, either planned or unplanned. At a production rate at close to 400' barrels per day, each day represents a monetary value of approximately MNOK 150. Every dot in the figure represents one day of production. The red and yellow dots represent a loss from the normal production, and green dots represent production within or above the normal quantity. Avoiding only one or some of the red or yellow dots, that is daily loss from normal production resulting from e.g. poor planning of maintenance, can save the company tens of MNOK justifying the expenses of the implementation of a complex data warehouse system.

In order to reach the objective of maximizing production a major goal is to move from a reactive to a proactive mode in operations as shown in Figure 3-2.

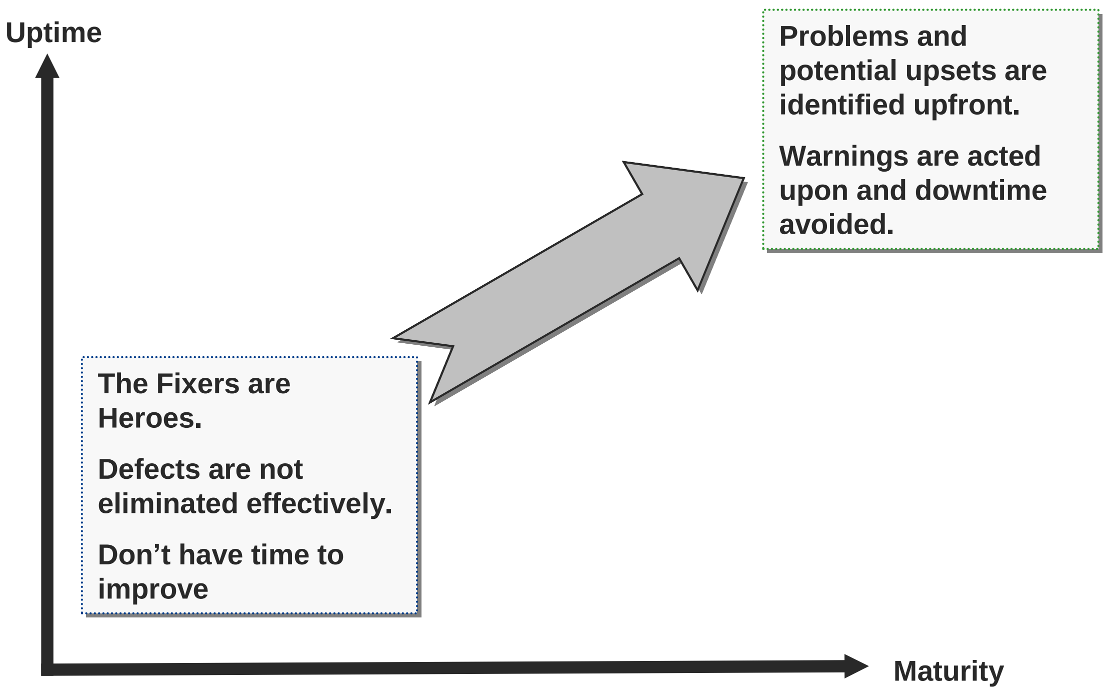

Uptime
Problems and potential upsets are identified upfront.
Warnings are acted upon and downtime avoided.
The Fixers are Heroes.
Defects are not eliminated effectively.
Don't have time to improve
Maturity

Figure 3-2 – From reactive to proactive mode [8]

Several programs and initiatives supports this transition, the "From Good to Great" program and Onshore Operating Centre (OOC) are vital elements.

Access to relevant information is a key enabler for these initiatives and in many cases also an integral part of the initiative. An example is "Integrated Planning" which would be almost impossible to implement without integrated information systems supporting the planning process.

3.3.2 About the system

ConocoPhillips uses the term "Knowledge Infrastructure" to describe the technology, architecture, data and metadata it has put in place to collect, organize, analyze and present information about its operations (see Figure 3-3). The "Knowledge Infrastructure" is now put in place and is leveraged across several subject areas.

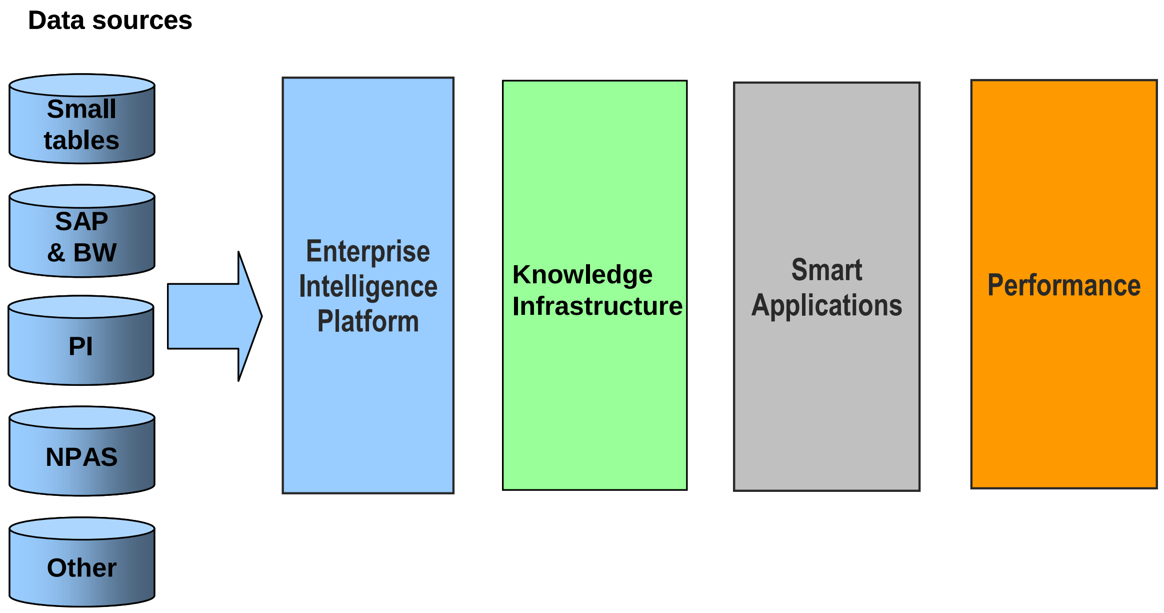


Figure 3-3 – The value chain "From data to performance" [8]

The following main areas are currently being addressed by ConocoPhillips:

- Integrated Planning
    - One integrated plan
    - Automated Scheduling
    - Optimised Scheduling
    - Manual changes and write back to source systems
- Well data
    - Alerts and alarms related to production – asset integrity

- Production reports
- Maintenance optimisation related to production
- Forecasting and trend graphs related to production – advanced analytics

▪ Performance Management

- Business Process Support (MCRS meetings)
- Daily Morning reports
- Strategy Visualisation and Measurement

Other emerging areas are well integrity and integration of cost/finance information into operational information. Increasing the capability of merging data from different subject areas e.g. GGRE, prod well data and well service is a focus area as well. There may also be a potential in safety incident management, e.g. emergency response, communication info about situation and so on.

Increased quality and trust in existing information is also an important item on the agenda.

Data sources

ConocoPhillips started development of their Knowledge Infrastructure within the Data to Decisions project in 2005, with a timeframe of at least one more year as of today. Currently CP’s Knowledge Infrastructure has some 30 source systems, although about 200 potential source systems have been identified. SAP, the main source system, is a system collecting data from yet other sources. Other important sources include:

- DaWinci
- PI
- NPAS
- Microsoft Project
- Manual KPI’s

- MS Excel

- Project systems at super contractors – e.g. contractors used for geology surveys.

- Div. databases e.g. Oracle databases from drilling, well service, loss and historical well data.

The fact that some systems like SAP contains data collected from other systems introduces a question about where to maintain the main sources. Having a source system structured like SAP in the KI, or data warehouse, means that it is not always a given who is in charge of the information or the department where the information comes from. These questions will not be addressed in this thesis, but are mentioned to point out some of the difficulties and complications around a large system like CP's KI. If the data warehouse system is not carefully designed with thoughts to such problems it may very well lead to more serious troubles business wise. Problems like those mentioned also indicate the importance of the metadata and the right use of this.

4 System structure and design

In data warehouse relation, COPNO’s Knowledge Infrastructure is unique in two areas. It is conceptually one of the largest existing data warehouses with respect to the number of columns. There are probably many data warehouses around the world containing more tables and rows, but very few containing more columns than COPNO’s KI. There are in other words vast amounts of data with connection to each other needed to be combined in a meaningful matter.

The second matter making the KI special is the way CP handles calculated data within the data warehouse. In a traditional data warehouse all the sources “shall” be transaction systems. Data from the transaction systems are either transferred directly into the data warehouse, or going through a staging area to get all the data on the same format before going into the warehouse. In ConocoPhillips’s system the data from the sources may be processed and performed calculations upon before they are brought in to the warehouse. These new data are in turn new sources in the system. This is the reason why COPNO has chosen to call their system Knowledge Infrastructure instead of data warehouse.

4.1 Conceptual Design of Knowledge Infrastructure

Contrary to what was discussed in the theory chapter (Chapter 2), the Knowledge Infrastructure is not strictly based on one single approach to data warehousing. SAS does however, like most data warehouse practitioners, mostly base their system on the theory of Inmon and Kimball. At COPNO SAS has based the Knowledge Infrastructure mainly on the Inmon approach. That is to say that the KI follows the data mart method.

4.1.1 Segments and data flow

Splitting the KI up in logical parts, one can say it consists of five segments as depicted in Figure 4-1 - first part being the sources as shown in the figure below. As these sources are independent proprietary systems, they would not normally be seen as a part of a data warehouse system. Nevertheless they constitute a part of the Knowledge Infrastructure as this concept covers the information flow from source to business intelligence e.g. in the form of reports.

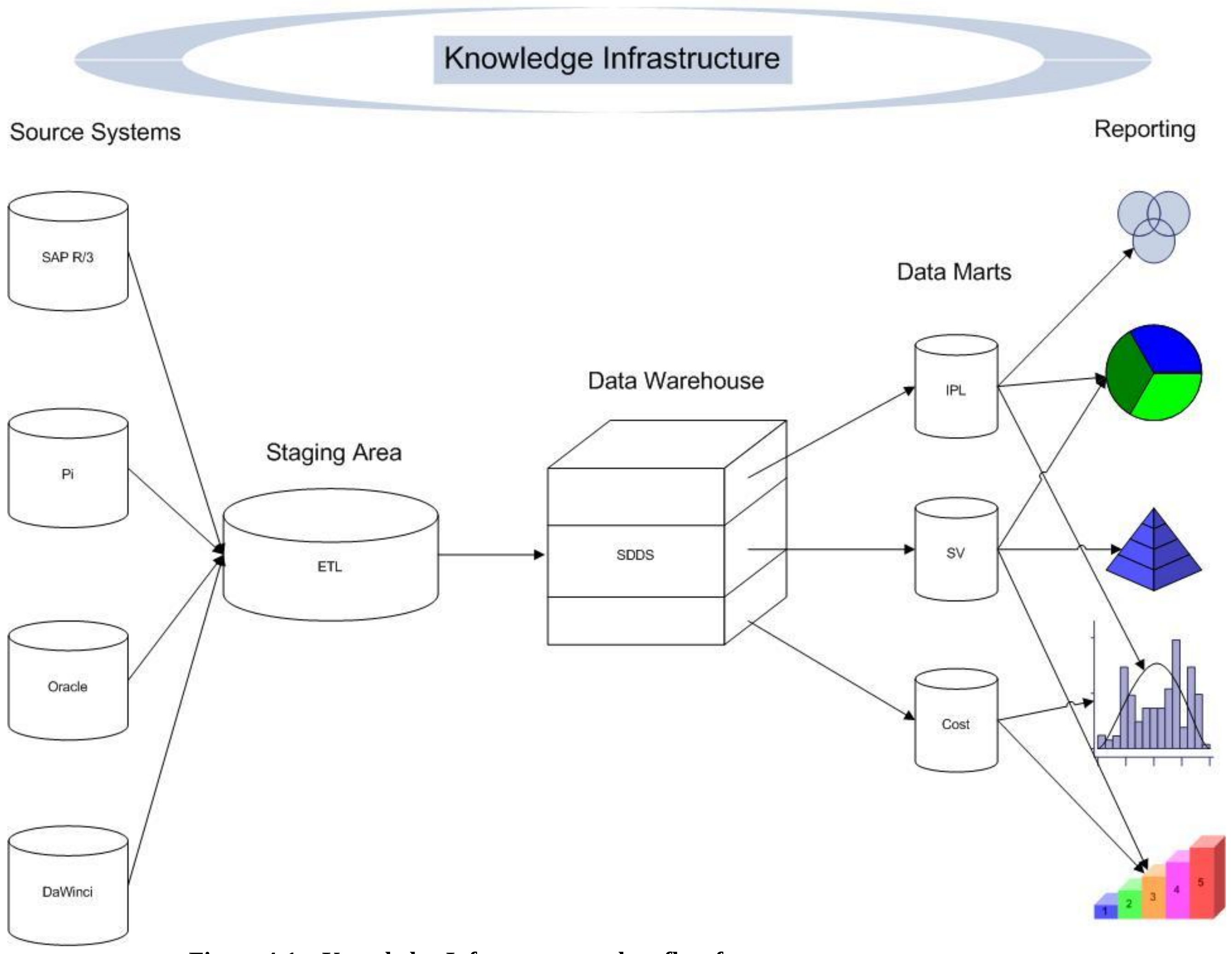


Figure 4-1 – Knowledge Infrastructure; data flow from source to report

Between the data warehouse, or main data storage, and the source systems is an intermediate step called staging area. This is where data needed in the warehouse is extracted from the sources, transformed into the desired format and loaded into the warehouse. Loading the data into the data warehouse implies that the staging area saves the data in the Stored Detail Data Store (SDDS). Optimally all the data in the KI should exist in the SDDS. Successfully implementing this means that all other components (i.e. data marts) in the KI can find the needed data in the SDDS and will go here to get them. With thought to consistency of the data within the data warehouse this is an important aspect.

After entering the SDDS, the next step for the data is the data marts. Data marts are virtually smaller data warehouses containing snapshots or segments of the main warehouse. By partitioning the data in the data warehouse into data marts, the corporation can easily separate and collect the different data of importance to each part of the business structure. The Knowledge Infrastructure system is divided into 20 data marts, each representing one part of the business structure. Examples of these are Integrated Planning (IPL), Specialized Maintenance (SV) and Cost. Put in other words, COPNO has designed and developed a data warehouse system, the KI, successfully reflecting their business structure.

Creating “specialized” (with regard to what data they contain) data marts cf. the data mart method, is a good way of designing the data warehouse to reflect the business structure. When all the information needed in a business domain is easily accessible through a separate data mart, the process of converting the data into information becomes much easier with regard to finding and filtering the data of interest.

On top of the data warehouse system, software to produce readable and understandable information of the data constitutes the fifth segment of the KI. Business intelligence software of various characters is collecting data mainly from the data marts to generate reports, statistics and different kinds of analysis. Even though this may be the least technically complicated of the five segments, it is nevertheless the most visible and perhaps the most important stage. The resulting reports etc. are often the very reason why most companies choose to implement a data warehouse system in the first place.

4.1.2 Exceptions

Usually reporting tools only have to use information from one single data mart. Exceptions do however occur and in those cases it may be necessary to utilize data from two or more data marts. Special jobs like these are not always possible to generate using the point-and-click functionality of the development tools, and it may be necessary to make a “stored process”. A stored process is simply put a hard-coded job, meaning that the job is more or less manually programmed and can do everything the programming language and system allows for.

One of the drawbacks of this system is the possibility a user has to bypass the data marts to go directly to the SDDS, staging area or even the source systems to fetch data. Not only does this increase the chance of inconsistency of the data in the warehouse system, but it raises some questions about security. This is in other words an undesirable feature of the system. As of writing ConocoPhillips is installing a new separate server to physically separate the data marts from the rest of the system. After the implementation of this new server, there will be two more or less identical sets of data marts and the new server only contains a duplication of the marts. Software analyzing data from the warehouse will then only be able to read data from the data marts on the separate server. By doing this one will also effectively eliminate those jobs bypassing the data marts or other segments because the data does not need to be transformed from the state they are in on the source systems. “Quick” jobs also bypassing these segments of the system will likewise be eliminated.

All the segments shown in Figure 4-1 are stored as separate spaces on the KI data server, with exception of the sources and the reports. Understandably the sources, being solitary systems, do not have anything physically to do with the data warehouse system (other than the physical extraction of data from - and in some cases loading of data into - the source systems). Source data is in other words stored on the servers of their proprietary systems. The end-information in the KI, such as reports, can therefore be argued to be a part of the KI not having a separately defined storage space.

4.1.3 Metadata and reporting

Graphically one can imagine the whole previously portrayed system is build on top of a metadata layer as depicted in Figure 4-2. All metadata is stored on a separate server with pointers to its appurtenant data. As mentioned, reports and KPIs and such do not have a separate storage facility, but are stored as metadata on the metadata server. Most reports and KPIs are dynamic, meaning that they generate their result from the newest available data from the data marts. Every time someone opens a dynamic report, the report file points to the metadata, which again points to the real data of interest. Further the report file tells the application what to do with that data – how to present it graphically, calculations and aggregations and so on. Alternatively, if there is no need for updated data or the data never is updated, there is the possibility to make a static report. Static reports use stored data and will therefore not need to have calculations or aggregations or anything like that performed on the data every time the report is opened. As such the metadata layer lies beneath the whole system maintaining things like the report settings.

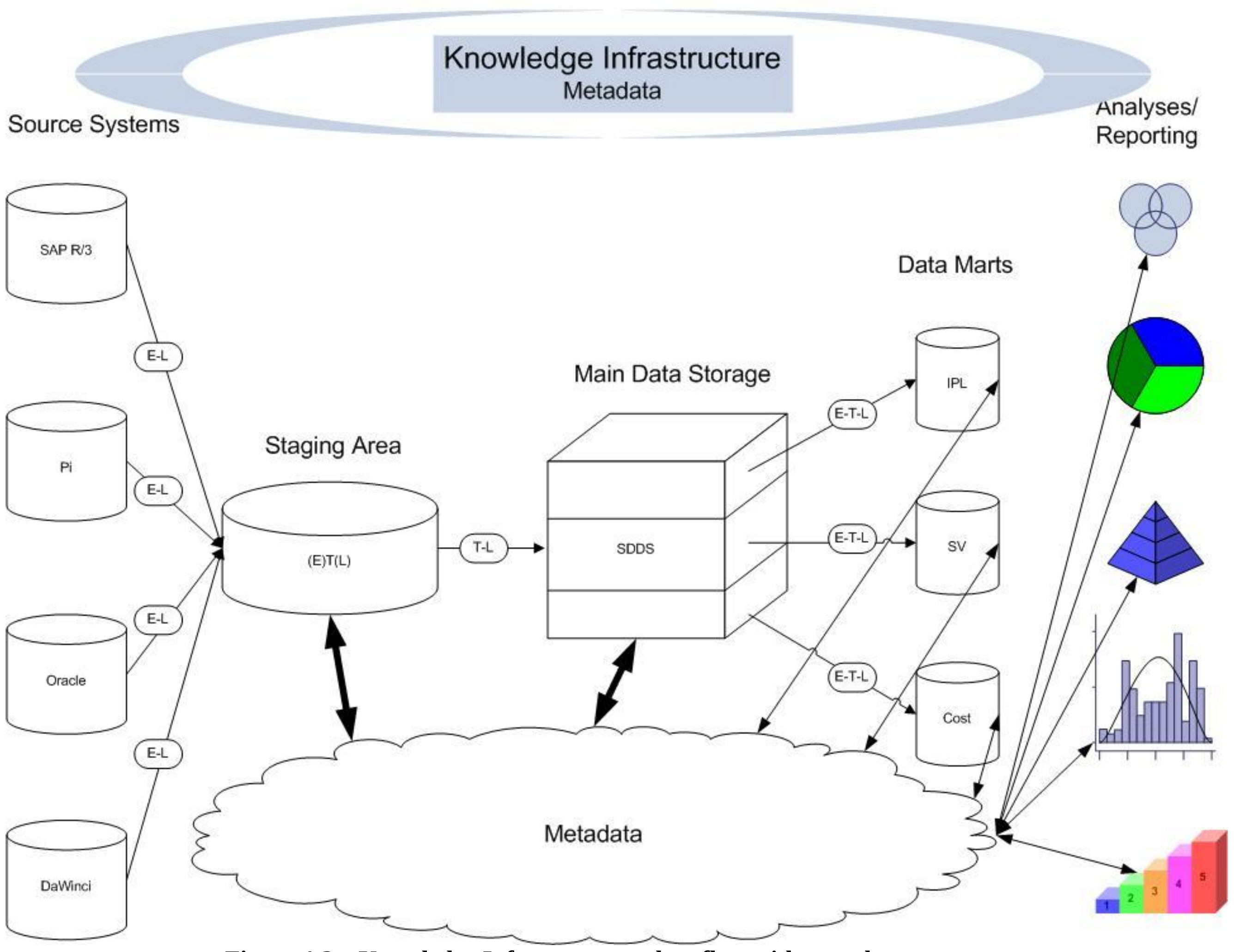


Figure 4-2 – Knowledge Infrastructure; data flow with metadata

4.1.4 Processed data as new source

As the thesis has touched on earlier, the Knowledge Infrastructure gives the opportunity to do calculations on data in the warehouse in order to allocate the calculated data back into the source systems. These calculations are most likely to occur at the data mart level as shown in Figure 4-3, creating a sort of loop in the information flow. Calculated data then act as an input to the source systems, acting as a new source for the data warehouse or KI.

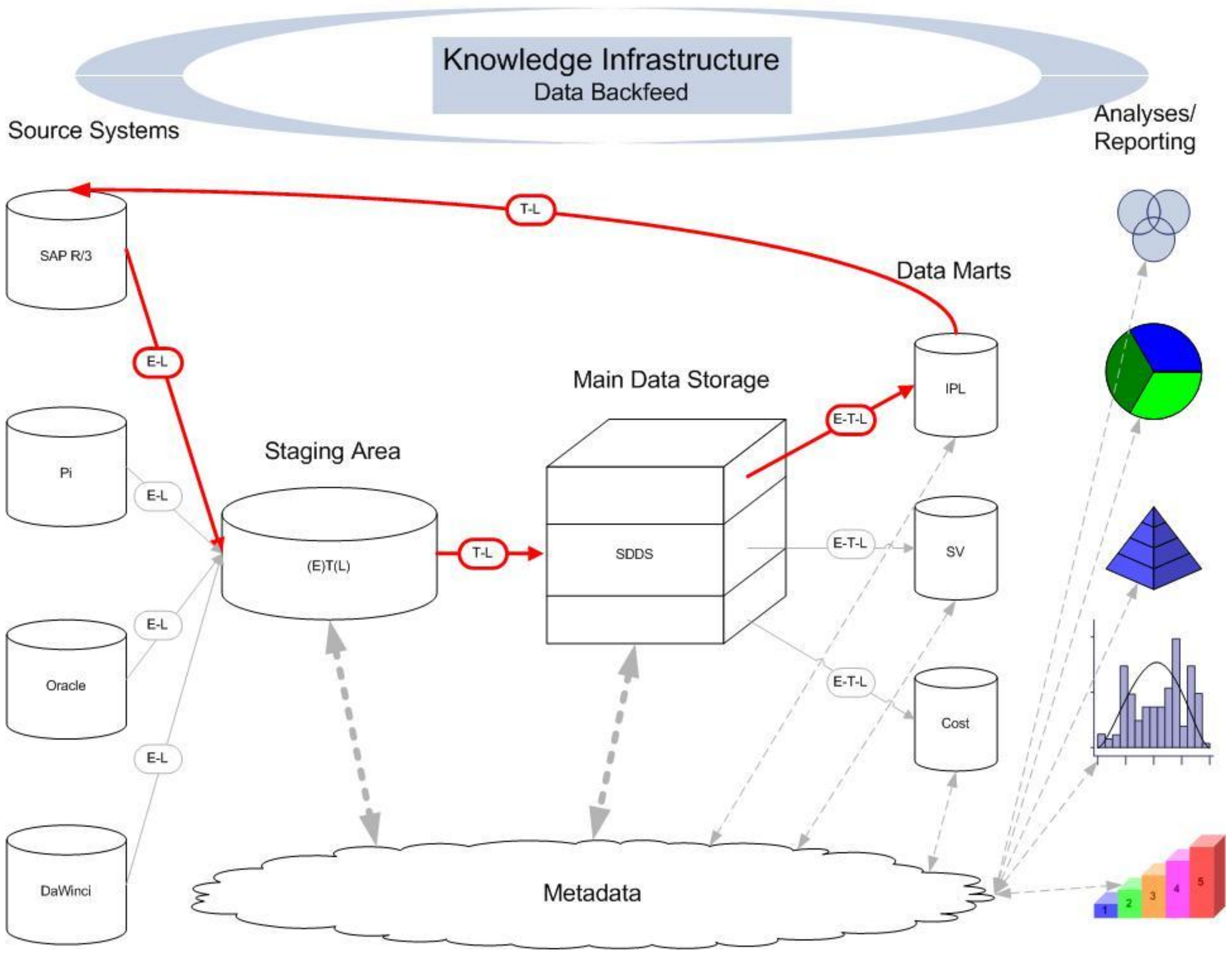


Figure 4-3 – Knowledge Infrastructure; feedback to source system

4.1.5 Observing the logical parts of the KI system

The logical division of the KI into the above mentioned parts is possible to observe through the data warehouse applications. Figure 4-4 below illustrates this as a hierarchy in a folder-tree view.

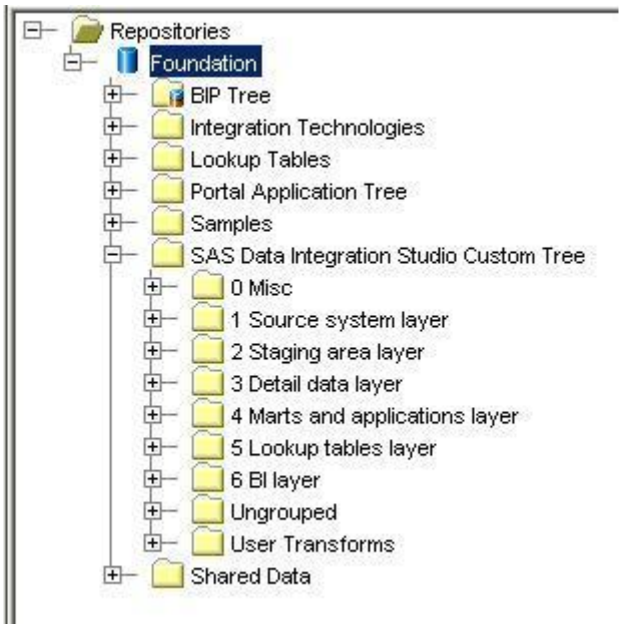
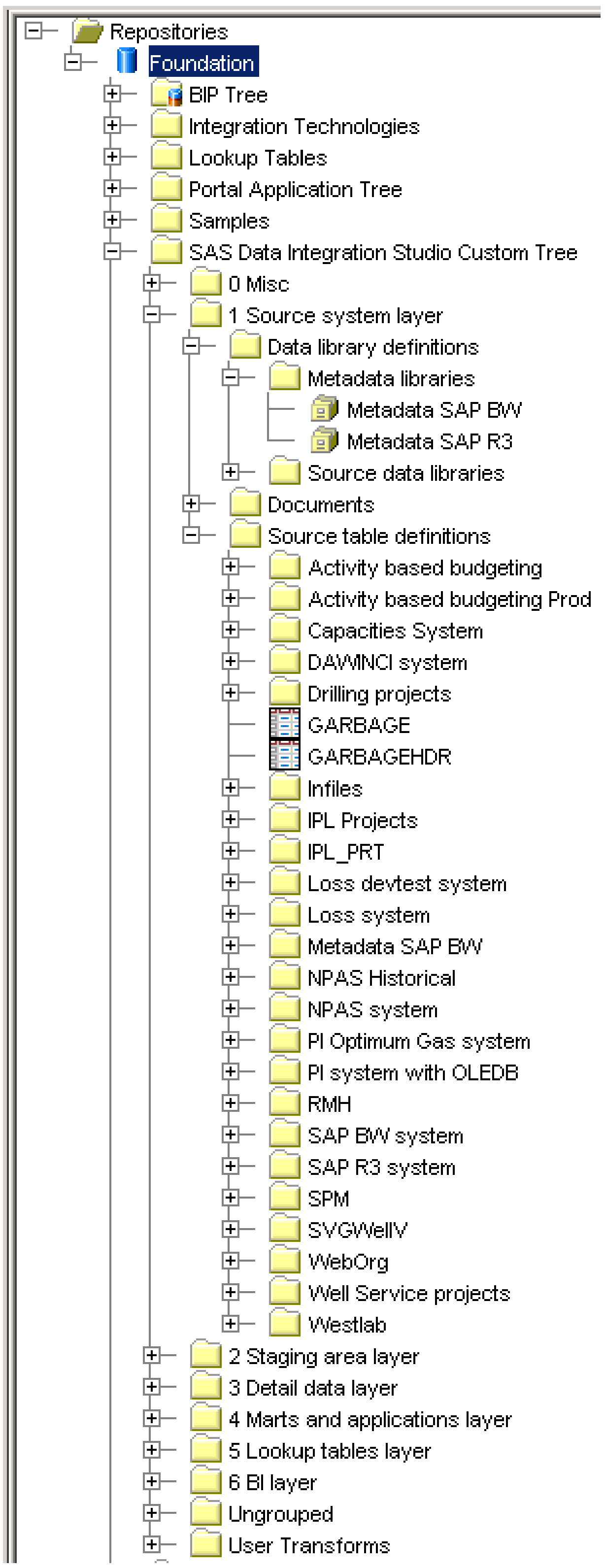

Repositories
Foundation
BIP Tree
Integration Technologies
Lookup Tables
Portal Application Tree
Samples
SAS Data Integration Studio Custom Tree
0 Misc
1 Source system layer
2 Staging area layer
3 Detail data layer
4 Marts and applications layer
5 Lookup tables layer
6 BI layer
Ungrouped
User Transforms
Shared Data

Figure 4-4 – DI Custom Tree; Layers

The expanded folder shows the expanded custom tree in one of the data warehouse applications. Layer 1 through 4 contains the important folders illustrating the previously described structure of the Knowledge Infrastructure.

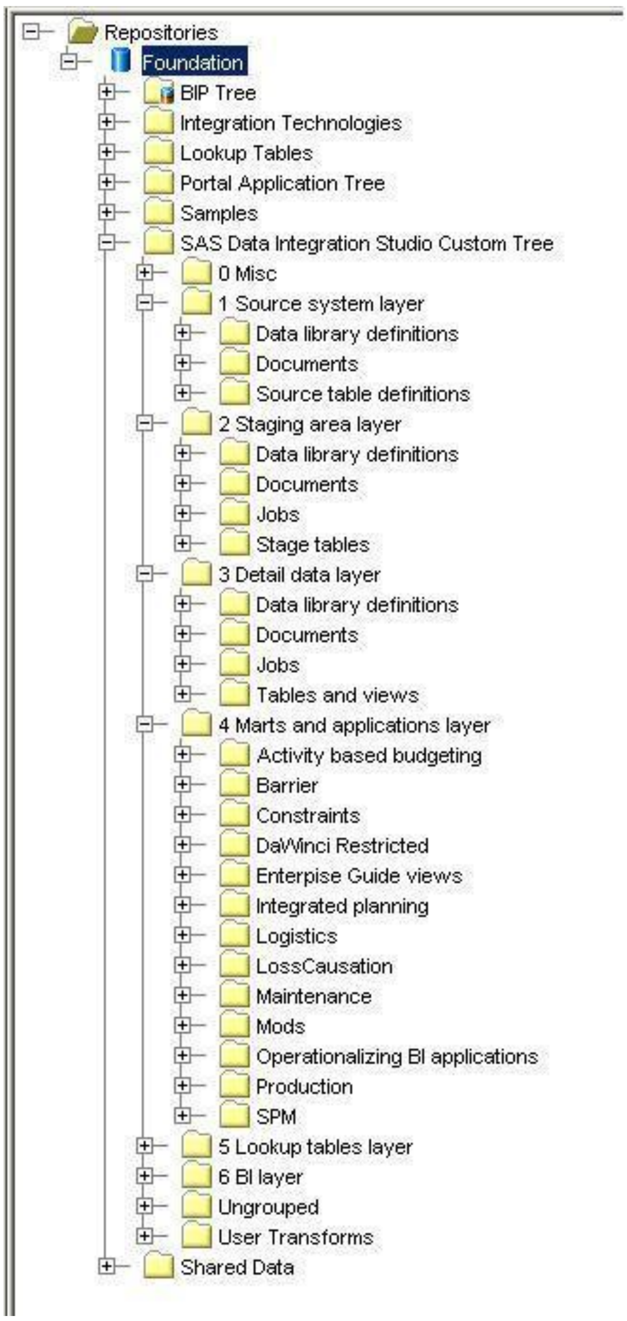
Expanding layer 1 – the source system layer, as seen in Figure 4-5 below, one can see that it contains the definitions for the tables from the source systems. Every layer also has a linkage to the metadata for the data contained in the respective layer. Also indicated in the picture are the main source systems. Manual sources and more custom sources than the automatic systems are not visible in the tree.



```
Repositories
  Foundation
    BIP Tree
    Integration Technologies
    Lookup Tables
    Portal Application Tree
    Samples
    SAS Data Integration Studio Custom Tree
      0 Misc
      1 Source system layer
        Data library definitions
          Metadata libraries
            Metadata SAP BW
            Metadata SAP R3
          Source data libraries
        Documents
        Source table definitions
          Activity based budgeting
          Activity based budgeting Prod
          Capacities System
          DAWINCI system
          Drilling projects
          GARBAGE
          GARBAGEHDR
          Infiles
          IPL Projects
          IPL_PRT
          Loss devtest system
          Loss system
          Metadata SAP BW
          NPAS Historical
          NPAS system
          PI Optimum Gas system
          PI system with OLEDB
          RMH
          SAP BW system
          SAP R3 system
          SPM
          SVGWellV
          WebOrg
          Well Service projects
          Westlab
      2 Staging area layer
      3 Detail data layer
      4 Marts and applications layer
      5 Lookup tables layer
      6 BI layer
      Ungrouped
      User Transforms
```

Figure 4-5 – DI Custom Tree; Source Systems

Figure 4-6 gives an overview over the four layers expanded. As expected, with exception of layer 4, all the layers contain the same folders. These are the “Data library definitions” containing metadata information (and in layer 1 information about the source systems data libraries), “Jobs” accommodates the extraction jobs, transforming jobs and loading jobs – also known as the ETL jobs. The folders named “Tables and views” should be pretty much self-explanatory as to what is stored in those branches of the tree, but this is where the tables and views (logical subset – as opposite to extracts which are physical subsets) containing all the data in the respective layers are located.

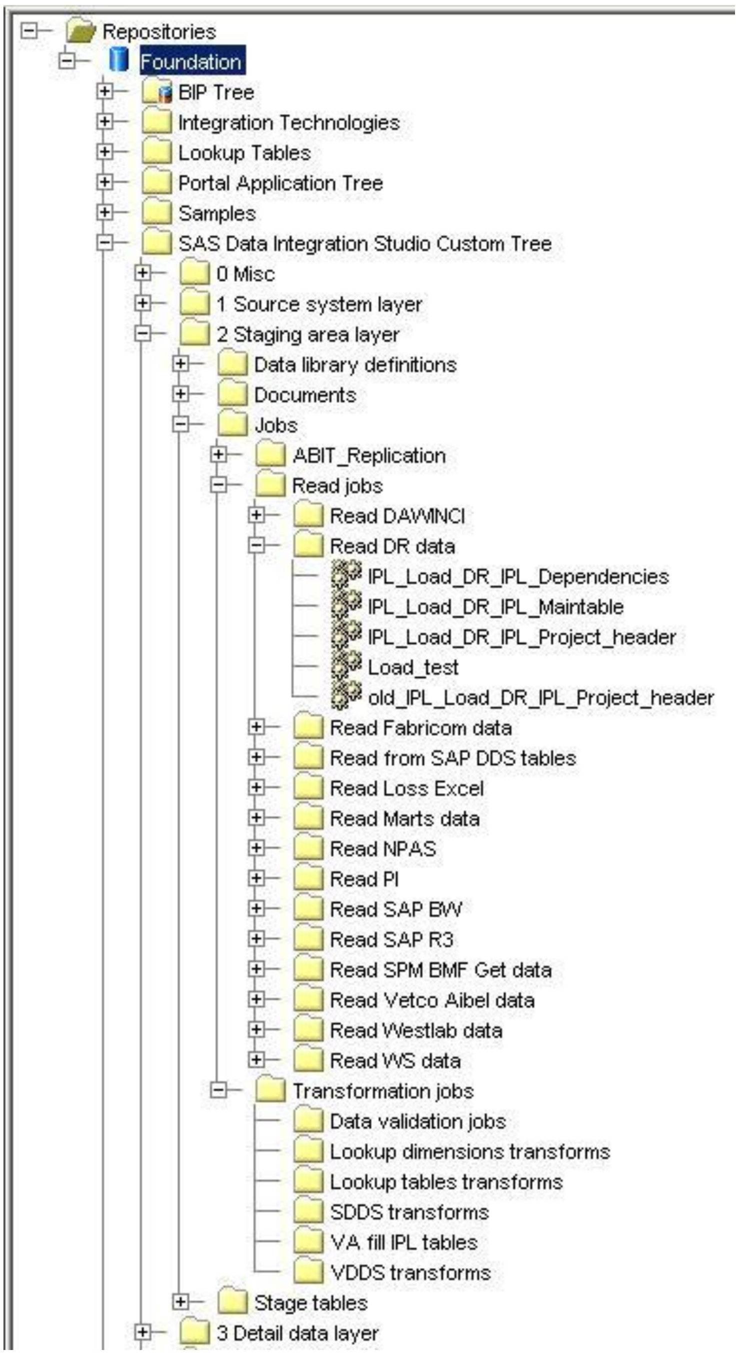


```
Repositories
  Foundation
    BIP Tree
    Integration Technologies
    Lookup Tables
    Portal Application Tree
    Samples
    SAS Data Integration Studio Custom Tree
      0 Misc
      1 Source system layer
        Data library definitions
        Documents
        Source table definitions
      2 Staging area layer
        Data library definitions
        Documents
        Jobs
        Stage tables
      3 Detail data layer
        Data library definitions
        Documents
        Jobs
        Tables and views
      4 Marts and applications layer
        Activity based budgeting
        Barrier
        Constraints
        DaVinci Restricted
        Enterpise Guide views
        Integrated planning
        Logistics
        LossCausation
        Maintenance
        Mods
        Operationalizing BI applications
        Production
        SPM
      5 Lookup tables layer
      6 BI layer
      Ungrouped
      User Transforms
    Shared Data
```

Figure 4-6 – DI Custom Tree; Layer structures

Layer 4 does not immediately seem to have the same folder/branch structure as the rest of the layers. However, as discussed earlier, marts are extracts or snapshots of a data warehouse. These specialized versions of a data warehouse are not each a layer in the structure themselves, but they are detached data stores, each one of them functioning as the DS for that particular business domain. As with layer 3 that represents the SDDS, each of the marts under layer 4 represents individual DS's, meaning that each DS needs their own data library definitions, jobs, documents, and tables and views branches. This is easily shown in Figure 4-9 of the expanded Maintenance mart in layer 4.

The next picture, Figure 4-7, shows the staging area. The staging area is an intermediate storage facility, where most of the E and T jobs are done. Depicted are some jobs extracting data from Drilling (DR) that is supposed to be loaded into the Integrated Planning (IPL) data mart. Beneath the reading jobs – the jobs extracting data from the different source systems – are the transformation jobs.



```
Repositories
  Foundation
    BIP Tree
    Integration Technologies
    Lookup Tables
    Portal Application Tree
    Samples
    SAS Data Integration Studio Custom Tree
      0 Misc
      1 Source system layer
      2 Staging area layer
        Data library definitions
        Documents
        Jobs
          ABIT_Replication
          Read jobs
            Read DAWINCI
            Read DR data
              IPL_Load_DR_IPL_Dependencies
              IPL_Load_DR_IPL_Maintable
              IPL_Load_DR_IPL_Project_header
              Load_test
              old_IPL_Load_DR_IPL_Project_header
            Read Fabricom data
            Read from SAP DDS tables
            Read Loss Excel
            Read Marts data
            Read NPAS
            Read PI
            Read SAP BW
            Read SAP R3
            Read SPM BMF Get data
            Read Vetco Aibel data
            Read Westlab data
            Read WS data
          Transformation jobs
            Data validation jobs
            Lookup dimensions transforms
            Lookup tables transforms
            SDDS transforms
            VA fill IPL tables
            VDDS transforms
        Stage tables
      3 Detail data layer
```

Figure 4-7 – DI Custom Tree; Staging area layer

Apart from the particular jobs layer 3 looks more or less just like layer 2. First when the specific levels directly under the layer level is expanded the distinctions between layer 2 and 3 emerges (Figure 4-8). Instead of heaps of reading and transformation jobs there are more loading jobs, or maintaining jobs – jobs that maintain the data and tables in the SDDS, at layer 3. The image below gives an impression compared to Figure 4-7 above about the similarities and differences between level 2 (above) and 3 (below).

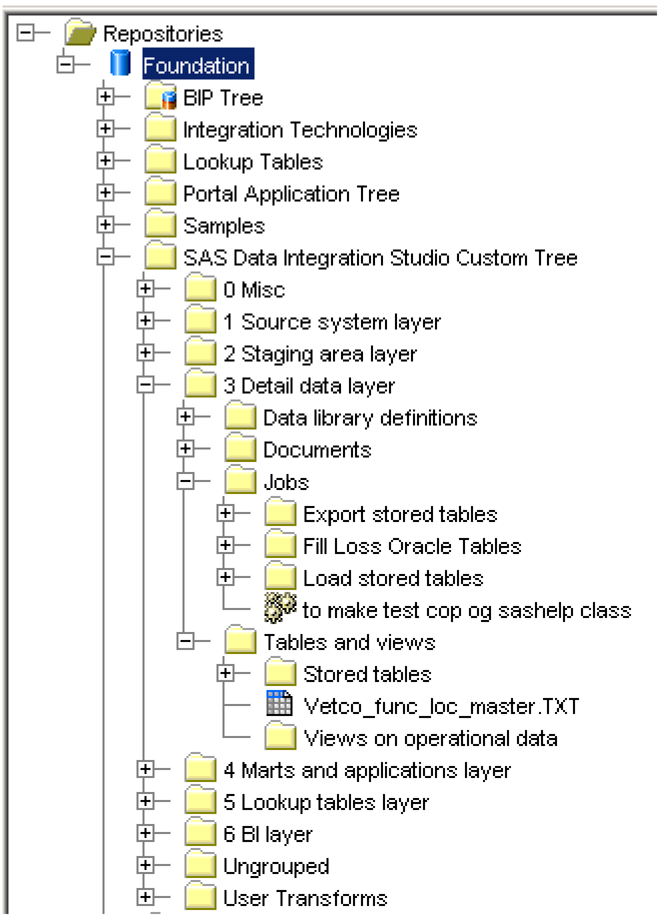

Repositories
Foundation
BIP Tree
Integration Technologies
Lookup Tables
Portal Application Tree
Samples
SAS Data Integration Studio Custom Tree
0 Misc
1 Source system layer
2 Staging area layer
3 Detail data layer
Data library definitions
Documents
Jobs
Export stored tables
Fill Loss Oracle Tables
Load stored tables
to make test cop og sashelp class
Tables and views
Stored tables
Vetco_func_loc_master.TXT
Views on operational data
4 Marts and applications layer
5 Lookup tables layer
6 BI layer
Ungrouped
User Transforms

Figure 4-8 – DI Custom Tree; Detail data layer

“Marts and applications layer” or layer 4 is the last layer this chapter will discuss in detail. The picture below, Figure 4-9, has been cut and put together again to avoid the picture taking too much space, but only jobs have been omitted. The expanded folder under Maintenance -> Jobs called “Output to external parties” contains a job feeding transformed data back into the source system. This is what is illustrated in Figure 4-3.

The job “Vetco_Transfer” seen in Figure 4-9 is one of the jobs making the Knowledge Infrastructure unique compared to most other data warehouse systems. This job takes processed data from level 4 in the KI structure, and feeds it back into the source system. In this case the source system is a contractor for COPNO, and thus an external source from the company’s point of view. However, all source systems are external for the KI, even though the systems themselves may be internal to the company.

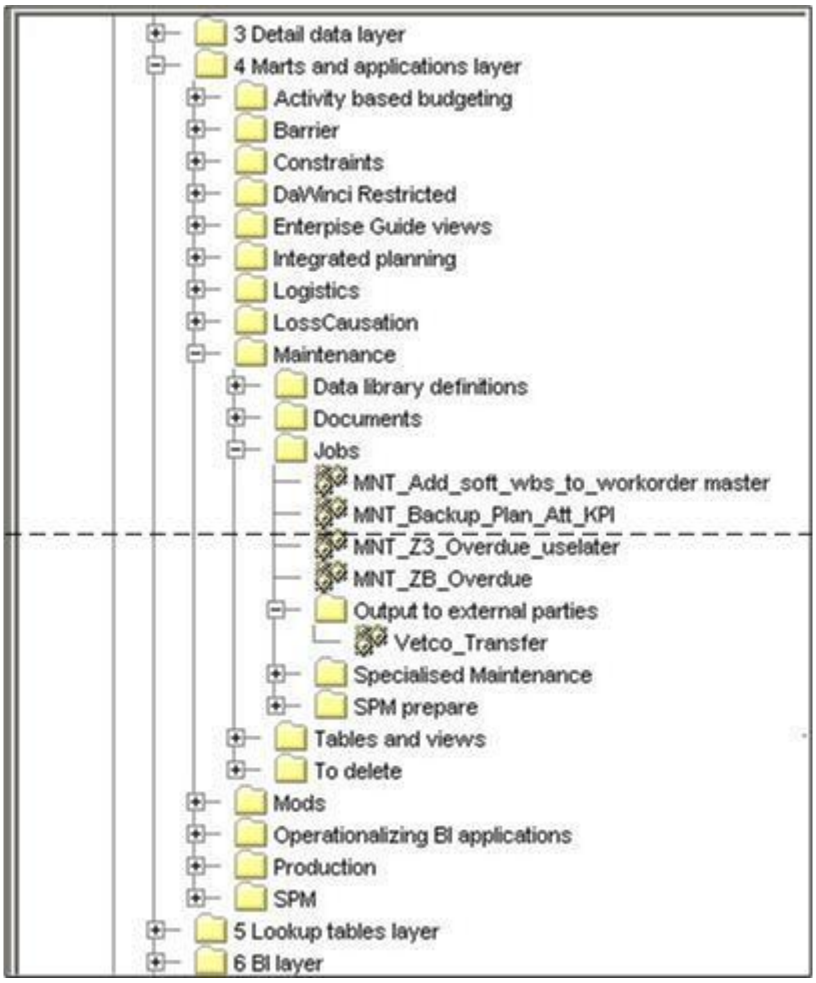


Figure 4-9 – DI Custom Tree; Output from KI to source (feedback)

4.1.6 Strategic Performance Management visible in the Custom Tree

Also seen in the picture above is the branch called “SPM prepare” which is found in some of the marts. SPM prepare is not a separate mart, but in each of the marts where this branch exists, it holds specialized data meant for the SPM application.

SPM (Strategic Performance Management) is a server application to support performance management. It is a sort of database where it is possible to add elements and attributes to the datasets, and also pointers to other elements, key figures and KPIs. Functions such as indicators and alarms can be embedded and SPM enables the user to view data over time. “Extra” functions like these make this an application better suited to support BI decisions than e.g. a mere Excel solution.

As a tool SPM works to give leaders, and other users, a quick overview and compare the present situation to key statistics and effectively communicate this further in the organization. The common user does not see the SPM software, but views the output through a web-based portal that can display scorecards, dashboard diagrams and incidentally all types of contents SPM can show.

SPM requires the data arranged in a special setup. While data marts are meant for information consumers through infomaps/web reports, and/or stored processes and/or self served exploration in e.g. Enterprise Guide (a SAS application supporting their data warehouse system), SPM prepare is exclusively prepared to fill the SPM MySQL database. Contrary to where the system user would be the consumer of a data mart, the information consumer for SPM prepare would be SPM.

4.1.7 Functional view of the KI system

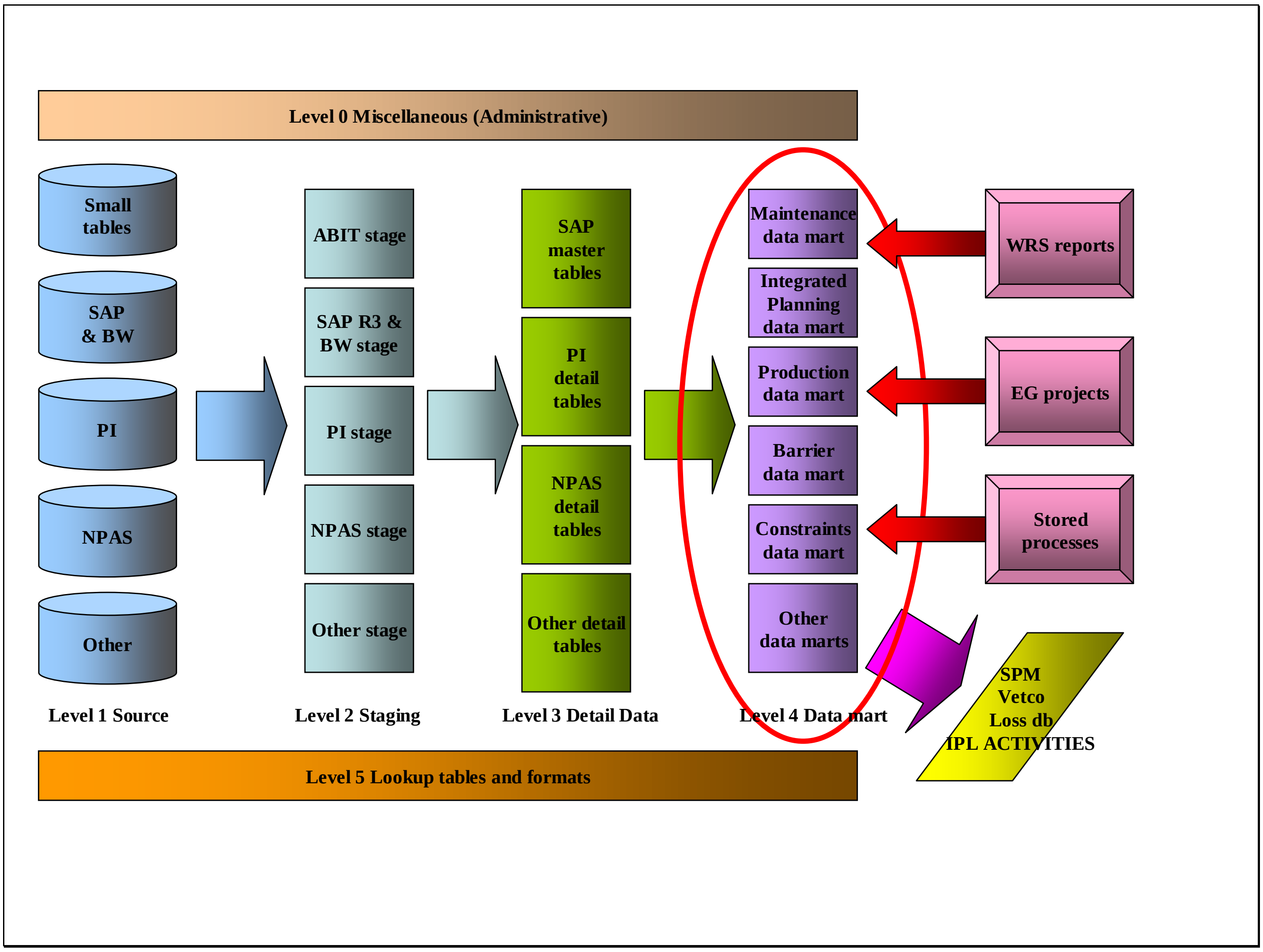


Figure 4-10 – Logical view of the DI Custom Tree Structure

Figure 4-6 and Figure 4-9 suggests that the tree structure is divided in two different ways. Layer 1 through 3 is split into DI objects like the "Data library definitions" - giving the target of the data, where the tables belong and the definitions of libnames. Layer 4 – the mart layer – is split into subdivisions of subjects, where each subject determines what DI objects is to exist in their sub nodes.

Above in Figure 4-10 is a functional illustration of all of the layers, except layer 6, seen in Figure 4-4. Layer 6 is not included in Figure 4-10 but will be discussed subsequently. In Figure 4-10 it is easily illustrated that the layers (levels in this figure) 1 through 4 constitute the components which make out the “main” parts of the data warehouse. Although level 1 – the sources are, as stated before, not really a part of the data warehouse, they are key elements needed for any data warehouse system. As the sources are also seen as a part of the KI they belong as one of the components in this figure.

Level 0 and 5 enclose the rest of the system, or levels. Whereas levels 1 through 4 are the main building blocks of the data warehouse system, level 0 and 5 are rather collections of jobs and data to support the rest of the system. The “Miscellaneous” layer contains administrative “backend” jobs. Backend jobs found at this level are mapping jobs, jobs checking errors and table updates, jobs copying formats from prod to test, jobs loading formats, jobs calculating formulas and comparing tables and jobs, and jobs loading users and user groups for the KI system. Routines to make sure jobs do not run in case the jobs they depend on have failed or did not run are also implemented at level 0. In those cases where jobs either have failed, or not been run or not run properly, causing the data in the corresponding tables not to be updated, routines at level 0 are also responsible to give notice to users that the data is faulty or out-of-date.

“Lookup tables and formats” is a cross layered level containing formats and lookup tables used in the other levels, especially levels 2 through 4 and 6. Lookup tables are usually small tables with few data entries used for transformations – e.g. “codeA” gives “TextA”. Formats are much the same, but may contain more arguments than a lookup table. In practice these can be used to e.g. connect two or more different tables together where the key columns containing the primary keys are not in the same format.

The Detail Data Layer can be regarded as the last safety barrier before layer 4 where the data is finally made available for the end user. Only developers should have access to the data and tables on level 3 and the levels above. Sometimes it may however be necessary or desired in business context for an end user to have access to a flat source table. Tables originating from levels above level 4 will therefore be put in level 6 when this is properly up-and-running.

4.2 Knowledge Infrastructure Architecture and Configuration

Up till now there have been only six layers in the configuration of the KI, namely layers 0 through 5. Layer 6 is the result of several improvements done to avoid certain drawbacks of the system originating in the way the system was set up.

When a table is open or in use that table is locked, meaning that updates or other write-jobs to the table by other users or processes than the one that opened the table will not be possible. In the case that a user forgets to exit the application holding one or more tables open, or at least close the tables, when going home after a finished work day, scheduled jobs depending on the topical tables will fail or not run at all.

Introduction of the new Layer 6 is closely connected with the installation of a new server. More about this follows in the subsequent sections. In the subsequent sections the division into tiers and the names of the tiers differ slightly owing to the fact that Figure 4-11 and Figure 4-12 are taken from a general descriptions of the SAS platform [9] while the others are descriptive of the KI. For the KI, the relevant system for this thesis, please refer to Figure 4-13, Figure 4-14, Figure 4-15 and Table 4-1 regarding the different tiers.

4.2.1 High-level architecture for SPM

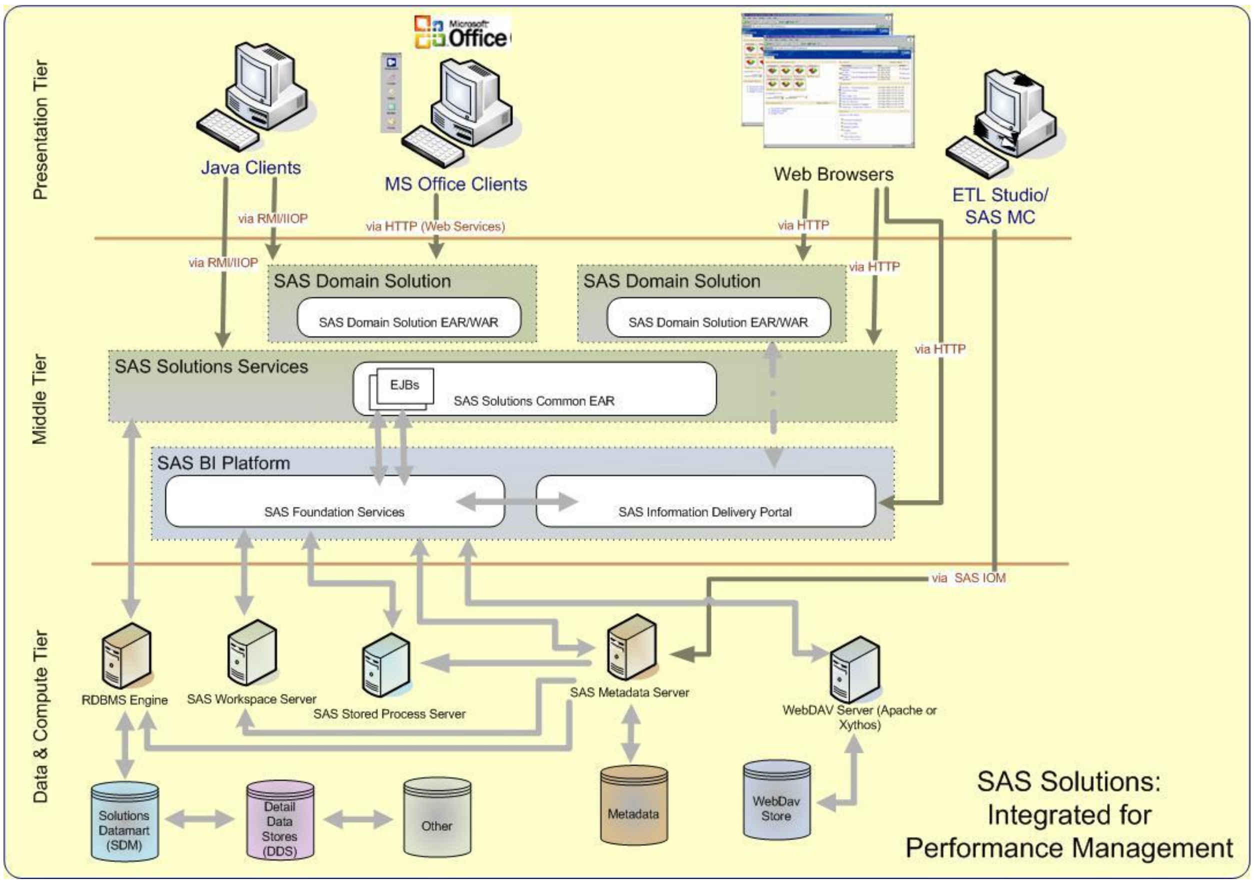


Figure 4-11 – High-level architecture for SPM [9]

The drawing above shows the SPM architecture. The architecture is here divided in tree layers and shows at which layer the different servers are located. An example on how the logical flow is through the architecture can be visualized when a user accesses SPM through a web browser. The client part, or presentation tier (later also Client Tier), (Data Integration Studio, MS Office integration etc.) connects to the metadata server and asks for resources of different types. The metadata server then tells the application where to connect further, for example to a workspace server. If the request comes from the portal, the flow goes through SAS BI Platform and from there to BEA Weblogic if the user has chosen to open the SPM application. The permissions are still maintained in the metadata server.

The SAS BI Platform services acts like an interface between all the software servers at the server tier and BEA Weblogic on the web-tier.

The storage for SPM is the MySQL database. When a user adds a scorecard or key metric, that information is stored in the SPM part or the solution part of MySQL, depending on whether the content is specific for SPM or if it is content that can be shared between SAS Solutions.

The Detailed Data Store (DDS) is a SAS storage, which can act as the interface towards the legacy systems. Every data from the ETL process can go trough this area on its way into MySQL.

BEA Weblogic is the application server that compiles all the JSP-applications. The SPM application is such an application, and the same goes for the portal, Web Report Studio and the other applications. Data Integration Studio is a java application.

The basic servers in use in the architecture are listed in Table 2-1 below:

Tier	**Server**	**Logical name (SAS)**	**Environment**
Data-Tier	SVGAPSAS44	SASAPP	DevTest
Data-Tier	SVGAIXSAS02	ETLAPP	DevTest
Mid-Tier	SVGAPSAS41	SVGAPSAS41	DevTest
Metadata Tier	SVGAPSAS40	SVGAPSAS40	DevTest
Data-Tier	SVGAPSAS45	SASAPP	Prod
Data-Tier	SVGAIXSAS02	ETLAPP	Prod
Mid-Tier	SVGAPSAS43	SVGAPSAS43	Prod
Metadata Tier	SVGAPSAS42	SVGAPSAS42	Prod

Table 4-1 – The different Tiers used in the Architecture

4.2.2 Logical configuration

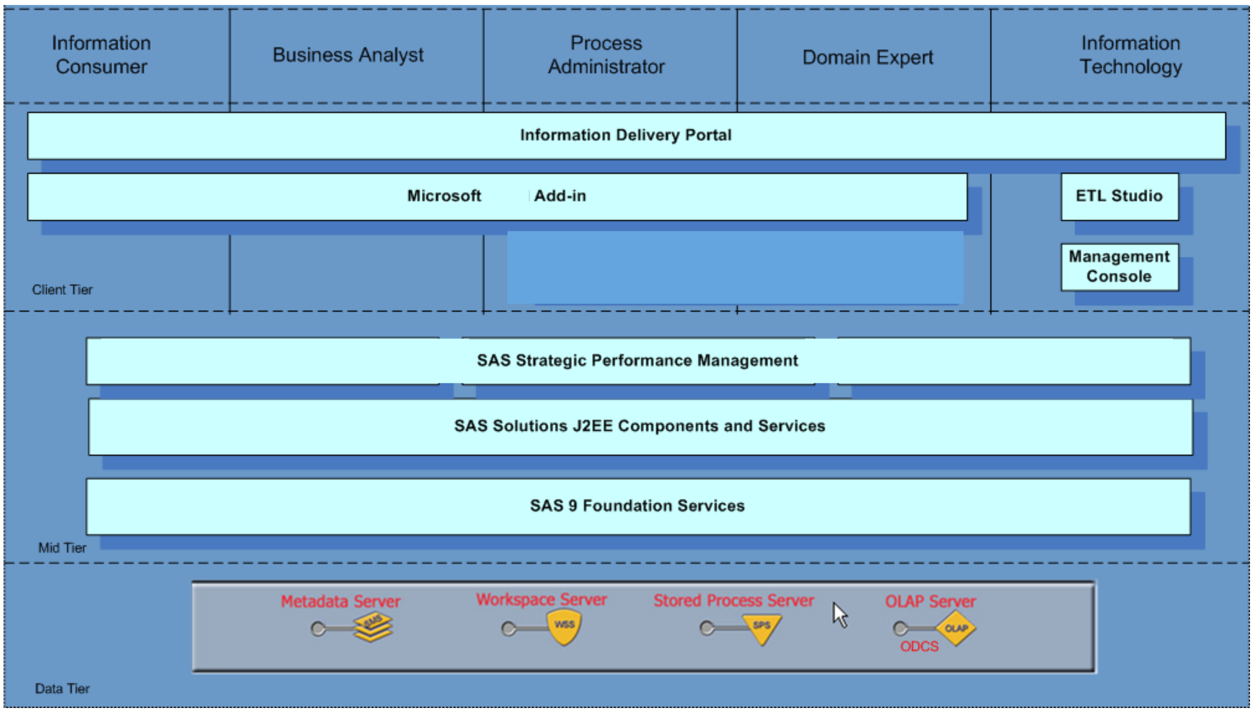


Figure 4-12 – Logical configuration for the SAS9 platform [9]

The logical configuration in Figure 4-12 shows how the different layers interact with each other. Data-Tier is basically just a place where tasks are being done on behalf of other parts of either mid tier or client tier. The mid tier is the Java layer that binds SAS clients and portal together. Client tier is all the activity that is done on the users own pc, such as the portal in MS Internet Explorer or SAS Clients like SAS Data Integration Studio.

4.2.3 Physical Architecture

The physical architecture (Figure 4-13) is where the various items in the logical drawing (Figure 4-12) are placed physically. Currently there is a setup with two environments in the KI; the Dev/Test environment where development and testing of the jobs are done before deployment of ready jobs to the second environment – the Prod, or Production, environment. System setup for the two environments is identical with regards to number and type of computers and services run on them.

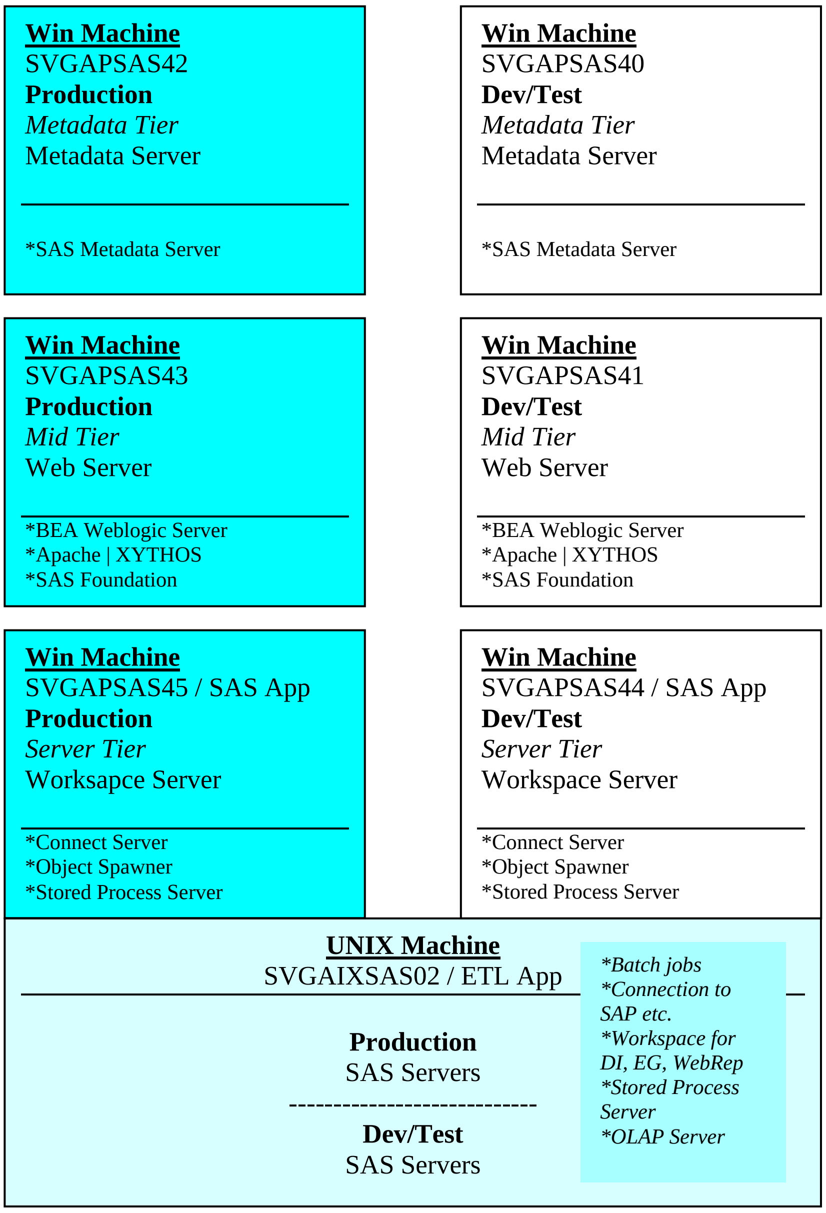


Figure 4-13 – Architecture Diagram of KI

In both Dev/Test and in Prod there are three Windows machines plus a shared UNIX machine. The Windows machines constitute three system tiers in both environments: the server tier, the mid tier and the metadata tier.

SVGAPSAS40 is the logical name on the Windows machine on the metadata tier in the dev/test environment, while SVGAPSAS42 is the name for the corresponding machine in production. These are both dedicated metadata servers on their respective environments, running the SAS metadata server. The metadata server keeps information about all data from scheduling of jobs, job dependencies and stored processes to graphical view of e.g. reports.

The mid tier is the web server running i.a. a BEA Weblogic server, an Apache server with Xythos and SAS Foundation. Logical names for the web servers are SVGAPSAS43 and SVGAPSAS41 in Production and Dev/Test respectively.

Xythos is a web-based tool that will allow users to upload files. After uploading a file, Xythos can be used to store, share, or conveniently access that file. The flexibility of Xythos allows a user to control file sharing: a file can be private, shared with one user (whether or not they have a Xythos account), a group of users, or published to the web. Shared files can be read only, or writable for collaborative work.

SVGAPSAS44/45 (in DevTest/Prod) are the workspace servers. These servers are called SAS APP servers, used to run processes in need of special windows resources and to extract SQL data needed for jobs run on the UNIX system (see below). Also on SAS APP a connect server, an object spawner and a stored process server is running. However, no schedules jobs are run on SAS APP, these are all run on the ETL APP server.

At the bottom of this design resides a UNIX machine, SVGAIXSAS02, shared between both the Production and DevTest environments. This server, the ETL APP runs SAS servers for both environments. Examples of processes run on SVGAIXSAS02 are batch jobs, connection to source systems like SAP, stored process server, OLAP server and workspace for DI Studio, Enterprise Guide and Web Report Studio.

Another way to illustrate the physical architecture, including the client tier with client applications, is illustrated in Figure 4-14. The server tier is here divided in metadata server, workspace server, stored process server and MySQL database that all can reside on different machines. In Addition, there is the workspace-, batch and stored process server on the SASAIXSAS02 UNIX machine.

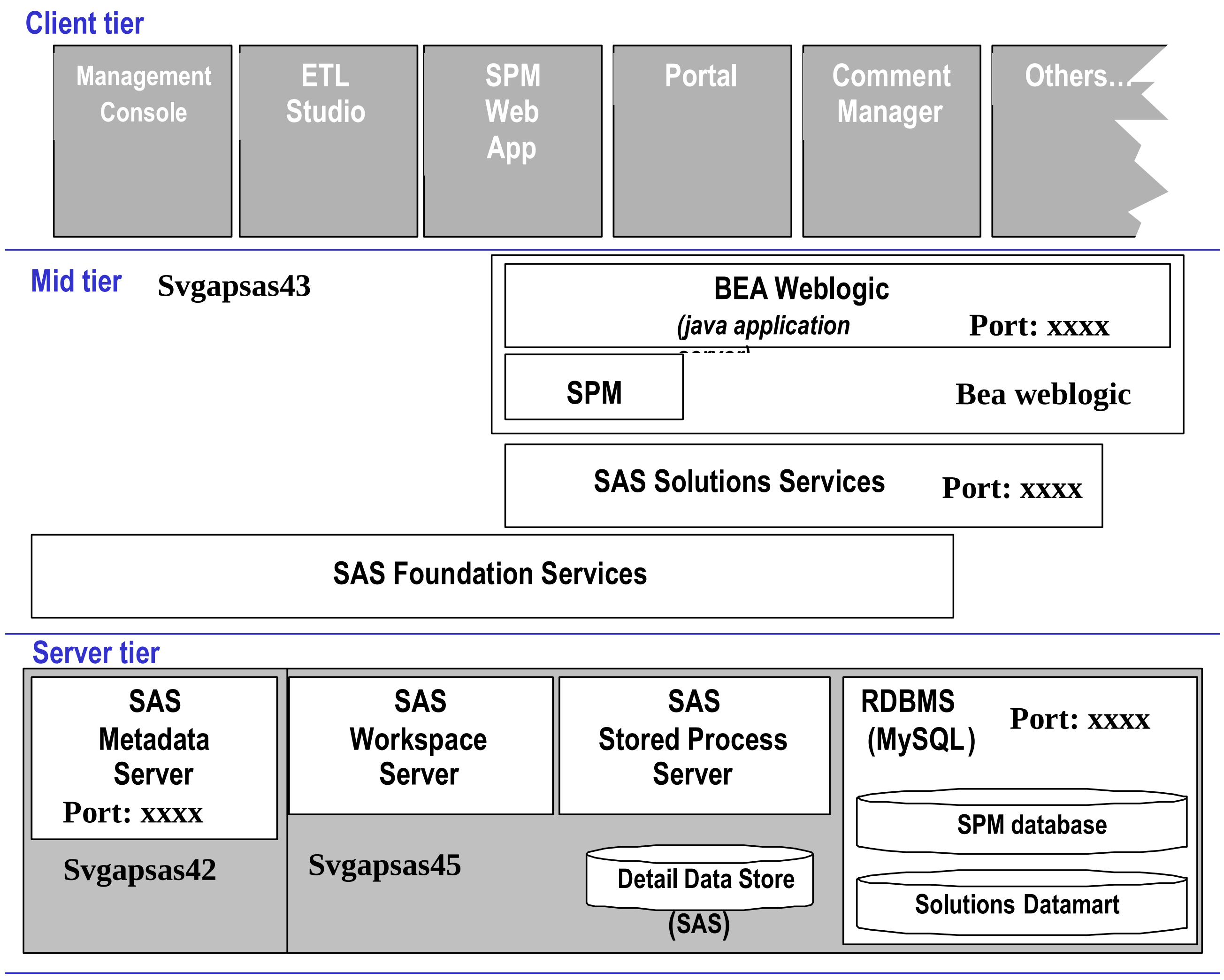


Figure 4-14 – Physical architecture with server names from Production environment [10]

To avoid confusion, only server names from the Production environment have been included in this illustration. However, except from the server names, the DevTest environment also look identical to the illustration in Figure 4-14.

New UNIX server

As mentioned earlier, Layer 6 is the result of the addition of a new UNIX server, see Figure 4-15. This server is called SVGAIXSAS03 or the BI APP server. Both UNIX machines will run on AIX – IBM's UNIX operating system. However the new UNIX machine will only be used for the Production environment, not the DevTest environment as shown in Figure 4-15. After complete installation of BI APP, data mart tables will be copied from SVGAIXSAS02 to the new 03 server and BI APP will take over BI applications and servers serving end users or clients from SVGAIXSAS02 in Prod. The services running on BI APP will then be workspace for Web Report Studio, Enterprise Guide and DI Studio, stored process server and OLAP server. Services needed for support of users are no longer needed on SVGAIXSAS02 and will therefore not run on this server in the production environment. Remaining services on Prod for SVGAIXSAS02 will be batch jobs, connection to source systems and workspace for DI Studio. The new design is illustrated in the diagram below (Figure 4-15):

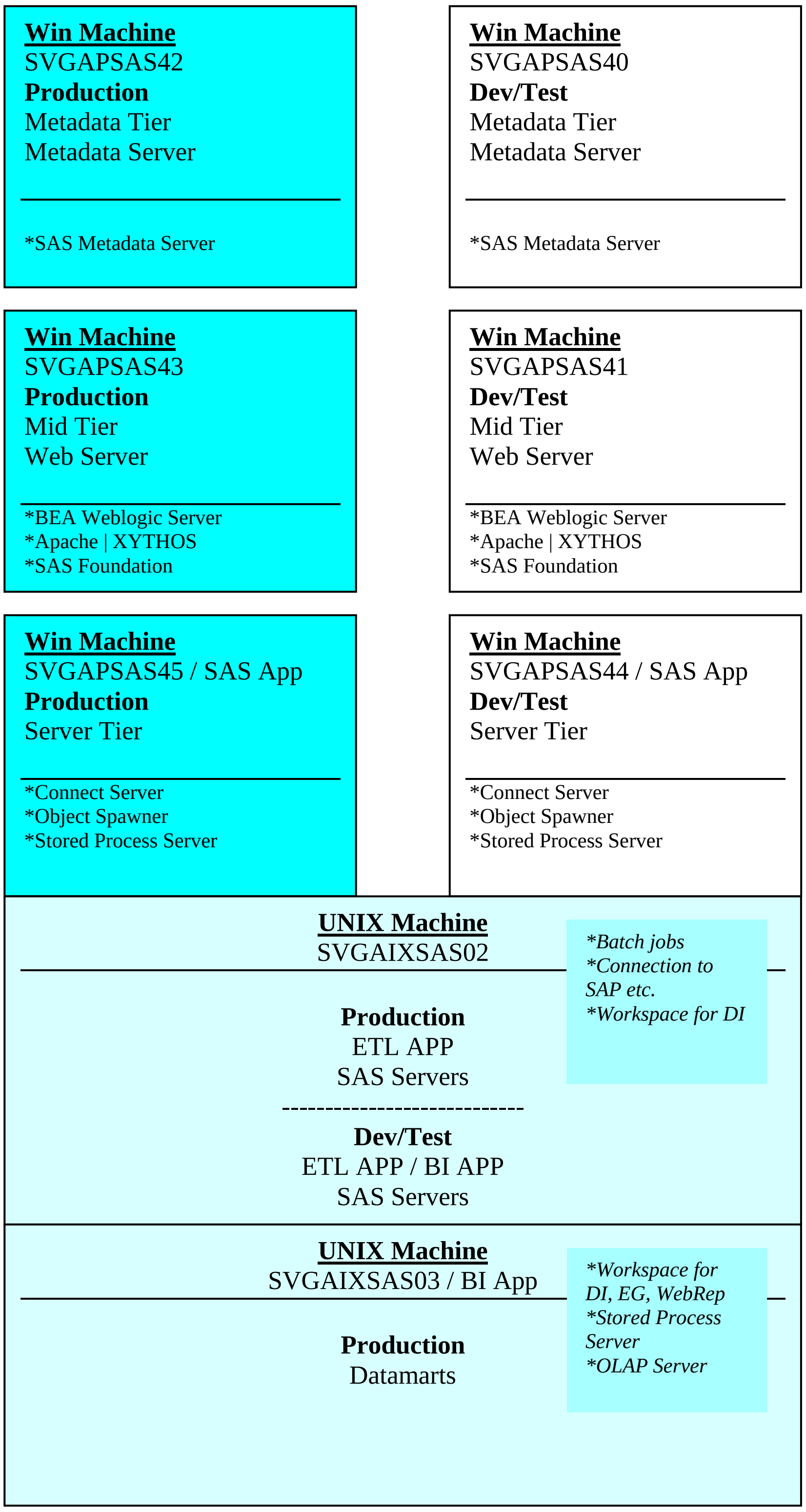


Figure 4-15 – Architecture Diagram with new UNIX server

4.2.4 Services

Windows NT services are essential to make the system work. Listed below in Table 4-2 is a table of the running services the system relies on.

Service name	**Service label**	**Machine**	**Start order**	**Description**
SAS Lev1 MS - SASSolutio	SAS Lev1 MS – SASSolutio	Svgapsas40	1	This service starts the SAS Metadata server. The metadata is the basis for all security lookup, library lookup and user registration. Not depending on other services.
MySQL	MySQL	Svgapsas44	2	The MySQL service starts the relational database MySQL. The service itself is not depending on other services, but the metadata server may utilize libraries in MySQL
SAS Lev1 CS - SASSolutio	SAS Lev1 CS – SASSolutio	Svgapsas44	3	The connect service is utilized for setting up connect sessions between SAS sessions on different machines. Depending on the metadata server.
SAS Lev1 OB - SASSolutio	SAS Lev1 OB - SASSolutio	Svgapsas44	4	The object spawner process is used by several clients, like Data Integration Studio and Enterprise Guide. Depending on the metadata server.
Xythos	Xythos	Svgapsas41	5	Service that runs the webDAV server. The DAV is "Web-based Distributed Authoring and Versioning". It is a set of extensions to the HTTP protocol which allows users to collaboratively edit and manage files on remote web servers. Here DAV is implemented using a Xythos DAV server, with its storage in Oracle. The oracle instance is on svgsundb03.conocophillips.net:1521:svgsas01 The server is not depending on other services.
beasvc SASSolutions_Admin	beasvc SASSolutions_Admin	Svgapsas41	6	The BEA Weblogic administration service. This is used by Weblogic for administration tasks, like deploying web applications. Not depending on any service.
SASFoundationServices *(not as service in test)*	SAS Foundation Services	Svgapsas41	7	The Foundation services (SAS Remote services). Used for building a interface between SAS solutions (metadataserver etc) and weblogic. Depending on the metadataserver.
beasvc SASSolutions_ManagedServer *(not as service in test)*	beasvc SASSolutions_ManagedServer	Svgapsas41	8	The BEA Weblogic ManagedServer service. This is the Weblogic application server for the web applications. Depending on the BEA Weblogic administration service and SASFoundationServices

Table 4-2 – The services that SPM relies on, with references to DevTest server

Additionally some of the services have dependencies to a service not running on the same machine. SAS Foundation Services on Svgapsas41 is an example of a service dependent of another service. The service SAS Foundation Services relies upon is the SAS Metadataserver located on Svgapsas40. In order to get these services to start in correct sequence, the servers need to start in a specific succession. The list below shows the startup order for DevTest and Prod.

DevTest

1. SVGAPSAS40
2. SVGAPSAS44
3. SVGAPSAS41

Production

1. SVGAPSAS42
2. SVGAPSAS45
3. SVGAPSAS43

On the Mid Tier servers there is a startup script that starts the local services in correct order. It contains pauses (sleep) to allow the services to startup properly before the next service starts. The script is scheduled to start at machine startup using the MS Scheduled Task functionality.

Some of the processes are started using NT resource Kit tools, such as Service Control (SC) letting the server start a process on another server.

5 Discussion

5.1 Introduction

This chapter will discuss solutions, mootings and conclusions from earlier in the thesis. The chapter represents the thoughts and meanings around the solutions and results found for a data warehouse system designed to support Business Intelligence. In relation to the Knowledge Infrastructure at ConocoPhillips Norge it was important to expose problems relating to the system and the design and development of the system.

Relevant problems were divided into three sections, chapters 5.2, 5.3 and 5.5 where one deals with general problems from a project point of view. The next section discusses an issue put on the agenda from the Data to Decisions team in the course of the work with this thesis, which was separation of batch and users in the production environment.

To solve the most precarious issues, new additional hardware may be necessary to accommodate the best solutions. During the last period of the work with this thesis ConocoPhillips Norge indeed obtained a new UNIX machine to comply with one of the current problems addressed in this thesis.

One of the methods to arrive at possible solutions to the current question concerning how many environments are advantageous to implement in the Knowledge Infrastructure, is to consider best practice from other systems. During this project, the author has conducted interviews with SAP practitioners to obtain an understanding of how other systems are designed with regards to this topic. Under the section “5.4 Best Practice from SAP” the thesis takes a look at how the problem is addressed in the SAP system.

Finally, based on among other factors the best practice from SAP, the discussion presents the current environment setup of the Knowledge Infrastructure. In this part the thesis will discuss the properties of those solutions, to conclude with possible ways to how the current issues may be solved.

5.2 General Issues

Starting a project of implementing a data warehouse system to support integrated operations in the enterprise is a major task to undertake for any company. A great deal of effort has to be put into planning, and the planning stage is arguably the most important stage of the process.

One of the first steps in developing a data warehouse that is supposed to be utilized by every department in the company, is to get people to define what they really want. In order to succeed getting consistency of the data in the final data warehouse system, one must make sure that the data from the different departments are correct. To make good metadata for the source data resulting in accurate use and automation of the information in the data warehouse, one needs well planned and punctilious specifications of the processes in the departments.

The process of defining the metadata in the Knowledge Infrastructure is mostly finished, but even so while the implementation of these metadata was in progress, one found erroneous information and calculation descriptions in some of the documentations from the departments. This may still occur, and it is important to always have reliable methods of checking for errors.

Even though the Knowledge Infrastructure implementation comes to an end, new needs coming from the departments arise and show how important it is in a project like the Knowledge Infrastructure to have good routines to deal with these kinds of issues.

Planning for the Knowledge Infrastructure one of the debates for the Data to Decisions team was to demarcate the scope of the system. What information the data warehouse should implement was, and still is, a question open for debate. With over 200 potential source systems it is important to filter the essential data to end up with a useful and practical to use data warehouse system. Additionally an issue arises when the system implements information not owned by any of the specific departments in the company. The Knowledge Infrastructure’s task is to support integrated operations, but who is the real owner of integrated data? In the case of ConocoPhillips Norge a new department, the Onshore Operations Centre (OOC), was established to own information not owned by any other existing departments.

An issue many new systems, at least pioneering systems like the Knowledge Infrastructure, run into is immature software. Software bugs are a constant risk to any system and may yield incorrect data, or even put whole systems out of operation. For jobs dependent on such systems, downtime resulted by bugs may cause deferment of operations and loss of income. Expenditures induced by such incidents are hard to prevent, but are nevertheless important to try to avoid. This is the responsibility of the software vendor, and difficult if not impossible for anyone else to deal with. It is therefore imperative that bug-fixes and new versions dealing with any flaw in the software are released on a reliable schedule.

In the case of the Knowledge Infrastructure it is also a problem that the source systems are not developed for this way of thinking. Chapters 2.2, 2.4, 2.4.1 and 2.4.2 discuss the fact that most source systems in a data warehouse are transactional systems. Likewise the source systems for the Knowledge Infrastructure are developed for cost transactions. This becomes problematic with regard to the Knowledge Infrastructure where all data shall look like they originate from the same system. Making this happen requires thorough planning and good software for the data warehouse system. Software can transform data, but the software vendor must be aware that this is a requisite in the system and be willing to comply with these demands. Although the data is transformed and everything looks as it should, the underlying processes of transforming the data may affect the system in ways of speed. This is a property worth taking notice of in the early stages of planning and designing of the overall system.

5.3 Separating marts from core data warehouse

In chapter 4.1.2 the thesis discussed the installation of a new UNIX server to separate online data marts from the core data warehouse system. Introducing this new server benefits the system in several ways. Most likely this will deal with some of the most negative features of the system the way it is today.

As mentioned in chapter 4.2, tables being held open by a user or an application will prevent dependent jobs from running. If in turn any other jobs are dependent on a job prevented from running because of an open table, neither of those jobs will run. This represents a major deficiency in the system. In itself the whole purpose of a data warehouse system is to support business intelligence by delivering information to the organization in form of end users. Whereas operational systems exist to provide for the operating needs of the organization, informational systems exist to provide for the intelligence needs of the organization.

In a system like the Knowledge Infrastructure with about 3.400 registered users, it would be near unacceptable to have a design dependent on all the users always to release any open tables in order to function properly. Especially when fault tolerance towards open tables needed by scheduled jobs are zero.

Additionally a user group of the mentioned size represents another issue. The amount of transactions this huge user group can result in, one of the bottlenecks in the Knowledge Infrastructure system is IO on disk. There is a limited amount, however large it may be, to how much IO one can have during a day. As a matter of fact the batch jobs are not very IO requiring. Most of the IO load occur during the day when users are online and putting strain on the system. A result of this is that critical scheduled jobs should not run during the day while most users are utilizing the system. While users have higher priority in the system than batch jobs, users will steal resources from any batch jobs and hinder the jobs from running. On a system deploying four CPUs as on the SVGAIXSAS02[4], three active users have thread priority on three of the CPUs leaving only one CPU open for lower priority jobs like batch jobs. With these three users active 75% of the server resources are engaged, and only 25% of the resources remain idle or free for potential batch jobs. If enough users are employing the system the result becomes blocked jobs.

Immediately this may not seem like a big problem as most of the scheduled batch jobs are running during nightly hours. In a multinational company like ConocoPhillips it may however be a greater issue than first considered. Being an international corporation some of the users of the data warehouse system are stationed in other countries, for instance in the USA where normal work hours coincide with night time here in Norway which is exactly when the batch jobs are running.

Furthermore there is a large user group for the Knowledge Infrastructure residing offshore on oil drilling platforms. Offshore, the situation is a bit different than onshore. People offshore work on shifts during both day and night all days a week, leading to users of the Knowledge Infrastructure being at work on a 24/7 period.

In an energy company like ConocoPhillips Norge where most of the work revolves around offshore activity, a large percentage of the important data is produced offshore. The data produced offshore constitutes an equal large percentage of the data going into the Knowledge Infrastructure. All offshore activities need to be planned in order to have the right equipment and necessary personnel available to carry out the pending jobs. The system used for this type of planning is the SAP system which in turn is the greatest source system in the Knowledge Infrastructure. Most of the offshore work is planned (at least) one week in advance. On Saturdays lots of data is input into the SAP system and Sunday evenings and Monday mornings are used to decide what work is to be done the rest of the week.

This offshore schedule produces lots of new data in the source systems for the Knowledge Infrastructure on Saturdays, leading to huge amounts of new data going into the Knowledge Infrastructure on Sundays. The most critical time for the system is thus on Sundays when more data than usual are updated or fed into the system. It is therefore imperative that the servers are able to run the batch jobs and do the necessary updates on these days.

[4] See Appendix A

Understandably this is not a desired situation in a system where online users steal resources from scheduled batch jobs, and may even prevent such jobs from running. In the case where enough users are online during the time scheduled jobs run, those jobs can accordingly be prevented from running in two or even three ways. The first way is the case where the users are "stealing" all the resources from the jobs, leaving no resources to run batch jobs. Secondly the users may prevent scheduled jobs from running by keeping essential tables open. Thirdly, in the event of the users consuming all available IO from disk, jobs may not be able to read from or write the required data to disk, causing the jobs to fail.

To illustrate the time it takes for the scheduled batch jobs to run consider Table 5-1.

Scheduled Jobs		
Start Time	*End Time*	*Job Type*
20:00	24:00	Extracting data from sources.
00:10	02:30	Generation of master tables. *Used as input to data mart tables*
02:30	07:00	Jobs generating data for users.
07:00	As long as it takes. Normally to approx. 11:00	Jobs to update data. *Not heavy jobs.*

Table 5-1 - Times for scheduled batch jobs

The heavy jobs requiring full access to the tables being used by the jobs start at around midnight and run until normal business hours start. These are the most critical jobs that will fail if any user or application keep the tables required by the jobs open.

As discussed in chapter 4.2 the introduction of the new SVGAIXSAS03 UNIX server resulted in the new Layer 6 shown in Figure 4-4. This layer is called the "BI layer" and the server is appropriately called the "BI APP" server. Business intelligence is the concept of data turning to information that is delivered to the appropriate people in the corporation. These persons are often leaders or people in charge of making decisions and need proper information to make the right decisions. In a data warehouse system like the Knowledge Infrastructure, this information is collected from the data marts.

Put differently, the information business users of the Knowledge Infrastructure need is the information present in the data marts. Separating the data marts for use by end users by means of physically duplicating them into a new machine serves several beneficial purposes. As discussed, batch jobs will no longer be prevented by running neither by users keeping a lock on a required table nor by stealing the CPU resources, because the users are working on a different server with separate data sets. Also, seeing that the online users are the main cause of heavy IO load on the system, the IO strain from users will be limited to this new server. The result being that the batch jobs, which are not very IO consuming, will most probably not exhaust this system resource on the server where the jobs are run.

By limiting the access of the users of the Knowledge Infrastructure to the data marts residing on a server separate from the “operational” parts of the data warehouse system, one also gain some possible security benefits. One can more easily prevent unknowing users from tampering with important data, or executing jobs resulting in damaged or incorrect data in for instance the SDDS or the staging area.

In addition one can enforce the right use of the system by making users only to use data in the data marts. Data not present in the appropriate data mart should by business rules be extracted from the corresponding source system or the SDDS to the correct data mart. The situation being as it is today it is far too easy to take the shortcut and bypass the data marts to fetch the data directly from the SDDS or source. This is an unwanted situation an may cause inconsistencies between data used in analysis of data and the data present in the data marts or even the data warehouse (if the data is collected directly from the source system).

Last but not least there is even one more advantage to be gained from separating the batch environment and the user environment. Having two servers with duplicates of the data marts one has the chance of disaster recovery. In case one of the servers breaks down the other one is still running. Both servers have the same setup with all the necessary services installed, though only the services required for normal daily work are run. Running services differ from one server to the other (see Figure 4-15, Table 4-2 and chapter 4.2.4), but in case one of the servers is having downtime the other server can take over. All one has to do is to start the inactive services and set up an alias from the disabled server to the running one.

5.4 Best Practice from SAP

Currently the Data to Decisions project is in a phase where it is putting focus on the number of environments implemented in the Knowledge Infrastructure. At the present moment there are two separate environments in the Knowledge Infrastructure, the Production environment and the DevTest environment. Whether the solution at hand is the optimal scheme for the Knowledge Infrastructure is one of the main issues to be evaluated in this thesis.

In the process of analyzing the needs and desires, compared to what would be a good solution with regard to the environments issue, in the Knowledge Infrastructure, one line of action is to take a look at how similar problems are solved in corresponding systems. Seeing that most data warehouse systems differ greatly from one another, it is difficult to find an external system and merely adopt the solution in the external system and apply that concept to the Knowledge Infrastructure.

A relevant system to look into in this context is the SAP system. SAP is one of the largest business application and Enterprise Resource Planning (ERP) solutions on the market today. As seen in Figure 4-5 SAP R/3 and SAP BW are two of the main source systems in the Knowledge Infrastructure. The name SAP R/3 gives a clue to its functionality. The “R” stands for real-time data processing and the number 3 relates to a 3-tier architecture: Database, application server and client (SAP GUI). BW in SAP BW denotes SAP Business Information Warehouse.

As discussed in chapter 2.2, source systems for data warehouses are often transactional systems and SAP is no exception. Though SAP R/3 is an operational system, as opposed to the Knowledge Infrastructure which is an analytical system, SAP BW is a data warehousing solution. In any case the SAP system solution has an interesting environment setup worth investigating.

Principally the SAP system deployed at ConocoPhillips utilizes three environments: Development, test and production. However the development- and test environments are mirrored one or up to two times. In addition there are two sandbox environments and an environment used for educational purposes. To illustrate this regard the illustration in Figure 5-1.

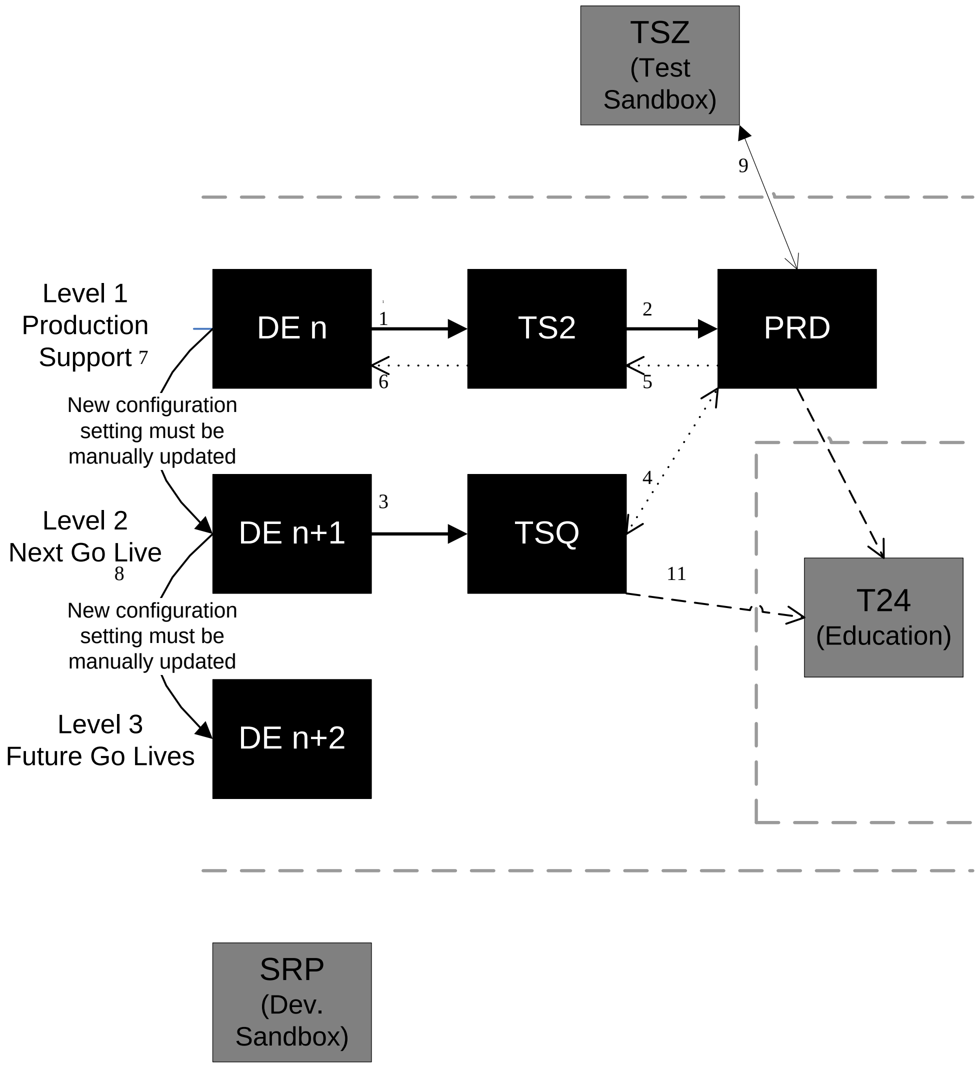


Figure 5-1 – SAP environments; configuration with rotating DEV/TST/PRD environments

[1] Deployment of jobs from development to test environment
[2] Deployment of successfully tested jobs from test to production
[3] Deployment of large jobs or new versions to level 2 development to level 2 test environment
[4] PRD is mirrored to TSQ to keep these environments alike.
When testing of new versions and big jobs are complete and the FIT cycle is at an end, the TSQ environment is migrated to PRD.
[5] After an upgrade from a FIT cycle, the PRD environment is copied to TS2. This way the test environment at level 1 is always equal to the production environment.
[6] Necessary upgrades and new jobs and configurations are copied from TS2 to DE n after a FIT upgrade.
[7 / 8] If any configuration settings are changed in the development environment at level 1 these must be manually updated to the development environments at level 2 and 3.
[9] Necessary objects like jobs are copied directly from production environment to the sandbox environment when needed.
Before an upgrade resulting in downtime for PRD, like a FIT upgrade, the production environment is copied to this sandbox environment.
[10 / 11] When needed, things are copied into the T24 environment from any of the other environments. Typically from PRD, TS2 and/or TSQ.

Level 1 in Figure 5-1 represents the main level of environments in this system. Normal chain of development takes place along this level. This means that ordinary jobs are developed in the "DE n" environment before it is deployed to the "TS2" environment for testing. If the job works according to plan, it is moved to the production environment "PRD" where it is put in production.

In order to explain the three levels it is necessary to introduce a new term: "FIT cycle". FIT stands for "Functional Integration Testing" and is the process of developing and testing new versions and larger projects. This is done at level 2 where "DE n+1" is the development environment and "TSQ" is the test environment. The FIT has a cycle of about half a year with deployment of the new versions together with any big jobs typically every December and April.

The "n" in "DE n(+x)" denotes the version of this development environment. After each FIT cycle DE n+1 becomes the current development environment with the correct settings and program versions, and is moved from level 2 to level 1. In the cases where there exists a development environment in level 3, this is then moved to level 2. If there is no current level 3 after a FIT cycle upgrade, the development environment at level 2 is "emptied" and ready to be used as development environment for the next FIT cycle.

At present ConocoPhillips has reached the 24th version of the development environment, meaning that the FIT cycle has run 24 times (at least since this versioning system was commenced). The correct notation for the current development environment at first level is therefore DE24 (n=24). At level 2 the development environment is consequently called DE25 (n+1), and any existing level 3 environment would be called DE26 (n+2). Any further reference to these environments will however continue to be denoted as in Figure 5-1. It should also be mentioned that all different environments in this SAP system run on separate servers.

Indicated in Figure 5-1, level 2 is called "Next Go Live" and the third level is labeled "Future Go Lives". The second level is used to develop big jobs, jobs too risky or uncertain to start developing at the Production Support level, and to develop and test new versions of applications and software. The label for the third level then suggests that this level is for use in case development of next versions of applications has already started before deployment of the versions undergoing development and/or testing at level 2.

Workflow for the FIT cycles follows arrows 3 and 4 in Figure 5-1. Like in most systems development takes place in a development environment, and development of new versions of software in this SAP system is no exception. When the new versions are ready for testing they are deployed from “DE n+1” to “TSQ” where they are tested. “TSQ” is an exact copy of the production environment “PRD” so that anything successfully tested in “TSQ” is going to work successfully in “PRD”. When testing of the new versions is successful and one of the two FIT upgrades during a year is at hand, the “TSQ” environment is migrated to “PRD”. However, before this operation is carried out, the production environment is copied to the sandbox environment called “TSZ”. While the production environment is down, typically around 12 hours during a FIT upgrade, “TSZ” functions as the production environment for end users. This practice serves two purposes: Users in need of the system in order to carry out their functions do not have to wait until the FIT upgrade is done and the “PRD” environment is up-and-running again. Secondly, “TSZ” works as a backup system that has all the data and settings belonging to the production environment in case anything goes wrong during the upgrade and “PRD” has a breakdown. Having backup systems like this can be essential to a company depending on operational systems to stay in operation. Disaster recovery may not be possible to perform without this kind of essential backup.

Typical for a FIT upgrade is that the “TSZ” has two days old data. Although this may sound like it is outdated data, it may be enough to perform most tasks where one requires the system. Especially if one keeps in mind that the FIT upgrades typically take place in December and April, and is probably scheduled during the holidays while there may not be planned as many jobs, e.g. offshore maintenance, as during normal days. Also, as discussed in chapter 5.3, most offshore work is planned on Sunday evenings through Monday mornings a week in advance. If the FIT upgrade is performed in the beginning of the week, after the offshore planning is done, it will most likely not adversely affect the offshore work schedule too much.

Lastly, if the “TSZ” sandbox environment is kept fairly up to date, it could also be used as a backup environment for the production environment in case of any unexpected downtime.

It is also worth noticing that the FIT cycle can be sped up in case of high demand. This occurred i.a. at the merger of Conoco and Phillips where the FIT was sped up to a cycle of every three months.

After a FIT upgrade and when the “PRD” is up and running again, the production environment is as indicated copied to “TS2”, “TSQ” and “TSZ”. Copying from “PRD” does however not lead to downtime for this environment. Between FIT upgrades the “TSZ” sandbox environment is used as a test environment for end users that normally do not have access to the development or test environments. Allowing for only authorized users to have access to the production line, or level 1, improves security and reduces the risk of ignorant users in any way doing devastating damage to these environments. At the same time the sandbox environment lets unskilled personnel run tests, make jobs or try things they otherwise would not have the possibility to do.

The two last environments not yet described, are the “T24” and the “SRP”. These will not be covered in great depth as they are of lesser importance, but they are nevertheless worth mentioning. “T24” is only used for educational purposes. The argument for having a purely educational environment becomes clear when reflecting on teaching. When training new personnel in the use of SAP, as with teaching in most other connections, one accumulates enough students to start a class. Teaching a system with different levels of security to trainees with different access levels it would be near impossible to use the live system, or in any way finding relevant cases in the system to be used as good problems or assignments in the lessons. Yet, if one were to use the live system for this purpose, the chance of these cases to be obsolete by the next time one starts instructing a class in the use of the system would be great. A Work Order in SAP may for instance in production become out of date and would perhaps not be possible to open in the course, particularly if the course is repeated after half a year or so. The outcome of this would be that the instructor would have to find new problems or relevant examples in the system, make new notes and new teaching material. In other words, having a separate environment for training people in the use of the system saves a great deal of effort. It also makes it possible to utilize good problems over and over even though these may have been removed from the operational system.

“SRP” is merely a sandbox environment for development. Possible uses for this environment may be to assist a programmer who wants to try something completely new without knowing the impact it will have on the system. Jobs not yet decided whether or not one needs or wants in production may also be started here. One advantage of this is that the developer does not have to account for this job being in the production line, nor does he have to carefully document the job before it is clear that the job really is needed in production. Neither does the developer have to put much work into correctly removing the job from the environment if the job is turned down as this is only a sandbox environment. Another advantage of having a development sandbox environment is for new and perhaps somewhat unskilled developers to have a place to “play around” without affecting the production line. The “SRP” environment is updated at will from any of the other environments that may be of relevance.

Advantages of having separate environments for development, test and production are several. One is that as long as the test environment is separate from and up to date towards the production environment, the need for data in the development environment is limited to the data necessary to make the jobs at hand. By reducing the data in the development environment to the required data only, one achieves a much less heavy system.

Versus security, separate Dev/Test/Prod environments make it easier to apply access restrictions - increasing the overall security of the system. This can be a desirable feature, especially on large systems where one may want to e.g. use lesser skilled manpower to carry out testing while the skilled and perhaps also better paid people concentrate on development, thus possibly saving both time and money for the company. In any case it may give more opportunities towards governing security rules in the system. At the same time, the more complicated a system is the more it renders the system difficult and time consuming to administer.

Development-wise, having separate environments, one also avoids downtime in development whilst the test environment is being updated from production. Test environments need to be up to date at all times to serve as an adequate test area for jobs supposed to be deployed to production. Assuming development and test are not separate environments, every time it was called for an update on test to keep it equal to production one would risk dead time in development. Having the test environment separate from development also gives developers the chance of dismissing development projects without any affection on the test environment.

There is nevertheless also a drawback to having separate test and development environments. Every time a project has undergone one development cycle and needs to be tested, which often needs to be done several times before the job is complete, the project must be migrated to the test environment. This goes for both new projects as well as new iterations of a project.

One disadvantageous issue arising when a system is designed like the one discussed in this section may be the number of servers needed. As mentioned earlier every single environment in the SAP configuration run on separate servers. The more environments the more servers, and the whole system design may easily get out of hands if not every environment is carefully thought through. More servers mean more to maintain and more to keep track of. A too complex configuration and setup may also complicate the process of inducting people into the system.

5.5 Environments

The current environments configuration of the Knowledge Infrastructure consists of two environments; a dual purpose “Dev/Test” environment and a “Prod” environment. According to Pål Navestad, the project manager for the Data to Decisions project and consequently the Knowledge Infrastructure, it would in some ways have been desirable with only one environment. Keeping the number of environments low helps to ease the management and service operations of a system, and would probably keep down costs as well – both licensing and manpower. Nevertheless, to keep the intentions and the objectives of the Knowledge Infrastructure intact the need for more than one environment is explicit.

The need of having a production environment is patently. All the jobs extracting data from source systems, processing the data and turning the data into readable output usable to make business decisions, take place in the production environment. Following this line of thought, it is easy to see that doing all other work on the system, like developing jobs and testing them would potentially be very harmful to the system in case of anything going wrong. A job going wrong in either the development or the test phase could render the whole system inoperative. Consequently more than one environment ought to be implemented in such a system.

Having established the necessity of more than one environment, the question still remains open to whether or not there should be more than two environments, as is the scheme at the present time. Taking into consideration the best practice from SAP discussed in chapter 5.4, separation of the production environment and the test environment seems to have many advantages. Likewise, many of the other environments in SAP also do seem to have a lot going for them.

In most systems one would normally say that development and test environments should always be separate, precisely to avoid influence on the development environment from things being tested and vice versa. Separating the environments like this also makes it possible for a test environment to be identical to the production environment, something a development environment hardly will be over time. Joint environments as the Knowledge Infrastructure employs today easily leads to inconsistency between the data in the production environment and the DevTest environment. A separate development environment does not need to be totally consistent with the production environment, or the test environment for that matter. The development environment only needs data relevant to the jobs being developed, and with this environment separate from the others one can get away with only copying the required data to the development environment. This can also render a less “heavy” development environment.

A disadvantage of a separation of test and development would be space. Most datasets would have to be stored three times, optimally on three different servers. As long as the server capacity requirement for this is met, it would not be a major drawback. However it could represent a large item of expenditure if the company has to make new acquisitions for a new server or even several machines.

If the data machinery equipment remains as it is today, a separate test environment would have to fight over limited runtime recourses with the production environment. As a consequence of the fact that a separate pure test environment would need both a scheduling manager and a (work) flow manager to replicate the whole value chain of the production environment, separating the test environment would be undesirable because it would have to share the HW resources with the production environment. This is something that can not be done while simultaneously maintaining a stable production environment in this system.

One alteration from the system the way it is today, if the development and test environments were to be separated, would be that developed jobs would have to be promoted from development to test in order to be tested. If one is to have the whole “dev -> test -> prod” regime, the whole production setting must be replicated in test. Even though constant promoting jobs to test from development may seem like a small disadvantage, and also maybe more time consuming compared to the system today, it would really mean an improvement of quality and at bottom line less “stressful” for the developers.

Testing in an environment where there are no consequences for the users if anything goes wrong is a major point of advantage. Also, those promoting the jobs would have a simpler task promoting from test to development, because they would only run the exact same procedure of promoting as from development to test. In this final promoting stage the developers would not need to be involved, as the code would be successfully tested. System-wise one would achieve quality assurance of the promoting procedure to production, as the promotion procedure of the different jobs already would have been tested and proven between the development and test environments. A quality improvement like this could be very advantageous seeing that it is easy while documenting promotion to forget a table, forget a job or similar. Sudden missing tables or jobs in production as a consequence of a faulty promotion procedure could in the worst case signify a stop in the work flow and possible prevention of other jobs to be run.

Today in the Knowledge Infrastructure, efforts are made to have as good data in DevTest as possible. Still, keeping the development environment, or in this case the DevTest environment, consistent with the production environment is very difficult over time. Probably this means that the jobs today are in many cases of higher quality when promoted to production than in a tripartite environment when promoted to test. Even so, today one can not be positive that all the jobs promoted to production will function correctly, and when something fails it needs to be corrected and promoted all over again.

The situation as described above is actually the reason why nothing is promoted on Fridays or the day before public holidays. As discussed in chapter 5.4 the weekends are used for heavy updates of data in the system along with important offshore scheduling, and it is imperative that the Knowledge Infrastructure are up and running during the weekends to support these tasks.

Further disadvantages by separating the DevTest environment into two separate environments comprise more administration, more investment costs and higher management costs. Whether these costs outweigh the advantages of i.a. security in the form of more watertight bulkheads between the development and production environments, must be the responsibility of the project managers to contemplate.

In the event of the project managers deciding on carrying through a separation of the DevTest environment, there are many hardware considerations to do. The illustrations Figure 4-13, Figure 4-15 and the discussion in Chapter 4.2.3 points out the need for several servers for each environment. The ideal situation in the event of a new environment would be three new Windows servers and one new UNIX server. The most important server for an environment, or put in other words the server doing the heaviest jobs is the UNIX server. Unfortunately, without going into detail about price, costs and the funds available to the Knowledge Infrastructure, one UNIX server costs far more than three windows servers.

Although not mentioned earlier in the thesis, there is a possible solution for this at ConocoPhillips Norge even though acquisition of a new UNIX server is out of the question. There is at the moment an old UNIX server running for old datasets used in earlier versions of SAS. When this server is decommissioned it would be possible to reuse this server as the UNIX server for the new test environment. It is also possible for some of the environments to share a UNIX server, although the production environment should run on separate UNIX machine(s).

Another alternative is to set up one windows server to handle all three layers on the windows side; that is the metadata layer, the web layer and the data layer (called Metadata Tier, Mid Tier and Server Tier respectively in Figure 4-15). This solution can be combined with any one feasible setup on the UNIX side. Altogether this makes up for a well of feasible hardware combination possibilities, even though they are a bit fake compared to the solution with one UNIX server combined with three windows servers.

On the windows side one can also consider VMWare [25]. VMWare is a software suite allowing users to set up multiple x86 and x86-64 virtual computers and using one or more of these virtual machines simultaneously with the hosting operating system. Each virtual machine instance can execute its own guest operating system, such as (but not limited to) Windows, Linux and BSD variants. Using VMWare it is possible to set up as many virtual servers as one likes on the available hardware, although the number of virtual servers are of course restricted by CPU power, memory and other hardware specifications and requirements.

However smart a solution like this might seem, it is still a questionable solution. Managing virtual servers can be a more complicated job than having separate servers. At the same time one would limit the backup solutions in case of a hardware crash if one is running only one or a few Windows servers which again run all necessary servers on VMWare. Also one single hardware error could in this scenario potentially take out all services in one or more environments at the same time. On the other hand, a large collection of servers running VMWare can potentially provide better backup and recovery solutions by letting one of the other hardware servers start a new virtual machine, taking over for any other lost servers run on a virtual machine.

Concerning a prospective sandbox environment, this should be a true copy of and completely separate from the production line (the development, test and production environments) as discussed in chapter 5.4 about the sandbox environment in SAP. To briefly summarize the reason for this: It would be highly undesirable for a “playground”, for e.g. unskilled users, to be able to affect the production line of essential jobs for the company. Seeing this, there is only one prevailing solution, and that is a new UNIX server and three new Windows servers.

Having a separate sandbox environment available can be advantageous, especially at upgrades of the system like version upgrades of all or parts of the software. A separate environment enables the system managers to first run and test the upgrade on the sandbox environment. If anything goes wrong, this does not affect the production line in any way, and one can find out what went wrong and correct this at leisure. Once the upgrade is successful one can start upgrading the other environments, and now the people responsible would also have experience in doing the upgrade on this system. The minimum advantage of this is that the upgrade on the production line would probably go faster.

Also, after having successfully upgraded the sandbox environment, this could be used as a backup for the other environments during their respective upgrade procedures. That could require a rollback on the already upgraded sandbox environment, but would significantly increase the security during upgrades. If the other environments are migrated to the sandbox before upgrading, one could also avoid downtime on these environments if the sandbox could be utilized with the migrated environment. If desired, the sandbox could also be used for educational purposes, as discussed about the T24 SAP environment in chapter 5.4.

Irrespective of the solution for the other environments, production should and must run separate from the rest of the environments. As discussed in chapter 5.3 this means at present three Windows servers and two UNIX servers; one UNIX for batch and one for users as one cannot let the batch jobs be affected by e.g. heavy stored procedures. This was the reason for the procurement of the new SVGAIXSAS03 UNIX server.

As mentioned in chapter 4.2 the DevTest environment shares the SVGAIXSAS02 server with the production environment today. This is a sinister solution and as the use of the Knowledge Infrastructure only will increase hereafter, it is not a problem to justify the use of two UNIX machines for the production environment alone.

To avoid any misconception it is also worth mentioning that a separate test environment, although requiring jobs copying production data to test every night, would not inexpediently affect the production environment in any serious ways. Sheer copying from production to DevTest is also done today for some data, and this is not affecting the production environment at all.

5.6 Further work

Much of the objective of this thesis is to describe the Knowledge Infrastructure system. For this reason it is to a certain extent limited what is adequate to discuss. However, while some of the problems discussed in this chapter may be general as to what concerns data warehouses, some of them may be of a system specific character and not applicable to other although similar systems.

Even though this thesis is discussing issues specific for the Knowledge Infrastructure, this is a system still (as of writing) undergoing development. More relevant issues may emerge as a result of the ongoing development process. The choice of solutions for certain issues may change the system or otherwise affect the design of the system in such a way that the description of the system provided in this thesis yields faulty.

In a situation where the above-mentioned situation applies, it will be necessary to update the description of the system to comply with the changes made to it.

Further work on the solutions presented in this thesis lies mainly in evaluation of the requirements and mapping of demands with regard to the environment configuration. This is something ConocoPhillips Norge must do, preferably in collaboration with SAS. If any of the proposed solutions is chosen, ConocoPhillips Norge must contemplate what hardware configuration to use. For the company it will be important to map the vulnerability of the system with regards to both stability and security to assess any procurement of new hardware or software.

After the development process has finished, the system will most probably go into a continuous maintenance and update cycle. It may be of interest to apply changes, other than the ones proposed here, to the system to ease this cycle or to optimize for e.g. software updates or hardware upgrades.

This thesis has mainly explored today’s system at ConocoPhillips and possibilities based on the current system setup. Introduction of new hardware, software or new environments can make it pertinent to look into new possibilities. Different environment setups arising out of new system modules can turn out to be more favorable than the ones proposed in this thesis.

6 Conclusion

In this thesis the possibilities for environment configuration in the Knowledge Infrastructure data warehouse system have been the subject of investigation. Focusing on the current setup the task was to describe the system and to outline any current problem areas. The most relevant current problem areas, namely isolation of data marts and environment setup, were documented and explored in the thesis. Supplementary the objective was to investigate any other suitable system to survey best practice from similar areas.

ConocoPhillips's Knowledge Infrastructure is a huge project with many aspects to consider when developing and introducing it into the daily operation of the company. For the time being it seems like this is going well and that it will be a successful data system, serving the company in ways of both time, economical and operational aspects. After full deployment and start of use of the system, it is the understanding of this thesis that the Knowledge Infrastructure will be a system well worth the effort invested in development and funding of new hardware and software. As long as the Knowledge Infrastructure remains stable it will very likely be a valuable asset to the company.

The resent separation of online marts from the batch server in the Knowledge Infrastructure is in the opinion of this thesis a well thought-through amelioration of the system. Bringing few if any disadvantages, this upgrade of the system results in higher system stability as well as increased security.

At choice of environment configuration the main focus was that any solution had to be advantageous to the system, and the same time must be practical to implement. One of the advantages to follow such a solution is an increase of overall stability to the system. In the process of evaluating different configurations, the thesis looked into best practice from the SAP system in use at ConocoPhillips.

Having a system configuration like the one in SAP seems to be advantageous in several cases. Even so this thesis confines itself to suggesting one or two additional environments for the Knowledge Infrastructure. Implementation of new environments into an already existing system is a complex and time consuming operation. Much can go wrong, and the system may not behave exactly as foreseen. After a big change to the system, like adding new environments, needs may shift and urgent requirements may change. It would therefore seem like a good idea, in case of implementing any new environments, to start off with one or two additional environments. Further one should in turn evaluate the situation and redefine the requirements according to the new configuration of the system, how it works and whether or not the new solution has solved the issues it was meant to deal with.

In comparison of the SAP system with the Knowledge Infrastructure, the Knowledge Infrastructure has different performance specifications from those of the SAP system. A FIT cycle of two upgrades a year is far too infrequently for the Knowledge Infrastructure, and would be impossible to implement into this system the way it is today.

It is the opinion of this thesis that ConocoPhillips Norge should aim at separating the DevTest environment completely from the production environment. Having the production environment share one of the UNIX servers with the DevTest environment like in today’s solution is risky, especially with stability issues in mind.

Further the thesis recommends separating the development from the test environment, resulting in a three-environment solution for the Knowledge Infrastructure. The optimum would be to run all the environments on separate server parks, but taking costs into consideration, then a development, test and a potential sandbox environment can run on a shared UNIX server. Limitations on the UNIX server are after all system IO and not CPU power.

In the case of having the resources for a sandbox environment available to the Knowledge Infrastructure project, the thesis also suggests that the Data to Decisions team further evaluate the advantages and gains of a sandbox environment on the system, compared to costs of implementing it. A separate sandbox environment can greatly reduce the risks of version upgrades, and also brings advantages in training new users of the system. Viewed in the light of this, the thesis strongly advises ConocoPhillips Norge to consider implementing a sandbox environment.

The opinion of the thesis is that a good recommendation for the distribution of hardware on the environments[5] would be a park of Windows servers running VMWare, and two UNIX machines performing the roles as the “ETL APP” and “BI APP” servers. To save on costs it is possible to leave out one UNIX box and let the remaining UNIX computer play the roles of both ETL and BI APP servers in all the development, test and any potential sandbox environments.

A short conclusion to the issue of environments would be that more environments yield higher security for a higher cost.

In the opinion of the author this thesis has delivered a good survey of the system with regard to the current configuration. For ConocoPhillips Norge it will be highly feasible to separate the DevTest environment completely from the production environment, and also to separate DevTest into two individual environments. In the future it remains to see how full use of the Knowledge Infrastructure after completion and final deployment will influence the performance of the system.

[5] These are the development, test and sandbox environments - not the production environment.

References

Ref. Nr.	Reference
[1]	Mallach, E., *Decision support systems and data warehousing*, McGraw-Hill, 2000, ISBN 0072899816
[2]	Inmon, W.H.,*Building the data warehouse 3rd ed*., Wiley, 2002, ISBN 0-471-08130-2
[3]	McFadden, F., Hoffer, J.A. & Prescott, M., *Modern Database Management 7th ed.,* Prentice Hall, 2004, ISBN 0-131-45320-3
[4]	Kimball, R., Reeves, L., Ross, M. & Thornthwaite, W., *The data warehouse lifecycle toolkit*, Wiley, 1998, ISBN 0-471-25547-5
[5]	Poe, V., Klauer, P. & Brobst, S. *Building a Data Warehouse for Decision Support 2nd ed.*, Prentice Hall PTR., 1997, ISBN 0137696396
[6]	Codd, E.F., The Relational Model for Database Management, Version 2, Addison Wesley Publishing Company, 1990, ISBN 0-201-14192-2
[7]	Corey, M.J., Abbey, M.S., Abramson, I. & Taub, B., *Oracle8i Data Warehousing,* The McGraw- Hill Companies, 2001, ISBN 0072126752
[8]	SAS Institute, ConocoPhillips, *BICC Workshop Report ConocoPhillips*, January 2007
[9]	SAS Institute, *IT Operations Document – Data to Decisions – KPI-Reporting ConocoPhillips,* 2006
[10]	SAS Institute, *SAS Enterprise Solutions Configuration Checklist – Strategic Performance Management 2.2*

Ref. Nr.	Reference	Accessed
[11]	Erlend Falch-Pedersen, *Exploitation of Data Warehouse When Monitoring Vehicle Information, Master Thesis in Information and Communication Technology*, Agder Univeristy College, May 2003	26.05.2007
	http://student.grm.hia.no/master/ikt03/ikt6400/g31/rapport/rapport.htm	
[12]	Relational Model: Normalization, *Windows Services – Introduction to Data Modeling*, The University of Texas at Austin, 2004	26.05.2007
	http://www.utexas.edu/its/windows/database/datamodeling/rm/rm7.html	
[13]	Wikipedia, the free encyclopedia, *Online Transaction Processing*, May 2007	26.05.2007
	http://en.wikipedia.org/wiki/OLTP	
[14]	Wikipedia, the free encyclopedia, *Online Analytical Processing*, May 2007	26.05.2007
	http://en.wikipedia.org/wiki/OLAP	
[15]	Nigel Pendse, The Olap Report, *What is OLAP?*, 2005	26.05.2007
	http://www.olapreport.com/fasmi.htm	
[16]	Eckerson. W., *Smart companies in the 21st century: The secrets of creating successful BI solutions. (In A report of the Data Warehouse Institute)*, 2003	25.09.2006
	http://www.dw-institute.com/bireport/	

[17] IBM Software Information Center, *IBM Red Brick Warehouse 6.3*, 2004 26.05.2007
http://publib.boulder.ibm.com/infocenter/rbhelp/v6r3/index.jsp

[18] Leon Gong, Mike Olivias, Christine Posluszny, Donna Venditti & Geroge McMillian, IBM developerWorks, *Deliver an effective and flexible data warehouse solution, Part 2: Develop a warehouse data model*, 2005 26.05.2007
http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0507gong/

[19] Sybase, *WarehouseArchitect User's Guide, PowerDesigner Version 6.1.3, Document number: AA04531*, 1999 26.05.2007
http://manuals.sybase.com/onlinebooks/group-wa/wag0320e/waug

[20] Il Blog di: Eccentrica Sparagli Dopo, *SQL Server 2005 appunti al volo*, May 2005 26.05.2007
http://community.visual-basic.it/sabrina/archive/2005/05/10/11942.aspx

[21] Wikipedia, the free encyclopedia, *Vertical integration*, May 2007 26.05.2007
http://en.wikipedia.org/wiki/Vertically_Integrated_Company

[22] Wikipedia, the free encyclopedia, *Horizontal integration*, May 2007 26.05.2007
http://en.wikipedia.org/wiki/Horizontal_integration

[23] VMWare, *VMWare: Virtualization, Virtual Machine and Virtual Server Consolidation*, 2007 26.05.2007
http://www.vmware.com/

[24] DataModel.org, *Rules of Data Normalization*, 2005 26.05.2007
http://www.datamodel.org/NormalizationRules.html

[25] Wikipedia, the free encyclopedia, *Boyce-Codd normal form*, May 2007 26.05.2007
http://en.wikipedia.org/wiki/BCNF

Appendix A – Hardware Specification for Servers at COPNO

AIX Machine 1
This is the SVGAIXSAS01server

This is the AIX machine currently running SAS 8.2 in production and will be used for the Dev/Test level when it's no longer needed in production. This will be done at a later stage in the process.

- *No specifications available* -

AIX Machine 2
This is the SVGAIXSAS02 server

IBM eServer p5 with POWER5 architecture.
AIX 5L Version 5.3 ML05
4 * 1.50 GHz 64-bit POWER5 processors
16 GB of 533Mhz DDR2 memory
High speed I/O
Different disk volumes for Data/work/temp

AIX Machine 3
This is the newly installed SVGAIXSAS03 BI APP server

IBM System p5 520Q
4 * 1.65 GHz 64-bit POWER5+ processors
16 GB of 533Mhz DDR2 memory
High speed I/O
4x 144 GB disks 15 k speed Raid 0
2x 72 GB disks , 10k speed (system)
2x 144 GB disks, 10k speed

Win Machine 1
Wintel 32bit
Windows 2003 Server Enterprise Edition
2 CPU
16GB fast memory
(Should be upgradeable to at least 8 processors and 32GB of memory)

Win Machine 2
Wintel 32bit
Windows 2003 Server Enterprise Edition
4 CPU
16GB fast memory
(Should be upgradeable to at least 8 processors and 32GB of memory)

Win Machine 3
Wintel 32bit
Windows 2003 Server Enterprise Edition
2 CPU
16GB fast memory
(Should be upgradeable to at least 8 processors and 32GB of memory)

Win Machine 4
Wintel 32bit
Windows 2003 Server Enterprise Edition
2 CPU
16GB fast memory
(Should be upgradeable to at least 8 processors and 32GB of memory)

Win Machine 5
Wintel 32bit
Windows 2003 Server Enterprise Edition
4 CPU
16GB fast memory
(Should be upgradeable to at least 8 processors and 32GB of memory)

Win Machine 6
Wintel 32bit
Windows 2003 Server Enterprise Edition
4 CPU
16GB fast memory
(Should be upgradeable to at least 8 processors and 32GB of memory)