Dissertation

Zero-Latency Data Warehousing

Toward a Zero Latency Event Sensing and Responding Data Warehousing

Ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

unter der Leitung von
o.Univ.-Prof. Dipl.-Ing. Dr.techn. A Min Tjoa
Institut für Softwaretechnik & Interaktive Systeme
Technischen Universität Wien
und
Univ.-Prof. Dr. Roland Wagner
Institut Für Anwendungsorientierte Wissensverarbeitung
Johannes Kepler Universität Linz

eingereicht an der
Technischen Universität Wien
Fakultät für Technische Naturwissenschaften und Informatik

von

Nguyen Manh Tho

Matrikelnummer: 0226752
Viktor-Kaplan Strasse 6-8/2/418
A-1220 Wien

Wien, im August 2005

Acknowledgement

Pursuing a PhD program is an enjoyable trying progress where dissertation is a significant checkpoint's deliverable. The work in this dissertation has been conducted within 3 years studying at the Institute of Software Technology and Interactive System, Vienna University of Technology. However, without help, support, and encouragement of several persons, I would never have been able to finish this work.

First of all, I would like to express my sincere gratitude to my supervisor, Prof. A Min Tjoa for his inspiring and encouraging way to guide me to a deeper understanding of knowledge work, and his valuable guidance during the whole work with this dissertation. He showed me different ways to approach a research problem, broaden my view to investigate the topic, and the need to be persistent to accomplish any goal. More important, I deeply indebted for his spiritual supports in many aspects of life. This is the best motivation which helps me passing any difficulty to reach the target.

I would like to thank Prof. Roland Wagner, my second supervisor for his guidance and comments to my work as well as for significant experience gaining from DEXA and DaWaK conferences.

Special thanks are given to Dr. Josef Schiefer for valuable discussions and prompt assistance in technical aspects. I very much appreciate for his kindly attitude to offer discussion time in weekends and I am very happy about our fruitful collaboration in joint publications.

I am also grateful to Prof. Peter Brezany, Institute for Software Science, University of Vienna, for his interesting and inspiring discussions on Grid computing. He also gave me valuable comments and revision on my publications.

I would like to express my deep appreciation to Dr. Michael Aigner, Dr. Martin Windisch, and Dr. Jaromir Nemec at T-Mobile Austria GmbH. Their friendly collaborative discussions offer me a lot valuable experiences in real world's practical environment.

Thanks to all my colleagues at the Institute of Software Technology and Interactive System for providing a good working collaboration atmosphere as well as further debates and discussion in many topics. Especially, my obligation is given to Prof. Gerti Kappel, Dr. Beate List for their collaborative assistance in project work, Dr. Andres Rauber for valuable advice in performing research, Shuab Karim and Khalid Latif for their English revision efforts and interesting discussion. Thank also for the friendship with Hoang Huu Hanh, Khabib Mustofa, Shah Khusro, Ahmed Mansoor and Amin Andjomshoaa. I also would like to send my appreciation to Mrs. Maria Schweikert, Mag. Markus Klemen and Ing. Michael Schadler for their support in providing me the perfect working environment.

Last but not least, I am greatly indebted to my devoted wife Thuan An and my lovely daughter Annie Thien Nga for supporting me with love and spiritual understanding. Their love and support without any complaint or regret has enabled me to complete this dissertation.

Financial support has been provided by the Technology Grant South East Asia (No.322/1/EZA/ 2002) of the Austrian Council Research and Technology and the ASEA-UNINET.

Nguyen Manh Tho.

Abstract

Increased data volumes and accelerating update speeds are fundamentally changing the role of the data warehouse in modern businesses. More data, coming in faster, and requiring immediate conversion into decisions means that organizations are confronting the need for zero-latency Data Warehousing.

The *Zero-Latency Data Warehouse* (ZLDWH) is an extended stage in the data warehouse development, which will enable a complete business intelligence process to observe, understand, predict, react to, reorganize, monitor, automate and control feedback loops in the minimal latency. With the aim to decrease the time it takes to make a business decision, ZLDWH must satisfy some new requirements of the speed driven business such as operational decision support, closed-loop integrated analytics, and automated reaction.

This doctoral thesis reviews the start-of-the-art of the Zero-Latency Data Warehouse and investigates several approaches towards its implementations:

- The *event-feeded Comprehensive Slowly Changing Dimension (cSCD).* We propose an extended event-oriented SCD solution which is mixed from SCD type 2 and type 3 to validate the event-messages and reconstruct the complete history of the dimension. The solution could be applied to any dimension data, allow the user to execute the well-performing queries on both historical and current state of dimension. The prototype package is implemented and tested at T-Mobile Austria.

- *The Grid-based Zero-Latency Data Stream Warehouse* (*GZLDSWH).* We present a framework of building a Grid-based Zero-Latency Data Stream Warehouse to tackle the resource limitation issues in data stream processing without using approximation approaches. The GZLDSWH is built upon a set of Open Grid Service Infrastructure (OGSI)-based services and Globus Toolkit 3 (GT3) with the capability of capturing and storing continuous data streams, performing analytical processing, and reacting autonomously in near real time to some kinds of events based on well-established Knowledge Base.

- *Sense & Response Service Architecture (SARESA).* We introduce Sense & Respond loops that support a complete Business Intelligence process to sense, interpret, predict, automate and respond to business processes. Our approach enables real-time analytics across corporate processes, notifies with actionable recommendations or automatically triggers, effectively closing the gap between Business Intelligence systems and business processes. We also describe the prototype implementation applied in the Mobile Phone Fraud Detection application.

Finally, we mention some future trends of non-standard Data Warehousing applications and discuss about the important role of zero-latency data warehouses techniques which could be applicable in these emerging Data Warehousing applications.

Zusammenfassung

Das drastisch zunehmende Wachstum des Datenvolumens und die Anforderungen für beschleunigte Updates in Unternehmen haben die Aufgaben von Data Warehouses signifikant verändert. Immer mehr Daten, die in immer kürzeren Intervallen anfallen, erfordern das sogennante Zero Latency Data Warehousing (Fast Echtzeit Data Warehousing) um für den Entscheidungsträger und Institutionen zeitgerechte Entscheidungen treffen zu können

Das Zero-Latency Data Warehouse (ZLDWH) stellt eine weitere Stufe der Entwicklung von Data Warehouses, welche eine ganzheitliche unternehmensweite Prozeßführung eines Unternehmens unterstützt. Eine solche Prozeßführung ist durch die intelligente Prozesse, welche die Aktivitäten der Beobachtung, des Auffassens, der Vorhersage, des Reagierens und der automatischen Rückkoppelung inkludiert, charakterisiert. Das Ziel der Erreichung einer minimalen Zeitverzögerung für Entscheidungsprozesse impliziert neue Anforderungen, die zu lösen Inhalt dieser Dissertation ist. Vor allem sind dies Aspekte der operationalen Entscheidungsunterstützung, des unmittelbaren („closed loop") Durchgriffs von Data Warehouses auf operationale Datenbanken und der automatisierten Reaktion auf besondere Zustände des Data Warehouse.

Die Dissertation behandelt den Stand der Forschung auf dem Gebiet des ZLDWH und bietet Lösungsansätze und Implementierungen in folgenden Bereichen:

- Das *event-feeded Comprehensive Slowly Changing Dimension (cSCD)* wird hier vorgeschlagen als eine Erweiterung der SCD Typ 2 Lösung, welche Komponenten des SCD Typ 3 beinhaltet. Ziel ist es eine vollständige Rekonstruktion historischer Daten der Dimensionsdaten u erreichen. Eine prototypische Implementierung wurde für eine sehr große T-Mobile Applikation getestet.

- Das *Grid-based Zero latency Data Stream Warehouse* (*GZLDSWH)* stellt ein neuartiges Konzept dar, welches zur Erreichung der obigen Ziele für sogenannte Data Streams die Grid-Computing-Technologie anwendet. Das GZLDSDWH ist aufgebaut auf Grid-Services des Open Grid Service Infrastructure (OGSI) und Globus Toolkit 3 (GT3).

- *Die Sense & Response Service Architecture (SARESA)* wurde hier entwickelt um eine Rückkoppelung zwischen Analyse im Data Warehouse und den notwendigen operationalen Aktivitäten zu erreichen. Der vorgestellte Ansatz ermöglicht eine beinahe Echtszeitreaktion basierend auf Analysen der Data Warehouse Informationen durch automatische Trigger und Entscheidungsempfehlungen, welche die Zeitspanne zwischen Analyse und Prozeßaktivität erheblich verkürzt. Eine prototypische Implementierung für die Auffindung von Berügereien für T-Mobile wird hier vorgestellt.

Schließlich werden mögliche Zukunftsentwicklungen für Nicht-Standard.Anwendungen skizziert um auch hier die Notwendigkeit von Fast-Echtzeit-Ansätzen zu dokumentieren.

Table of Contents

List of Figures

List of Tables

PART I.

FUNDAMENTALS

Chapter 1. Introduction

"Like fish, information that's fresh has the most value"...
Ross Altman, Research Director Gartner Group, 2001

In today's competitive global business environment, understanding and managing enterprise wide information is crucial for making timely decisions and responding to changing business conditions. The tremendous amount of data generated by day-to-day business operational applications and critical applications require the means for a near-instantly change of raw data into information for making effective business decisions. This requirement could be satisfied with the Near Real-time Data Warehousing approach which provides the RIGHT data... to the RIGHT people... at the RIGHT time.

1.1. Overview and Motivation

The widespread use of the Internet and related technologies in various business domains has accelerated the intensity of competition, increased the volume of data/information available, and shortened decision-making cycles considerably. Consequently, strategic managers are being exposed daily to huge inflows of data and information from the businesses they manage and they are under pressure to make sound decisions promptly [Westerman 2000]. Typically, in a large organization, many distributed, heterogeneous data sources, applications, and processes have to be integrated to ensure delivery of the best information to the decision makers. In order to support effective analysis and mining of such diverse, distributed information, a data warehouse (DWH) collects data from multiple, heterogeneous (operational) source systems and stores integrated information in a central repository. The traditional role of a data warehouse is to support strategic decision-making with the rich historical data analysis requirements.

Among the changes brought by the Internet over the past decade, one of the most significant is the acceleration in the pace of business. For many organizations, getting information and services to decision makers, partners, and customers when they want it and where they want it has simply become a reality of doing business. As the pace of commerce quickens, the amount of real-time electronic information that must be analyzed i.e. real-world event streams flowing into today's business systems, is exploding and the required response to these real-world events often comes down to milliseconds [StreamBase 2005]. For instance, individuals outside the organization have come to expect instantaneous online shopping, banking, and customer service. Likewise, constituencies within the organization are demanding immediate access to information on sales metrics, supply chain, operations, and financials [GoldenGate 2004].

Since market conditions can change rapidly, up-to-date information should be made available to decision makers with as little delay as possible. It is required that the DWH must have the ability to support analysis at the speed of the business, so that the end-users always have the information available in the DWH when they need it. However, for a long time it was assumed that data in the DWH can lag at least a day (if not a week or a month) behind the actual operational data. That was based on the traditional assumption that business decisions did not require actual information but very rich amount of historical data. Existing data integration tools often rely on this assumption and achieve high efficiency in periodically loading large amounts of data into the DWH system. Traditionally, there is no real-time connection between a DWH and its data sources, because the write-once read-many decision support characteristics is in conflict with the continuous update workload of operational systems with the consequence of poor response times. Therefore, batch data loading is performed during frequent update windows (e.g. every night). With this approach the analysis capabilities of DWHs are not affected. Existing batch oriented approaches often take for granted that they are operating during a batch window and that they do not affect or disrupt active user sessions.

While this still holds true for a wide range of DWH applications, today's DWHs are broadening their roles to include more and more business critical task, supporting not only strategic decisions but also the operational business process. We see a fundamental shift in the operations of businesses when they combine strategic data with operational data. For example, a large credit card company uses data from its own databases (account balance, customer name, billing zip code) to process a transaction. In addition, it may augment that data with tertiary sources such as credit scoring models, FICO scores and even neural networks to determine not only whether the transaction should be authorized, but also patterns of historical data and account profiling methods to evaluate risk potential on a single transaction [SUGI29 2004]. In such a system, the data warehouse must gather the data from all source systems in a timely manner, and make the data immediately available to the analysis processes, to support all sorts of analysis required. Particularly, some applications require even the automatic reaction to critical situations such as Fraud detection, business activity management (BAM) etc.

Due to the fact that many operational decisions e.g. promotion effectiveness, customer retention, key account information [Hasten 2002] need actual yet integrated and subject oriented data in real-time or near real-time [Langseth 2004]. Data Warehousing may contribute to an efficient information supply between transactional applications and decision support applications. However, information 'backflows' take their way indirectly (i.e. by actions of decision makers) into the transactional systems and not directly through the data warehouse (open-loop approach).

What is decisive and desirable is a closed loop approach interlinking operational and strategic decision-making. Therefore, Business Intelligence has to be enhanced towards a closed loop real-time Business Intelligence, shortening the period of time between the occurrence of a business event that requires an appropriate action by the organization and the time the action is finally carried out (Sense and Respond).

Furthermore, businesses are increasingly operating in real time, and customers around the world expect service on demand with the requirement to access information 24x7, 365 days a year, i.e. no downtime in operation [Informatica 2004]. The new desire for monitoring information about business processes in near real-time is breaking the long-standing rule that data in a DWH is static except during the downtime for data loading. Continuous data integration [Bruckner 2002] aims to decrease the time it takes to integrate certain (time-critical) data and to make the information available to knowledge workers or software agents. This enables them to find out what is currently happening, decide on what should be done (utilizing the rich history of the DWH), and therefore react faster to typical and abnormal data conditions (tactical decision support). Continuous and near real-time data integration for DWHs minimizes propagation delays from operational source systems of an organization, which are responsible for capturing real world events. This improves timeliness by minimizing the average latency of the time-span when a fact is first captured in an electronic format somewhere within an organization until it is available for knowledge workers who need it.

1.2. Target Criteria

Started from the kernel concepts of zero-latency data warehouse, this thesis discusses the related issues which enable a zero-latency analytical environment in which streams of real-time complex events are processed, analyzed, and acted upon with virtually zero latency. Towards such a zero-latency analysis environment, in this thesis the following criteria are considered and investigated:

- **Data freshness**. The need for data freshness escalates significantly, because sensitive data have to be updated more frequently in order to improve decision-making, we need a support for various data freshness requirements (high/low priority data).

- **Continuous data integration**, which enables (near) real-time capturing and loading from different operational sources and also event-based triggering of actions even during data integration. This sort of data integration results in an increasing number of late-arriving data (e.g. due to propagation delays), because timeliness becomes more important.
- Highly available analytical environments based on an **analysis engine** that can consistently generate and provide access to current business analyses at any time not restricted by loading windows typical of the common batch approach.
- **Active decision engines** that can make recommendations on automatically recognized situations or exceptions, or even (rule-driven) tactical decisions for routine decision tasks encountered in an analytical environment.
- Changes of a business process or settings in the operational environment must not disrupt the interoperability with the event stream processing. An **adaptive platform for the event stream processing** is required to deal with the changes of the operational environment.
- The number of users and performance requirements for a zero-latency data warehouse will increase by orders of magnitude the deployment of analytic applications which enables tactical decision support. Therefore, **high availability** and **scalability** are indispensable criteria.

1.3. Outline

The structure of the thesis is organized into three parts as follow:

Part I. Fundamentals

In this part, we give a brief overview of the Zero-Latency Data Warehousing Environment, the background concepts and other related work. This part includes the following chapters:

- Chapter 1 gives a general overview and introduction.
- Chapter 2 describes the concepts and motivation of the Zero-Latency Data Warehouse. In this chapter, we will review the state-of-the-art of the zero-latency data warehouse concept and its related technologies.

- Chapter 3 deals with the background concepts and related work. In this chapter, we introduce some essential concepts and other research results related to our topic such as Zero-Latency Enterprise, Active Data Warehouse, Real-time Data Warehouse, Real-time Decision Support, Continuous Data Streams processing, analyzing and mining. The concepts and applications of Service Oriented Architecture (SOA) and Grid Computing will also be briefly introduced in this chapter.

Part II: Implementing Zero-Latency Data Warehousing Environment.

In this part, we will present our research activities in enabling Zero-Latency Data Warehousing environment in different perspectives. We first introduce a case study and research work conducted at T-Mobile, Austria in chapter 4 with the aims to solve the event-oriented Slowly Changing Dimension. Chapter 5 presents our frame work of applying Grid Computing to build a Zero-Latency Data Stream Warehousing system without using the traditional statistical approximation methods. Finally, in chapter 6, we describe the Sense & Response approach to enhance the traditional Business Intelligence and Data Warehousing system with the capability of sense, interpret, predict, automate and respond to events in the near real time.

- In Chapter 4, we discuss about the event-oriented processing in Data Warehouse and present our practical work at T-Mobile, Austria where we solve the event-feeded Slowly Changing Dimension issue. The event-based Data warehouse refresh approach will be presented and compared with the existing snapshot based approach. We will also discuss how the invalid events are sold to avoid inconsistent result in the Data Warehouse. Parts of this chapter will be published in the Lecture Notes in Computer Science Volume 3589 of Springer Verlag, at the 7th International Conference in Data Warehousing and Knowledge Discovery (DaWaK 2005).

- Chapter 5 discusses about Grid-based Zero-Latency Data Stream Warehouse (GZLDWH) which is in our vision, to apply the Grid Computing technology for lossless storing and analyzing the continuous data streams. We discuss about the components of the GZLDWH built upon on the Open Grid Service Infrastructure (OGSI)-based services and the Globus Toolkit version 3 (GT3). The collaboration and interaction between these services will be investigated. The Dynamic Service Invocation controller and the OLAP Cube Management service are described in some implementing aspects. This part is submitted to the

International Journal of Data Warehousing and Mining (IJDWM) and has been accepted for publication in Volume 1, Issue 4, 2005.

- Chapter 6 introduces another approach to conduct Zero-Latency analysis and response to continuous event streams. An enhanced Business Intelligence architecture that covers the complete process to sense, interpret, predict, automate and respond to (business) events is applied to decrease the time it takes to make business decisions. A service oriented infrastructure namely SARESA which enables smoothly embed the infra-communication between event analytical services and system services will be described in details. We also describe in details the SARESA prototype implementation applying in Mobile Phone Fraud Detection scenario. The prototype is implemented using C# Visual Studio 2005 beta and Microsoft SQL Server 2005 beta.

Part III. Non-Standard Data Warehousing – Future Trends and Research Challenges

In this part, some other non-standard Data Warehousing applications will be introduced in chapter 7. We will also discuss about the future trends and the potential research challenges in this chapter. Finally, the last chapter (chapter 8) gives the conclusion and ends the thesis.

- Chapter 7 presents some research activities and challenges in other non-standard DWH applications such as Spatial and spatio-temporal Data Warehousing, Process Warehousing, Web Data Warehousing, Data Warehouse and the Grid. In each topic, we briefly discuss about its motivation, its definition and requirement, and elaborate the research challenges. We also mention the relation between Zero-Latency Data Warehousing and these non-standard Data Warehousing applications.

- Finally, chapter 8 briefly concludes the thesis and sketches future work.

Chapter 2. Zero-Latency Data Warehousing: Concepts and the State-of-the-art

Information latency can make the difference between good and disastrous decisions.

Vision Solutions, 2002

Data Warehouse (DW) and Business Intelligence (BI) applications are separate stand-alone decision-support systems used for strategic planning and decision-making. As these applications have matured, however, it has become apparent that the information and analyses they provide are vital for tactical day-to-day decision-making as many organizations can no longer operate their businesses effectively without them. Consequently, there is a trend towards a Zero-Latency Data Warehousing system in which decision processing is integrated into the overall business process. The advent and evolution of e-business is also an encouraging factor for this integration because e-organizations need to react much faster to changing business conditions in the e-business world. This chapter reviews the state-of-the-art of the Zero-Latency Data Warehousing and discusses its related perspectives.

2.1. Introduction

In the competitive market environment nowadays, information becomes a vital business asset for every organization. To take advantage of that asset, organizations are turning to Data Warehousing, which lets them combine large amounts of data to create a unified, consistent view of information source for better monitoring and management of their (business) activities. The concept and application of Data Warehousing solutions has progressed rapidly since its inception in the mid-eighties. Data Warehousing has become very popular among organizations seeking competitive advantage by getting strategic information fast and easy. Since William Inmon coined the phrase "data warehouse" in 1990 [Inmon 1992], it has attracted the attention from Information System (IS) managers and vendors.

In the 1990s, the growth of enterprise applications like ERP and CRM prompted the need for data warehouses to address issues such as degrading system performance, escalating reporting requirements, and the limitations of the installed infrastructure [GoldenGate 2004]. By implementing the data warehouses on separate servers and databases to the Online Transaction Processing Systems (OLTP), organizations could run reporting applications and perform complex analytical queries without negatively affecting the OTLP system.

Within that traditional Data Warehouse environments [Chaudhuri 1997], the data feeding from OLTP data sources to the separate Data Warehouse was accomplished via sets of comprehensive ETL utilities. These tools are reliable and capable of performing many required data movement tasks. However, nearly all ETL tools are batch-oriented, which means there is an inherent lag between the time data changes on the source systems and when it is available for query by business users. An alternative approach to populating the data warehouse is to leverage the expertise of DBAs or programmers to complement ETL tools using custom-coded scripts, but this can be a costly layer to add to an infrastructure. Specialized coding also delays implementation of new features that users require, which can reduce system flexibility and decrease access to timely data. In addition, custom scripts are hard to maintain as the infrastructure changes.

As demand for real-time information and services grew in the late 1990s and early 2000s, many organizations found that their Data Warehousing infrastructures were unable to keep up with the faster pace of business. In particular, traditional methods of capturing data from OLTP databases and moving it periodically to the Data Warehouse were not able to cope with the need for speed and high-volume operation. These requirements demand towards enhanced Data Warehousing trends so called Zero-Latency Data Warehousing (or Real-time Data Warehousing interchangeably).

2.2. Traditional Data Warehouse

Data Warehousing arose out of the need of many businesses to view and analyze data from many different operational systems in order to get a complete understanding of the business. A Data Warehouse is a separate architecture used to maintain critical historical data that has been extracted from operational data storage and transformed into formats understandable to the organization's analytical community. Gradually, Data Warehouses have grown to be one of the largest sectors of data management systems. It generates benefits for the business when it transforms the intelligence contained in the data into better decision-making, which results in effective action.

The origin of the data warehouse can be traced to studies at MIT (Massachusetts Institute of Technology) in the 1970s which were targeted at developing an optimal technical architecture guideline for analytical applications. A core principle that emerged was to segregate operational and analytic processing into layers with independent data stores and radically different design principles.

Meanwhile, IBM was tackling a different aspect of the information management problem: enterprise integration. In 1988, Barry Devlin and Paul Murphy of IBM Ireland introduced the term "information warehouse" for the first time and defined it as: "A structured environment supporting end users in managing the complete business and supporting the (information technology department) in managing data quality." [Devlin, Murphy1988]

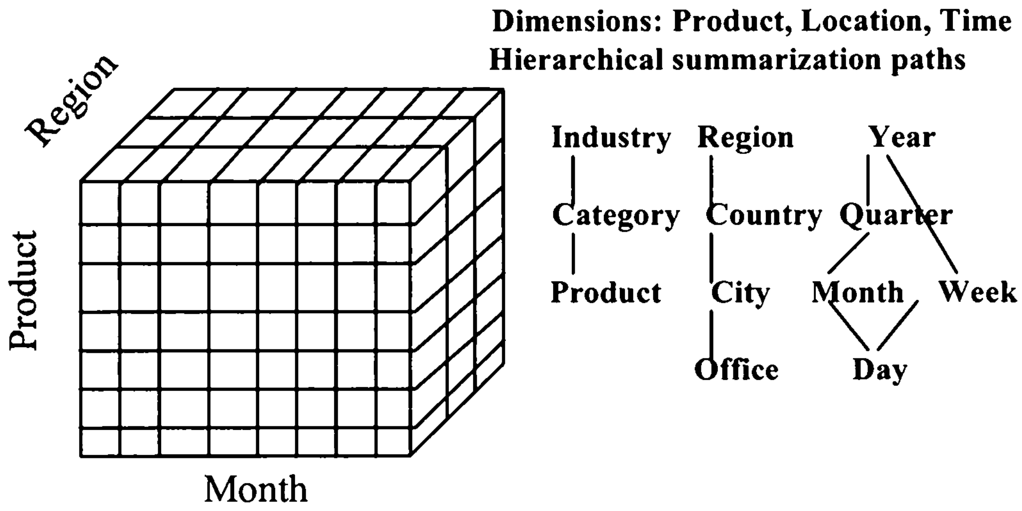


Figure 2.1 Multidimensional (MD) Data Model and Dimension Hierarchy

William H. Inmon, who is widely accepted as the mental-father of Data Warehousing has been working on Data Warehousing concepts since 1983, and used for the first time this term in 1992 [Inmon 1992]. He defined "*Data Warehouse is a subject oriented, integrated, non-volatile and time-variant collection of data in support of management's decisions*". In 1993, Ted Codd coins the term OLAP (On-Line Analytical Processing) and defined the famous 12 OLAP rules [Codd 1993]. Based on their definitions, a data warehouse is actually a comprehensive system that includes all the processes, tools and technologies necessary for extracting data from many operational, legacy, possibly heterogeneous data sources and managing them in a separate storage (warehouse) to provide end-user decision-support access. OLAP tools are well-suited for complex data analysis, such as multidimensional data analysis, and to assist in decision support activities. The multidimensional (MD) data model has been proved to be the most suitable for OLAP applications (Figure 2.1).

In database research terms, the work has concentrated on the physical and logical level rather than the conceptual level. The data models employed have been of the multidimensional variety, where data is divided into measurable business facts and dimensions, which characterize the facts and contain hierarchies [Kimball 1996]. Among the dimensional hierarchies, there are several analytical operations such as drill down, roll up, slide and dice, pivot (rotate), etc conduct complex analytical queries on aggregation measures (which are calculated from the fact measures).

Analysis and decision making usually requires the integration of data from across multiple subject areas and systems, such as customers, product usage and billing. It may also require the inclusion of data from sources outside the company's applications, such as demographic data. This is very different from operational systems, where all the data needed for an application is collected and used within that application. Performance issues are also different between operational systems and analytical systems. Operational transactions place a fairly consistent burden on computing resources. Querying functions, however, tend to be unpredictable, with alternating periods of intense activity and minimal use. By providing two distinct environments, one for operations and another for analysis, an organization can tailor each environment to meet the specific needs. In fact, there are four main reasons for this separation:

1. **Performance**: The peaks and valleys of requests degrade performance of the OLTP system.
2. **Data access**: Organizations often maintain multiple databases, which serve different OLTP functions. The data warehouse, being an integration of all of the enterprise's data, combines all these data sources, and adds external data sources. The data for EIS, DSS, and other decision-support applications use these multiple sources and archived historical data. The typical warehouse user doesn't care where the data are stored. They need and want access to data irrespective of which OLTP system has them.
3. **Data formats**: Because the data in the warehouse are integrated, information is kept in a single, standard format.
4. **Data quality**: The data in the warehouse are "clean" and validated, on its consistency and plausibility, and properly aggregated. They have been reviewed to make sure that only one value is stored in the warehouse for each data item. The warehouse provides a "single version of the truth." Thus, when people use the warehouse, they spend their time trying to understand what the data means, not arguing about what the correct values of the data are.

With the aim of speeding up analytical queries and not negatively effecting of the operational system's performance, the Data Warehouse is implemented separately and redundantly to the operational data sources. Batch and transaction processing data first has to be extracted from operational data sources and then cleaned up to remove redundant data, fill in blank and missing fields and organized into consistent formats. The data is then loaded into a Warehouse. Business analysts can then dig into the Warehouse data using data access and reporting software including On-Line Analytical Processing (OLAP) tools, statistical modeling tools, geographic information systems (GIS) and data mining tools. Building a data warehouse thus includes a set of multiple related processes, and is not just simply a deployed software.

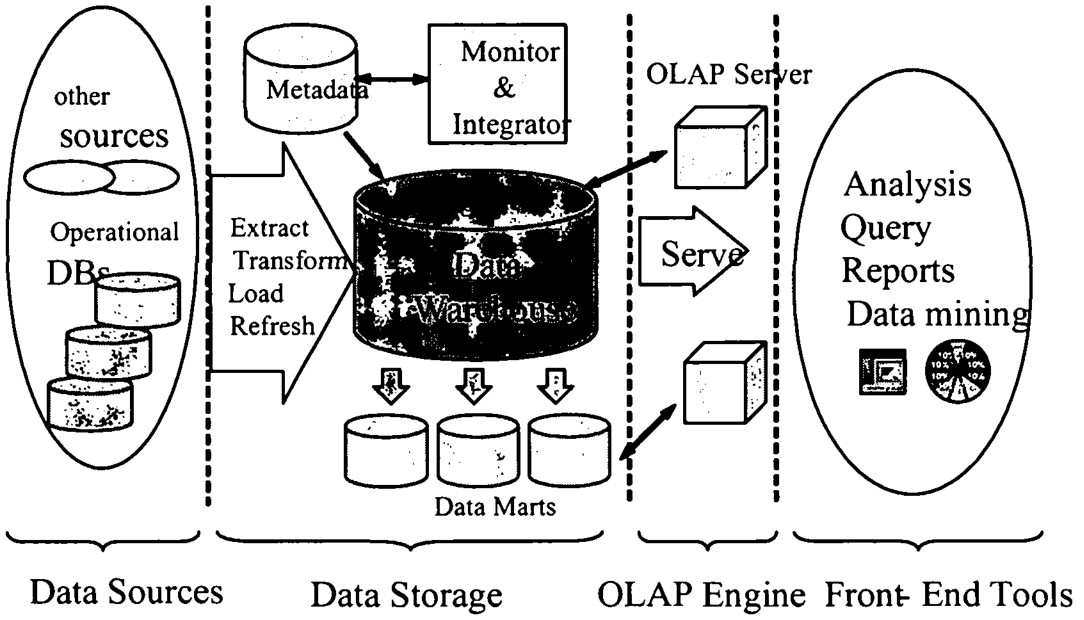


Figure 2.2 Classical multi-tier Data Warehouse Architecture

In the most popular practical multi-tier Warehouse architecture (Figure 2.2) [Chaudhuri 1997], the Data Warehouse is a comprehensive system that includes all the processes, tools, and technologies necessary for extracting relevant data from the sources to managing the data in the warehouse to providing end-user decision-support facilities. Data from multiple operational databases and heterogeneous external sources are extracted, cleaned, transformed and integrated into a common format and representation before being loaded into the Data Warehouse. This process is extremely costly, time-consuming and usually conducted by the Extract-Transform-Load (ETL) tools in batch mode. Furthermore, some ETL tools support also the process of periodically refreshing the Warehouse to reflect data updates at the sources and to purge data from the Warehouse (e.g. onto slower archival storage).

In the storage tier, Warehouse data is stored and managed by one or more Warehouse servers. Besides the main Data Warehouse, there may be several departmental data marts which contain the frequently accessed or highly aggregated subsets of the Data Warehouse. The Monitor and Integrator component performs all operations associated with the management of the data in the Warehouse including the enforcement of the consistency of the data in the warehouse, the creation of views and indexes, and the generation of aggregations. Data in the main Data Warehouse (and the data marts) present multidimensional views of data to a variety of front end tools via the OLAP Engine. OLAP systems have typically been implemented using two technologies: ROLAP (Relational OLAP), where data is stored in an RDBMS, and MOLAP (Multidimensional OLAP), where a dedicated multidimensional DBMS (MDDBMS) is used. HOLAP (Hybrid OLAP) is a mixture of ROLAP and MOLAP storage [Chaudhuri 1997].

The data analysis uses the Frond-End tools (query tools, report writers, analysis tools, and data mining tools) to access the multidimensional view of warehouse data, conduct analytical queries for the decision making support. There is a variety of Frond-End tools supported by different the Vendors, provide powerful analytical query features.

Most of the components in the architecture are described by the metadata stored in the Repository. The metadata describes does not describe only the data structures in the warehouse but also a variety of processes including the extraction and loading processes, the warehouse management, and the processing and execution of queries. Since many of the processes are very dynamic in order to respond to the constantly changing business needs, the metadata must be constantly updated with these changes.

2.3. Time Latency in Data Warehousing and Business Intelligence

According to Hackathorn [Hackathorn 2004] the additional business value of an action decreases with the amount of time elapses from the occurrence of the event to taking action. However, the data about that transaction is stored within the warehouse environment only after some time-windows. Afterwards, the data is analyzed, packaged, and delivered to the user-application. This process also took time to be accomplished. Only after a time-window, the decision based on these analysis results and the relevant action can be performed. Therefore, Data Warehousing and Business Intelligence have to be enhanced towards a closed loop real-time Business Intelligence which shortens the period of time between the occurrence of a business event that requires an appropriate action by the organization and the time the action is finally carried out.

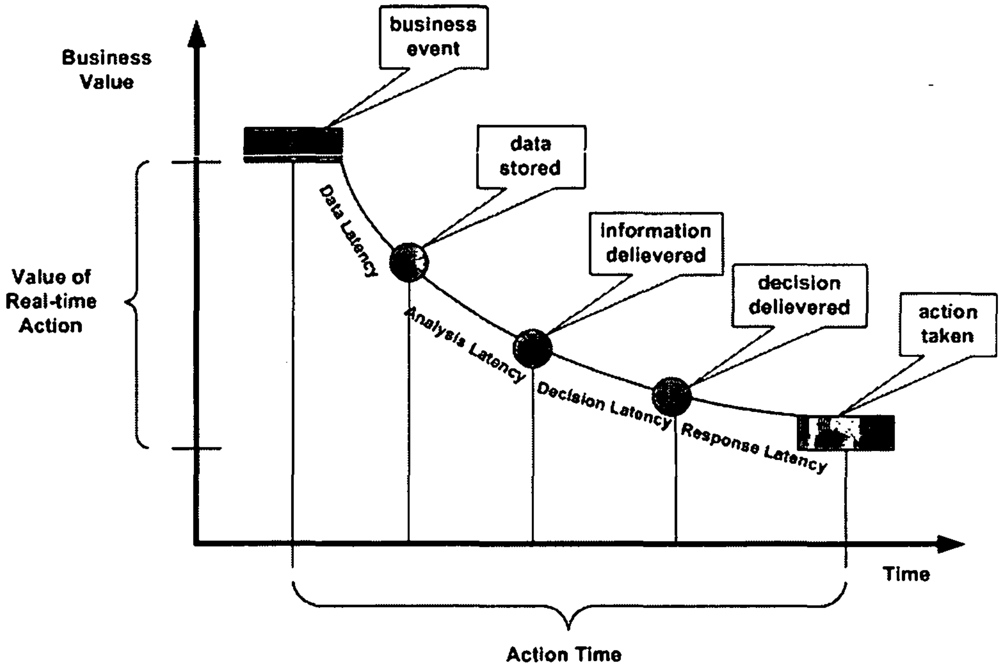


Figure 2.3. Business value and action time

The end-to-end time (elapsed time) required to respond by taking action in response to the business transaction in an intelligent manner is called action time and can be regarded as the latency of an action. Action time comprises four components:

1. **Data latency** is the time from the occurrence of the business event until the data is stored and ready for analysis.
2. The time from the point when data is available for analysis to the time when information is generated out of it is called **analysis latency**; it includes the time to determine root causes of business situations.
3. **Decision latency** is the time it takes from the delivery of the information to selecting a strategy in order to change the business environment. This type of latency mostly depends on the time the decision makers need to decide on the most appropriate actions for a response to the business environment.
4. **Response latency** is the time needed to take an action based on the decision made and to monitor its outcome. This includes communicating the decision made as a command or suggestion, or executing a business action in a target system.

It is important to realize that only the decision latency really counts on the bottom line. The first two are overhead, simply infrastructure necessary for the third, and the forth is just a final fire. Technology advances are greatly reducing data and analysis latencies. However, decision latency will increasingly become the limiting factor.

2.4. Information Evolution in Data Warehousing: From Report to Zero-Latency Data Warehouse

Most successful data warehouse implementations deliver business value on an iterative and continuous basis. Each iteration builds upon its predecessor to increase the business value proposition for information delivery. In recent years, the evolution of Data Warehousing has reached a new pinnacle with the deployment of decision support capability throughout an organization and even beyond its conventional boundaries to partners and customers [Brobst 2002a].

In the early days, data warehouses focused on strategic decision-making for knowledge workers in the corporate "ivory tower". Access to information improved the quality of their decision-making. However, developing a great strategy without efficient execution will get you nowhere [Brobst 2002b]. The emerging generation of data warehouses deployments imposes an ever-increasing set of service levels upon the data warehouse architect. In this section, we review the five-stage evolution maturation model of active Data Warehousing discussed by Stephen Brosbt [Brobst, Rarey 2001] and extend the model to the sixth stage for Zero-Latency Data Warehouse.

Stage	Stage Name	Action	Question
1	Reporting	Observe	What happened?
2	Analysis	Understand	Why did it happen?
3	Prediction	Predict	What will happen?
4	Operational Data Warehouse	React	What is happening? What should we do now?
5	Active Data Warehouse	Reorganize	How can we do it better? What do I want to happen?
6	Minimized Latency	Monitor, Automate & Control	What's my feedback? What is the effect of our reactions and reorganizations?

Table 2.1: Information Evolution Stages in Data Warehousing

Table 2.1 gives an overview of the evaluation model. Traditional Data Warehousing satisfies only the first three stages (Figure 2.4), which deliver just basic benefits. This type of business intelligence effort does not provide the return on investment many organizations expect.

In contrast, stages 4 and 5 are the prerequisites for a zero-latency data warehouse environment. They enable us to react to the current business demands, threats and opportunities based on our observations, understanding and predictions. Furthermore, it closes the loop on information integration and automated information delivery to knowledge workers in order to control the relationship between decision-making and the effects (feedback) of decisions (see Figure 2.5).

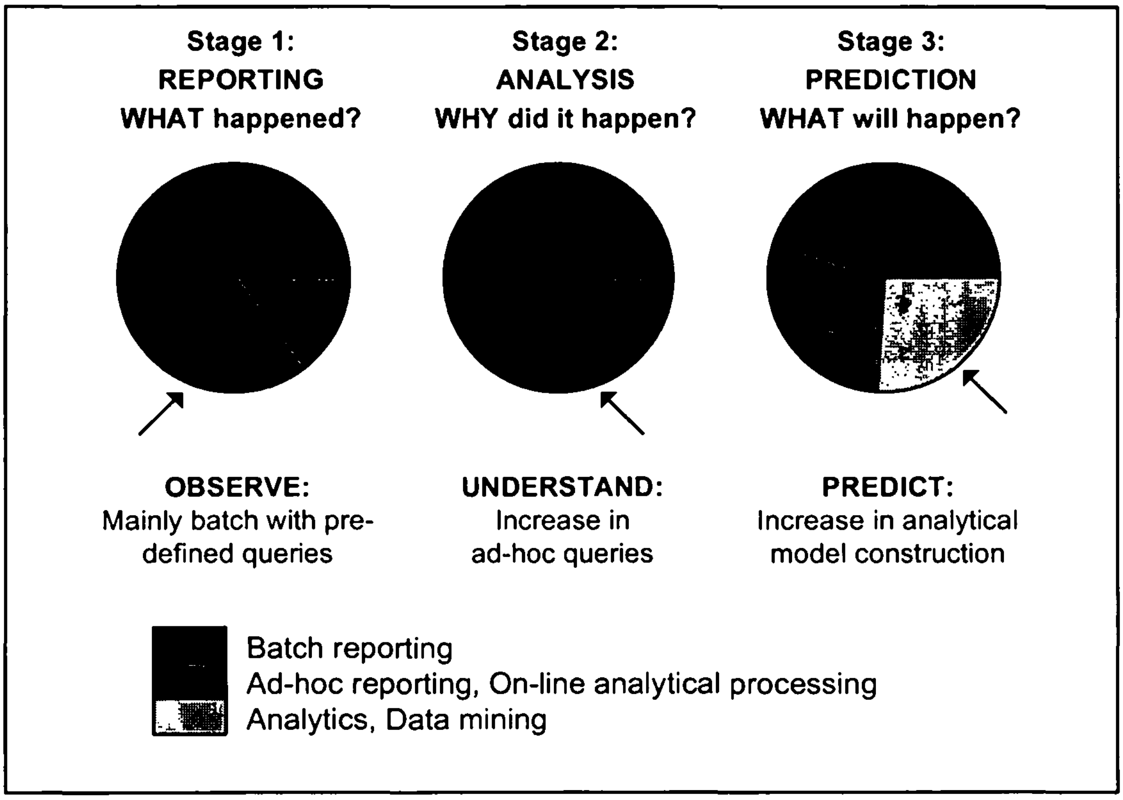


Figure 2.4: Data Warehouse Evolution, Stages 1–3

Stage 1: Reporting

The initial stage of data warehouse deployments and architectures typically focuses on reporting from a single source of truth within an organization. The main effort of stage 1 goes into the integration of disparate sources of information within an organization. The data warehouse brings huge value simply by driving decision-making across functional and/or product boundaries. The challenge is the construction of a repository with consistent and cleansed data [English 1999]. However, setting up the initial loads of the data warehouse does not do the work, because the source systems are evolving information systems that could change significantly in their structure (e.g. database schema, data types) and their semantics.

The questions in a reporting environment are usually known in advance. This kind of decision support is characterized by a large quantity of predefined queries. They are set up by the technologists after receiving requests from the business users or are predisposed managerial methods of communicating what needs to be known after a period or a process is completed.

Thus, it is possible to optimize database structures in order to deliver good performance even when queries access and process huge amounts of data. Data integration is typically done by frequent high-performance batch loads (i.e. monthly, weekly). During these "update windows" the data warehouse is not available for reporting.

Reporting applications provide some answers found in many business databases, but Data Warehousing provides new views and an ability to use combined cross-organizational detailed data to understand the past.

Sophisticated reporting systems imply a summary of observations that highlight exceptions, such as abnormal situations, and suggest actions to correct those abnormalities. Hence, it involves more than the first stage. It requires understanding business dynamics (stage 2) and the prediction of future events (stage 3).

Stage 2: Analysis

At this stage decision makers focus less on what happened but more on why it happened. In order to understand recent changes of numbers and to find out influencing factors, analysis activities are concerned with drilling down beneath the numbers on a report to slice and dice data at a detailed level.

Ad hoc analysis is very important at this stage. Different from predefined reports (stage 1), the queries against the database cannot be known in advance. Furthermore, performance becomes even more important because the data warehouse is used interactively. Whereas reports are typically scheduled to run on a regular basis with business calendars as a driver for timing (e.g. end of month), ad hoc analysis is a hands-on activity with interactive refinement of questions in an interactive environment.

Support for concurrent query execution and large numbers of users against the data warehouse is typical of this type of data warehouse. On-line analytical processing (OLAP) environments [Codd 1993] are built to support more stringent response time requirements (e.g. drill-down to more detailed information within a few seconds). In a typical on-line analytical application, a query aggregates a numeric measure at higher levels in the dimension hierarchy. Performance management relies a lot more on advanced query optimizers. Their ability to determine efficient access paths, using indexing and sophisticated join techniques [Salzberg et al. 1999], plays a critical role in allowing flexible access to information within acceptable response times.

The business users discover trends and patterns not readily apparent from the straight reporting found earlier at stage 1. So we also see that awareness and allowing people to ask questions drive a new desire to utilize the data warehouse infrastructure.

Stage 3: Prediction

As an organization improves in quantitative decision-making techniques and experiences the value proposition for understanding the "whats" (stage 1) and "whys" (stage 2) of its business dynamics, the next step is to leverage information for predictive purposes.

Understanding what will happen next has huge implications for proactively managing an organization's strategy. Data mining tools provide such advanced predictive and analytical functionality by identifying distribution patterns and characteristic behavior within a dataset using historical detail [Chaudhuri et al. 2001]. Knowledge discovery is the process of specifying and achieving a goal through iterative data mining. Typically it consists of three phases: (1) data preparation, (2) model building and evaluation, and (3) model deployment.

However, you need human experts for building, understanding and interpreting the predictive models. Model construction typically involves derivation of many complex metrics for thousands (or more) observations as the basis for training the predictive algorithms for a specific set of business objectives. Scoring is frequently applied against a larger set (millions) of observations because the full population is scored rather than the smaller training sets used in model construction. Advanced data mining methods often employ complex mathematical and statistical functions to obtain the predictive characteristics desired.

The number of users ("data miners") tends to be a relatively small group of very sophisticated analysts. However, the data warehouse workloads associated with model construction and scoring are intense. This heavy resource utilization is due to the complexity of data access and the volume of data handled in a typical data mining environment.

Stage 4: Operational Data Warehouse

As business intelligence applications have matured, it has become apparent that the information and analyses they provide are vital to tactical day-to-day decision-making, and many organizations can no longer operate their businesses effectively without them.

Data freshness service levels are more stringent and require continuous update architectures.

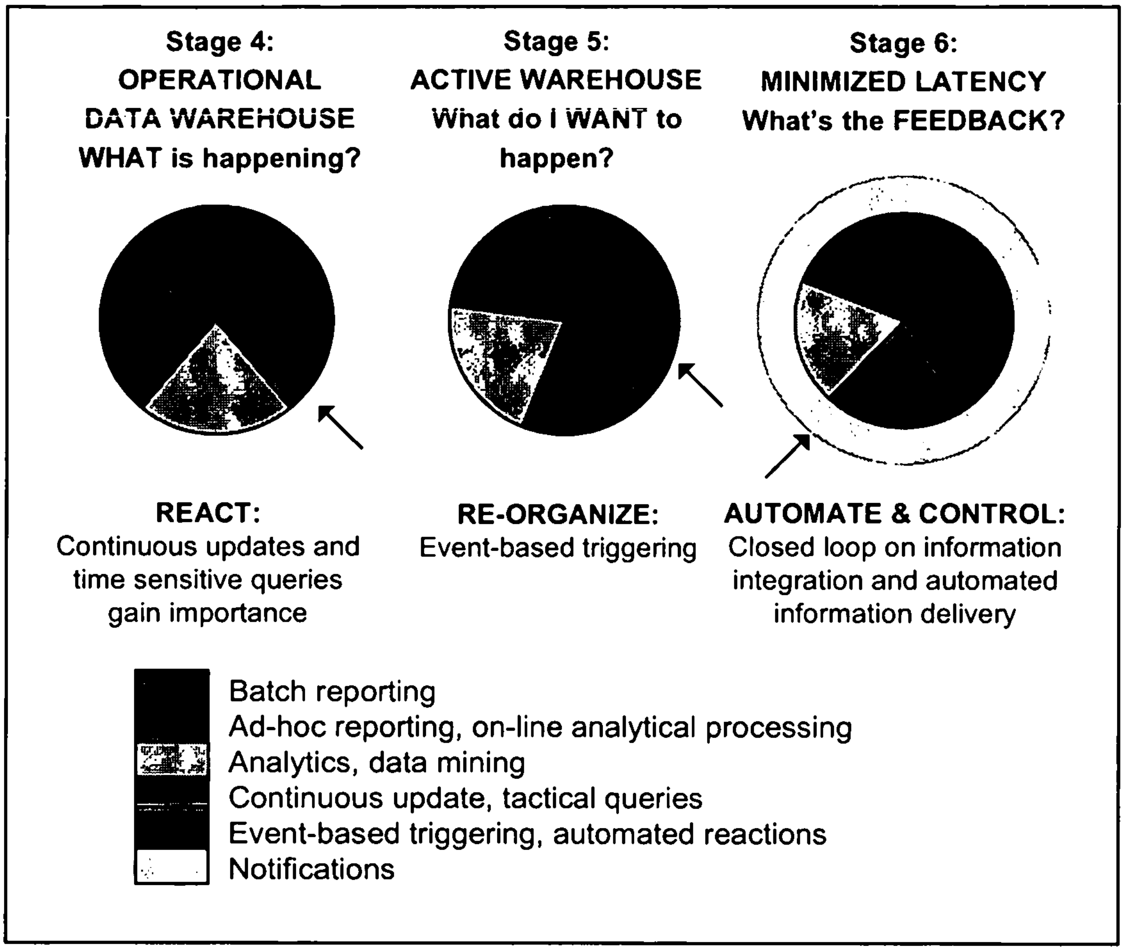

Stage 4:
OPERATIONAL
DATA WAREHOUSE
WHAT is happening?
Stage 5:
ACTIVE WAREHOUSE
What do I WANT to happen?
Stage 6:
MINIMIZED LATENCY
What's the FEEDBACK?
REACT:
Continuous updates and time sensitive queries gain importance
RE-ORGANIZE:
Event-based triggering
AUTOMATE & CONTROL:
Closed loop on information integration and automated information delivery
Batch reporting
Ad-hoc reporting, on-line analytical processing
Analytics, data mining
Continuous update, tactical queries
Event-based triggering, automated reactions
Notifications

Figure 2.5: Data Warehouse Evolution, Stages 4–6

In a traditional data warehouse environment, timeliness requirements (the relative availability of data to support a given process within the time frame required to perform the process) were postponed to mid-term or long-term. An operational data warehouse aims to decrease the time it takes to integrate particular (time-critical) data and to make the information available to knowledge workers. This enables them to find out what is currently happening (based on continuous data integration), decide on what should be done, and therefore react faster to typical and abnormal data conditions (tactical decision support).

Tactical decisions differ from strategic decision in several ways:

- Smaller scope (one product instead of many, one customer instead of all or one transaction instead of millions)
- More stringent response-time requirements

- Typically high data freshness requirements
- Larger number of tactical decision

Whereas stages 1 to 3 focus on strategic decision-making within an organization, stage 4 focuses on tactical decision-making, supporting the people in the field who execute the organization's strategy. Examples include (1) inventory management with just-in-time replenishment, (2) scheduling and routing package delivery, and (3) prioritizing passengers for lost connecting flights. Many retailers are moving toward vendor-managed inventory (e.g. Wal-Mart [Westerman 2000]), with a retail chain that is supplied by manufacturers working as partners. The goal is to reduce inventory costs through more efficient supply chain management. In order for the partnership to be successful, access to information regarding sales, promotions, inventory, etc. must be provided to the vendor at a detailed level. To be useful, the information must be up-to-date and query response times must be fast. This means continuous data acquisition of particular data into the data warehouse. Whereas a strategic decision support environment can use data that is loaded once per month, once per week or even daily, this lack of data freshness is unacceptable for tactical decision support.

Stage 5: Active Data Warehouse

Both for efficiency reasons and for consistency in decision-making, an organization will want to *automate* decisions if humans do not add significant value. In the case of "*routine decision tasks*", analysts extract information from the data warehouse to solve well-structured problems by applying generally accepted procedures in decision-making [Thalhammer 2001]. The automation of such decision tasks is the goal of an active data warehouse.

In e-commerce business models there is often no choice but to automate decision-making when a customer interacts with a web site. In retail industry seasonal items in stores where inventories are higher than they ought to be will automatically be identified. The active data warehouse can initiate sophisticated markdown strategies to drive maximum sell-through with minimum margin erosion [Thalhammer et al. 2001]. Moreover, event-based triggering allows these decisions to be made in an optimal fashion on an item-by-item and store-by-store basis supported by sophisticated decision support technologies. This enables faster reorganizations in response to changing environments.

The active data warehouse has a mixed workload environment involving both complex and simple decision-making queries. Data freshness service levels require continuous update architectures. The ratio of reads to writes in the workload mix is much higher in an active data warehouse than for an OLTP workload.

Stage 6: Minimized Latency

As technology evolves, more and more decisions become executed with event-driven triggers to initiate (fully) automated decision processes. However, the power of event-based triggering is of no value without some control of the effects of decisions in time:

- Minimize latency between the cause and the effect of a decision.
- Notify knowledge workers of actionable recommendations.
- Effectively close the gap between business intelligence systems and business processes.
- Decrease the time it takes to make the decision.

The final stage of our data warehouse evolution model proposes a *notification* concept. Notifications happen in a universal context. Minimized latency implies the continuous monitoring of business events, predicting situations that will become adverse, and suggesting actions to avoid the undesirable. It identifies those who should perform the actions and decides how to notify them and how to control and measure the feedback.

Therefore, a notification does not necessarily imply an operational decision – but it informs decision makers that something very important is occurring, which needs to be investigated in more detail. Notifications are appropriate where it is not possible to fully automate decision tasks because of their complexity or the need for human supervision and decision-making.

Notifications are *not restricted* to the analytical environment. In order to minimize latency we use notifications and event handling twofold: in the active data warehouse component and in the continuous data integration process. If we want to effectively minimize the latency between the cause and the effect of a decision, there are situations where we cannot wait until the response (i.e. data extracted from an operational system) has been integrated into the data warehouse and an event in the analytical environment has triggered some action. We need events during data integration, which further minimize feedback latency of decisions causing operational actions. Furthermore, this approach decreases the load of the data warehouse, because part of the event handling (probably causing notifications or even operational actions) is done outside the data warehouse database.

This enables activated business processes by embedding analytics directly into business processes with little or no human intervention. Activated processes incorporate the full capabilities of an active data warehouse (stage 5) and support improved controlling techniques based on minimized latency for the measurement and propagation of delays into the data warehouse [Bruckner 2002].

2.5. Zero-Latency Data Warehouse

In our vision, the *Zero-Latency Data Warehouse* (ZLDWH) is a data warehouse, which enables a complete business intelligence process to observe, understand, predict, react to, reorganize, monitor, automate and control feedback loops in the minimal latency.

A ZLDWH aims to decrease the time it takes to make the business decisions. In fact, there should be ideally a minimal almost zero-latency between the cause and effect of a business decision. It enables analysis across corporate data sources and notifies the business of actionable recommendations, effectively closing the gap between business intelligence systems and business processes. Business requirements may appear to be different across the various industries, but the underlying requirements are similar – information that is integrated, current, detailed, and immediately accessible.

Transforming a standard data warehouse using batch loading during update windows (where analytical access is not allowed) to an analytical environment providing current information involves various issues to be addressed in order to enable (near) real-time dissemination of new information across an organization. The goal of a so-called *zero-latency* data warehouse is to allow organizations to deliver *relevant* information as fast as possible to decision systems or knowledge workers who need it to react in near real-time to new information captured by an organization's computer system. Therefore, it supports an immediate discovery of data conditions in a manner not supported by an on-line transaction processing (OLTP) system. As a result, deployment of data warehouse solutions for operational, tactical decision-making is becoming an increasingly important use of information assets. A variety of architectural frameworks – such as *Active Data Warehousing* [Brobst 2002b], the *Corporate Information Factory* [Inmon et al. 2001], and *Zero-Latency Enterprises* [Gartner 1998] – have emerged in recognition of the importance of tactical decision support as an extension of traditional data warehouse capabilities.

While operational systems are designed to meet well-specified (short) response time requirements, the focus of data warehouses is generally the strategic analysis of data integrated from heterogeneous source systems. Because the operational source systems always collect data from real-world events captured by computer systems, the data warehouse needs to be updated periodically to reflect those source data changes. Such data integration process can be very complex, covering the acquisition, extraction, transformation (staging area), and loading of the data into the data warehouse. Data loading is done during frequent update windows (e.g. once a week, every night), in this way the analysis capabilities of data warehouses are not affected. The "observation" of these real-world events is characterized by a delay. This so-called propagation delay is the time interval it takes for a monitoring (operational) system to realize that a state change has occurred. Real-world changes are generally discovered by computer systems with a certain delay. The typical update patterns for traditional data warehouses on weekly or even overnight basis increase the propagation delay until the information is available to knowledge workers.

Consequently, up till recently, *timeliness* requirements (the relative availability of data to support a given process within the timetable required to perform the process) have been restricted to mid-term or long-term [English 1999]. W. H. Inmon, known as the founder of Data Warehousing, cites *time variance* as one of four silent characteristics of a data warehouse. Timeliness can be viewed as an aspect of data quality, which has a strong influence on the time it takes until a system realizes a certain state of data. While a complete, real-time enterprise data warehouse might still be a future challenge, some yet feasible approaches may well enable data warehouses to react "just-in-time" [SUGI29 2004] to changing customer needs, supply chain demands, and financial concerns. The *zero-latency* data warehouse takes the goal of timeliness to its logical extreme: **instantaneous awareness** and **appropriate response** to events captured in the data warehouse environment.

Enabling minimized latency for decisions in a data warehouse environment requires near real-time data integration in order to close the feedback loop on decisions. In addition, the capability of handling events even during the data integration process, complements the capabilities of the active data warehouse (analyzing data that has already been integrated) and creates an environment that allows for reactions with minimized latency.

In general, data integration is affected by propagation delays, which need to be addressed by the DWH data model to represent history as accurately as possible. The advantages provided by built-in temporal consistency support in data warehouses include higher-fidelity data modeling, more efficient gathering of the organization's history, as well as analyzing the sequence of changes within that history. This enables the introduction of active behavior for data warehouses, where rule-driven decision engines initiate operational actions for routine decision tasks. The "analysis loop" of these automatic actions is closed by the continuous (near real-time) data integration mentioned above. This leads to *zero-latency Data Warehousing*.

2.6. Summary

The requirements of analysis and reaction at the speed of business transaction are not satisfied by the traditional Data Warehousing and Business Intelligence. For better monitoring and management of business activities, there is a trend towards Zero-Latency Data Warehousing which aims to minimize the reaction time elapsed to make decision. We has introduced in this chapter the concepts and techniques used in the traditional Data Warehouse, its limitation in supporting real time analysis. The Information evaluation in Data Warehousing introduces the sixth stage (Zero-Latency Data Warehouse) with the capability of monitoring, controlling the business event and sending timely notification. The notion of Zero-Latency Data Warehouse and its perspectives are also thoroughly discussed.

Chapter 3: Background and Related Work

> *"Zero Latency is the real-time, enterprise-wide dissemination of new information distributed in such a way that allows businesses to react quickly to it, driving the competitive business advantage to its ultimate limits".*
>
> **Paul Larson, Talarian Corporation, 2002**

In this chapter, we first introduce other research activities related to "Zero-Latency Data Warehousing" and discuss their motivations and requirements. We then present some background concepts and technologies which are later mentioned in the thesis such as Continuous Data Streams, Service Oriented Architecture and Grid computing.

3.1 "Zero-Latency" and "Real-time"

In IT terminology, latency is defined as the time required for a system to respond to an input. A zero-latency strategy is a plan to decrease latency through out the enterprise to the absolute minimum. Since the Gartner Group coined the term "*zero latency*" in 1998 [Gartner 1998], a lot of related terms such as *Zero-Latency Enterprise* [Compaq 1999, HP 2002], *Active Warehousing* [Brobst 2002b], *Real-time Analytics* [Raden 2003], *Real-time Warehousing* [Haisten 2002, Colin 2002 , Simon 2004, Langseth 2004], *Real-time Decision Support* [SUGI29 2004], *Business Activity Monitoring* [BAM 2002], started to appear more often in articles, seminars, or even front pages of Data Warehousing or Business Intelligence related publications in recent years.

3.1.1 Zero-Latency Enterprise

Zero-Latency Enterprise (ZLE), refers to information management technologies that integrate diverse enterprise computer applications and eliminate delays in the propagation of new data throughout the enterprise. The basic idea behind a Zero-Latency Enterprise strategy is to speed up the flow of information and business processes so as to achieve a competitive advantage. It implies that *all* parts of the enterprise can respond to events as soon as they become known to any part of the enterprise.

ZLE technologies provide a kind of central nervous system for the enterprise, allowing individuals and systems to respond to events in real time. Examples of applications that require immediate responses to events include the following:

- Real-time attrition or churn management
- Real-time billing and bill presentment
- Customer relationship management
- Fraud detection, security and crime prevention
- Marketing campaign analysis and tuning
- Sales performance analysis
- Inventory tracking and management
- Financial exposure control

In 1999, Compaq, now part of HP, was the first vendor to convert the ZLE concept to a working architecture, which has been proven in other rapidly changing environments such as retail, telecommunications, and banking. ZLE framework [Compaq1999, HP 2002] is a data-oriented architecture that centers on a real-time ZLE data store which is a new construct, a "hot cache" of data from across the enterprise. Rather than using Enterprise Application Integration (EAI) toolsets to keep applications synchronized, the ZLE framework uses them and other technologies to keep remote applications and their data in sync with the ZLE data store. And it uses the ZLE data store to keep remote applications in the ZLE environment synchronized with one another.

The ZLE core represents a state-of-the-art EAI environment. As shown in figure 3.1, enterprise applications such as SAP, Siebel, and other enterprise systems attach to the core via a variety of application and technology adapters. These adapters convert proprietary messages to the standards used by the ZLE core.

A ZLE data store is a high-performance database platform for managing data from any application in the enterprise. Unlike traditional operational data stores, it maintains state data, which is current value information such as a customer's current account balance, as well as event data, which is detailed, transaction-level data such as call detail records (CDRs), credit card transactions, and so on. In a Zero-Latency Enterprise, the role of the ZLE data store is to keep databases and application data stores synchronized so that an update of any one application also updates applications downstream. The utility of the ZLE data store is that it delivers, at any given time, a single, consolidated view of all pertinent data from all applications in the enterprise. Data is redundant between each application and the ZLE data store, but not between all applications. At the same time, the structure of the data in the ZLE data store is changed so that it is subject-oriented, persistent, and can be updated in real time.

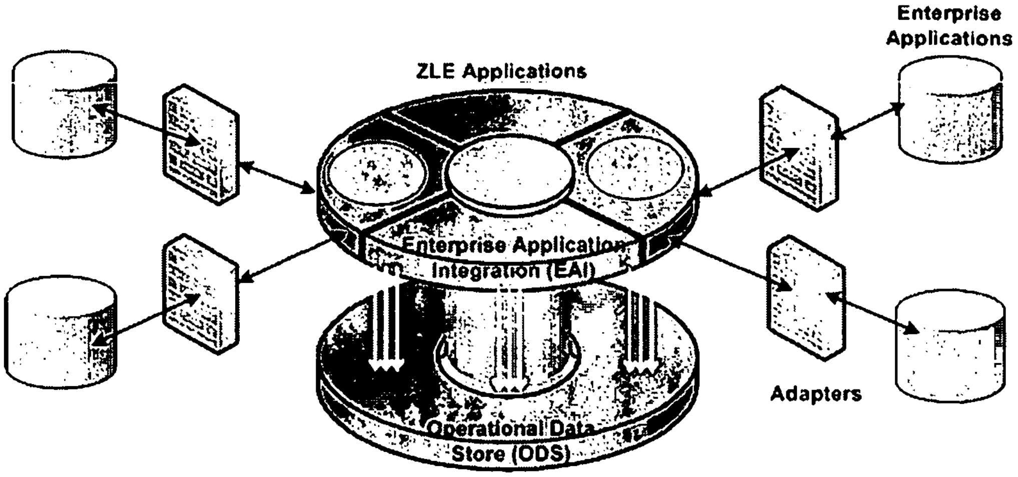

Enterprise Applications
ZLE Applications
Enterprise Application Integration (EAI)
Operational Data Store (ODS)
Adapters

Figure 3.1. The Zero-Latency Enterprise framework

The EAI hub serves as "instant messenger". The role of EAI within the zero latency enterprise is to transmit information between ZLE applications (such as personalization and fraud management) and the ZLE data store. For example, in a telecommunications company, EAI technology sends each new CDR to the ZLE data store in real time. If the business rules determine that the call appears to be fraudulent—for example, it's the second international call made within minutes by a customer who has never made even one— then EAI technology instructs another application to alert a human agent to call the customer. All events happen in real time, so the fraudulent call can be averted as it happen, saving money and pleasing a customer.

3.1.2 Active Data Warehousing

The concept of active Data Warehousing [Thalhammer 2001, Brobst 2002b] has also gained attention. It combines active mechanisms based on (somewhat restricted) ECA (event-condition-action) rules known from active databases [Paton et al. 1999] with the integrated analysis capabilities of data warehouse environments to extend (passive) systems with reactive capabilities: An *active data warehouse* (ADWH) is event-driven, reacts in a timeframe appropriate to business needs, and will make tactical decisions or cause operational actions rather than wait to produce periodic reports [Brobst 2002b].

In the traditional passive DWH approach, the analysts perform interactive analyzes using OLAP tools to find solutions for different kinds of decision tasks. The decision tasks can be classified into three styles [29] (1) "*non-routine decision tasks*", (2) "*routine decision tasks*" and (3) "*semi-routine decision tasks*".

In the case of "*non-routine decision tasks*", analysts query the DWH to create different scenarios and then make the decision. Such tasks do not occur regularly and there are no generally accepted decision criteria. Some decisions in strategic business management such as setting up the new product or changing the business policy are examples of "non-routine decision tasks".

In the case of "*routine decision tasks*", analysts extract information from the DWH to solve well-structured problems by applying generally accepted procedures in decision making. Such decision tasks occur frequently at predictive time points. Examples can be found in areas of customer management (e.g. grant the discount to special customers) product management (e.g. change price or withdraw products).

The "*semi-routine decision tasks*" typically occur when routine problems require non-routine treatment (e.g. if there are contradictory in decision before withdraw a product need to be discussed more before making the decision).

One of the main functions of *Active Data Warehouse* (ADWH) is automatizing decision-support activities of "*routine decision tasks*" and the *routine elements* of "*semi-routine decision tasks*". The autonomous reaction processes are based on complex incremental multi-dimensional analysis of the collected data in the Data Warehouse, which was usually implemented manually by the analyst using OLAP tools. The Active Data Warehouse extends the Passive Warehouse with the *Active Decision Engine* component. The engine combines Data Warehouse, Active Database (ADB), and OLAP tool to automatize decision processes that occur on a regular schedule and for which well-established decision criteria exist. The "glue" among these technologies are the *analysis rules*. They realize the basic ECA rule structure from ADB, but carry out complex OLAP analyzes on warehouse data instead of evaluating simple conditions as in case with ECA rules in OLTP system.

Like ECA rules, the *event part* of an analysis rule represents the time points at which decision processes should be initiated. However, compared to ECA rules in ADB which offers a variety of events [Paton et al. 1999] (e.g. structure operation, behavior invocation, transaction, abstract or user-defined, exception, clock, external), the scope of the event part of analysis rules is limited (e.g. OLTP method event, absolute and relative temporal event).

The *condition part* represents the incremental multidimensional analysis that is usually carried out manually by the analyst using some OLAP tools. In contrast to conventional ECA rules, the condition part in analysis rules is not a simple Boolean predicate but consists of hierarchically organized predicates that require the incrementally performing of multi dimensional analysis to evaluate.

The *action part* of an analysis rule is the directive to execute a particular transaction in the OLTP system, representing the decision for which analyzes have been carried out. The action could also be a recommendation or notification to the users for further analysis in the case it can not make the decision positively. Compare to the action part of ECA rules in ADB [Paton et al. 1999], which maybe an arbitrary sequence of DML statements update the database or rule set, transaction statements, behavior invocation, external calls, do-instead, etc., we can stay that the actions which analysis rules can cope are more simple.

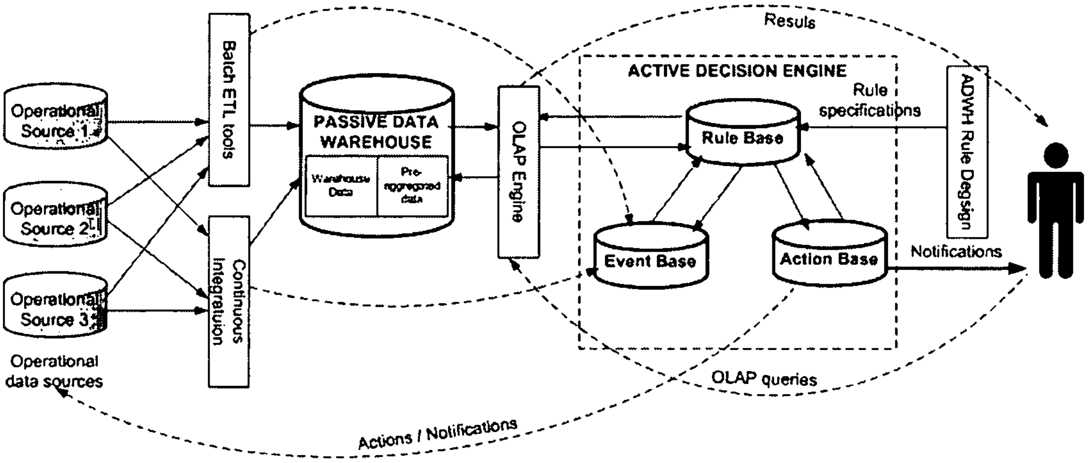


Figure 3.2. Active Data Warehouse Architecture

The Active Data Warehouse architecture is shown in Figure 3.2. Data is extracted, loaded, transformed and integrated in the DWH through the batch ETL or the continuous data integration [Bruckner 2002]. Besides loading conventional passive warehouse data, changes in the operational data sources (OLTP events) are also loaded into the *data event repository* (the event base in figure 3.2). These events are used to trigger of other complex types of events (e.g. temporal events).

Within the *active decision engine*, the temporal events (e.g. end of month) will be used to determine when analysis rules should be fired. The passive DWH, which maintains integrated data and pre-aggregated data (e.g. materialized views) is responsible to answer two types of requests: (1) The ad-hoc OLAP request issued by analysts using conventional OLAP tool, (2) Pre-defined OLAP requests issued by analysis rules when these rules are executed.

The active decision engine includes three basic components. The *Rule base* stores executable rule definition and meta data about these rules. The *Event base* maintains occurrences of events that fire the analysis rules. The *Action base* maintains all information about the actions can be happened (e.g. specifications of the action, parameter binding, etc.). The ADWH rules-design-tool provides a graphical user interface with which the users can define new rules, update existing rules, inspect generated decisions and modify the event base. Analysis rules are modeled from the perspective of the analyst who specifies: (1) the time points at which the analysis rule has to be fired (*the event*); (2) the data cubes that have to be inspected (*the analysis graph*) by using certain criteria for decision making (*the decision steps*); and (3) the transaction that has to be executed for a particular entity in an OLTP system if one of the decision criteria is satisfied (*the action*). The data cubes in the analysis graph will be analyzed to evaluate some criteria defined by the Rule Base, and if certain decision criteria are satisfied, the action (in the Action Base) will be realized without user interaction. As concerned in [Thalhammer et al. 2001], analysis rules can be realized by cascading triggers, which are available in most of commercial database system nowadays.

3.1.3 Real-time Data Warehousing

Traditionally data warehouses do not contain current data. They are usually loaded with data from operational systems at most weekly or in some cases nightly, but are in any case a window on the past. The fast pace of business today is quickly making these historical systems less valuable to the issues facing managers and government officials in the real world.

Increased data volumes and accelerating update speeds are fundamentally changing the role of the data warehouse in modern businesses. More data, coming in faster, and requiring immediate conversion into decisions means that organizations are confronting the need for real-time Data Warehousing. This is due to the followings:

- **Decision support has become operational.** The distinction between transactional systems — which run the business day to day — and decision support — which traditionally have engaged business intelligence issues around product, customer, and market trends — remains a valid distinction. However, a related trend, which has not been sufficiently noted in either the press or by industry analysts, is that the distinction between decision support and operational has broken down.

- **Integrated business intelligence requires closed-loop analytics.** First-generation data warehouses fed data downstream from transactional systems to the data warehouse to perform market trend analysis at a high level. Second-generation warehouses drilled down in detail to enable advanced applications. Third-generation warehouses close the loop back from the data warehouse to the transactional system, using the data warehouse information to optimize the transactional system. With third-generation warehouses, the latency must be low on a par with that expected by transactional processing.

- **Yesterday's operational data stores (ODSes) won't support today's requirements.** The Data Warehousing hardware and database infrastructure of the early 1990s did not support the data volumes, response-time requirements, and data integration needs of the single view of the data warehouses of the day. So the ODS was born to provide architectural support for the data warehouse. However, ODSes introduce redundant copies of detail data, which complicates data consumption and analysis.

According to the Data Warehousing Institute-Forrester Quarterly Technology Survey of 1/2004 [Agosta et al. 2004], during the past 18 months, Data Warehousing refresh rates have accelerated measurably. Respondents performing the "frequent update" to their data warehouse (at least multiple times a day) have increased from 2% to 9% during the past 18 months. While a daily update frequency is still the most common refresh rate, the number of frequent updaters will grow to 22% during the next 18 months (see Figure 3.3). Clearly, the pressure to squeeze latency out of Data Warehousing systems is a priority at a growing number of enterprises.

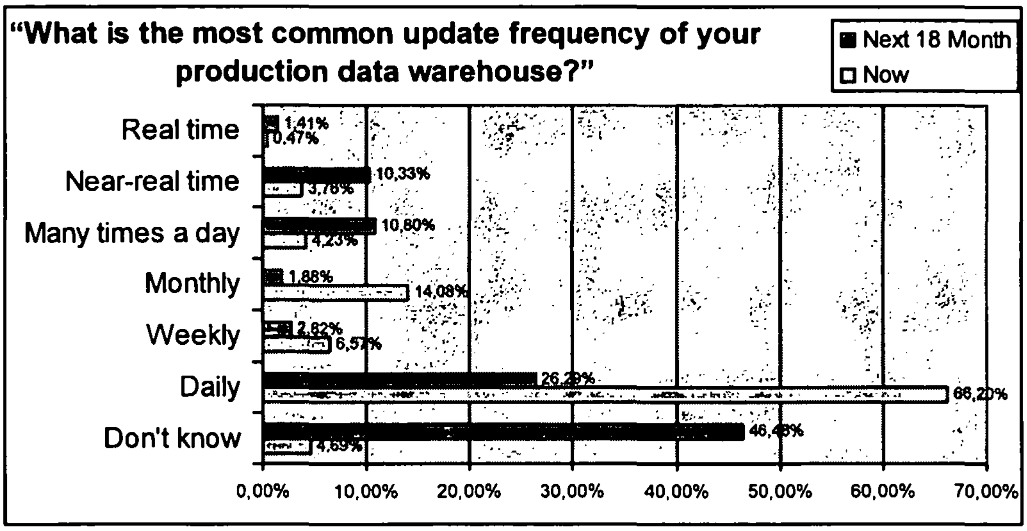


Figure 3.3. Data Warehouse Updates are Accelerating

Real-time Data Warehousing (RTDW) refers to the technical aspects of a data warehouse that updates as data is presented to it. A strict definition of real-time implies that any data change occurring in a source system is automatically and instantaneously reflected in the data warehouse. This would mean that all changes in the Data Warehousing environment take place simultaneously with the change in the source system – something that is only achievable when both changes are part of the same atomic transaction. RTDW concepts include physical modifications to the database schema and the database environment, movement of data across the enterprise, ETL processes, modification of downstream processes, especially alerts, creation of extracts, cubes and data marts, and the whole new methodology for designing and implementing RTDWs [Simon 2004, Agosta et al. 2004].

However, Real-time Data Warehousing is not right for every case, and can introduce new, unnecessary headaches if not managed carefully. There are some other options which significantly constrain (slow but not stop) the forward motion of real-time Data Warehousing:

- **On time.** For those enterprises without a formal service-level agreement (SLA), on time means data is updated and delivered according to whatever consensus or implied agreement exists between the IT function and the end user organization, that is, by agreement or consensus.′
- **Simulated.** Simulated real time means that update is performed in batch, but that access is performed interactively, giving the appearance of an answer "right now," even though the timeliness (currency) of the data is stale. For example, an end user is sitting at a workstation and launching inquiries or what-if analysis against the data warehouse, receiving a response from the system within human think time of between two seconds and two minutes. Even if the data warehouse was updated overnight by a batch process, the data access is interactive. Here the access is in real time, but not the update, which remains a background process.
- **Right time.** This was the catch-all phrase prior to the decomposition into on time and simulated. It means near-real time and is tied to a specific technology such as change data capture (CDC) to a log and subsequent distribution by means of an ETL tool or store-and-forward mechanism such as a message broker.
- **Real time.** This means a process in which resources such as databases, networks, and CPUs are locked synchronously until a commit point is reached, at which point other concurrent processing may proceed. The data in the system reflects the state of the business on literally a moment-by-moment basis.

Type	Definition	How it works	Example
On time	Data is updated and delivered according to policies, service-level agreement, or consensus.	Business groups tell IT how often they need to update and access data, and IT delivers data on that schedule.	Inventory

Simulated	An end user at a work station executing self-service query and reporting or what-if analysis. Updates and roll-up calculations are performed in batch, delivered in interactive "think time."	The results have been pre-computed and stored in the data warehouse for latter delivery as if the calculation were done in real time, but it is not.	Customer recommendation
Right time	A catch-all phrase meaning near-real time — tied to a specific technology such as change data capture to a database log.	Allows for a variety for response times, none committing to synchronous processing (see real time) — allows for distribution by an ETL tool or message broker.	Web log analysis
Real time	The answer is absolutely the most up-to-date information physically possible in terms of both update and access.	Resources such as databases, networks, and CPUs are locked synchronously until a commit point is reached, at which time other concurrent processing may proceed.	Fraud detection

Table 3.1. The Many Meanings of Real Time

(source: Forrester Research, Inc.) [Agosta et al. 2004]

Table 3.1 describes in more details the time horizon of the variety of meanings of Real-time Data Warehousing.

3.1.4 Real-time Analytics

Real-time analytics [Raden 2003] is the ability to use all of the resources in an organization, especially data, to improve the operations and quality of service, at the moment they are called for. If, at the moment (or very soon after) a piece of information is created or modified in an operational system, it is sensed and acted upon by an analytical process, real-time analytics have occurred. Real-time analytics complement real-time operational systems and real time Data Warehousing. Agile organizations will need to measure, evaluate and react to events with a closed-loop of telemetry-like information, rules, decisions and triggers, all in real-time.

Similar to Real-time Data Warehousing, Real-time Analytics is not for everyone. There are some applications, where the ability to include up-to-the-second analytics is extremely useful (see Table 3.2). What is important to understand is that the value in merging operational and analytical processes in real-time has to be found in the business processes that leverage it.

Industry/Audience	Type of Analytics
Financial Services	Credit card fraud Risk Management (Market, credit, ops)
Energy	Forward price projection Dynamic triggering
Retail	Pricing optimization, replenishment, markdown and inventory management
Government	CDC monitors Rx for epidemic outbreak; Anti-terrorism port traffic
Consumer Products	Promotional effectiveness Spot promotions
Web Commerce B2B or B2C	Targeted incentives at touch point Collaborative filtering/customization
Finance	Instant financial close Continuous planning
Executive	Real-time dashboards Business Activity Monitoring (BAM)
Healthcare	Service level optimization

Table 3.2: Real-time Analytics by Sample Industry or Audience

Generally, real-time analytics are suitable for time-sensitive decisions that also have one or more of the following characteristics:

- High risk or high cost (a customer could be immediately lost or gained).
- Numerous, often-conflicting constraints (the revenue potential must be quickly weighed against the cost of obtaining it).
- Potential for optimization through injection of context data (breadth, depth, history to enable a more comprehensive decision on-the-spot).

- Significant competitive advantage could result (earlier identification of anomalies or opportunities to more quickly limit exposure or optimize gain).
- Increased business efficiency (consistent, optimal execution of frequent actions which individually seem insignificant but collectively have high value/cost).

3.1.5. Real-time Decision Support

Motivated by the requirement of *"delivering the right information to the right people just in time"*, real time decision support [SUGI29 2004] is a strategy aimed at solving a business problem that can not be solved by the operational systems or the Data Warehousing systems alone. It requires more than just the ETL tool, the database, the business intelligence (BI) portal, but requires an integrated view of how data moves through the organization to provide the most valid and reliable decision-support metrics to the right people, just in time. It is about identifying the source systems, staging and target databases, data acquisition and integration strategies, network and server capacity requirements, messaging systems, development/quality assurance/production environments, and error and recovery processes.

In Real-time Decision Support system, the key components involve an organization's ability to detect events, evaluate and then respond to them much more quickly than traditional Data Warehousing strategies have allowed. By blending operational and strategic systems (Data Warehousing and analytics), a much richer environment for decision support exists. It is important to understand what types of decisions are good candidates for the real time revolution and which business processes can be morphed into economical processes that are inherent in the real time information architecture.

The goal of real time decision support is to facilitate access to information that cuts across multiple applications, systems and data sources. The blending of operational systems that support business-as-usual can combine with historical and analytic results to help provide a superior framework for decision making. Real time decision support thus focuses on both the process events important to operations and the business events that support strategic management. By virtue of its label, real time means much more frequent updates of content. We think of the result as getting the most valid and reliable decision-support metrics to the right people, just in time.

3.1.6 Business Activity Monitoring (BAM)

Failure to identify problems, anticipate opportunities, and respond rapidly is immensely costly. With the broad spread IT through most operations, businesses are striving to achieve "zero-latency enterprise" status, where all data about a business are captured, analyzed and acted upon in real-time. "Business activity monitoring" (BAM) is Gartner's term defining how we can provide real-time access to critical business performance indicators to improve the speed and effectiveness of business operations [Gartner 2002]. Unlike traditional real-time monitoring, BAM draws its information from multiple application systems and other internal and external (inter-enterprise) sources, enabling a broader and richer view of business activities. As such, BAM will be a natural extension of the investments that enterprises are making in application integration. BAM takes business process management beyond the smooth flow of processes across multiple applications to allow analysis and monitoring of business processes and conditions as they happen [BAM 2002]. BAM applies operational business intelligence and application integration technologies to automated processes to continually refine them based on feedback that comes directly from knowledge of operational events. It provides more accurate information about the status and results of various operations, processes, and transactions so we can make better decisions, more quickly address problem areas, and reposition the organization to take full advantage of emerging opportunities.

Live or near-live monitoring of business processes and KPIs (key performance indicators) is precisely the mission of BAM, which typically pulls information from a variety of applications and data sources then presents it to the user as live, integrated, "dashboard" displays filled with speedometers, bar graphs, maps, and tables. BAM also filters the information it monitors based on user-defined business rules, sending out alerts when key thresholds are reached, including inventory running below a certain level, or, in the case of OfficeMax, lack of response from a vendor within a certain time period. In some cases alerts may initiate other actions, such as e-mails to staff, customers, or suppliers.

Features	Comments
1. Real-time event data computation.	This means that the tool can use the data contained in events to compute metrics (i.e., KPIs) in real-time and continuously update them on the graphics displays or feed the metrics to other applications.
2. Single event triggering.	The tool can react to single events, either predefined or defined by the user, e.g., alerts indicating critical KPIs, and then takes actions also predefined or specified by the user. Usually the tool will supply a capability to define event triggered reactive rules, and may

	combine this with capabilities under feature item #1.
3. Event streams processing.	An event stream is a real-time, continuous, ordered (by arrival time at the tool, or by a time stamping mechanism) sequence of events. Here, the tool assumes the event input is a stream. This assumption allows optimization of some kinds of event processing that fall under item #1. It is an appropriate way to handle stock-market feeds, for example.
4. Complex event pattern triggering.	Tools that satisfy this checklist item can detect complex patterns of many events in the global event cloud of the enterprise and then react to them in various ways. Typically, a complex pattern is made up of many events, often created at different locations in an enterprise and possibly in different time zones. Complex patterns may involve events that are causally related and other events that happened independently. A complex pattern consists not only of the events but also of the relationships between the events. For example, the patterns of events in fraudulent activity.
5. Event pattern abstraction	Tools that satisfy item #4 may go a step further by providing rules to create new events whenever an event pattern matches. The new events can be constructed to contain some of the information contained in the events that matched the pattern. And the new events can be processed by the tool just like other events in the cloud. This is called event pattern abstraction. It is a step towards supplying higher level views – e.g., executive reports – of patterns of complicated business activity. Important data is included in the new event, and unnecessary details are omitted, the new event is an abstraction.

Table 3.3: BAM tools checklist (adopted from [Luckham 2005])

Table 3.3 presents the checklist to evaluate the BAM tools, currently, most BAM tools will only check off items #1 and #2. Even so, they're proving to be very useful. New tools are entering the market place that can check off the first four items. If a BAM tool can check off four or five items on this list, it is certainly doing some complex event processing.

3.2 Continuous Data Streams

So far, most of existing data analytical solutions work only with traditional (stored) datasets. However, in the recent years, advances in modern technologies have allowed us to automatically record daily transactions at a rapid rate. Such processes lead to large amounts of transactional data which grow at an "unlimited" rate and are available as continuous data streams. Data streams arise naturally and are used in the various scientific and business application domains. Fields of application include stock ticks in financial applications, manufacturing processes, log records or click-streams in web-personalization, data feeds from sensor applications, network packets and messages in firewall-based security and call detail records in telecommunication and so on [Chen et al., 2000]. Data streams thus become now fundamental to many data processing applications and the need for complex analyses of these high-speed data streams is substantial. Further, the ability to make decisions and discover interesting patterns on-line (i.e., as the data stream arrives) is crucial for several mission-critical tasks that could be decisive for an organization (e.g., telecom fraud detection, stock market monitoring, etc.)

A data stream is defined as a continuous transmission of data elements. Each data element, also called tuple or record, is a collection of data items. Data items may be of different data types. Data elements have a discrete time, which is realized by designated data items containing sequence numbers (stream relative). Additional time stamps have the advantage to allow for time correlation between data streams. These sequence numbers allow for recognition of missing data elements and correct ordering in a stream.

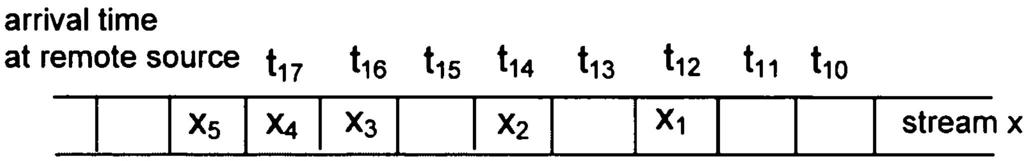


Figure 3.4. Continuous Data Streams

Data streams have some specific characteristics that make them different from traditional store data set. They arrive in a sequence of unbounded, real time data items with a very high data rate that can only read once by an application. ([Strauss et al. 2003], [Golab,Ozsu 2003], [Babcock et al.2002]). They could be "infinite", and once a data element has arrived, it should be processed and either archived or deleted, i.e. only a short history can be stored. It is impossible and impractically to control the order in which items arrive, nor is it feasible to locally store a stream in its entirety because sorting even sub streams of a limited size, is also a blocking operation.

By nature, a stored data set is appropriate when significant portions of the data are queried again and again, and updates are small and / or relatively infrequent. In contrast, a data stream is appropriate when the data is constantly changing (often exclusively through insertions of new elements), and it is either unnecessary or impractical to operate on large portions of the data multiple times.

	Static data set	Continuous data stream
Relationship	Persistent	Transient
Query method	One-time	Continuous
Access method	Random	Sequential
Store character	Passive Repository	Active Storage
Respond Speed	No requirement	Must quickly
Update Speed	Low update	High update
Response	No real-time	Real-time

Table 3.4. Differences between static data set and continuous Data stream

In the recent years, we have witnessed many interest research activities on continuous data streams. Some of the researchers are trying to build an entire data stream management system (DSMS; instead of DBMS - data base management system) from scratch [Widom J. et al 2003]. The alternative approach is to modify or extend an existing DBMS so as to include the envisioned functionality. So far, research results have been reported for modelling and handling data streams including algorithms for data stream processing to fully-fledged data stream systems. The unique characteristics of data streams and continuous queries dictate the following requirements of data stream management systems [Golab,Ozsu 2003]:

- The data model and query semantics must allow order-based and time-based operations (e.g. queries over a five-minute moving window).
- The inability to store a complete stream suggests the use of approximate summary structures, referred to in the literature as synopses [6] or digests [72]. As a result, queries over the summaries may not return exact answers.
- Streaming query plans may not use blocking operators that must consume the entire input before any results are produced.

- Due to performance and storage constraints, backtracking over a data stream is not feasible. On-line stream algorithms are restricted to making only one pass over the data.
- Applications that monitor streams in real-time must react quickly to unusual data values.
- Long-running queries may encounter changes in system conditions throughout their execution lifetimes (e.g. variable stream rates).
- Shared execution of many continuous queries is needed to ensure scalability.

In continuous query processing, several approximation methods are used for data reduction and synopsis construction such as sketches [Dobra et al. 2002], random sampling, histograms [Strauss et al. 2003], and wavelets [Chakrabarti et al. 2001]. Some other approximate methods are applied to tackle the blocking operator such as Sliding Window [Franklin et al. 2002], load shedding [Babcock et al. 2004], punctuation [Tucker et al. 2003]. k-Constraints [Babcock, Olston 2003] are used in clustering and monitoring data stream. [Kim, Park 2005] proposes an efficient periodic summarization method with a flexible storage allocation to store large volumes of streaming data in stable storage. Other research topics cover data stream management system models, architectures and related issues such as memory minimization, operator scheduling, query optimization, multiple query, distributed query processing and so on [Babcock et al. 2002, Widom et al. 2003].

As another approach, conventional OLAP and data mining models have been extended to deal with data streams, such as multi-dimensional analysis (Han et al., 2002), clustering (Motvani et al., 2000) and classification (Hulten et al. 2001). Some other researchers investigate in union/join multiple streams [Gibbons et al. 2001], stream reductions [Kumar et al. 2002].

In general, most of previous approaches on data stream processing focus on approximation methods based on statistical estimations due to the limited storage and computing resources. Recent research activities in data stream toward the use of super computing infrastructure to tackle the bound limitation of resource in stream processing such as using Peer-to-Peer networking techniques for data stream sharing [StreamGlobe 2005], applying Grid infrastructure for storing lossless and analyzing the whole data streams [Tho et al. 2005] (see Chapter 5 for the more details)

3.3. Service Oriented Architecture

Recent years, Service Oriented Architecture (SOA) [SOA 2004a, SOA 2004b] becomes an increasingly popular concept, but in fact, it isn't a new idea at all. It's been around since the mid-1980s. But it never really took off, because there has been no standard middleware or application programming interfaces that would allow it to take root. There were attempts to build them, such as the Distributed Computing Environment (DCE) and Common Object Request Broker Architecture (CORBA), but neither really caught on, and SOA languished as an interesting concept, but with no significant real-world applications.

The roots of service-orientation can be found in three different areas: programming paradigms, distribution technology, and business computing. The development of different programming language paradigms has not only contributed to the implementation platform for the different elements of an SOA but also has influenced the interfacing techniques used in an SOA, as well as the interaction patterns that are employed between service providers and service consumers. Many of the concepts originally found in programming languages have also made their way into the distribution technology that is presently used to offer remote access to services provided by different applications on different technical platforms. Finally, and maybe most importantly, the evolution of business computing has resulted in a large number of proprietary as well as packaged applications (see Figure 10) such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), and Supply Chain Management (SCM), which today are providing the data content and the business logic that brings an enterprise SOA to life. Because of the closeness of services to concrete business functionality, service-orientation has the potential to become the first paradigm that truly brings technology and business together on a level where people from both sides can equally understand and talk about the underlying concepts.

SOA is defined as a specific way to consider the structuring of an IT landscape of some company starting from a business perspective. The core of SOA is to divide single applications into smaller, meaningful, business-oriented services. Services are self-contained and can be reused (opposite to large, monolithic systems). SOA then allows combining such services in a more flexible manner than the fixed logic incorporated in large monolithic applications. In general, an SOA intends to create a set of loosely

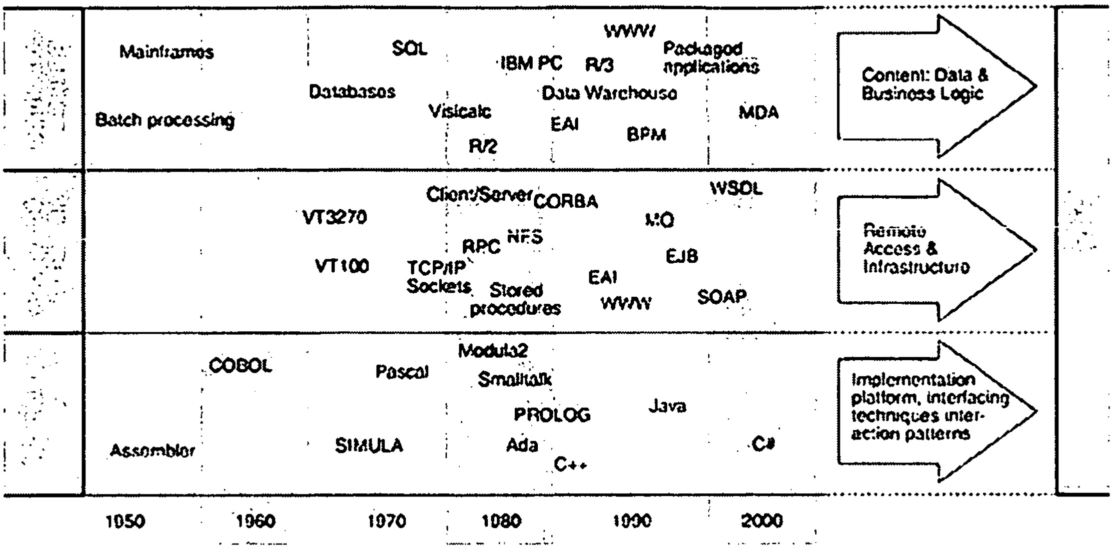


Figure 3.5. The long history of programming languages, distribution technology, and business computing toward a SOA paradigm ([SOA 2004a])

coupled services. It is a style of design that guides all aspects of creating and using business services throughout their lifecycle (from conception to retirement). An SOA is also a way to define and provision an IT infrastructure to allow different applications to exchange data and participate in business processes, regardless of the operating systems or programming languages underlying those applications.

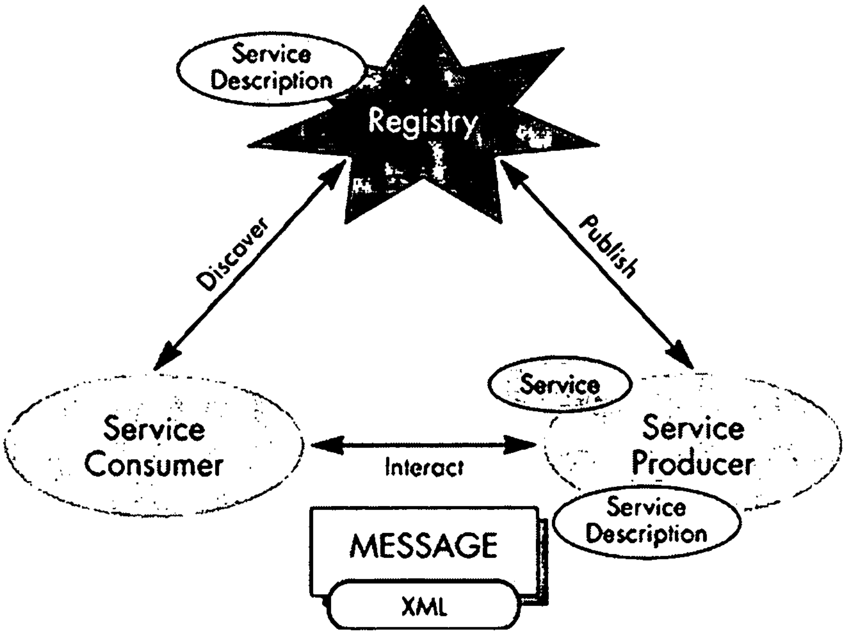


Figure 3.6. The SOA Interactive pattern

For better align with SOA principles, this architecture is polished with more emphasis on standard definitions of the service description, registration, discovery, and interactions. Figure 3.6 shows a commonly used interactive pattern.

In this interactive pattern, there are the following concepts:

- A *service provider creates* a service for interaction and exposes the service's description for the consumers with the *necessary message format* and *transport bindings*.
- The service provider may decide to *register* this service and its description with a *registry* of choice.
- The *service consumer* can *discover* a service from *a registry* or *directly* from the service provider and can start sending messages in a well-defined XML format that both the consumer and service can consume.

Service-oriented development provides the following benefits:

- **Reuse**: The ability to create services that are reusable in multiple applications.
- **Efficiency**: The ability to quickly and easily create new services and new applications using a combination of new and old services, along with the ability to focus on the data to be shared rather than the implementation underneath.
- **Loose technology coupling**: The ability to model services independently of their execution environment and create messages that can be sent to any service.
- **Division of responsibility**: The ability to more easily allow business people to concentrate on business issues, technical people to concentrate on technology issues, and for both groups to collaborate using the service contract.

Recently, when the Web services [Web Service 2005] came along, SOA has been given a boost to take off. The Web services underlying architecture dovetails perfectly with the concept of SOA. With the widespread adoption of Web services, the Web Services Description Language (WSDL) [SOA 2004b] has become a standard programming interface to access any application, and SOAP has become a standard interoperability protocol to connect any application to any other. These two standards are a great beginning, and they are followed by many additional Web services specifications that define security, reliability, transactions, orchestration, and metadata management to meet additional requirements for enterprise features and qualities of service. Altogether, the Web services standards the best platform on which to build an SOA the next-generation IT infrastructure.

3.4 Grid Computing

In today's pervasive world of needing information anytime and anywhere, the explosive Grid Computing environments have now proven to be so significant that they are often referred to as being the world's single and most powerful computer solutions.

The worldwide Grid Computing discipline involves the actual connections of a potentially unlimited number of machines within a grid, and can be most simply thought of as a massively large power "utility" grid, such as what provides power to our homes and businesses every day

3.4.1. What is the Grid?

The theory of Grid Computing simple enough, Hugh Bradlow, Telstra Research Laboratories, Clayton says "*If we could use all the spare cycles on all the millions of computers around the world we could unleash massive computing power*". Grid Computing solutions are constructed using a variety of technologies and open standards [Foster et al. 2001, Fellenstein et al. 2003] to provide highly scalable, highly secure, and extremely high-performance mechanisms for discovering and negotiating access to remote computing resources in a seamless manner. This makes it possible for the sharing of computing resources, on an unprecedented scale, among an infinite number of geographically distributed groups. This serves as a significant transformation agent for individual and corporate implementations surrounding computing practices, toward a general-purpose utility approach very similar in concept to providing electricity or water. These electrical and water types of utilities, much like Grid Computing utilities, are available "on demand," and will always be capable of providing an always-available facility negotiated for individual or corporate utilization.

"*The dream behind the Grid is of a common resource space in which we can work together using shared resources*". In [Foster& Kesselman 1998,] the Grid concept is defined as the controlled and coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. This sharing of resources, ranging from simple file transfers to complex and collaborative problem solving, is accomplished within controlled and well-defined conditions and policies. The dynamic grouping of individuals, multiple groups, or organizations that defined the conditions and rules for sharing are called virtual organizations

One of the significant operational concepts in Grid Computing is the notion of the virtual organization. This involves the dynamic computation-oriented task of defining groupings of individuals, such as multiple groups or organizations, defined around a set of resource-sharing rules and conditions [Foster et al. 2001]. All these virtual organizations share some commonality among them, including common concerns and requirements, but may vary in size, scope, duration, sociology, and structure. Although this is perhaps simple to understand, in theory, it remains complex across several dimensions. The complexities involved in this dynamic assembly revolve around identifying and bringing in those humans that initially defined the conditions in order to instantiate the grid. For instance, automated consideration of the rules, the policies, and the specific conditions affecting operations in the grid are, hence, the generating force for processing and sharing the information with those individuals in any virtual organization of a grid.

The simplest way of thinking about this advanced Grid Computing concept is captured in the term virtual organizations. This type of computation-oriented grouping serves as the basis for identifying and managing the grid computer groups, associated with any particular grid community of end users.

In the later published article "The Physiology of the Grid" [Foster et al. 2002], the foundation of the Open Grid Service Architecture, two important new terms were added to the Grid definitions: *Integration* and *Service.* They emphasize the importance of the integration of distributed, heterogeneous systems providing their functionality via Services, using common, open and standardized protocols. Based on Service Oriented Architecture (SOA) concepts these services allow a level of abstraction enabling applications to access and share resources and services across networks easily. *"to* integrate services *across distributed, heterogeneous, dynamic 'virtual organizations' formed from the disparate resources within a single enterprise and/or from external resource sharing and service provider relationships."* [Foster et al. 2002]

3.4.2 The Grid Infrastructure

The grid infrastructure forms the core foundation for successful grid applications. This infrastructure is a complex combination of a number of capabilities and resources identified for the specific problem and environment being addressed.

Grid computing environments usually consist of a complex infrastructure embracing various classes of components. They enable the creation of virtual computing resources that Grid applications use as execution environment. In general, a Grid Computing infrastructure component must address several potentially complicated areas in many stages of the implementation. These areas are:

- Security
- Resource management
- Information services
- Data management

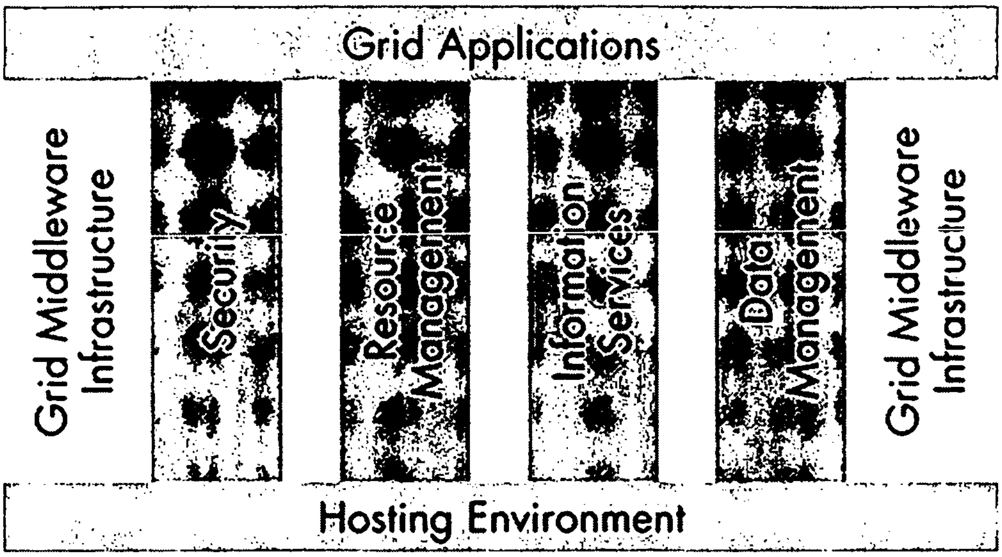


Figure 3.7. The Grid Infrastructure

Security: The heterogeneous nature of resources and their differing security policies are complicated and complex in the security schemes of a Grid Computing environment. These computing resources are hosted in differing security domains and heterogeneous platforms. Simply speaking, our middleware solutions must address local security integration, secure identity mapping, secure access/authentication, secure federation, and trust management. The other security requirements are often centered on the topics of data integrity, confidentiality, and information privacy. The Grid Computing data exchange must be protected using secure communication channels, including SSL/TLS and oftentimes in combination with secure message exchange mechanisms such as WS-Security. The most notable security infrastructure used for securing grid is the Grid Security Infrastructure (GSI). In most cases, GSI provides capabilities for single sign-on, heterogeneous platform integration and secure resource access/authentication.

Resource Management: Resource management is concerned with the assignment of resources to submitted jobs, tracking their state during execution and providing capabilities for job and job-lifecycle management. It therefore acts as an abstract interface to available resources. The resource manager has to consider several issues in conjunction with the information services: It has to choose the appropriate resources that are assigned to a job, usually by working in conjunction with a broker (match-maker). It has to deal with concurrency and coordination of multiple sub-jobs (resolve dependencies, data sharing, coordination, IPC). Furthermore it must provide access to job results, states and messages.

Information Services: Information Services maintain knowledge about the resources available on the Grid, their capacity and current utilization. Examples for information collected within this central component include static host information (operating system, processor, memory, hardware, software, attached storage type and capacity, installed batch queues), dynamic host information (list of jobs, load, queue entries, CPU-times, memory usage, storage space available, job-related files and their properties), storage system information (disk space) as well as network information (bandwidth, latency).

Data Management: Many current Grid projects see their most important asset in data. Providing fast, reliable and transparent access to data to all users is one of the main objectives of those projects and their data management components. Reliable data transfer over wide-area networks is another important task related within this field, which is e.g. addressed by the GridFTP protocol. Another issue in data management is replica management, where identical copies of data are generated and stored at various globally distributed sites to reduce data access latency and increase performance [Guy et al. 2002].

3.4.3 The Grid Architecture

A new architecture model and technology was developed for the establishment, management, and cross-organizational resource sharing within a virtual organization. This new architecture, called grid architecture, identifies the basic components of a grid system, defines the purpose and functions of such components and indicates how each of these components interacts with one another [Foster et al. 2002]. The main attention of the architecture is on the interoperability among resource providers and users to establish the sharing relationships. This interoperability means common protocols at each layer of the architecture model, which leads to the definition of a grid protocol architecture as shown in Figure 3.8. This protocol architecture defines common mechanisms, interfaces, schema, and protocols at each layer, by which users and resources can negotiate, establish, manage, and share resources.

Fabric Layer: Interface to Local Resources

The Fabric layer defines the resources that can be shared. This could include computational resources, data storage, networks, catalogues, and other system resources. These resources can be physical resources or logical resources by nature.

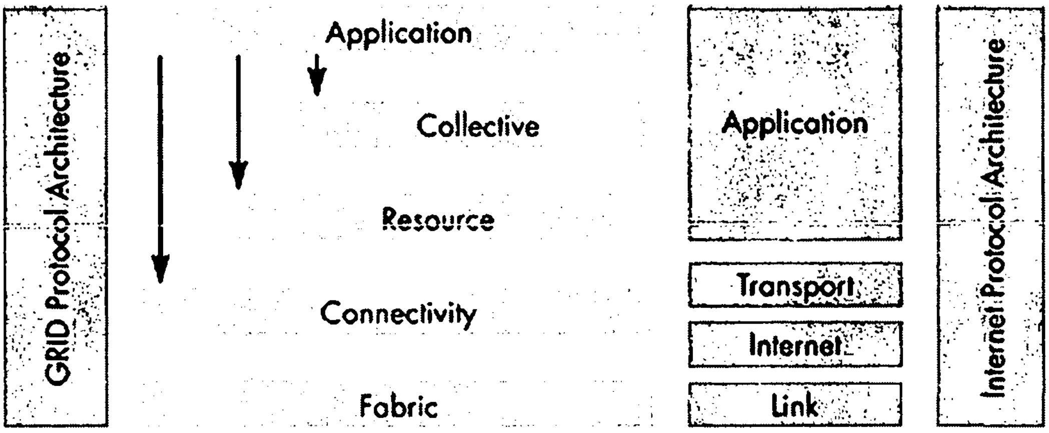


Figure 3.8. The Grid Architecture

Connectivity Layer: Manages Communications

The Connectivity layer defines the core communication and authentication protocols required for grid-specific networking services transactions. Communications protocols, which include aspects of networking transport, routing, and naming, assist in the exchange of data between fabric layers of respective resources. The authentication protocol builds on top of the networking communication services in order to provide secure authentication and data exchange between users and respective resources.

Resource Layer: Sharing of a Single Resource

The Resource layer utilizes the communication and security protocols defined by the networking communications layer, to control the secure negotiation, initiation, monitoring, metering, accounting, and payment involving the sharing of operations across individual resources.

The Collective Layer: Coordinating Multiple Resources

While the Resource layer manages an individual resource, the Collective layer is responsible for all global resource management and interaction with a collection of resources. This layer of protocol implements a wide variety of sharing behaviours (protocols) utilizing a small number of Resource layer and Connectivity layer protocols.

Application Layer: User-Defined Grid Applications

These are user applications, which are constructed by utilizing the services defined at each lower layer. Such an application can directly access the resource, or can access the resource through the Collective Service interface APIs (Application Provider Interface).

Each layer in the grid architecture provides a set of APIs and SDKs (software developer kits) for the higher layers of integration. It is up to the application developers whether they should use the collective services for general-purpose discovery, and other high-level services across a set of resources, or if they choose to start directly working with the exposed resources. These user-defined grid applications are (in most cases) domain specific and provide specific solutions.

3.4.4 Evolution of Grid Computing

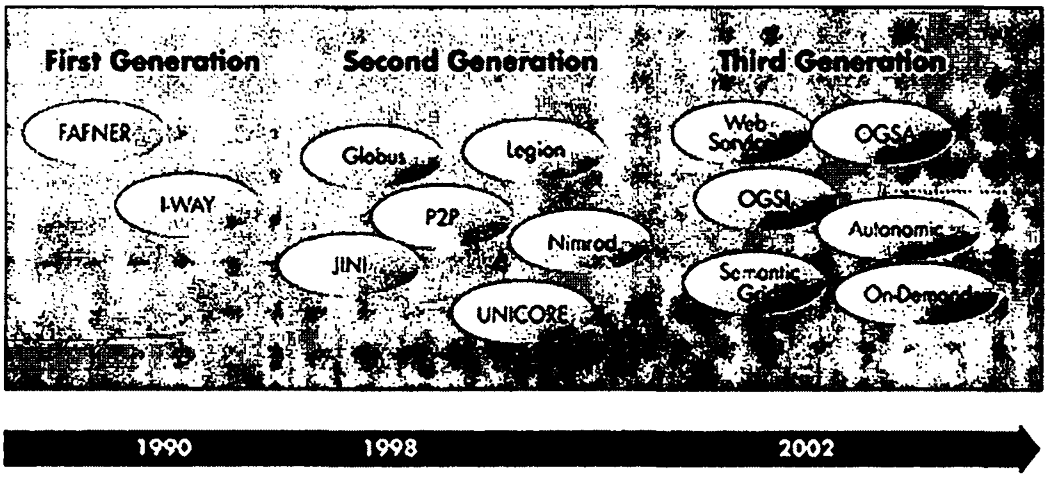


Figure 3.9. The Grid Generations

As shown in Figure 3.9, the evolution of Grid Computing is broadly classified into three generations. This Grid Computing evolution is discussed in detail by Roure, Baker, Jennings, and Shadbolt [Roure et al. 2003].

The First Generation

In the 1980s researches were busily working on techniques for parallelizing compute- and data-intensive applications for tightly-coupled parallel systems. Up to the 1990s the focus laid on efficient and powerful mechanisms for managing communication between processors and on execution environments for parallel machines. For example, the Message Passing Interface is a widely-used standard for building parallel programs that exchange messages between the involved nodes. High Performance Fortran is an example for efforts on parallel programming languages and the Parallel Virtual Machine for an scalable parallel execution environment. I-WAY, the information wide area year, is usually seen as the beginning of Grid computing. In December 1995 at the Supercomputing Conference [SuperComputing] in San Diego the I-Way prototype was presented, linking 17 big computer centers in the United States for one week to demonstrate the execution of simulations as if run on a single machine. Led by Tom DeFanti of the University of Illinois at Chicago and Rick Stevens of Argonne National Laboratory about 60 application demonstrations were run on the the I-Soft system, the first attempt in constructing a unified software infrastructure for such systems.

The first generation of Grids is characterized by attempts to realize meta-computing environments for very specialized purposes, mainly to link supercomputing centers. Although I-WAY dealt with many common problems rudimentary it was highly innovative and successful and opened the door to follow-on projects.

The Second Generation

Once the term was defined and first generation Grid projects had successfully demonstrated their capabilities, several groups around the world started research and development activities in this field. These efforts created many building blocks of to days second generation Grids. Some of them already had to prove their suitability for production environments, while others are still emerging or under development.

The Globus Toolkit has evolved from its original incarnation as I-WAY to a production-ready middleware infrastructure addressing many requirements that Grid applications have. The Globus Project is a US research and development project delivering one of the most important Grid toolkits that is available today. The University of Virginia developed another meta-system called Legion providing a software infrastructure for heterogeneous, geographically distributed, high-performance Grid computing. The main difference between Legion and Globus is the underlying programming model. Legion represents all Grid resources as objects, while Globus is providing a bunch of different APIs.

High-Throughput scheduling is addressed by toolkits like Condor, a system that allows to utilize idle computing power from pools of workstations often in local networks. The Network Weather Service focuses on resource monitoring and prediction, the NetSolve environment on remote numerical calculations and SDSCs Storage Resource Broker on uniform access to heterogeneous data resources.

Nimrod/G is a Grid broker performing resource management and scheduling. Most of today's Grid projects rely on second generation Grid efforts.

The Third Generation

Currently the transition between the second and the third generation is ongoing. Modern, state-of-the-art software engineering results are incorporated and integrated into the Grid community. Additionally, a shift in emphasize from information to knowledge, as some novel projects for e-Science [eScience a, eScience b] and

Knowledge Grids [KGrid a, KGrid b] demonstrate, can be noticed. Furthermore the "Semantic Grid" term [Roure et al. 2003] appeared and introduced a new semantic-layer for enhancing interaction with and perception of Grid computing environments.

As a consequence of these trends the Global Grid Forum6 (GGF) was created in the late 1990s with the objective to define global standards for the Grid community. These efforts try to ensure the crucial interoperability between the different Grid research and development projects. Recently the GGF integrated the concept of service-oriented architectures, Grid computing and web services technologies into the "Open Grid Service Architecture". Another shift in emphasize can be found in the main objectives, which many Grid computing projects address nowadays. Instead of only creating software and toolkits dealing with high-end distributed computing requirements, most projects now address the long-term goal of building infrastructures for e-Science.

3.4.5 Classification of Grids

Grids may be classified following different criteria. Examples are the target user community or the class of problems they address. Depending on the major goals that are to be achieved different kinds of middleware frameworks and components have to be deployed to support the applications that will use them. A widely used discrimination is based on the major asset the Grid is providing for the user community: computational power or data storage.

Computational Grids

Computational Grids are defined as "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" [Foster& Kesselman 1998]. They focus on computational intensive problems (many high-throughput challenges belong to this category) characterized by a high computing cycle per data unit ratio. Significant pools of computing resources with high-performance connections build the basis for Computational Grids. A layer of middleware components is responsible for allocation, scheduling, resource brokering and job management aiming at maximizing job-throughput.

Data Grids

The challenges coming with data-intensive computing problems have already been outlined using the LHC data challenge example at CERN7. Data Grids address such situations where large amounts of distributed dataset have to be accessible and high-volume data flows over distributed resources and wide-area networks have to be managed. Efficient middleware components for data access and transfer, caching, replica-management are crucial to Data Grids. The European DataGrid [EUGrid] project was one of the leading projects addressing these data challenges.

3.5 Summary

In the first half of this chapter, we review some other concepts and terms related to the topic of Zero-Latency Analytical Environment such as Zero-Latency Enterprise, Real-time Data Warehousing, Active Data Warehousing, Real-time Analysis, Just-in-time Decision Support, Business Activity Monitoring (BAM) and explain their relationship with the Zero-Latency Data Warehousing topic. The rest of the chapter presents other technologies and concepts which are used later in the dissertation: Continuous Data Streams, Service Oriented Architecture and Grid Computing. For each subtopic, we have mostly discussed the overview of the topic, the fundamental concepts and related research activities and challenges.

PART II.

IMPLEMENTING ZERO-LATENCY DATA WAREHOUSING ENVIRONMENT

Chapter 4. Event-feeded Slowly Changing Dimension (Case Study at T-Mobile- Austria)

> *"The events in IT systems contain untapped information. Complex Event Processing (CEP) lets you extract it and use it in ways you want to".*
>
> **David Luckham, "The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems", 2002**

From the point of view of a data warehouse system, collecting and receiving information from other systems is crucial for all subsequent business intelligence applications. The incoming information can be generally classified in two types, the state-oriented data and event-oriented data usually called transactional data, which contains information about the change performed by processes on the instances of information objects. On the way towards active data warehouses it becomes more important to provide data with minimal latency. We focus on dimensional data provided by general data warehouse applications. The information transfer is done via messages containing the change-information of the dimension instances. The receiving data warehouse system is able to validate the event-messages, reconstruct the complete history of the dimension and provide a well applicable "comprehensive slowly changing dimension" (cSCD) interface for well-performing queries on the historical and current state of the dimension. A prototype implementation of "active integration" of a data warehouse is proposed.

4.1 Introduction

The presence of time and the dependence upon it is one of the properties that sets data warehouse applications apart from traditional operational systems. Time is one of the four basic characteristics of a Data Warehouse [Inmon 1996]. In Data Warehousing the explicit existing of time enables historical data to be held and queried. This means that users of data warehouses can view aspects of their organization at every specific point or over every period of time for which historical data is recorded. This enables the observation of behavioural patterns in the course of time so that we can make comparisons between similar or dissimilar periods, e.g. this year versus last year, seasonal trends. Ultimately we could also, in fact, using information of the past to predict the future.

The temporal requirements (i.e. time based historical presentation of data) of a data warehouse are very different from those of an operational system. While in a data warehouse we must keep the full historical nature of data, the operational system aims to keep only the current or very recent historical data. Therefore, the operational system must feed information about changed data to the data warehouse to refresh the data warehouse. One major role of the data warehouse is to keep the consistent historical data to provide the correct information at any point in time the user require.

The upcoming integration technology standards [TIBCO,Vandermay 2000] based on message exchange between information systems provide much benefits not only to operative systems. Also the data warehouses gain significant profits out of this development [Brobst 2002c]. In the past a restricted integration of the source systems has led to batch-oriented data load approaches for data warehouses. The data warehouse refresh is thus performed in batch mode when it receives multiple operational source snapshots, extract the changes, and refresh the warehouse data.

However, there are some significant limitations of the snapshot base approach. Multiple change events between snapshots have been completely missed. Furthermore, for each snapshot-comparison the number of records to be processed is high, requiring high computing-resources and -time and finally the history is kept by tremendous daily snapshot versions consuming a lot of storage. Therefore, a more efficient approach towards a near-real time solution is considered.

In this chapter, we introduce the research work at T-Mobile Austria to develop an event-based refresh paradigm which is applied to solve the Slowly Changing Dimension (SCD) issue. For further processing of event-messages, we develop one comprehensive and general applicable SCD representation inspired by Kimball's three SCD-types [Kimball 2002] and we propose a valid alternative to the snapshot based information transfer. This solution is especially applicable in cases, where the information requirements of the receiving system is focused on complete and detailed historical information for all instances combined with a minimal latency demand. For a given latency time-interval the advantages of this method are evidently given for a dimension with a small number of changes compared to its cardinality.

The proposed method provides much more than a data replication. The primary target is not a physical mirror of the dimension object. All necessary views including the change history of this object are implemented in a standardized way. The event feeded cSCD approach has been designed according to the main goals of T-Mobile Austria's data warehouse, which are simple: "to provide a single point of truth - easy to access".

4.2. State-oriented Data versus State-change or Event-oriented Data

In abstract system theory, the notation of state is introduced in order to separate the past from the future [Kopetz et al. 2002]. A *state* is something that has extent in time. Something is true about an object for a period of time, but was neither true before nor after. An *event* is instantaneous [Jensen, Dyreson 1998]; it is something that "happens", rather than being true over a period of time. Events delimit states. The occurrence of an event results in a fact becoming true; later, the occurrence of another event renders that fact no longer valid. Hence, events and states are duals: states can be represented by their delimiting events, and events are implied by states [Jensen, Snodgrass 1996].

In a data warehouse, data exists in two types: (1) State-oriented data and (2) State-change or Event-oriented data. Examples of state-oriented data include, e.g., address, prices, account balances, and inventory levels. Examples of event-oriented data are sales, inventory transfers, and financial transactions.

Since *every event is significant* in event-oriented data, the loss of a single event can lead to a loss of synchronization in state between a source system and the data warehouse. Datasets containing event information must be queued and removed on reading to ensure that every event is processed just once and no event gets loss.

If the system processes *state information*, e.g., the current address, it is reasonable to integrate it with the old version of the state value stored. In such a system there is no need to guarantee that every dataset is processed just once. In addition, if a single state change is lost during data integration, the data warehouse is automatically resynchronized by processing the next state information.

Event-oriented data in a data warehouse uses an *event representation*, meaning that each row in the fact table represents some event and has a timestamp, capturing the event occurrence time. For state-oriented data, we employ the *state representation.* Every row in the table describes some state and has two timestamps – i.e. the beginning and ending times of the period throughout which the state persisted (see Table 4.1).

Transaction Date	Transaction Amount
2005-05-05	-100
2005-05-07	+500

Begin Date	End Date	Account Balance
2005-04-30	2005-05-05	1000
2005-05-05	2005-05-07	900
2005-05-07	*until changed*	1400

Table 4.1: Event Representation vs. State Representations of Data

4.3. Slowly Changing Dimension

In a data warehouse the information storage is typically divided into fact tables and dimension tables [Kimball 1996]. Fact table data, by its nature, represents a time series of measurements and is always augmented with an explicit time dimension. Analyzing old fact-table data is one of the standard queries in a data warehouse.

Dimensional data requires more special treatment. Dimensions do not change in predictable ways, i.e. individual entities (e.g. customers, products) evolve slowly. Some of the changes are true physical changes (e.g. customer's new address). Other changes are actually corrections of mistakes in the data.

One of the major contributors to the development of solutions in the area of changes in dimension attributes is Ralph Kimball. His common term for such changes is the notion of *slowly changing dimensions* (SCDs) [Kimball 2002a]. The purpose of the SCD solution is to maintain the un-altered relationship between the facts and dimension table without updating the fact tables when the dimension data is changing.

In the following sections, we will briefly review the SCD type 1, 2, 3 proposed by Kimball and discuss about their limitations before suggesting an enhanced SCD type 2 which we have used at T-Mobile Austria.

4.3.1. SCD type 1

Overwrite the old values in the dimension record with the new values

This is the simplest and fastest SCD solution because it simply overrides the old dimensional value with the new one without keeping any trace. Obviously, it does not maintain any past history and we loose the ability to track the old history. The technique thus does not address the implications of evolving data, because a fact is associated with only the current value of a dimension column. Nevertheless, overwriting is frequently used when the data warehouse team legitimately decides that the old value of the changed dimension attribute is not interesting.

Customer_ID	Name	Address	(additional columns)	Load Timestamp
C1	Robert	20 Rennweg	...	2005-02-14

↓

Customer_ID	Name	Address	(additional columns)	Load Timestamp
C1	Robert	25 Favoritenstr	...	2005-02-16

Table 4.2: Example using SCD Type 1

Table 4.2 illustrates an example using SCD Type 1 to keep information about customer Robert change his address from 20 Rennweg to 25 Favoritenstr on 2005-02-16. As we can see, the old value of address (20 Rennweg) is overridden by the new one.

4.3.2 SCD type 2.

Create an additional dimension record using a new value of the surrogate key.

This technique segments history between the old and the new value. A new surrogate key is created when a true physical change occurs in a dimension entity at a specific point in time, such as a customer's new address.

A fact is always associated with the original value of a dimension column. A sequence of facts describes the evolving data. This technique automatically partitions history and an analytical application are not required to place any time constraints on effective dates in the dimension. However, finding out the whole sequence of changes from the data repository (in order to do analysis across history) is difficult and involves complex and expensive queries.

Surrogate key	Customer_ID	Name	Address	(additional columns)	Load Timestamp
C11	C1	Robert	20 Rennweg	...	2005-02-14

↓

Surrogate key	Customer_ID	Name	Address	(additional columns)	Load Timestamp
C11	C1	Robert	20 Rennweg	...	2005-02-14
C12	C1	Robert	25 Favoritenstr	...	2005-02-16

Table 4.3: Example using SCD Type 2

The use of SCD Type 2 requires that the dimension key is generalized (*surrogate key*) to ensure the uniqueness property of the primary key of the dimension table when adding the new record with the same key with the previous record. It may be sufficient to take the underlying production key and add some version digits to the end of the key to simplify the surrogate key generation process. Future insertions to the fact table will be related to the new identifying codes, and so the segmentation will remain consistent with respect to time. One minor issue compared to SCD Type 1 is that, by making use of surrogate keys, it becomes impossible to recognize individual customers by using the identifying attribute (i.e. Customer_ID). Table 4.3 illustrates the example of using SCD Type 2 to keep trace the address information after the changing.

4.3.3 SCD Type 3

Create an "old" field in the dimension record to store the immediate previous attribute value.

In application which requires the comparisons across transitions, SCD type 3 is an appropriate solution. In SCD type 3, there will be two columns to indicate the particular attribute of interest, one indicating the original value, and one indicating the current value. There will also be a column that indicates when the current value becomes active. A fact is associated with both the original value and with the current value of a dimension column.

Compare to SCD Type 2, SCD Type 3 does not increase the size of the table, since new information is updated while it still keeps part of history. However, SCD Type 3 is rarely used in actual practice because it modifies the structure of the dimension tables (adding more columns). Further more, SCD Type 3 is not be able to keep all history where an attribute is changed more than once because it keeps only the original and the current values of the changed attribute. Intermediate values are lost.

Customer_ID	Name	Original_Address	Current_Address	Effective_Date	(add. columns)
C1	Robert	20 Rennweg	20 Rennweg	2005-02-14	...

↓

Customer_ID	Name	Original_Address	Current_Address	Effective_Date	(add. columns)
C1	Robert	20 Rennweg	25 Favoritenstr	2005-02-16	...

Table 4.4: Example using SCD Type 3

4.3.4 Limitations of the existing SCD Approaches

Among the above three SCD solution approaches, SCD Type 2 is the most widely used in Data Warehousing projects [Kimball 2002b]. While SCD Type 1 and Type 3 update the existing data thus conflict with the non-volatile Warehouse design criteria [Inmon 1996] and destroy the old information, SCD Type 2 insert the new record without delete or update anything. However, SCD Type 2 partitions history strictly and does not allow for overlapping valid times. Due to the absence of dates, it is impossible to determine precisely when changes occur. The only way to extract the time is via a join to the fact table. This will give an approximate time for the change. The degree of accuracy depends on the frequency of fact table entries relating to the dimensional entry concerned. The more frequent the entries in the fact table, the more accurate the traceability will be of the history of the dimension, and vice versa.

Three following main issues explain why the existing SCD approaches are not sufficient:

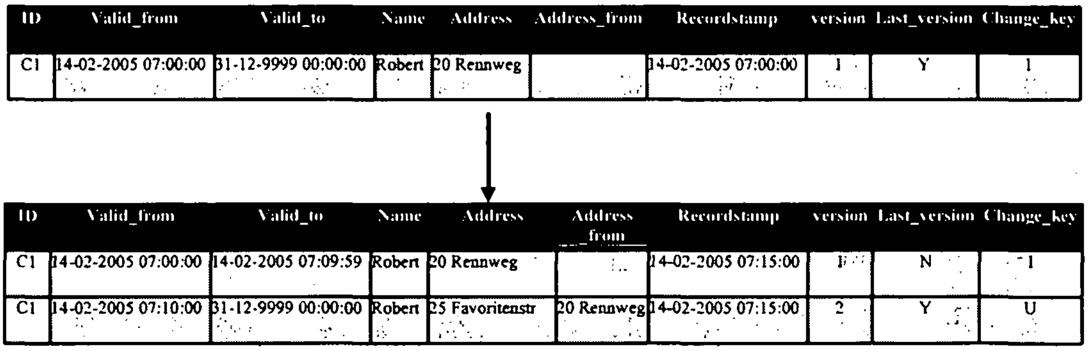
1. Comparing one dimension row with another to determine what changed involves performing analysis across rows; something at which SQL is notoriously bad [Kimball 2002c].
2. Even when the next row in the dimension correctly tracks a change, it is impossible to examine the row in isolation to determine exactly what changed, especially if more than one attribute has been changed.
3. If the data warehouse has to "tie to the books", then it is not allowed to change e.g. an old monthly sales total, even if the old sales total was incorrect [Kimball 2002a]. Late-arriving fact and dimension data cannot be integrated in such data warehouses using traditional SCD techniques.

4.3.5 Comprehensive enhanced SCD Solution

As discuss in Section 4.3.2 and 4.3.3, SCD Type 2 can keep traces of multiple changes of dimensional attribute, while SCD Type 3 can keep the current value and previous value of dimensional attribute. One can think about the possibility of mixing SCD Type 2 and SCD Type 3 to support increased analytical application complexity.

We propose an enhanced SCD which are the mix between SCD Type 2 and SCD Type 3 as follow. For each dimensional attribute change, as in case of SCD Type 2, create a new record with the new surrogate key into a table namely TRANS table. The fact table records will link to the relevant TRANS record via surrogate key. However, for the attribute value chain tracing purpose, additional columns are added into TRANS table. *Previous_Value* keeps the previous value of the attribute before it changes to current value, *Valid_from* and *valid_to* determine the valid duration of the dimension attribute during the whole history of dimension entity. *Version* is the counter to keep the version of dimension attribute in time. *Last_version* is the redundant attribute which indicates the newest version (which means that the correspondent record stores the current attribute value and still be valid at the moment) [see also EDER 1986].. *Recordstamp* points out the processing time of the change of dimension attribute. *Change_key* is the attribute indicates which operation has effect the attribute (i.e. insert or delete or update). This column is used for checking the validation of the transaction which will be discussed later in this chapter.

Please note that each record in the TRANS table is corresponding to a change value of one dimension attribute we are interested in which we called *traced attribute*. In case we do not consider the change of dimension attributes, these attributes are classified as *flat attributes* and the TRANS table only keeps the current value (overridden as in SCD type 1). For example, consider the case of *customer* dimension table which contains the following columns: ID, Name, Address. ID is the dimension key, Name is the *flat attribute* (i.e. the change of name value is not of some interest and thus is not traced). Address is a *traced attribute*. Table 4.5 gives us an example how TRANS keeps the records when customer Robert changes his address from 20 Rennweg to 25 Favoritenstr. The valid_to open date value: 31-12-9999 00:00:00 implies *until change.*



ID	Valid_from	Valid_to	Name	Address	Address_from	Recordstamp	version	Last_version	Change_key
C1	14-02-2005 07:00:00	31-12-9999 00:00:00	Robert	20 Rennweg		14-02-2005 07:00:00	1	Y	1

ID	Valid_from	Valid_to	Name	Address	Address from	Recordstamp	version	Last_version	Change_key
C1	14-02-2005 07:00:00	14-02-2005 07:09:59	Robert	20 Rennweg		14-02-2005 07:15:00	1	N	1
C1	14-02-2005 07:10:00	31-12-9999 00:00:00	Robert	25 Favoritenstr	20 Rennweg	14-02-2005 07:15:00	2	Y	U

Table 4.5: Example using enhanced SCD (mix Type 2 and Type 3)

4.4. Data Warehousing at T-Mobile

The data warehouse at T-Mobile Austria runs on an Oracle 9.1.3 relational database and has a data volume of about six terabyte (TB). The complexity and number of operational source systems is very high. Therefore, the data warehouse provides its information as a single point of truth for nearly all units of the enterprise.

One of the most important operational sources for the data warehouse is the *BSCS* system (*Business Support & Control System*). It is the billing system and stores customer relevant data. However, the customer care system at T-Mobile Austria does not directly use BSCS functionality for performance reasons. It has an independent data repository and reads/ updates relevant data by using BSCS interfaces.

4.4.1 Snapshot based SCD approach

Since 2001, the data warehouse at T-Mobile Austria is built and refreshed using the batch approach. The dimensional data loaded from legacies like the billing system (BSCS), SAP or the CRM Systems is received via daily snapshots. Although the data is currently batch loaded, the data freshness for transactional data is very high, e.g. CDRs (call detail records) are usually loaded every four hours.

The Data Warehousing team at T-Mobile Austria has implemented a flexible *snapshot comparison* approach. Figure 4.1 gives an overview of the model. Applying the snapshot comparison technique to a particular data warehouse dimension involves three tables:

Table SNAP

This table contains full snapshots of relevant datasets and tables of an operational source system. A snapshot describes datasets, which are already cleansed and transformed. However, a snapshot only contains current data, i.e. if a dataset / key is deleted within an operational source system, it is not part of the snapshot. A sequence of snapshots is distinguished by their snapshot date.

Table TRANS

This table contains the transactions, which were "derived" from two arbitrary (in general succeeding) snapshots taken from the table SNAP. The snapshot comparison approach distinguishes between flat and traced attributes.

- If a change affects only flat attributes, it does not cause the generation of a new version of the according dimension dataset in the data warehouse.
- If a change affects a traced attribute, the according transactions (insert, update, or delete) in the operational system are reconstructed by comparing the snapshots.

Table FLAT

This table contains the full history of the life-cycle of dimension datasets. If an update or delete transaction was identified by the snapshot comparison, a new version of the involved dimension dataset is inserted into the data warehouse using an enhanced SCD (mix Type 2 and Type 3) technique. The effective date of the changed attribute is derived from the snapshot date.

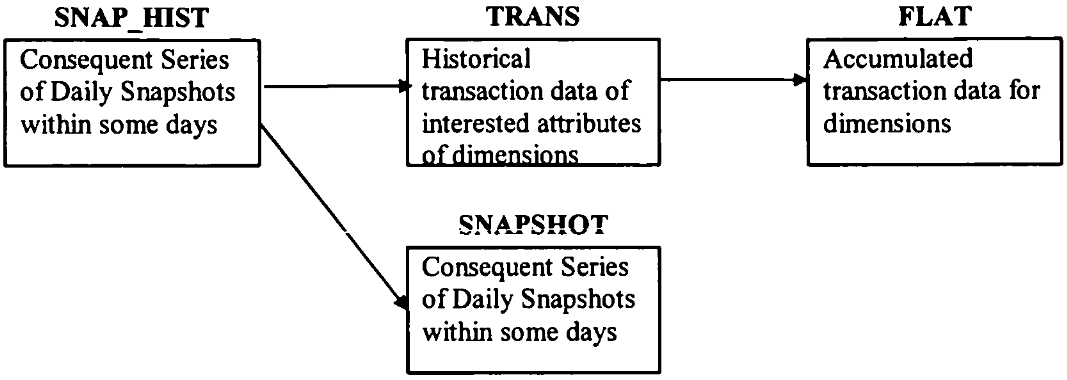

SNAP_HIST
Consequent Series of Daily Snapshots within some days
TRANS
Historical transaction data of interested attributes of dimensions
FLAT
Accumulated transaction data for dimensions
SNAPSHOT
Consequent Series of Daily Snapshots within some days

Figure 4.1. Snapshot-based comparision SCD approach

With this approach, a series of complete snapshots are stored chronologically to trace the complete history of dimensions without considering the number of actual data changes. This demand has been implemented via a PL/SQL package namely UTL_SCD. This functionality has often been reused within the data warehouses ETL processes which are performed with the Informatica tool.

However, some issues remained unsolved. It is possible that more than one change to the same attribute occurs between two succeeding snapshots. If the data is collected by the snapshot comparison method, only those values will be captured, which exist at the time of the snapshot. All intermediate changes are completely missed in this case. Additionally, the tremendous daily snapshot versions are kept in the data warehouse for a specified time-period and significantly increase the data size while the amount of data changes is actually relatively small. For each snapshot comparison the number of records to process is high, requiring also high computing-resources and -time.

4.4.2 Event based SCD approach

Due to many issues of the snapshot based approach, a more efficient near-real time approach is considered. Data changes in the operational sources are captured and provided near real time as event messages via the event-based infrastructure TIBCO [TIBCO]. This approach implies also the validation of the message content, which is necessary because of the highly differing quality provided by the message sources. Mainly three quality aspects, which are independent from the event-based infrastructure, are under inspection: the completeness, uniqueness and order of the event-messages.

For the further processing of the event-messages we developed one comprehensive and general applicable SCD representation inspired by Kimball's three SCD-types [Kimball 2002a, Koncilia, Elder 2001] as discussed in Section 4.3.5 and we propose a valid alternative to snapshot based information transfer. This solution is applicable especially in cases, where the information requirements of the receiving system is focused on complete and detailed historical information for all instances enhanced with a minimal latency demand. Especially in cases, where a dimension contains a large number of instances compared with the number of instance-changes within the time interval, which represents the time-latency requirements of the receiving system, the advantages of this method are obvious.

The proposed method is in fact a kind of logical information replication which can be implemented successfully with quite realistic efforts. But the target of this logical replication is not to build a mirror of the dimensional physical object but to provide all necessary views on this objects to fulfill a much higher variety of reader demands, than the original source object can support. The event feeded cSCD approach has been designed according to the main goals of T-Mobile Austria's data warehouse, which are simple: "to provide a single point of truth easy to access".

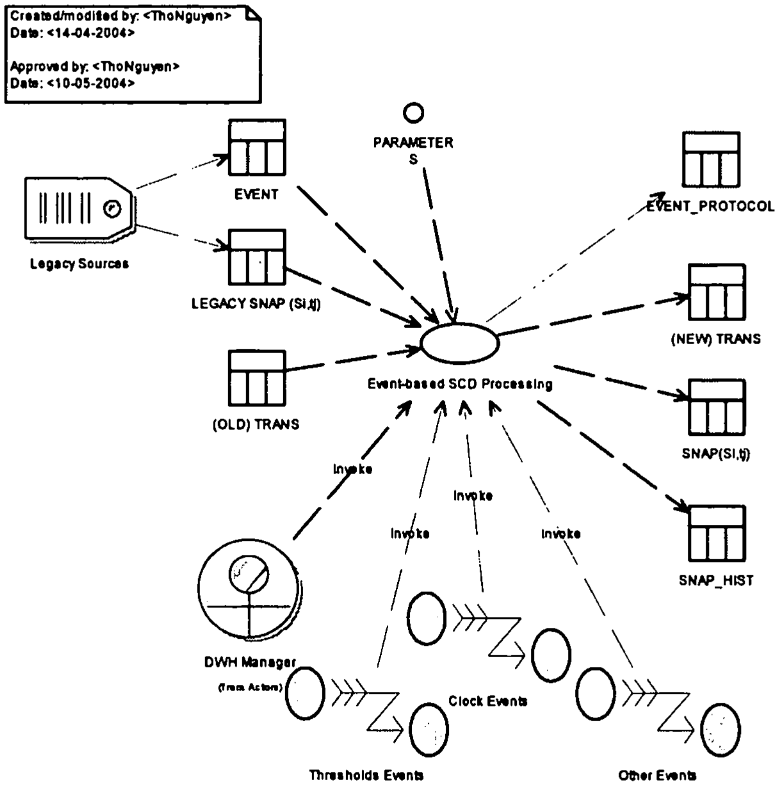


Figure 4.2: Global view of Event-based SCD process

Figure 4.2 depicts the overall view of our Event-based SCD process. Compared to the snapshot-based approach, the Event-based SCD process does not require to keep all consequent snapshots data, instead it requires the event data which reflects all changing data since the last refresh cycle. The legacy systems thus do not need to send the full snapshot every night as they have to do at the moment. However, they need to provide the event data as frequently as required in the near-real time refresh cycle.

The Event-based SCD process will access the Event data, filters those which happen since the last refresh (those records which not appear in TRANS or different with current records in TRANS) and then update the TRANS table.

The new event-based approach requires the high level of truthfulness of the event. Therefore, the event-based SCD approach must provide the feature of event validation which validates the events with automatic correction options to override some invalid events before applying these events to refresh the TRANS. The Table Event_Protocol keeps the invalid events as an operational log.

Because we do not keep all snapshots, if in the case that there is requirement to have the snapshot at one point in time of any subset of ntity instances, it must be possible to rebuild such Snapshot data. From the TRANS table, we can rebuild the snapshot of any subset instance at any point in time. The snapshot generation process can be totally based on TRANS or it could receive a truthful snapshot as the based snapshot to generate the on demand snapshot of any subset Si at any time point tj.

Such on demand Snapshot could be inconsistent with the Legacy Snapshot at some time point, we thus have to check our TRANS to be consistent with the Legacy SNAP at these time points. When the inconsistency status is detected, the inconsistent data will be applied as the incoming events to re-establish the consistency. This process will also store the truthful Legacy Snapshot table in the periodic SNAP_HIST.

4.5 Event Model

For a formal description of an event and event processing a UML based model is created. The core part of this model is a UML profile describing the event meta-model. Additionally, the semantic of event is defined and shown by a simple example. Possible strategies of event interpretation are discussed.

Based on the defined model and the interpretation of the event we broader discuss and indicate that the traditional distinction between fact and dimension in event based DWH environment only represents a different specialization of our event based model.

4.5.1. Profile

To describe a general event it is necessary to raise the model to the meta-level M2 [9] as the structure of each event type is very proprietary based on the transferred business information. The simplified profile definition is depicted in Figure 4.3.

The key concepts of the event profile are as follows:

- Stereotype «Event» describes the object containing the event data.
- Stereotype «Efd» (abbreviation for event feeded dimension) depicts the target object that is maintained via the event stream.
- Stereotypes «Trans» and «Snap» as subtypes of «Efd» are discussed below

Those stereotypes are applicable on the class level, the rest of stereotypes are connected with an attribute:

- Stereotype «Key» is used to mark the (natural or surrogate) primary key of the dimension
- Stereotype «Order» is intended to define the order in which the events were created and should be processed.
- Stereotype «Timestamp» identifies an attribute containing the timestamp information of an event. The transaction time, event creation time, event processing time are various examples of this stereotype.
- Stereotype «Action» describes the nature of the change represented in the event (insert / update / delete)
- Stereotype «Status» enables the depiction of a logical deletion of a dimension instance.

Not all of the listed stereotypes are mandatory, the usage is constrained by semantic rules (see below) such as: The «Event» and «Efd» classes must contain at least one *Key* attribute (i.e. an attribute with stereotype «Key»). *Order* and *Timestamp* attributes may coincide, e.g. in cases when the time grain is to large to distinct the events uniquely, the *Order* attribute is used to define the unique event sequence. The opposite extreme when neither of those attributes is defined is also valid. In that case the "timestamp of event processing" can be used as a default *Timestamp* attribute (of course the unique order of events must be established in this case as well).

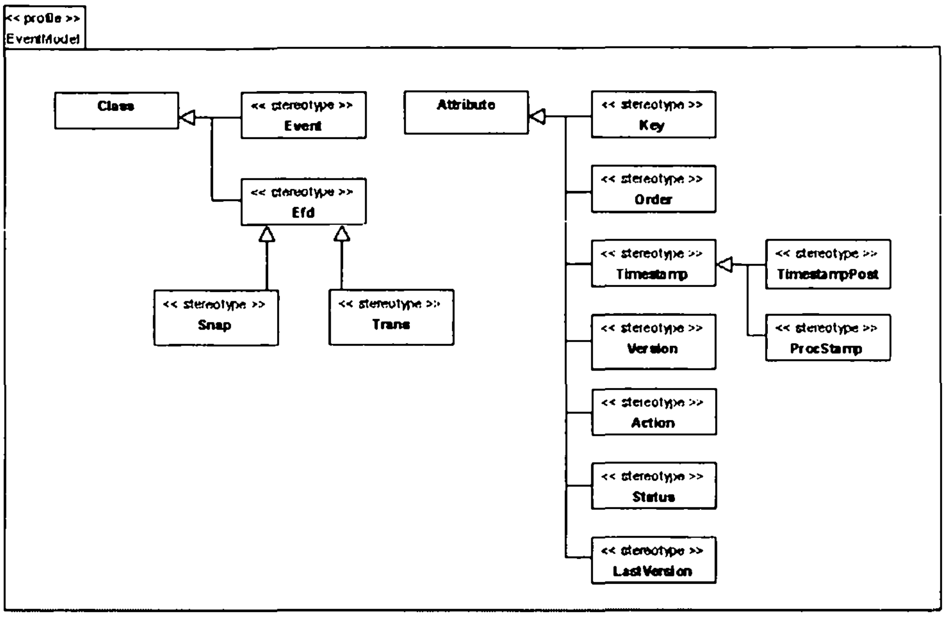


Figure 4.3. Simplified Profile Event Definition

4.5.2. Example

To illustrate the usage of the Event profile let us consider a simplified application that maintains the customer attributes via an event interface. The customer is identified with an attribute *id*, the customer attributes consist of *name, address and tariff.*

The class with its associated stereotype *Event* describes the customer-value-change event. This event is generated on each change of at least one attribute of a particular customer. As marked with stereotype *Key* the primary key of the customer dimension is the attribute *id.* The attribute timestamp is stereotyped as *Timestamp* i.e. this attribute defines the point in the time of the change of customer attributes. The rest of attributes have no stereotypes they are regular event attributes containing additional information.

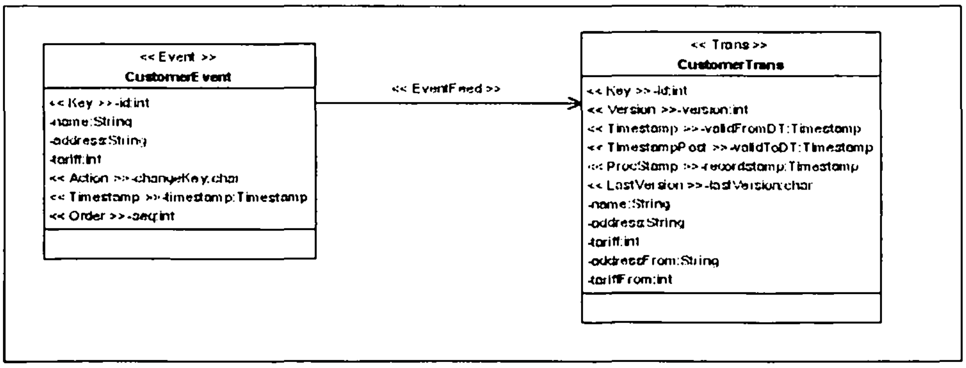

<< Event >>
CustomerEvent
<< Key >>-id:int
-name:String
-address:String
-tariff:int
<< Action >>-changeKey:char
<< Timestamp >>-timestamp:Timestamp
<< Order >>-seq:int
<< EventFeed >>
<< Trans >>
CustomerTrans
<< Key >>-id:int
<< Version >>-version:int
<< Timestamp >>-validFromDT:Timestamp
<< TimestampPost >>-validToDT:Timestamp
<< ProcStamp >>-recordstamp:Timestamp
<< LastVersion >>-lastVersion:char
-name:String
-address:String
-tariff:int
-addressFrom:String
-tariffFrom:int

Figure 4.4. Customer Event Profile Example

The second class in Figure 4.4 describes the target object maintained via the event feed (stereotype *Trans* defines that a full versioned history of the target object will be build; see the detailed discussion in 4.5.3). The meaning of the additional attribute is discussed below.

4.5.3. Event Processing

The profile based event model must be enriched with semantic rules defining the interpretation of an event. The most important feature is the sub-typing of the *Efd* object. In the profile two main examples are defined *Snap* and *Trans*.

The Snap object is maintained with overwrite policy, i.e. new records are inserted; existing records are updated or deleted. In a *Snap* object only one record per primary key is stored. *Snap* is mnemonic abbreviation for dimension snapshot.

The *Trans* object is maintained cumulatively, each event is added to the target object, building a complete transactional history of the dimension.

The handling of primary key of the build dimension can be configured. The Primary key option defines if the target object uses the natural key (as provided within the event) or if a surrogate key should be generated while the event is processed. In any case the primary key always uniquely identifies the dimension instance, so if a complete history of the dimension is maintained an additional attribute stereotyped as «Version» must extend the primary key of the target table.

Other option is defined on the level of attribute; an attribute noted as *TimestampPost* is applicable for *Trans* object only. It is filled with the value of the corresponding *Timestamp* attribute of the successor version decreased by the smallest grain of the time dimension (e.g. 1 ms). The default value is an artificially set high date (e.g. 31-12-9999 00:00:00). The usage of two timestamps in a full history table is not a "pure relational" design but extreme practical solution as for the selection of a version of a particular dimension occurrence a simple logic can be applied (*required timestamp* between *Timestamp* and *TimestampPost).*

If an attribute has a suffix *From* it contains the value "before the change", i.e. in *Trans* object this is the value stored in the preceding version. The association between the corresponding attributes is established with naming conventions.

A different semantic aspect is the validation of the event model, i.e. if the model is well-formed. Examples of constraints that must be checked are listed below:

- Event class must have at least one *Key* attribute
- Each *Timestamp* attribute must have a type compatible with date/time.

The final role of semanticchecking in the event context is the event validation. It is possible to extend the event data with redundant information that can be checked while the event is processed. The exceptions can be interpreted as an advice of lost or corrupted events.

For examples adding an *Action* attribute to the event (possible values: insert / update / delete) enables additional checks:

- key must exists in the target object on update and delete
- key must not exists in the target object on insert

Other types of validation can be alternatively implemented as services on the event transport layer, e.g. guaranteed delivery or de-dup filtering [Hohpe, Woolf 2004].

4.6. Event-feeded cSCD Implementation

The described implementation represents a particular instantiation of the presented event model in Section 4.5. The target object is implemented as a *Trans* table; natural key option is used; *Action* and *from attributes* are supported.

4.6.1. Development environment

Because the target DWH is also based on Oracle DBMS, we decided to keep the current development environment, i.e. developing the event feed cSCD solution as an Oracle PL/SQL package and easily call the functionality from ETL (e.g. Informatica Powercenter) mappings.

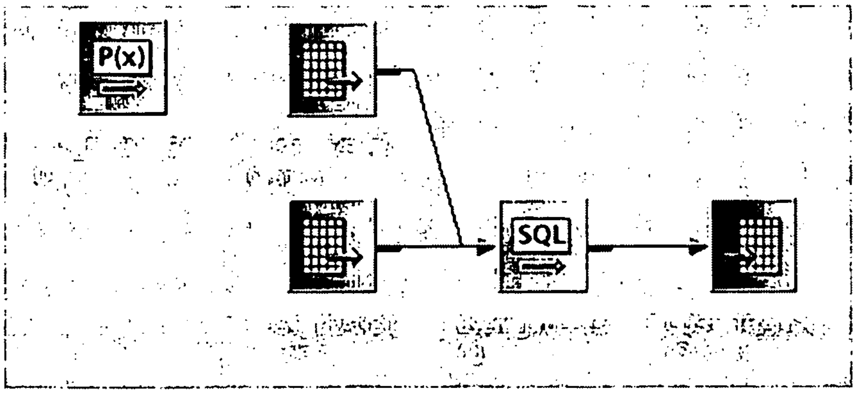


Figure 4.5. Informatica mapping diagram of event processing

The example mapping shown in Fig. 4.5 illustrates the source-target dependency (Note: the event transformation is done in UTL_EVENT_SCD). The source tables are the EVENT table (which contains only the new arriving events) and the old TRANS table, the target table is the new (updated) TRANS table (which keeps the full history of the dimension life cycle). With this approach, the requirement of retrieving the state of each entity at any time point is still supported (generated on demand) without keeping a series of state-based snapshots.

4.6.2. The UTL_EVENT_SCD Package

The package is used to trace the changing attributes of any (dimensional) table. It accepts a variant of parameters for the detailed configuration of the event-processing and -correction such as *traced entity* (via table name parameter), *correct option* (optional, mandatory or automatically), *refresh option* (incremental or from scratch), *filer criteria*.

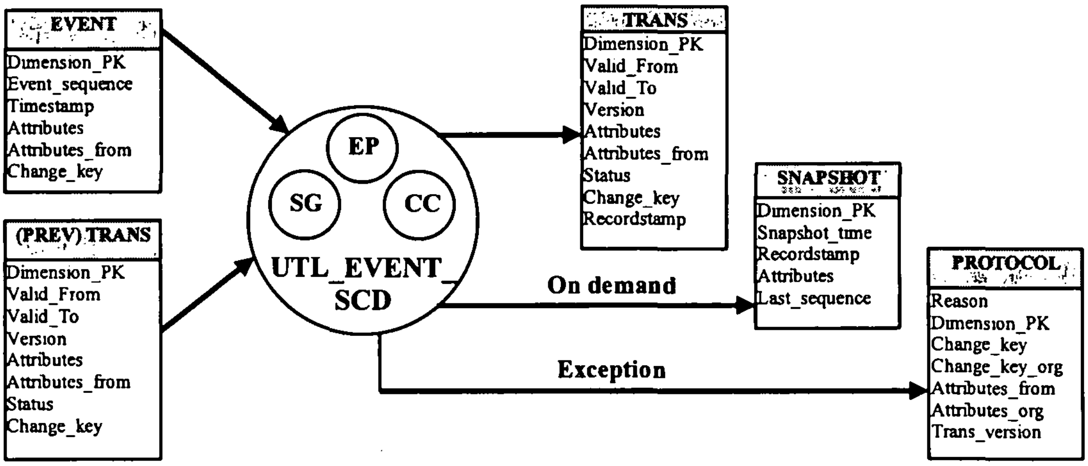


Figure 4.6. UTL_EVENT_SCD Package modules and its related tables

The package (Figure 4.6) contains 3 main modules: Event Processing (EP), Snapshot Generation (SG), and Consistency Checking (CC) providing the following options:

- *Validating* the events before *refreshing the historical transactions* of the entity instances (update TRANS table) with full historical tracing and versioning.
- *Providing the state information* at any point in time for any instance or subset of instances (generate SNAPSHOT table on demand)
- *Checking the consistency* between the entity state data of the legacy system and the data in DWH, and *solving the inconsistency issue.*

4.6.2.1. Event Processing (EP)

The Event Processing (EP) module is the main module of the package, it processes event data and refreshes the TRANS table as follows. It first accesses the event data, filters those which occured since the last refresh (i.e. those records which do not appear in TRANS or have different states with the current records in TRANS). The event validation then checks the events with automatic correction options to override some invalid events. This validation and correction processes are based on some useful attributes such as *change_key, attribute_from* or *sequence order.* The invalid or overridden events are kept in the PROTOCOL table. Table 4.6 enumerates the invalid cases and how to make the corrections or throw exceptions as invalid events.

Case	Description	Remark
Duplicated events	More than one events received with identical PK, timestamp and sequence number	Only one of those events is processed, all other are moved to protocol table (as there is no distinction, the processed event is randomly selected)
Duplicated time	More than one event have identical PK and time, but they are distinct in their sequence numbering	All events are processed in the order of sequence numbers. As the time of all events is identical, all but the last event are stored in the TRANS table as "semi invalid" i.e. validfrom > validto.
Insert after Insert	Two (or more) events with the same PK are inserted (change_key = 'I') subsequently with different timestamp.	**Correction:** The first event is "normally" processed. The following ones are corrected to Update (i.e. change_key is set to 'U') The change is written in the protocol table.
Update non existing PK	An update event is received without preceding insert event.	**Correction:** The update event is interpreted as an insert event (i.e. change_key is set to 'I') The change is written in the protocol table.
Delete non existing PK	A delete event is received without preceding insert event.	**Invalid:** The delete event is moved to protocol table
Change Key NULL or invalid	The event interface contains change key. An event is received with invalid or NULL change key	**Invalid:** The event is moved to protocol table

Table 4.6: EFD SCD Package – Corrections and Exceptions

Only the valid events are used to refresh the TRANS table. For each event data related to an entity instance, an equivalent transaction record in the TRANS table is created. If there are other events related to the same entity, the enhanced SCD (Section 4.3.5) is applied to keep trace over all transactions (with versions). The TRANS table thus contains the complete transaction history of dimension changes.

Examples: We apply the UTL_EVENT_SCD package to trace the Customer's attribute changes. Suppose that we have currently two customers Robert and Sonja until 7 am,14/02/2005. At 7:10, Robert informs that he changes his address from 20 Rennweg to 25 Favoritenstr. 7:12 am, a new customer Micheal has registered into the system, and Robert changes his tariff from type 1 to type 2 at 7:13. At 7:14, Sonja changes her tariff from type 2 to type 1. The UTL_EVENT_SCD package is executed at 7:15 to refresh the previous TRANS table. (Table 4.7)

CUST_TRANS (Before Event applying)

ID	Valid from	Valid to	Name	Address	Tariff	Address from	Tariff from	Recordstamp	version	Last version	Change key
1	14-02-2005 07:00:00	31-12-9999 00:00:00	Robert	20 Rennweg	T1			14-02-2005 07 00:00	1	Y	I
2	14-02-2005 07:00:00	31-12-9999 00:00:00	Sonja	15 Kargan	T2			14-02-2005 07 00:00	1	Y	I

CUST_EVENT

ID	Timestamp	Seq	Name	Address	Tariff	Address_from	Tariff_from	Change_key
1	14-02-2005 07:10:00	1	Robert	25 Favoriten	T1	20 Rennweg		U
3	14-02-2005 07:12:00	2	Micheal	10 Rathaus	T2			I
1	14-02-2005 07:13:00	3	Robert	25 Favoriten	T2		T1	U
2	14-02-2005 07:14:00	4	Sonja	15 Kargan	T1		T2	U

CUST_TRANS (After Event applying)

ID	Valid from	Valid to	Name	Address	Tariff	Address from	Tariff from	Recordstamp	version	Last version	Change key
1	14-02-2005 07:00 00	14-02-2005 07:09 59	Robert	20 Rennweg	T1			14-02-2005 07 15:00	1	N	I
1	14-02-2005 07.10 00	14-02-2005 07:12 59	Robert	25 Favoriten	T1	20 Rennweg		14-02-2005 07 15:00	2	N	U
1	14-02-2005 07 13 00	31-12-9999 00:00:00	Robert	25 Favoriten	T2		T1	14-02-2005 07:15:00	3	Y	U
2	14-02-2005 07 00.00	14-02-2005 07:13.59	Sonja	15 Kargan	T2			14-02-2005 07:15:00	1	N	I
2	14-02-2005 07 14:00	31-12-9999 00:00 00	Sonja	15 Kargan	T1		T2	14-02-2005 07:15:00	2	Y	U
3	14-02-2005 07:12:00	31-12-9999 00:00:00	Micheal	10 Rathaus	T2			14-02-2005 07 15:00	1	Y	I

Table 4.7: TRANS table refresh after UTL_EVENT_SCD package execution

The investigation of the performance behaviour based on the prototype implementation showed a near linear scalability of the processing-time per event with an average throughput of about 300 TRANS-records per second on a dimension with the cardinality of one million records. The minimum refresh period is about 3-4 seconds caused by process overheads. However, with the high number of events (e.g. over 20000 events), the more events accumulated, the less efficient of the event-SCD approach compared to the snapshot based SCD approach (see Figure 4.7).

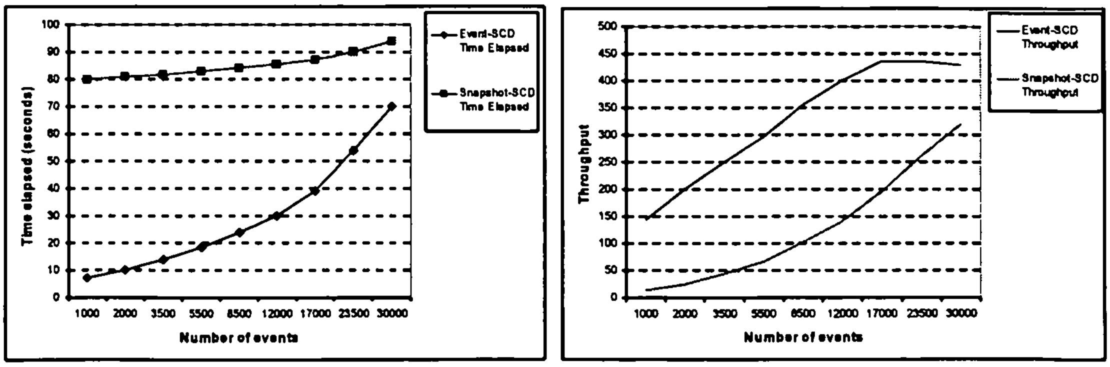


Figure 4.7. Elapsed processing time and performance throughput comparison between event-SCD and snapshot based SCD approach

4.6.2.2. On demand Snapshot Generation (SG)

Despite the series of snapshots is not kept as previously, the requirement to have a snapshot at one point in time for any subset of entity instances remains. From the TRANS table, we can rebuild these required snapshots. The package provides two options to generate a snapshot: (1) from scratch (Fig. 4.8) and (2) based on an existing snapshot, further referenced as based snapshot (Fig.4.9). The generated Customer snapshots at 7:00 and 7:15 are shown in Table 4.8.

```
CREATE TABLE CUST_SNAP AS
SELECT ID, i_timepoint as Snaptime, Name,Address, Tariff
FROM CUST_TRANS
WHERE CHANGE_KEY <> 'D' AND
i_timepoint BETWEEN VALIDFROM_T AND VALIDTO_T;
```

Figure 4.8. Create Snapshot from scratch (i_timepoint is the time point of the snapshot data)

```
CREATE TABLE CUST_SNAP AS
SELECT * FROM
(SELECT ID,i_timepoint as Snaptime, Name, Address, Tariff
 FROM CUST_TRANS WHERE CHANGE_KEY <> 'D' AND
 i_timepoint BETWEEN VALIDFROM_T AND VALIDTO_T
 AND VALIDFROM_T > v_prev_time
UNION ALL
 SELECT ID,i_timepoint as Snaptime, Name, Address, Tariff
 FROM BASED_CUST_SNAP
 WHERE ID NOT IN
      (SELECT ID FROM CUST_TRANS WHERE
       i_timepoint BETWEEN VALIDFROM_T AND VALIDTO_T
       AND VALIDFROM_T > v_prev_time)
 );
```

Figure 4.9. Create Snapshot from based snapshot (BASED_CUST_SNAP is the based snapshot table, v_prev_time is the time point of the based snapshot data)

SNAPSHOT at 14-02-2005 7:00				
ID	**Snaptime**	**Name**	**Address**	**Tariff**
1	14-02-2005 07:00:00	Robert	20 Rennweg	T1
2	14-02-2005 07:00:00	Sonja	15 Kargan	T2

SNAPSHOT at 14-02-2005 7:15				
ID	**Snaptime**	**Name**	**Address**	**Tariff**
1	14-02-2005 07:15:00	Robert	25 Favoriten	T2
2	14-02-2005 07:15:00	Sonja	15 Kargan	T1
3	14-02-2005 07:15:00	Micheal	10 Rathaus	T2

Table 4.8. SNAPSHOT tables generated at 7:00 and 7:15

4.6.2.3. Consistency checking and recovery (CC)

In the event based cSCD approach, an inconsistent state could be detected when we are able to access on a truthful snapshot source (usually provided from the legacy systems). The input requirements of this process are the mandatory truthful snapshot (Si, tj) table and the metadata parameters describing the record-structure. The consistency checking process compares a truthful snapshot(-part) taken on any subset of instances Si, at any point of time tj with the corresponding on demand snapshot (Si, tj) (see Section 4.6.2.2) which is temporary stored in a TEMP_SNAP (Si, tj) table. The found inconsistencies between the snapshots are applied again as new change events to correct the TRANS records.

4.7. Summary

In this chapter, we have introduced the event-feeded comprehensive Slowly Changing Dimension approach to overcome the limitation of existing SCD approaches and the snapshot based solution. The event feeded cSCD approach significantly reduces the number of records to be processed compared to the snapshot based approach. Besides, compared with the Kimball's classification of SCD [Kimball 2005a] we see that the SDC types 1,2 and 3 are only examples of possible instantiations of the proposed cSCD approach (SDC 1 and 2 respectively use the *Snap* object without and with *from attributes*; SDC 3 is based on *Trans* object without *from attributes).*

Although the target object was up to now considered as a dimension, this is not a limitation of the proposed model. A typical fact table can be described also as a versioned dimension (fast changing dimension), using the add-version update policy (each event creates a new record in the fact table) with appropriate validation e.g. to maintain a balance attribute.

Further more extending our model with summarizing stereotypes (e.g. add the actual value of the attribute to the previous value) the way is opened for describing running aggregates. On the other hand the correlation of system-dependent event-messages as an alternative to the join of dimensional snapshots needs further inspection.

Chapter 5. The Grid-based Zero Latency Data Stream Warehouse (GZLDSWH)

"The internet lets computers talk together, Grid computing lets computers work together."

Thomas Hawk, IBM General Manager of Grid Computing, 2002.

Continuous data streams are information sources in which data arrives in high-volume, in un-predictable rapid bursts. Processing data streams is a challenging task due to (1) the problem of random access to fast and large data streams using present storage technologies, (2) the exact answers from data streams are often too expensive.

In this chapter, we present a framework of building a Grid-based Zero-Latency Data Stream Warehouse (GZLDSWH) to overcome the resource limitation issues in data stream processing without using approximation approaches is specified. The GZLDSWH is built upon a set of Open Grid Service Infrastructure (OGSI)-based services and Globus Toolkit 3 (GT3) with the capability of capturing and storing continuous data streams, performing analytical processing, and reacting autonomously in near real time to some kinds of events based on well-established Knowledge Base.

The requirements of a GZLDSWH, its Grid-based conceptual architecture, and the operations of its service are described in this paper. Furthermore, several challenges and issues in building a GZLDSWH such as the Dynamic Collaboration Model between the grid services, the Analytical Model, the Design and Evaluation aspects of the Knowledge Base Rules are discussed and investigated.

5.1 Introduction

We are entering a new area of computing in today's complex world of computational power, very high speed machine processing capabilities, complex data storage methods, next generation telecommunications, new generation operating systems and services, and extremely advanced network services capabilities. At the same time, the number of emerging applications which handle various continuous data streams [Babcock et al.,2002; Franklin et al. 2002; Widom et al, 2003; Stonebraker et al. 2003; Lerner, Shasha 2003], such as sensor networks, networking flow analysis, telecommunication fraud detection, e-business and stock market online analysis, is growing.

It is demanding to conduct advanced analysis over fast and huge data streams to capture the trends, patterns, and exceptions. However, to fully extract the latent knowledge inherent within the huge data is still challenging effort because of the existing insufficient technology. Data streams arrive in high-volume, in un-predictable rapid bursts, and need to be processed continuously.

Processing data streams, due to the lack of resources, is challenging in the following two respects. On the one hand, random access to fast and large data streams is still impossible in the near future. On the other hand, the exact answers from data streams are often too expensive. Therefore, the approximate query results [Babcock et al. 2002; Guha, Koudas 2002; Dobra et al. 2002; Tucker et al. 2003; Widom et al, 2003; Kim, Park 2005] are still acceptable because there is no existing computing capacity powerful enough to produce exact analytical result on continuous data streams.

The significant increased data volume of information manipulated in several domains has affected Data Warehousing (DWH) and Business Intelligence (BI) applications. Data Warehousing and Business Intelligence applications are normally used for strategic planning and decision making. However, existing DWH technologies and tools (e.g. ETL, OLAP) often rely on the assumption that data in the DWH can lag for a tolerable time span (e.g. on a few hours) behind the actual operational data and the decisions are based upon the analytical process on that "window on the past". For many business situations, especially, in data stream analysis, this decision making approach is too slow due to the fast pace of today's business. Today's decisions in the real world thus need more real-time characteristics and consequently DWH, BI, ETL tools and OLAP systems are quickly beginning to incorporate real-time data. A new ETL approach using widely accepted Web technologies has recently been announced [Schlesinger et al., 2005].

Starting from the concept of a Zero-Latency Data Warehouse (ZLDWH) [Bruckner 2002; Tho ,Tjoa 2003], we extend the system to tackle continuous data streams with the capability of capturing data streams, performing analytical processes, and reacting automatically to some kinds of events based on well-established knowledge.

We do not follow the approximation approach; instead, we capture and store all data streams continuously while performing the analytical processing. Obviously, such systems require a very high computing capacity which is capable of huge storage and computing resources. Fortunately, in the last few years we have witnessed the emergence of Grid Computing [Foster et al. 2001; Fellenstein et al. 2003] as an important new technology accepted by a remarkable number of scientific and engineering fields and by many commercial and industrial enterprises. Grid Computing provides highly scalable, secure, and extremely high performance mechanisms for discovering and negotiating access to remote computing resources in a seamless manner. With the Grid as "a flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources", our requirements seem to be satisfied. Furthermore, due to unpredictable characteristics of data streams, Grid technology is more convincing due to its flexibility.

This chapter describes our ongoing work in developing the GZLDSWH built upon a set of OGSA-based grid services and GT3 toolkit. The GZLDSWH is composed of several specific Grid services for capturing, storing, performing analysis on continuous data streams and issuing relevant actions or notifications.

The chapter is organized as follow. First, we describe the basic concepts and techniques for building the system which are Open Grid Service Architecture (OGSA), Open Grid Service Infrastructure (OGSI) and the Globus Toolkit version 3 (GT3). Next, the GZLDSWH system overview will be described. We then discuss several approaches in building the GZLDSWH and specify its conceptual architecture. The next section describes the operation of GZLDSWH in conducting analysis processes and reaction on continuous stream. Thereafter, we focus on the main contributions of the chapter:

- A model that describes the required dynamic collaboration of the grid services.
- A Grid-based OLAP Cube management service.
- The Knowledge Base rules and rule evaluation process.
- A guideline on how to increase performance when nodes fail and need to roll back to previous checkpoints.

Our ongoing prototype implementation will be described briefly afterwards. Finally, we give the summary and mention the future work.

5.2. Open Grid Service Architecture (OGSA), Open Grid Service Infrastructure (OGSI) and Globus Toolkit 3 (GT3)

In recent years, Grid computing [Foster et al. 2001; Fellenstein et al.2003; Foster, Grossman 2003] is emerging as the best solution to the problems posed by the massive computational and data handling requirements. Starting from the concept of linking super computers to benefit from the massive parallelism for computation needs, Grid's focus has recently shifted to more data-intensive applications where significant processing is conducted on very large amounts of data.

New-generation Grid technologies are evolving towards an Open Grid Services Architecture (OGSA) [OGSA 2003] in which a Grid provides an extensible set of services that virtual organizations can aggregate in various ways. The development of OGSA technical specification is in progress within the Global Grid Forum covered by the tasks called the Open Grid Services Infrastructure (OGSI) with Globus Toolkit 3 (GT3) [Sotomayor 2004]. Figure 5.1 summarizes these major concepts in the Grid Services world. The next subsections will describes them in the more details.

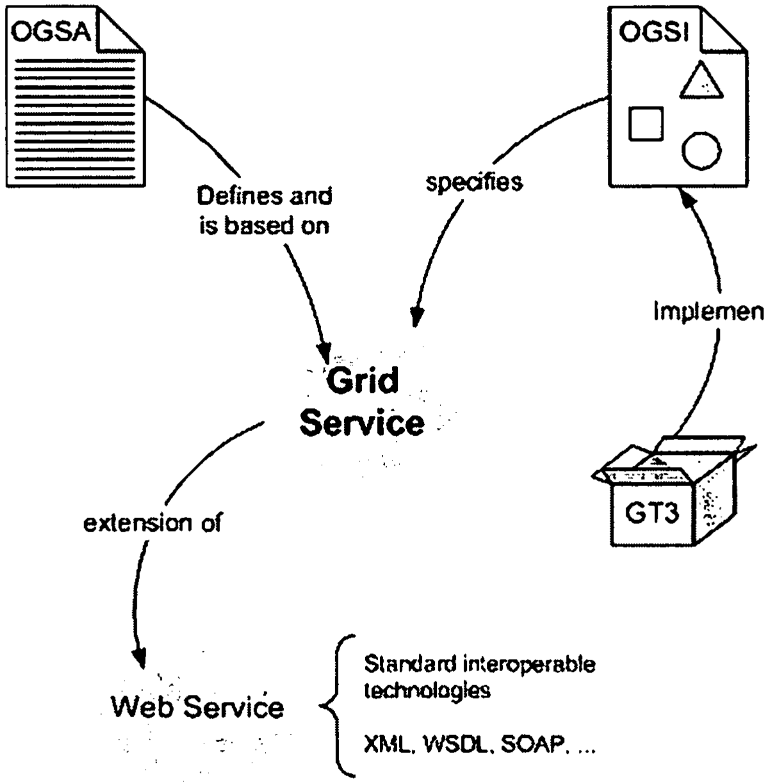


Figure 5.1. Key concepts of the Grid service world

5.2.1 Web Services

As mention in Section 3.3, Service Oriented Architecture (SOA) has emerged as a direct consequence of specific business and technology drivers that have materialized over the past decade. It's an approach to building software systems that is based on loosely coupled components (services) that have been described in a uniform way and that can be discovered and composed. Web services, a standards-based, XML-centric, represent one important approach to realizing an SOA. The Web Services Model allows applications to communicate using agreed, wide-spread standards and protocols independent of their implementation and platform. The World Wide Web Consortium

(W3C), which has managed the evolution of the SOAP and WSDL specifications, defines Web services as follows:

"A Web *service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with XML serialization in conjunction with other Web-related standards."* [11]

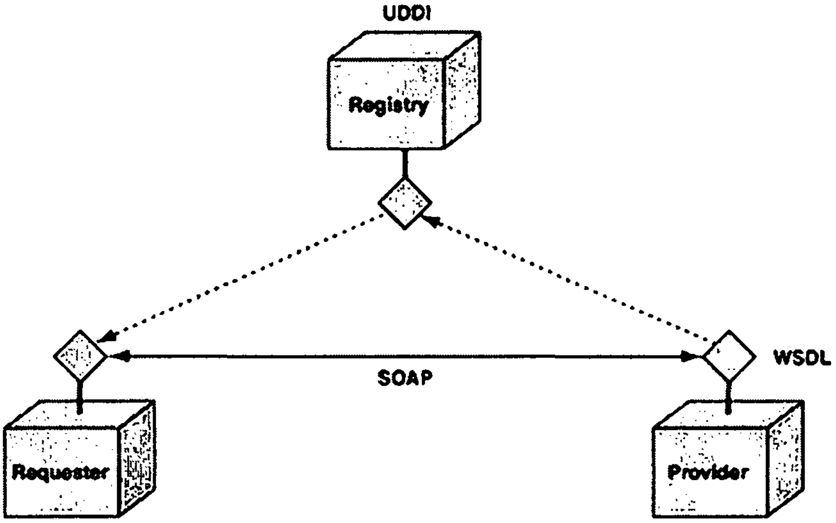


Figure 5.2. Simple Web Service architecture

The parts of the simple Web Service architecture are presented in Figure 5.2 which includes *service discovery, service description, service invocation* and *service transportation.*

Service Discovery: This part of the architecture allows us to find Web Services which meet certain requirements. This part is usually handled by UDDI (Universal Description, Discovery, and Integration) [Web Service 2005].

Service Description: The Web Services Description Language (WSDL) [Web Service 2005] is an XML format for describing Web services as a set of endpoints operating on messages containing either document-oriented (messaging) or RPC payloads. The main elements in a WSDL document are schematically illustrated in Table 5.1. As a broad overview, the various components in a WSDL document define services as collections of network end-points, or ports.

types	data types used within WSDL documents usually defined using XML Schema
message	an abstract, typed definition of the data being communicated
portType	contains one or more descriptions of operations supported by a service
binding	a concrete protocol and data format specification for a particular port type
port	a single endpoint defined as a combination of a binding and a network address
service	a collection of related end points

Table 5.1. WSDL 1.1 elements

A port is an association of a network address with a reusable, network-independent, binding. A binding is the association of a concrete network protocol and data format specification – for example HTTP and SOAP – with a portType. A portType is a collection of operations each of which handles messages. Each message is constructed from a set of application-specific types. A new WSDL standard draft (version 1.2) is also in existence but has yet to be accepted as a standard. This draft provides features required for specifying Grid services including the notion of an open content model and inheritance of portTypes, features which are currently provided via an extension to WSDL, called GWSDL [GWSDL 2003]. GWSDL is extensible to allow description of endpoints and the concrete representation of their messages for a variety of different message formats and network protocols.

Service Invocation: Invoking a Web Service involves passing messages between the client and the server. SOAP (Simple Object Access Protocol) [Web Service 2005] specifies how we should format requests to the server, and how the server should format its responses. In theory, we could use other service invocation languages (such as XML-RPC). However, SOAP is by far the most popular choice for Web Services.

Transport: Finally, all these messages must be transmitted somehow between the server and the client. The protocol of choice for this part of the architecture is HTTP (Hyper-Text Transfer Protocol), the same protocol used to access conventional web pages on the Internet. Again, in theory we could be able to use other protocols e.g. FTP (File Transfer Protocol) or SMTP (Simple Mail Transfer Protocol), but HTTP is currently the most used one.

5.2.2. OGSA versus OGSI - what is a Grid Service

In both e-business and e-science, we often need to integrate services across distributed, heterogeneous, dynamic Virtual Organization 's formed from the disparate resources within a single enterprise and/or from external resource sharing and service provider relationships. This integration can be technically challenging because of the need to achieve various qualities of service when running on top of different native platforms. The Open Grid Services Architecture addresses these challenges by building on concepts and technologies from the Grid and Web services communities.

The Open Grid Services Architecture (OGSA), developed by The Global Grid Forum (GGF), aims to define a common, standard, and open architecture for grid-based applications. The goal of OGSA is to standardize practically all the services one finds in a grid application (job management services, resource management services, security services, etc.) by specifying a set of standard interfaces for these services.

As depicted in Figure 5.1., Grid Services are defined by OGSA. OGSA defines what Grid Services are, what they should be capable of, what types of technologies they should be based on, but doesn't give a technical and detailed specification. A Grid Service is simply a Web Service with a lot of extensions that make it adequate for a grid-based application (and, in particular, for OGSA). In the diagram: **Grid Services are an extension of Web Services.**

However, OGSA alone doesn't go into much detail when describing Grid Services. It basically outlines what a Grid Service should have (that Web Services don't) but little else. That is why OGSA spawned another standard called the Open Grid Services Infrastructure (OGSI, also developed by The Global Grid Forum) which gives a formal and technical specification of what a Grid Service is. Therefore, Grid Services are specified by OGSI [Sotomayor 2004]. The Open Grid Services Infrastructure is a formal and technical specification of the concepts described in OGSA, including Grid Services. The Globus Toolkit 3 (GT3) is an implementation of OGSI. GT3 is a usable implementation of everything that is specified in OGSI (and, therefore, of everything that is defined in OGSA).

We'll take a brief look at the main improvements introduced in OGSI to specify a Grid Service as "*a stateful web service following additional conventions and restrictions and implementing a set of well defined interfaces*"

Stateful and potentially transient services: Web services are *stateless*, which means there is no record of previous interactions and each interaction request has to be handled based entirely on information that comes with it. Besides, Web services are non-transient, which means that they outlive all their clients. After one client is done using a Web Service, all the information the Web Service is remembering could be accessed by the next clients. In fact, while one client is using the Web Service, another client could access the Web Service and potentially mess up the first client's operations. This certainly isn't a very elegant solution.

Grid Services solve both problems by allowing programmers to use a *factory/instance* approach to Web Services. Instead of having one big stateless service shared by all users, there is a central Service Factory in charge of maintaining a bunch of service instances, each instance is created (and destroyed afterward) for a specific client's request.

Service Data allows us to easily include a set of structured data to any service, which can then be accessed directly through its interface. Service Data which could be included in a service will fall into one of two categories:

- **State information**: Provides information on the current state of the service, such as operation results, intermediate results, runtime information, etc.
- **Service metadata**: Information on the service itself, such as system data, supported interfaces, cost of using the service, etc

Notification: A Grid Service can be configured to be a *notification source*, and certain clients to be *notification sinks* (or subscribers). This means that if a change occurs in the Grid Service, that change is *notified* to all the subscribers

Service Groups: Any service can be configured to act as a *service group* which aggregates other services. This is the base of more powerful directory services (such as GT3's IndexService) which allow us to group different services together and access them through a single point of entry (the service group).

Port Type Extension: The interface is usually called *portType* (due to a WSDL tag of the same name). A normal Web Service can have only one portType. Grid Services, on the other hand, support *portType extension*, which means we can define a portType as an extension of a previously existing portType.

GSH & GSR: A "Grid Service URI" is called the *Grid Service Handle*, or simply GSH. Each GSH must be unique. There cannot be two Grid Services with the same GSH. The only problem with the GSH is that it tells us *where* the Grid Service is, but doesn't give us any information on *how* to communicate with the Grid Service (what methods it has, what kind of messages it accepts/receives, etc.). To do this, we need the *Grid Service Reference*, or GSR. In theory, the GSR can take many different forms, but since we will usually use SOAP to communicate with a Grid Service, the GSR will be a WSDL file (remember that WSDL *describes* a Web Service: what methods it has, etc.).

5.2.3. Globus Toolkit 3 (GT3)

The Globus Toolkit is an open source software project initiated in 1996 under the leadership of Argonne, ISI and the University of Chicago. Its declared objective is to enable people to "*share computing power, databases, and other tools securely online across corporate, institutional and geographic boundaries without sacrificing local autonomy*" [Globus Toolkit 2003].

The Globus Toolkit is now developed by the Globus Alliance. The evolution of Globus started with the first release in 1998 providing resource management and information services (GRAM and MDS) later on extended by version 2.0 in 2001 by basic data management (GridFTP), packaging and reliability facilities. With the emergence of the Open Grid Service Architecture a new era began. Released in June 2003 the Globus Toolkit 3 (GT3) combines the efforts of both the Grid and web service community and is the first major implementation of the OGSA/OGSI idea and specification.

Figure 5.3. describes the Architecture of the Globus Toolkit 3. The OGSI (i.e. "Grid Services") is the "**GT3 Core**" layer written entirely in Java and is shipped with a standalone container. This layer consists of two major components, the web service engine and the globus container. The Globus container itself is built on the web service engine and transforms stateless web services into stateful Grid services. Once a service is created via a OGSI factory, the container assigns a unique GSH and registers it within a repository for all service instances. The container is responsible for identifying services, invoking methods, activation and passivation of instances, GSH-to-GSR resolving and instance persistence.

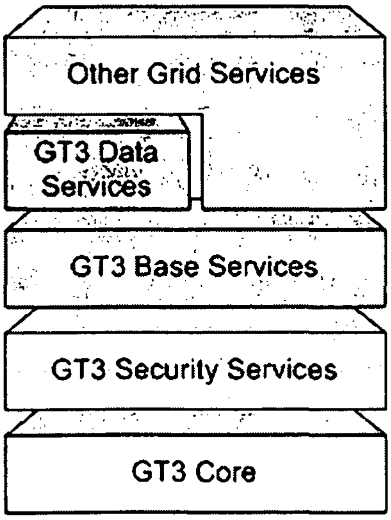


Figure 5.3. Globus Toolkit 3 Architecture

GT3 Security Services restrict access to the Grid Services, so only authorized clients can use them. Besides the usual security methods (putting the web server behind a firewall, etc.), GT3 provides one more layer of security with technologies such as SSL and X.509 digital certificates.

GT3 Base Services: provides a lot of the following interesting services:

- **Managed Job Service** to check on the progress of the operation periodically and to trace the task situation. It allow us to control service operation (i.e. pause a service, stop it, etc).
- **Index Service** for query Grid Service to find what Grid Service meets certain requirements (similar to UDDI in Web Service).
- **Reliable File Transfer (RFT) Service**: allows us to perform large file transfers between the client and the Grid Service. Furthermore, RFT guarantees the transfer will be reliable (hence its name). For example, if a file transfer is interrupted (due to a network failure, for example), RFT allows us to restart the file transfer from the moment it broke down, instead of starting all over again.

GT3 Data Services: This layer includes Replica Management, which is very useful in applications that have to deal with very big sets of data. When working with large amount of data, we're usually not interested in downloading the whole thing, we just want to work with a small part of all that data. Replica Management keeps track of those subsets of data we will be working with.

Other Grid Services: Other non-GT3 services can run on top of the GT3 Architecture. Our Gird services built up the GZLDSWH are belongs to this layer.

5.3 The Grid-based Zero-Latency Data Stream Warehouse (GZLDSWH) Over View

Starting with the idea of building a Zero-Latency analytical environment dealing with heterogeneous data sources, we extend the system to conduct analysis on continuous data streams. A ZLDWH [Bruckner 2002; Tho,Tjoa 2003] aims to significantly decrease the time to react to business events allowing the organizations to deliver relevant information as fast as possible to applications which need a near real-time action to new information captured by an organization's information system. It enables analysis across corporate data sources while still continuously update the new arriving data and notifies the handling of actionable recommendations, alerts or notifications. In Data Stream applications, events take the form of continuous data streams.

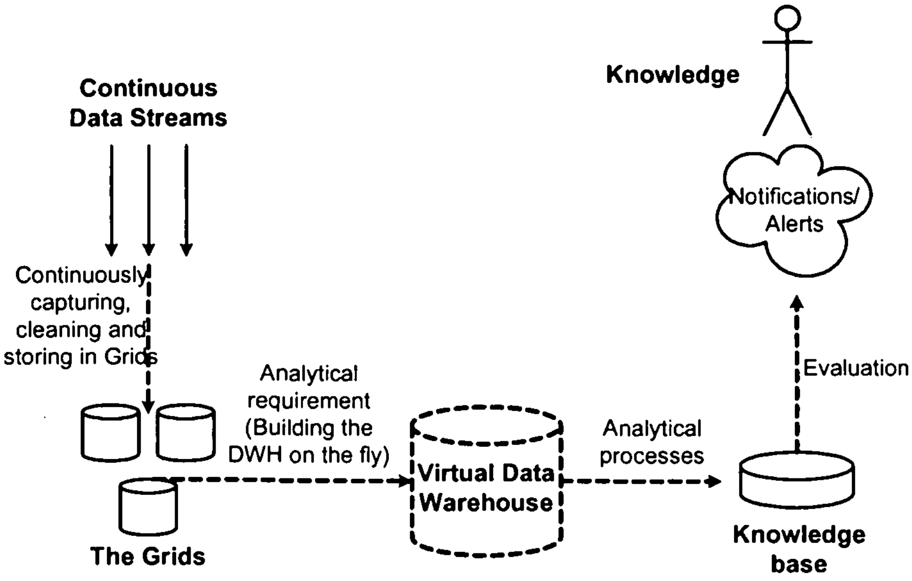


Figure 5.4. The overall process of the Grid-enabled Zero-Latency Data Warehouse System (GZLDSWH)

The exact analysis results on these data stream events are very expensive because they require high computing capacity which is capable of huge storage and computing resources. Therefore, a Grid-based approach is applied in ZLDWH to tackle the lack of resources for continuous data stream processing. Figure 5.4. depicts significant phases throughout the overall process of such GZLDSWH. Continuous data streams will be captured, cleaned and stored within the Grids. Whenever the analytical processes need to be executed, immediately after the arrival of new data or based on predefined timely scheduling, the virtual Data Warehouse will be built on the fly [Foster,Grossman 2003] from data sources stored in the Grid nodes. Obviously, this approach is not concerned with traditional incremental updating issues in Data Warehouse because the virtual DWH is built from scratch using the most current data. Analytical processes will then be executed on such virtual DWH and the results will be evaluated with the use of the Knowledge Base. Finally, dependent on the specification of the Knowledge Base, the system sends notifications, alerts or recommendations to the users.

There are a significant number of applications in which the conducting of analytical processing on continuous data streams is necessary for detecting trends and abnormal activities. The following scenario describes an example in Mobile Phone Fraud Detection although the usage of such system could also be applied in other time critical decision support applications.

Expert users defined rule specifications for fraud detection, e.g. "*if an international mobile call lasts over 1 hour it will be considered as a fraudulent call*". The rule could also be more complex such as "*an international mobile call from Austria to China of a certain customer lasts over 30 minutes will not be considered as a fraud, if its duration is not over 1.5 times of his/her average call duration from Europe to Asia within the last 3 months, otherwise, it is considered as a fraudulent call*". These rules are stored in the Knowledge Base and are referenced by the Rule Evaluation module before it makes the final decisions. Expert users could also specify how and when the system operates to detect the Fraud situation by submitting the pre-defined plan workflow. In this plan, the experts specify the order of module executions and the time point when they should be executed. The whole system operation will then be monitored and controlled by the Workflow Control module which follows the pre-defined plan.

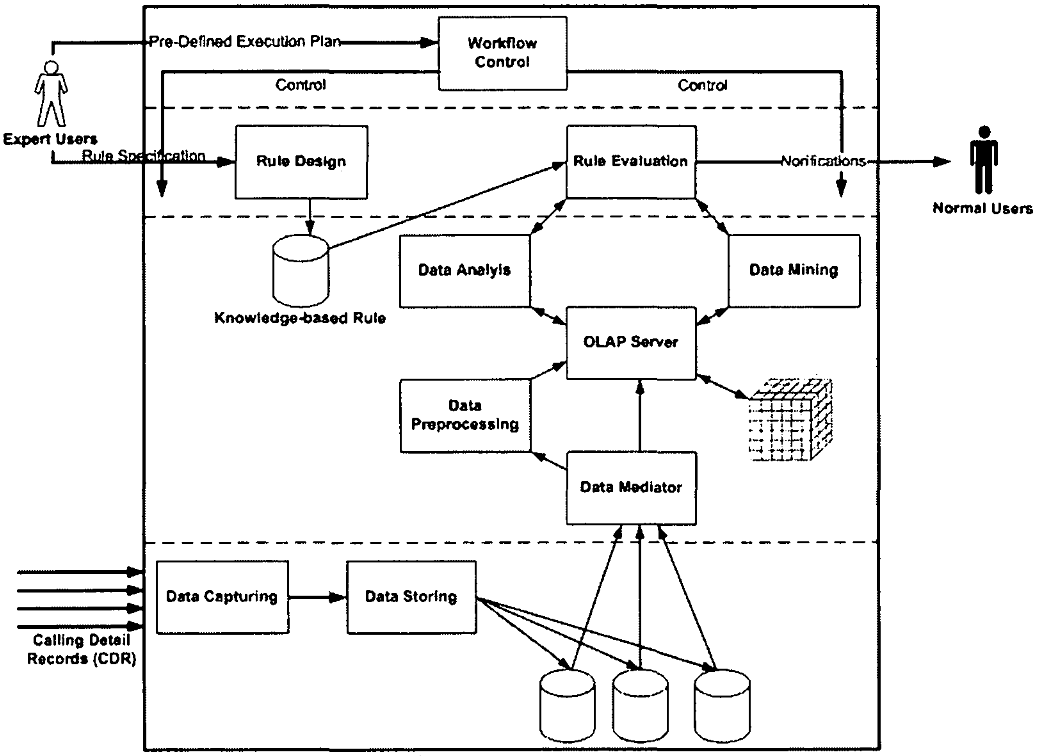


Figure 5.5. Mobile phone fraud detection

When the end-user makes a phone call, the Call Detail Records (CDRs) are issued continuously as continuous data streams. Because of the special characteristics of continuous data streams, these CDRs must be captured and stored in timely fashion. The data storage is heterogeneous and geographically distributed in several grid nodes.

For supporting analytical processing, the virtual DWH is built on the fly from the heterogeneous, distributed data sources as follows. The OLAP server accesses the raw data items from multiple Grid nodes and creates the pre-aggregated data cubes. The Data Mediator allows the OLAP server to access the distributed, heterogeneous data sources as if they were local data sources. In some situations, Data Preprocessing could be necessary to clean the data, standardize the data, or transform data into the common format before storing into the OLAP cubes.

When the virtual DWH data is available, the OLAP server accepts the queries from Data Analysis or Data Mining tools, executes these queries and returns the results. To detect a fraud situation, it is necessary to analyze the CDRs at multiple levels with different dimensions and in a variety of time ranges e.g. to calculate the average international mobile call length from Europe to Asia within the last 3 months of a certain customer. The OLAP server therefore accepts analytical commands such as "drill up", "drill down", "slide and dice" and performs these operations on the data cubes.

The results are returned to the Rule Evaluation module, and it will access the Knowledge Base to evaluate the rules. If some pre-defined criteria in the Knowledge Base are satisfied, the Rule Evaluation module performs suitable actions, for instance, sending of notifications to the users, or stopping the telephone services. Particularly, if the rule criteria lead to an ambiguous situation, the Rule Evaluation module would issue other analytical queries to further investigate the data.

The above scenario highlights following requirements for a GZLDSWH system:

- **Knowledge-Base Rule Support:** The final decisions, e.g. whether a phone call is fraudulent or not, is based on the set of Knowledge-Base rules specified by expert users. The Knowledge-Base Rules preserve the experiences, and knowledge to drive the decision support process. It is necessary to develop a component that allows the expert users to easily manipulate the existing rules in the Knowledge Base i.e. insert, delete or update rules. The autonomous validation and consistency checking of these rules should also be considered. The facilities of auto-inference and auto-learning are also the challenges of Knowledge-rule management.

- **Multi-level Analysis Support:** In order to evaluate the rules, it is necessary to conduct analytical process on the multi-dimensional historical transaction of customer CDRs to identify the customer's pattern. Besides, in some situations, there is the uncertainty or ambiguity in evaluating the rules during online analysis process. Then it is necessary to conduct analytical process on the whole CDRs at multiple levels to define the final decision. CDR streams thus need to be stored without loss within the Grid. DWH repositories and online analytical processing (OLAP) cubes are built on the fly from these Grid node's data. The significant parts are: (1) the creation and maintenance the OLAP Cube, (2) the OLAP query engine that executes analytical queries on OLAP data, and optionally (3) the OLAP Data Mining Engine for execution of the on-line analytical mining (OLAM) algorithms.

- **Automated Reaction Support**: Whenever the Fraud is detected via CDRs analysis and rule-based evaluation, the system must have the ability to issue automatically the relevant actions respective to the Fraud prevention methods. That could be the alarm or recommendation message sent to the mobile phone customers, or it could also be the service disconnection to prevent more fraudulent activities. However, this automatic reaction must follow the rules stored in pre-defined Knowledge Base.
- **Distributed, Heterogeneous Data Acquisition:** The ability to use different kinds of data sources and data-warehouse repositories is mandatory for Data Grid systems. Therefore, the Data Mediator layer is usually used to transparently access the heterogeneous Grid-based data sources.
- **Transparent User interface:** The system should be able to provide the users an interface which is easy and transparent to the Grid, Network, and Location. Customers can access the system to register the services or send their feedback from anywhere using a variety of devices (mobile, PDA, laptop, etc.) without considering the complexity of system architecture. Expert users use the designed interface to specify the rules without considering the Grid structure, network feature and the physical details or location of data sources.
- **Flexibility and Open Architecture:** The essence of Grid is heterogeneity. The GZLDSWH thus should allow the integration of components that are heterogeneous i.e. written in different programming languages or are optimized for different platforms. Technical and contextual changes to the underlying data sources, interface implementations, libraries, etc. must not affect the module operability. The components and features shall be easily extensible allowing for plug-ins to be executed. Furthermore the architecture must be open to integrate specialized third-party toolkits.
- **Grid Environment Information and Monitoring:** The Grid environment always varies significantly during the runtime. Therefore the system should be able to invoke and execute its component dynamically dependent on the runtime environment. For this purpose, the Grid environment information should be monitored by a special component. It maintains the knowledge about the resources available on the Grid, their capacity and current utilization. It should be easily query-able, highly scalable and have to react quickly even under heavy load.
- **Dynamic Component Invocation and Execution:** As described in the scenario, the component execution flows are pre-defined by expert users. However, the Grid environment always varies significantly during the runtime. Therefore the system should be able to invoke and execute its components dynamically depending on the runtime environment.

- **Integrity and Availability:** Assuming that all grid nodes are trusted entities, there still is a need to implement integrity checks and address non-availability. If nodes cannot be trusted, data integrity checks are required. Kamvar et al. [Kamvar et al. 2003] provide an excellent overview of requirements and also shows how these issues are addressed in P2P networks. Their solutions can also be adapted to Grid computing if one is prepared to accept the computational overhead.

There are a lot other requirements such as security, high availability, scalability and performance but they will be planned for the future work.

5.4. The Framework for building a GZLDSWH

Based on the requirements discussed above in section 5.3., we consider three following approaches for building the GZLDSWH system.

5.4.1 Building the whole system from scratch

Following this way, we have to develop not only the components of the system, but also have to deal with many other issues in Grid Computing such as protocol for communication, message passing mechanism, resource management, scheduling, life cycle management, etc. The only advantage is that we do not have to obey any specifications and freely apply any container technology such as J2EE Container, .Net Container to control and manage our components. However, a lot of work must be done, and the system then can not be integrated easily with the Grid Community standard toolkits and specifications. We thus do not choose this approach to develop our system.

5.4.2. Following the OGSA, OGSI implemented by Globus Toolkit 3

The second option is to build the system using the Open Grid Service Architecture (OGSA) [OGSA 2003] based on the Open Grid Service Infrastructure specification (OGSI) and Globus Toolkit version 3 (GT3) [Globus Toolkit 2003] discussed in Section 5.2.2 and 5.2.3.

The Open Grid Service Architecture (OGSA) represents the convergence of Web service and Grid computing technologies with the aim of describing the next generation of Grid Architecture in which the components are exchangeable on different layers. Consequently, in OGSA, all kinds of storage and computational resources, components, databases, file systems, etc are exposed as services. This approach has the big advantage that the upper-layer components have to be concerned only on a small amount of interfaces because the implementation is hidden behind the interface.

As mentioned previously, the Open Grid Service Infrastructure (OGSI) defines specifications with many interesting features such as factory service discovery, instance creation, invocation, lifetime management, notification and manageability. The Globus Toolkit 3 (GT3) implements most of these OGSI specifications and provides us the infrastructure components for resource management, monitoring and discovery, security, information services, data and file management, communication and fault detection. Some features still missing are a resource broker, load balancing and scheduling functionality. However, if we develop the system on top of the OGSI and GT3 toolkit, some of our above mentioned requirements are satisfied, such as Flexibility, Open Architecture, Grid-Network-Location Transparency and Grid Environment Information. Hence we chose this approach to develop the GZLDSWH system.

5.4.3. WS-Resource Framework

Around the same time when the OGSI work has been progressing, the Web services architecture has evolved as well, for example, the definition of WSDL 2.0 and the release of new draft specifications such as WS-Addressing [Globus Alliance 2004]. WS-Resource Framework can be viewed as a straightforward re-factoring of the concepts and interfaces developed in the OGSI V1.0 specification in a manner that exploits recent developments in Web services architecture. The difference is that WSRF uses different constructs for modelling the stateful resources and the stateless Web services, while OGSI uses the same construct (the service instance). The WS-Resource Framework is still new and on the way to become a maturity standard, the Globus Toolkit version 4 which integrates this framework, is expected to release at the end of 2004. Therefore, in the future, the GZLDSWH system developed upon OGSI/GT3 should be based on the WS-Resource Framework.

5.4.5. The Grid-based Conceptual Architecture of GZLDSWH

Each phase of the process in Figure 5.4. includes several tasks such as capturing, storing data stream, building OLAP cube, conducting multi-dimensional analysis etc. Sharing the same approach with GridMiner project [Tjoa, Brezany 2003], each particular task will be realized as a grid service. The GZLDSWH is thus composed of several specific Grid services for capturing, storing, performing analysis on continuous data streams and issuing relevant actions or notifications reflecting the trends or patterns of the data streams. As we have discussed, these grid services are built on top of OGSI and GT3 toolkit and can be grouped into several layers based on their functionality as described in Figure. 5.6.

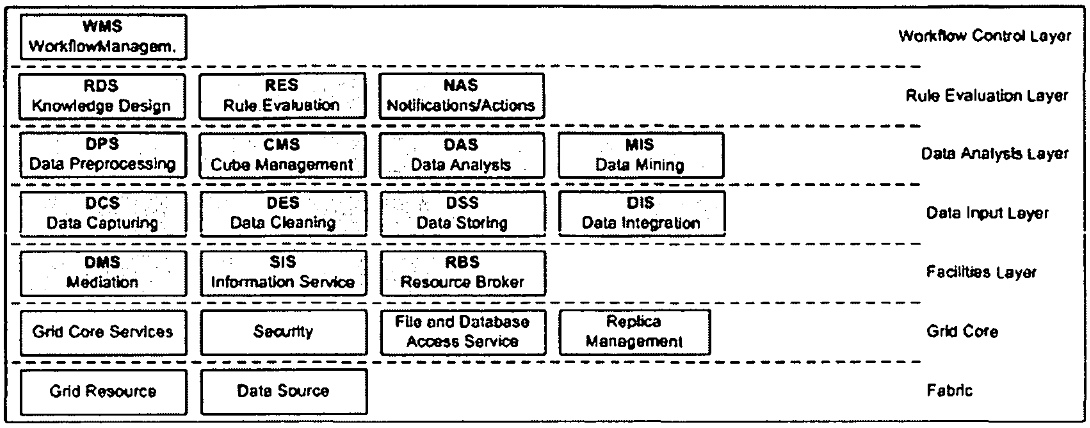


Figure 5.6. The Service Components of GZLDSWH

The Fabric and Core Layer: The services in this layer are provided by the Globus Toolkit 3 which enables most of the basic operations and communication between the Grid Services. We do not need to develop these fabric services and Grid core services (the white box in Figure 5.6)

The Facilities Layer: This layer provides some services for monitoring the Grid environment, scheduling, load-balancing and so on. It also provides the transparency of resources, network and location to the heterogeneous data sources. The services in this layer include the following:

- **System Information Service (SIS).** The SIS is a vital service within every grid infrastructure, providing static and dynamic information on available grid resources. It is a specialized implementation enabling specific decision making and monitoring processes.
- **Resource Broker Service (RBS).** The Resource Broker is used to find best-fitting resources for resource allocation, such as match-making of the requests and resources. In our system, the Resource Broker is used as a reference for the Workflow Engine in dynamic discovery, creation, binding and invocation of other available services instances in the runtime.
- **Data Mediation Service (DMS).** The DMS provides a single virtual data source having the same client interfaces as classical grid data sources but it integrates data from multiple heterogeneous federated sources from several Grid nodes. This service simplifies access requests and implements transparency for heterogeneous data sources.

The Data Input Layer: This layer provides the services for capturing, cleaning, and storing data within the Grid. The following services belong to this layer:

- **Data Capturing Service (DCS).** The Data Capturing Service works closely with the stream sources such as sensors, cameras, satellites etc., for capturing data streams in the limited time without data loss.
- **Data Cleaning Service (DES).** In some special situation, the Stream Data should be cleaned prior to being stored in the Grid. Therefore, the Data Cleaning Service is an optional service.
- **Data Storing Service (DSS).** The Data Storing Service resides in each Grid node and is responsible for storing the lossless data streams. This service could also convert the data into an appropriate format for the local data source.
- **Data Integration Service (DIS).** This service is responsible for secure, reliable, efficient management and operation of the necessary data transfers within the grid environments.

The Data Analysis Layer: The aim of the Data Analysis layer is to support the analytical process on data stream stored somewhere within the Grid. This layer contains the following services.

- **Data Preprocessing Service (DPS).** The Data Preprocessing Service performs several pre-processing activities such as data cleaning, normalization, selection, reduction, transformation, etc. before storing the data into the OLAP cube.
- **OLAP Cube Management Service (CMS).** The OLAP Cube Management Service is one of the major components of the system. This service creates distributed OLAP cubes from several data sources stored at specified Grids nodes. After the initial cube creation, the service can be used for cube interaction and life cycle management.
- **Data Analysis Service (DAS).** The Data Analysis Service is another major component of the system. It works very closely with the OLAP Cube Management Service and performs analytical process by sending queries and commands such as “drill up”, “drill down”, “slide and dice”. It thus allows analyses of datasets at different levels of abstraction. The output of the Data Analysis Service is the analysis result which will then be evaluated by the Knowledge Base for further actions.
- **Data Mining Service (MIS).** The Data Mining Service is created as an extensible framework providing necessary data mining algorithms making it convenient for the related application developers to easily plug in their algorithms and tools.

The Rule Evaluation Layer: This layer contains the services supporting the rule specification and rule evaluation. The services in this layer provide the Interface so that the expert users can interact with the system to specify the rules. In addition, normal users can receive notifications, alarms and recommendations from the system. The following services are available in this layer:

- **Knowledge Base Rule Design Service (RDS).** This service allows the expert users to specify the rules stored in the Knowledge Base of the system. The Knowledge Base Rules embed ECA rules consisting of an event, condition and action part. However, it carries out complex OLAP analyses on DWH data instead of evaluating simple conditions as compared to ECA rules in OLTP system.
- **Rule Evaluation Service (RES).** The RES service takes the analytical results from the DAS, evaluates these results against the Knowledge Base rules, and finally takes suitable actions. It could invoke the NAS to perform suitable actions if the rule criteria are satisfied. It could also invoke the DAS service or the Data Mining service (MIS) to perform further analytical process when the rule criteria are too ambiguous to make the final decision.
- **Notification/Action Service (NAS).** The NAS service manages all possible actions of the GZLDSWH system such as issuing notifications, alarms, and recommendations to the users to inform about the abnormal trends. In addition, it invokes the Analysis service when receiving the local update message.

The Workflow Control Layer: contains the Workflow Management Service that controls the dynamic service invocations, executions, and destructions.

- **Workflow Management Service (WMS).** This service allows expert users to specify the logical workflow of system activities and execute complex and highly dynamic workflows for several heterogeneous grid services. The WMS service dynamically controls service execution, service termination and communication, etc. dependent on the Grid environment. Because the Grid environment significantly varies over time, the Dynamic Workflow Control Service is an extremely important component in the system.

5.5. The operation of GZLDSWH

Within the Grids environment as described in Figure 5.7, there are one Master node and several child nodes (Node 1, 2..., Node N). The Master node controls other child nodes to fulfil system activities. The role of these child nodes is to store data within the Grid environment. The Master node therefore includes most of the essential services while the child nodes only contain some data input services and local data update detection services. The Master node also keeps the Grid metadata for Grids management and the Knowledge Base rules for controlling event reaction behaviour.

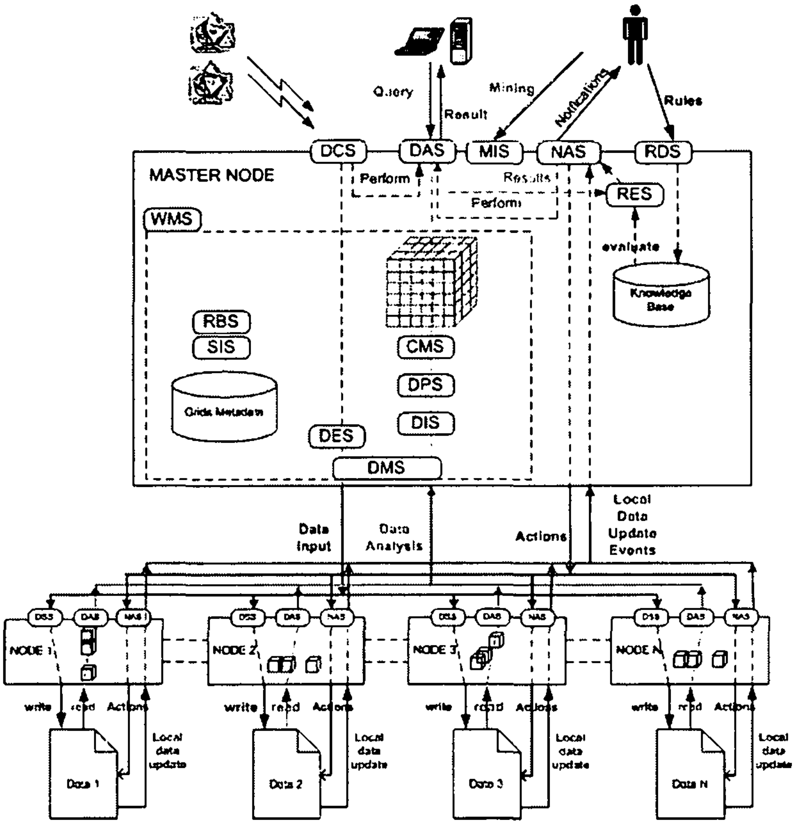


Figure 5.7. The operations of GZLDSWH services

The operation plan of the services in GZLDSWH is specified in the logical workflow as described in Figure 5.7 in which the services are arranged in the following logical order. The Data Capturing Service (DCS) receives continuous data streams from stream sources such as sensor systems, satellites, etc. Due to the huge amount of incoming data, the DCS must capture the data timely and invoke available Data Storing Services (DSS) residing at several child nodes for storing data. The DCS could also invoke Data Cleaning Service (DES) to clean the data before storing if necessary.

After storing data at child nodes, the Analysis Service (DAS) at the Master node will be invoked immediately or after predefined timely schedule depending on application requirements and performance trade off. DAS execution will create the virtual Data Warehouse from scratch. For this purpose initially, the DASs available at several local child nodes are invoked. Due to this, the *Cube Management Service* (CMS) gains the essential raw data at the child nodes to build the global OLAP cube. Each child node contains part of the cube namely "cube chunk". Data will then be integrated into the common format by the *Data Integration Service* (DIS). Before being stored into the virtual Data Warehouse, data can be passed to pre-processing phase via the *Data Preprocessing Service* (DPS). The DPS can perform several tasks such as data cleaning, data transformation, data normalization or data reduction. After the global cube is formed, the DAS will perform analysis queries or data mining algorithms (via the equivalent *Mining Service* - MIS) based on the data inside the virtual DWH. The analysis results then will be sent to the *Rule Evaluation Service* (RES).

The RES accesses the Knowledge Base and evaluates the rules. The Knowledge Base rules are provided by the user through the *Rule Design Service* (RDS). Dependent on the rule criteria, the RES could invoke the DAS or MIS for further analysis before issuing the final decision or invoke the *Notification-Action Service* (NAS) to issue relevant notifications, recommendations, alerts, etc. to the users. The NAS could also send back other action commands to several grid child nodes for executing several data manipulation operations at the local data sources such as insert, delete, update, etc. Furthermore, the analysis process above could be executed to answer the analysis queries issued by other applications. Especially, in case the local data update takes place at the grid child nodes, the analysis process could also be invoked. Whenever the local data update happens, the NAS at local child node sends the "local data updates" message events to the NAS of the Master node. The NAS then invokes the DAS and the Analysis process will be executed.

The invocation between the services described above, in fact, is more complicated in the dynamic Grid environment because of the computational and networking capabilities, and availability of the Grid nodes. Therefore, the Grid services invocation process is strictly monitored by the Resource Broker Service (RRS) and the System Information Service (SIS). These services manage the available resource and find the best-fitting resources for resource allocation and dispatch. The role of Workflow Management Service (WMS) is to execute the complex, highly dynamic workflows involving different grid service instances.

5.6. Dynamic Grid Service Collaboration

As previously mentioned, each OGSI based service in GZLDSWH is able to perform an individual task within the whole process. Obviously, these services have to collaborate with each other to fulfill the common purpose of the GZLDSWH system.

5.6.1. Dynamic Service invocation requirement

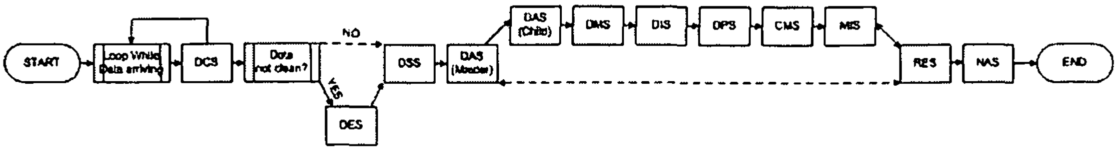


Figure 5.8. The predefine logical Workflow of Service Invocation

The Dynamic Service Control Engine (DSCE) has been developed in GridMiner [Kickinger et al., 2003; Kickinger 2004] to control the service execution via Dynamic Service Control Language (DSCL) document. The exact services handles are specified in the DSCL. Users have to know which service factory to use to perform a certain task. Therefore, the services do not need to communicate with each other. The output of the first service serves as the input of the second service, the output of the second one serves as the input of the third one and so on. No service thus is aware of other existing services and each of the services is able to run completely independently.

The independence of the various services also allows a parallel execution without any communication overhead. This results in an improvement of performance.

However, in our system, due to the requirement of automated event-based reaction, a service must be able to discover, create, bind, and invoke relevant service instances within the Grid environment. Only the execution flows are specified in advance, in which the services are arranged in the specified logical execution order (Figure 5.8). During the execution time, the services have their autonomy to discover, create, bind, and invoke relevant physical service instances within the Grid environment depending on the context at that time. Consider the Telecommunication Fraud Detection scenario, for example, the Call Detail Records (CDRs) are issued continuously as continuous data streams. The Data Capturing Service (DCS) instance is created in the Master Node and its operations should be executed while the CDRs are still arriving. The DCS instance will invoke the Data Storing Services (DSSs) at several child nodes to store data. However, according to the Grid environment at the runtime, some child nodes are available, the others could be corrupted or out of resources. The DCS instance thus must be "intelligent" enough to create and invoke the instances at the suitable Grid nodes. If the CDRs data arrives in a burst fashion, the DCS instance has to create many DSS instances to store all data in a timely fashion, otherwise, it should destroy some non-used DSS instances to free the resources. The similar situation happens when the DAS instance at Master node decides to create and invoke the DAS instances at the suitable child nodes. Especially, in case the Rule Evaluation Service (RES) could not issue the final decision (due to ambiguity between Fraud and non-Fraud call), it has to invoke other analysis services or data mining services for further data analysis instead of invoking the Notification/Action service to alarm the users. Therefore, the dynamic service invocation requirement is extremely important in the GZLDSWH.

To our best knowledge, there are 2 possible approaches for the service flow execution: *centralized control* and *distributed control*. In the former approach, there is a central service control engine which controls all service executions from the start node to the end node of the workflow. The engine itself is responsible in discovering, creating, binding, invoking, and destroying service instances to follow the logical workflow. Thus the engine must keep the information of the whole workflow and should trace the information of the Grid environment such as grid nodes status, resource availability, workloads etc. to coordinate the services execution. In the later approach, there is no such central engine but each service instance has its own "knowledge" to invoke the next service instances throughout the workflow. It is not necessary for each service to keep information of the whole workflow. Instead, each service needs to keep only part of the workflow metadata relevant to itself such as its direct ancestor and descendant services, and the Grids environment context at its execution time. That information should be passed to the service as parameters as it is invoked by its ancestor services and the service will use such information to invoke the next relevant service instances.

Both of the two approaches have advantages and disadvantages. In the centralized approach, the central service control engine, which could also be realized as a service, copes with the coordination between other services. The other services thus only focus on their specific functionality without taking into account the workflow execution. However, it could be the heavy work-load for the engine service if it processes the high complexity workflow or if the number of service instances increases. The distributed control approach, on the other hand, does not have to deal with the bottle-neck issue. However, it is more complicated to develop the services because each service besides its specific functionality must be realized as an agent-based solution to adapt with flexible service instance invocations. Moreover, the service invocation would also become more complex due to the parameters transferred between the service instances. Further investigation on distributed control approach is out of the scope of this paper. It will be one of our considerations in the future work.

In GridMiner [Kickinger et al. 2003; Kickinger 2004], we have developed the Dynamic Service Control Engine (DSCE) which receives the workflow specification document written in DSCL (Dynamic Service Control Language) and executes the workflow by invoking the corresponding Grid service instances specified in the documents. DSCL allows the user to specify variables, workflow structure, operations to be executed, and DSCE will execute the workflow followed by the DSCL document specification. Although the DSCL and DSCE provide some levels of dynamic workflow execution, they still have some limitations. The DSCL does not support branch conditions and loop structures; the DSCE only works with the physical workflow specification document i.e. the document specifies exactly which factory handle URIs should be invoked to create the instances which are usually unknown in advance due to the variant Grid environment. We can improve the DSCL and DSCE in GridMiner to support the new requirements in GZLDSWH. The extended DSCL (Tho et al. 2004) will support the condition branches, loop structures as well as allow the references of the service instance handles could be transferred as parameters. The logical workflows are specified with the unknown service instance handles declared as variables. During the execution time, the *Re-Writer* queries the *Resource Broker Service* and *Information Service* to have the relevant dynamic service factory handle references at that time and rewrites the logical workflow to the physical one. The DSCE engine then will invoke these service instances via the reference variables in the physical workflow. That operation will be repeated at each step of the workflow until the whole process is finished.

5.6.2. Extended Dynamic Service Control Language

DSCL is a XML based language allowing the users to specify the workflow of services activities. It contains exactly of two sections:

- The <**variables**> section: all variables must be defined here. The variables could be either the parameters of service calls, or the results of service calls. XML Schema Simple Type, Complex Type and SOAP Arrays Type are supported as variable type.

```
<variables>
  <variable name="iAge">
     <value type="int" 25 </value>
  </variable>
</variables>
```

- The <**composition**> section contains the description of the workflow to be executed. A workflow is comprised of a set of activities which could be classified as "*control flow*" or "*operational*". The control flow activities control the execution of the workflow and thus must contain other activities while the operational activities are the atomic activities which perform operations.

```
<composition>
    control flow activities
    other control flow activities or operational activities
    operational activities
  </compostion>
```

DSCL allows the users to specify the workflow structure and define the workflow operations

- **Workflow structure**

Our DSCL supports 4 basic execution styles (Sequential execution, Parallel execution, Condition Branch and Loop) by providing several tags namely **<Sequence>**, **<Parallel>**, **<Condition>**, and **<Loop>** respectively. These tags could be nested to realize the complex workflow composition. Figure 6 states an example of a workflow including all control activities and the respective DSCL document

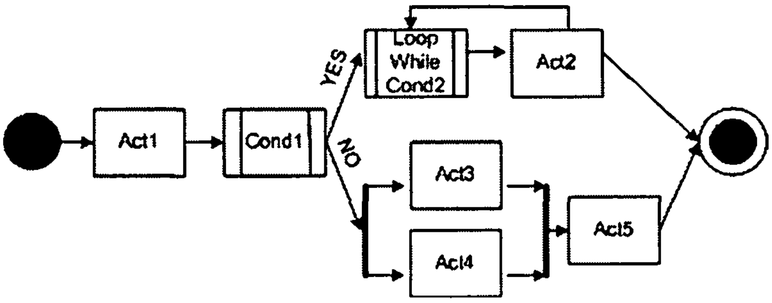


Figure 5.9. Composite Workflow Example

```
<composition>
  <sequence>
    activity1
    <condition>
      cond_var1 = TRUE
        <loop while cond_var2 = TRUE>
          activity2
        </loop>
      cond_var1 = FALSE
      <sequence>
        <parallel>
              activity3
              activity4
        </parallel>
        activity5
      </sequence>
    </condition>
  </sequence>
</compostion>
```

- **Workflow operations**

Besides the control flow activities, DSCL supports other activities namely operational activities. Operational activities perform operations for interacting with the underlying Grid services such as creating new service instances, destroying instances, invoking operation of services, and querying service data element. DSCL provides respective tags to specify these operational activities: <**createService**>, <**destroyService**>, <**invoke**>, and <**querySDE**>. The operational activity could not contain other activities and must have the mandatory attribute namely activityID which is of type DTD. This attribute is necessary for the workflow engine to identify the activity.

The major difference between Grid and common Web services is the fact that the Grid service could be either persistent or transient. The persistent service is created and available if its container is running. In contrast, the transient one is created and invoked as and when required and destroyed afterwards. The transient service is always created by its Factory service. The following information is necessary to create a new service instance

1. The location of the factory service
2. Additional service parameters
3. A virtual instance name of the newly created instance

```
...
<createService activityID ="Act1"
factory-gsh="http://url/serviceFactory"
instance _ name="newInstance1"/>
...
```

In GZLDSWH, there is a situation when we have more than equivalent factory services at different grid nodes at the same time. For example, the Data Storing services located at several child nodes when we need to store data stream. In such situation and in other cases when the Grid environment changes rapidly, the Resource Broker Service decides which Service Factory should be executed to create a new instance according to the availability and resource capacity of the different Grid nodes. DSCL provides the dynamic service creation by allowing Grid service handle references transferred via variables. It is also possible to create the service instance with user defined parameters via <**parameter**> tag.

```
...
<variables>
  <variable name="factgsh">
    <! Default value of the factory service handle>
    <value type="string" http:url/serviceFactory </value>
  </variable>
...
</variables>
```

```
...
<!other services set the value of factgsh, e.g. Resource Broker >
...
<! Create the service instance via reference to factory handle >
<createService activityID ="Act1"
factory-ref="factgsh"
instance_name="newInstance1"/>
```

After a service instance is created, it is possible to destroy the instance, invoke its operations or query its data elements. These activities require the service instance reference to identify the relevant instance. We provide 3 optional attributes namely *instance-name*, *instance-gsh*, and *instance-ref* enabling to reference a service instance (1) via instance name (2) via Grid service handle and (3) via a variable reference to the instance. The usage of each attribute is optional, however exactly one of the three attributes must be used together with the activity.

```
...
<destroyService activityID ="Act1"
instance-gsh="http://url/SerInst01" (or instance-name = "Instant1" or instance-ref =
"varInstGSH")
...
```

Invoking an operation of a Grid service is similar to invoking a method in common programming language. To invoke an operation of a Grid service, the following information is necessary:

1. The required Grid service instance – referenced by one of the attributes: *instance-name*, *instance-gsh*, *instance-ref*
2. Name of the operation – mandatory defined within attribute *operation* and optional attribute *portType*
3. The required parameters – specified in <**parameter**> tags, the parameter is simply a reference to a variable.
4. Result – storing the result of a Grid invocation via <**result**> tag

```
...
<Invoke activityID ="CleanData01"
instance-gsh="http://url/DES01" (or instance-name = "DES01" or instance-ref =
"varInstGSH")
operation = "Clean_Data"
<parameter variable= "Data_Strore">
<result variable= "Data_Result">
</Invoke>
...
```

Some of operations do not return the results; but store them into so called service data elements. To allow querying the contents of these elements, DSCL provides the tag <**querySDE**>. This operation requires the reference to Grid service instance and the name of the required service data element (stored in attribute *sdName*).

```
...
<querySDE activityID="act1"
  instance _ name="instance01"
  sdName="value"
  <result variable="var02"/>
</querySDE>
```

5.6.3. Dynamic Workflow Management Service

In GridMiner project (Kickinger G. & Brezany P., 2004), we have developed an engine service, called the Dynamic Service Control Engine (DSCE), which processes DSCL documents and controls the service execution in both interactive and batch modes. It provides some interesting features such as (a) independent processing (without any interaction of the user) of a workflow described in DSCL, (b) the provision of all intermediate results from the services involved, (c) the possibility for a user to stop, cancel or resume a workflow and (d) the possibility to change workflow at run time (by stopping the engine, changing the DSCL document, and restarting engine again).

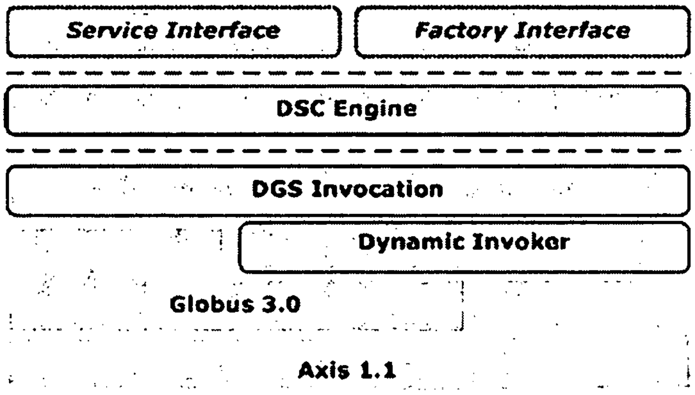


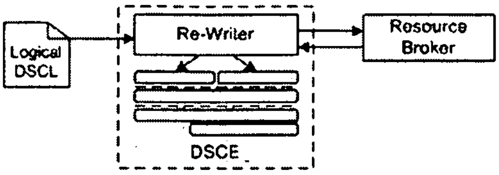
Figure 5.10. Conceptual Architecture of DSCE

Figure 5.10 describes the Conceptual Architecture of DSCE [Kickinger 2004]. The engine is implemented as a stateful, transient OGSI Grid service and has several structured layers. The "top" layer is the Interface layer which provides essential operations to control the engine. The Factory interface allows users to create a new DSCE instance for a specific DSCL document via operation *CreateService (DSCLDocument dscl).* The DSCE engine instance now will be created and its state will change within its life cycle according to user interactions and the activities execution results. The possible states could be *empty*, *initialised*, *running*, *stopping*, *waiting*, *finishing*, or *failure.* The Service interface provides interactive control operations such as *changeWorkflow(), start(), stop(), resume()* as well as several service data elements containing information about the DSCL document, Workflow state and activity state.

The "middle" DSC Engine layer covers the main functionality of DSCE. It controls the whole workflow execution by controlling the execution of activities specified by the DSCL document. First, the DSCL workflow description is parsed, then the "network of activities", an internal model of the workflow, is constructed before processing the activities. Such a network of activities describes the dependency between the activities. Each activity could have succeeding and preceding activities. Succeeding activities are executed right after the execution of actual activity is finished. If an activity has more than one successor, all of them will be executed in parallel after the actual activity is finished. Similarly, the activity could not be started until all of its preceeding ones are finished. This could happen in some situations like loop or parallel execution. Several internal operations are provided in this layer for managing the workflow such as *start(), stop(), resume(), reset(), setDSCLWorkflow()* etc. as well as some operations for controlling the activities such as *startActivity(), EndActivity(), CreateInstanceActivity(), DestroyInstance Activity(), InvokeActivity(), QuerySDE-Actvity(), startNext Activites(), wait-ForPrevious-Activities()* ,etc. The necessary parameters of all underlying services are also prepared at this layer.

Normally, when a Grid service is developed and implemented, additional stub and proxy classes are generated to hide the complexity of communication between the client and the service. This approach is very common and practically used in all distributed object systems like CORBA, Java RMI, and Web Service. To benefit from this approach, the required services or remote objects must be known at the compilation time. However, this requirement is not satisfied in DSCE because DSCE receives a DSCL workflow description document and shall be able to communicate with all services specified within that DSCL. The "lowest" layer namely Dynamic Grid Service Invocation (DGSI), is composed of the DGS Invocation and Dynamic Invoker. It provides classes which allow accessing Grid services and their operations without using common stubs/proxy approach. The Dynamic Invoker, the lowest layer, provides the possibility to invoke any operation on any underlying Grid service. It uses much of ApacheAxis (The Axis Project, 2003), and SOAP engine which are based of GT3 toolkit. Dynamic Invoker translates an operation invocation into a SOAP1.1 message and sends it to the corresponding service to invoke specified operations. It provides all necessary marshalling and un-marshalling of arguments, first by fetching the WSDL of the corresponding Grid service (via its handle GSH), and then setting service port type via *setPortType(String port-TypeName)*, setting operation via *setOperation (String operation Name)* , setting parameters of the operation via *setParameters (Object[] params*). All of information is used to construct essential SOAP operation call. Finally *invoke()* executes the operation by sending that SOAP message to correspond services.

At higher layer, the DGS Invocation provides the classes to use stub-less operation invocation and to access the functionality of the GT3. It provides three classes namely *DGSIService, DGSIFactory, DGSI- Listener* allowing the workflow engine to handle its underlying services such as creation and destruction of Grid service instance, invocation of operations, querying of service data element and synchronization of asynchronous service calls.



(a) Service Control Engine

```
<dscl:variables>
    <variable name="dataArrive"> TRUE </variable>
    <variable name="dataNotClean"> TRUE </variable>
    <variable name="dcsfgsh"/> // unknown value
    <variable name="desfgsh"/> // unknown value
    <variable name="dssfgsh1"/> // unknown value
    <variable name="dssfgsh2"/> // unknown value
    ... // other variables
</dscl:variables>
<composition>
    <sequence>
      <loop while dataArrive = TRUE>
          create,invoke DCS
      </loop>
      <condition> dataNotClean = TRUE
          <sequence>
             create,invoke DES

          </sequence>
      </condition>
       create, invoke several DSSs
   ... // other activities
    </sequence>
<composition>
```

(b) Logical DSCL

```
<dscl:variables>
    <variable name="dataArrive"> TRUE </variable>
    <variable name="dataNotClean"> TRUE </variable>
    <variable name="dcsfgsh"> http://..Master/DCSFactory </variable>
    <variable name="desfgsh"> http://..Master/DESFactory </variable>
    <variable name="dssfgsh1"> http://..child1/DSSFactory </variable>
    <variable name="dssfgsh2"> http://..child2/DSSFactory </variable>
    ... // other variables
</dscl:variables>
<composition>
    <sequence>
      <loop while dataArrive = TRUE>
         <dscl:createService activityID="DCS-Fact01" factory-ref = "dscfgsh"
                 instance-name="DCS-Cap01"/>
          <dscl:invoke activityID="DCS-Inv01" dscl:factory-ref = "dscfgsh"
                  operation="captureData"/>
      </loop>
      <condition> dataNotClean = TRUE
          <sequence>
             <dscl:createService activityID= "DES-Fact01"
                 factory-ref="desfgsh" instance-name="DES-Clean01"/>
             <dscl:invoke activityID="DCS-Inv01" dscl: factory-ref="desfgsh"
                 operation="cleanData"/>
          </sequence>
      </condition>
      <dscl:createService activityID="DSS-Fact01" factory-ref = "dssfgsh1"
              instance-name="DSS-Store01"/>
      <dscl:invoke activityID="DSS-Inv01" dscl: factory-ref = "dssfgsh1"
              operation="storeData"/>
     <dscl:createService activityID="DSS-Fact02" factory-ref = "dssfgsh2"
              instance-name="DSS-Store02"/>
      <dscl:invoke activityID="DSS-Inv02" dscl: factory-ref = "dssfgsh2"
              operation="storeData"/>
   ... // other activities
    </sequence>
<composition>
```

(c) Physical DSCL

Figure 5.11. The Extended Service Control Engine and corresponding DSCL documents

DSCE suits well in GridMiner where the interaction role of user is important. The engine operates based on the "*physical*" DSCL document specified by the user, i.e. it only works with the DSCL that specifies exactly the service handles. It does not accept the "*logical*" workflow which only specifies the logical name of the required service. In GZLDSWH, we sometimes do not know in advance which service factory should be executed to create a new instance. Instead, the decision should depend on the runtime environment. Besides, because of the *"automated event-based reaction"* feature of GZLDSWH, a higher level of automation in service invocation engine is required. Therefore, the Dynamic Workflow Management Service in GZLDSWH extends the DSCE with the automatic Workflow Re-writer ability. Now, the WMS Service will accept the logical DSCL, parse it and find out logical services i.e. services which do not have the exact physical factory handle. It then queries the Resource Broker Service to have the relevant physical service factory handle and then re-write the DSCL with the new factory handle value. It finally passes the re-write DSCL to the DSCE engine to invoke the services.

Figure 5.11 (a) describes the extended Dynamic Service Control Engine (EDSCE) based on that DSCE engine. This EDSCE engine accepts the logical workflow in Figure 5.11 (b), parses it and converts to the physical executable workflow as depicted in Figure 5.11 (c). The significant difference is the values of the parameters that have been supplied after the Re-Writer queries the Resource Broker Services.

5.7. The OLAP Cube Management Service

As mentioned previously in the overview section, our approach in GZLDWSH is to store all streaming data into Grid nodes, and build OLAP cubes from these Grid-based sources prior to executing analytical queries that evaluate the rules. Following this approach, we have to implement the OLAP Cube Management Service that manages the creation, updation and querying of the associated cube portions distributed over the Grid nodes. The kernel part of this service is the *OLAP engine*. The first prototype of this engine [Fiser et al. 2004] has been already implemented in Java.

The OLAP data cube structure consists of an increasing number of chunks, which again consists of a fixed maximum number of measures. A measure is the smallest unit of the cube, one atomic element, and it actually contains just a numeric value. The chunk is a part of the whole cube; it has the same dimensionality like the cube but collects aggregation data at one grid node. The chunk contains the measures associated with a number of positions of each dimension. Because the amount of memory used by the whole cube usually will be much higher than a system may provide, each chunk offers methods for storing and loading its data onto and from the disk storage. Thus, always only a limited number of chunks is kept within memory at the same time. Storing and loading targeted chunks is called chunk swapping and is a subsystem of the data cube structure implementation. This is similar to paging in modern operating systems with the distinction that our chunks may grow up to a specific size, hence, the memory resident chunk location table, which is a list of chunks currently resident in memory, varies in size. This is because the aggregation results are also stored within the same data cube.

Special indexing structures and paging mechanisms are necessary to manage the Grid-based OLAP cubes. The index database contains the literal positions, the meta-information, of each dimension and maps unique integer values to position indexes within each dimension. Furthermore, it provides methods for the linearization of multi-dimensional position indexes used for addressing specific measures of the OLAP cube.

Several methods are available, e.g. hashing, bit encoded sparse structure (BESS), binary trees or others. In order to deal with a huge number of tuples, we need an algorithm, which on the one hand is fast, and on the other hand, is not limited to some upper boundary. This is necessary to avoid multiple scans of the source data and allows insertion of measure aggregation after cube construction. The method, called Dynamic bit encoding (DBE) [Fiser et al. 2004] was developed for indexing the bit maps of OLAP chunks. DBE is based on BESS with the difference that the bits used for each dimension are kept within bit masks which are extensible and mutually exclusive to each other from a binary point of view. The position indices are processed by the OR operation using these masks, which results in a linear measure address, called the global index.

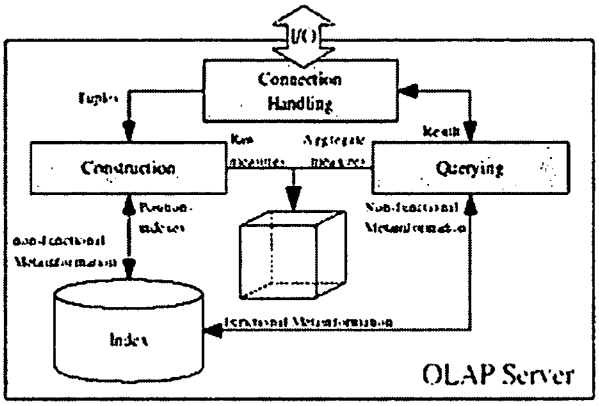


Figure 5.12. OLAP engine architecture

Figure 5.12 describes the system architecture of the OLAP engine prototype. The cube Construction reads the tuples one after the other, passes over the items to the index database, retrieves their global index and then passes the (raw) measure and its associated global index to the data cube structure. The Querying functional block is some kind of highly sophisticated, recursively nested loops for aggregation of measures. Because the number of computational operations of nested aggregations depends on the size of the dimensions and thereby on the order in which dimensions are aggregated, the engine uses a kind of query plan optimization to select dimensions in a "good" way. The procedure of dimension selection is done by traversing a tree which in the literature usually is called the query lattice. The task of aggregation is realized sequentially by loading one chunk after the other and aggregating them one by one. To avoid repeated computation of same aggregates, they are also stored in the cube structure as if they were raw measures and also get index entries within the index database within appropriate dimensions.

Connection handling is the network interface, which allows user interactions with the system. A typical workflow of the system usage is as follows. After start-up, the index tables and the base cube are constructed. This is done upon loading and parsing a structured or semi-structured text file representing database tuples. It is called the origin input stream (see Figure 5.12). To each position, a unique index value is assigned and this assignment is kept within the index database. Then all index values from the tuple are merged together using the DBE algorithm. The encoded global index is used to uniquely locate and store the measure within the cube. After the step of tuples import, the server opens a listening TCP socket and accepts client connections. A simple command language was defined [Fiser et al. 2004] for communication between server and client. This is called the control input (output) stream. A client now is able to submit queries. The server supports concurrent sessions, which allows multiple users to login concurrently.

5.8. Knowledge Based Rule Design and Evaluation Mechanism

The Knowledge Based Rule is the “brain” of the GZLDSWH and controls how the system reacts automatically to the continuously arriving events based on the complex incremental multi-dimensional analysis of the collected OLAP data. The rules follow the basic Event-Condition-Action (ECA) rule structure, but carry out the complex OLAP analysis instead of evaluating the simple conditions as in ECA rules in OLTP. It is necessary to design a model for maintenance of the Knowledge Base rules that allows users to insert, update, replace or delete the rules easily. If possible, the rules should be managed in a consistent manner with the option of checking the validation of the rule, avoiding rule conflicts, and maintaining the consistency among the rules.

The mechanism to evaluate the rule is also a challenge when the events come from different sources within the Grid environment. There are several causes that can trigger a rule. It could be the temporal time events generated from the scheduling service, explicit invocation from other service or auto-triggered when the Data Service Element (DSE) in another service exists. Right now, we just use a simple XML-based file to manage the rules. The Rule Evaluation Service is explicitly invoked by the Workflow Management Service.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReactiveRule name = “Mobile Fraud Analysis">
<variables>
   <variable name=”average_call_length”> </variable>
   <variable name=”call_length”> </variable>
```

```
    <variable name=”country”> </variable>
   <variable name=”area”> </variable>
    <variable name=”time_point”> </variable>
<variables>
<Events>
    <Event name = “International long call">
   <fact > country = “Vietnam”  area = “Asia”
   call_length in “current_time” > 30 minutes </fact >
  </Event>
</Events>
<Conditions>
      <fact >
     country = “Thailand” or
        country = “Indonesia” or country =”Malaysia”
     average_call_lenghth in “last 3 hours” > 2 * average_call_lenghth in “last 3
months”, area = “Asia” </fact >
</Conditions>
<Actions>
    <Action name = "Fraud Situation Alarm"/>
</Actions>
</ ReactiveRule>
```

Figure 5.13. Mobile Phone Fraud Detection Rule

Figure 5.13 describes a Fraud Detection Rule which monitors the international mobile call. This rule will be triggered when the current international call length to Vietnam of a certain customer is over 30 minutes. It will check whether the average call length of this customer to other countries in South East Asia such as Thailand, Malaysia, Indonesia within the last 3 hours exceeds twice the average call length to Asia for this customer. If so, the Fraudulent Call alarm will be issued and sent to the customer.

5.9. Reliability and Efficiency

In our architecture there are two options of how incoming data streams are assigned to nodes in a Grid. First, a master node can select which node an incoming data stream will be stored on. Second, any node in the Grid may accept an incoming data stream. The node can choose to handover the data stream to the master node at any point in time.

Grids are characterized by a large number of cooperating nodes. The rationale is that storing of multiple data streams can be handled in parallel and is thus more efficient. Amdahl's law [Gene 1967] describes the speedup of programs if the performance of some parts (but not all) can be improved. A special case of this improvement is parallel processing. If parts of the code can be executed in parallel, the overall performance will be better.

1 / (S + (1-S) / N) shows that speedup depends on how much must be executed sequentially (S in the range of 0.0 to 1.0).

As the size of the grid (N nodes) increases, reliability of nodes becomes an issue. Even though the mean time to failure (MTTF) of individual nodes decreased over the last years, the size of grids grows faster so that the MTTF of the system decreases if one requires all nodes to be online. Clearly, in a large grid with N nodes, the setup usually requires only A nodes (A<N) to be available for the grid to work.

If a node receives an incoming data stream and then fails, parts of the data stream might be lost. To avoid inconsistent states, the failed node and possibly some neighboring nodes will need to roll back to the last checkpoint.

Following Elnozahy's models [James et al. 2004] we look at specific requirements of data streams and how system parameters need to be modified to achieve the primary goal of increasing performance. Each failure causes additional work to be performed; useful work is work that a system performs that will not be lost by a failure. Obviously, the goal is to maximize the ration U of useful work / total work. When serial parts are low then increasing the number of active nodes A to values near N increases U [James et al. 2004]. Building on Elnozahy's results, simulations show that checkpoint intervals for grids with fewer than 4000 nodes, should be greater than 20 minutes. For growing numbers of N the optimal (maximizing U) interval to set checkpoints is approximately the checkpoint latency, assuming a checkpoint latency of five minutes.

To minimize overhead of creating checkpoints it is advisable to reduce the length of serial parts in the process of storing data streams. This can be achieved by distributing the stream to several nodes: for instance, 32 KBytes to the first node, the subsequent 32 KBytes to the next node, etc. Assuming a data rate of 100 KBit/s the sequential duration of a 32 KByte block is 32*1024*8 / (1024 * 100) = 2.56 sec.

By adjusting the block length of the stream the overhead of setting checkpoints can be adjusted. Shorter blocks will reduce the overhead but will increase fragmentation of the stream. Depending on the requirements of retrieving and analyzing the data stream a decision concerning the tradeoff checkpoint overhead versus data fragmentation has to be made.

5.10. Implementation

In this section, we will describe our ongoing prototype implementation and some experimental performance results. So far, we have developed the prototypes for the Dynamic Service Control Engine (DSCE) and the Sequential OLAP Cube Engine.

5.10.1. Dynamic Service Control Engine Prototype

Figure 5.14 describes the class architecture of the DSCE prototype which follows the conceptual architecture mentioned earlier. It is implemented as a stateful and transient OGSA Grid service.

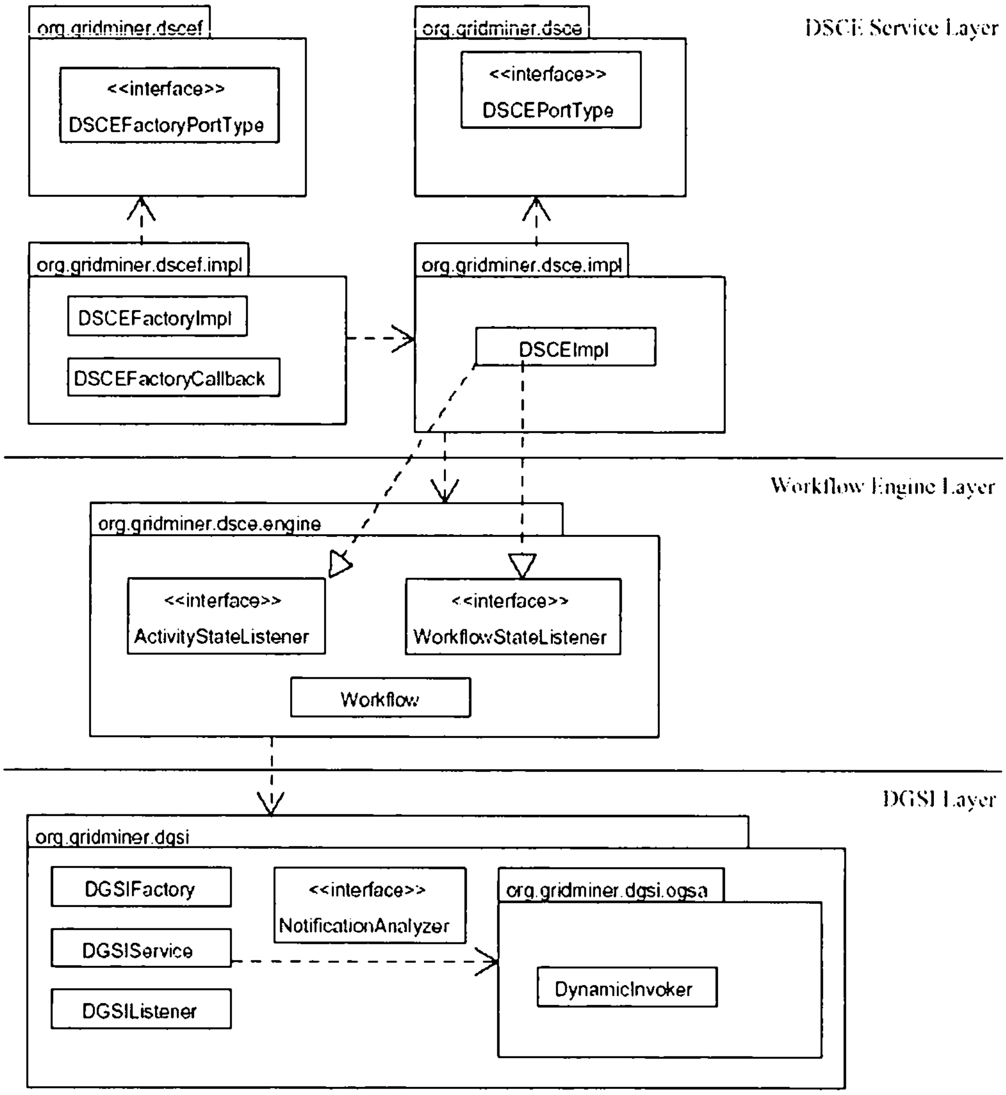


Figure 5.14. DCSE Class Architecture

DSCE is a workflow enactment engine, which executes a workflow of OGSA Grid Services described by the Dynamic Service Control Language (DSCL). The engine is always in a particular state. Dependent on its state, it could accept the client's command, executes the command and changes the state respectively. This state tells the client about what the engine is doing at the moment and if it is ready to accept certain commands. The state diagram of the DSCE engine is depicted in Figure 5.15.

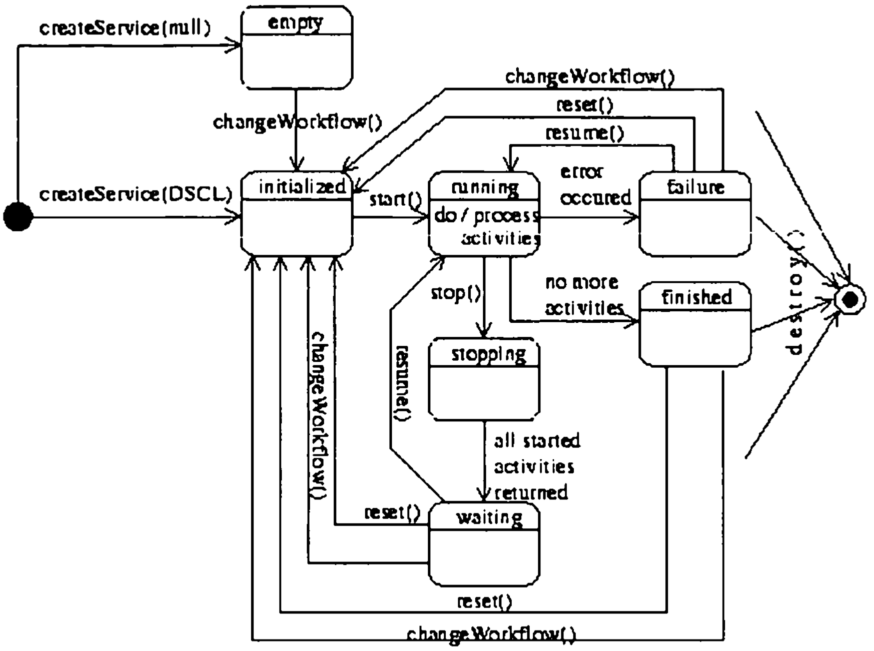


Figure 5.15. DCSE State Diagram

After being created as a service instance, the DSCE engine is in *empty* state. If a client sends a new DSCL workflow description and the engine accepts it, its state will change to *initialized*. The state will change to *running* if the client starts the workflow and the change of DSCL workflow description is not possible thru changeWorkflow() command. If the client need to change the DSCL description, the engine execution has to be stopped first and its state will change to *stopping* (during the processing stopping the activities) and *waiting* (when all stated activities have been returned) respectively. The engine could finish without error (*finished* state) or stop the execution with error report (*failure* state).

To execute the performance tests, we define a minimal sub-workflow with no computational cost. This is used as the smallest item throughout all test runs. The sub-workflow consists of the following sequence:

1. Create new *TestGridService* instance
2. Invoke the operation *getFloatValue()*
3. Invoke the operation *setFloatValue(float)*
4. Query the service data element
5. Destroy previously created *TestGridService* instance

To test the workflow engine, 24 different DSCL documents are used (pt1.xml, pt2.xml,..., pt24.xml). DSCL document pt1.xml contains exactly one of the previously defined sub-workflow, DSCL document pt2.xml contains two successive sub-workflows and so on. Finally, DSCL document pt24.xml contains 24 successive sub-workflows, this result in 24 x 5 = 120 activities.

A client application (*DSCE-Client*) is implemented to invoke the DSCE service instance, one for each DSCL documents. To determine the overhead of the DSCE engine compared to a direct execution of the workflow, another client has been implemented which executes the sub-workflows without usage of DSCE and DSCL (*DirectClient*).

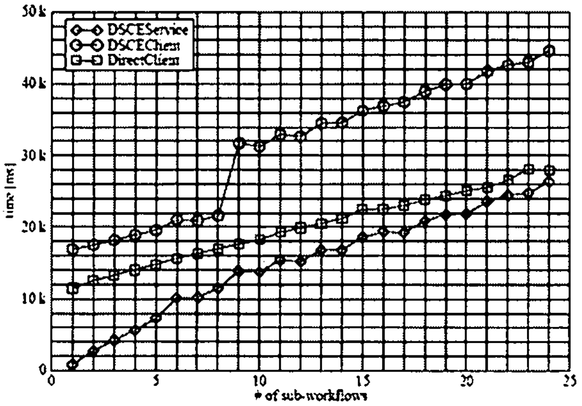


Figure 5.16. Comparison of overall workflow execution time.

In Figure 5.16, we can see the execution time of the overall workflow between the two methods: invoke DSCE engine or direct execution of the workflow.

5.10.2. Sequential OLAP Cube Engine

Our prototype implementation of the OLAP engine was tested with three input files - tab separated text files - of different sizes. The test files were structured in a way so that each column represents the values for a dimension, whereby the first column contained the measure values for the cube. Using these input files, we have built three different OLAP cubes with four dimensions, which mean that each input file contained data in five columns. The following table gives an overview of the characteristics of our test files.

	Input I	**Input II**	**Input III**
NOL	2000	8000	20000
NODV in column I	100	300	600
NODV in column II	10	15	25
NODV in column III	45	86	125
NODV in column IV	2000	8000	20000
BPCS	90000000	3096000000	37500000000

Table 5.2. Characteristics of the test files used in the test scenarios.

NOL.......................Number of lines
NODV....................Number of distinct values
BPCS......................Biggest possible cube size

(BPCS = (NODV – I) x (NODV – II) x (NODV – III) x (NODV – IV)

We have treated each test session – which uses a different input file - as a new OLAP scenario and sent the same queries to the OLAP cube in order to provide an overview of the querying performance relative to the increasing cube size. Each query is represented in a form like [ANY, ANY, ANY, 0] whereby the term ANY indicates that the cube shall be aggregated along this dimension. This means that the costs for a query increases with the increasing number of distinct values in a dimension, when the query value is set to "ANY". Figure 5.17 demonstrates the first performance results, which are more than satisfying for a prototypical sequential OLAP engine. In our future research effort, we plan to investigate the architecture of a parallel and distributed OLAP engine.

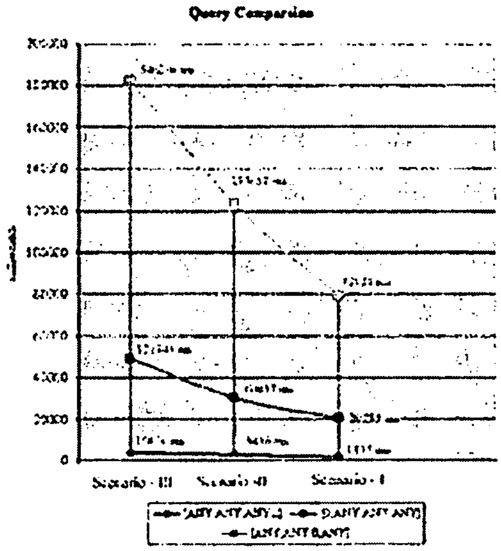


Figure 5.17. Performance results of queries for three different scenarios.

5.11. Summary and Future Work

In this chapter we have presented a framework of building the Grid-based Zero-Latency Data Warehouse (GZLDSWH) system for continuous data streams processing and analysis. The GZLDSWH system is built upon the set of OGSI/GT3-based services. Following the pre-defined reactive Rule-based Metadata specified by the user, the system can react automatically to continuous data streams in near real time. An adaptive service interaction mechanism is used for the flexible service collaboration. The system then can execute some relevant actions at the data sources or send significant awareness such as alarms, notifications, recommendations, etc. to the user.

Our proposed GZLDSWH system is currently an ongoing research and some of the other issues require further investigation. Major focus of future research should be on solving several significant services within the GZLDSWH such as Grid Resource Allocation and Scheduling, Heterogeneous Data Mediation and Integration, Grid Data Replica Synchronization [Taniar et al. 2005], Data Cube Construction and Management, Rule-based Metadata Construction, Embedding and Evaluation. Other open issues such as Performance Efficiency in building Data Warehouse on the fly, Distributed Service Control Management, Stream-based Distributed Processing, Heterogeneous Stream, Availability and Security should also be considered. Besides that both Grid and Data Stream processing technologies are young and still evolving. The Semantic Grid [Roure et al. 2003] and Grids services have the role similar to Semantic Web and Web services. Recently, WS-Resource Framework & WS-Notification [Globus Alliance 2004] proposals have been announced as an evolution of OGSI with the purpose of effective integration of Grids and Web services standards. The work presented here is closely related to OGSA/OGSI so it has to adapt with the WS-Resource Framework with suitable modifications. The distributed control of service discovery, creation, invocation and destroying will be considered as an alternative of collaboration model. The orchestration of Grids or Web services, another approach for solving the workflow problem, should also be further investigated.

Chapter 6. Sense & Response Service Architecture (SARESA): An Approach towards a Real-time Business Intelligence

The ability to collect and analyze information in real time is
a cornerstone of the "intelligent" organization
Colin White, DataBase Associates International, 2001

The dynamic business environment of many organizations require to monitor their business, IT and organizational processes in near real-time in order to proactively respond to distincntive features or exceptions and to take advantage of time-sensitive business opportunities. The ability to sense and interpret events about a changing business environment requires an event-driven IT infrastructure for making fast and well-informed decisions and putting them into action. However, traditional Business Intelligence (BI) and Data Warehousing technologies do not directly address time sensitive monitoring and analytical requirements. We introduce an enhanced BI architecture that covers the complete process to sense, interpret, predict, automate and respond to business environments and thereby aims to decrease the reaction time needed to make business decisions. We propose an event-driven IT infrastructure to develop BI application which enables real-time analytics across corporate business processes, notifies the business of actionable recommendations or automatically triggers business operations, effectively closing the gap between Business Intelligence systems and business processes. The Mobile Phone Fraud Detection scenario was chosen for building a prototype that illustrates the proposed approach by using current available IT technologies.

6.1. Introduction

Advances of modern technologies in various domains has accelerated the intensity of competition, increased the volume of data/information available, and shortened decision-making cycles considerably. More and more information is managed and stored electronically, and the data is getting increasingly complex in both structure and semantics. Consequently, strategic decision makers are being exposed to the huge inflows of data and information from their various resources and they are under rigid time constraints to make the right decisions.

The requirements of monitoring and reacting to enterprise- or organization-level events that cannot be detected by a single operational system become more essential in the competitive business market. Monitoring vast amounts of data is more than just providing real-time alerts. It about saying: "There is something going on here and we are continuously receiving more data. Here is all the information available at this moment that will help you to find the needle in the haystack and to make a well informed timely decision". Companies need to track business processes, such as order processing, quality assurance, inventory, logistics, compliance, etc. in near real-time, to improve operational efficiency as business events are happening. They are also looking for answers to questions such as: How can we improve revenues? What characteristics do our most profitable customers share, and how can we serve them better? Executives today are more challenged than ever to make quick, well informed decisions that address growing business issues and regulatory standards. To answer these questions, companies need a window into the current health of the organization and the tools to act.

However, with traditional Data Warehouse and Business Intelligence techniques, there is still a gap between the time the operational data is created and the time the analysis information is becoming available. Based on the traditional assumption that business decisions did not require most actual information but very rich amount of historical data for tactical decision making, the Data Warehouse refresh process is typically performed in large batches (e.g. once a week or over-night) thus potentially causing a considerable delay when the information from operational systems is reflected in the Data Warehouse. Out-dated warehouse information is therefore not satisfactory for applications which require analysis at the speed of the business. A bank or telephone company requires real-time analytical functionalities to detect suspicious activity in customer accounts in a timely manner before it results in financial damages. As a result, deployment of business intelligence solutions for both operational and tactical decision-making is becoming an increasingly significant use of information assets.

In this chapter, we propose an Event-driven Business Intelligence solution that integrates a real-time closed loop decision-making (Sense & Respond loop). This loop is a dynamic discovery process which continuously 1) observes and collects events from a business environment, 2) converts the event data into meaningful business information, 3) discovers and analyzes business situations and exceptions, 4) automatically selects the most appropriate actions for a response to the business environment, and finally 5) executes the business actions based on the decision that has been made. Based on this approach, we develop a Sense & Response Service Architecture (SARESA) as an infrastructure to build the real-time business applications. The prototype implementation using current OLAP and Data Warehousing technologies is described to illustrate real-time sensing of events, real-time data analysis and instant proactive response in a Fraud Detection application.

The remainder of the chapter is organized as follow: In Section 2, we introduce the requirements of Real-time Business Intelligence and discuss about a traditional Real-time Business Intelligence architecture. This architecture includes also the Real-time Data Warehouse component. Section 3 introduces a novel architecture for an enhanced Real-time Business Intelligence called SARESA (Sense And REspond Service Architecture) which uses a Sense & Response approach for overcoming the real-time business analysis requirements. The prototype implementation of SARESA architecture applied in Fraud Detection application is described in Section 4. Finally, the paper ends with a conclusion and an outlook for future research (Section 5).

6.2. Real-time Business Intelligence

6.2.1. Real-time BI Requirement

As mentioned earlier, many operational decisions (e.g. promotion effectiveness, customer retention, key account information [Inmon 1999] need actual yet integrated and subject-oriented data in or near real-time [Schulte 2000]. However, these real-time operational/tactical decisions are not supported by traditional Business Intelligence Systems which aim to support strategic decision makers and thus uses analytical applications that are periodically fed with data from the Data Warehouse. These analytical applications are generally completely disconnected from operational IT systems. The decisions are executed by communicating them as a command or suggestion to humans, thus always have latency. The real-time analysis requirements demand a set of service levels that go beyond what is covered by typical of a traditional Business Intelligence System:

- **Data freshness**, the need for data freshness escalates significantly, because particular data have to be updated more frequently in order to improve decision-making, support for various data freshness requirements (high/low priority data).

- **Continuous data integration** [Bruckner et al. 2002], which enables (near) real-time capturing and loading from different operational sources and event-based triggering of actions even during data integration. This sort of data integration results in an increasing number of late-arriving data (e.g. due to propagation delays), because timeliness becomes more important.

- Highly available analytical environments based on an **analysis engine** that can consistently generate and provide access to current business analyses at any time not restricted by loading windows typical of the common batch approach.

- **Active decision engines** that can make recommendations on discovered situations or exceptions, or even (rule-driven) tactical decisions for routine decision tasks encountered in an analytical environment.

- Changes of a business process or settings in the operational environment must not disrupt the interoperability with the event stream processing. An **adaptive platform** for the **event stream processing** is required to deal with the changes of the operational environment.

- The number of users and performance requirements for a real-time Data Warehouse will increase by orders of magnitude with deployment of analytic applications enabling tactical decision support. Therefore, **high availability** and **scalability** are indispensable criteria.

While a complete real-time enterprise Business Intelligence System might still be a future challenge, some yet feasible approaches may well enable Data Warehouses to react "just-in-time" to changing customer needs, supply chain demands, and financial concerns. In our vision, the Real-time Business Intelligence steers towards the goal of timeliness to its logical extreme of **instantaneous awareness** and **appropriate response** to events captured in the business environment.

6.2.2. Real-time BI Architecture

To satisfy the requirements mentioned above, it is necessary to enable a complete business intelligence process to observe, understand, predict, react to, reorganize, automate and control the feedback loops in real-time. Therefore, the Real-time Business Intelligence in our vision must include *analytical services* which are continuously fed with data from the operational environment and can be directly invoked by other systems. The objective of analytical services is to handle real-time data and to cope with continuously updated data. The fresh data for the continuous analysis requests is provided by the Real-time Data Cache.

In the popular classical three-tiered Data Warehouse architecture [Chaudhuri 1997], data from data sources is extracted, transformed and loaded into the Data Warehouse via ETL components (tier 1). The warehouse storage (tier 2) manages huge of details and pre-aggregated data which is available for complex multi-dimensional analytical query from OLAP server and other reporting tools (tier 3). In this architecture, ETL technology is designed for batch updates while the warehouse system is offline, and hence is not suitable for real-time processing. Therefore, besides the traditional ETL components, specialized real-time ETL components are necessary in tier 1.

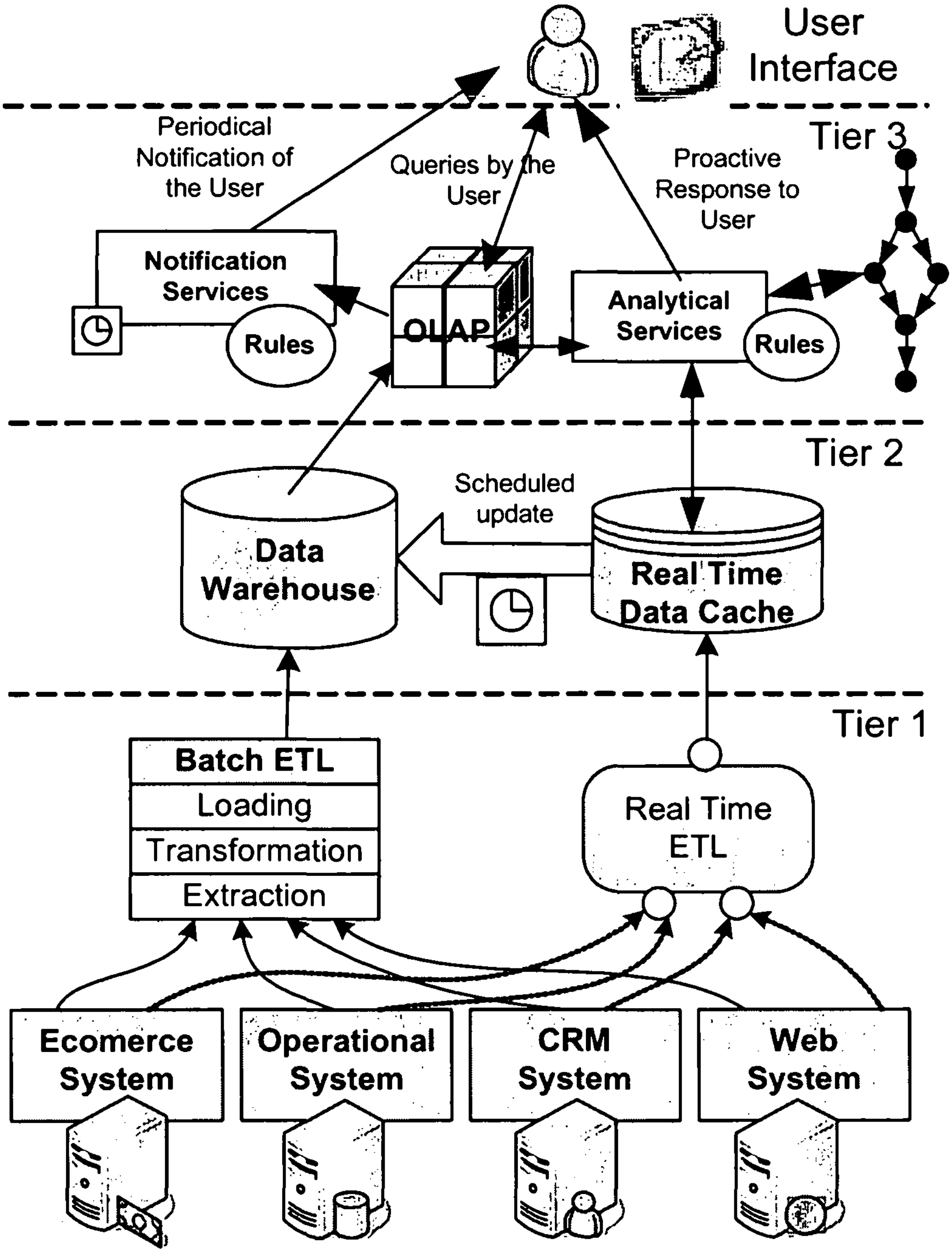


Figure 6.1. Traditional Business Intelligence Architecture

The continuous updates to a Data Warehouse would reveal new problems such as interference with complex analytical queries, materialized aggregates, sophisticated index structures, views, and the costly maintenance of cubes. In tier 2, a traditional Data

Warehouse storage system needs to be extended by a Real-time Data Cache which serves as a staging area by managing real-time updates and periodically batches updates to the Data Warehouse.

In tier 3, the analytical services retrieve their data from the Real-time Data Cache as well as the from the OLAP Cube (which built upon the Data Warehouse) as soon as analytical requests are required by a business process. The analytical services continuously analyze the data patterns and discovery of situations and exceptions. The rule engine assists the services to recognize certain situations and exceptions as well as generates an appropriate response. Therefore, by continuously observing and analyzing data, the analytical services can propose proactive responses to optimize a business process and adapt the business environment. The notification service also analyzes data and sends relevant notifications to the user in the periodical manner.

One major problem of existing BI architectures as shown in Figure 6.1 is the lack of integration between OLAP systems and the operational business environment. As Figure 6.1 shows, in traditional BI architectures data for OLAP purposes are periodically refreshed. In the following section, we propose an architecture that can use OLAP as analytical service with current business data and that allows business processes to take full advantage of OLAP capabilities during the process execution.

6.2.3. Enhanced Real-time BI Architecture

We propose an enhanced BI infrastructure that could reduce the action time and thereby increase the value of Business Intelligence (discussed in Section 2.3). Therefore, traditional BI has to be enhanced towards a closed loop real-time BI, shortening the period of time between the occurrence of a business event that requires an appropriate action by the organization and the time the action is finally carried out. Figure 6.2 shows an architectural diagram for real-time analytics with two infrastructure types: 1) information integration infrastructure and 2) business integration infrastructure. The main objective of this architecture is to seamlessly integrate the two infrastructure types in order to minimize the aforementioned latencies. In the diagram we extended the components and modules of traditional BI in order to enable real-time analytics for business environments.

The information integration infrastructure is responsible for managing the data for Business Intelligence purposes and offers data analysis to decision makers and to IT systems. Traditional Business Intelligence aims to support strategic decision makers and therefore uses analytical applications that are periodically fed with data from the data warehouse. These analytical applications are generally completely disconnected from operational IT systems. Decisions are executed by communicating them as a command or suggestion to humans. On the other hand, the enhanced Business Intelligence includes analytical services which are continuously fed with data from the operational environment (e.g. via the ODS) and can be directly invoked by other systems. The object of analytical services is to provide continuous data analysis that is able to also cope with current changes in the business environment.

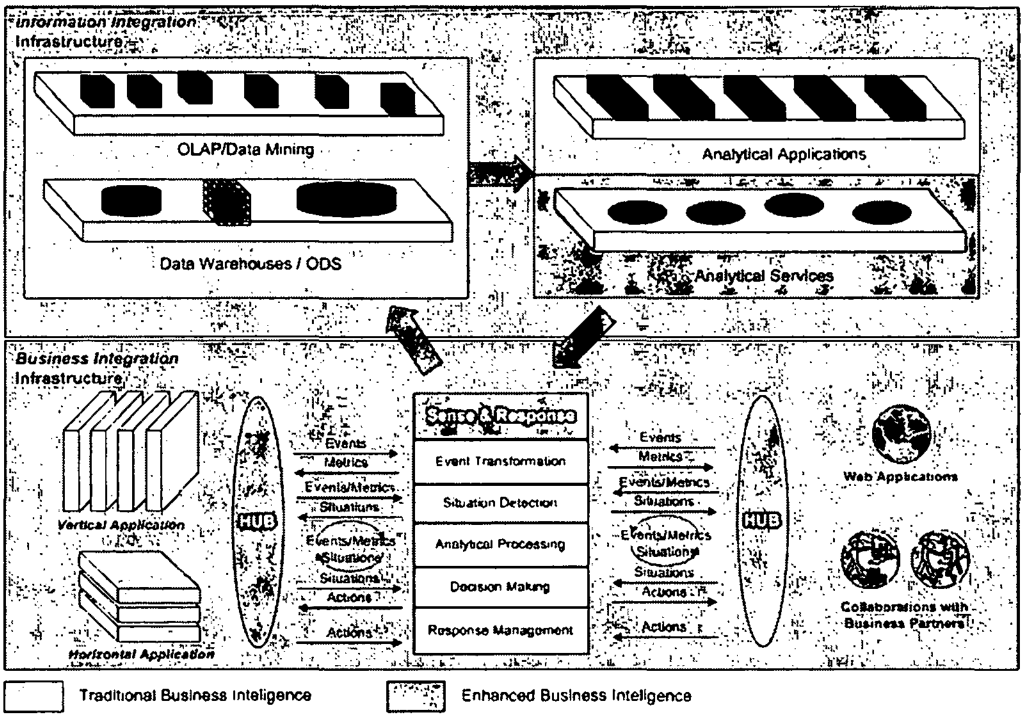


Figure 6.2. Enhanced Architecture with real-time Business Intelligence

The central piece of the Business Integration infrastructure is a Sense & Respond (S&R) system that communicates events via hubs with the internal and external business environment. The internal business environment comprises vertical and horizontal applications which are shown on the left side of the diagram. From the external business environment on the right side, events are captured during the collaboration with business partners, or when the contents of websites changes (e.g. a competitor updates the product prices).

During the event processing in the S&R system, business information is continuously generated and decisions are made to which a response follows. The response has an effect on the source systems (from which the S&R system originally received the events) and consequently also on the performance and the success of the organization. In order to reduce action time, latency has to be reduced in all five stages of the sense and response loop. The next Section will describe in more detail our Sense & Response Architecture.

6.3. Sense & Response Service Architecture (SARESA)

The main objective of SARESA is to help organizations to monitor their business processes and IT systems in order to proactively respond to business situations or exceptions with minimal latency. SARESA is able to continuously receives, processes and augments events from various source systems, and transforms in near real-time these events into performance indicators and intelligent business actions. It automatically discovers and analyses business situations or exceptions and can create reactive and proactive responses such as generating early warnings, preventing damage, loss or extensive cost, exploiting time-critical business opportunities, or adapting business systems with minimal latency.

6.3.1. Sense & Response Loops

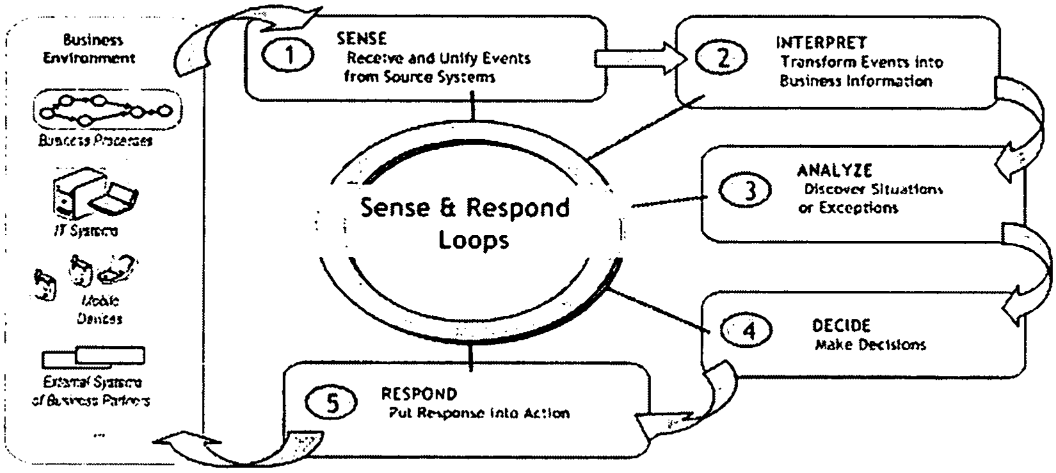

Business Environment
Business Processes
IT Systems
Mobile Devices
External Systems of Business Partners
...
① SENSE Receive and Unify Events from Source Systems
② INTERPRET Transform Events into Business Information
③ ANALYZE Discover Situations or Exceptions
④ DECIDE Make Decisions
⑤ RESPOND Put Response into Action
Sense & Respond Loops

Figure 6.3. Sense and Response Loops

The data processing in SARESA is controlled by "Sense & Response loops" which can be divided into 5 stages as described in Figure 6.3. SARESA system continuously receives, processes and augments events from various source systems, and transforms in near real-time these events into performance indicators and intelligent business actions (see Figure 6.3). Continuous event streams are processed via the Sense & Respond loops that include the transmission, unifying and transforming events into business indicators, a discovery of event patterns, detailed data analysis, and a generation and evaluation of potential responses. Table 6.1 describes the function of each phase and how SARESA supports it.

Stage	Description	Function in SARESA
Sense	Which is the current state of the business environment?	Events are continuously captured and transmitted to SARESA where they are initially unified before the actual data processing begins. Events capture

		changes of the business environment.
Interpret	What do the captured data indicate? What do the data mean for the current situation of the organisation?	Transformation of the captured events (raw data) into business information, such as key performance indicators, business situations and exceptions.
Analyse	Which business opportunities and risks can arise? What are the possibilities to improve the current situation of the organisation?	Analysis of key performance indicators and determination of root causes for business situations and exceptions. Prediction of the performance and assessment of the risks for changing the business environment.
Decide	Which strategy is the best to improve the current situation of the organisation? What are the actions required to successfully put a decision into action?	SARESA proposes the best option for improving the current business situations and determines the most appropriate action for a response to the business environment. This step can be automated with rules or by involving humans (a decision maker selects from alternative choices prepared by SARESA; the system continues the processing with the choice of the decision maker).
Respond	Who has to implement the decision? How can the decision be put into action?	Response to business environment by communicating the decision as a command or suggestion (e.g. by e-mail) or by directly adapting and reconfiguring business processes and IT systems.

Table 6.1. Stages of Sense & Respond Loops

6.3.2. SARESA Architecture

Following the popular Service Oriented Architecture (SOA) approach, we model Sense & Response Service Architecture (SARESA) as a pool of services (system services and Sense & Respond services) and establish the infrastructure that enables a robust communication and interaction between them (Figure 6.4).

The underlying infrastructure offers many system services, which can be universally used by the Sense & Respond services. The system services fulfill basic tasks such as event correlation, event synchronization, logging, thread management, exception handling and centralized configuration management. The Event Service Bus provides the core infrastructure that enables a robust and flexible communication between Sense & Respond services.

Each phase in the Sense & Respond loop introduced in section 3.1 is supported by special Sense & Respond services which can flexibility interact with each other via the Event Service Bus. For instance, the Event Transformation services include Event

Adapters to receive events from a business environment in order to transform them into a standardized format (→ Sense phase).

Additionally, they include components to manage metrics such as the calculation of performance indicators (→ Interpret phase). The remaining Sense & Respond services correspond to the stages of Sense & Respond loops. With SARESA it is possible to include user defined services for various tasks such as discovering situations, a third-party analysis tool as an analytical service or an external rule engine for making automated decisions in Sense & Respond loops.

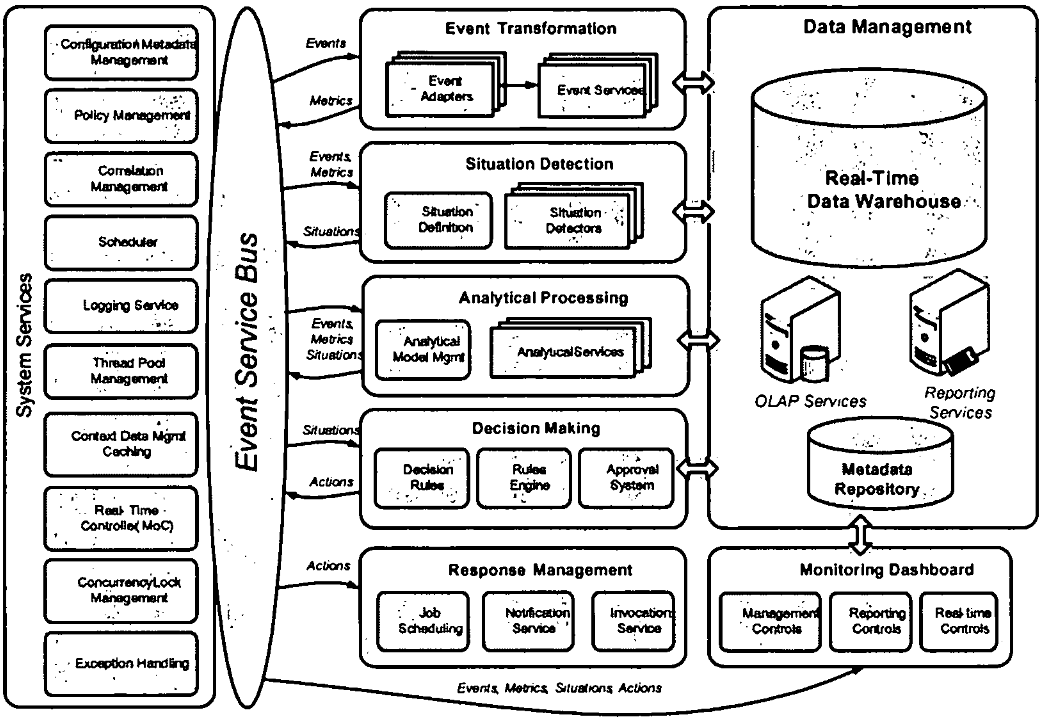


Figure 6.4. SARESA Architecture

For the data management, we distinguish different types of data: historical data, real-time data and metadata. The real-time Data Warehouse provides a single view of historical data and real-time data thereby supporting real-time data processing and real-time analysis in the event services. For multidimensional data analysis, SARESA supports also services for Online Analytical Processing (OLAP) and reporting. All metadata of the SARESA system is stored in a separate metadata repository.

Finally, SARESA includes a monitoring dashboard which provides a user interface for the administrator. It gives an overview of the current status of the event processing during the execution of Sense & Respond loops. With the monitoring dashboard, the administrator can easily recognize overloading situations in order to reconfigure the system. If failures arise during event processing, the administrator can intervene immediately and fix the problem.

6.3.3. Event Processing Model (EPM)

The processing steps, their relationships to each other, as well as the parameters of the analysis and data transformation processes can be individually defined for every organisation. SARESA uses an event processing model (EPM) for modelling Sense & Respond loops. Similar to a construction kit, the EPM offers various building blocks for Sense & Respond services which can be used to construct a Sense & Respond loop. Dependent on the requirements and the business problem, these building blocks can be flexibly conjoined or disconnected. Links between the building blocks represent a flow of events from one service to the next. The EPM allows for example:

- Definition of the structure for the processed events and data
- Interfaces to external systems for receiving data (Sense) and also for responding by executing business transactions (Respond)
- Data transformations, data analysis and persistence
- Definition of situations and exceptions in data to which a response should be triggered
- Modelling the data and control flows for the Sense & Respond loop (e.g. calculation of a metric always occurs before data analysis)
- Declaration of Sense & Respond services for processing steps including their input and output parameters
- Definition of the relationships and dependencies of Sense & Respond services and event data (e.g. data that has to be correlated before the processing starts)

Figure 6.5 shows the EPM of SARESA. Data is collected and received from source systems and continuously processed. The EPM shows the event flows of Sense & Respond processes and visualizes complex processing steps. For Sense & Respond loops it is vital that the processing steps work seamlessly together to enable a continuous and efficient execution of all processing stages. The EPM provides the following capabilities:

- **Flexible control of Sense & Respond loops**: Complete Sense & Respond processes can be modelled and user friendly visualized.
- **Capturing and integration** of events from various source systems.
- **Adaptiveness**: The processing steps of a Sense & Respond loop can be flexibly changed and updated.
- **Business Intelligence**: The processing steps can be combined with data analysis.
- **Real-time processing**: All processing steps are executed straight through without any delays.

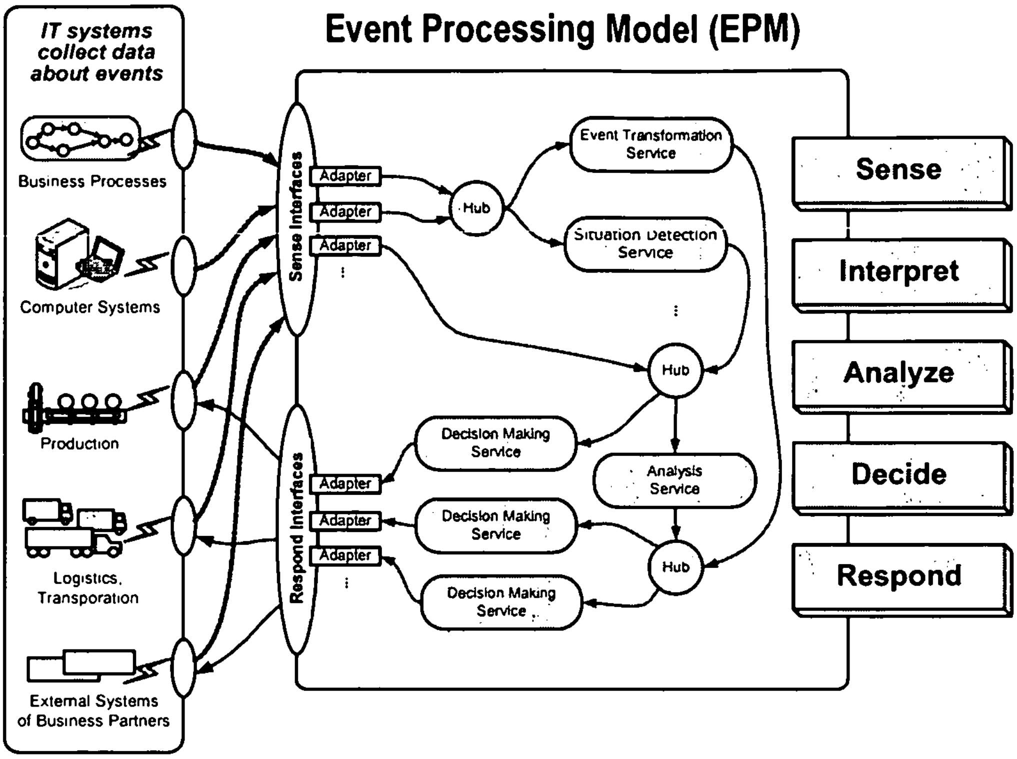


Figure 6.5. Event Processing Model

6.3.4. Analysis Service and Analytical Process Architecture

One of the main components of the SERESA architecture is the *Analysis Service* which conducts the real-time analytical process and publishes the relevant analysis result (Figure 6.6). Event streams from multiple source systems are collected and processed by an analysis service. For each event, the analytical processing is performed as follows. The analysis service invokes the analysis function supported by the *Analysis Server* to retrieve the analytical results relevant to that event. The Analysis Server (normally an OLAP server) uses the information integrator in order to get access to both real-time data and historical data. The zero-latency data warehouse contains both a real-time and a static partition. The real-time partition (real-time data store) contains the most recent metrics and analytical results and is continuously fed by event services. The static partition maintains a rich history of data which is integrated with a periodic batch-processing (usually over night or on weekend) by means of classical ETL (extraction/ transformation/loading) into the data warehouse.

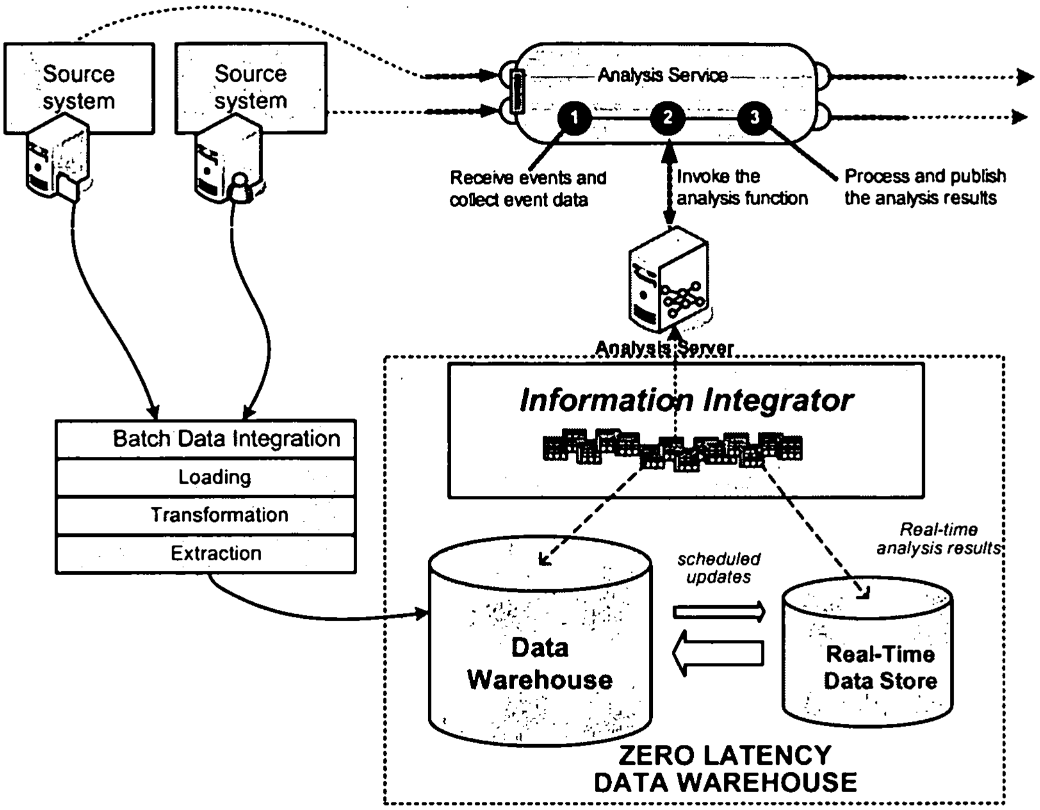


Figure 6.6. Analysis Service and Analytical Process Architecture

The above Analytical Process architecture with its analysis service has the following advantages compared to the traditional analysis approches in data warehouse systems:

- *Real time updated information*: The real-time data store is located in an separate database within the zero latency data warehouse, therefore its data could be updated instantly by the dedicated event service without any effect to the others analysis requests on the static data warehouse data.

- *Active mechanism to discovery of situations and exceptions*: During the event integrations, the analysis service could recognize certain conditions and could immediately react on it. In the classical ETL process, such conditions are uncovered periodically which is sometimes too late.

- *Pro-active response*: The analysis of real-time information by the analysis service allows the system to intelligently adapt and respond to the business environment. During the integration of events, the analysis service can predict problems in a business environment and can issue corresponding actions in time. This performs the main advantage of SARESA.

- *Automating decision making*: Traditional data warehouse systems provide a comprehensive history of data available to decision makers who use OLAP or data mining tools to analyze the data. In these cases, decisions are always made manually by humans. Thus, the analytical process architecture also allows us also to automate the decision making processes.

6.3.5. SARESA Advantages

The business value for organisations generated by SARESA can be summarized as follows:

- **Real-time business information**: Minimal latency for preparing and analysing data and hence improved visibility and accuracy of business performance indicators. The indicators are also available for operational and tactical decisions.
- **Optimized business processes**: By integrating real-time analytics business processes can be optimized by more efficient and intelligent control mechanisms. SARESA operates as an "external advisor" for the operative systems.
- **Situation discovery and business analytics**: The automatic discovery of business opportunities and exceptions in business processes. For instance, companies are able to detect suspicious customer behaviour (e.g. fraud behaviour) which can be countered with a proactive response.
- **Quick implementation and adjustments** of Sense & Respond loops in order to rapidly evaluate decision scenarios in a constantly changing business environment (e.g. marketing activities).
- **Proactive responses**: By continuously observing and analysing customers, business partners and the competition, business processes can be proactively adapted and optimized.
- **Generating more accurate forecasts** in near real-time under consideration of current and historic data. Example: Continuous update and optimization of production plans based on the current orders.
- **Integrating multiple external source systems:** SERESA effectively supports the correlation and merging of event streams from (potentially inter- organisational) business processes or a heterogeneous IT landscape.
- **Less integration effort**: Sense & Respond solutions with SARESA have a significantly lower integration effort than traditional Data Warehouse solutions since only events from the external source system have to be integrated. For the event processing SERESA does not have to know internal details of operational systems,

such as the data model. Consequently, SARESA can be integrated with business processes in a shorter period of time.

6.4. Real-time BI Implementation

This section discusses the implementation aspects of SERESA. As in the case of Gird-based Zero-Latency Data Stream Warehouse (GZLDSWH) described in Chapter 5, we select the Mobile Fraud Detection application to illustrate the features of SERESA. We then describe how the SERESA system services are deployed and the way the analysis service uses real-time features of OLAP Cube to conduct analytical process.

6.4.1 Mobile Fraud Detection Scenario

A Mobile Fraud Detection scenario is chosen to illustrate the Real-time event sense and response requirement. In this scenario, a timely analysis and response to prevent fraudulent activities is required. The most prominent fraud detection methods are based on the analysis of the usage patterns of mobile users. Call Detail Records (CDRs) are gathered as events and analyzed in order to generate business data such as calling time, geographic position of the mobile devices, call duration, and call frequency to recognize individual caller patterns of normal or fraudulent behaviour. In our application, it is a requirement that no CDR should be lost and a fraud should be countered as soon as it is detected.

Fraud is detected by checking some pre-defined rules. The mining approach to generate these rules is considered in a working paper [ZELESA 2005]. The rules can be of a complex nature such as "*an international mobile call from Austria to China of a certain customer lasts over 30 minutes will not be considered as a fraud if its duration is not over 1.5 times of his/her average call duration from Europe to Asia within the last 3 months, otherwise, it will be considered as a fraud and should be stopped immediately when it reaches such a thread hold*". The rules use aggregates which are provided and managed by an OLAP server (e.g. average call duration from Europe to Asia within the last 3 months for a certain user).

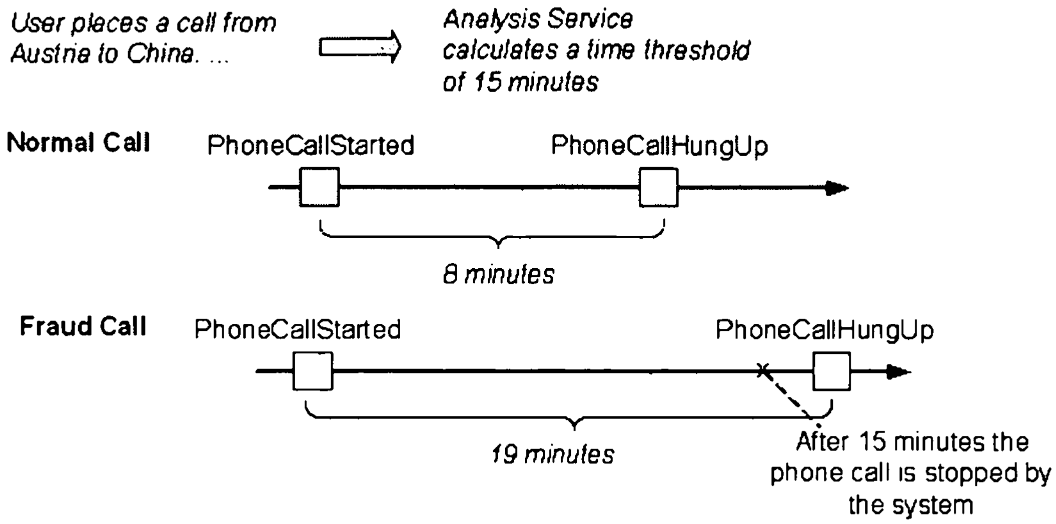

User places a call from Austria to China. ...
Analysis Service calculates a time threshold of 15 minutes
Normal Call
PhoneCallStarted
PhoneCallHungUp
8 minutes
Fraud Call
PhoneCallStarted
PhoneCallHungUp
19 minutes
After 15 minutes the phone call is stopped by the system

Figure 6.7. A fraudulent phone call from Austria to China

Figure 6.7 illustrate the Normal Call and Fraud Call from Austria to China of a certain customer. A phone call starts with the *PhoneCallStarted* event and ends with *PhoneCallHungUp* event. In the case of Normal Call, the *PhoneCallHungUp* occurs 8 minutes after the *PhoneCallStart* event happens. Because the fraud thread hold (which is calculated by the Analysis Service based on the historical CDR data of this customer) is 15 minutes, the 8 minute call is considered as a normal call. However, a Fraud Call which last longer than 15 minutes will be stopped immediately by the system when it reaches the threshhold. The *PhoneCallHungUp* in this case will never occur.

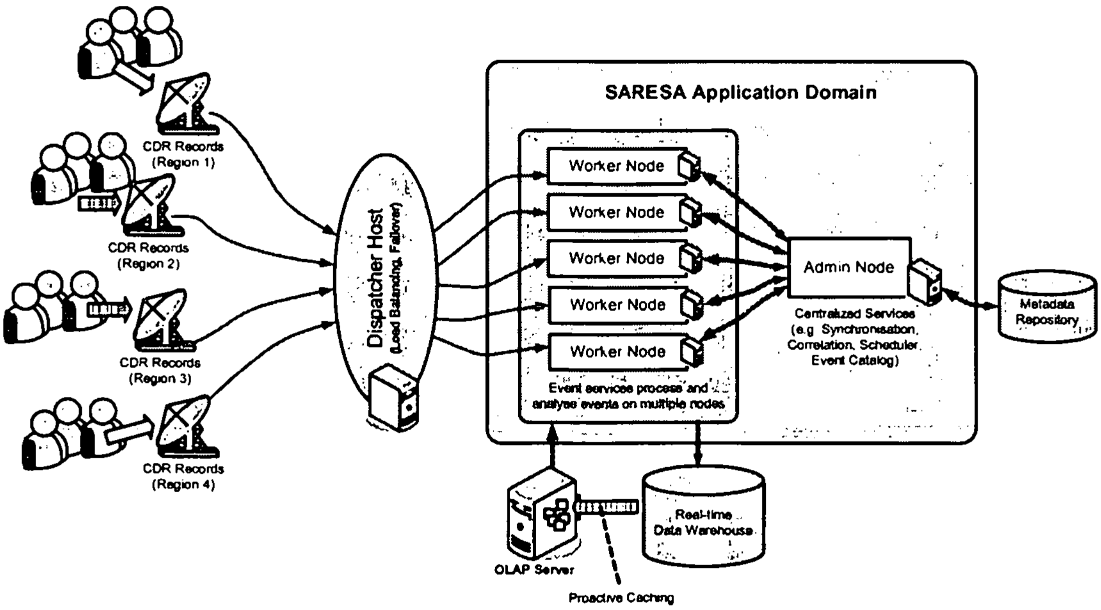
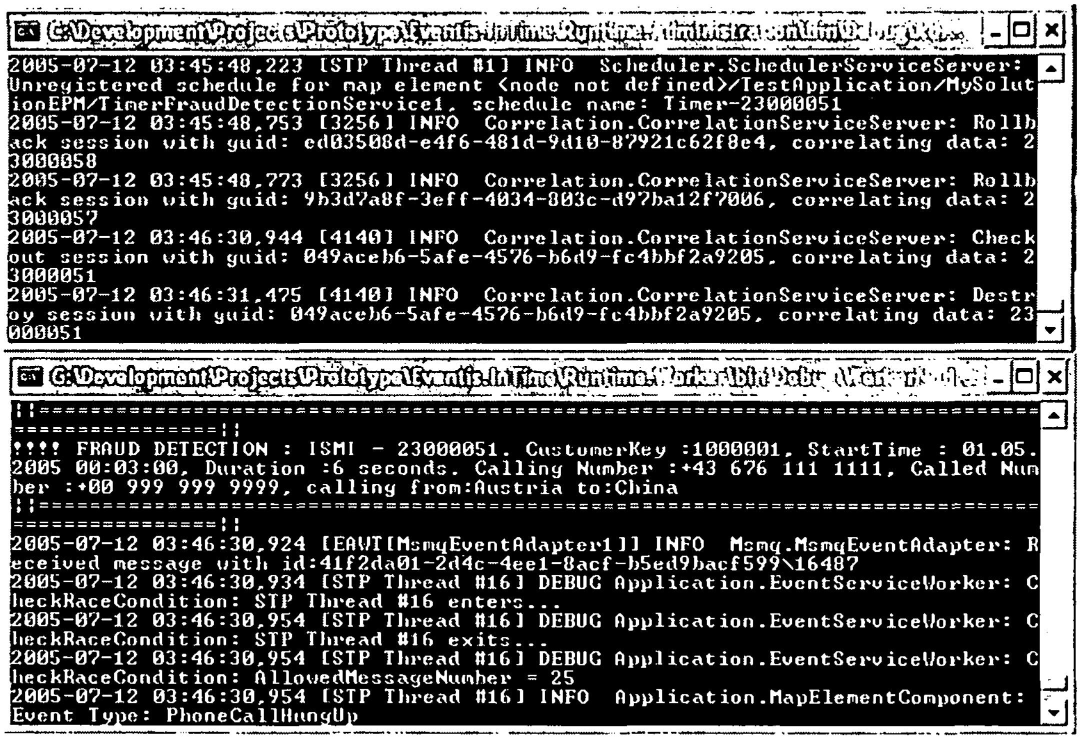


Figure 6.8. SERESA System Deployment

6.4.2. SARESA System Deployment

We have implemented the SARESA prototype and have used it to discover the fraud patterns for mobile phone calls. The prototype is implemented using Visual Studio.Net 2005 beta 2 and MS SQL Server 2005 beta 3.

Traditional Web applications architecture [Web 1999] has been proved as the "Best Practice" for various very scalable applications. To achieve high scalability, SARESA follows a similar approach for distributing load and fail over. Figure 6.8 depicts how applications are distributed in the SARESA system.



```
2005-07-12 03:45:48,223 [STP Thread #1] INFO  Scheduler.SchedulerServiceServer:
Unregistered schedule for map element <node not defined>/TestApplication/MySolut
ionEPM/TimerFraudDetectionService1, schedule name: Timer-23000051
2005-07-12 03:45:48,753 [3256] INFO  Correlation.CorrelationServiceServer: Rollb
ack session with guid: ed03508d-e4f6-481d-9d10-87921c62f8e4, correlating data: 2
3000058
2005-07-12 03:45:48,773 [3256] INFO  Correlation.CorrelationServiceServer: Rollb
ack session with guid: 9b3d7a8f-3eff-4034-803c-d97ba12f7006, correlating data: 2
3000057
2005-07-12 03:46:30,944 [4140] INFO  Correlation.CorrelationServiceServer: Check
out session with guid: 049aceb6-5afe-4576-b6d9-fc4bbf2a9205, correlating data: 2
3000051
2005-07-12 03:46:31,475 [4140] INFO  Correlation.CorrelationServiceServer: Destr
oy session with guid: 049aceb6-5afe-4576-b6d9-fc4bbf2a9205, correlating data: 23
000051
```

```
||==============================================================================
================||
!!!! FRAUD DETECTION : ISMI - 23000051. CustomerKey :1000001, StartTime : 01.05.
2005 00:03:00, Duration :6 seconds. Calling Number :+43 676 111 1111, Called Num
ber :+00 999 999 9999, calling from:Austria to:China
||==============================================================================
================||
2005-07-12 03:46:30,924 [EAWT[MsmqEventAdapter1]] INFO  Msmq.MsmqEventAdapter: R
eceived message with id:41f2da01-2d4c-4ee1-8acf-b5ed9bacf599\16487
2005-07-12 03:46:30,934 [STP Thread #16] DEBUG Application.EventServiceWorker: C
heckRaceCondition: STP Thread #16 enters...
2005-07-12 03:46:30,954 [STP Thread #16] DEBUG Application.EventServiceWorker: C
heckRaceCondition: STP Thread #16 exits...
2005-07-12 03:46:30,954 [STP Thread #16] DEBUG Application.EventServiceWorker: C
heckRaceCondition: AllowedMessageNumber = 25
2005-07-12 03:46:30,954 [STP Thread #16] INFO  Application.MapElementComponent:
Event Type: PhoneCallHungUp
```

Figure 6.9. Fraud Detection Service running at the Worker and Admin Node

Events (CDR records) from heterogeneous sources are collected, normalized and dispatched to multiple process nodes (Worker Nodes) by a centralized service (dispatcher host). On each worker node, an instance of a SARESA application is running to process and to manage the Sense & Response loops. Some universal services such as synchronization and event correlation must be provided by a central coordination node (Admin Node) which is called by worker nodes.

A central server (Dispatcher Host) receives the events from various event sources and distributes them to computer nodes (Worker Nodes). The dispatcher host thus controls the load balancing among the worker nodes and assures the failover and recovery in case a worker node fails. Each worker node is hosting an application instance which processes events of Sense & Response loops. A central admin node hosts centralized system services such as event correlation and synchronization

Figure 6.9 shows the console of the worker and the admin node. The admin console shows the correlation session management and timer management activities since these services are executed centrally on the admin node. The worker console shows the results of the event processing such as the detection of a fraud case.

6.4.3. Event Processing Model (EPM)

The Event Processing Model (EPM) which controls the event process order in Sense & Response loops is described in an XML document (*FraudDectionApplication.xml*). The EPM includes all application settings and parameters that are required for the event stream processing. Figure 6.10 shows a graphical representation of the EPM for the fraud detection. Section 6.4.5 will describe in more detail how these services mapping are declared using XML configuration file.

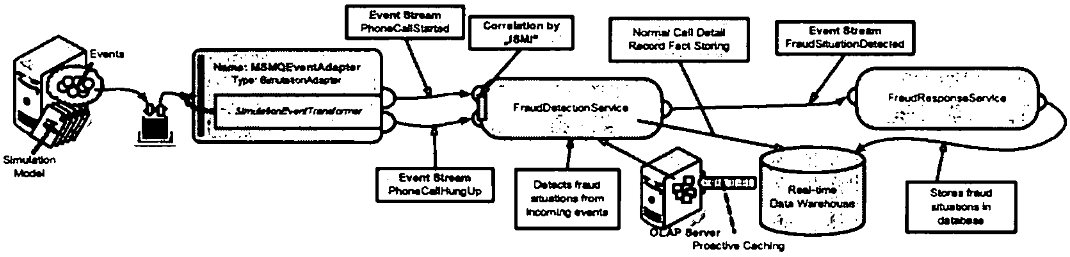

Events
Simulation Model
Name: MSMQEventAdapter
Type: SimulationAdapter
SimulationEventTransformer
Event Stream PhoneCallStarted
Correlation by „JSMI"
Event Stream PhoneCallHungUp
FraudDetectionService
Detects fraud situations from incoming events
Normal Call Detail Record Fact Storing
Event Stream FraudSituationDetected
FraudResponseService
OLAP Server
Proactive Caching
Real-time Data Warehouse
Stores fraud situations in database

Figure 6.10. Fraud Detection Event Processing Model (EPM)

In the prototype, we use a simulator to generate sample phone call events which are published to a message queue (see Section 6.4.6 for further information of Simulator). An event adapter in the EPM receives the sample phone call events from that message queue. In the EPM the event adapter has multiple connections with the *FraudDetectionService* which represent event flows.

The *FraudDetectionService* (Section 6.4.7) is a service to analyze the phone call and to decide whether it is fraudulent or not. The fraud detection policy and its logical process are conducted by this service. It receives *PhoneCallStarted* and *PhoneCallHungUp* events, conducts the multi-dimensional analysis, decides whether a phone call is a fraudulent call and raises in a fraud case the *FraudSituationDetected* event. Correlation sessions are used for correlating event pairs [Schiefer at al. 2004a] of *PhoneCallStarted* and *PhoneCallHungUp* to set a time threshold which indicates a fraud case (for e.g. 30 minutes). If a *PhoneCallHungUp* event is not received within the time threshold, a *FraudSituationDetected* event is raised. A time threshold is determined based on a multi-dimensional analysis of historical records of the user. In a fraud case, the *FraudResponseService* receives the *FraudDetectionStituation* event and issue the relevant response such as sending alarms to the customer, or stopping the phone call.

6.4.4. Event Types

To flexibly map such an event processing model while still keeping the consistency, we restrict events generated and processed in SERESA must follow the event types definition,

6.4.4.1. Definition of Event Types

Every event in the SERESA system has to conform to an event type. The event type defines the structure of the incoming events. It has a complete definition of all attributes that an event can have. All event type definitions must conform to EventType.xsd and the XML files must include the substring "EventType" in the file name (e.g. PhoneCallStarted**EventType**.xml).

```
<EventType catalogName="PhoneCallStarted">
	<Attributes>
		<Attribute catalogName="ISMI" reuseable="true">
			<DataType catalogName="String" multiValue="false">
				<RuntimeType>System.String</RuntimeType>
			</DataType>
			<Name>ISMI</Name>
			<Mandatory>true</Mandatory>
		</Attribute>
		...
		<Attribute catalogName="CallingParty" reuseable="true">
			<DataType catalogName="String" multiValue="false">
				<RuntimeType>System.String</RuntimeType>
			</DataType>
			<Name>CallingParty</Name>
			<Mandatory>true</Mandatory>
		</Attribute>
		<Attribute catalogName="CalledParty" reuseable="true">
			<DataType catalogName="String" multiValue="false">
				<RuntimeType>System.String</RuntimeType>
			</DataType>
			<Name>CalledParty</Name>
			<Mandatory>true</Mandatory>
		</Attribute>
		...
	<AllowUnknownAttributes>false</AllowUnknownAttributes>
	<ImplementationType>Eventis.Core.Event.BaseEvent</ImplementationType>
</EventType>
```

The following section explains a typical attribute definition.

```
<Attribute catalogName¹="ISMI" reuseable="true">
	<DataType catalogName²="String" multiValue="false">
		<RuntimeType>System.String³</RuntimeType>
	</DataType>
	<Name>ISMI⁴</Name>
	<Mandatory>true</Mandatory>
</Attribute>
```

(1) The name of the attribute.

(2) The name of the data type (can be any name – in the current version DataType. definitions are not catalogued – therefore, they are repeated for each attribute definition

(3) The .NET framework data type (e.g. System.String, System.Int32, System.Decimal, System.DateTime etc.).

(4) The name of the attribute type (can be any name – in the current version attribute type definitions are not catalogued – therefore, they are repeated for each attribute)

6.4.4.2. EventObject Types

EventObject types are used to define business objects that are sent with events. EventObject types define the structure of these business objects and can be reused in many event types. Examples for EventObject types are "Book", "Order", "Customer" etc. The following example shows an EventObject capturing the attributes of a book. This EventObject can be referenced by event types such as "BookSearchRequest" or "BookSold" etc. All EventObject type definitions must conform to EventType.xsd and the XML files must include the substring "EventObjectType" in the file name (e.g. Book**EventObjectType**.xml).

```
<EventObjectType catalogName="Book">
        <Attributes>
                <Attribute catalogName="ISBN" reuseable="true">
                        <DataType catalogName="String" multiValue="false">
                                <RuntimeType>System.String</RuntimeType>
                        </DataType>
                        <Name>ISBN</Name>
                        <Mandatory>true</Mandatory>
                </Attribute>
                <Attribute catalogName="BookTitle" reuseable="true">
                        <DataType catalogName="String" multiValue="false">
                                <RuntimeType>System.String</RuntimeType>
                        </DataType>
                        <Name>BookTitle</Name>
                        <Mandatory>true</Mandatory>
                </Attribute>
                <Attribute catalogName="Price" reuseable="true">
                        <DataType catalogName="Decimal" multiValue="false">
                                <RuntimeType>System.Decimal</RuntimeType>
                        </DataType>
                        <Name>Price</Name>
                        <Mandatory>true</Mandatory>
                </Attribute>
        </Attributes>
</EventObjectType>
```

The following event type definition shows how the EventObject "Book" can be referenced.

```
<EventType catalogName="BookSearchRequest">
    <Attributes>
        <Attribute catalogName="UserSessionID" reuseable="true">
            <DataType catalogName="String" multiValue="false">
                <RuntimeType>System.String</RuntimeType>
            </DataType>
            <Name>UserSessionID</Name>
            <Mandatory>true</Mandatory>
        </Attribute>
        <Attribute catalogName="Book" reuseable="true">
            <DataType catalogName="Book" multiValue="false">
```

```
                <RuntimeType catalogName="Book" />
            </DataType>
            <Name>Book</Name>
            <Mandatory>true</Mandatory>
        </Attribute>
    </Attributes>
    <AllowUnknownAttributes>false</AllowUnknownAttributes>
        <ImplementationType>Eventis.Core.Event.BaseEvent</ImplementationType>
</EventType>
```

6.4.5. Event Processing Model & Application Configuration file

The basics for the Event Processing Model are explained in the attached document. The Event Processing Model (EPM) is defined in an XML-file that must be conform to epm.xsd. The XML file name must include the substring "Application" in the file name (e.g. FraudDection**Application**.xml). The EPM includes all application settings and parameters that are required for the event stream processing. Refer to Figure 6.9 to see the Event Processing Model for the fraud detection.

6.4.5.1. Application Components Section

The Event Processing Model includes a section called "ApplicationComponents" that defines all application components that are somehow used within the Event Processing Model. The following shows this section for the fraud detection example:

```
<ApplicationComponents>
    <EventAdapterComponent name="MsmqEventAdapter">
            <ImplementationClass>Eventis.Components.EventAdapters.Msmq.MsmqEventAdapter</ImplementationClass>
            <OutputPort>
                <EventType name="PhoneCallStarted"/>
            </OutputPort>
            <OutputPort>
                <EventType name="PhoneCallHungUp"/>
            </OutputPort>
            <EventTransformer name="MsmqSimulationTransformer"/>
        </EventAdapterComponent>
        <EventTransformerComponent name="MsmqSimulationTransformer">

    <ImplementationClass>Eventis.Components.EventAdapters.Msmq.MsmqSimulationTransformer</ImplementationClass>
        </EventTransformerComponent>
        <EventServiceComponent name="TimerFraudDetectionService">
            <ImplementationClass>MySolution.EventServices.TimerFraudDetectionService</ImplementationClass>
            <ProcessInOrder>true</ProcessInOrder>
            <InputPort>
                <EventType name="PhoneCallStarted"/>
            </InputPort>
            <InputPort>
                <EventType name="PhoneCallHungUp"/>
            </InputPort>
            <OutputPort>
                <EventType name="FraudSituationDetected"/>
            </OutputPort>
            <CorrelationSessionSettings>
                <CorrelationSet>
                        <WaitingTimeout>200000</WaitingTimeout>
                        <CheckoutTimeout>200000</CheckoutTimeout>
                        <CorrelationData eventType="PhoneCallStarted">
                                <XPathSelector>
                                        <XPathExpression>//ISMI</XPathExpression>
                                </XPathSelector>
                        </CorrelationData>
                        <CorrelationData eventType="PhoneCallHungUp">
                                <XPathSelector>
                                        <XPathExpression>//ISMI</XPathExpression>
                                </XPathSelector>
                        </CorrelationData>
                </CorrelationSet>
                <Recoverable>true</Recoverable>
            </CorrelationSessionSettings>
        </EventServiceComponent>
        <EventServiceComponent name="FraudResponseService">
            <ImplementationClass>MySolution.EventServices.FraudResponseService</ImplementationClass>
            <ProcessInOrder>true</ProcessInOrder>
            <InputPort>
                <EventType name="FraudSituationDetected"/>
            </InputPort>
        </EventServiceComponent>
    </ApplicationComponents>
```

For specifying instances the **EventAdapterComponent MsmqEventAdapter** defines the implementation class, the input and output ports, and the EventTransformer for receiving event via messages from MSMQ (Microsoft Message Queuing Service). The **EventTransformerComponent MsmqSimulationTransformer** defines a components that translates MSMQ messages into standardized events. The two **EventServiceComponent sections (TimerFraudDetectionService, FraudResponse Service)** define the base configuration for the event services. After defining all components in the ApplicationComponents section, these components can be used in Event Processing Maps.

6.4.5.2. Event Processing Map Section

In this section, event processing components are used as elements of a map which can be linked together. First, all elements of a map (as illustrated in the above picture) must be listed. This is necessary since an application component could be potentially used multiple times in a single map or in multiple maps. Each map element has its **own name and can have its own set of parameters**. The following example shows how to define an MSMQ EventAdapterComponent as map element. The MSMQ EventAdapter element has some parameters for indicating the queue from which messages (=events) should be received.

```
<EventProcessingMap name="MySolutionEPM">
        <EventAdapterMapElement name="MsmqEventAdapter1"
        componentName="MsmqEventAdapter">
                <InitParameters>
                        <Parameter name="QueueName">
                                <Value>.\Private$\TestQueue</Value>
                        </Parameter>
                        <Parameter name="Mode">
                                <Value>Transactional</Value>
                        </Parameter>
                </InitParameters>
        </EventAdapterMapElement>
...
```

Similar to the EventAdapters also EventServices can be defined as elements with their own parameters. The following example shows the TimerFraudDetectionService.

```
        <EventServiceMapElement name="TimerFraudDetectionService1" componentName="TimerFraudDetectionService">
                <InitParameters>
                        <Parameter name="MyParam">
                                <Value>dsfsdfsdfsdfsdf</Value>
                        </Parameter>
                        <Parameter name="MyParamSet">
                                        <ParameterSet>
                                                <Parameter name="Param1">
                                                        <Value>value1</Value>
                                                </Parameter>
                                                <Parameter name="Param2">
                                                        <Value>value2</Value>
                                                </Parameter>
                                        </ParameterSet>
                        </Parameter>
                </InitParameters>
        </EventServiceMapElement>
```

After defining the map elements, the elements can be linked with each other. This can be done by connectors and hubs. The following example shows how to link an EventAdapter with an Event Service with a connector. Every connector represents an event stream that supports a certain event type.

```
<Connector>
        <From>
                <MapElement name="MsmqEventAdapter1" eventType="PhoneCallStarted"/>
        </From>
        <To>
                <MapElement name="TimerFraudDetectionService1"
                eventType="PhoneCallStarted"/>
        </To>
</Connector>
```

Hubs can be used to combine event streams or split event streams. The following example shows a hub that receives and dispatches ProcessStarted and ProcessCompleted events. Please note that multiple EventAdapters can pump ProcessStarted and ProcessCompleted events into the hub.

```
<Hub name="Hub1">
        <InputPort>
                <EventType name="ProcessStarted"/>
        </InputPort>
        <InputPort>
                <EventType name="ProcessCompleted"/>
        </InputPort>
        <OutputPort>
                <EventType name="ProcessStarted"/>
        </OutputPort>
        <OutputPort>
                <EventType name="ProcessCompleted"/>
        </OutputPort>
</Hub>
```

6.4.5.3. Filters

Connectors can also include filters. The following example connects ProcessStarted events from a Hub ("Hub1") with the EventService "EventService1". However only ProcessStarted events with a processed <=20 are forwarded.

```
<Connector>
        <From>
                <MapElement name="Hub1" eventType="ProcessStarted"/>
        </From>
        <To>
                <MapElement name="EventService1" eventType="ProcessStarted"/>
        </To>
        <Filter>
                <XPathSelector>
                        <XPathExpression>//ProcessID[number(.)&lt;=20]</XPathExpression>
                </XPathSelector>
        </Filter>
</Connector>
```

6.4.5.4. Data Sources

The EPM also includes a section for data sources that can be used in EventAdapters or EventServices. The following data sources are currently supported: SqlDB, OleDB, OdbcDB, and OracleDB. The following example defines a SQL Server data source. The data source definition includes the data source type ("SqlDB") and the connection string:

```
        <DataSources>
                <DataSource name=" FraudDetectionDB" dbType="SqlDB">

        <ConnectionString>server=192.168.0.2;database=FraudDetectionDB;User
ID=sa;Password=test</ConnectionString>
                </DataSource>
        </DataSources>
```

The data sources can be accessed in an EventService with the method GetDataSource(). The following example shows how this method can be used:

```
DbConnection conn = GetDataSource("FraudDetectionDB").CreateConnection();
conn.Open();
try
{
        DbCommand cmd = conn.CreateCommand();
        cmd.CommandText = "INSERT INTO FraudSituations (ISMI) VALUES(@ISMI)";
        SqlParameter myParam = new SqlParameter("@ISMI", SqlDbType.VarChar, 50);
        myParam.Value = ismi;
        cmd.Parameters.Add(myParam);
        int result = cmd.ExecuteNonQuery();
        if (result != 1) throw new EventServiceException("Insert failed.");
        cmd.Dispose();
}
finally {
        conn.Close();
}
```

6.4.5.5. Correlation

Event Services can use correlation sessions (**Note: Throughout this dissertation the term "correlation" should not be understood in the sense of correlation is statistics – it should be used as a synomym of "relatedness"**). The following section shows how the PhoneCallStarted event can be correlated with the PhoneCallHungUp event by ISMI

```
<CorrelationSessionSettings>
                <CorrelationSet>
                        <WaitingTimeout>200000</WaitingTimeout>
                        <CheckoutTimeout>200000</CheckoutTimeout>
                        <CorrelationData eventType="PhoneCallStarted">
                                <XPathSelector>
                                        <XPathExpression>//ISMI</XPathExpression>
                                </XPathSelector>
                        </CorrelationData>
                        <CorrelationData eventType="PhoneCallHungUp">
                                <XPathSelector>
                                        <XPathExpression>//ISMI</XPathExpression>
                                </XPathSelector>
                        </CorrelationData>
                </CorrelationSet>
                <Recoverable>true</Recoverable>
</CorrelationSessionSettings>
```

A correlation set defines correlation data from multiple event types. Each event type can have its own way of retrieving selected event attributes for the correlation. Therefore, every event type has its own XPathSelector in order to retrieve this attributes. If there is no complex logic required for retrieve a single event attribute, also an EventAttributeSelector (instead of the XPathSelector) can be used. The EventAttributeSelector is easier to use if only a single event attribute is required. The following example shows the usage of the EventAttributeSelector:

```
<CorrelationData eventType="PhoneCallStarted">
        <EventAttributeSelector>
                <EventAttributeValue name="ISMI">
        </EventAttributeSelector>
</CorrelationData>
<CorrelationData eventType="PhoneCallHungUp">
        <EventAttributeSelector>
                <EventAttributeValue name="ISMI">
        </EventAttributeSelector>
        </CorrelationData>
</CorrelationSet>
```

Correlation session can be used concurrently. However, a correlation session can only be used at a single Event Service at the same time. Therefore, sometimes Event Services have to wait to retrieve a correlation session. In the <WaitingTimeout> section you can define the maximal time how long the event service should wait in milliseconds to retrieve the correlation session. If the correlation session cannot be retrieved in this time the system will automatically throw an exception and start the exception handling process.

```
<WaitingTimeout>200000</WaitingTimeout>
```

The CheckoutTimeout section defines how long (in milliseconds) a correlation session can be used in an event services. If the processing takes longer, the system will deny accepting the correlation session and automatically throw an exception. This parameter is important when Event Services "hang" on a certain node and won't release the session for other nodes.

```
<CheckoutTimeout>200000</CheckoutTimeout>
```

The Recoverable section defines whether the session data should be stored in memory or in the database. Recoverable true means that the sessions will be stored in the database (in order words – the session data won't be lost due to a computer crash).

```
<Recoverable>true</Recoverable>
```

6.4.5.6. Synchronsiation

Synchronisation is used to partially serialize the event processing. By defining synchronisation blocks the processing in multiple event service can be serialized. The following example shows a synchronisation of 2 event services (EventService1 and EventService2). Both event services process ProcessStarted and ProcessCompleted events. By defining this synchronisation block, the system ensures that no event (ProcessStarted or ProcessCompleted) with the same processID will be processed simultaneously:

```
<SynchronizationBlock name="SyncBlock">
                <CorrelationSet>
                        <CorrelationData eventType="ProcessStarted">
                                <XPathSelector>
<XPathExpression>//ProcessID</XPathExpression>
                                </XPathSelector>
                        </CorrelationData>
                </CorrelationSet>
                <CorrelationSet>
                        <CorrelationData eventType="ProcessCompleted">
                                <XPathSelector>
<XPathExpression>//ProcessID</XPathExpression>
                                </XPathSelector>
                        </CorrelationData>
                </CorrelationSet>
                <MaxParallel>1</MaxParallel>
                <WaitingTimeout>1000000</WaitingTimeout>
                <LockTimeout>2000000</LockTimeout>
                <EventServiceMapElement name="EventService1"/>
                <EventServiceMapElement name="EventService2"/>
</SynchronizationBlock>
```

6.4.6. The Simulator

6.4.6.1. Simulation Scenarios

Simulation scenarios define how the default settings for simulating event sequences. These settings include the number of instances for an event sequence and whether the simulation should be executed serialized or parallel.

```
<SimulationScenario name="SS_NormalAndFraudPhoneCall">
        <EventSequenceSimulation>
                <EventSequence name="ES_NormalAndFraudPhoneCall"/>
                <Instances>100</Instances>
                <Concurrency>Parallel</Concurrency>
        </EventSequenceSimulation>
</SimulationScenario>
```

6.4.6.2. Event Receiver

Event Receivers define targets for the event simulation. In most cases the target is a queue or a database table. Currently, only MSMQ queues are supported. The following example shows an EventReceiver for a MSMQ queue. The messages are send in a transactional mode.

```
<EventReceiver name="TestReceiver">
        <MsmqQueue>
                <QueueName>.\Private$\TestQueue</QueueName>
                <Transactional>true</Transactional>
        </MsmqQueue>
</EventReceiver>
```

6.4.6.3. Event Sequences

Event Sequences define the sequences and causal relationships between events. The following example shows an event sequence with the two events PhoneCallStarted and PhoneCallHungUp. Both are correlated by ISMI (in other words the ISMI will be in both events the same in an instance of an event sequence). The DataSequence section indicates that there is a temporal relationship of the StartTime attribute (of PhoneCallStarted) and the EndTime attribute (of PhoneCallHungUp) and that the EndTime should be always 4 seconds after the StartTime.

```
<EventSequence name="ES_NormalPhoneCall">
        <EventPattern name="PhoneCallStarted"/>
        <Wait>
                <Seconds>4</Seconds>
        </Wait>
        <EventPattern name="PhoneCallHungUp"/>
        <EventCorrelation>
                <Attribute name="ISMI"/>
        </EventCorrelation>
        <DateSequence>
                <Attribute name="StartTime"/>
                <Attribute name="EndTime"/>
                <Increment>
                        <Seconds>4</Seconds>
                </Increment>
        </DateSequence>
</EventSequence>
```

6.4.6.4. Event Patterns

Event Patterns define the structure of events. These event patterns must be in sync with the event types. The following example shows the PhoneCallHungUp event pattern. The values for the attributes are generated by so called ValueGenerators which are explained in the next section.

```
<EventPattern name="PhoneCallHungUp">
        <EventType>PhoneCallHungUp</EventType>
        <Recoverable>false</Recoverable>
        <Priority>Medium</Priority>
        <Attribute name="ISMI">
                <ValueGenerator name="VG_ISMI"/>
        </Attribute>
        <Attribute name="EndTime">
                <ValueGenerator name="VG_DateTime2004"/>
        </Attribute>
</EventPattern>
```

Please note that you can use also fixed values instead of value generators. These values can be defined as follows:

```
<Attribute name="ISMI">
        <Value>0234234234234234</Value>
</Attribute>
```

6.4.6.5. Value Generators

Value generators generate the values for the simulation. Values can be a number, date or string. The following example shows how a number generator for generating a customer ID can be defined. The customer ID starts with 1000000 and ends with 20000000. The generator always increments the latest used customerID by 1. The data type is System.Int32 (could be also System.Int64, System.Double, System.Decimal etc.)

```
        <ValueGenerator name="VG_CustomerID">
                <NumRange>
                        <Start>1000000</Start>
                        <Stop>2000000</Stop>
                        <Increment>1</Increment>
                </NumRange>
                <DataType>System.Int32</DataType>
        </ValueGenerator>
```

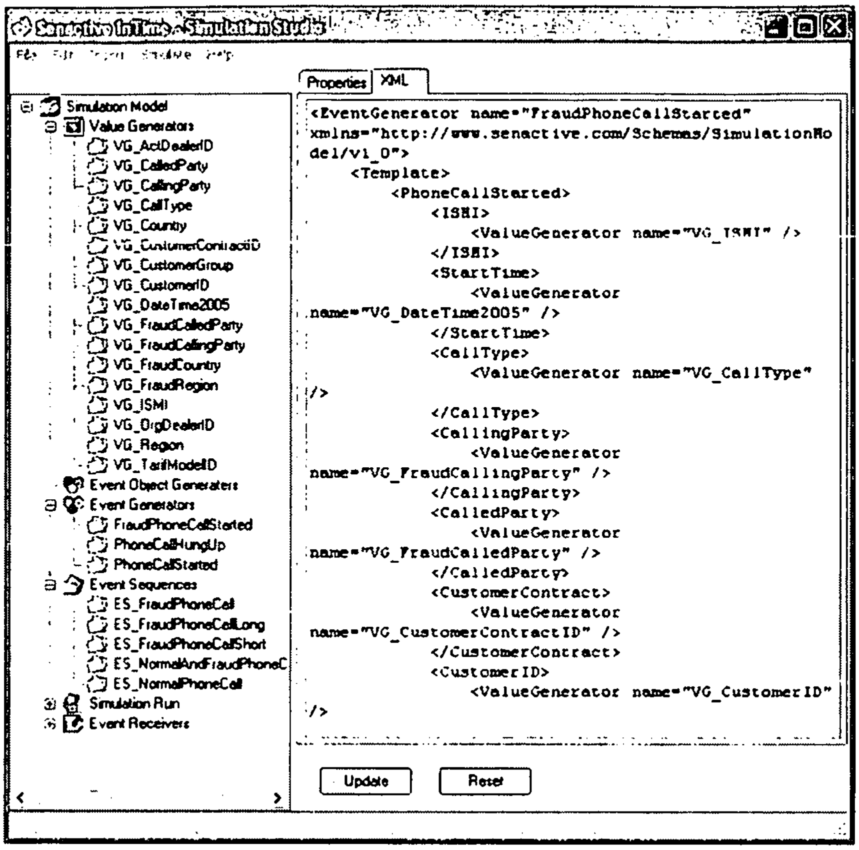
The next example shows a value generator for dates:

```
<ValueGenerator name="VG_DateTime2004">
                <DateRange>
                        <Start>2004-01-01</Start>
                        <Stop>2004-12-31</Stop>
                        <Increment>
                                <Minutes>3</Minutes>
                        </Increment>
                </DateRange>
                <DataType>System.DateTime</DataType>
</ValueGenerator>
```

The next example shows a value generator for strings. A String is selected from a pool of strings defined by an enumeration:

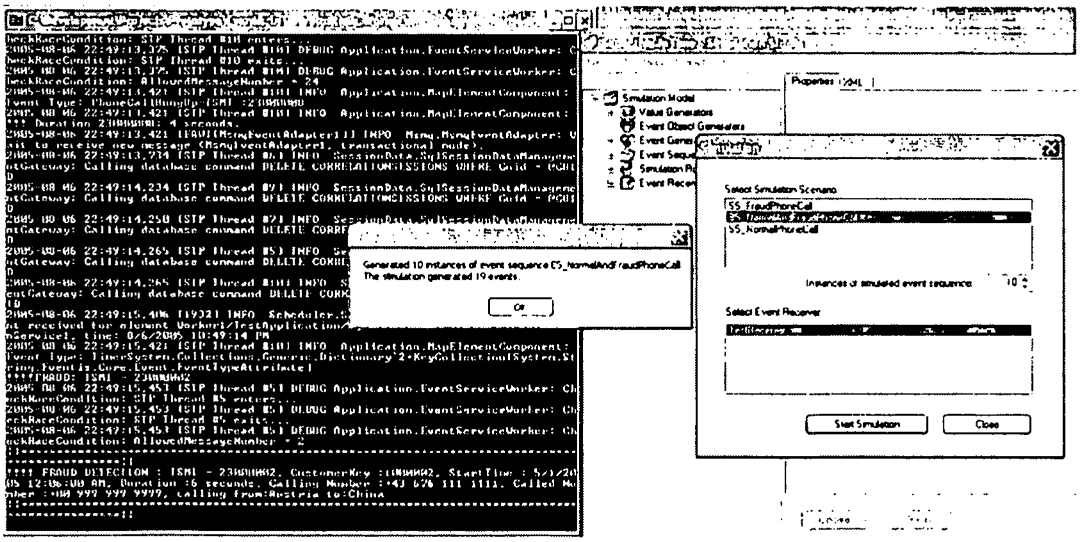
```
        <ValueGenerator name="VG_CalledParty">
                <Enumeration>
                        <Element>+43 650 323 322</Element>
                        <Element>+43 669 544 9090</Element>
                        <Element>+43 650 580 3390</Element>
                        <Element>+43 650 180 9090</Element>
                        <Element>+43 669 590 9090</Element>
                        <Element>+43 664 520 9090</Element>
                        <Element>+43 650 280 9090</Element>
                        <Element>+43 669 980 9090</Element>
                </Enumeration>
                <DataType>System.String</DataType>
        </ValueGenerator>
```

Figure 6.11 shows the simulation Studio where all Simulation Model declarations are listed. We can modify the value of XML tags to generate the dynamic simulation. The normal and fraud phone call scenario simulation events are demonstrated in Figure 6.12



```
<EventGenerator name="FraudPhoneCallStarted"
xmlns="http://www.senactive.com/Schemas/SimulationMo
del/v1_0">
    <Template>
        <PhoneCallStarted>
            <ISMI>
                <ValueGenerator name="VG_ISMI" />
            </ISMI>
            <StartTime>
                <ValueGenerator
name="VG_DateTime2005" />
            </StartTime>
            <CallType>
                <ValueGenerator name="VG_CallType"
/>
            </CallType>
            <CallingParty>
                <ValueGenerator
name="VG_FraudCallingParty" />
            </CallingParty>
            <CalledParty>
                <ValueGenerator
name="VG_FraudCalledParty" />
            </CalledParty>
            <CustomerContract>
                <ValueGenerator
name="VG_CustomerContractID" />
            </CustomerContract>
            <CustomerID>
                <ValueGenerator name="VG_CustomerID"
/>
```

Figure 6.11. Simulation Studio

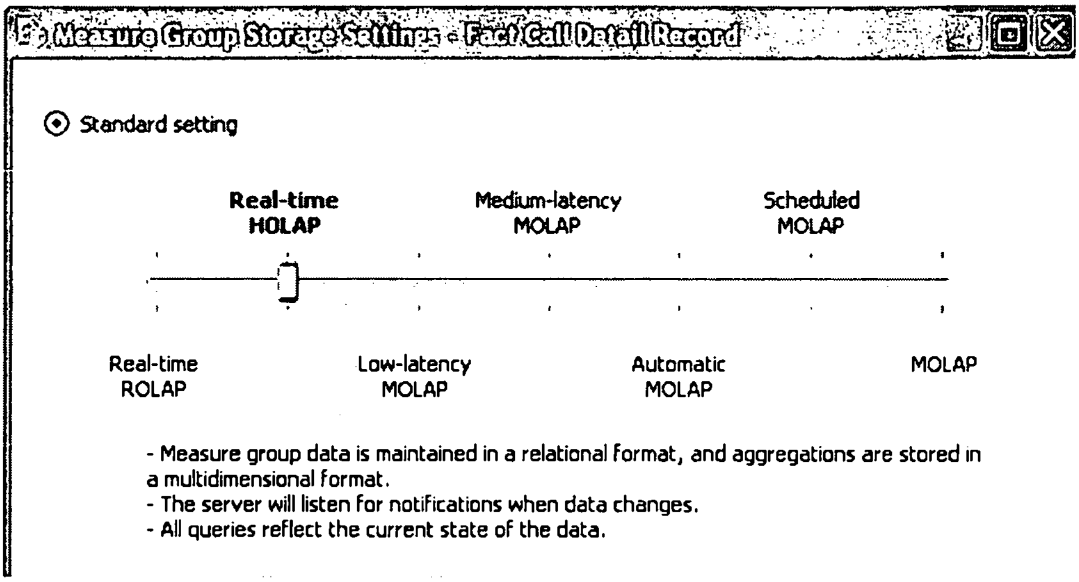


```
heckRaceCondition: STP Thread #10 enters...
2005-08-06 22:49:13,375 [STP Thread #10] DEBUG Application.EventServiceWorker: C
heckRaceCondition: STP Thread #10 exits...
2005-08-06 22:49:13,375 [STP Thread #10] DEBUG Application.EventServiceWorker: C
heckRaceCondition: AllowedMessageNumber = 24
2005-08-06 22:49:13,421 [STP Thread #10] INFO  Application.MapElementComponent:
Event Type: PhoneCallHungUp-ISMI :23000000
2005-08-06 22:49:13,421 [STP Thread #10] INFO  Application.MapElementComponent:
*** Duration 23000000: 4 seconds.
2005-08-06 22:49:13,421 [FAU][MsmqEventAdapter1] INFO  Msmq.MsmqEventAdapter: W
ait to receive new message (MsmqEventAdapter1, transactional mode).
2005-08-06 22:49:13,234 [STP Thread #6] INFO  SessionData.SqlSessionDataManageme
ntGateway: Calling database command DELETE CORRELATIONSESSIONS WHERE Guid = @GUI
D
2005-08-06 22:49:14,234 [STP Thread #9] INFO  SessionData.SqlSessionDataManageme
ntGateway: Calling database command DELETE CORRELATIONSESSIONS WHERE Guid = @GUI
D
2005-08-06 22:49:14,250 [STP Thread #2] INFO  SessionData.SqlSessionDataManageme
ntGateway: Calling database command DELETE CORRE
D
2005-08-06 22:49:14,265 [STP Thread #5] INFO  Se
ntGateway: Calling database command DELETE CORR
D
2005-08-06 22:49:14,265 [STP Thread #10] INFO  S
entGateway: Calling database command DELETE CORR
ID
2005-08-06 22:49:15,406 [1932] INFO  Scheduler.S
nt received for element Worker1/TestApplication/
nService1, time: 8/6/2005 10:49:14 PM
2005-08-06 22:49:15,421 [STP Thread #10] INFO  Application.MapElementComponent:
Event Type: TimerSystem.Collections.Generic.Dictionary`2+KeyCollection[System.St
ring,Eventis.Core.Event.EventTypeAttribute]
????FRAUD: ISMI - 23000002
2005-08-06 22:49:15,453 [STP Thread #5] DEBUG Application.EventServiceWorker: Ch
eckRaceCondition: STP Thread #5 enters...
2005-08-06 22:49:15,453 [STP Thread #5] DEBUG Application.EventServiceWorker: Ch
eckRaceCondition: STP Thread #5 exits...
2005-08-06 22:49:15,453 [STP Thread #5] DEBUG Application.EventServiceWorker: Ch
eckRaceCondition: AllowedMessageNumber = 2
||------------------------------------------------------------------------------
-----------------||
???? FRAUD DETECTION : ISMI - 23000002, CustomerKey :1000002, StartTime : 5/1/20
05 12:06:00 AM, Duration :6 seconds, Calling Number :+43 676 111 1111, Called Nu
mber :+00 999 999 9999, calling from:Austria to:China
||------------------------------------------------------------------------------
-----------------||
```

Generated 10 instances of event sequence ES_NormalAndFraudPhoneCall
The simulation generated 19 events.

Figure 6.12. Normal and Fraud Phone Call Events generated by Simulation Studio

6.4.7. Real-time Analysis Service



● Standard setting

- Measure group data is maintained in a relational format, and aggregations are stored in a multidimensional format.
- The server will listen for notifications when data changes.
- All queries reflect the current state of the data.

Figure 6.13. Real-time HOLAP Cube Structure in MS Analysis Server

The current version of SQL Server 2005 Beta 3 supports Real-time OLAP. For SARESA, we use these real-time analysis capabilities and deploy and manage the OLAP cubes with the MS Analysis Service which is part of SQL Server 2005 (see Figure 6.13).

We have created an Analysis Project, created and deployed the cube in MS SQL Server 2005. The real-time data cache and refresh of the OLAP cube is automatically managed by MS SQL Server 2005. Figure 6.14 shows the cube structure of the *FraudDetectionDW* which is designed for the Fraud Detection application. There are 2 facts (FactCallDetailRecord and FactFraudSituation) connected to several dimensions such as DimCallType, DimContractType, DimCustomer, DimTarifModel, DimCallingParty, DimCalledParty, DimTime, etc. All the call detail records are stored in FactCallDetailRecord. All detected Fraud calls are stored into the FactFraudSituation. The only interested measure is the call duration. Figure 6.15 presents the cube browser where we could browse and analyze the call detail record data in multiple of hierarchical levels.

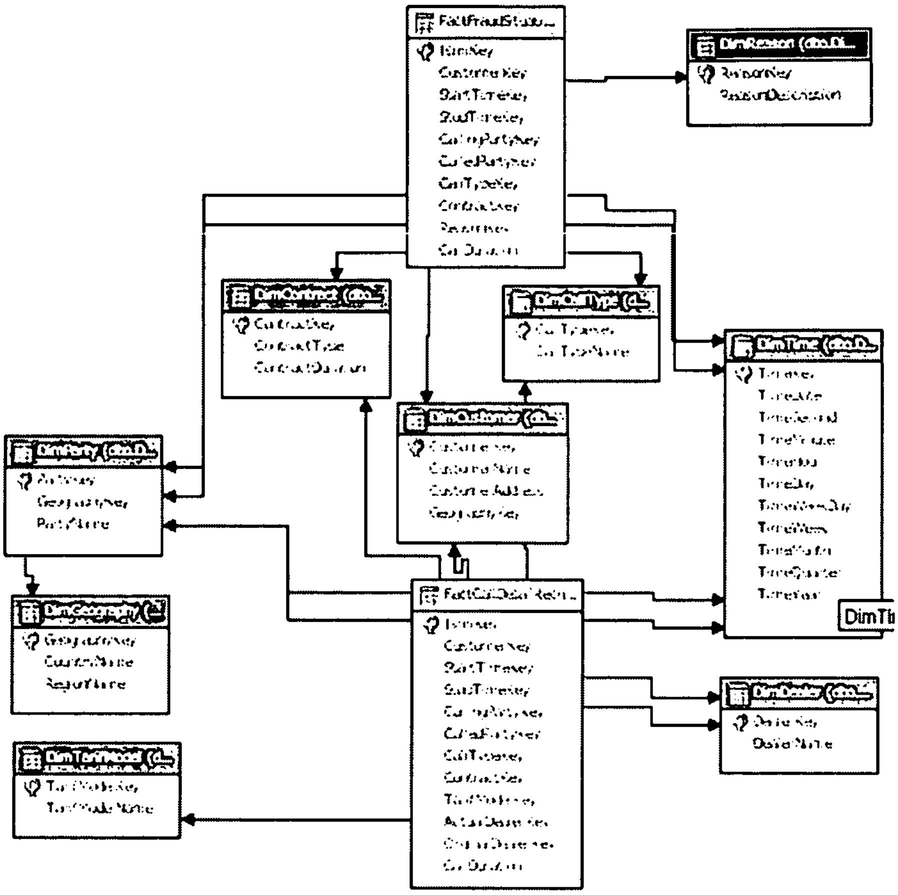


Figure 6.14. FraudDetectionDW cube structure

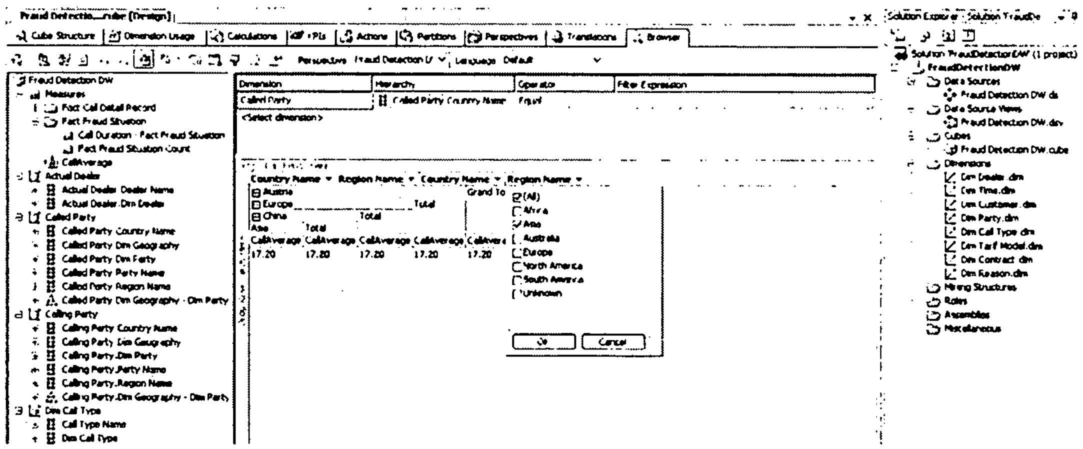
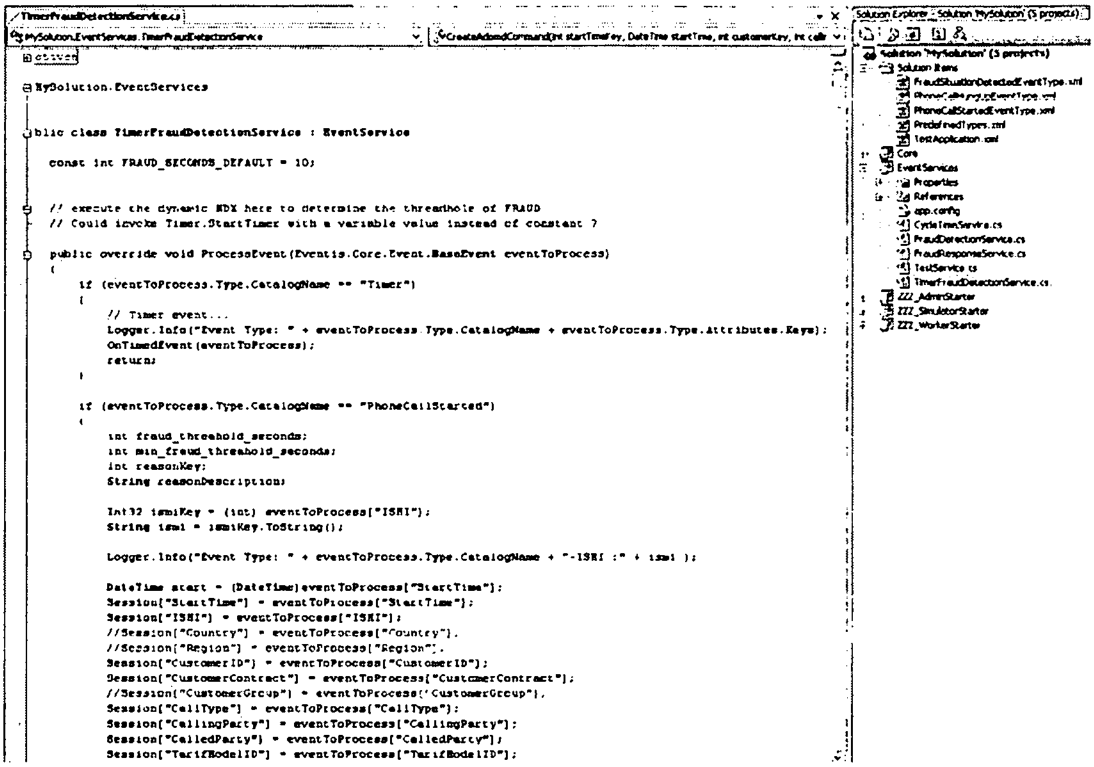


Figure 6.15. FraudDectectionDW Cube Browser

The analysis service for the fraud detection is implemented as an event service which is running at the worker nodes (*TimerFraudDetection.cs*). The service receives the PhoneCallStarted event and conducts the real-time OLAP analysis to evaluate whether the call is fraudulent or not. For each event PhoneCallStarted, the analytical service executes MDX queries and calculates the time threshold for a potential fraud case, i.e.

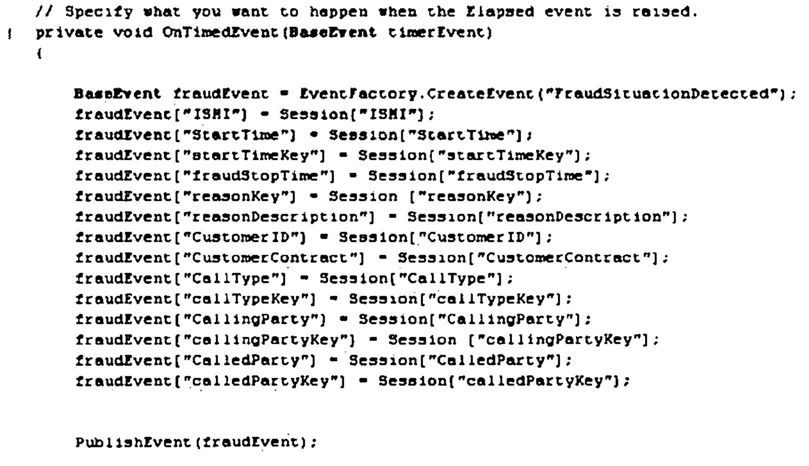
*1.5 * average phone call from caller's continent to receiver's continent in the last 3 months*. This time threshold is used to set a timer to prevent fraudulent phone call. (see Figure 6.16.)


TimerFraudDetectionService.cs
MySolution.EventServices.TimerFraudDetectionService
CreateAdomdCommand(int startTimeKey, DateTime startTime, int customerKey, int calle
Solution Explorer - Solution 'MySolution' (5 projects)
Solution 'MySolution' (5 projects)
Solution Items
FraudSituationDetectedEventType.xml
PhoneCallHangUpEventType.xml
PhoneCallStartedEventType.xml
PredefinedTypes.xml
TestApplication.xml
Core
EventServices
Properties
References
app.config
CycleTimeService.cs
FraudDetectionService.cs
FraudResponseService.cs
TestService.cs
TimerFraudDetectionService.cs
ZZZ_AdminStarter
ZZZ_SimulatorStarter
ZZZ_WorkerStarter

```
using
MySolution.EventServices

ublic class TimerFraudDetectionService : EventService

   const int FRAUD_SECONDS_DEFAULT = 10;

   // execute the dynamic MDX here to determine the threadhole of FRAUD
   // Could invoke Timer.StartTimer with a variable value instead of constant ?

   public override void ProcessEvent(Eventis.Core.Event.BaseEvent eventToProcess)
   {
       if (eventToProcess.Type.CatalogName == "Timer")
       {
           // Timer event...
           Logger.Info("Event Type: " + eventToProcess.Type.CatalogName + eventToProcess.Type.Attributes.Keys);
           OnTimedEvent(eventToProcess);
           return;
       }

       if (eventToProcess.Type.CatalogName == "PhoneCallStarted")
       {
           int fraud_threshold_seconds;
           int min_fraud_threshold_seconds;
           int reasonKey;
           String reasonDescription;

           Int32 ismiKey = (int) eventToProcess["ISMI"];
           String ismi = ismiKey.ToString();

           Logger.Info("Event Type: " + eventToProcess.Type.CatalogName + "-ISMI :" + ismi );

           DateTime start = (DateTime)eventToProcess["StartTime"];
           Session["StartTime"] = eventToProcess["StartTime"];
           Session["ISMI"] = eventToProcess["ISMI"];
           //Session["Country"] = eventToProcess["Country"].
           //Session["Region"] = eventToProcess["Region"].
           Session["CustomerID"] = eventToProcess["CustomerID"];
           Session["CustomerContract"] = eventToProcess["CustomerContract"];
           //Session["CustomerGroup"] = eventToProcess["CustomerGroup"].
           Session["CallType"] = eventToProcess["CallType"];
           Session["CallingParty"] = eventToProcess["CallingParty"];
           Session["CalledParty"] = eventToProcess["CalledParty"];
           Session["TariffModelID"] = eventToProcess["TariffModelID"];
```

Figure 6.16. TimerFrauDetection.cs

If that timer is expired, the analysis service generates the *FraudDetectionStituation* event. (Figure 6.17.) The response service (*FraudResponseService.cs*) consumes that event, stores information about the fraudulent call into the *FactFraudStituation* table of the *FraudDetectionDW* database and issues the alarms to the user.



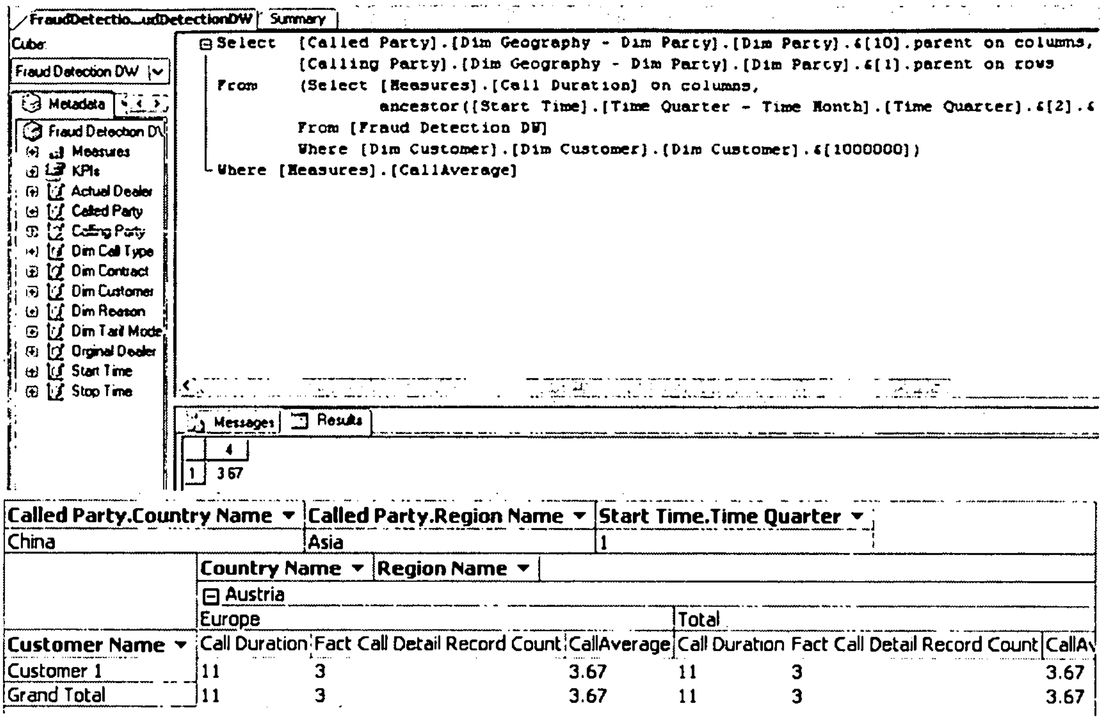
```
// Specify what you want to happen when the Elapsed event is raised.
private void OnTimedEvent(BaseEvent timerEvent)
{

    BaseEvent fraudEvent = EventFactory.CreateEvent("FraudSituationDetected");
    fraudEvent["ISMI"] = Session["ISMI"];
    fraudEvent["StartTime"] = Session["StartTime"];
    fraudEvent["startTimeKey"] = Session["startTimeKey"];
    fraudEvent["fraudStopTime"] = Session["fraudStopTime"];
    fraudEvent["reasonKey"] = Session ["reasonKey"];
    fraudEvent["reasonDescription"] = Session["reasonDescription"];
    fraudEvent["CustomerID"] = Session["CustomerID"];
    fraudEvent["CustomerContract"] = Session["CustomerContract"];
    fraudEvent["CallType"] = Session["CallType"];
    fraudEvent["callTypeKey"] = Session["callTypeKey"];
    fraudEvent["CallingParty"] = Session["CallingParty"];
    fraudEvent["callingPartyKey"] = Session ["callingPartyKey"];
    fraudEvent["CalledParty"] = Session["CalledParty"];
    fraudEvent["calledPartyKey"] = Session["calledPartyKey"];


    PublishEvent(fraudEvent);
```

Figure 6.17. The processing when timer is expired.

```
//Determine the FRAUD_THREADHOLD_SECONDS

using (AdomdConnection connection = new AdomdConnection ("Data Source=localhost;Catalog=FraudDetectionDW"))
{
connection.Open();
AdomdCommand command = new AdomdCommand(CreateAdomdCommand(startTimeKey,start,customerKey,callingPartyKey,calledPartyKey), connection);
CellSet set = command.ExecuteCellSet();
double average_callduration_seconds;
if (set[0].Value == null)
{
    average_callduration_seconds = 0;
}
else
{
    average_callduration_seconds = (double) set[0].Value;
}

fraud_threahold_seconds = (int) Math.Round(1.5 * average_callduration_seconds);
connection.Close;
}

// set the timer threadhold seconds
if ((fraud_threahold_seconds > FRAUD_SECONDS_DEFAULT) || (fraud_threahold_seconds <= 0))
{
    min_fraud_threahold_seconds = FRAUD_SECONDS_DEFAULT;
    reasonKey = 1; // call too long
    reasonDescription = "Call is too long";

}

else
{
    min_fraud_threahold_seconds = fraud_threahold_seconds;
    reasonKey = 2; // call over the threadhold
    reasonDescription = "Call duration is over threadhold";
}

 private String CreateAdomdCommand(int startTimeKey, DateTime startTime, int customerKey, int callingPartyKey, int calledPartyKey)
 {
 int month = startTime.Month;
 int quarter = 1;
 switch (month)
 {
     case 4:
     case 5:
     case 6:
         {
             quarter = 2;
             break;
     }
case 7:
case 8:
case 9:
    {
        quarter = 3;
        break;
    }
case 10:
case 11:
case 12:
    {
        quarter = 4;
        break;
    }
}
String cmdStr = "Select [Called Party].[Dim Geography - Dim Party].[Dim Party].&[" + calledPartyKey + "].parent on Columns,"
            + "[Calling Party].[Dim Geography - Dim Party].[Dim Party].&[" + callingPartyKey + "].parent on Rows "
            + " From  (Select [Measures].[Call Duration] on Columns, "
            + "ancestor([Start Time].[Time Quarter - Time Month].[Time Quarter].&[" + quarter + "].&[" + month + "].&[" + startTimeKey
            + "From [Fraud Detection DW] Where [Dim Customer].[Dim Customer].[Dim Customer].&[" + customerKey + "]) "
            + "Where [Measures].[CallAverage]";
return cmdStr;
}
```

Figure 6.18. Determine the Timer thread hold in Analysis Service

If the *PhoneCallHungUp* event arrives before the timer, the analysis service simply calculates the call length, retrieves necessary information of the phone call (i.e. calling number, calling type, customer id, etc) and stores the results into the *FactCallDetailRecord* table of the *FraudDetectionDW* database. Figure 6.18 shows the dynamic MDX query code in TimerFraudDetection service and the equivalent MDX query in SQL Server Management Studio to calculate the time threshold is shown in Figure 6.19.



```
Select  [Called Party].[Dim Geography - Dim Party].[Dim Party].&[10].parent on columns,
        [Calling Party].[Dim Geography - Dim Party].[Dim Party].&[1].parent on rows
From    (Select [Measures].[Call Duration] on columns,
                ancestor([Start Time].[Time Quarter - Time Month].[Time Quarter].&[2].&
        From [Fraud Detection DW]
        Where [Dim Customer].[Dim Customer].[Dim Customer].&[1000000])
Where [Measures].[CallAverage]
```

```
     4
1    3.67
```

Called Party.Country Name ▾	Called Party.Region Name ▾	Start Time.Time Quarter ▾
China	Asia	1

	Country Name ▾	Region Name ▾				
	⊟ Austria					
	Europe			Total		
Customer Name ▾	Call Duration	Fact Call Detail Record Count	CallAverage	Call Duration	Fact Call Detail Record Count	CallAv
Customer 1	11	3	3.67	11	3	3.67
Grand Total	11	3	3.67	11	3	3.67

Figure 6.19. MDX query and on FraudDetectionDW cube

6.5. Summary

In this chapter we presented a real-time Business Intelligence architecture called SARESA with the aim of providing continuous, real-time analytics in order to enable proactive responses to a business environment for effectively managing and controlling time-sensitive business processes. We introduced Sense & Respond loops and a service-oriented architecture that is able to detect situations and exceptions, perform complex analytical tasks and reflect on the gap between current situations and desired management goals. Traditional Business Intelligence architectures lack in the support of real-time BI and closed-loop decision making. A major goal of SARESA is to tightly integrate Business Intelligence with business processes by monitoring various IT system or other observables and generating reactive and proactive responses such as generating early warnings, preventing damage, loss or extensive cost, exploiting time-critical business opportunities, or adapting business systems with minimal latency. We presented our mobile phone fraud detection prototype implementation on Visual Studio.Net 2005 and MS SQL Server 2005. The work presented in this chapter is part of a larger, long-term research effort aiming to develop a service-oriented Business Intelligence platform for supporting time-sensitive business processes. Future research and development efforts will focus on enhancing the SARESA system with advanced

)

analysis and decision-making capabilities such as data mining on event streams and tightly integrating OLAP with rule engines.

PART III.

Non-Standard Data Warehousing – Future Trends and Research Challenges

Chapter 7. Future Trends in DWH and Research Challenges[1]

"Computing is like Hubble's Universe; everything is getting farther away from everything else"
Pat Helland (circa 1992)

Research in Data Warehousing and OLAP has generated successful and remarkable results and reached maturity in some aspects. However, new applications still continuously generate new challenges to the research community. Therefore, much research efforts remain to be done across many different areas of Data Warehousing. Beside the trend towards the active and real time Data Warehousing for continuous stream processing previously discussed in this thesis, there are some other emerging trends such as Spatial Data Warehousing, Web-enabled Data Warehousing, Process Warehousing, Data Warehouse and the Grid. This chapter discusses some challenges and new requirements needed to be address in these emerging non-standard Data Warehouse applications. The important techniques used in Zero-Latency Data Warehousing which are applicable to these non-standard Data Warehousing applications will also be briefly mentioned.

7.1. Introduction

Advances of modern technologies in variant application domains results in the exponential growing amount of information to be electronically managed in data warehouse system and included in the decision making process. Consequently, the Data Warehouses continue to increase in size while the data to be considered becomes more and more complex in both structure and semantics. Current emerging real world applications such as real-time Data Warehousing, analysis of spatial and spatio-temporal data, click stream analysis, OLAP mining, mobile OLAP and more recent applications in natural sciences (especially bioinformatics) requires novel representation and

---

[1] Special thanks to Prof. Tjoa to broaden my scope in Data Warehousing and OLAP technologies to the issues of this chapter.

manipulation techniques for non-standard data and tailored efficient algorithms for the computation of dedicated aggregate queries and application-specific index structures. The following are just a few of the fields and challenges, which drive current and future research in Data Warehousing.

Spatial Data Warehousing. Spatial data and spatio-temporal data are inherently more complex than traditional (alphanumerical) business data. Spatial data refers to geometric objects like points, lines, regions, surfaces, and volumes in the two-dimensional and/or three-dimensional space. Further, operations on geometric objects are much more complex than those on alphanumerical values. Spatio-temporal data add the temporal aspect. Spatial objects can change their position, shape, and extent over time and are then called moving objects. Geographical information systems (GIS) and Data Warehousing share the common interest to analyze data and to facilitate decision-making processes. But they use different technologies, and the capabilities of GIS for decision support are limited. Estimates state that 80% of all data have a spatial context or reference [Gonzales 2000]. This makes the combination of both technologies an interesting and promising approach. Especially the support of special spatial and spatio-temporal aggregation operators is here of interest.

Web Data Warehousing, Web data warehouses store massive amounts of Web data including URLs, structured and unstructured text, and multimedia content. Similar to spatial data warehouses, new data types and query operators have to be supported, for example, for searching URLs, or for traversing graphs. In the case of Web data warehouses, there have been repeated efforts to develop new data models for the Web that retain its topological structure and support efficient querying that goes beyond the keyword searches provided by today's search engines. However, much less is known about how to integrate the Web data represented by these models with more conventional corporate data to enable OLAP-style decision-support. Web data warehouses also face a massive scalability challenge. Simply treating the Web as a collection of operational systems and data stores and applying traditional ETL and batch update techniques will not be feasible.

Data streams, real-time and active data warehouses. Since data warehouses integrate data from different operational systems, they provide unique functionality for discovering information across these otherwise isolated data sources. This makes them ideal candidates for monitoring enterprise- or organization-level events that cannot be detected by a single operational system. Unfortunately there is a considerable delay until information from operational systems is reflected in the data warehouse: updates are typically performed in large batches once a week or over-night. A large retailer could better optimize the supply at geographically distributed stores and warehouses, if it had more timely information in the data warehouse. Other examples include Data Warehousing solutions for *data stream* applications like stock trading, military logistics, traffic monitoring, and sensor networks in general. All these applications require real-time or close to real-time update propagation, and hence also *real-time ETL*.

Data Warehousing for the sciences. Data warehouse technology seems like a natural fit for integrating scientific results obtained by different teams of researchers. For instance, a medical data warehouse could manage clinical data like medical records and diagnoses from different hospitals. There exist several databases with genome and life science data, which could be integrated in a data warehouse. In practice this is a challenging problem because multiple (independent) organizations are involved. It is difficult to force them to agree on data formats and it is not clear who would run a centralized data warehouse. In addition, there are limits to how much data each of the collaborating organizations would be willing to share. Hence Data Warehousing for the sciences requires novel approaches to protecting privacy. It also calls for a novel Data Warehousing architecture, which departs from the traditional centralized model towards distributed and Peer-to-Peer (P2P) style interaction.

Research challenges of new emerging applications are driven by the integration of novel data into the data warehouse. An essential, characteristic feature of these novel data is that they are *not* simply numerical, textual, or symbolic but *complex* both in their structure and in their semantics. We therefore coin the term *non-standard Data Warehousing* (in contrast to simple, standard, or traditional Data Warehousing).

In the Sections 7.2, we briefly review some promising non-standard Data Warehousing applications which handle variant kinds of complex data with different properties. We will focus on *Spatial Data Warehousing* (Section 7.2.1), *Web Data Warehousing* (Section 7.2.2), *Process Data Warehousing* (Section 7.2.3) and *Data Warehouse and the Grid* (Section 7.2.4). For each new kind of Data Warehousing, we first introduce its motivation, objectives and requirements. Next, the definition and research activities on the topic will be reviewed. Then, we specify the main research challenges and potential solutions. The essential techniques used in Zero-Latency Data Warehousing which are applicable in these non-standard Data Warehousing applications will also be investigated (Section 7.3). Finally, Section 7.4 concludes the chapter.

7.2. Non-standard Data Warehousing Applications

7.2.1. Spatial Data Warehousing

7.2.1.1. Motivation

Spatial data exists pervasively in business information systems. It is claimed that 80 percent of the overall information stored in computers is geo-spatial related, either explicitly or implicitly [Gonzales 2000]. Geo-spatial data are increasingly available from multiple providers (governmental agencies, private organizations) and accessible on the Web. With the advent of mobile computing and location-based services, popular use of satellite telemetry, remote sensing systems, GPS, medical imaging, and other computerized data collection tools, even more amount of spatial data is collected and stored in database. Often, organizations have collected geo-spatial data for their immediate internal needs (such as censuses, resource inventory, land management, and routing), accumulating huge amounts of data over time.

While storing spatial data is the first step, it is more important to analyze spatial data in business intelligent systems to provide insight into business from the geo-spatial point of view [Rao et al. 2003]. With the rapid growth of *geographical information systems* (*GIS*) usage and the increasing importance of spatial data analysis, we can observe that location and spatial data is becoming a core part of business databases and decision-making.

For the application area of environmental data, already several big data warehouses exist. The United Nations Environment Program offers data sets covering topics like freshwater, population, forests, emissions, climate, health or GDP. The NASA collects several Terabyte of environmental data a day. Similar data are collected for several data warehouses; an integration can increase the level of detail. Reports are produced periodically and for given areas; individual user-set queries would find many applications.

The location dimensions have been widely integrated in Data Warehouse and OLAP systems. However, these dimensions are usually represented in an alphanumeric, non-cartographic manner, since these systems are neither able to store nor to manipulate spatial data. The management of this kind of data is usually carried out by *spatial databases* [Güting 1994] or *Geographic Information Systems* (GIS) [Elzbieta et al. 2004]. Currently, although spatial databases and GIS are able to manipulate spatial data, the cost and response time of GIS queries is still a limiting factor. Furthermore, databases and GIS are not able to support the decision-making process on the basis of spatial data and require the integration or aggregation of spatial data from multiple sources.

7.2.1.2. Spatial/GIS Data Warehouse

Spatial data warehouses [Bédard et al. 2001, Bédard et al. 2002, Elzbieta et al. 2004] combine data warehouses and spatial databases for managing significant amounts of historical data that include spatial location. Merging these two technologies allows us to exploit the capabilities of both systems for improving data analysis, visualization, and manipulation of huge amount of spatial data. While data warehouses offer efficient access and analysis methods and enable the management of high volumes of analysis-supported, historical data, spatial databases have already a relatively long history in managing spatial data, and there is extensive research referring to spatial index structures, storage management, and dynamic query formulation [Elzbieta et al. 2004].

So far, research on spatial data warehouses has tackled several aspects and has gained some considerable results. The work in [Stefanovic et al. 2000] defines a spatial data warehouse as a conventional data warehouse that contains both spatial data and non-spatial data. They distinguish three types of spatial dimensions based on the spatial references of the hierarchy members: (1) the *non-spatial dimension* that contains only non-spatial data, (2) the *spatial-to-nonspatial dimension* where the primitive level is spatial but generalizations at some higher level are of non-spatial character, and (3) the *spatial-to-spatial* (*fully spatial*) *dimension* where the primitive level is spatial and also all its high-level generalized data are spatial. They also distinguish two types of measures: numerical measures containing only numerical values and *spatial measures* that contain one or a collection of pointers to spatial objects.

The experience gained in managing aggregated data in OLAP systems has already been extended to spatial data in *spatial OLAP* (*SOLAP*) systems [Rivest et al. 2003]. The concept is that extending a multidimensional model by the inclusion of spatial data provides a concise and organized spatial data warehouse representation. For this purpose, a *map cube* operator, which extends the concepts of data cube and aggregation to spatial data, is developed. Further, based on the classification used for non-spatial data, the authors present a classification and examples of different types of spatial measures, for example, spatial distributive, algebraic, and holistic functions. In [Miquel et al. 2004], the definition of spatial measures has been extended by the inclusion of measures represented as spatial objects or calculated using spatial metric or topological operators. A *spatial measure* is defined as "a measure, which is mapped to at least one spatial dimension whose members hold spatial representation". Thus, a spatial dimension must be included in a model if a spatial measure is required. The work in [Paradias et al. 2001] points out that the main differences between traditional OLAP and spatio-temporal OLAP is the lack of predefined hierarchies. Hence the positions and ranges of spatio-temporal query windows do not conform to pre-defined hierarchies and are not known in advance. They propose several indexing solutions and develop multi-tree indexes that combine spatial and temporal dimensions.

The integration of GIS and data warehouse/OLAP environments has been mentioned in [Pourabbas 2003]. This work focuses on spatial hierarchies and proposes a mapping between the hierarchical structures of data warehouses/OLAP and spatial databases. The concept of mapping between hierarchies is also exploited in [Kouba et al. 2002]. To ensure the consistent navigation in a hierarchy between OLAP systems and GIS, the authors propose a dynamic correspondence through classes, instances, and action levels defined in a meta-data repository.

7.2.1.3. Research Challenges

The following research challenges can be identified in Spatial/GIS Data warehouses:

- *Spatial data integration.* Spatial data is stored in different data formats that are structure-specific (for example, raster- versus vector-based spatial data, object-oriented versus relational models, different spatial storage and index structures) and vendor-specific (for example, ESRI, MapInfo, Intergraph, etc.). To integrate these heterogeneous, non-standard data types into data warehouses is a big issue. A lot of knowledge can be taken from data integration and data exchange for standard data, but the specific characteristics and requirements of spatial data have to be taken into account.

- *Consolidation and generalization of current data warehouse models for standard data.* A number of models for Data Warehousing and a number of operations for OLAP have been proposed and formally defined in the literature. These models are predominantly incompatible so that the first task is to accommodate them in a unified and formalized model. This model must be the basis of and thus extensible for new types of more complex data like spatial, spatio-temporal, multimedia, and image data and incorporate corresponding type-specific aggregation operations (for example, union, intersection in the spatial case). It is, in particular, important that there is a clear and strict separation between the conceptual user level and the implementation level. The user should not be bothered with implementation details. At the user level, we see the data cube with corresponding operations as the central metaphor that can be understood by the user.

- *Extension of the generalized data warehouse model to spatial and spatio-temporal data.* The extension mainly refers to attribute data types that are allowed to be spatial or spatio-temporal data types. The result is a spatial or spatio-temporal data cube. OLAP operations have to be tailored to these new kinds of data; new operations of interest have to be identified.

- *Fast and flexible OLAP support for spatial/GIS data warehouses.* In practice, querying the GIS spatial data can be an expensive and time-consuming task. There may be a large number of maps (regions) stored in a GIS; furthermore each such region may be made up of vast amounts of sub-regions (polygons). Each polygon on such a map will have associated with it a value for each function being measured (for example, temperature) at a given point in time. If a user, for example, wants to compute the average temperature of the entire region depicted in the temperature map at a given point in time, then the computation of the result would involve retrieving the value for each region and then calculating the average. This can be "serious" computation since vast amounts of data must be processed. Spatial database research has investigated some spatial accessing and indexing methods for efficient storage and access of spatial data. However, spatial OLAP operations require summarization and characterization on a large set of spatial objects in different dimensions and at different levels of abstraction.
- *Object-based, selective (partial) materialization.* Similar to materialized views in traditional data warehouses, to efficiently support analytical queries, it is necessary to materialize (that is, pre-compute) aggregated spatial data. However, a full materialization requires too much storage space, while the online-merge approach is too slow and requires expensive computational cost. A better approach may be obtained by an object-based, selective (partial) materialization of the spatial data.
- *Multiple presentations in spatial/GIS data warehouses.* Multiple representations [Bédard et al. 2002] result from seeing the world from different abstraction levels as well as different points of view. In fact, each user of a spatial database has her own needs regarding the representation of the objects. These representations may vary depending on several factors such as the user's needs or the level of details desired. Storing these multiple representations in order to better fulfill the needs of spatial database users has been a challenge since many years. Spatial/GIS data warehouses have the same requirement because different users have different abstraction levels and requirements in analyzing the spatial OLAP data.

7.2.2. Web-enabled Data Warehousing

7.2.2.1. Motivation

The rapid expansion of the Internet has made the WWW a popular place for disseminating and collecting information. The Web has become a primary conveyor of information to anyone, anywhere, anytime. Internet thus matures as a medium for doing business, and companies are trying to move beyond the basics of business transactions into a deeper level of understanding of what is occurring at their web site. Primary to this is the understanding of how their customers are interacting with the site, which includes not only navigation patterns, but also other customer information such as demographic data, buying habits, cancelling the transaction or changing interest to other competitors. A key component here is not only seeing the navigation patterns of the customer, but combining that with other information about that customer, including demographic and purchase information. It is this deeper level of understanding of the customer and how the customer is behaving in the store that allows the retail manager to make more informed business decisions.

Besides the profitable reason, monitoring how users behave on the system enriches the knowledge about their needs and allows the system to better adapt based on their previous behaviours [Yan et al. 1996]. There are other benefits associated with monitoring the use of the Websites. On one hand, it supports the evaluation of the system against its initial specifications and goals for better features and utilities. On the other hand, it enables the development of personalization strategies [Andersen et al. 2000b, Cooley 2003, Eirinaki 2003], helps increasing the system's performance [Eirinaki 2003, Joshi et al. 1999], supports marketing decisions [Chen et al. 1996], helps on detecting business opportunities that otherwise could remain unnoticed [Kohavi 2001] and may contribute to increase the system's security [Cooley 2003, Sweiger et al. 2002].

The user interactive with the websites are traced via Web server log file so called *Clickstream*. Clickstream logs are numerous, large and have integration challenges specific to the Web environment. However, every aspect of the e-business environment is changing so rapidly and frequently that it is far more difficult to relate cause and effect. Consolidating logs from hundreds of separate servers in an effort to transform, integrate, analyze and aggregate them required significant innovation given the rapidly growing volume of data. The data warehouse must play an integral role in the Web revolution as the analysis platform for all the wonderful behaviour data coming in from the clickstream, as well as for the many Web sites that rely on the data warehouse to customize and drive the end user's Web experience in real time. This analysis process must be accomplished in the near real-time by an environment that has now become to be known as the *Data Webhouse*

7.2.2.2. Clickstream and Data Webhouse

Web Servers can record every click that a user makes when browsing the web pages. Gigabytes of data showing the visitor's browsing pattern in the Web site, are collected every day from the Web servers. This results in extremely large amounts of data, in the form of web logs. A web log is a text file that records every request that a client makes to the web server and is usually used for web site management and monitoring purpose. However, "*Clickstream is a wonderful source of data because it's the behaviour of visitors to your Web site—it's related to revenue in a very direct way*" [Kimball 1999]. By analyzing these clicks, we can begin to answer many questions in regards to how customers are interacting with the web site. To identify the right customers and their interests, sophisticated methods are needed to store and analyze the data collected over a period of time.

The storage of clickstreams and other contextual data in a data warehouse, to understand user behaviour, is what Kimball and Merz [Kimball,Merz 2000] called a data webhouse. According to these authors, a data webhouse is crucial to any activity that uses the Web to deliver its services, allowing the improvement of the organization's systems. In another article, Kimball also defined "A data webhouse is a distributed data warehouse with no central data repository that is implemented over the Web to harness clickstream data" [Kimball 1999]

The data webhouse could be considered as a web instantiation of the data warehouse. It stores the clickstream data and performs all the traditional data warehouse functions to facilitate the analysis of the data. The Webhouse plays a central and a crucial role in the operations of the web enabled business by providing the following features:

- Houses and publishes click stream data and other behavioral data from the web that drive an understanding of customer behavior.
- Act as a foundation for web enabled decision making. The data webhouse must allow its users to make decisions about the web, as well as make decisions using the web.
- Act as a medium that publishes data to the customers, business partners, and employees appropriately, but at the same time protects the enterprise's data against unattended use.

A clickstream Webhouse may be the single most important tool for identifying, prioritizing, and retaining e-commerce customers. The Webhouse can produce the following useful information:

- Site statistics: Number of hits/page and total number of site hits?
- Visitor conversions: How many visitors have become registered customers with affiliate relationships and referring domain analysis?

- Ad metrics: How many clicks result in orders and how does the size, color, and location of ads effect sales?
- Referring partner links: What are top referring partner links, and which partner links lead to orders?
- Site navigation resulting in orders : What are the most common navigation paths taken through the site resulting in orders?
- Site navigation not resulting in orders
- Pages that are session killers : Which pages are session killers?
- Relationships between customer profiles and page activities
- Best customer and worst customer analysis

By building the Webhouse to analyze the user behaviour patterns contained in these clickstream, savvy businesses can expand their markets, improve customer relationships, reduce costs, streamline operations, strengthen their Web sites, and hone their business strategies.

Similar to traditional data warehouse, effort will involve the extraction, transformation, and loading of the clickstream data to the Webhouse repository; building dimensional schemas from the clickstream data and deploy information delivery system from the Webhouse.

7.2.2.3. Challenges

The following challenges should be considered in building and deploying the Webhouse:

- *Detecting and capturing data change.* The Web offers access to large amounts of heterogeneous information and allows this information to change at any time and in any way. These rapid and often unpredictable changes to the information create a new problem of detecting and representing changes. This is a challenging problem because the information sources in the Web are autonomous and typical database approaches to detect changes based on triggering mechanisms are not usable. Moreover, these sources typically do not keep track of historical information in a format that is accessible to the outside user [Andersen et al. 2000a].
- *Heterogeneous, unstructured, poor semantic data sources.* The World Wide Web is rarely seen as a potential data source. One reason for sure is that data in the WWW are rather semi structured or not structured at all: information is stored in HMTL format without any semantic meta data, or – in very few cases – in XML files based

on different or no grammars (DTDs or XML Schemas). Other information is encoded in pictures or arbitrary binary file formats. Besides these problems of technical heterogeneity the semantic heterogeneity is even worse: different WWW documents might refer to different meanings when using the same expression year, e.g. financial year or calendar year. As long as the semantics of the online documents is not understood it is extremely dangerous to integrate it into a data warehouse with regard to data quality ([Kimball 2000]).

- *Data Volumes.* The big move to mass customization means we now capture, analyze, and respond to every transaction in the business including every gesture a customer makes, both before and after operational or sales transactions and there seems to be no volume limit. For instance, the combined Microsoft-related Web sites, analyzed daily as a single entity, on some busy days have captured more than a billion page events. The webhouse's capacity thus is much bigger than the capacity of traditional data warehouse. It is obviously that we have to deal with many potential challenged administrative and operative tasks in such a tremendous webhouse.
- *Quick response time.* The Web makes fast response times critical. If something useful doesn't happen within 10 seconds the user may navigate to another page. Those of us who run big data warehouses must experience that many queries will take more than 10 seconds. Therefore, respond to an end-user request in roughly 10 seconds, regardless of the complexity of the request is sometimes impossible but that is a challenged Webhouse's expectation. The web cache approach and parallelism mechanism were considered to speed up the response time.
- *Security Issue.* Besides the classical security issues have to be solved as in traditional Data Warehousing, in literally, when deploying the Web-warehouse, you are putting your corporate data, one of your most valuable assets, on the line. Therefore, security must be a top priority for any Web-warehouse initiative. However, implementing security in Intranet (as in case of traditional Data Warehousing) is not as complicated as tackling Internet security threat where every kind of malicious people are looking for a way to hack into the data warehouse. With the advancements in hacking techniques and ever increasing number of industrial spy, the Internet security threat in Web warehouse becoming more challenged issue.

7.2.3. Process Data Warehousing

7.2.3.1. Motivation

Due to the increasing competition and pace in today's business, continuously monitoring, analyzing and assessing business processes becomes a more and more important issue. However, in comparison to process innovation and improvement efforts, in depth process analysis has not got the attention it would need. For the execution and monitoring of business processes, many organizations are increasingly using *BusinessProcess Management Systems* (*BPMSs*) and *Workflow Management Systems* (*WFMSs*) [Koksal et al. 1998, Muth et al. 1999] to improve the efficiency of their processes and reduce costs. During the execution of the business process, WFMS record many types of events, such as the start and completion time of each activity, the assigned resources, and the outcome of the execution. Major BPMSs and WFMSs provide comprehensive support for the early stages of the business process lifecycle but often lack capabilities for providing feedback and transparency about the performance of business processes. Although WFMSs often log detailed information during the process execution, they have difficulties in accumulating and condensing audit trails of business processes and using this information for monitoring and analysis purposes [Schiefer et al. 2004b].

Until now, most data warehouses have not been used for capturing state changes of business processes (process control data). As efficiency, accuracy, transparency, and flexibility of enterprise's business processes have become a major competitive advantage for organizations, paying attention to monitoring and controlling of workflow execution at a formal and strategic level will become a focus of future information management.

The *process warehouse* (*PWH*) [Kueng et al. 2001, List et al. 2000] is a separate read-only analytical database that is used as the foundation of a process oriented decision support system. The main aim of a PWH is to analyze and improve business processes continuously. PWH enables multidimensional process analysis for process owners, process managers or other authorized staff members to very quickly obtain comprehensive information on business processes, at different aggregation levels, from different and multidimensional points of view, over a long period of time, using a huge historic data basis prepared for analyzing purposes to effectively support the management of business processes.

7.2.3.2. Process Data Warehouse (PWH)

The PWH as part of an enterprise's data warehouse provides the data foundation for all strategic process-oriented DSS processing. In many cases, the data warehouse is the first place where integration of business data is achieved and much historical processing is done. As a component of an enterprise's data warehouse, its architectural structure is equivalent to traditional data warehouses. For that reason, it supports data that is subject-oriented, integrated, time-variant, and non-volatile.

The *subject orientation* of the PWH implies that it is organized along the lines of the major entities of a process. These entities define the context for the business process analysis. The business process context defines various perspectives of the business process and can be divided into 5 major categories: (a) process definition context (for example, process model, process attributes, process targets), (b) resource context (for example, organizational context, demographic information about workflow participants, resource history), (c) business object context (for example, business object attributes, measures, process inputs/outputs), and (d) runtime environment context (for example, information about source systems, workflow environment, applications).

Integration of PWH data refers to the physical unification and cohesiveness of the data as it is stored in the warehouse. Process data integration in the PWH is not achieved by merely copying data from the WFMSs or from the operational environment. Instead, as raw data passes through the ETL layer, a fundamental alteration is done to the data to achieve an integrated foundation that resides in the PWH. Major challenges for integrating data from WFMSs into the PWH are the transformation of workflow audit trail data into valuable business metrics and the representation of the process model as a dimensional table.

Another characteristic of the PWH is that of *time variancy*. Because a PWH is made up of a massive series of workflow events, it can contain data over a lengthy period of time. It is common for a PWH (and also for data warehouses) to hold detailed data (active or archival) that is up to five years old.

Non-volatility refers to the fact that an update (in its purest sense - that of finding a record and making changes to the record) does not normally occur in a PWH. If an update occurs at all, it occurs on an exception basis as with traditional data warehouse systems.

7.2.2.3. Research Challenges

Process Data Warehousing extends traditional Data Warehousing with a process perspective. By including process metadata and process runtime data in the ETL process, a data warehouse can be enriched with typical process performance indicators (such as cycle times, costs, and rework) and a new dimension for representing the process model. A process-driven analysis [Boehnlein et al. 2000] of business performance indicators facilitates analysts to identify root causes in context of a business process. Process Data Warehousing also allows a closer integration of the operative source systems with a data warehouse environment.

The main research issues of Process Data Warehousing include the following:

- *Applying process concepts to data warehouse design.* The main difference between the PWH and traditional data warehouses is the inclusion of information about the process models within a data warehouse environment in order to drive the DSS. Traditional data warehouses lack in providing facts that measure the performance of certain parts of a business process, such as metrics about activities, sub-processes or the entire process. For the computation of these facts often multiple workflow events have to be processed. The PWH captures a history of these facts including their context information. The data model that supports the process analysis therefore must include various dimensions that form the context of activity instances. The dimensions are used to show the activity facts at various levels of granularity. Analysts can use these dimensions for drill-downs and roll-ups. The data model on the activity-level has, of course, some common dimensions with the process-level data model. These common dimensions can be used for drill-across operations. For instance, if an analyst is investigating the duration of a process instance (process-level), they can drill-across to all activity instances that belong to that process instance.
- *Methods for ETL processing of workflow audit trails.* As mentioned before, many organizations use BPMS and WFMS for modeling business processes and defining the target KPIs (key performance indicators). During execution, these KPIs have to be continuously generated as soon as sufficient workflow events are available. Delays in propagating workflow events from WFMSs to the PWH can significantly decrease the value of the event data for the users. It is necessary to find out the ETL approach that allows workflow audit trail information to be available in the PWH in the near real-time. This allows business users and process analysts to have the accurate and detailed information about current business situations to identify weaknesses and bottlenecks in the process handling earlier. Besides, the calculation of process metrics often requires additional information from other data sources. For instance, in the case of an order process, the workflow events do not include detailed information about the orders and customers. Nevertheless, order and customer information might be needed for the calculation of workflow metrics about the order transactions. The ETL process thus must be able to merge the workflow audit trail with additional information from other data sources.

- *Analysis of workflow audit trails.* PWH must provide comprehensive information on business processes, at different aggregation levels (process abstraction levels), from different and multidimensional points of view, using a historic data prepared for analyzing purposes to effectively support the management of business processes. It should also include detailed information about the execution paths and bottlenecks of business processes and thereby enable a fast detection of weak spots in the process and organizational structures.
- *Improve business process performance with operative decision support.* The monitoring or analysis of workflow audit trails often entails a direct or indirect feedback into the WFMS or the operational systems. This response is usually referred to as closed loop analysis and can be done manually or automatically. It does not necessarily imply an operational decision but informs decision makers that something significant is happening which needs to be investigated in further detail. In some routine or semi-routine decision tasks, it could enhance the operational system with business intelligence such as process performance improvement, potential bottlenecks prevention and interruptions elimination.

7.2.4. Data Warehousing and the Grid

7.2.4.1. Motivation

Emerging advanced applications are continuing to generate ever larger amounts of valuable data, which could be obtained from business transactions, medical investigations, scientific simulations, telescopes, microscopes, satellites, etc. There are major challenges involved in the efficient and reliable storage, fast processing, and extracting descriptive and predictive knowledge from this great mass of data [Lowell 2003]. They are often geographically distributed and their complexity is increasing, meaning that the extraction of meaningful knowledge requires more and more computing resources. This combination of large dataset size, geographic distribution of users and resources, and computationally intensive analysis results in complex and stringent performance demands that, until recently, were not satisfied by existing computational and data management infrastructure [Darvas et al. 2004].

In tackling these problems, yet we deal with the advent of a new form of information technology—the *Grid* [Foster et al. 2001, Fellenstein et al. 2003, Boden 2004, Oracle 2004] — a software infrastructure that enables flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources. The research and development in Grid has been almost exclusively driven by "big science" like distributed high-performance simulations, high-energy physics, material science, and biological applications, etc. However, the focus is recently going to shift to more general domains, closer to every day life, like medical, business, and engineering applications. Grid Computing provides highly scalable, secure, and extremely high performance mechanisms for discovering and negotiating access to remote computing resources in a seamless manner.

7.2.4.2. Grid Technology and Data Warehousing

In some emerging data warehouse applications such as warehouses for sensor data [Bonnet et al. 2001], biological warehouses [Darvas et al. 2004], Hastings et al. 2005] etc., it is necessary to handle non-traditional sources and data types. The data types include a broad collection of high throughput genomic and proteomic data, outcome data, and digitized imaging and anatomic pathology studies [Hastings et al. 2005]. The data volume produced in these large-scale applications is too huge to fit in a single main-memory, and the extreme high data flow rate requires very powerful performance of data processing operations. Furthermore, it is likely that diverse data types will be captured and stored on distributed heterogeneous platforms. This confronts even experienced data managers with a host of potential problems, including matching, transformation, and integration of various disparate data sources.

In general, a data warehouse comprises two processes: data loading (input) and data analysis (output). As the volume of data in the warehouse continues to grow, the time it takes to mine the data does too. Both individual queries as well as loading data processes can consume enormous amounts of processing power and time, impeding other data warehouse activities and causing sluggish response times [Butte 2004]. Solutions for accelerate processing such as upgrading hardware, parallel model, etc are not optimal due to IT budget restrictions. Therefore, a nearly ideal solution would help to reduce load process time and optimize available resources for the analysis process. The Grid technology could be applicable in both aspects to gain compute resources without purchasing additional hardware. The following reasons for using Grid technology to tackle some of the existing dilemmas in data warehouses have been explored in [Butte 2004]:

Parallelization of sub-processes. Grid technology allows us to dispatch jobs in parallel to multiple compute nodes and execute several independent jobs at once. This parallelization of previously serial load tasks to multiple CPUs is a powerful Grid's advantage. The ability to execute transformation tasks independently enables the load process to be broken into multiple sub-processes, which can be sent to different nodes in the virtual pool. Furthermore, each sub-process could be sent to multiple nodes to guard against downtime and failures. These sub-processes compete to each other to return the results first, thus will eliminate the single point of failure while guaranteeing the fastest response time.

Information grid technology. Traditionally, a data warehouse is created as a (centralized) storage-oriented database. The data must be extracted, loaded and integrated into the centralized warehouse storage no matter where it is located. Even conventional distributed data warehouses are restricted to a relative small number of nodes where data is concentrated. A federated solution, in contrast, can maintain the data at its points of origin rather than bringing data to the data warehouse. Federated solutions help to address the size and complexity of data warehouses by applying a logical model to the existing physical infrastructure instead of imposing a new data warehouse environment. Information grid technology gives users and applications security-rich access to virtually any information source, anywhere, over any type of network supports sharing of data for processing and large-scale collaboration. It also helps to realize the federated model to distributed and complex data sources. Building on the concept of virtualizing computing power into resource pools, information grids treat data silos as information pools with nearby storage pools. Data required by the warehouse can be accessed via the information grid where domain security management is handled automatically.

Data transformations can be performed at the remote client and the results brought back across the network, encrypted for privacy. Retrieved data is cached to improve performance with mechanisms for expiration to verify that only the most recent data is available.

7.2.4.3. Research Issues

Several problems and research questions arise from the utilization of the computational Grid technology for Data Warehousing applications. The main research issues in building the *Grid-based Data Warehousing* (*GWH*) environment include:

- *Dynamic resource location and service invocation.* Most Grid-based applications are built upon a set of (grid) services located in different grid nodes. Theoretically, to invoke these services, it just needs to find out a node which has enough resources to execute the service. However, the service invocation process is actually more complicated due to the dynamic Grid runtime environment. The computational and networking capabilities, the availability of the Grid nodes can vary from time to time. There are several policies in resource location and dispatch such as best-fit model, speed priority, cost priority, quality of service priority etc.

- *Acceptable response timeliness.* As mentioned above, the continuous growing volume of data in the data warehouse causes sluggish response times. Obviously, with a GWH application, the acceptable response timeliness is mandatory. Prototype implementations still need significant improvements with respect to response time criteria. Innovations in computer network, data communication, data transportation protocols, etc., and further research efforts on Grid technology will gradually shrink the response timeliness-gap. The computational Grid can federate systems into supercomputers far beyond the power of any current computing center and these centers will become super-data and super-application centers [Bell 2004].
- *Grid transparency of accessing, communication protocols and standards.* Similar to other grid-based applications, the GWH should be able to provide the users with an interface which is easy and transparent to the Grid, network, and location. The users can access the system to register the services, execute the ETL process, query the OLAP cube [Niemi 2003], or send their feedback from anywhere using a variety of devices (mobile, PDA, laptop, etc.) without considering the complexity of system architectures. The user can access the data without considering about data source format (heterogeneity transparency), where the data is located (location transparency), local data or distributed data (distributed transparency), whether original or replicated (replication transparency). The underlying communication protocols and standards between the services are also hidden.
- *Integrity and availability of the grid nodes.* Assuming that all grid nodes are trusted entities, there still is a need to implement integrity check. If nodes cannot be fully trusted, appropriate data integrity checks are required. The work in [Kamvar et al. 2003] provides an excellent overview of requirements and also shows how these issues are addressed in P2P networks. The solutions can also be adapted to Grid computing if one is prepared to accept the necessary computational overhead.
- *The security and trust of grid resources.* The most significant challenge for Grid computing is to develop a comprehensive set of mechanisms and policies for securing the Grid. One of the critical differences between Grid security and host or site security is the absoluteness of site autonomy. While the site administrator has complete control on the security mechanisms and security policies, the grid resource providers must follow the security mechanisms and policies that are not strictly under their control [Humphrey et al. 2005]. With GWH applications, much more research efforts to enable additional security related to data access, user authentication and authorization, user privileges etc. within the Grid environment are needed.

7.3. Zero-Latency Warehousing techniques as Enabler for Advanced Non-standard Data Warehousing applications.

Many of the advanced facets of non-standard Data Warehousing applications discussed in Section 7.2 to 7.5 require fast response to the user requests. For example, the Spatial Data Warehousing and GIS systems are mostly used in the interactive mode, thus require highly ***acceptable response** times* when the user analyse and view the GIS information. *Fast response* is also a mandatory criterion of the Clickstream Warehouse application where the users require that their interested web pages should be available in the acceptable idle time. The acceptable response is also an important requirement in Grid-based Data Warehousing application where the reasonable response time is still a big challenge (Section 7.5). In these applications, the Zero-Latency Data Warehousing techniques could be applicable to speed up the analysis request such as *real time OLAP cube*, *pre-built cache* or *parallelism* in OLAP analysis queries.

Data freshness is one of the most important criteria. Although it is not the mandatory criteria in Spatial Data Warehousing, GIS and Grid-based Warehousing, data freshness is the mandatory requirement in time-sensitive applications. The Tsunami Early Warning System is a drastic example for the need of such a system where all these requirements have to be met.. In these applications, it is required that the data change is monitored, collected, and integrated into the Data Warehouse in the way as near to zero-latency method as possible. The Data Warehouse is refreshed in near real time thus provide the fresh data for analysis. The techniques of Zero-Latency Warehousing such as *real-time ETL*, *continuous integration* and *real-time Warehousing* could be applicable in these applications.

Automated reaction is another important feature. In monitoring applications such as Process Warehousing, the situations and exception generated during the process monitoring and workflow analysis should be processed automatically without user intervention. Automation is also necessary in interactive mode application such as Clickstream Warehousing where the system could suggest some interesting websites or related information pages to the users when he/she is browsing the web sites. It could also be necessary for a long running process in Grid based Data Warehousing where the services could be invoked automatically in the dynamic resource changing environment. *Active decision engine* and *knowledge based rules* are the techniques in Zero-Latency Data Warehousing could be applied here for automation reaction control. The Earth Simulator developed by NEC Corp. is one extreme example focusing on automated and fast response for huge date warehouses which ultimately need massive parallel processing and storage. [Watanabe 2005]

Data integration is the natural characteristic of Data Warehousing applications. However, in the case of non-standard Data Warehousing applications, the data is much more complex. The GIS Warehouse contains Spatial data and data in multiple information layers. The Webwarehouse (or Clickstream Warehouse) contains hyper links, text, graphics, video, etc. Process Warehouse contains workflow information and process statues while the Grid-based Warehousing could contain data extracted from heterogeneous data sources from the Grid. The data complex on both semantic and structure requires the flexible and generic data integration method which could integrate multiple data in heterogeneous format into the integrated common format. We could use the concepts of *adapter* (Section 6.4.5.2) to flexible and dynamically deal with different *data souces* (6.4.5.4) in Zero-Latency Data Warehousing to solve that integration issue.

In general, as a further step the techniques used in Zero-Latency Data Warehousing applied on the non-standard Data Warehousing applications will enable to solve many of the future goals in these areas. Table 7.1 gives a first sketch of these potential enabling facilities.

Criteria	Spatial DWH	Web WH	Process WH	Grid- based WH	Proposed Techniques
Acceptable Response	X	X			Real-time OLAP, pre-built cache, parallelism in OLAP
Data Freshness		X	X		Real-time ETL, continuous integration, real-time DWH
Automated Reaction		X	X	X	Automated decision engine, Knowledge-based rules
Data Integration	X	X	X	X	Data adapter, Data Sources

Table 7.1. Zero-Latency Data Warehousing techniques applied in non-standard Data Warehousing applications

7.4. Summary

In this chapter, we have presented a brief overview of some emerging trends in non-standard Data Warehousing and introduced possible challenges proposed to the Data Warehousing and OLAP research community. It is possible to recognize that the basis themes of Data Warehousing research has not changed, but the sources, applications, and enabling technologies have become more complex and diverse. Besides the well-known, traditional and mature solutions, developing this new stage of Data

Warehousing of non-standard application requires some further research efforts, dependent on the applications' specific requirement and its data sources characteristics. In our opinion the aspects of Zero Latency could boost Data Warehouse Research due to its novel potential for non-standard applications to a new quality. Nevertheless much research efforts are still necessary to detect and solve the specific requirements of non-standard applications which can be accomplished by our Zero Latency characteristics in DWH.

Chapter 8. Conclusion and Future Work

"Imagination is more important than knowledge"
Albert Einstein.
Over the next couple of decades IT professionals will help
workers integrate computing and communications onto and
into their bodies and brains, with wearables and implants.
James Hughes, 2004

8.1. Conclusion

In this thesis, we have introduced the concept of a Zero Latency Data Warehousing which aims to minimize the reaction time to any happening event. We have introduced the state-of-the-art of the Zero Latency Data Warehousing and have explored and investigated some of its aspects in this thesis. The core components of the Zero-Latency Data Warehouse address important requirements to achieve near zero-latency between the cause and effect of a business decision.

Traditional Data Warehousing "only" contributes to an efficient information supply between transactional applications and decision support applications, but information 'backflows' take their way indirectly (i.e. by actions of decision makers) into the transactional systems and not directly through the data warehouse (open-loop approach).

To archive the zero response latency and gain the high business value, it is required to minimize the reaction time in a closed loop system in several steps:

- *Minimize Data latency*: This could be archived by providing the sense component to timely capture and store the data when it occurs.
- *Minimize Analysis latency*: This requires the real-time analysis component which is capable of integrating both real time data and historical data to conduct the analysis requests.
- *Minimize Decision latency*: The Decision Time could be reduced by employing the rule-based decision engine to automate some routine decision tasks.
- *Minimize Response latency*: This could be archived by providing the instant message exchange mechanism for real time communicate between the components.

8.2. Contribution

In the thesis, we present 3 different approaches, all are towards the purpose of reduce reaction time to happening event : Event-feeded Slowly Changing Dimension, Grid-based Zero-Latency Data Stream Warehouse (GZLDSWH), and Event Sense & Response loop Architecture (SARESA).

8.2.1. Event-feeded Slowly Changing Dimension

This approach introduces a practical way to validate the event message, to reconstruct the complete history of the dimension and to provide a well applicable "comprehensive slowly changing dimension" (cSCD) interface for well-performing queries on the historical and current state of the dimension of the Data Warehouse. Events are accumulated and processed in the near real time instead of using the batch processing with the series of consequent snapshots. This approach thus provides more fresh information as well as saves the storage and processing time compared with the snapshot based approach.

Events which contain the transaction operations on entities are accumulated and processed via an *Event Processing* module. This module refreshes the TRANS table using an enhanced SCD type which is the mix of SCD type 2 and SCD type 3. Before being processed, events are carefully checked to avoid invalide and inconsistent situations. The on-demand snapshot at any time point can be generated from the TRANS table using the *Snapshot Generation* module. Finally, occuring inconsistencies between the data in TRANS table and operational data source can be resolved by the *Consistency Checking* module which aims to recover the consistency status.

8.2.2. Grid-based Zero-Latency Data Stream Warehouse

The Grid Computing technology is used to store lossless continuous data stream without using the statistical approximation approach. Storing continuous data streams without information-loss requires a tremendous data storage and super power computing systems which does not exist yet. The Grid seems to suit well with this requirement and enables us to develop the Grid-based Zero-Latency Data Stream Warehouse (GZLDSWH).

The GZLDSWH system is built upon the OGSI-based grid service and the GT3 tool kit. The dynamic collaboration between the services is investigated in detail and a prototype for a centralized service invocation control called Dynamic Service Control Engine (DSCE) is developed. We have proposed some necessary extensions to use DSCE in GZLDSWH. Another achievement is the Grid-based OLAP engine management service. We also introduce other services which are necessary to build up the GZLDSWH and discuss its requirements. However, it should be stressed that this approach is still in an on-going phase of work.

8.2.3. Sense & Response Architecture

This approach enhances the traditional Data Warehousing and Business Intelligence by introducing a close loop system which is capable of sensing, interpreting, predicting, automating and responding to business environments and thereby aims to decrease the reaction time needed to make business decisions. We have introduced the Sense and Response loop which include several services. Each of them operates at a specific stage of the Sense and Response loop. The services are running and managed by the underlying infrastructure which provides smoothly event delivering, flexible event mapping, collaboration, fault tolerance, transaction support, scalability and performance. The analysis process is conduct in real time whenever the event is sensed and delivered through out the system. In general, the Sense and Response Architecture (SARESA) provides the following features:

- *Timely event capture and delivery*: Event streams could arrive from heterogeneous source system. SARESA must be able to capture rich context information of events as they occur and deliver them in a timely manner to the required event services. It also ensures that no event is lost or delivered more than once.
- *Service-oriented architecture*: SARESA has a distribute, scalable architecture which allows us to model and execute various forms of Sense & Respond processes. The data processing of SARESA is controlled by "Sense & Response loops" which have multiple processing stages that are supported by event services.
- *Real-time analytics support:* This is one of the key capabilities of SARESA in order to continuously analyze event patterns and business data. Business performance indicators (analysis metrics) are generated on the fly when relevant events are available and they are refreshed in near real-time to reflect the currently arriving event data (only refresh windows are not acceptable).
- *Automated decision support and instant response:* The timely reaction is the most significant feature of an event stream processing system. SARESA allows the services to automatically detect and analyze business situations and exceptions and respond to them. The response could be alarms, notifications, or any other kind of feedback sent to appropriate users without human intervention.
- *Transactional support in event stream processing*: Unlike existing stream processing systems that often shed the streams at some levels in order to handle large event

streams, SARESA assures that no events are lost during the processing. In SARESA, each event is processed by a transaction to avoid any data loss.

- *High scalability and performance:* Architecture scalability and high performance are important requirements for any event stream processing system, especially when the amount and frequency of events can significantly vary (e.g. caused by event bursts). Scalability and high performance in SARESA are archived by transparent distribution of the event processing to event services on multiple computer nodes. The SARESA system ensures *failure tolerance* and provides scalable, distributed system services for *event correlation* and *event synchronization*.

8.3. Future Work

Concerning the Event-feeded Slowly Changing Dimension solution, advanced features such as auto-correction of invalid events (i.e. insert an existing key will be corrected as an update event, update on non-existing key will be corrected as an insert event, etc), dealing with late arrival event, dealing with concurrent and redundant duplicate events will be considered in the next steps.

As mentioned previously, the Grid-based Zero-Latency Data Stream Warehouse system is still in the on-going-work phase. Further investigation and research efforts are required to build up the whole service. To build up the Grid-based OLAP engine, we should consider current vendor technologies and products instead of building everything from scratch. The timeliness criteria in Grid environment is currently the weak point of the system which should be considered. The trend towards integration between Grid and Web services (i.e. WS standard) will last as a main reseach focus.

SARESA prototype should be enhanced with the visualization tools to easily modeling the event processing map (which is currently implemented as a configuration XML file). Currently the decision engine use the static hard coded rule specified by the expert, which make the system inflexible and not "intelligent". Therefore, the data mining approaches to generate the appropriate decision rules are to be considered in our future work. Other enhanced features such as scalability and transaction recovery will be further investigated.

The emerging non-standard Data Warehousing trends discussed in Chapter 7 are the good starting point for further research on Zero Latency Data Warehousing and important future applications of OLAP.

Last but not least, vendors like Oracle, IBM and Microsoft already tightly integrate OLAP and data mining in their DBMS commercial tools. Therefore, we are observing a trend to close the gap between Data Warehousing and data mining. It is even imaginable that in the medium-term future the separation of OLAP-data warehouses (with its semantic redundant storage requirement) from its OLTP-database-sources could be bridged by innovative view-generation techniques as originally proposed in the three-level architecture.

Abbreviations

ACM	Association for Computer Machinery [www.acm.org]
ADB	Active Database
ADWH	Active Data Warehouse
AI	Artificial Intelligence
API	Application Provider Interface
BAM	Business Activity Monitoring
B2B	Business to Business
B2C	Business to Customer
BESS	Bit encoded sparse structure
BI	Business Intelligence
BPMS	Business Process Management System
BSCS	Business Support & Control System
CC	Consistency Checking
CDC	Change Data Capture
CDR	Call Detail Record
CMS	Cube Management Service
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CRM	Customer Relationship Management
cSCD	Comprehensive Slowly Changing Dimension
CWM	Common Warehouse Model [www.cwmforum.org]
DAS	Data Analysis Service
DAWAK	Int. Conf on Data Warehousing and Knowledge Discovery
DBA	Database Administrator
DBE	Dynamic Bit Encoding
DBMS	Database Management System
DCE	Distribute Computing Environment
DCS	Data Capturing Service
DDL	Data Definition Language
DES	Data Cleaning Service
DEXA	Int. Conf. on Database and Expert System Applications
DIS	Data Integration Service
DLL	Dynamic Link Library
DMS	Data Mediation Service
DML	Data Manipulation Language
DPS	Data Processing Service
DGSI	Dynamic Grid Service Invocation
DSCE	Dynamic Service Control Engine
DSCL	Dynamic Service Control Language
DSMS	Data Stream Management System
DSS	Decision Support System
DSS	Data Storing Service
DTC	Distributed Transaction Controller
DWH	Data Warehouse
EAI	Enterprise Application Integration
ECA	Event-Condition-Action

EER	Extended Entity-Relationship Model
EFD	Event-feeded Dimension
EIS	Enterprise Information System
EP	Event Processing
EPM	Event Processing Model
ER	Entity-Relationship Model
ERP	Enterprise Resource Planning
ETL	Extract-Transform-Load
FTP	File Transfer Protocol
GB	Gigabyte
GIS	Geographic Information System
GridMiner	GridMiner Project (www.gridminer.org)
GWH	Grid-based Data Warehouse
GGF	Global Grid Forum
GSI	Grid Security Infrastructure
GT3	Globus Toolkit version 3 (www.globus.com)
GZLDSWH	Grid-based Zero-Latency Data Stream Warehouse
GWSDL	Grid-based Webservice Description Language
HOLAP	Hybrid On-Line Analytical Processing
HTTP	Hyper Text Transfer Protocol
I/O	Input / Output
IDE	Integrated Development Environment
IEEE CS	IEEE Computer Society
IEEE	Institute of Electrical and Electronics Engineers [www.ieee.org]
IJDWM	International Journal of Data Warehousing and Mining
IRDS	Information Resource Directory System
ISO	International Standardization Organization [www.iso.org]
IT	Information Technology
J2EE	Java Platform 2 – Enterprise Edition [www.sun.com/java]
J2SE	Java Platform 2 – Standard Edition [www.sun.com/java]
JAR	Java Archive
JDBC	Java Database Connectivity
JMS	Java Message Service
KB	Kilobyte
LAN	Local Area Network
MB	Megabyte
MDDBMS	Multidimensional Database Management System
MDX	Multidimensional Expression
MIS	Data Mining Service
MIT	Massachusetts Institute of Technology
MOLAP	Multidimensional On-Line Analytical Processing
MPPs	Massively Parallel Processors
MSMQ	Microsoft Message Queuing [www.microsoft.com/msmq]
MS SQL Server	Microsoft SQL Server
NAS	Notification Service
ODS	Operational Data Store
OLAP	On-Line Analytical Processing
OLAM	On-Line Analytical Mining
OLTP	On-Line Transactional Processing
OGSI	Open Grid Service Infrastructure

OGSA	Open Grid Service Architecture
PWH	Process Warehouse
RBS	Resource Broker Service
RDBMS	Relational Database Management System
RDS	Rule Design Service
RES	Rule Evaluation Service
RFT	Reliable File Transfer
ROLAP	Relational On-Line Analytical Processing
RPC	Remote Procedure Call
RTDW	Real-time Data Warehouse
SARESA	Sense and Response Service Architecture
S&R	Sense and Response
SCD	Slowly Changing Dimension
SCM	Supply Chain Management
SDK	Software Development Toolkit
SG	Snapshot Generation
SIS	System Information Service
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TB	Terabyte
T-SQL	Temporal Structured Query Language (based on TSQL2)
TSQL2	Temporal Structured Query Language
UDDI	Universal Description, Discovery, and Integration
UML	Unified Modeling Language
WfMS	Workflow Management Systems
Win2K	Microsoft Windows 2000 ™
WinXP	Microsoft Windows XP ™
WC3	World Wide Web Consortium (www.w3.org/)
WMS	Workflow Management Service
WSDL	Web Service Description Language
WFMS	Workflow Management System
XML	Extensible Markup Language
XPath	XML Path Language
XSD	XML Schema Definition
ZLDWH	Zero-Latency Data Warehouse
ZLE	Zero-Latency Enterprise

Bibliography

[Agosta et al. 2004] Lou Agosta and Keith Gile, Real-Time Data Warehousing: The Hype And The Reality, Forrester Research Inc., December 2004.

[Andersen et al. 2000a] Jesper Andersen, Anders Giversen, Allan H. Jensen, Rune S. Larsen, Torben Bach Pederson, Janne Skyt. "Analyzing clickstreams using subsessions", *Proceedings of the 3rd ACM international workshop on Data Warehousing and OLAP (2000)*

[Andersen et al. 2000b] Jesper Andersen_ Rune S. Larsen, Anders Giverse, Torben Bach Pedersen, Allan H. Jensen, _ Janne Skyt, Analyzing Clickstreams Using Subsessions. The 3rd ACM Workshop on Data Warehousing and OLAP (DOLAP), 2000.

[Babcock et al. 2004] Load Shedding for Aggregation Queries over Data Streams", *In Proceedings of International Conference on Data Engineering (ICDE 2004).*

[Babcock et al. 2002] Models and Issues in Data Stream Systems, In Proceedings.of the 2002 ACM Symp on Principles of Database Systems, June 2002.

[Babcock, Olston 2003] Distributed Top-K Monitoring. In Proceedings. of the ACM International Conference on Management of Data (SIGMOD 2003), June 2003.

[Bailey, J. et al. 2003] An Event-Condition-Action Language for the Semantic Web. In Proceedings of the first International Workshop on Semantic Web and Databases (SWDB' 03), 2003.

[BAM 2002] Paul Skeldon

BAM ! The counter punch

Coggan Applix Article, 2002

[Bell 2004] Gordon Bell. Which key challenges do we face, which hot areas do we work on? Lecture at the Vienna university of technology, August 2004.

[Bertino, E. et al. 2000] Trigger Inheritance and Overriding in an Active Object Database System. In IEEE Transaction on Knowledge and Data Engineering, 2000. Vol.12 (4), pp. 588-608.

[Bédard et al. 2001] Bédard Y., Merrett T., Han J.: Fundaments of spatial Data Warehousing for geographic knowledge discovery. In H. Miller and J. Han, editors, Geographic Data Mining and Knowledge Discovery, pages 53–73, 2001.

[Bédard et al. 2002] Bédard Y., Marie-Josée P., Suzie L., Eveline B. : Modeling Multiple Representations Into Spatial Data Warehouses: A

Uml-Based Approach, Symposium on Geospatial Theory, Processing and Applications, Ottawa 2002.

[Boden 2004] T Boden. The grid enterprise structuring the agile business of the future. *BT Technology Journal*, 22(1), January 2004.

[Boehnlein et al. 2000] Boehnlein, M., Ulbrich vom Ende, A.: Business Process Oriented Development of Data Warehouse Structures. In: Proceedings of Data Warehousing 2000, Physica Verlag, 2000.

[Bonnet et al. 2001] Philippe Bonnet; Johannes Gehrke; Praveen Seshadri. Towards sensor database systems. In 2nd International Conference on Mobile Data Management (MDM '2001), January 8-10 2001.

[Brobst 2002a] Stephen A. Brobst:
Data Warehousing from the Trenches: Your Business Strategy is Only as Good as Its Execution.
DM Review, March 2002.

[Brobst 2002b] Stephen A. Brobst:
Active Data Warehousing – A New Breed of Decision Support.
Keynote Speech at the Intl. Workshop on Very Large Data Warehouses (VLDWH), Aix-en-Provence, France, Sept. 2002.

[Brobst 2002c] Brobst, S., Enterprise Application Integration and Active Data Warehousing, In Proc. Data Warehousing 2002, pp. 15-22, Physica-Verlag 2002.

[Brobst, Rarey 2001] Stephen A. Brobst, Joe Rarey:
The Five Stages of an Active Data Warehouse Evolution.
Teradata Magazine, pp. 38–44, Spring 2001.

[Bruckner 2002] Zero-Latency Data Warehousing: Toward an Integrated Analysis Environment with Minimized Latency for Data Propagations, Ph.D. Thesis, Vienna University of Technology, November 2002.

[Bruckner et al. 2002] BRUCKNER, R; LIST, B; SCHIEFER, J, Striving Towards Near Real-time Data Integration for Data Warehouses..Proc. of the 4th Intl. Conf. on Data Warehousing and Knowledge Discovery (DaWaK 2002), Springer LNCS 2454, pp. 317–326, Aix-en-Provence, France, Sept. 2002.

[Butte 2004] Brian Butte. Solving the data warehouse dilemma with grid technology, ibm global services. http://www-1.ibm.com/grid/pdf/GW510- 5041-00F.pdf, August 2004.

[Cannataro, Talia 2003] The Knowledge grid: An architecture for distributed knowledge discovery, In Communications of the ACM, Vol. 46, No. 1, January 2003.

[Chakrabarti et al. 2001] Approximate query processing using wavelets. In The VLDB Journal vol. 10 (2001).

[Chaudhuri 1997] S. Chaudhuri and U. Dayal, "An Overview of Data Warehousing and OLAP Technology," SIGMOD Record, Vol. 26, No. 1, 1997, pp. 65-74

[Chaudhuri et al. 2001] Surajit Chaudhuri, Umeshwar Dayal, Venkatesh Ganti: *Database Technology for Decision Support Systems.* IEEE Computer, Vol. 34(12), pp. 48–55, Dec. 2001.

[Chen et al. 1996] M. S. Chen, J. S. Park, P.S. Yu. "Data mining for path traversal patterns in a web environment", *Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS 96)*, pp. 385, IEEE Computer Society, (1996).

[Chen et al. 2000] Qiming Chen, Meichun Hsu, Umeshwar Dayal: *A Data-Warehouse / OLAP Framework for Scalable Telecommunication Tandem Traffic Analysis.* In Proc. of the 16th Intl. Conf. on Data Engineering (ICDE), IEEE CS Press, pp. 201–210, San Diego, CA, Mar. 2000.

[Codd 1993] E.F. Codd & Associates : Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate, white paper ,1993.

[Cooley 2003] Robert Cooley. "The use of web structure and content to identify subjectively interesting web usage patterns", *ACM Trans. Inter. Tech.*, 3(2), pp. 93-116, (2003).

[Colin 2002] Colin White

Intelligent Business Strategies: Real-Time Data Warehousing Heats Up

DMReview Publication, August 2002.

[Compaq 1999] Compaq Corporation. (1999)

Compaq Global Services - Zero Latency Enterprise

Website: http://clac.compaq.com/globalservices/zle/

[Darvas et al. 2004] Darvas F.; Papp A.; Bagyi I.; Ambrus-Aikelin G.; Urge L. Openmolgrid, a grid based system for solving large-scale drug design problems. In *The 2nd* IST Concertation Meeting and AxGrids2004, January 28-30 2004.

[Devlin, Murphy 1988] Barry A. Devlin, Paul T. Murphy: *An Architecture for a Business and Information system.* IBM Systems Journal. Vol. 27 (1), pp. 60–80, 1988.

[Dobra et al. 2002] Processing complex aggregate queries over data streams. In Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, 2002.

[Eder et al. 1986] Eder, J., Kappel G., Tjoa A M., Wagner R.R

BIER - The Behaviour Integrated Entity Relationship Approach, Proceedings of the Fifth International Conference on Entity-Relationship Approach, Dijon, North-Holland 1986, [Eirinaki 2003] Magdalini Eirinaki, Michalis Vazirgiannis. “Web mining for web personalization”, *ACM Trans. Inter.Tech.*, 3(1), 1-27, (2003).

[Elzbieta et al. 2004] M. Elzbieta and Z. Esteban. Representing Spatiality in a Conceptual Multidimensional Model. 12th ACM Symp. on Geographic Information Systems (ACM GIS), pp. 12–22, 2004.

[English 1999] Larry P. English: *Improving Data Warehouse and Business Information Quality.*J.Wiley and Sons, New York, 1999.

[eScience a] UK eScience Core. http://www.escience-grid.org.uk/.

[eScience b] UK eScience Program. http://www.research-councils.ac.uk/escience.

[EUGrid] European DataGrid. http://www.eu-datagrid.org.

[Fellenstein et al. 2003] Grid Computing, Prentice Hall PTR, December 2003.

[Fiser et al. 2004] On-Line Analytical Processing on Large Databases Managed by Computational Grids. Invited paper, International Conference on Database and Expert Systems Applications (DEXA04), Zaragoza, 2004.

[Foster et al. 2001] The anatomy of the Grid: Enabling scalable virtual organizations. In International Journal. Supercomputer Applications, Vol. 15, No. 3, 2001.

[Foster et al. 2002] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. http://www.globus.org/research/papers/ogsa.pdf [01.02.2004], January 2002.

[Foster, Grossman 2003] Data Integration in a Bandwidth-Rich World. In Communications of the ACM, Vol. 46, No. 11, November 2003.

[Foster, Kesselman 1998] Ian Foster and Carl Kesselman, The Grid: Blueprint for a new computing infrastructure, Morgan Kaufmann, 1998.

[Franklin et al. 2002] Streaming queries over streaming data. In Proceedings of 28th International. Conference. on Very Large Data Bases, August. 2002.

[Gartner 1998] Gartner Group:
Introducing the Zero-Latency Enterprise.
Research Note COM-04-3770, June 1998.

[Gartner 2002] David W. McCoy, Business Activity Monitoring: Calm Before the Storm, Gartner Research, April, 2002.

[Gene 1967] Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. In Proceedings AFIPS Conference (30), pp. 483-485, 1967.

[Gibbons et al. 2001] P. Gibbons and S. Tirthapura. Estimating simple functions on the union of data streams. In *Proc. Of the 2001 ACM Symp. on Parallel Algorithms and* Architectures, pages 281–291, 2001.

[Globus Toolkit 2003] The Globus Toolkit. Documentations Retrived at http://www-unix.globus.org/toolkit/

[Globus Alliance 2004] The WS-Resource Framework. retrieved at http://www-fp.globus.org/wsrf/ default.asp

[GoldenGate 2004] Ramon Chen, Tim Mueting

Rethinking Data Movement for Real-Time Business Tdwi Research, Volume 17, May 2004.

[Golob, Ozsu 2003] Lukasz Golab, M. Tamer Özsu, Issues in data stream management - ACM SIGMOD Record, Volume 32 Issue 2.

[Gonzales 2000] Gonzales L.: Seeking Spatial Intelligence. Intelligent Enterprise Magazine, Number 3, Volume 2., 2000.

[Guha, Koudas 2002] Approximating a data stream for querying and estimation: Algorithms and performance valuation. In Proceedings. of the 2002 International. Conference on Data Engineering, 2002.

[Guy et al. 2002] Leanne Guy, Peter Kunszt, Erwin Laure, Heinz Stockinger, and Kurt Stockinger. Replica Management in Data Grids. Global Grid Forum GGF5 Informational Document, July 2002.

[Güting 1994] R. H. Güting. An Introduction to Spatial Database Systems. VLDB Journal, 3(4):357–399, 1994.

[GWSDL 2003] S. Tuecke, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt, Open grid services infrastructure (ogsi), Tech. report, Global Grid Forum, 2003. 17, 20, 23

[Hackathorn 2004] Hackathorn, R., Current Practices in Active Data Warehousing, htttp://www.dmreview.com/whitepaper/WID489.pdf

[Haisten 2002] Michael Haisten, Real-Time Data Warehousing Defined, Library article from BetterManagement.com, 2002

[Han et al. 2002] Multi Dimensional Regression Analysis of Time-Series Data Streams. In Proceedings of the 28th VLDB Conference, Hong Kong, 2002.

[Hastings et al. 2005] Shannon Hastings; Stephen Langella; Scott Oster; Tahsin Kurc; Tony Pan; Umit Catalyurek; Dan Janies; Joel Saltz. Gridbased management of biomedical data using an xmlbased distributed data management system. In SAC'05 - ACM 2005 SYMPOSIUM ON APPLIED COMPUTING, March 13-17 2005.

[Hohpe, Woolf 2004] Hohpe G., Woolf B., Enterprise Integration Patterns, Designing, Building, and Deploying Messaging Solutions, Addison-Wesley, 2004

[HP 2002] Hewlett-Packard Company

Zero latency enterprise architecture

White paper, June 2002.

[Hulten et al. 2001] Mining Time-changing Data Streams. In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD01), CA, 2001.

[Humphrey et al. 2005] Marty Humphrey; Mary R. Thompson; Keith R. Jackson. Security for grids. Proceedings of the IEEE, 93(3), March 2005.

[IBM 2003] The IBM business process execution language for web services (BPEL4WS). Retrieved at http://alphaworks.ibm.com/tech/bpws4j

[Informatica 2004] Informatica Corporation

Informatica *PowerCenter Real Time*

White paper, 2004

[Inmon 1992] W. H. Inmon:Building the Data Warehouse 1st edition, 1992.

[Inmon 1999] INMON, W., Building the Operational Data Store, 2nd edition, Wiley: New York et al. 1999.

[Immon et al. 2001] William H. Inmon, Claudia Imhoff, Ryan Sousa:
Corporate Information Factory.
Second Edition, J.Wiley and Sons, New York, 2001.

[James et al. 2004] Checkpointing for Peta-Scale Systems: A Look into the Future of Practical Rollback-Recovery. In IEEE Transactions of Dependable and Secure Computing, 1(2), 97-108, 2004.

[Jensen, Dyreson 1998] Christian S. Jensen, Curtis E. Dyreson: The Consensus Glossary of Temporal Database Concepts. In Temporal Databases: Research and Practice, Springer LNCS 1399, pp. 367–405, 1998.

[Jensen, Snodgrass 1996] Christian S. Jensen, Richard T. Snodgrass: *Semantics of Time-Varying Information.* Information Systems, Vol. 21(4), pp. 311–352, June 1996.

[Joshi et al. 1999] Karuna P. Joshi, Anupam Joshi, Yelena Yesha, Raghu Krishnapuram. "Warehousing and Mining Web Logs", *Proceedings of the 2nd International Workshop on Web Information and Data Management*, 63-68, ACM Press, (1999).

[Kamvar et al. 2003] The Eigentrust algorithm for reputation management in P2P networks. In Proceedings of the 12th International Conference on World Wide Web, WWW2003, Budapest, Hungary 2003.

[KGrid a] Knowledge Grid Lab, Italy. http://dns2.icar.cnr.it/kgrid/.

[KGrid b] Knowledge Grid Project, Institute for Software Science, University of Vienna. http://www.par.univie.ac.at/project/kg.

[Kickinger 2004] The Grid Knowledge Discovery Process and Corresponding Control Structures, Technical Report, March, 2004. Retrieved at (http://www.gridminer.org/publications/gridminer2004-2.pdf)

[Kickinger et al. 2003] Workflow Management in GridMiner, In 3rd Cracow Grid Workshop, Cracow, Poland, October 27-29, 2003

[Kimball 1996] Ralph Kimball:
The Data Warehouse Toolkit
J.Wiley and Sons, New York, 1996.

[Kimball 1999] Ralph Kimball, The Data Webhouse Has No Center, Intelligence Enterprise Magazine, 1999, available at:

http://www.intelligententerprise.com/db_area/archives/1999/991307/warehouse.jhtml

[Kimball 2002a] The Data Warehouse Toolkit: The Complete Guide to Dimensional Mod-eling, 2nd Edition, John Wiley & Sons, 2002.

[Kimball 2002b] Ralph Kimball: Tricky Time Spans. Intelligent Enterprise Magazine, Vol. 5(10), June 2002.

[Kimball 2002c] Ralph Kimball: *Realtime Partitions.* Intelligent Enterprise Magazine, Vol. 5(3), Feb. 2002.

[Kimball,Merz 2000] Ralph Kimball, Richard Merz, The Data Webhouse Toolkit: Building the Web-Enabled Data Warehouse, John Wiley & Sons, Inc., 2000.

[Kim, Park 2005] Periodic Streaming Data Reduction Using Flexible Adjustment of Time Section Size, Intl J. of Data Warehousing and Mining, 1(1), 2005, pp.37-56.

[Kohavi 2001] Ron Kohavi. "Mining e-commerce data: the good, the bad, and the ugly", *Proceedings of the 7nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 8-13, ACM Press, (2001).

[Koksal et al. 1998] Koksal, P.; Arpinar, S. N.; Dogac; A.; Workflow History Management; SIGMOD Record, Vol. 27 No.1, 1998.

[Koncilia, Eder 2001] Koncilia, C., Eder, J., Changes of Dimension Data in Temporal Data Warehouses, DaWaK 2001, Springer-Verlag LNCS 2114, pp. 284–293, 2001.

[Kopetz et al. 2002] Hermann Kopetz, Günther Bauer , The Time-Triggered Architecture, Proceedings of the IEEE Special Issue on Modeling and Design of Embedded Software, Oct 2002.

[Kouba et al. 2002] Z. Kouba, K. Matousek, and P. Miksovsky. Novel knowledge discovery tools in industrial applications. In Proc. of the

Workshop on Intelligent Methods for Quality Improvement in Industrial Practice, pages 72–83, 2002.

[Kueng et al. 2001] Kueng, P., Wettstein, Th., List, B.: A Holistic Process Performance Analysis through a Process Data Warehouse. In: Proceedings of the American Conference on Information Systems, 2001.

[Kumar et al. 2002] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proc. of the 2002 Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 623–632, 2002.

[Langseth 2004] Justin Langseth, Real-Time Data Warehousing: Challenges and Solutions, Article published at DSSResources.COM, 02/08/2004

[Lerner, Shasha 2003] The Virtues and Challenges of Ad Hoc + Streams Querying in Finance. In Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, March 2003.

[Leymann 2003] Web services flow language (WSFL 1.0). Retrieved at http://www-4.ibm.com/software/solutions/ webservices/pdf /WSFL.pdf

[List et al. 2000] List, B., Schiefer, J., Tjoa A M., Quirchmayr, G.: Multidimensional Business Process Analysis with the Process Warehouse. In: W. Abramowicz and J. Zurada (eds.): Knowledge Discovery for Business Information Systems, Kluwer Academic Publishers, 2000.

[Lowell 2003] Serge Abiteboul; Rakesh Agrawal; Phil Bernstein; Mike Carey; Stefano Ceri; Bruce Croft; David DeWitt; Mike Franklin; Hector Garcia Molina; Dieter Gawlick; Jim Gray; Laura Haas; Alon Halevy; Joe Hellerstein; Yannis Ioannidis; Martin Kersten; Michael Pazzani; Mike Lesk; David Maier; Jeff Naughton; Hans Schek; Timos Sellis; Avi Silberschatz; Mike Stonebraker; Rick Snodgrass; Jeff Ullman; Gerhard Weikum; Jennifer Widom; Stan Zdonik. The lowell database research self assessment. http://research.microsoft.com/ gray/lowell/LowellDatabaseResearchSelfAssessment.pdf, June 2003.

[Luckham 2005] David Luckham, Can We Bring Some Order to the Business Activity Monitoring Space? 2005, Available at : http://www.complexevents.com/about/articles/chaos_to_order.html

[Marinescu 2002] Internet-Based Workflow Management: Toward a Semantic Web, John Wiley, 2002.

[MetaNEOS] MetaNEOS: QAP nug30 is solved.

http://www-unix.mcs.anl.gov/ metaneos/nug30/.

[Miquel et al. 2004] M. Miquel, A. Brisebois, Y. B'edard, and G. Edwards. Implementation and evaluation of hypercube-based method for spatio-temporal exploration and analysis. http://sirs.scg.ulaval.ca/yvanbedard/enseigne/SCG66124/345-A.pdf, 2004.

[Moore 2001] Knowledge-Based Grids, Technical Report TR-2001-02, San Diego Supercomputer Center, Jan. 2001.

[Motvani et al. 2000] Clustering Data Streams. In Proceedings IEEE Symposium on Foundations of Computer Science (FOCS00), CA, 2000.

[Muth et al. 1999] Muth, P., Weissenfels, J., Gillmann, M., Weikum, G., Workflow History Management in Virtual Enterprises using a Light-Weight Workflow Management System, Workshop on Research Issues in Data Engineering, Sydney, March 1999.

[Niemi 2003] Tapio Niemi; Marko Niinimki; Jyrki Nummenmaa; Peter Thanisch. Applying grid technologies to xml based olap cube construction. In *The 5th International Workshop on Design and Management of Data Warehouses - DMDW'2003*, September 2003.

[OGSA 2003] The Globus Project: Open Grid Service Architecture. Retrived at http://www.globus.org/ogsa/

[Oracle 2004] Oracle Cooperation. Grid computing for business intelligence and Data Warehousing, white paper, July 2004.

[Paradias et al. 2001] Paradias, D., Kalnis, P., Zhang, J., and Tao, Y., 2001: Efficient OLAP operations in spatial data warehouses. In Advances in Spatial and Temporal Databases, 7th International Symposium, SSTD, Redondo Beach, CA, USA, 443 - 459.

[Paton et al. 1999] Paton W., Diaz O. Active Database System, ACM Computing Survey, Vol 31(1) March, 1999.

[Pourabbas 2003] E. Pourabbas : Cooperation with geographic databases, Multidimensional Databases: Problems and Solutions, M. Rafanelli, editor., Idea Group Publishing, 2003.

[Raden 2003] Neil Raden

Exploring the Business Imperative of Real-Time Analytics

Teradata white paper, October 2003.

[Rao et al. 2003] Fangyan Rao, Long Zhang, Xiu Lan Yu, Ying Li, Ying Chen: Spatial Hierarchy and OLAP-Favored Search in Spatial Data Warehouse. ACM International Workshop on Data Warehousing and OLAP (DOLAP'03), November 7, 2003, New Orleans, USA.

[Rivest et al. 2003] Rivest, S., Bedard, Y., Proulx, M.J. and Nadeau, M.: SOLAP: A new type of user interface to support spatiotemporal multidimensional data exploration and analysis, Proc. of ISPRS workshop on Spatial, Temporal and Multi-Dimensional Data Modeling and Analysis, Québec City, Québec, Canada. 2-3 October, 2003.

[Roure et al. 2003] The Semantic Grid: A Future e-Science Infrastructure, retrieved at http://www.semanticgrid.org/documents/semgrid-journal /semgrid-journal.pdf

[Salzberg et al. 1999] Betty Salzberg, Vassilis J. Tsotras: *Comparison of Access Methods for Time-Evolving Data.* ACM Computing Surveys, Vol. 31(2), pp. 158–221, 1999.

[Schiefer et al. 2004a] SCHIEFER, J., MCGREGOR, C., Correlating Events for Monitoring Business Processes, International Conference on Enterprise Information Systems, Porto, 2004.

[Schiefer et al. 2004b] Schiefer J., Jun-jang J., Kapoor S., Chowdhary P.: Process Information Factory: A Data Management Approach for Enhancing Business Process Intelligence, 2004 IEEE International Conference on E-Commerce Technology (CEC 2004), 6-9 July 2004, San Diego, CA, USA.

[Schlesinger et al. 2005] Supporting the ETL-Process by Webservice Technologies. Intl J. of Web and Grid Services, 1(1), 2005.

[Schulte 2000] SCHULTE, W., Application Integration Scenario: How the War is Being Won, in: Gartner Group (Ed.): Application Integration – Making E-Business Work, London, 6-7 September 2000.

[Simon 2004] Simon Terr,

Real-Time Data Warehousing 101

Article published at DataWarehouse.com, March 29,2004

[SOA 2004a] Dirk Krafzig, Karl Banke, Dirk Slama, Enterprise SOA : Service-Oriented Architecture Best Practices, Prentice Hall PTR, November, 2004

[SOA 2004b] Eric Newcomer, Greg Lomow, Understanding SOA with Web Services, Addison Wesley Professional, December 2004.

[Sotomayor 2004] The Globus Toolkit 3 Programmer's Tutorial. Retrieved at http://www.casa-sotomayor.net/gt3-tutorial/

[Stefanovic et al. 2000] N. Stefanovic, J. Han, and K. Koperski. Object-based selective materialization for efficient implementation of spatial data cubes. IEEE Trans. on Knowledge and Data Engineering, Number 12 volume 6, pp 938–958, 2000.

[Stonebraker et al. 2003] The Aurora and Medusa Projects. In Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, March 2003.

[Strauss et al. 2003] Maintenance of Multidimensional Histograms, In the 23rd Conference FSTTCS 2003, India.

[StreamBase 2005] StreamBase Systems (2005) *The World's first Stream Processing Engine.* Website: http://www.streambase.com/

[StreamGlobe 2005] Richard Kuntschke Bernhard Stegmaier Alfons Kemper Angelika Reiser, StreamGlobe: Processing and Sharing Data Streams in Grid-Based P2P Infrastructures, VLDB 2005.

[SUGI29 2004] Greg Barnes Nelson, Jeff Wright

Real Time Decision Support: Creating a Flexible Architecture for Real Time Analytics, 29th SAS User Group International Conference (SUGI29), May 9-12, 2004.

[SuperComputing] Supercomputing Conference. http://www.sc-conference.org/.

[Sweiger et al. 2002] Mark Sweiger, Mark R. Madsen, Jimmy Langston, Howard Lombard. "Clickstream Data Warehousing", *John Wiley & Sons, Inc.*, (2002).

[Taniar et al. 2005] Replica Synchronization in Grid Databases, Intl J of Web and Grid Services, 1(1), 2005.

[Teradata Corp. 2002] Teradata Online Document, 2002.

[TIBCO] TIBCO Software Inc.: http://www.tibco.com

[Tjoa, Brezany 2003] GridMiner: An Infrastructure for Data Mining on Computational Grids, In APAC Conference. and Exhibition on Advanced Computing, Grid Applications and eResearch ,October, 2003.

[Thalhammer 2001] Thomas Thalhammer: *Active Data Warehouses: Complementing OLAP with Analysis Rules.* Ph.D. Thesis, University of Linz, Austria, Nov. 2001.

[Thalhammer 2002] Realizing active Data Warehouses with off-the-shelf database technology. In Software Practical Experiment, 2002.

[Thalhammer et al. 2001] Thomas Thalhammer, Michael Schrefl, Mukesh Mohania: *Active Data Warehouses: Complementing OLAP with Analysis Rules.* Data & Knowledge Engineering, Elsevier Science Ltd., Vol. 39(3), pp. 241–269, 2001.

[The Axis Project 2003] Webservices – Axis 1.1. http://ws.apache.org/axis

[Tho et al. 2004] Towards Service Collaboration Model in Grid-Based Zero-Latency Data Stream Warehouse (GZLDSWH). In Proceedings IEEE International Conference on Service Computing (SCC04), Shanghai, September, 2004.

[Tho et al. 2005] Tho Manh Nguyen, Peter Brezany, A Min Tjoa, and Edgar Weippl, Towards a Grid-based Zero-Latency Data Warehousing for Continuous Data Streams Processing Implementation, Volume 1, Issue 4, 2005, International Journal of Data Warehousing and Mining, Idea Group Publication Inc. (IJDWM)

[Tho, Tjoa 2003] Zero-Latency Data Warehousing: Continuous Data Integration and Assembling Active Rules, In

Proceedings of 5th International. Conference on Information Integration and Web-based Applications and Services (IIWAS2003), Jakarta, Feb. 2003.

[Tucker et al. 2003] Applying Punctuation Schemes to Queries Over Continuous Data Streams, In Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, March 2003.

[Vandermay 2000] Vandermay J., Considerations for Building a Real-time Oracle Data Warehouse, DataMirror Corporation White Paper,2000.

[Watanabe 2005] Tadashi Watanabe, The Earth Simulator and Future Technologies for High Performance Computers, Keynote at 11th International Conference on Parallel and Distributed Systems, Fukuoka, 2005

[Web 1999] Tim Berners-Lee, Dan Connolly, Ralph R. Swick, Web Architecture: Describing and Exchanging Data. W3C Note 7 June 1999 (http://www.w3.org/1999/06/07-WebData)

[Web Service 2005] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, Donald F. Ferguson , Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More, Prentice Hall PTR, March, 2005.

[Westerman 2000] Paul Westerman:
Data Warehousing: Using the Wal-Mart Model.
Morgan Kaufmann Publishers, San Francisco, 2000

[Widom et al 2003] Query processing, approximation, and resource management in a data stream management system, In Proceedings First Biennial Conference on Innovative Data Systems Research (CIDR), Jan. 2003.

[Yan et al. 1996] T. W. Yan, M. Jacobsen, H. Garcia-Molina, U. Dayal. “From User access patterns to dynamic hypertext linking”, Computer Networks, 28, 7-11, (1996).

[ZELESA 2005] Tho, M. Nguyen, Josef Schiefer., A Min Thoa, ZELESSA (Zero-Latency Event Sensing and Responding): An Enabler for Real-time Business Intelligence, Technical Report 123/IFS/2005 (submitted to OeNB project proposal).

Curriculum Vitae

Nguyen Manh Tho
Born: 08.12.1975, HoChiMinh City, Vietnam

Education

2002 – 2005	Ph.D. studies of computer science at Vienna University of Technology
April 1998	Diploma of Engineering. Field of Study: Distributed Database System.
1993 - 1998	HoChiMinh City University of Technology, Department of Information Technology.
06/1993	Graduation from high school.
1990 – 1993	Le Hong Phong High School, HoChiMinh City. (Specialized in Physics)
1986 – 1990	Hoang Van Thu Secondary School, HoChiMinh City. (Specialized in Mathematics)
1981 – 1986	Hoang Van Thu Elementary School, HoChiMinh City.

Work Experience

since 02/2004	Business Intelligence Solution at T-Mobile Austria (part-time)
since 2003	Research Assistant at Institute of Software Technology
2003 – 2005	GridMiner Research Project (www.gridminer.org), focus on Data Stream Processing and Analysis
2000 – 2002	Lecturer at FPT-APTECH Programmer Training School (part-time)
1998 – 2002	Being involved in a number of software projects: Developing Academic System for HoChiMinh City University of Technology (1999-2002). Developing Equipment Management System for HoChiMinh city University of Technology (1998-1999)
1998 – 2002	Lecturer at The HoChiMinh City University of Technology.

Grants / Awards

04/2005	Being 1 of 14 selected outstanding students from Vienna University of Technology attended the "Einstein in the City" Conference, New York.
09/2004	Microsoft Student Travel Awards for Research (STAR program)
12/2003	Being selected as Austrian Representative at IBM Europe Student Event Recognition, IBM-Hursley, UK 15-15 December 2003.
10/2002	Technology Grant South East Asia (No. 322/1/EZA/2002), Austrian Council Research and Technology and the ASEA-UNINET.
2/1998	Grant for Outstanding Informatics Students, Vietnamese Computer Society

Program Committee Member: DaWaK 2005

External Reviewer: iiWAS 2005, BP-UML 2005, DOLAP 2005

Peer-Reviewed Publications

- 1 articles in scientific journals
- 10 papers for international conferences and workshops
- 1 poster

Conferences and Workshops

1. **"Zero-Latency Data Warehousing for Heterogeneous Data Sources and Continuous Data Streams"**, (Tho Manh Nguyen, and A Min Tjoa), full paper accepted at iiWAS 2003, *the Fifth International Conference on Information and Web-based Applications and Services, Jakarta, Indonesia, September 15-17, 2003.*
2. **"Grid-Enabled Zero-Latency Data Warehousing for Processing Continuous Data Streams"** (Tho Manh Nguyen, and A Min Tjoa), short paper accepted at COSCI 2004, *School on Computational Sciences and Engineering: Theory and Applications, Ho Chi Minh City, Vietnam. March 3-5, 2004.*
3. **"Grid-Enabled Zero-Latency Data Warehousing for Processing Continuous Data Streams"** (Tho Manh Nguyen, and A Min Tjoa), full paper accepted at iiWAS 2004, *the Sixth International Conference on Information and Web-based Applications and Services, Jakarta, Indonesia, September 27-29, 2004.*
4. **"SemanticLIFE:A FRAMEWORK FOR MANAGING INFORMATION OF A HUMAN LIFETIME"** *(*Ahmed M., Hoang H.H., Karim M.S., Khurso S., Lanzenberger M., Latif K., Michlmayr E., Mustofa K., Mustofa K., Nguyen H.T., Rauber A., Schatten A., Tho M. N., and Tjoa A M*)* , full paper accepted at iiWAS 2004, *the Sixth International Conference on Information and Web-based Applications and Services, Jakarta, Indonesia, September 27-29, 2004.*
5. **"Towards Service Collaboration Model in Grid-Based Zero-Latency Data Stream Warehouse (GZLDSWH)"** *(*Tho Manh Nguyen, A Min Tjoa, Günter Kickinger, and Peter Brezany), full paper accepted at SCC2004, *IEEE International Conference on Services Computing, Shanghai, RP China, September 15-18, 2004.*
6. **"Grid-based Mobile Phone Fraud Detection System"** *(*Tho Manh Nguyen and A Min Tjoa), full paper accepted at the *Workshop on Access to Knowledge through Grid in a Mobile World* (Akogrimo) held in conjunction with PAKM2004, *the 5th International Conference on Practical Aspects of Knowledge Managemen, Vienna, December 2-3, 2004.*
7. **'Toward the Stream Analysis Model in Grid-based Zero-Latency Data Stream Warehouse"** (Tho Manh Nguyen, A Min Tjoa and Josef Schiefer), full paper accepted at the *Workshop on Information Just-in-Time* (WIJIT2005) *Seeking a New Knowledge Management Paradigm.* Proceedings the *3rd Conference Professional Knowledge Management: Experiences and Visions* (WM2005), *Kaiserslautern, Germany, April 10-13, 2005.*
8. **"Event-feeded Dimension Solution"** (Tho Manh Nguyen, Jaromir Nemec, and Martin Windisch), full paper accepted at DaWaK2005, *the 7th International Conference on Data Warehousing and Knowledge Discovery, Copenhagen, Denmark, August 22-26, 2005.*
9. **"Data Warehousing and Knowledge Discovery: A Chronological View of Research Challenges"** (Tho Manh Nguyen, A Min Tjoa, and Juan Trujillo), full paper accepted at

DaWaK2005, *the 7th International Conference on Data Warehousing and Knowledge Discovery, Copenhagen, Denmark, August 22-26, 2005.*

10. "**ZELESSA: Sensing, Analysing and Acting on Continuous Event Streams**" (Tho Manh Nguyen, Josef Schiefer, A Min Tjoa), full paper accepted at iiWAS 2005, *the Seventh International Conference on Information and Web-based Applications and Services, Kuala Lumpur, Malaysia, 19-21 September 2005.*

Posters

11. "**A Grid-based Zero-Latency Data Warehousing System (GZLDSWH)**" (Tho Manh Nguyen and A Min Tjoa), poster accepted at Einstein in the City, *a Student Research Conference at The City College of New York, April 11-12, 2005.*

International Journals

12. "**Towards a Grid-based Zero-Latency Data Warehousing for Continuous Data Streams Processing Implementation**" (Tho Manh Nguyen, Peter Brezany, A Min Tjoa, and Edgar Weippl) accepted for publication in *Volume 1, Issue 4 of the International Journal of Data Warehousing and Mining, Idea Group Publication Inc.* (IJDWM), 2005.