Faculté de Sciences Appliquées
Service Ingénierie Informatique
et de la Décision (CoDE)

UNIVERSITÉ LIBRE DE BRUXELLES,
UNIVERSITÉ D'EUROPE
ULB

Incorporating the Service-Oriented Architecture into Data Warehouses

Zineb El Akkaoui

Director: Prof. Esteban Zimányi

Research progress report,
Formation Doctorale
in Engineering Sciences
Academic year: 2008–2009

Contents

Acknowledgements

I would like to thank my professor Esteban Zimányi, the principal director of my thesis for his continuous support, his pertinent advices and his permanent availability. Thanks also to all my colleagues in the Web & Information Technologies (WIT) service for their help and sympathy.

I would like to warmly thank Thierry Lepage and his wife Catherine for their encouragements and presence during all these two years. Also, I would like to thank all the Cooperation Department of the Université Libre de Bruxelles, which funded my sojourn at Brussels.

I am so grateful to my husband for his sincere assistance and efficient help. His great support makes me succeed at balancing between the thesis work and the other life activities.

I would like to dedicate this work to my parents who made big sacrifices in giving me the best education. And finally, a special thank goes to my brothers for their kind encouragements.

Abstract

Decisional systems are crucial for enterprise improvement. They allow the consolidation of heterogeneous data from distributed enterprise data stores into strategic indicators. An essential component of this data consolidation is the Extract, Transform, and Load (ETL) process. In the research literature there has been very few works defining conceptual models for ETL processes. At the same time, there are currently many tools that manage such processes. However, each tool uses its own model, which is not necessarily able to communicate with the models of other tools.

In this thesis, we have proposed a solution using some Service-Oriented Architecture (SOA) standards. We elaborate a platform-independent conceptual model of ETL processes based on the Business Process Model Notation (BPMN) standard. We also show how such a conceptual model can be implemented using Web Service Business Process Execution Language (WS-BPEL), a standard executable language for specifying interactions with web services.

Chapter 1

Introduction

1.1 The Research Problem

The software market offers a diversity of ETL tools but with no specific standards, neither for modelling ETL processes nor for their implemention. As a consequence, there is no possibility to reuse the generated result, such as ETL workflows, between different tools. Also, if we consider ETL processes design models, we will find approximately as many models as of ETL tools. Unfortunately, this diversity of models generates a separated growth of the best practices and methodologies in the ETL field.

In the first steps of the data warehouse project development, the designer should focus on the business requirements specification. He should produce high level models to depict these requirements, such as the conceptual model. However, ETL tools tend to make him already worry about implementation issues at this early stage. At the same time, there is no consensus about standardizing ETL processes life-cycle.

1.2 Introduction to the Proposed Solution

The proposed solution relies essentially on the service-oriented architecture technology. The service-oriented architecture is a technology that aims to facilitate the interoperability between business services and applications and to make easy collaboration within and among organisations.

The main purpose of our thesis is to use the service-oriented architecture to enhance the interoperability among ETL tools. In this sense, the first important challenge faced was the harmonization between ETL tools languages. To deal with this problem, we have listed and explored a set of solutions and gone through the solution that seems to be the most pertinent in our opinion.

The solution we have adopted consists of an ETL process modelling framework. This framework comprises both a conceptual and a logical model for designing and implementing ETL processes inside a service-oriented architecture environment. The solution considers ETL processes as business processes and manages them in respect to the same standard life-cycle, starting by their design, implementation, execution, monitoring, and ending by analysis of the improvement factors.

The conceptual model is built on the emerging and commonly-used service-oriented architecture standard, the Business Process Model Notation (BPMN) [18]. Several reasons has motivated the use of this notation, in particular:

- Firstly, BPMN is an emergent and largely accepted standard for designing business processes;
- Secondly, it provides a palette of conceptual tools to express business processes in an intuitive language, which can be later mapped into a target execution language;
- And finally, BPMN is a notation used to design all enterprise processes uniformly so as every process can communicate easily with the others.

The logical model is based on another service-oriented architecture standard which is the Web Service Business Process Execution Language for Web Service (WS-BPEL or BPEL for short) [17]. This language is meant to be used in modelling the behaviour of web services. Some advantages of using BPEL for designing the logical model of ETL processes are as follows:

- BPEL is a mature and very popular execution language standard, which is specialized in process implementation and execution;
- Also, BPMN and BPEL are closely coupled languages;
- Many ETL tools have introduced some concepts around BPEL and adjacent technologies. It means that this proposal goes along with the competitive market of ETL tools.

In this report, we will present our solution and the adopted methodology during the research. We will start by introducing the context of the thesis and specifying the related concepts. A case study is also provided to illustrate the benefits of the proposed framework.

1.3 Contribution

Despite the particular attention given to the ETL languages harmonization as a first objective, the research have dealt with many other intermediate issues. The main contributions of our research are listed below:

- We present a conceptual model for ETL processes based on business process modelling notation [18];
- We study the translation of the proposed conceptual model into the web service business process execution language [17], a standard executable workflow language for web services;
- We list and categorize a generator base for ETL tasks composing the ETL processes;
- We analyse the pertinence of using some technologies like ontologies in ETL design automation;
- We show how ETL processes may be integrated into the service-oriented architecture. At the same time, we propose one possible design of the service-oriented ETL architecture.

This report is organized as follows. In Chapter 2 we present the data warehouse context, ETL processes being one of its components. Then, Chapter 3 introduces a set of technologies around the service-oriented architecture relevant for data warehouse enhancement. In Chapter 4 we describe related works around ETL process design in both research and industry fields. In Chapter 5 we describe our approach using BPMN as a conceptual model for ETL processes and present a running example for illustration. Chapter 6 shows the mapping of BPMN to BPEL in ETL context. And finally, Chapter 7 contains final conclusions and points to further work.

Chapter 2

Data Warehouse Context

In this chapter, we introduce business intelligence, the asset of skills, applications and technologies dedicated to help managers in decsion making. We then examine in detail the concepts related to the data warehouse, a fundamental data infrastructure for business intelligence. And we finally study the relevant data warehouse components from two points of view: technology and modelling.

2.1 Business Intelligence

Business intelligence consists of an enterprise discipline responsible of using data for analysis, making decisions, and managing performance. Business intelligence purpose is to insure enterprise management performance through making better decisions. For this purpose, it makes use of people, processes, data, business intelligence tools and methodologies. The essential sources of business intelligence are found in the operational databases, data warehouses and data marts. Business analysts use these data sources to generate reporting that informs managers of strategic trends and opportunities in the market.

Since few years ago, business intelligence has only focused on querying, reporting and analysis on top of a data warehouse. Today's dynamic and competitive business environment demands a different and a larger approach. Indeed, business intelligence starts to provide near real-time operational data. Generally, this data is extracted from systems like enterprise resource planning (ERP), customer relationship management (CRM), supply chain, marketing and other databases. This generation of business intelligence is called "operational" business intelligence.

Business intelligence supports a broad category of applications and technologies for gathering, storing, analyzing, and providing access to data employed in business decisions. Business intelligence applications include querying, reporting, online analytical processing (OLAP) and data mining. Yet, data warehouses stays the fundamental application of business intelligence, it is considered as its main information infrastructure.

2.2 Data Warehouse Definition

Data warehouses (DW) constitute the main information infrastructure for the organisation business intelligence. It is essentially designed to facilitate reporting and analysis of the organisation information. The data warehouse plays a central role in decision support systems since it provides crucial business information to improve strategic decision-making

processes [8]. The crucial information is mostly represented either as general statistics or as specific key performance indicators (KPI). The key performance indicators are measures used to evaluate the effectiveness of the organisation. A data warehouse answers generally to many related data management challenges like data quality, complex data management and storage management. Also, it may involve many options like the ability to marry unstructured data with structured data, search functions, content management systems and so on.

Like most information technology projects, building a data warehouse project respects the standard life-cycle; it is composed of several steps such as the specification of needs, the conceptual and logical modelling before the deployment and maintenance. The next section gives a general overview on the methodology adopted in data warehouse creation.

2.3 Building a Data Warehouse

Building a new data warehouse project follows an overall methodology. It starts by specifying the business and technical requirements of the data warehouse. It consists of, firstly, qualifying and quantifying the future decisional data and, secondly, analysing the whole information system where the data warehouses will be integrated. These preliminary tasks will be followed by the three main steps of the data warehouse construction, see Section 2.6.1, which consist of, first, the generation of the conceptual design. Second, the conceptual design is refined progressively and enables to produce the data warehouse logical design. Finally, the logical design is traduced to the physical one, that is the directly implementable model of the data warehouse. Once the data warehouse is built, the maintenance phase will be required during the whole data warehouse life-cycle. The maintenance task keeps the data warehouse consistent by managing updates in both data and schema levels. Further, in real world the data warehouse construction depends closely on the used implementation software, see Section 4.2.

In the following, we go through the presentation of the building blocks of a traditional data warehouse architecture.

2.4 Traditional Data Warehouse Architecture

The first widely accepted data warehouse architecture has emerged in the 1980's and continues to be the traditional consensus architecture. That points to four fundamental principles:

1. The separation between operational and decisional data;

2. The data warehouse repository tends to be complete and wide enough to hold most required decisional data;

3. Each data warehouse is characterized by a metadata dictionary that well documents the data semantics and its manipulation; and finally

4. The ability to subsetting data warehouses through data marts allows an adequate subdivision of data depending on the group of users.

These foundation principles have lead to the common traditional architecture, also called ETL architecture, shown in Figure 2.1. Further, a competitive and much closed architecture to the traditional one has emerged and has been supported by many huge industries, called ELT architecture.

The Figure 2.1 presents a traditional data warehouse and shows its four main parts, that are: the data sources providing the operational data used to compute decisional information, the data warehouse sources, the storages containing the decisional information, the Extraction Transformation and Loading (ETL) processes responsible of the conciliation between the two first components and the front-end applications.

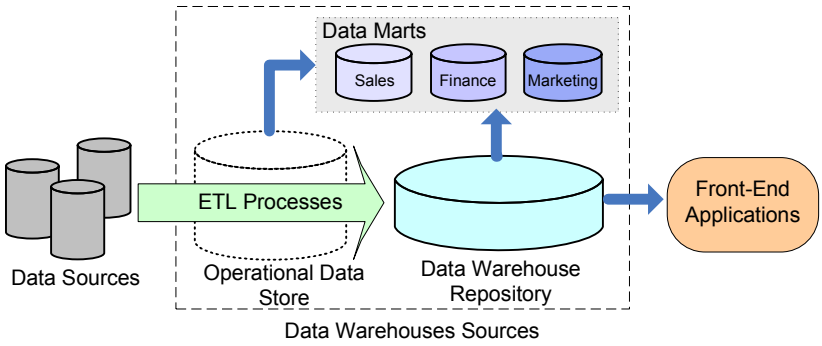


Figure 2.1: Data warehouse architecture.

2.4.1 Data Sources

In the data warehouse architecture, data sources consist of the operational databases or views that contain the production data used in populating the data warehouse. In regard to the reached decisional information, the designer should specify the adequate data sources. There exist many types and forms of data sources, such as on-line transactional processing database (OLTP), enterprise resource planning (ERP), client relationship management (CRM) or flat files.

2.4.2 Data Warehouse Sources

A data warehouse includes three types of databases: the data warehouse repository, the data marts and an intermediate data store called the operational data storage (ODS).

Data Warehouse Repository

Data warehouse repository is a database optimized for speed data retrieval. There are two main factors that enhance the speed retrieval in the repository. On the first hand, the data warehouse repository is organized in a multi-dimensional model in most of the time, see Section 2.6.1. On the other hand, data in the data warehouse repository is often stored multiple times. It is stored in both its most detailed form and its summarized form called aggregates. Note finally that the repository data is gathered from the data sources.

Data Mart

A data mart is intended to answer specific demands of a group of users. It is an analytical storage with common business purposes. One pertinent advantage of the data mart is

that it provides data in a usable form with faster access for analysis. A data mart may gather data either from the data warehouse repository or directly from the operational data storage or the data sources. Typically, each department, such as finance, marketing, or sales, has its own data mart.

Operational Data Store

An operational data store is an operational database that stores temporary the extracted data from data sources. It cleanses and prepares the extracted data to be ready for exploitation in the data warehouse repository. The schema of the operational data store is a unified representation of the data sources. An operational data store differs from a data warehouse repository by having a limited history, more frequent updates and fits to the relational model instead of multi-dimensional. In practice, operational data store aims to speed up the implementation of the data warehouse and to provide a prompt representation of the production data. However, the usage of the operational data store is not mandatory in a data warehouse, and its role can be shared between the data sources and the data warehouse repository. In the case of operational data store usage, operational data is loaded from data sources to the operational data store, is transformed and then matched and loaded into the data warehouse.

There is no consensus yet about the operational data store notion, especially between the two big leaders of data warehouses, Kimball and Inmon. Actually, Inmon asserts that an operational data store may traditionally be a subject-oriented, integrated, volatile, current valued data store, containing only corporate detailed data [7]. On the other hand, according to Kimball, an operational data store is a closed part of the data warehouse repository used to save micro-transactions history inside the data warehouse repository. Based on these micro-transactions, it is possible to make detailed and up to date analysis such as the customer behaviour. During the construction of the data warehouse, the decision about the usage of operational data store should be taken earlier in the specification step.

The next component we will present in the traditional data warehouse architecture after the data sources and data warehouse sources is the ETL processes.

2.4.3 ETL Processes

As mentioned before, the data warehouse repository obtains its data from multiple operational sources in various formats. In order to reconcile this data before transferring it into the data warehouse repository, a set of transformations and cleansing actions should be applied.

The extraction, transformation and loading processes ensure the expected quality of data required by the data warehouse repository. These processes are responsible of extracting data from heterogeneous operational data sources, transforming it (formatting, computing, structuring, applying functions, etc.) and loading it into the data warehouse. ETL development constitutes the most costly, in both time and resource, part of the data warehouse; It can take up as much as 80% of the development time in a data warehouse project. In general, ETL processes are performed at the operational data store level.

An ETL process is a composite of many ETL tasks. Each task performs a unit of work of type transforming, cleansing or controlling. The identification of ETL task is a pertinent subject, that captivates our attention during this research. In fact, an interesting question that should be raised is about the explicit definition of ETL tasks. In the real

Table 2.1: ETL tasks categorization

ETL label	**Description**	**Category**
PK	Primary key not null	Verify
NULL	Attribute not null	Verify
ANOMAL	Type, sign, coherence problems	Verify
UUNIT	Unit normalization between the mapped data	Transform
UFORMAT	Format normalization between the mapped data	Transform
CONCAT/ DECONCAT	Concatenation of fields to fit another field that is mapped to	Transform
TRANSFER	Simple transfer of data trough the network	Transfer
COMPRESS/ DECOMPRESS	Compress data in order to have better network speed	Transfer
LOADFACT/ LOADDIM	Load fact and dimensions tables	Transfer
JOIN	Scalar product without repetition	Assemble
FUSION	Simple add rows from table to another similar one	Assemble
EXTRACT	Selecting aimed data	Assemble
AGGR	Table aggregation to calculate the indicator formula, e.g.: sum, average, account, min, max...	Aggregate
ORDER	Ordering results	Aggregate

world, there is infinity of ETL tasks among transformation and cleansing activities. For instance, consider the ETL date format, the number of data format transformations found is approximately the square of the existing data formats number. The solution we have adopted to better specify ETL tasks is to create a full generator base that will define any possible ETL task as a combination of its elements. At the same time, we have tried to categorize the ETL tasks based on their semantics.

The recapitulative Table 2.1 lists the different elaborated elements of the generator base. Each line of the table describes an element and relates it to the correspondent category. The five categories are as follows:

- Verify: includes all processes that imply a constraint on data so as to guaranty its own integrity and coherence. Independently to the result;
- Transfer: is the category of tasks responsible of controlling data in intermediate level through the network, e.g. compressing or encrypting;
- Transform: in regard to the target data schema many transformations are needed to make extracted data suitable;
- Assemble: merging or splitting data among tables;
- Aggregate: depending on the indicator we want to generate, data will be filtered and grouped.

In parallel, different industrial and research attempts have made a generator base of ETL. There is always the possibility of programming an ETL activity using the provided palette by the ETL tool, or by creating customized and new function. Up to now, we have presented the traditional data warehouse architecture and examined its main concepts.

Other possible architectures will keep some or all of these components and may add others. Next, we continue by comparing this traditional architecture, called ETL architecture, with another interesting and evolving one.

2.4.4 ETL vs. ELT Architectures

The traditional ETL architecture operates in the three well-known phases consecutively: i. extracting the data from various sources; ii. transforming the data on a proprietary, middle-tier ETL engine, and finally iii. loading the transformed data onto the target data warehouse repository or, in the general case, onto the integration server.

Unlike the ETL architecture, the transformation phase in the ELT architecture is performed on the target DBMS using native SQL operators. It is also possible to use both a DBMS and a dedicated ELT engine at the target server. In fact, the ELT process starts by the extraction of the data from source tables, which is loaded into the destination server, where a DBMS is deployed. The transformations will be applied in this server and no middleware is required. The emergence of ELT is due to many reasons, some of them are to compensate some weaknesses in the ETL architecture and are as follows:

1. During the designing step, a strong understanding of the data structure and the information system architecture is required. However, at this stage the designer is more concerned about business requirements and may not be aware of the data structures of data sources;

2. Any change on business rules or accommodation of additional data sources or targets, means a significant rework on the existing ETL workflows. In fact, the highly fragmention of ETL workflow through ETL architecture components leads to a fragmented and doubled effort in order to adjust the workflow according to changes. In consequence, edits on many sub-blocks have to be done. This entails much effort especially when the person who maintains is not the designer himself;

3. The most compute intensive task in ETL workflows is the data transformations. Transformations are entirely performed in the ETL engine. The ETL engine performs generally data transformations and data quality checks often on a row-by-row basis. Hence it can easily cause bottlenecks in the overall process;

4. Further, in the ETL architecture data must be moved over the network twice: from data sources to operational data store and from operational data store to the data warehouse repository. This increases again the runtime of ETL process computing.

On the other hand, there are many positive points that motivate the use of the ELT architecture:

1. Previously, DBMS did not support an enough rich set of SQL operators to handle the complex data transformations required in the ELT workflow, but now it does;

2. ELT processes make intensive use of the bulk processing which differs from the traditional cumbersome row-by-row processing. The bulk processing allows the execution of ELT tasks on a set of data at once, and does not depend on the asynchronous data flow availability. Also, the bulk technique gives a better visibility on the flow for the designer and simplifies the debug and recovery procedures.

3. The ELT architecture uses a minimal hard and software. For example, it gets rid of the operational data store and the software implemented in.

2.5 Data Warehouse Technologies

The common ETL architecture presented in Section 2.4 and its adjacent architecture ELT architecture have been adopted by major organisations since they present much flexibility and vulnerability to requirements change. Each data warehouse component from this architecture has special related technologies.

2.5.1 Introduction

The data warehouse architecture supports a diversity of evolving software. They aim to automate the functioning of the data warehouse components.

The most popular types of these software are DBMS, ETL tools and front-end tools. ETL tools have fundamentally to communicate with a DBMS. While the use of a specific front-end tool such as an OLAP server or a data-mining system stays optional. Sometimes we find separated data warehouse functionalities assembled in the same software, as there are software which gather the totality of data warehouse functionalities. Also, the use of open source tools seems to be limited while the dominating tools are proprietary ones as shown in Section 4.2 for more details.

2.5.2 Database Management Systems

A database management software (DBMS) is a software designed in the purpose of managing all databases that are installed on a system hard drive. A database management system aims to control the creation, maintenance, transaction, user access and every manipulation on a database. A database administrator is the role responsible for controlling the database through the DBMS.

The DBMS are characterized by four essential elements, presented in the next points:

- It supports and implements database models through specific modelling languages. There is several models currently in use, such as hierarchical, network, relational, and object models. Essentially, the modelling language is used to define the schema of the hosted database in the DBMS, according to the DBMS database model.

- The DBMS administrates the data structures. There are many data structures in the area, like data records, files, fields and their definitions, and objects such as visual media. Data structures are what allow DBMS to interact with the data adequately.

- The third element of DBMS software is the data query language. The data query language works in respect to data structures. It enables users to interactively interrogate the database, analyze and update its data according to the users privileges on data. Also, it is used for maintaining the security of the database by monitoring the use of login data, and managing access rights and user privileges assignment.

- The last element is the transaction mechanism that manages multiple and concurrent access to the database by multiple users. This mechanism prevents for example

the manipulation of one record by two users at the same time since it may cause incoherence results or data records redundancy. It guarantees the famous ACID (Atomicity, Consistency, Isolation and, Durability) properties.

In the data warehouse architecture, the DBMS is commonly deployed at the data source and operational data store level. Sometimes, it is possible to use a DBMS instead of an ETL tool and the transformations are performed using the SQL native operators. It is the case of the ELT architecture that was presented in the section 2.4.4.

2.5.3 ETL Tools

The designer can set up ETL processes using almost any programming language, however building these processes from scratch can be very complex. ETL tools are thus to help in the creation and maintenance of ETL processes through built-in ETL task programs. In fact, they aim to offer a framework for design, implement and, execute ETL processes. ETL tools take place among a bunch of business intelligence tools used in the data warehouse project. An ETL tool may exists either as a specialized software or as a component of another software providing supplementary data warehouse related features.

In order to classify ETL tools, we should take into consideration an asset of factors. Some of them are of type *performance*, like execution speed, environment control, scalability, monitoring, statistics and, the others are of type *ease of use*, like windows look and feel, drag and drop files and tables, automatic mapping features. In order to get an annual classification of ETL tools refer to the web site: http://etltool.com/etltoolslist.htm.

ETL tools performance is increasing day by day. Currently, it is estimated at about multiple terabytes records system per hour using powerful servers with multiple CPUs. The use of multiple hard drives, multiple gigabit-network connections, and lots of memory contributes furthermore in this performance progress. On the other hand, the data load and extraction phases are the slowest parts of the ETL process that usually occurs in the data sources DBMS. Indeed, the DBMS may perform slowly because it has to take care of concurrency, integrity maintenance, and indexes. Thus, for better performance, ETL processing should run independently to the data sources.

Once the data is ready, it will be exploited by analysis tools like OLAP servers or data-mining systems.

2.5.4 Front-End Tools

OLAP Tools

Online analytical processing (OLAP), also called a multi-dimensional cube or a hypercube is a quick multi-dimensional analysis of data. Since historical data in the data warehouse repository is multiplying, its exploitation becomes more complex. OLAP gains data from the data warehouse repository and uses it to build a more specialized and organized insight. The output of an OLAP query is typically displayed in a matrix (or pivot) format. Inside the matrix, the rows and columns of the matrix depict the dimensions and the measures form the values.

Also, OLAP tools enable the creation and exploitation of this hypercube; they guaranty the insight into data through fast, consistent and interactive access to a wide variety of possible views on the hypercube. Next are some functionalities gained by the OLAP tools:

- Calculations and aggregations applied across dimensions, through hierarchies and members;
- Trend analysis over sequential time periods;
- Personalizing and slicing subsets for on-screen viewing;
- Drill-down to deeper levels of consolidation;
- Reach-through to underlying detail data;
- Rotation to new dimensional comparisons in the viewing area;

OLAP is implemented in a multi-user client/server architecture. It offers a consistent and rapid response to queries regardless to the data warehouse repository size and complexity. The OLAP tools encompass data mining and reporting tools that will be exposed in the next paragraphs.

Data Mining Tools

Data mining aims to extract the hidden predictive information from data. Based on the past events analysis provided by the data warehouse repository, data mining predicts future data trends and patterns. Data mining tools gain and manipulate the ready data from the data warehouse repository.

Reporting Tools

Reporting tools are used to query data sources through different views to produce a human readable and pertinent reports. Data is displayed to show relevancy to the business and keep track of key performance indicators (KPI). Reporting tools gain and manipulate the data either from the data warehouse repository or directly from operational data sources.

In this section we have reviewed the data warehouse technologies. In the next two sections we go through the design of both the data warehouse sources and the ETL process components, the two specific components in the data warehouse architecture.

2.6 Data Warehouse Modelling

In this section, we study the data warehouse from the modelling viewpoint. We will concentrate on the two components: data warehouse sources and ETL processes, since the design of the other components is quite far from our main topic. Further in the report, the focus will be particularly given to the ETL processes design.

2.6.1 Data Warehouse Sources Modelling

The data warehouse is a specialized database aiming to prepare and manage decisional information for easy exploitation. In this purpose, the data warehouse should respect an adapted conceptual model. One conceptual model widely known under the name of *dimensional fact model* was proposed by Golfarelli et al. in [4]. It organizes information using fact and dimension tables. A fact table represents the focus of analysis and contains measures that represent analysis indicators [13]. A dimension table includes attributes

allowing the user to explore the measures from different analysis perspectives. These analysis perspectives are represented through referential integrity constraints, depicted as dimension keys at the logical level. Dimension tables are not-normalized still dimensions can be composed of attribute hierarchies. Their normalization generates a *snowflake schema*. Note finally that the conceptual model of a date warehouse is generated starting from the schemas of the operational data systems. There are two principal types of conceptual modelling data warehouse: dimensional modelling [5, 4, 10] the favourite modelling technique in practice as we mentioned below, and the second type is the entity-relationship modelling [6, 16, 23].

Like the conceptual model, the data warehouse logical model consists either of a relational or a multidimensional model. The logical model is constructed using the conceptual model and a set of additional business and technical information (update frequencies, total disk space availability, etc.). Further, this model tries to produce a data warehouse schema that minimizes the query response constraint. In the case of using the relational model, the logical schema can be implemented through the *star schema*. In the case of using a multidimensional model, the implementation is likely to be both expensive and inconvenient. Note that the star schema is considered as a special case of the snowflake schema.

In the physical level, the star schema consists of the fact and dimension tables. Fact tables hold the main data, whereas dimension tables, usually smaller than fact ones, describe each value of a dimension. Dimension tables can be joined to fact ones in the way to answer the analysis requirement. The dimension tables have a simple primary key, while fact tables have a compound primary key consisting of the aggregate of relevant dimension keys.

It is overall agreed [2, 6, 3, 25] that a conceptual or logical model should precede the actual implementation of a data warehouse project. That explains the important work performed in modelling data warehouse and its components.

2.6.2 ETL Process Modelling

We believe that ETL design is a particularly closed component to its physical implementation; it deals directly with attributes data and transformations on these attributes. Although, we can depict the three traditional levels of ETL design. The conceptual model for ETL workflow is a high-level and generic model. It must be an independent-platform model that depicts the principal features of an ETL process, such as ETL tasks, the metadata used by the tasks and a preliminary order of the tasks (which generally differs from the implementation one).

In spite of the abundant literature about ETL issues, there are only a few works dealing with conceptual modelling for ETL processes. For example, the model proposed by Simitsis et al. [24] represents ETL processes as a graph where nodes match to transformations, constraints, attributes, and data stores, and edges correspond to data flows, inter-attribute relations, compositions, and concurrent candidates. An approach of mapping conceptual design to logical design was proposed in [19]. Starting from a conceptual representation, matching rules allow to obtain a logical representation ready for implementation. A wide presentation of ETL design literature is given in Section 4.1.

The ETL process logical design is a blueprint of the task flow. It should give to users a concise description of the occuring tasks, and the detailed business information about the flow. However, it stays a high-level workflow in comparison to the physical workflow.

In fact, the physical workflow defines the implementation using an ETL tool technology. It takes into consideration many technical aspects like the availability of data, error handling and debugging, the loading strategies, and so on. Also, the workflow is designed in a more optimized order.

The next chapter introduces the service-oriented architecture used in our approach to enhance data warehouse components and in particular the ETL processes.

Chapter 3

Service-Oriented Architecture Platform

The main challenge considered in our work is the harmonization between ETL tool models. To address this challenge, we have first started by listing and comparing a set of possible solutions. The result of this comparison encourages us to go through a solution based on the service-oriented architecture and some of its standards. In this chapter we present the service-oriented architecture and some of its main concepts.

3.1 Service-Oriented Architecture

The service-oriented architecture (SOA) is an architectural approach that facilitates the creation of interoperable business services and applications. Also, it intends to easily share services within and among organisations. In this sens, the idea of our thesis was to make advantage of service-oriented architecture in order to enhance interoperability within and among ETL applications.

3.1.1 Platform Presentation

The service-oriented architecture promises to be the architecture of the next generation information systems. It provides a platform for development of more flexible applications; It guarantees an efficient propagation of the business changes, i.e. business process changes, through rest of the organisation information system. Also, the service-oriented architecture isures an efficient communication between business processes and applications through the use of specific standards such as BPEL language.

We can cite briefly an asset of fundamental service-oriented architecture advantages towards the business. Most of these advantages are consequences of the closed coupling between business processes and applications:

- Any modification in the business processes are more easily and quickly replicated on the whole information system;
- It provides an end-to-end support for business process: designing, automating and optimizing;
- The service-oriented architecture approach introduces new horizons in business process optimization;

- Finally, an organisation using the service-oriented architecture technology benefits from a better reactivity to external factors, such as competitor strategies, opportunities, changes in the global market...

The service-oriented architecture is based on processes and services. Processes are responsible of composing services, hence services represent the activities of the business processes. Below, we define these two notions and some related concepts.

3.1.2 Business Process

A business process is a set of coordinated activities that are performed either by humans or by tools and which aims to realize a certain business result. Business process or workflow is the basis of almost enterprise activities like: planning, marketing, billing, shipping and reporting. Recently, it has become the main unit of business improvement measuring. It allows to master two important characteristics of the work inside an organisation: the order of the activities and their execution quality. Business processes are thus a fundamental component in the organisation since it affects directly the efficiency and effectiveness of the work.

Managing business process is hence a pertinent task because it implies a rigorous definition of the processes. The process definition includes, first, the well specification and evaluation of each activity characteristics such as the execution time and the activity participants. Second, it requires the understanding of business flow details, like synchronization and control of the flow, to construct the blueprint process. Such specification will enable the organisation to optimize the business processes and thus achieves one of its central objectives. In fact, optimizing a business process intends to minimize the workload on employees and reducing the usage of resources. Further, the automation of the business process life-cycle is possible through specific technologies, enhances the building of high-optimized business processes. In sum, a good business process management makes the organisation more reactive to internal and external changes.

3.1.3 Business Process Management

We have introduced in the previous paragraph the business process management (BPM) and shown its importance for the organisation. Next we define the business process management in more details. The business process management consists of a set of standards aiming to manage efficiently the various enterprise processes.

Business Process Management Definition

In order to make organisation stronger and profitable, business process management provides technical and business specialists with common languages for achieving their shared and separate goals. On the first hand, BPM is a strategy aiming to improve the performance of the business process through a closed cycle composed of three steps: modelling, executing and measuring. On the second hand, instead of organizing the enterprise as a number of single systems or silos, BPM organizes it around processes. Further, application of BPM is not specific to any context and does not require a particular technology. The only condition is to ensure the maximum of BPM objectives: model, simulate, execute and optimize the business processes [11], however, many standards exist around BPM. Every approach relying on BPM may be called a process driven approach.

In the real world, the emergence of horizontal software like enterprise resource planning, customer relationship management, and business intelligence solutions, has enhanced BPM in both technology level (tools) and methodology aspect. In general, BPM technologies offer some indicators to evaluate the process management quality, called key business management approaches, such as Total Quality Management, business process re-engineering, and Six Sigma.

Examples of BPM Usage in Data Warehouse Field

The BPM can be associated to the data warehouse in different ways. A traditional way is to apply a process driven approach in managing the fundamental steps of the data warehouse. In fact, there are many works showing how a process driven approach is applied in data warehouse creation and maintenance. For example, the Kimball BPM-based approach [7] has proven its success in various projects and has shifted many enterprises to a process-centred organizing. Also, the work described in [1] has adopted the process-oriented approach for making a formal requirement analysis in data warehousing. These works focus on applying BPM to the process of data warehouse creation or refreshment, however there is no approach that deals with the ETL process component as a business process itself and employs the BPM features to improve ETL processes.

3.1.4 Web Services

The web service is a combination of three technologies: the interface description using WSDL documents, the communication protocol SOAP using XML, and UDDI repositories allowing users to find services. The web service provides hence a rich description of both the service capability and the behaviour, enabling not only humans but also machines to understand. Note that the web service capability informs about the functional aspect of the web service e.g. the service executable program, whereas the behavioural aspect is addressed by its interface. Indeed, the interface informs about both the choreography and the orchestration of the service. Choreography is the focus on collaboration among processes through web services, and it explains how the service achieves its capability by means of interactions with users. While the orchestration mentions how the service achieves its capability by using other services and informs about the sequence and cardinality of exchanged messages.

3.2 Business Process Languages

We have shown that in order to manage business processes life-cycle we should represent them in respect to a model. In consequence, we need a modelling language. In the real-world, the commonly used languages to represent business processes are XML-based execution languages [14] such as BPEL [17]. These languages are proposed by BPM technology.

There are many tools providing support for the XML-based execution languages. These tools manage the life-cycle of business processes: design, implementation, execution, monitoring, and analysis of the improvement factors. In addition, these tools generally provide graphical notations for describing the logic of business workflows. Although the frequent usage of the execution language tools, they still comprise some inconveniences. For instance, the provided graphical notations are too complex and detailed to be used by

business analysts and managers. Also, each tool provides its own graphical notation and palette of constructs, which complicates communication and understanding of business processes among different organizations. In order to recover these problems, a solution through the addition of a new standard notation was proposed by BPM.

3.2.1 Business Process Management Notation

The Business Process Model Notation (BPMN) [18] was proposed by the object management group as an answer to the raised issues in the last paragraph. BPMN is a standard notation for modelling business processes. It offers to users a common and understandable framework in order to describe business processes independently of the used tools. BPMN aims at providing a notation that is understood by all categories of business users: business analysts that create an initial draft of the processes, technical developers responsible of implementing the technology that perform those processes, and business people who manage and monitor those processes. Further, there are currently many research works that study the mapping of BPMN to execution languages, in particular BPEL.

BPMN allows the representation of the enterprise processes in order to analyze and improve them in the future. In the market, many tools, such as TIBCO, intaglio, and Sparx Systems, hold this notation and give the possibility to represent a business process in BPMN and to run a simulation of the process. In spite BPMN is becoming a de facto standard for modelling business processes, it requires further efforts. In fact, the object management group is working on the next version 2.0 of BPMN which should adjust or add features like standardizing BPMN metamodel language and enabling the exchange of business process designs and their diagrams among tools to preserve semantic integrity.

BPMN provides a set of elements used in designing conceptual workflows. A summarized list is given next:

- Activities: They represent the work done as part of a business process. They can be atomic (tasks) or combined (sub-processes).
- Gateways: They allow forking or merging the flow of a process.
- Events: An event is something that happens during the execution of a process, and they can catch or throw processes. For example, a process could be modelled to start when a certain condition is met or when a timer notifies that a certain time period has elapsed.
- Swimlanes: They allow subdividing a diagram in Pools (process containers) and Lanes (division of a pool in roles).
- Artifacts: They provide additional information on the process with the purpose of making it more self-contained.

A workflow consists of a conjunction of these elements. The flow and order aspect among activities is represented by the flow connexions, depicted as solid line connectors. The exchange of messages is specified using dotted line connectors. In the chapter 5, the various categories of elements will be analyzed and related to the ETL context, the central subject in this thesis.

3.2.2 Web Service for Business Process Execution Language

WS-BPEL [17], standing for web-service for business process execution language, is a model and grammar that describes and implements the business workflow logic. It provides the means to specify business processes and interaction protocols. This language is considered as the first attempt to model the visible behaviour of services. It focuses on modelling orchestration rather than choreography.

WS-BPEL is meant to be used in modelling the behaviour of both executable business processes, and abstract business processes, the two types of processes describing web service interactions. An executable business process models actual behaviour of a participant in a business interaction. An abstract business processes is a partially specified process that is not intended to be executed. An abstract process may hide some of the required concrete operational details. It plays a descriptive role, with more than one possible use case, e.g. observable behaviour and process template.

In the real world, a BPEL process is an XML document mostly generated through graphical design tools by business analysts rather than programmers. It is executed by a BPEL engine which is a dedicated server. This engine can publish a BPEL process through a web services interface or react to trigger conditions set up inside the process itself. Practically, processes in the BPEL are constructed from primitive and structured activities. *Primitive activities* represent basic constructs, such as invoke to call a web service, receive to wait for a particular message to arrive, reply to send a response, throw to launch faults and exceptions, etc. *Structured activities* are used for combining these primitives. The most important ones are: sequence for defining an ordered set of invoked activities, flow for parallel ones, if-else for branching, while for loops, and pick for selecting one among a number of alternative paths. Each process in BPEL can declare variables and define partner links. *Partner links* describe links to partners, where partners might be services invoked by the process, services that invoke the process, or services that do both.

Furthermore, BPEL allows particular enhancements for business processes execution:

- BPEL provides constructs to describe arbitrarily complex business processes: data transiting through the process and the activities performed during the execution of the process;
- A process can interact synchronously or asynchronously with its partners and with the other process services;
- BPEL processes can be executed via their own web service interfaces, or through internal triggers defined inside the process. An external trigger is a message received on a port exposed by the process, whereas an internal trigger is time driven and is defined inside the process.

3.2.3 Alignment of SOA, BPMN, and BPEL

The service-oriented architecture aims, similarly to the business process management, at providing an end-to-end support for business processes. It is a platform for designing, automating and optimizing business processes. Also, one of the major contribution of the service-oriented architecture is the composition among the existent business processes or services in order to get new ones. In order to reach these purposes and others, the service-oriented architecture employs an asset of technologies such as the web service,

BPEL, BPMN languages, being currently the de facto standards in the service-oriented architecture. Nevertheless, the service-oriented architecture is not linked mandatory to these or any others technologies as mentioned before.

Hence, the service-oriented architecture uses the web service technology to develop services, as well as its adjacent technologies: WSDL, SOAP and UDDL. To compose services into processes we need a language, which may be a programming language like java or c#, as done by some programmers. However, this kind of languages don't allow essential features of business process management, like:

- This type of languages does not separe between the process flow through services and the business logic inside services, what restrains the implementation flexibility;
- Also, it may not be able to support the required specificities of the process management, such as multi-instances flow, long-running processes, parallel flows, complex dependencies between processes, asynchronous invocations and so on.

Execution languages like BPEL are thus more adapted for this kind of programming. In addition, in order to model the business process at the conceptual level, service-oriented architecture also proposes the BPMN standard technology. Note that a work around the mapping between BPMN and BPEL is currently in progress.

Later in Section 6.2, we will show how BPEL is used to implement business processes and the multiple technical information it requires.

Chapter 4

State of the Art

This chapter examines the state of the art around ETL process modelling. There were different approaches that have considered the problem from different viewpoints. The presented solutions are based on graphs, ontologies, UML and MDA. Further, the second section of the chapter will give a seesight on the ETL tools market.

4.1 ETL Process Design

Various approaches for designing ETL processes have been proposed in the last few years. In this section we review in a nutshell some of these different approaches.

4.1.1 Graph-Based Model

As presented in the section 2.6.2, Simitsis et al. [24] builds a meta-model that introduces a set of concepts around the ETL activity and presents them using graphical notations. This meta-model represents the ETL process as a graph where nodes match to transformations, constraints, attributes, and data stores, and edges correspond to data flows, inter-attribute relations, compositions, and concurrent candidates. The meta-model is made in regard to the existing design of data warehouse components like the star schemata. However, this work consists of basic model that needs more effort in managing: a. firstly, the interactions between ETL workflows and data stores, and b. secondly, the composition of ETL entities. In addition, more work has to be done at the optimization level.

4.1.2 Ontology-Based Models

Ontology may be defined as a formal, explicit specification of a shared conceptualisation. Concretely, an ontology provides a vocabulary of terms used to describe a certain reality. This vocabulary specifies formally the terms semantics and their inter-relationships. An ontology is presented as a set of explicit assumptions. Thus, a new term from the ontology can be formed by combining existing ones. It is important to note that an ontology is more profitable in these two types of domains: first, when there is a shared information of a domain of interest, and second, when the domain of interest is formal and machine computing.

The vocabulary of an ontology contains nine distinct components:

- *Individuals*: instances of the ontology concepts of the domain;

- *Classes*: sets, collections, concepts, types of objects, or kinds of things;
- *Attributes*: aspects, properties, features, characteristics, or parameters that objects (and classes) can have;
- *Relations*: between classes and individuals;
- *Function terms*: complex structures formed from combination of certain relations;
- *Restrictions*: constraints on an assertion inputs. They are formally stated descriptions of what must be true in order for some assertions to be accepted as input;
- *Rules*: statements in the form of an if-then sentence;
- *Axioms*: assertions in a logical form comprising the overall theory that the ontology describes in its domain of application;
- *Events*: the attributes or relations modifications.

In order to automate ETL process creation, a metadata driven approach is required. Thus, many approaches focus on the use of ontology. In fact, they all describe the source schemas by creating an application ontology as a metadata. Then, they base on ontology reasoning [20], logical algorithms [21] or even rules on graph transformation [22] to exploit the ontology and retrieve the ETL workflow.

In general, there are four fundamental steps to automate ETL processes design. All these steps may be performed semi-automatically and require the user validation when they are achieved: (a) full description of the data sources and the data warehouse schemas, (b) full description of the matching between data warehouse attributes to their corresponding data source attributes, (c) set up of automatic rules for identifying the required transformations on data source attributes to populate the data warehouse ones and (d) set up of automatic refreshment strategy of the ETL design after any data or schema changes.

Through Ontology Reasoner

In both the graph-based approach and commercial ETL tools the identification of the required mappings and transformations on operational data needs to be done manually. The work exposed in [21] proposes to automate this identification and to process the mapping between the data sources and the data warehouse automatically. In this purpose, the information about data sources and data warehouse schemas should be complete and formal. This work uses thus the ontology to capture the knowledge and the requirements regarding to the studied domain, and uses it to semantically annotate all the data stores.

The approach starts by the creation of the ontology and the annotation of the correspondent sources using this ontology. The schemas of sources and the schema of ontology are depicted as graphs in order to deal with structured and unstructured data. The data sources graphs are then mapped to the ontology graph, either graphically or using schema mapping techniques.

The approach proposes then a formal way for automatically derive the conceptual ETL design, based on the well-established graph transformation theory. Considering a target attribute, the design derivation consists of two main steps: first, selecting its relevant source attributes and second identifying the required data transformations using proposed algorithms.

Another solution using ontology was proposed in [20]. It aims at the identification of an abstract workflow, whose detailed functionality will be determined later by the designer. The approach describes the data from sources and data warehouse using the ontology and applies the ontology reasoner to guess the streaming of transformations.

Through Business Rules

In [22], ontology is used to annotate source and target data stores and build a shared ontology of the domain. Then, the designer maps the data stores attributes to the domain ontology which [22] considers to be easier than mapping it directly to target attributes as he does classically. Based on this ontology, a semi-automatic process will provide the required ETL scenario.

This approach consists of the development of a conceptual design framework based on graph edit operations. The data stores are presented as graphs where nodes are schema elements and edges are containment and reference relationships. Domain ontology is represented as a RDF schema where classes and properties are the concepts and relationships of the domain. The specific relationships "partOf" and "typeOf" are added to the typical hierarchy relationship "isa". Furthermore, ETL processes are considered as series of operations applied to the sources to meet the target specification. In fact, in the first hand there are insertion, deletion and relabeling of a node or an edge. In the second hand, specific operations are added to represent ETL transformation, cleansing and to control the applicability of these operations.

In order to automate the creation of ETL tasks, the approach builds on the domain ontology a set of graph transformation rules. The construction of the ETL process is performed step by step according to these rules until no rule can be applied. Also, a qualitative evaluation of the approach has been applied to the approach in order to evaluate its pertinence.

Discussion

The initial objective through automating the creation of ETL processes is to make this task easier for non-technical designers. However, the ontology usage presents a somewhat burden responsibility. Actually, the whole task of annotating the source and target schemata is done manually. It means that the designer should be an expert of all the sources schemas and structures in addition to all the business constraints of the domain. In this sense, the primary objective of simplicity is not well-answered.

Moreover, building a complete and well formed ontology remains an important concern. An ETL designer is not necessary a specialist in OWL/RDF ontology languages and ontology tools like Protégé. So, creating this ontology would not be obvious for the designer, he will need trainings or an external expertise has to be reached. At the same time, in ontology construction the risk of errors or ambiguities stays important, which will be reflected on the automatically generated ETL process.

Although, the creation of an ontology can be defended. On the first hand, ontology creation effort may just replace the effort during the traditional phase of the manual mapping between to two source and target schemas. On the other hand, an ontology will be created once and used for all the ETL processes specification in contrast to the traditional mapping which is performed evey time. Further, this ontology can be used in other applications as well. We mention two prominent examples, the production of NL-style (natural language) of reporting, and the web data warehousing.

Hence, the adequate solution may be a compromise between the advantages reached by the use of ontology and the effort necessary for its construction depending on the context.

4.1.3 UML-Based Model

The Unified Modelling Language (UML) framework for modelling the mapping between the data sources and the target data warehouse has been proposed in [12]. It formulates this mapping at the database and table levels to detailed inter-attribute mappings at the attribute level. The approach aims to extend the widely accepted formalism, the unified modelling language, in order to model ETL processes. Hence, it adds to the UML formalism some new feature like modelling data attributes as first-class citizens. Also, it provides complementary views of the design artifacts in different levels of detail and allows zooming in and out on the ETL scenario design.

In contrast with the previous presented approaches, this solution focuses on proposing a conceptual model for ETL processes through a graphical design, and the identification of the required mappings and transformations needs to be done manually.

4.1.4 MDA-Based Model

The model-driven architecture (MDA) is a software design approach for software development. It relies essentially on the basic following models: the computation independent model (CIM) as a requirements specification model, the platform independent model (PIM), an abstract solution model for requirements, equivalent to the conceptual model and the platform specific model (PSM), a solution model dependent upon specific platform technologies.

The approach in [15] concerns the application of the model-driven architecture to automate the generation of the logical schemata from the conceptual one during the data warehouse development steps. It describes how to build the different MDA models for each data warehouse components, based on the UML and the common warehouse metamodel (CWM). Transformations between models are also clearly and formally established through the use of the Query/View/Transformation (QVT) language. Note that QVT is a standard of the object management group for modelling transformations. This language formalizes the process of converting between models in respect to their meta-models.

Concretely, the first step in this approach is the development of a CIM that collects and specifies user requirements. Then, according to this CIM, each data warehouse layer is modelled by a corresponding PIM using several UML profiles. The resulting PIM models represent conceptual models of each data warehouse component. Then, each PIM can be automatically mapped to several PSM models using formal QVT transformations, depending on the required target platform. Finally, each derived PSM model provides the implementable code according to a specific data warehouse platform.

In this section, we have passed in review a set of research field approaches aiming to model or automate the ETL processes. The next section gives a seesight on the industrial market of ETL tools.

4.2 ETL Tools

We have seen that ETL tools are widely used for extracting, cleaning, transforming and loading data from different systems, often into a data warehouse. They provide developers with an interface for designing source-to-target mappings, transformation, and control parameters. The present section thoroughly examines a set of open source and proprietary ETL tools. There are many potential scenarios where ETL tools are typically used. For example, we can cite data conversion and migration projects, synchronisation of data between applications such as CRM/ERP, business-to-business data exchange and so on.

4.2.1 Open Source Tools

Oracle Warehouse Builder

Oracle Warehouse Builder (OWB) is an open source ETL tool for relational and dimensional data warehouse modelling. It uses the Oracle database as its main metadata repository and transformation engine. Also, Oracle Warehouse Builder manages security and scalability of the built-in data warehouses.

OWB generates an Oracle metabase (OMB) code as a translation of the graphical user interface. OMB is the scripting language in Oracle that allows the use of OWB features programmatically. The user interface of OWB version 11g and earlier is quite user-friendly and requires a minimum knowledge of the Oracle database administration to work on.

Talend Open Studio

Talend Open Studio is an open source that embodies a set of data integration products. This tool generates a code in Java and Perl. In order to make the modelling task easier for business analysts, Talend proposes a notable feature of the product which is the business modeller. This feature makes the analyst quickly involved in the process development and enables collaboration. Talend bundles other interesting services as well, such as metadata management, high-performance deployment options and has recently introduced a support for change data capture to enable real-time operating environments.

Pentaho Data Integration

Pentaho Data Integration includes many sub-tools such as query and reporting, interactive analysis, dashboards, data integration or ETL, and data mining. However, it is possible to use only one part of the suite. Note that the popular ETL project Kettle product was merged into Pentaho to constitute its data integration platform. the Kettle project has been successfully used for spatial ETL through its GeoKettle version. GeoKettle includes extensions which enable the use of geographical data in order to create maps and perform spatial analysis. Similarly to Talend, Pentaho development is coded in the Java language.

4.2.2 Proprietary Tools

SQL Server Integration Services

SQL Server Integration Services, called SSIS, is the Microsoft component dedicated for data integration. It works well in SQL Server environments, but it can also be used in

other non-SQL server databases.

Business Object Data Integrator

With Business Objects Data Integrator the following benefits may be gained: there is an easy-to-use codeless user interface to faster development, it supports any heterogeneous data environment, provides a powerful data quality capabilities, and includes prebuilt solutions for enterprise application using data marts as data sources,

4.2.3 Summary

In the ETL software market, there is a direct competition between proprietary ETL tools and various open source such as OWB, Pentaho Kettle, and Talend.

Specialized ETL tools vendors competes at the same time with database vendors, analytics and business intelligence vendors with an ETL component, new start-ups, and indirect competitors enterprise application integration.

In order to explore the commercial proposed ETL process models, we have studied a set of ETL tools and tried to make a small personal survey. Unfortunately, it is difficult to rely on the tools presentation web sites since all of them have clearly a marketing aspect. A such survey requires thus to test every platform with an exhaustive use case. The table 4.1 below depicts some results of this comparison in regard to a set of important characteristics.

The table 4.1 lists a set of criteria, some of them are technical, others are qualitative and subjective opinions. The technical criteria are as follows:

- Supported databases, which is the set of the supported databases;
- Integrated OLAP system, that says if an OLAP system is integrated to the ETL tool or not;
- ETL tasks extensibility meaning the possibility to program new ETL tasks;
- ETL expressions is the language used in programming new ETL tasks;
- Deployment language is the ETL process execution language used for orchestration between processes; and finally
- ETL philosophy depicting which architecture is used by the tool: ETL or ELT.

Finally, the survey gives two subjective opinion about the easy to use and the cost of the software.

The choice among these tools depends upon the type and size of the project. In this regard, Microsoft and open source solutions are preferred by medium-range projects but these are regarded as less suitable for large-scale projects. Business objects, Informatica and Oracle warehouse builder are all deemed appropriate for large-scale projects but not so much for medium and smaller projects.

In the next chapter, we will expose the proposed conceptual model for ETL process design. We will start by describing a running example used to illustrate the model.

Table 4.1: ETL tools comparison

Feature	**Oracle OWB**	**Microsoft SSIS**	**Talend**
Supported databases	Oracle, SAP R3, Flat Files, ODBC, DB2, Sybase, Informix, SQL Server, Mainframe	Through its OLE DB, Oracle, SQL Server and Sybase, flat file	ALL databases
Integrated OLAP system	YES	NO	NO
ETL tasks extensibility	YES through MDL	YES through TSQL routines	YES
ETL expressions	Specific Oracle Expression	Integration Services Expression, c#, T SQL	GUI ETL traduced to Java - Perl, specific Talend ETL expression
Deployment language	XML Process Definition Language	SSIS Engine language	SpagoBI - Talend engine (Tomcat, JBoss, JonAS)
ETL philosophy	ELT, rather than providing an external data-transformation engine, OWB executes all of its transformations within an Oracle database	ETL like Business Objects, SAS	ELT
Easy to use	NO	YES	YES
Cost	FREE	SQL Server License, relatively not expensive	FREE

Chapter 5

Applying BPMN to ETL Processes

Data warehouse project, considered as a set of enterprise activities, can de facto make advantage of the standards and techniques used in Business Process Management. This chapter shows how BPM may describe ETL processes as a coordinated BPMN workflows. It proposes a high-level and platform-independent approach for ETL design that consists of a conceptual model based on the Business Process Model Notation (BPMN) [18]. Before we go through the description of the conceptual model, we will expose the running example used in the rest of the report.

5.1 Running Example

Populating a data warehouse is done by a complex ETL process composed of many tasks. Each task achieves a unit of work, e.g., loading a dimension or updating a fact table, and usually consists of several steps. To illustrate our contribution, we consider an excerpt of an ETL process that loads a geographical table DimGeo into a data warehouse. Figure 5.1a shows two tables of the source database from which the data will be extracted. Figure 5.1b shows the temporary tables used in the transformation process with sample data illustrating several issues that must be resolved. Figure 5.1c shows the geographical dimension of the data warehouse. This dimension is composed of four tables defining a hierarchy: DimGeo, DimState, DimCountry, and DimArea.

The DimGeo table takes its data from the City, State, ZipCode, and Country attributes from tables Customer and Supplier. However, populating the DimGeo table is not straightforward since the geography data needs some cleansing and transformation operations to fit the data warehouse requirements. There are many issues that the ETL process has to resolve in order to consolidate the data. For concision reasons we will only consider here three types of issues.

The first issue concerns *data completion*, in particular dealing with null values. As can be seen in the sample data in Figure 5.1b, the attribute State may be null in the Customer and Supplier tables. Therefore, data should be filled with its corresponding value during the ETL process. For this, an external source file Cities.txt is used. The file contains three fields city, state, and country, separated by tabs as follows:

```
Atlanta - Georgia - USA
Austin - Texas - USA
Berlin - Berlin - Germany
...
```

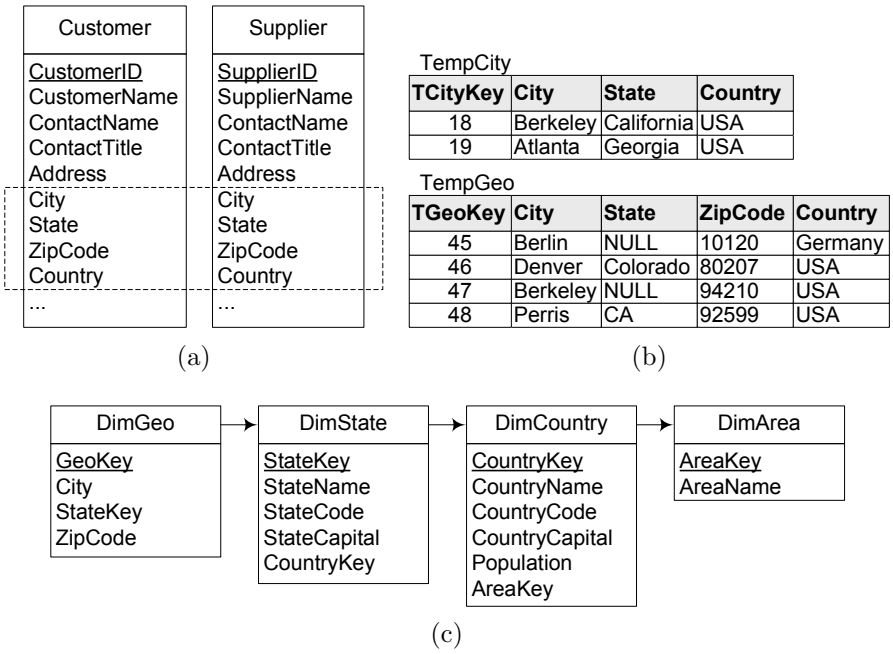


Figure 5.1: (a) Tables from source database; (b) Temporary tables; (c) Geography dimension in the data warehouse.

The second issue concerns *data consolidation*. In the source databases, the attribute State contains either a state name (e.g., California) or a state code (e.g., CA). Thus, in the latter case, the state code must be converted into the state name using an intermediate table StateAbbr with attributes StateId, StateName, and StateAbbr, which contains the link between the state name and its code.

The last issue concerns *consistency*, in particular with respect to referential integrity constraints. During the loading of the prepared data into the data warehouse, the data must comply with the referential integrity constraint between the DimGeo and DimState tables.

The workflow depicted in Figure 5.2 is used to populate the DimGeo table and takes into account the three above issues. We give next a detailed description of this workflow.

The first step consists in extracting data from the source database, the file Cities.txt,

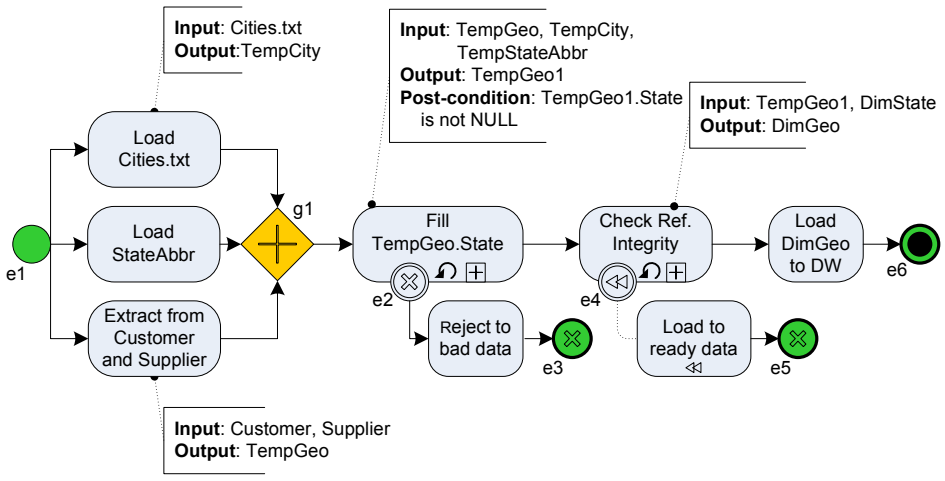


Figure 5.2: Example ETL workflow.

and the StateAbbr table and loading it into the temporary tables. This is done by the tasks Load Cities.txt, Load StateAbbr, and Extract from Supplier and Customer, respectively. Notice that these activities may be performed in parallel because there is no dependency between them; this allows the process execution to be optimized. After that, a synchronizing control g1 is used to ensure that all loads have been achieved, since all the data is required for the next task.

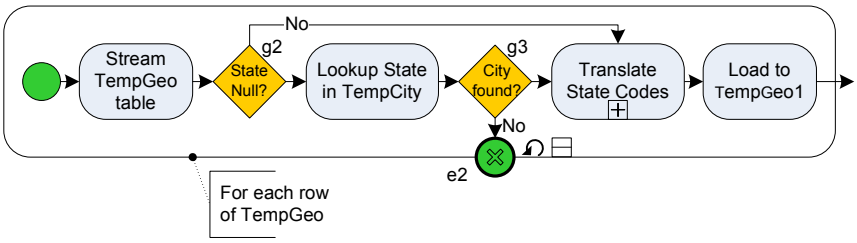


Figure 5.3: Fill TempGeo.State subprocess in expanded form.

The next task, Fill TempGeo.State, is shown collapsed in Figure 5.2. This task is a looping container that generates a cancel event e2. Figure 5.3 expands this task at a more detailed level and shows its chain of activities.

The first activity consists of streaming the table TempGeo for a row-by-row processing. Then, the null condition is checked on the TempGeo.State attribute for every row using the gateway g2. Depending on the evaluation of the condition, one of the two branches is activated: if the attribute is null, the activity Lookup State in TempCity is launched, otherwise, activity Translate State Codes is performed.

The activity Lookup State in TempCity addresses the data completion issue mentioned above. It attempts to fill in the rows with a null state by looking up the state name in the TempCity table using the city name in TempGeo.City. In the case the city is not found, an error event e2 is generated. Correct data is then ready for the outgoing task.

The next subprocess Translate State Codes handles the data consolidation issue mentioned above. It translates state code values in TempGeo.State into state name values using the table StateAbbr. This subprocess is repeated for all TempGeo rows as marked in the annotation linked to the loop container.

Finally, the subprocess Check Ref. Integrity addresses the consistency issue mentioned above. It checks the referential integrity constraint between the DimGeography and DimState tables. This subprocess filters inconsistent rows and sends them to the activity Load to Ready Data for subsequent processing and then the whole process is cancelled, whereas correct rows are carried to the task Load DimGeo to DW.

As shown in Figure 5.2, the workflow is initiated by the starting event e1, which is activated either manually or automatically at the end of the previous process. The workflow is finished by either launching the end event e6 in a normal execution, or by a cancel, error, or compensation event (e.g., compensation event, e5) in abnormal cases.

Notice that in this example workflow we assumed many simplifying hypothesis, such as: (1) the DimGeo table is initially empty; (2) Each city listed in the file Cities.txt matches to only one state and occurs only once; (3) the corresponding source and target attributes have the same type and format, etc. However, a realistic solution for this ETL process requires to handle these and several other issues.

As noted before, this workflow is a small excerpt from the ETL process populating the data warehouse. Figure 5.4 shows a fragment of the whole process in a high-level viewpoint, where the workflow in Figure 5.2 is represented by the activity Load DimGeo.

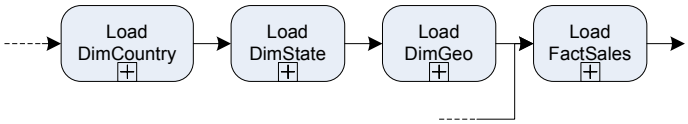


Figure 5.4: A high-level viewpoint.

As shown in the figure, this activity should be preceded by other tasks that load the DimState table, the DimCountry table, etc. Also, the Load DimGeo activity should precede the loading of the corresponding fact table.

In this section, we have illustrated the use of BPMN for describing ETL processes by an example. We will give in the next section a more general description of this approach.

5.2 Description of the Conceptual Model

An ETL process can be considered as a particular type of business process. As is the case with traditional business processes, there is no standard model for defining ETL processes, each of the existing tools provide their own model. Further, these models are often too detailed, since they take into consideration many implementation issues.

In this section we show how BPMN can be customized for designing ETL processes. We describe the constructs composing the proposed ETL palette. These constructs are grouped in four categories: flow objects, artifacts, connecting objects, and swimlanes. We divided flow objects into *ETL tasks*, which is the fundamental construct of workflows, and *control objects*, which highlight the control procedures provided by BPMN.

5.2.1 Flow Objects

ETL Tasks

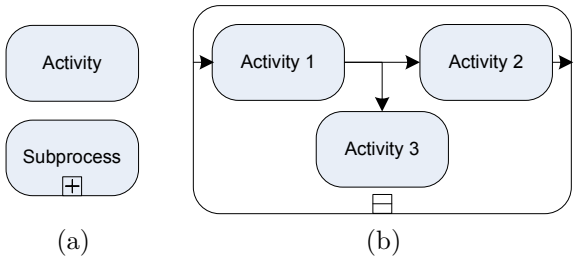


Figure 5.5: (a) ETL tasks; (b) Expanded subprocess.

An *ETL task* represents a simple or a composite unit of work of the integration workflow. It is the central object describing what is performed in the workflow. ETL tasks are modelled in BPMN as an activity or a subprocess, see Figure 5.5a. An *activity* describes a unitary task that is not further subdivided, whereas a *subprocess* represents compound activities and expresses the hierarchical nature of workflows. In the running example, see Figure 5.2, Load Cities.txt is a simple activity, while Fill TempGeo.State is a subprocess. A subprocess is also a context of exception; it allows the scope of the process where an exception is applied. For example, the above subprocess represents the context of exception for the compensation event e2. As shown in Figs. 5.5a and 5.5b, there are two ways to represent a subprocess: collapsed or expanded. Note that data annotation may

be associated to both activities and subprocesses in order to describe their semantics, as will be explained later.

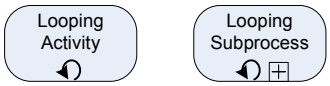


Figure 5.6: Activity and subprocess loop.

A loop, see Figure 5.6, is an execution control feature that leads to the reexecution of an ETL task a number of times depending on a Boolean condition. There are two types of conditions depending on whether they are checked before or after activity. A loop is ended if its condition is evaluated to false. The loop condition may be written in an annotation as depicted in the expanded Fill TempGeo.State subprocess in Fig 5.3. The loop feature is useful in the ETL context, especially for pipelining tasks that are performed in a row-by-row manner.

ETL tasks are subdivided into three categories: row operations, rowset operations, and control operations. We briefly present next the possible types of tasks inside each group.

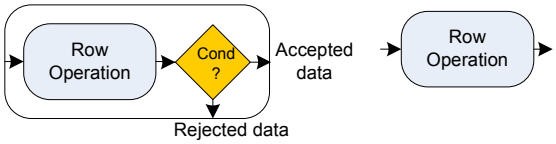


Figure 5.7: Row operation: (a) with and (b) without filter.

Row operations are transformations which are applied to the source or target data on a row-by-row basis. There are two types of row operations:

1. Row operations with data-based control. They apply a conditional filter and the rejected data may be either further manipulated or kept aside. These operations are depicted in BPMN as an activity linked to a data-based gateway (Figure 5.7a). Such operations are used for tasks such as checking primary keys and foreign keys, unique values, not null values, domain mismatch, and lookup.

2. The second type of row operations does not apply a filter. Examples of such operations are selection, projection, type conversion, and surrogate key assignment. They are represented as a simple task, see Figure 5.7b. Note that row operations are performed inside a loop that fetches data on a row-by-row basis and transfer them to the operation.

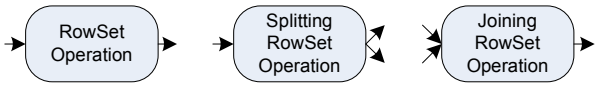


Figure 5.8: Rowset operations.

In contrast, *rowset operations* deal with a set of rows. For example, aggregation, sort, pivot, join, union, and difference are all rowset operations. This type of transformation is represented by a simple, splitting, or joining task, see Figure 5.8.

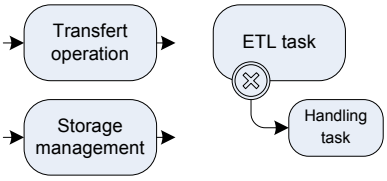


Figure 5.9: Control Operations.

Finally, *control operations* highlight part of the control aspect in ETL scenarios. This type of operations includes transferring data through the network, managing files and data storage, and error control operations, as depicted in Figure 5.9. Such operations are normal ETL tasks that use exception, error, cancel, and compensating events, see Figure 5.11b.

Control Objects

Control objects manage the workflow sequence, or orchestration, independently of the data transiting through the process. BPMN includes a set of control elements, like gateways and events.

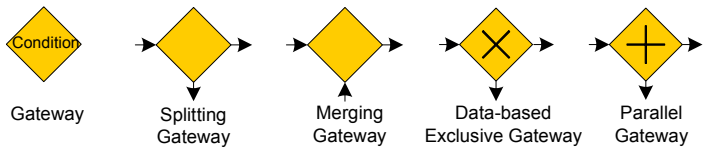


Figure 5.10: Different types of gateways.

Gateways, see Figure 5.10, are used to control the activity sequence in an ETL process based on conditions. They may either embody the condition or be linked to a *gateway condition* artifact that includes the condition (more details are given in Section 5.2.2). BPMN defines several types of gateways, like exclusive, inclusive, event based, or data based events; further, they can be splitting or merging. The most used types in an ETL context are data-based exclusive gateways and parallel gateways . Examples of such types are g1 in Figure 5.2 and g2 in Figure 5.3, respectively. An *exclusive gateway* models a decision: depending on a data condition the gateway activates one or more of its outgoing branches. A *parallel gateway* expresses synchronisation between ingoing flows as a condition for activating the outgoing flows.

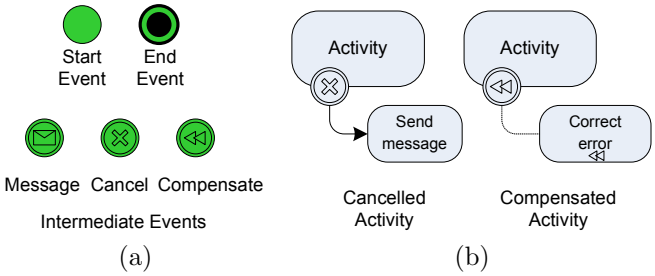


Figure 5.11: (a) Examples of start, end, and intermediate events; (b) Error and compensation handling.

Events, see Figure 5.11a, represent something that happens that affects the sequence and timing of the workflow activities. These events may be internal or external to the task into consideration. They are categorized into start, intermediate, or end events. Events may be placed along the workflow like an activity, as is the case of the event e1 in our example, or on the boundary of the activity shape or the subprocess, as for e3. There are typical usages of events in an ETL context. For instance, a start event of type Time may be used to represent the execution schedule of the ETL process, while a normal start event may be used if the process is simply triggered by the end of its predecessor process. Further, other common events include error, message, cancel, and compensate events. In this chapter we studie only error and compensation handling, and refer the reader to the BPMN specification [18] for more details.

The *error handling* is a particular intermediate event. It listens to the process errors and notify them either by an explicit action like sending message, e.g. the cancelled activity sends message as shown in Figure 5.11b, or by an implicit action that will be defined in the next steps of the process development. Besides detecting errors, *compensation handling* can also be employed to recover errors by launching specific compensation activities, which are linked to the compensation event with the association connecting object as shown in Figure 5.11b. For example, an error event may send an e-mail alert notifying the failure of a task or a process, while a compensate event will try to correct an error by executing an additional activity before restarting the execution of the concerned part of the workflow.

5.2.2 Artifacts

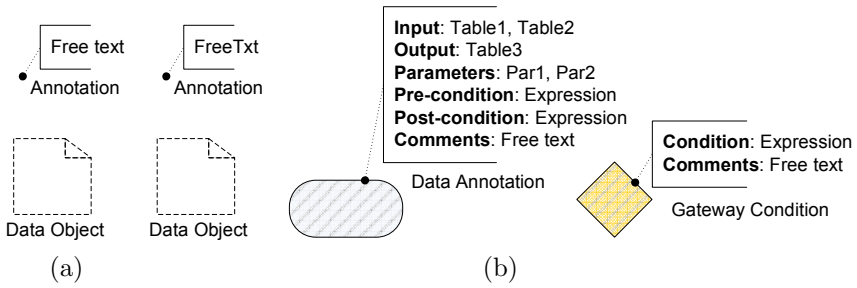


Figure 5.12: (a) BPMN artifacts; (b) Our anotations.

The main BPMN artifacts are annotations and data objects, see Figure 5.12a. A *data object* is used traditionally to represent transiting documents between tasks (like invoices or contracts), whereas an *annotation* is used to express semantics about flow objects. In our approach, annotations are specialized into two new objects, see Figure 5.12b: data annotations and gateway conditions.

Data annotations are used to make explicit the semantics of an ETL task. They include several features: *input* and *output* data of the task, *parameters*, which refer to additional data used by the task, *pre-* and *post-conditions*, and *comments*, describing any other useful semantics. Depending on whether the task is a row or rowset operation, the input and output data may be either a table or a row. As shown in our example, the traditional Table.Attribute notation is used in annotations.

Gateway condition state the conditional expression of a gateway that imposes some control on the flow, e.g., decision or synchronization. In some cases, the semantics of the gateway is expressed graphically and does not need an explicit gateway condition,

as for parallel gateways. Gateway conditions include two features: *condition*, stating the conditional expression of the gateway, and *comments*, which allows to express any useful aspect of the gateway.

5.2.3 Connecting Objects

In BPMN there are three types of connecting objects: *sequence flows*, *message flows*, and *associations*. Figure 5.13 illustrates these types of connectors.

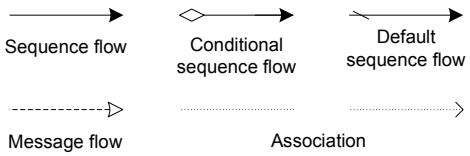


Figure 5.13: Connecting objects.

A *sequence flow* represents the sequencing constraints between flow objects. It is the basic and essential connecting object in an ETL workflow. It states that if two activities are linked by a sequence flow, the target activity will start only when the source one has finished. If multiple sequence flows are emerging from a flow object, all of them will be activated after its execution. In case there is a need to control a sequence flow, it is possible to add a condition to the sequence flow by using the *conditional sequence flow* or by using a gateway. A sequence flow may be set as the *default flow* in case of many splitting flows.

A *message flow* represents the sending and receiving of messages between the pools (see below). In a BPMN diagram, separate pools represent different entities. A message flow is the only connecting object able to get through the boundary of a pool and may also connect to a flow object within that pool.

An *association* relates artifacts to flow objects. It is also used to link the compensation activity for in case of compensation handling as noted previously.

5.2.4 Swimlanes

A *swimlane* is a structuring object that comprises *pool* and *lanes*, see Figure 5.14. Both of them allow the definition of process boundaries. As stated above, only messages are allowed between two pools, no sequence flows. Further, one workflow must be contained in only one pool. However, one pool may be subdivided into many lanes, which represent roles or services in the enterprise. Lanes within a pool do not have any special constraints and thus, sequence flows may cross a lane freely.

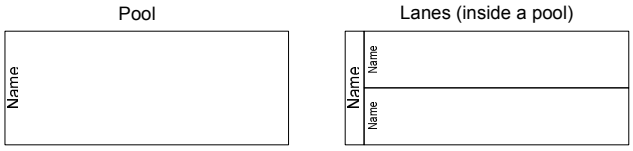


Figure 5.14: Swimlanes: Pool and lanes.

Swimlanes allows the organization and the hierarchization of several different and multi-level ETL processes for populating or updating a data warehouse. Swimlanes allow ETL processes to be organized according to several strategies: (1) technical architecture

(e.g., localisation of tasks in servers, applications, interfaces), as is the case of the example in Figure 5.15, (2) by user profile (e.g., a manager, analyst, or designer) that gives special access rights to the user, or (3) by business entities (e.g., a department or company).

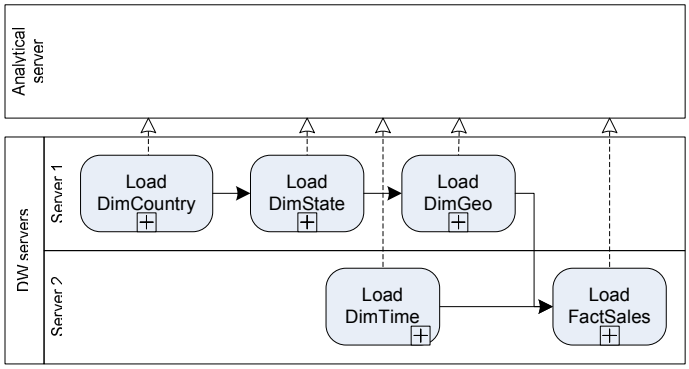


Figure 5.15: Swimlane example.

We illustrate the swimlane notion in the running example, seen from its high-level viewpoint, see Figure 5.4. Figure 5.15 shows some ETL subprocesses: loading the geography dimension, the time dimension, and the sales fact table; it also depicts their distribution between Server 1 and Server 2. Each one of these servers is considered as a lane contained inside the pool of DW servers. The figure shows some messages between the DW servers and the Analytical server pools in order to highlight the interactions between those systems.

Chapter 6

Implementation of ETL Processes in BPEL

In this report we has also studied the translation of BPMN into Business Process Execution language (BPEL) [17], a standard executable workflow language for web services.

6.1 Introduction to the Implementation in BPEL

Web Services Business Process Execution Language [17] is a standard executable language for specifying interactions with web services. It defines business processes using an XML-based language. BPEL is supported by many tools, which provide sophisticated supervision and analysis of process execution.

BPMN can be used as a graphical front-end to capture BPEL process descriptions. Mappings from BPMN to BPEL have been proposed and are implemented in several tools, e.g., Oracle BPM Suite and ActiveVOS. Nevertheless, there are some differences between BPMN and BPEL. For example, these languages are used in a different stage of the life-cycle of business processes: BPMN is used for designing business processes whereas BPEL is used for implementing them. Further, since BPMN is used by business analysts while BPEL is used by technical analysts and programmers, these languages use different paradigms and focus on separate issues when modelling a process. A more closed integration between BPMN and BPEL is expected in the next versions of the two languages.

In the ETL context, interactions between ETL tasks and subprocesses are managed by an ETL execution engine using dedicated languages. In our approach we have chosen BPEL for such purpose. However, in order to map a BPMN workflow into an executable language such as BPEL it is necessary that the designer provides additional information about the process. This is similar to what happens when translating conceptual models, such as the ER model, into logical ones, such as the relational model. In this activity, implementation-level information about the tasks, events, etc., must be provided. Furthermore, the designer must proceed to a decomposition of each top-level activity into a detailed subprocess. This action is repeated until atomic activities (i.e., implementable ETL operations) are reached.

Then, the mapping that consists in transforming each BPMN object into its equivalent in BPEL can be carried out. Such mapping may be automatically generated by some tools. Two approaches could be followed for this. One possibility is to describe in a very detailed way the BPMN workflows by filling in all the workflow attributes, prior to the automatic

mapping. This requires an important technical effort for the designer. Alternatively, the designer could complete the skeleton of the BPEL process generated by the automatic mapping.

6.2 Translating BPMN into BPEL

We transform next a fragment of our example ETL process depicted in Figure 5.2 into BPEL. In the discussion we emphasize how BPMN objects are mapped into BPEL elements.

A first step to perform is the translation of the basic information about the whole process. The four main sections composing the BPEL process definition must be specified. These sections are the following: partnerLinks, variables, faultHandlers, and process.

6.2.1 Mapping Basic BPMN to BPEL Attributes

We start by mapping the basic information about the process loading the geography dimension, such as the business process name and related name spaces. The code below depicts the header tag of a BPEL process.

```
<process name="LoadDimGeo"
  targetNamespace="http://nwetl.com/ws-bp/
    loadDimensions"
  xmlns="http://docs.oasis-open.org/wsbpel/2.0/
    process/executable"
  xmlns:lns="http://datawarehousing.org/wsdl/
    loadDimensions">
```

6.2.2 Mapping Services to PartnerLink Elements

The second step is to list the different partnerLink elements. In our approach, we consider ETL tasks in the BPMN process as being of type service; they are thus mapped to partnerlink elements. Each partnerLink is characterized by a partnerLinkType and one or two role names. This information identifies the functionality of the service or the partner providing the service.

```
<partnerLinks>
  <partnerLink name="loadTxtFile" partnerLinkType=
    "lns:loadTxtFileLT" myRole="fileLoader" />
  <partnerLink name="loadTable" partnerLinkType=
    "lns:loadTableLT" myRole="tableLoader" />
  <partnerLink name="SQLScript" partnerLinkType=
   "lns:SQLScriptLT" myRole="tableLoader" />
  ...
</partnerLinks>
```

For instance, the task Load Cities.txt will be represented in BPEL using the partner link loadTxtFile. This partner link represents all ETL operations that load text files. It is of type loadTxtFileLT and is performed by the role fileLoader. The partner link will be invoked using the BPEL variable File with value Cities.txt and will return the variable TempTable with value TempCity.

6.2.3 Mapping BPMN Properties to BPEL Variables and Messages

In BPMN, properties are used to state useful information about the process. They have a similar role as BPEL variables. During the mapping, the value of BPMN properties is transferred to BPEL variables. Variables are defined at the head of the BPEL file. For example, the following code excerpt defines the variables tempTable, which keeps the values of temporary tables, and loadFault, which keeps information about eventual loading errors.

```
<variables>
  <variable name="tempTable"
    messageType="lns:tableMessage"/>
  <variable name="loadFault"
    messageType="lns:LoadErrorMessage" />
  ...
</variables>
```

6.2.4 Mapping BPMN Errors and Events to BPEL Fault Handling

This section defines the mechanisms for detecting and handling errors resulting from the invocation of services. We match the error end event in BPMN to the faultHandlers element in BPEL, which is used for interrupting a process. For example, in the code excerpt below, if an exception is detected during the sequence of services contained within the scope tag, the cannotCompleteLoading process is launched. This implements a similar behaviour as the BPMN workflow: if an error occurs during the Fill TempGeo.State subprocess, an error end event will be launched.

6.2.5 Mapping the BPMN Workflow to BPEL Process

The process section contains the description of the normal behavior of the activities that load the geography dimension.

```
<process name="LoadDimGeo" ... >
  ...
  <bpel:sequence>
    <bpel:flow>
      <bpel:invoke partnerLink="loadTxtFile"
        inputVariable="file" outputVariable="tempTable"/>
      <bpel:invoke partnerLink="loadTable"
        inputVariable="table" outputVariable="tempTable"/>
      <bpel:invoke partnerLink="SQLScript"
        inputVariable="database" outputVariable="tempTable"/>
    </bpel:flow>
    <bpel:scope>
      <faultHandlers>
        <catch faultName="lns:cannotCompleteLoading"
            faultVariable="lookupFault">
```

```
          <reply partnerLink= "lookupData"
            portType="lns:lookupDataPT"
            variable="lookupFault" faultName=
              "cannotCompleteLoading"/>
        </catch>
      </faultHandlers>
      <bpel:while>
        <condition>
          bpel:getVariableProperty("exists","row")>0
        </condition>
        <bpel:sequence>
          <bpel:invoke partnerLink ="loadRow"
           inputVariable="tempTable"/>
          ...
        </bpel:sequence>
      <bpel:while>
    </bpel:scope>
  </bpel:sequence>
  ...
</process>
```

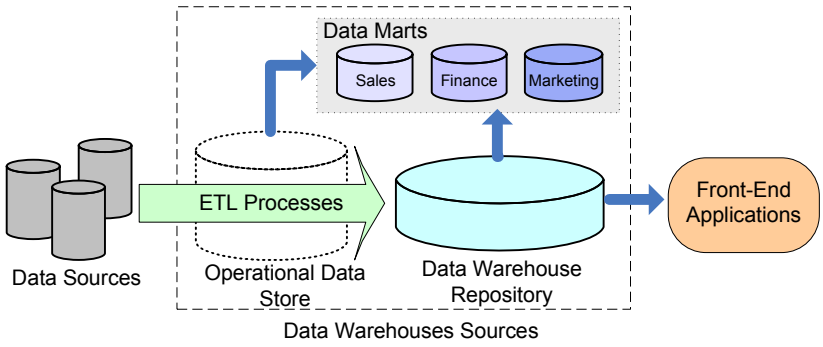
The above code excerpt emphasizes the general sequence of the process tasks. It shows the way to express parallelism of the first three activities using the flow construct. It also uses the catch exception inside a scope in order to launch the fault handling mechanism.

6.3 Service-Oriented ETL Architecture

We have reviewed the proposed approach that consists in incorporating the service-oriented standards into ETL process design. This approach has many positive consequences on the global ETL architecture. In fact, an interesting idea in our approach is to store separately, for reuse purpose, the implementation details of the workflows. It is possible to store them in any knowledge module library but in our approach we choose to beneficiate from the web service technology of the service-oriented architecture. For example, ETL task or workflow programs are published through interfaces as services that can be reused by other workflows. Several data warehouse components contribute in a party to the ETL process, hence in our approach each component will provide a set of ETL services related to its contribution.

In order to take great advantage of the proposed approach, we have elaborated a service-oriented ETL architecture that locates each ETL service inside the studied traditional data warehouse architecture in Section 2.4.

The figure 6.1 depicts the ETL services inside the three layers of the traditional data warehouse architecture: the source layer, the data warehouse layer and the front-end layer. We notice that at the source level a specific services may be provided of type verify and transfer such as Data Extract and Transfer Service or Delta Detection Service. Also some cleansing and transformations may be performed at the data source level as we have noted in Section 2.4, e.g. Data Cleanse Service. The second level of data warehouse sources, provided services that consist typically on assembling and transforming services. However, it can also include both source services and front-end services. Some examples of services



Layer	Source Layer	Data Warehouse Sources Layer	Front-End Layer
Type of Services	Source Services: Verify, Transfer	Consolidation Services: Transform, Assemble, Transfer	Analytics Services: Aggregate
Examples of Services	- Data Extract and Transfert Service - Delta Detection Service - Data Cleanse Service	- Data Cleanse Service - Dimension Surrogate - Data Decode Service Key Assignment Service - Data Merge Service - Data Load Service - Data New Derives Services	- Data Aggregate Service - Data Extract Service - Data OLAP Cube Service

Figure 6.1: Service-oriented ETL architecture.

at this layer can be Data Merge Service, Dimension Surrogate Key Assignment Service or Data Load Service which loads data to the front-end tools. Finally, the architecture depicts the front-end layer even if it stays optional in regard to the ETL process services. It is possible that some ETL tasks can be responsible of Data Aggregate Service or Data OLAP Cube Service. Further, in order to permit the development of new services the platform proposes the service Data New Derives Services and can be proposed in all the layers.

In this way, the ETL tasks and workflows details can be shared between multiple business rules within multiple ELT processes, either distributed or inside the same data warehouse platform. Also, this approach makes incremental changes very easy to perform. One possible perspective for this work is the use of an internal UDDI to publish ETL services to be profitable by all the data warehouse components. We assume that it is an evolved idea, since there is a sincere will to share knowledge inside an organisation and between collaborators, in contrary to the case of public UDDI where some commercial reasons restrict its progress.

Chapter 7

Conclusion

The Extraction, Transformation, and Load (ETL) process aims at extracting data from internal and external sources of an organization, transform this data, and load it into a data warehouse. It is well known that ETL processes are complex and costly. Therefore, it is necessary that these processes may be modelled using a conceptual language that, on the one hand, is easy to understood by end users and developers, and, on the other, can lead to their implementation into current tools. Further, current ETL tools propose specific languages for expressing such processes, which differ among tools and have different expressive power. As a consequence, there is no an agreed-upon way to specify ETL process at a conceptual level and that can be translated into executable specifications.

In this report we have presented a conceptual language for modelling ETL processes based on the Business Process Modelling Notation (BPMN), a de-facto standard for specifying business processes. Our model provides a large and useful set of primitives that cover the design requirements of frequently used ETL processes. In addition, this set of primitives can be extended to fit the requirements of particular applications.

There are several advantages of using BPMN for specifying ETL processes. One of them is that the language is already used for specifying business processes in general and thus, end-users must not learn another language for specifying ETL processes. Further, BPMN provides a conceptual and implementation-independent specification of such processes, which hides technical details and allows end users and designers to focus on essential characteristics of these processes. Finally, we have shown that ETL process expressed in BPMN may be translated into executable specifications using Business Process Execution Language (BPEL), a standard executable language for specifying interactions with web services.

There are several ways in which this work can be pursued. One of them is to study the translation of ETL processes expressed in BPMN into the specific languages proposed of current ETL tools in order to execute such processes. Another issue is to analyze the expressive power of our model using large real-world ETL scenarios.

Bibliography

[1] R. Bruckner, B. List, and J. Schiefer. Developing requirements for data warehouse systems with use cases. In *Proceedings of the 7th Americas' Conference on Information Systems, AMCIS'01*, pages 329–335, Boston, Massachusetts, USA, Aug. 2001.

[2] L. Cabibbo and R. Torlone. A logical approach to multidimensional databases. In H. Schek, F. Saltor, I. Ramos, and G. Alonso, editors, *Proceedings of the 6th International Conference on Extending Database Technology, EDBT'98*, LNCS 1377, pages 183–197, Valencia, Spain, Mar. 1998. Springer.

[3] M. Golfarelli, D. Maio, and S. Rizzi. Conceptual design of data warehouses from E/R schemes. In *Proceedings of the 31st Hawaii International Conference on System Sciences, HICSS-31*, pages 334–343, Kohala Coast, Hawaii, USA, Jan. 1998.

[4] M. Golfarelli and S. Rizzi. A methodological framework for data warehouse design. In I.-Y. Song and T. Teorey, editors, *Proceedings of the 1st ACM International Workshop on Data Warehousing and OLAP, DOLAP'98*, pages 3–9, Bethesda, Maryland, USA, Nov. 1998. ACM Press.

[5] M. Gyssens and L. Lakshmanan. A foundation for multi-dimensional databases. In M. Jarke, M. Carey, K. Dittrich, F. Lochovsky, P. Loucopoulos, and M. Jeusfeld, editors, *Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB'97*, pages 106–115, Athens, Breece, Aug. 1997. Morgan Kaufmann.

[6] B. Hüsemann, J. Lechtenbörger, and G. Vossen. Conceptual data warehouse design. In Jeusfeld et al. [9], page 6.

[7] W. Inmon. *Building the Operational Data Store*. Wiley, 1996.

[8] W. Inmon. *Building the Data Warehouse*. Wiley, 2002.

[9] M. Jeusfeld, H. Shu, M. Staudt, and G. Vossen, editors. *Proceedings of the 2nd International Workshop on Design and Management of Data Warehouses, DMDW'00*, Stockholm, Sweden, June 2000. CEUR Workshop Proceedings.

[10] R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses*. Wiley, 1998.

[11] B. List, J. Schiefer, and A. Min Tjoa. Process-oriented requirement analysis supporting the data warehouse design process: A use case driven approach. In M. Ibrahim, J. Küng, and N. Revell, editors, *Proceedings of the 11th International Conference on Database and Expert Systems Applications, DEXA'00*, LNCS 1873, pages 593–603, London, UK, Sept. 2000. Springer.

[12] S. Luján-Mora, P. Vassiliadis, and J. Trujillo. Data mapping diagrams for data warehouse design with UML. In P. Atzeni, W. Chu, H. Lu, S. Zhou, and T. Ling, editors, *Proceedings of the 23rd International Conference on Conceptual Modeling, ER'04*, LNCS 3288, pages 191–204, Shanghai, China, Nov. 2004. Springer.

[13] E. Malinowski and E. Zimányi. *From Conventional to Spatial and Temporal Applications*. Springer, 2008.

[14] B. Margolis, editor. *SOA for the Business Developer: Concepts, BPEL, and SCA*. MC Press, 2007.

[15] J. Mazón and J. Trujillo. An MDA approach for the development of data warehouses. *Decision Support Systems*, 45(1):41–58, 2008.

[16] D. Moody and M. Kortink. From enterprise models to dimensional models: A methodology for data warehouse and data mart design. In Jeusfeld et al. [9], page 5.

[17] OASIS. Web Services Business Process Execution Language Version 2.0. `http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf`, 2007.

[18] OMG. Business Process Modeling Notation (BPMN). `http://www.omg.org/docs/formal/09-01-03.pdf`, 2009.

[19] A. Simitsis. Mapping conceptual to logical models for ETL processes. In I. Song and J. Trujillo, editors, *Proceedings of the 8th ACM International Workshop on Data Warehousing and OLAP, DOLAP'05*, pages 67–76, Bremen, Germany, Nov. 2005. ACM Press.

[20] D. Skoutas and A. Simitsis. Designing ETL processes using semantic web technologies. In I. Song and P. Vassiliadis, editors, *Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP, DOLAP'06*, pages 67–74, Arlington, Virginia, USA, Nov. 2005. ACM Press.

[21] D. Skoutas and A. Simitsis. Ontology-based conceptual design of ETL processes for both structured and semi-structured data. *International Journal on Semantic Web and Information Systems*, 3(4):1–24, 2007.

[22] D. Skoutas, A. Simitsis, and T. Sellis. Ontology-driven conceptual design of ETL processes using graph transformations. In *Journal on Data Semantics XIII*, number 5530 in LNCS, pages 122–149. Springer, 2009.

[23] N. Tryfona, F. Busborg, and J. Christiansen. starER: A conceptual model for data warehouse design. In I.-Y. Song and T. Teorey, editors, *Proceedings of the 2nd ACM International Workshop on Data Warehousing and OLAP, DOLAP'99*, pages 3–8, Kansas City, Missouri, USA, Nov. 1999. ACM Press.

[24] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos. Conceptual modeling for ETL processes. In D. Theodoratos, editor, *Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP, DOLAP'02*, pages 14–21, McLean, Virginia, USA, Nov. 2002. ACM Press.

[25] M. Wu and A. Buchman. Research issues in data warehousing. In *Proceedings of Datenbanksysteme in Büro, Technik und Wissenschaft, BTW'97*, pages 61–82. Springer, Mar. 1997.