CONCEPTUAL MODELING FOR ETL PROCESS

A Thesis
Presented to the
Faculty of
California State Polytechnic University, Pomona

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science
In
Computer Science

By
Jijun Wang
2007
THESIS: CONCEPTUAL MODELING FOR ETL PROCESS

AUTHOR: Jijun Wang

DATE SUBMITTED: Fall 2007

Department of Computer Science

Dr. Daisy F. Sang
Thesis Committee Chair
Computer Science

Dr. H. K. Liu
Computer Science

Dr. Gilbert Young
Computer Science
Acknowledgement

I wish to thank Dr. Daisy F. Sang, my advisor, for her patience, guidance, and valuable inputs throughout the completion of this thesis. Language is not enough to express my appreciation to her support and encouragement to my thesis project that lasts for so long a period. I would like to express my gratitude to Dr. H. K. Liu and Dr. Gilbert Young for going over the drafts and providing me with invaluable suggestions.

Lastly and most importantly, words cannot fully describe my sincere and deepest gratitude to my family – my wife and my son, for their love, continual support, and endless encouragement. Neither can words truly express my heartfelt appreciation to my parents, who were both professors in the field of Computer Science and who led me into this line and always encouraged me to pursue more and higher in my career life, to them I dedicate this effort.
Abstract

A data warehouse is a copy of transaction data specifically structured for query, reporting, and analysis. Data warehouse makes use of an ETL process, which stands for Extraction – Transformation – Loading, to deal with data cleaning and loading. ETL is a consolidation process that involves retrieving data from many sources, transforming it to meet business needs, and ultimately loading into a data warehouse. In our practical example, ETL process extracts data from human resource system, financial system, and other text or Excel data sources, transform it, and populates the data to Manpower Data Warehouse. Building ETL is the most labor-intensive and lengthy procedure in data warehouse implementation, covering up to thirty to eighty percent of effort and expenses [Vassiliadis et al. 2002]. Though ETL is a software system and plays such an important role in data warehouse construction, it is often treated just as a special ad hoc process set running on an ETL tool, or a group of custom programs developed without serious plan and design. Nowadays, these problems are realized and some formal methodologies for ETL have been proposed. However, these theories or methods are still in laboratory phase, without being popularly applied in practice.

Since ETL process plays such an important role in data warehouse building, a well-modeled and robustly designed ETL process is required. In this research, we explored the conceptual model of ETL process proposed by Panos Vassiliadis and applied the conceptual model to an actual ETL process in the Manpower Data Warehouse project. Upon the implementation experience, we strongly underwent the advantages brought by the formal modeling methodology such as maintainability, reuse, and extendibility. However, we also discovered some places in the notation needing improvement. We found that existing notation symbols are awkward in denoting some practical situations or logics from practical projects. Based on these discoveries we suggested some extended notation symbols for the ETL conceptual model. These notation
extensions help to express graphically the timing order or sequence logic in ETL conceptual model and to remove ambiguity in diagram notes. Using the extended notation, we presented the ETL conceptual model of our practical example more clearly and concisely.
# TABLE OF CONTENTS

Signature Page .............................................................................................................................................. ii

Acknowledgement ........................................................................................................................................ iii

Abstract ......................................................................................................................................................... iv

Chapter 1 Introduction ................................................................................................................................ 1

1.1 Challenges and Problems in ETL ........................................................................................................... 2

1.2 Data model, Data Modeling, and Data Model Theory ........................................................................... 4

1.3 Research Goal ......................................................................................................................................... 6

Chapter 2 Background ............................................................................................................................... 8

2.1 The Development of ETL Technique ..................................................................................................... 8

2.2 Conceptual Data Modeling Approaches for ETL Processes ................................................................. 10

Chapter 3 ETL Conceptual Model ............................................................................................................... 15

3.1 Notation for ETL Activities .................................................................................................................... 15

3.2 Model Components ............................................................................................................................... 17

  3.2.1 Concepts and Attributes .................................................................................................................. 17

  3.2.2 Transformations, Constraints, and Notes ....................................................................................... 18

  3.2.3 Part-Of and Candidate Relationships ........................................................................................... 20

  3.2.4 Provider Relationships and Serial Composition of Transformations ........................................... 22

3.3 Model Structure: Instantiation and Specialization Layers .................................................................... 25

  3.3.1 Metadata Layer ............................................................................................................................... 26

  3.3.2 Template Layer ............................................................................................................................... 27

  3.3.3 Implementation of Template Layer ............................................................................................... 27

3.4 ETL conceptual model, ETL logical Model, and Implementation ......................................................... 30

  3.4.1 ETL Conceptual Model and ETL Logical Model .......................................................................... 30

  3.4.2 ETL Logical Model and Implementation ....................................................................................... 31
Chapter 4 Implementation of ETL Conceptual Model ......................................................... 33

4.1 Requirements of Manpower Data Warehouse and its ETL process .............................. 34

4.2 Conceptual Model for ETL process in Manpower Data warehouse .......................... 39

4.3 Logical Model for ETL process in Manpower Data Warehouse ............................... 42

4.4 ETL Process Implementation in Cognos DecisionStream ....................................... 45

4.5 Analysis and Comparison of ETL Implementation ...................................................... 50

4.6 ETL Implementation Summary ................................................................................ 51

Chapter 5 Analysis and Extension of ETL Conceptual Model ........................................ 52

5.1 ETL Conceptual Model Analysis .............................................................................. 52

5.1.1 Execution Orders or Timing .................................................................................. 52

5.1.2 Special Transformation ......................................................................................... 54

5.1.3 Schedule, Logging, Trace, and Recovery .............................................................. 56

5.2 ETL Conceptual Model Extension ............................................................................ 57

5.2.1 Execution Orders or Timing .................................................................................. 57

5.2.2 Special Transformation ......................................................................................... 61

5.2.3 Schedule, Logging, Trace, and Recovery .............................................................. 63

Chapter 6 Result Analysis and Comparison .................................................................... 66

6.1 Analysis and Comparison .......................................................................................... 66

Chapter 7 Conclusions .................................................................................................... 69

References ...................................................................................................................... 71
LIST OF FIGURES

Figure 3.1 Notation for the conceptual modeling of ETL activities.................................15
Figure 3.2 Part of ETL conceptual model for Manpower Data Warehouse.........................16
Figure 3.3 The ETL conceptual model with three layers..................................................26
Figure 4.1 Manpower Data Warehouse architecture.........................................................35
Figure 4.2 Manpower Data Warehouse ETL Conceptual Model........................................40
Figure 4.3 Manpower Data Warehouse ETL Logical Model..............................................43
Figure 4.4 Manpower Data Warehouse ETL Implementation - Fact Build 1.......................45
Figure 4.5 Manpower Data Warehouse ETL Implementation - Fact Build 2.......................46
Figure 4.6 Manpower Data Warehouse ETL Implementation - Fact Build 3.......................47
Figure 4.7 Manpower Data Warehouse ETL Implementation - Fact Build 4.......................47
Figure 4.8 Manpower Data Warehouse ETL Implementation – Job Build..........................48
Figure 4.9 Manpower Data Warehouse ETL Implementation – Database Connections.......49
Figure 4.10 Manpower Data Warehouse ETL Implementation – Transformation Functions....49
Figure 4.11 Manpower Data Warehouse ETL Implementation - Dimensions......................50
Figure 5.1 Timing Order Case 1.........................................................................................53
Figure 5.2 Special Transformation case 1...........................................................................55
Figure 5.3 UML sequence diagram....................................................................................58
Figure 5.4 Timing order diagram for ETL model.................................................................60
Figure 5.5 ETL conceptual model with time-division lines for Manpower Data Warehouse....61
Figure 5.6 Self-transformation diagram for ETL conceptual model.....................................62
Figure 5.7 ETL conceptual model with Self-transformation notation...................................63
Figure 5.8 New notes notation for summary notes and detail notes.................................64
Figure 5.9 The Manpower Data Warehouse ETL model with extended notation...............65
# LIST OF TABLES

Table 3.1 Transformation Classes................................................................................................ 29
Table 4.1 Data source and schema for Manpower Data Warehouse............................................37
Table 4.2 Fact and dimension tables in Manpower Data Warehouse ........................................38
Table 6.1 Comparison of extended notation and original notation ..............................................68
Chapter 1 Introduction

A data warehouse is the main repository of an organization's historical data, a corporate memory. For example, an organization would use the information that is stored in its data warehouse to find out what day of the week they sold the most products in May 2005, or how employee sick leave in the week before Christmas differed between California and Quebec from 2001-2005. In other words, the data warehouse contains the raw or historical material for management's decision support system.

During late 1990s, many enterprises came to recognize that the data they had at their disposal is an important asset that, if properly leveraged, can give them competitive advantages. Since then, many enterprises have created data warehouses (or data marts) in order to make quick reports or analysis to support their decision-making processes. They invested heavily, ran applications against them, and even re-engineered their business processes. However, the outcome was not ideal. Data warehouse projects had a high failure rate, which was over 50% based on a Gartner study [Sen, Sinha, and Ramamurthy 2006]. Among the major reasons, responsible for the high failure rate of data warehouse projects, is underestimating the complexity of an ETL process and the required time and labor to build it.

ETL (Extract, transform, and load) is a process that involves extracting data from outside sources, transforming it to fit business needs, and ultimately loading it into the data warehouse. Except these literal functionalities, data cleaning, profiling, and quality control are also considered part of ETL.
1.1 Challenges and Problems in ETL

In a complex corporate computing environment, ETL can be as complicated as an overall integration solution or even much more. It connects to many different operating systems and database platforms. It retrieves data from many different enterprise software systems, such as human resource system, financial system, and customer relationship system. Among these application systems, some may be still legacy systems, which are desperate to others, and the business terminology, coding, or information structure are not standardized or unified. The data may be in different date format, different currencies, or different granular levels. ETL needs to connect to all these quite different application systems, take diverse data, transform them into a unique form, and load into data warehouse. In other words, ETL bring heterogeneous and asynchronous source extracts to a homogeneous environment.

In addition, the data storage and schema in a data warehouse are different from those in source databases from where the data in data warehouse comes. This is because a data warehouse should be optimized for Online Analytical Processing (OLAP). Thus, data in data warehouses is heavily de-normalized, summarized, and stored in a star schema. The transformation from one storage and schema to another is one of ETL’s many tasks.

The complexity of ETL also results from the giant amount of data it needs to process. Though the data in a data warehouse can be summarized or consolidated, it is aggregated from the giant amount of detail data from various enterprise applications by ETL. In other words, ETL still needs to retrieve and process large amount of data.
The importance of a data warehouse, such as its support to a corporation’s decision making, even places to ETL higher system requirements in integrity, reliability, extendibility, maintainability, performance, reuse, and data quality, than other enterprise software systems.

ETL system should be easier to extend or to maintain. It is the most dynamic system in a corporation environment. The change or update requirements comes from both ends of ETL: data sources and data warehouse. Since ETL retrieves data from so many enterprise software systems, any change or upgrade to any one of these systems brings subsequent changes to ETL. In another end, the executives’ need to data from a data warehouse fluctuates frequently based on the requirements for decision-making. While it’s impossible to have all kinds of data prepared in the data warehouse, the only way is to update the ETL to retrieve and load new data.

ETL should not feed wrong, conflicted, or miscalculated data into the data warehouse. ETL should work reliably so that the data in data warehouse can be refreshed or updated on time. Since data warehouse has bigger impacts to a corporation’s operation, the damage to a corporation is also much bigger when ETL fails or makes errors.

However, the existing ETL systems are either special ad hoc process sets running on an ETL engine, or groups of custom programs developed without serious plan and design. The processes or custom programs are accumulated along a long period and written by different developers, even without design documents. When a new requirement comes, for example, an executive need some type of data from data warehouse, a developer wrote a piece of ETL program or a single process on a ETL engine to retrieve the required data from data sources and put into data warehouse. When another new requirement arises, another piece of program is created and added. This sounds quick and simple. However, seldom the integrity, reliability, extendibility, maintainability, and reuse of the overall ETL system, are concerned and considered. When the
bundle of program or process grow bigger, many challenges and problems occur. For instance, when a maintenance is requested, one needs to browse and read all the related programs developed by different developers or in different programming languages. If he or she doesn’t make a perfect maintenance, this leaves openings for poor integrity and reliability. The data quality is quite possibly affected.

In recent years, many researchers and professionals realized these problems and challenges in ETL. One of their efforts is to bring formal data modeling methodologies to ETL process.

1.2 Data model, Data Modeling, and Data Model Theory

A data model is an abstract model that describes how data is represented and used. In relational database, a data model describes structured data for storage in data management systems. A data model theory is a formal data model description. Some examples of data model theory are entity-relationship model (ERM) [Thalheim 2000] and object model [Coad and North and Mayfield 1996]. A data model instance is a data model specialized to an enterprise application such as financial system or human resource management system.

Data modeling is the process of creating a data model instance by applying a data model theory. The activities in data modeling include the analysis of data objects that are used in a business or other context, and the identification of the relationships among these data objects.

When data modeling, we are structuring and organizing data. These data structures are then typically implemented in a database management system. In addition to defining and organizing the data, data modeling will impose constraints or limitations on the data placed within the structure.
A data model instance may be one of a conceptual data model, a logical data model, and a physical data model. A conceptual data model describes the semantics of an organization. This consists of entity classes (representing things of significance to the organization) and relationships (assertions about associations between pairs of entity classes). A logical data model describes the semantics, as represented by a particular data manipulation technology. This consists of descriptions of tables and columns, object oriented classes, and XML tags, among other things. A physical data model describes the physical means by which data are stored. This is concerned with partitions, CPUs, tablespaces, and the like.

The significance of this approach, according to ANSI, is that it allows the three perspectives to be relatively independent of each other. Storage technology can change without affecting either the logical or the conceptual model. The table/column structure can change without necessarily affecting the conceptual model.

In addition, the three model structure follows the life cycle of software development and software methodology of top-down design. While early phases of software projects emphasize the design of a conceptual data model, in later stages, such a design can be detailed into a logical data model, which can be then translated into physical data model.

With significance in abstraction (expressing a concept in its minimum, most universal set of properties), transparency (being intuitively clear and understandable), and effectiveness (accurate and reliable), data modeling organizes data in a way that is flexible and adaptable to unanticipated business requirements.
1.3 Research Goal

Modern corporations and other organizations realized the importance of data warehouse. They invested heavily in the building of data warehouse to support their decision-making. However, data warehouse projects face a variety of challenges, many of which come from building process of Extraction – Transformation - Loading (ETL). ETL is the most labor-intensive and lengthy procedure in data warehouse implementation. It is much more complex than what corporates have assumed. The quality requirements applied to it, such as integrity, reliability, and performance, are much higher than what corporates have planned or targeted. The underestimate towards to the complexity and targets of the ETL process makes it being treated as a casual process running on an ETL tool or pieces of custom programs developed without serious plan and design. This brings a series of problems such as big maintenance cost, low process reliability, project delay.

Realizing these problems and challenges, researchers are introducing formal modeling methodologies into the analysis and design of ETL process. They borrow the models from other model theories such as object model, or create the model theory specialized for the domain of ETL. ETL conceptual model proposed by Panos Vassiliadis is the pioneering and first conceptual model specially designed for the process of ETL. It’s simple, structured, and well designed to cover all ETL activities, having many advantages over those borrowed. It helps the revision and redefinition of original ETL assumptions and mappings and serves as a communication tool between users and ETL modelers. It’s an important part of formal ETL model. However it’s still in a theory or laboratory phase, not popularly used in actual practices. The early and quick application and adaptation of this study to practical projects are significant to the success of data warehouse or ETL projects.
The goal of this research is to apply the ETL conceptual model proposed by Panos Vassiliadis to an actual ETL process in a manpower data warehouse project and to seek improvement or extension towards to any withdraws or weaknesses found during the model implementation so that the model can be used in industry. Based on the implementation, we analyze and compare the advantages brought by the ETL conceptual model in the fields such as maintainability, reuse, and extendibility. For any weaknesses of the model discovered during the implementation, we seek improvement and extend the ETL conceptual model to address these flaws so that it can soon benefit practical projects.
Chapter 2 Background

In this chapter, first, we investigate and explore the development of ETL process and technique. Then, we look at the conceptual modeling for ETL process. We identify the uniqueness of the ETL conceptual model proposed by Panos Vassiliadis. It is pioneering in the field of ETL modeling though there are many conceptual modeling methodologies of data warehouse.

2.1 The Development of ETL Technique

As computer systems started to evolve from monolithic mainframes to distributed computing systems, and as business intelligence made its debut, the first ETL solutions were introduced. Since that time, several generations of ETL systems have been produced.

Original ETL systems consisted of legacy code or program manually written or generated by tools. They were in native code for the operating system of the platform on which the data integration processes were to run. Most of them were actually in COBOL, since at that time data was largely stored on mainframes. The data integration processes were easier than they had been by taking advantage of a centralized ETL process. Performance was very good because of the inherent performance of native compiled code, but these ETL programs required an in-depth knowledge of programming on the different platforms. Maintenance was also difficult because the code was disseminated to different platforms and differed with the type of source and target. At the time, this ETL architecture provided the best possible performance, since data was stored in flat files and hierarchical databases and record-level access was fast. Although this worked well on mainframes, using such an approach on relational databases has proven to be less successful for managing large data volumes.
Next came the second generation of ETL system because of ETL tools, which are based on a proprietary engine (sitting between sources and the data warehouse target) that runs all the transformation processes. This approach solved the problem of having to use different languages on different platforms, and required expertise in only one programming language: the language of the ETL tool itself. However, a new problem arose: the proprietary engine performing all the transformations became a bottleneck in the transformation process. All data, coming from various sources to go to the target, had to pass through the engine that processed data transformations row by row. Row by row is very slow when dealing with significant volumes of data.

The next generation of ETL system with the E-LT (Extract - Load & Transform) Architecture addresses the challenges from the previous two generations while leveraging their respective strengths. The ETL tool is to generate and execute highly optimized data integration processes, in the native SQL or in the other languages of the databases involved in these processes.

Some trends in ETL technique are that (1) ETL tools evolve towards generic data integration tools, (2) ETL tools is linked with data quality tools, and (3) ETL is required with Lower latency requirement.

Currently ETL system and technique for most enterprises are still combinations of ETL bundle of programs and ETL engine. The development of ETL technique brought many ETL tools, which help a lot in the coding and implementation phase of ETL building. The development in ETL analysis and design is slow. We will discuss this in detail in next section.
2.2 Conceptual Data Modeling Approaches for ETL Processes

Conceptual Data Model (CDM) represents the overall logical structure of a database or a data processing system, which is independent of any software or data storage structure. A conceptual model often contains data objects not yet implemented in the physical databases. It gives a formal representation of the data needed to run an enterprise or a business activity. A conceptual model tends to describe concepts of the real world, rather than the modalities for representing them in a computer.

For example, if presented in Unified Modeling Language (UML) [Rational Partners 1997], a conceptual data model for a database application includes use-case diagram, class diagram, and data dictionary. The use-case diagram states the use-cases in terms of primary actors pursuing high-level goals, while the class diagram represents all the classes/entities in your application, their attributes, and the associations/relationships between them. The data dictionary, otherwise, includes columns for class, attribute, data type, and description.

There are many conceptual data models, which exist with different features and expressive powers, mainly depending on the application domain. In the context of data warehousing, traditional conceptual models for database modeling, such as the Entity-Relationship (E-R) model [Thalheim 2000], do not provide a suitable means to describe the fundamental aspects of such applications. The crucial point is that in designing a data warehouse, there is the need to represent explicitly certain important characteristics of the data contained therein. As we have mentioned in prior section, these characteristics of data, such as star schema, are not related to the abstract representation of real world concepts, but rather to the final goal of the data warehouse: supporting data analysis oriented to decision making. More specifically, it is widely recognized that there are at least two specific notions that any conceptual data model for data
warehousing should include: the fact (or its usual representation, the data cube) and the dimension. A fact is an entity of an application that is the subject of decision-oriented analysis and is represented graphically by means of data cubes. A dimension corresponds to a perspective under which facts can be analyzed.

The conceptual model for data warehousing can be either a specially designed conceptual model for data warehouse or a borrowed conceptual model from other application domain but amended with the special characteristics of data warehouse. Research efforts in conceptual model of data warehouse can be sorted as following four major trends [Vassiliadis et al. 2002], including: (a) dimensional modeling, (b) (extensions of) standard E/R modeling, (c) UML modeling and (d) sui-generis models.

The supporters of the dimensional modeling method argue that the model is characterized by its minimality, understandability (especially by the end-users), and its direct mapping to logical structures. The sponsors of the E/R and UML methods models base their arguments on the popularity of the respective models and the available semantic foundations for the well-formedness of data warehouse conceptual schemata. The sui-generis models are empowered by their novelty and adaptation to the particularities of the OLAP setting.

However, these conceptual modeling methodologies of data warehouse focus on capturing of the conceptual characteristics of the star schema, the subsequent data marts, and aggregations. The ETL conceptual modeling needs to aims towards the mapping of the attributes of the data sources to the attributes of the data warehouse tables. More specifically, the conceptual part of the definition of the ETL process deals with the earliest stages of the data warehouse design. During this period, the data warehouse designer is concerned with two tasks. The first of these tasks involves the collection of requirements from the users. The second task, which is of equal
importance for the success of data warehousing project, involves the analysis of the structure and content of the existing data sources and their intentional mapping to the data warehouse.

Upon these considerations, Panos Vassiliadis proposed the first formal conceptual model specialized for ETL process [Vassiliadis, Simitsis, and Skiadopoulos 2002]. His conceptual modeling method is different from the four major trends mentioned above. His conceptual model is with a particular focus on the interrelationships of attributes and concepts and the necessary transformations that need to take place when loading the warehouse. Transformations are employed as an official term for the restructuring of schema and values or for the selection and even the transformation of data. Attribute interrelationships are caught through provider relationships that map data provider attributes at the sources to data in the warehouse. Except for these fundamental relationships, the proposed model is able to capture constraints and transformation composition, too.

The conceptual model is also presented in a set of design steps, which lead to the basic target, i.e., the attribute interrelationships. These steps constitute the methodology for the design of the conceptual part of the overall ETL process.

In addition, different instantiation and specialization layers characterize this conceptual model. The generic metamodel proposed involves a small set of generic constructs that are powerful enough to capture all cases. These entities the metamodel layer in the architecture. Moreover, it introduced a specialization mechanism to allow the construction of a template of frequently used ETL activities. This set of ETL-specific constructs, constitute a subset of the larger metamodel layer, which is called the Template Layer. The constructs in the Template layer are also metaclasses, but they are quite customized for the regular cases of ETL processes. All the entities (data stores, inter-attribute mappings, transformations) that an ETL model designer uses in his
particular scenario are instances of the entities in the metamodel layer. For the common ETL transformations, the employed instances correspond to the entities of the template layer.

There is also study on the ETL conceptual modeling using (borrowing) the standard object model theory such as UML [Trujillo 2003]. Though the study stated that “The main differences, and advantages, of our approach in contrast to Vassiliadis’ are the use of a standard modeling language (UML)”, just the using of UML brings many disadvantages and limits to this approach.

Vassiliadis’ notation is simple and easy to understand, also powerful and efficient; however, UML looks complicated and awkward when used on ETL. The most important part in ETL is the attribute mapping and conversion. This is why in Vassiliadis’ the attribute is treated as ‘first class citizens”. In Trujillo’s, because of the limit of UML, ETL conceptual model is presented as a class diagram because “an ETL model can become exceedingly complex if every attribute is individually represented as a model element [Trujillo 2003]”. In addition, when the coding or implementation is not done in an object-oriented way, Object-Oriented Design (OOD) complicated the implementation process.

The ETL conceptual model is constructed in the early stages of the data warehouse project. During this phase, there is a lot of communication between the modelers and users, and many revisions and redefinitions of original assumptions and mappings. A simple and specialized conceptual modeling can serve these tasks better.

In this research, we will explore and implement this ETL conceptual modeling proposed by Panos Vassiliadis. In following chapters, when we refer to the ETL conceptual model, we mean the special conceptual model suggested by Panos Vassiliadis for ETL process.
The ETL conceptual model (the conceptual model proposed by Panos Vassiliadis for ETL process) employs an architecture of Metamodel layer, Template layer, and Schema layer. Metamodel layer holds a small set of generic constructs such as concepts and attributes, while template layer has a larger set of ETL-specific constructs. The schema layer consists of all the instances corresponding to the entities of the template layer.

The ETL conceptual model employs transformations for the restructuring of schema and values. It uses provider relationships to map data provider attributes at the sources to data consumers in the warehouse, and uses concept and attributes to describe data sources.
Chapter 3 ETL Conceptual Model

In this chapter, we discuss and analyze the ETL conceptual modeling proposed by Panos Vassiliadis in detail. First, we look at its abstraction of various ETL components and activities: concepts, attributes, relationships, constraints, transformations, and the three-layer instantiation structure. We explain and exemplify these abstract concepts with examples from the Manpower Data Warehouse project. Next, we explore the connections between ETL conceptual model and ETL logical model and their relationship to the ETL process building.

3.1 Notation for ETL Activities

In figure 3.1, we graphically depict the different entities of the proposed model. Basically this notation includes Concept, Attribute, Note, Transformation, Constraint, Provider, Serial Composition, Part of, XOR, and Provider N:M.

![Figure 3.1 Notation for the conceptual modeling of ETL activities](image-url)
In this model, standard UML notation for concepts and attributes are not directly employed because of the intention to stress the importance of attributes. Thus, the attributes are not embedded in the definition of their encompassing entity, such as a UML class or a relational table. Note symbol is provided to communicate any information, which can not be denoted through the graphic symbols.

Following figure 3.2 depicts part of an ETL Conceptual Model designed in the above notation. For example, concept and attribute symbols are used to denote the table O1.JOB, and XOR symbol is used to show the candidate relationship between O1.HISTORYJOB and O1.RECENTJOB. In next section, we explain all notation symbols in a detail.

Figure 3.2 Part of ETL conceptual model for Manpower Data Warehouse
3.2 Model Components

3.2.1 Concepts and Attributes

Attributes is a granular module of information. It is part of a set of information, which represents an entity, such as a field in a database table, or a property in an object. The role of attributes is the same as in the standard ER/dimensional models. As in standard ER modeling, attributes are depicted with oval shapes.

Concepts represent an entity in the source databases or in the data warehouse. Concept instances can be source data files, data warehouse fact and dimension tables, and so on. A concept is defined formally by a name and a finite set of attributes. In terms of the ER model, a concept is a generalization of entities and relationships; depending on the employed model (dimensional model or ER extension) all entities composed of a set of attributes are generally instances of class Concept.

Many physical storage structures can be treated as finite lists of fields, including relational databases, COBOL or simple ASCII files, multidimensional cubes, and dimensions. Concepts are fully capable of modeling this kind of structures. Employing simply concepts is sufficient for the problem of ETL modeling. As an option, the generic concept structure can be refined to subclasses pertaining to the characteristics of the aforementioned approaches (e.g., classes Fact Table and Dimension), achieving thus a homogeneous way to treat OLAP and ETL modeling.
In Manpower Data Warehouse, one can observe several concepts. The concepts, O1.JOB and O1.Personal Data, are depicted in Figure 2.2, along with their respective attributes.

3.2.2 Transformations, Constraints, and Notes

Transformations are abstractions that represent parts, or full modules of code, executing a single task, in the framework. Two major categories of transformations include (a) filtering or data cleaning operations, like the check for primary or foreign key violations and (b) transformation operations, through which the schema of the incoming data is transformed (e.g., aggregation).

Formally, a transformation is defined by (a) a finite set of input attributes; (b) a finite set of output attributes and (c) a symbol that graphically characterizes the nature of the transformation. A transformation is graphically depicted as a hexagon tagged with its corresponding symbol.

In Manpower Data Warehouse of Figure 2.2, one can observe several transformations. Noting the ones pertinent to the mapping of O1.JOB to DSA.HC Actuals, one can observe a surrogate key assignment transformation (SK), a function application calculating the system date (D), and a Search and Replace transformation (S) for attribute Department.
ETL Constraints express the fact that the data of a certain concept fulfill several requirements. For example, the designer might wish to impose a primary key or null value constraint over a (set of) attribute(s). This is achieved through the application of ETL constraints, which are formally defined as follows: (a) a finite set of attributes, over which the constraint is imposed and (b) a single transformation, which implements the enforcement of the constraint.

Despite the similarity in the name, ETL constraints are different modeling elements from the well known UML constraints. An ETL constraint is graphically depicted as a set of solid edges starting from the involved attributes and targeting the facilitator transformation. In Manpower Data Warehouse, observe that we apply a Primary Key ETL constraint to MDW.HC and Salary Fact for the attributes Period, Org, Jobtype, Employee Type, and Overtime.

ETL_Constraint

Notes, exactly as in UML modeling, notes are informal tags to capture extra comments that the designer wishes to make during the design phase or render UML constraints attached to an element or set of elements. As in UML, notes are depicted as rectangles with a dog-eared corner.

In this framework, notes are used for simple comments explaining design decisions and explanations of the semantics of the applied transformations, for example, in the case of relational selections/joins this involves the specification of the respective selection/join condition, whereas in the case of functions this would involve the specification of the function signatures. They are also used for tracing of runtime constraints that range over different aspects.
of the ETL process, such as the time/event based scheduling, monitoring, logging, error handling, and crash recovery.

For example, in Manpower Data Warehouse, we can observe a runtime constraint specifying that the overall execution time, for the loading of MDW.HC, Salay Fact, and MDW.Roster Fact, must be within certain time.

### 3.2.3 Part-Of and Candidate Relationships

Part-of Relationships emphasize the relationship of concept and attributes that a concept is composed of a set of attributes. In general, standard ER modeling does not treat this kind of relationship as a first-class citizen of the model; UML modeling on the other hand, hides attributes inside classes and treats part-of relationships with a much broader meaning. The relationship of a concept with its attributes is emphasized in this model rather than redefining UML part-of relationships since attributes as first class citizens in the inter-attribute mappings are needed. Naturally, it is not precluded to use the part-of relationship for other purposes, as in standard UML modeling. In this model, a part-of relationship is denoted by an edge with a small diamond at the side of the container object.

![Part of Relationship](image)

**Part of Relationship**

Candidate relationship is used to express the situation that more than one candidate source files/tables could populate a target, data warehouse table. Thus, a set of candidate relationships captures the fact that a certain data warehouse concept can be populated by more than one
candidate source concepts. Formally, a candidate relationship comprises (a) a single candidate concept and (b) a single target concept. Candidate relationships are depicted with bold dotted lines between the candidates and the target concepts. Whenever exactly one of them can be selected, the set of candidate relationships is annotated for the same concept with a UML \{XOR\} constraint.

Active candidate relationship denotes the fact that out of a set of candidates, a certain one has been selected for the population of the target concept. Thus, an active candidate relationship is a specialization of candidate relationships, with the same structure and refined semantics. An active candidate relationship is denoted with a directed bold dotted arrow from the provider towards the target concept.

In Manpower Data Warehouse, there are O1.HistoryJob and O1.RecentJob, which are candidates for O1.Job. Concept HistoryJob contains the full annual history until the last month about Job. It is used for reporting purposes. Concept RecentJob contains only the data of the current month. It is used online by end-users for the insertion or update of data as well as for some reporting applications. The diagram also shows that RecentJob is eventually selected as the active candidate.
3.2.4 Provider Relationships and Serial Composition of Transformations

Provider relationships map a set of input attributes to a set of output attributes through a relevant transformation. In the simple 1:1 case, provider relationships capture the fact that an input attribute in the source side populates an output attribute in the data warehouse side.

If the source and target attributes are compatible semantically and physically, no transformation is required. If this is not the case though, we pass this mapping through the appropriate transformation (e.g., European to American data format and not null check). The formal definition of provider relationships comprises (a) a finite set of input attributes; (b) a finite set of output attributes; (c) an appropriate transformation (i.e., one whose input and output attributes can be mapped one to one to the respective attributes of the relationship).

In the case of N:M relationships the graphical representation can not express the linkage between provider and target attributes perfectly. To overcome for this shortcoming, we annotate the link of a provider relationship with each of the involved attributes with a tag, so that there is no ambiguity for the actual provider of a target attribute.

In the 1:1 case, a provider relationship is depicted with a solid bold directed arrow from the input towards the output attribute, tagged with the participating transformation. In the general N:M case, a provider relationship is graphically depicted as a set of solid arrows starting from the providers and targeting the consumers, all passing through the facilitator transformation.

Finally, a syntactic add-on to the model should also be mentioned. When a certain provider relationship involves all the attributes of a set of concepts, for example, in the case of a relational
union operation, all the attributes of the input and the output concepts would participate in the transformation.

*Provider 1:1*

![Diagram](image)

*Provider N:M*

To avoid overloading the diagram with too many relationships, a syntactic notation is employed to map the input to the output concepts instead of attributes. This can also be treated as a zoom in/out operation on the diagram per se: at the coarse level, only concepts are depicted and an overview of the model is given; at the detailed level, the inter-concept relationships are expanded to the respective inter-attribute relationships, presenting the designer with all the available detail. These relationships can be examined in Manpower Data Warehouse. The relationship between the attributes of concepts O1.HC Actual and MDW.Roster Fact is an inter-concept relationship.

In Manpower Data Warehouse, the data in SDS.HC Actuals is from O1.Job, O1.Employment, O1.Personal_data, O1.Manpower_Hierarchy, and O1.All_chk_hrs_em.

Attribute SEQNO is produced via Surrogate Key transformation function. Surrogate Key assignment is common tactics in data warehousing, employed in order to replace the keys of the production systems with a uniform key. For example, it could be the case that the employee
‘John’ has PKEY=30 for source 1, PKEY=40 for source 2, while at the same time source 2 has PKEY=30 for employee ‘Peter’. These conflicts can be easily resolved by a global replacement mechanism through the assignment of a uniform surrogate key.

Attribute Department is populated from its homonymous attributes in the sources, with the need for search transformation. Attribute Name is populated from its homonymous attributes in the sources. As far as source O1 is concerned, we need a name transformation in order to convert the multiple fields to one single value. As far as source O1 is concerned, many key fields need applying a Not Null (NN) check, to avoid loading the data warehouse with rows having null value.

Note also that there can be input attribute, such as O1.Job.Department, which is ignored during the ETL process.

Transformation Serial Composition depicts the need to combine several transformations in a single provider relationship. For example, it’s possible to group incoming data with respect to a set of attributes, having ensured at the same time that no null values are involved in this operation. In this case, it would need to perform a not null check for each of the attributes and propagate only the correct rows to the aggregation. In order to model this setting, a serial composition of the involved transformations is employed.

Formally, a serial transformation composition comprises (a) a single initiating transformation and (b) a single subsequent transformation. Serial transformation compositions are depicted as solid bold lines connecting the two involved transformations.
Serial Composition

A rather complex part of Manpower Data Warehouse is the aggregation that takes place for the rows of source DSA. Practically, DSA.HC Actuals captures information for employees according to the particular department of the organization. Loading this data to the data warehouse that ignores this kind of detail requires the aggregation of data per Month Year, Department, various types and the summation of FTE and Head count. This is performed from the aggregation transformation S. Still, all the aforementioned elementary transformations are not ignored or banished. On the contrary, they are linked to the aggregation Transformation through the appropriate serial composition relationships.

3.3 Model Structure: Instantiation and Specialization Layers

Since this thesis focuses on the model notation which denotes the conceptual model of ETL, but not the structure of internal logic of Transformation, the Instantiation and Specialization Layers are simply introduced here as part of the model.

The instantiation and the reuse of template activities are also allowed in the framework of model. The activities that stem from the composition of other activities are called composite activities, whereas the instantiations of template activities will be grouped under the general name of custom activities. Custom activities are applied over the specific table of an ETL environment. Moreover, they also involve the construction of ad-hoc, user tailored elementary activities by the designer.
The key issue in the conceptual representation of ETL activities lies (a) on the identification of a small set of generic constructs that are powerful enough to capture all cases and (b) on an extensibility mechanism that allows the construction of a ‘palette’ of frequently used types (e.g., for data stores and activities).

Figure 3.3 presents the ETL conceptual model architecture. It consists of metadata layer, template layer, and schema layer. We will explain in detail this architecture in following sections.

![Figure 3.3 The ETL conceptual model with three layers](image)

### 3.3.1 Metadata Layer

The metamodeling framework is depicted in Figure 2.3. The lower layer is Schema Layer, involves a specific ETL scenario. All the entities of the Schema layer are instances of the classes Concept, Attribute, Transformation, ETL Constraint, and Relationship. The meta-class layer, namely Metamodel Layer, involves the aforementioned classes. The linkage between the
Metamodel and the Schema layers is achieved through instantiation (“instanceOf”) relationships. The Metamodel layer implements the generality desideratum: the five classes which are involved in the Metamodel layer are generic enough to model any ETL scenario, through the appropriate instantiation.

3.3.2 Template Layer

To make the metamodel truly useful for practical cases of ETL processes, it is enriched with a set of ETL specific constructs, which constitute a subset of the larger metamodel layer, namely the Template Layer. The constructs in the Template layer are also meta-classes, but they are quite customized for the regular cases of ETL processes. Thus, the classes of the Template layer as specializations (i.e., subclasses) of the generic classes of the Metamodel layer (depicted as “IsA” relationships in Figure 3). Through this customization mechanism, the designer can pick the instances of the Schema layer from a much richer palette of constructs; in this setting, the entities of the Schema layer are instantiations, not only of the respective classes of the Metamodel layer, but also of their subclasses in the Template layer.

3.3.3 Implementation of Template Layer

In the example of Manpower Data Warehouse, the concept MDW.HeadCount Fact must be populated from a certain source O1. Several operations must intervene during the propagation: for example, a surrogate key assignment and an aggregation take place in the scenario. Moreover, there are two candidates suitable for the concept O1.JOB; out of them, exactly one (History Job or Recent Job) is eventually selected for the task. As one can observe, the concepts that take part in this scenario are instances of class Concept (belonging to the metamodel layer) and specifically of its subclass ER Entity, because we adopt an ER model extension.
Instances and encompassing classes are related through links of type instanceOf. The same mechanism applies to all the transformations of the scenario, which are (a) instances of class Transformation and (b) instances of one of its subclasses, depicted in Figure 3. Relationships do not escape the rule either: observe how the provider links from the concept O1.HCActuals towards the concept MDW.HeadCount Fact are related to class Provider Relationship through the appropriate instanceOf links.

As far as the class Concept is concerned, in the Template layer we can instance it to several subclasses, depending on the employed model. In the case of the ER model, we have the subclasses ER Entity, and ER Relationship, whereas in the case of the Dimensional Model, we can have subclasses as Fact Table or Dimension. Following the same framework, class Transformation is further specialized to an extensible set of reoccurring patterns of ETL activities, depicted in Table 3.1

<table>
<thead>
<tr>
<th>Classes</th>
<th>Transformations</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filters</td>
<td>Selection ($\sigma$)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Not null (NN)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Primary key violation (PK)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Foreign key violation (FK)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unique value (UN)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Domain mismatch (DM)</td>
<td></td>
</tr>
<tr>
<td>Transfer Operations</td>
<td>Ftp (FTP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Compress/Decompress (Z/dZ)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Encrypt/Decrypt (Cr/dCr)</td>
<td></td>
</tr>
<tr>
<td>Unary Transformation</td>
<td>Push</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Aggregation ($\gamma$)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Projection ($\pi$)</td>
<td></td>
</tr>
<tr>
<td>Classes</td>
<td>Transformations</td>
<td>Notes</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>Function application (f)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Surrogate key assignment (SK)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tuple normalization (N)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tuple denormalization (DN)</td>
<td></td>
</tr>
<tr>
<td>File Operations</td>
<td>EBCDIC to ASCII conversion (EB2AS)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sort file (Sort)</td>
<td></td>
</tr>
<tr>
<td>Binary Transformation</td>
<td>Union (U)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Join (__)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Diff (Δ)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Update Detection (ΔUPD)</td>
<td></td>
</tr>
<tr>
<td>Composite Transformation</td>
<td>Slowly changing dimension (Type 1,2,3)(SDC-1/2/3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Format mismatch (FM)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data type conversion (DTC)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Switch (σ*)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extended union (U)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1 Transformation Classes

There are four groups of logical transformations and two groups of physical transformations. The first logical group, named Filters, provides checks for the respect of a certain condition. The semantics of these filters are the obvious (starting from a generic selection condition and proceeding to the check for null values, primary or foreign key violation). Other logical groups of transformations are Unary and Binary Transformations. The former consists of the most generic push activity (which simply propagates data from the provider to the consumer), as well as the classical aggregation and function application operations along with three data warehouse specific transformations (surrogate key assignment, normalization and denormalization). The latter group consists of classical binary operations, such as union, join and difference of concepts as well as with a special case of difference involving the detection of updates. A set of advanced, composite transformations involving the combination of simple transformations (with particular care to data warehouse specific tasks, such as slowly changing dimensions, format mismatches) completes the set of logical groups of transformations.
File Operations (EBCDIC to ASCII, sort file). Summarizing, the metamodel layer is a set of
generic entities, able to represent any ETL scenario. At the same time, the genericity of the
metamodel layer is complemented with the extensibility of the Template layer, which is a set of
“built-in” specializations of the entities of the Template layer, specifically tailored for the most
frequent elements of ETL scenarios. Moreover, apart from this “built-in”, ETL-specific extension
of the generic metamodel, if the designer decides that several ‘patterns’ occur repeatedly in his
data warehousing projects, he/she can easily fit them into the customizable Template layer
through a specialization mechanism.

3.4 ETL conceptual model, ETL logical Model, and Implementation

Data warehouse or ETL design and building follows the similar database implementation
process: starting from conceptual model, then logical model, and at last, the physical model or
implementation.

3.4.1 ETL Conceptual Model and ETL Logical Model

The ETL conceptual model is constructed in the early stages of the data warehouse project. It
represents the overall logical structure of an ETL process, which is independent of any software
or data storage structure. It gives a formal representation of the data needed for ETL activities.

A Logical Data Model (LDM) is a representation of business concepts laid out in a visual format
that clearly shows these concepts and their various relationships. It is independent of the
underlying database implementation.
LDMs also serve as a foundation for data quality. Models that don't follow first normal form or have the wrong relationships often store duplicate data, resulting in loss of data quality. Proper modeling of items such as domains and data types helps validate data quality checks. LDMs also must comply with data governance guidelines and any overall data standards in the enterprise.

In ETL process, the major activities are the transfer of data from source to destination, during which the conversion of data may happens. As far as data quality is concerned, ETL also filters or rejects some data, which is meaningless and useless to data warehouse. In such an environment, ETL logical model has a focus on transformation, data flow, and rejects processing.

Same as any other data processing scenario, in ETL, the development of a logical data model (LDM) begins with a broad set of data requirements. Using the requirements along with detailed attribution and mapping of data elements from the source environment to data warehouse creates a conceptual model showing the relationships between key entities. Metadata, including domains, data types, definitions, comments, and notes, is also created.

The next stage is detailed data flow and transformations. Detailed data flow and transformation are created based on the conceptual model and instantiation of meta-classes. This leads to the logical model. Following this, peers and users typically review the model. After review and, if necessary, revision, the model becomes available for the next step in the modeling continuum: the Physical Data Model (the PDM) or for the implementation in the ETL scenario.

### 3.4.2 ETL Logical Model and Implementation

The LDM of ETL Process is to present a formal logical model for the activities of an ETL environment. This model abstracts from the technicalities of monitoring, scheduling, logging
while it concentrates on the flow of data from the sources towards the data warehouse through
the composition of activities and data stores.

Different from the LDM for the whole data warehouse, the logical model for ETL will not cover
the Semantic Data Model, which is designed for the user-end applications, such as reporting or
OLAP tools. Nor will it cover a lot of Physical Data Model for the target data warehouse. It does
feed data to the data warehouse or data mart. However, it will not concern a lot about the PDM
design such as Star Schema or Snow Flake Schema.

A formal logical model is defined as a logical abstraction of ETL processes. The data stores,
activities and their constituent parts are defined formally. An activity is defined as an entity with
(possibly more than one) input schema(ta), an output schema, a rejection schema for the rows
that do not pass the criteria of the activity and a parameter schema, so that the activity is
populated each time with its proper parameter values. The flow of data from producers towards
their consumers is achieved through the usage of provider relationships that map the attributes of
the former to the respective attributes of the latter. The combination of ETL activities, provider
relationships, and data stores constitutes an ETL scenario.

The implementation of ETL will transform the ETL scenario, the combination of ETL activities,
provider relationships, and data stores, into a group of abstract objects in computer. These forms
of computer objects can be a bundle of procedural programs in programming languages such as
C, Basic, COBOL, or SQR, or they can be a group of classes and objects designed in object-
oriented language such as C++ and JAVA. In other case, they can be one or more web services,
bunch of SQL or database scripts, or at most cases a combination of these mentioned program
forms.
With the market of ETL grows, more and more commercial ETL tools, such as Congos DecisionStream, Informatica PowerCenter, are available. They provide more abstract and business-oriented objects to help ETL professionals to reduce programming time. For example, using Cognos DecisionStream, people construct objects, such as connections, dimensions, fact builds, and controls, in a graphical interface. The tool will help to translate these built objects into scripts or programs, and then execute them.

In the past, the ETL implementation has changed from bundle of legacy code or programs to the ETL engine. The next generation would be the E-LT (Extract - Load & Transform) Architecture, which generates and executes highly optimized data integration processes, in the native SQL or in the other languages of the databases involved in these processes.

No matter what ETL implementation to select, it only influences the final phase implementation. Our conceptual and logical model for ETL, as the UML is independent of the programming language, are independent of these ETL implementations. However, ETL models provide clear and dedicated designs for these solutions. How well and efficiently ETL models help the implementation, is one of the questions we hope to explore in this thesis.

**Chapter 4 Implementation of ETL Conceptual Model**

A properly selected practical project validates the model’s functionality, flexibility, and adaptability. To further discuss and verify the concepts and objects introduced in the ETL conceptual model, we apply the ETL modeling to a actual project.

In this implementation, a practical ETL system environment and settings is selected. This practical ETL project is to build an ETL system for a manpower data warehouse of a middle size corporation. Even though we focus on designing its conceptual model, we also present its logical
design in logical model, and implementation builds, functions, and interfaces in Cognos DecisionStream so that the whole ETL process and implementation can be demonstrated.

In the following sections, we first describe the system settings and requirements for ETL process of the Manpower Data Warehouse. Then, we present our conceptual model designed for ETL process and explained all the objects and activities covered. Next, we demonstrate the completed logical model and describe the data flow. At the end, we represent the builds, functions, and interfaces implemented in Cognos DecisionStream.

4.1 Requirements of Manpower Data Warehouse and its ETL process

The example project is a manpower data warehouse system, which analyzes, compares, budgets, and forecasts the employee and salary information. The data sources to the Manpower Data Warehouse include Human Resources Management Information System (HRMIS), Finance Information System (FIS), Web-based Data System (WDS), Text or Excel files. From the HRMIS, we retrieve active employee information, such as employee name, full time or part time, exempt or non-exempt, regular or temporary, annual salary, and hours worked. From FIS, we retrieve employee’s salary information, such as base salary, overtime income, and bonus. WDS provides budgeting or forecast information such as budgeted positions to be filled, budgeted overtime payment, budgeted Temp, Agent, or Contractor payment. Text or Excel files are used to provide above-mentioned information for department or organization whose data is not covered by HRMIS, FIS, and WDS. Also Text or Excel files are usually provided in different granularity level, which need complex ETL functions to deal with.

HRMIS, FIS, WDS, and Manpower Data Warehouse are based on Oracle databases. Therefore, we call these three data source respectively as O1, O2, and MDW. Moreover, we design a Data
Staging Area (DSA) in an Oracle instance to finish most of the transformations. See Figure 4.1 for the Manpower Data Warehouse architecture.

![Figure 4.1 Manpower Data Warehouse architecture](image)

In table 4.1, we list all the sources (databases or group of text files), tables, and table schemas (fields in tables or attributes). In ETL process, these data objects are involved and serve as source data, staging, or target data.

<table>
<thead>
<tr>
<th>Source</th>
<th>Table Name</th>
<th>Table Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>O1.Personal_data</td>
<td>Emplid, Name,</td>
</tr>
<tr>
<td>O1</td>
<td>O1.Job</td>
<td>Emplid, Department, Jobcode, Reg_temp, Full_part_time, FLSA_status, Empl_status,</td>
</tr>
<tr>
<td>Source</td>
<td>Table Name</td>
<td>Table Schema</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Std_hours, Annual_rate, Grade, Position_nbr,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Paygroup</td>
</tr>
<tr>
<td>O1</td>
<td>O1.Employment</td>
<td>Emplid, Termination_date, Rehire_date</td>
</tr>
<tr>
<td>O1</td>
<td>O1.All_chk_hrs_ern</td>
<td>Emplid, Al_hours</td>
</tr>
<tr>
<td>O1</td>
<td>O1.Job</td>
<td>Emplid, Action, Action_reason,</td>
</tr>
<tr>
<td>O1</td>
<td>O1.Manpower_Hierarchy</td>
<td>gfms, org_unit, Executive</td>
</tr>
<tr>
<td>O1</td>
<td>O1.Actn_reason_tbl</td>
<td>Action, Action_reason, descry</td>
</tr>
<tr>
<td>O1</td>
<td>O1.Position_data</td>
<td></td>
</tr>
<tr>
<td>O1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2</td>
<td>O2.GLi_or_out</td>
<td>File_nbr, GL_Paygroup, GL_Payday,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GL_Description, GL_New_desc, GL_Status,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GL_Row_source, GL_amount</td>
</tr>
<tr>
<td>O2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>T1.Exception_data</td>
<td>NOTES, Dept ID, Sal/hc, Reg_Temp,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Full_Part_Time,FLSA_Status, EMPL_Status,EX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fraction,JAN,FEB,MAR,APR,MAY,EX Jan,EX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Feb,EX Mar,EX Apr,EX May,EX Jun,EX Jul,EX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Aug,EX Sep,EX Oct,EX Nov,EX Dec,Non Ex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Jan,Non Ex Feb,Non Ex Mar,Non Ex Apr,Non Ex May</td>
</tr>
<tr>
<td>DSA</td>
<td>DSA.Manpower_Hierarchy</td>
<td>Total,Subtotal,Executive,Org_unit,gfms</td>
</tr>
<tr>
<td>DSA</td>
<td>DSA.HC Actuals</td>
<td>Month Year, SEQNO, Emplid, Name,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Department, Full_part_time, FLSA_status,</td>
</tr>
<tr>
<td>Source</td>
<td>Table Name</td>
<td>Table Schema</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MDW</td>
<td>MDW.Months or Periods Dimension</td>
<td>Year, Month</td>
</tr>
<tr>
<td>MDW</td>
<td>MDW.Hierarchy or Org Structure Dimension</td>
<td></td>
</tr>
<tr>
<td>MDW</td>
<td>MDW.Serial Num Dimension</td>
<td></td>
</tr>
<tr>
<td>MDW</td>
<td>MDW.Reg or Temp Dimension</td>
<td></td>
</tr>
<tr>
<td>MDW</td>
<td>MDW. Full or Part Time Dimension</td>
<td></td>
</tr>
<tr>
<td>MDW</td>
<td>MDW.FLSA Status Dimension</td>
<td></td>
</tr>
<tr>
<td>MDW</td>
<td>MDW.Employee Type Dimension</td>
<td></td>
</tr>
<tr>
<td>MDW</td>
<td>MDW.Versions Dimension</td>
<td></td>
</tr>
<tr>
<td>MDW</td>
<td>MDW.Roster Fact</td>
<td></td>
</tr>
<tr>
<td>MDW</td>
<td>MDW.Executive Fact</td>
<td></td>
</tr>
<tr>
<td>MDW</td>
<td>MDW.HC, Salary, Vacancy, and LOA Fact</td>
<td></td>
</tr>
<tr>
<td>MDW</td>
<td>MDW.Other Measures Fact</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1 Data source and schema for Manpower Data Warehouse
There are about nine dimension tables and four fact dimensions (see table 4.2) in the data store to provide three types of basic information of employee count, status, and salary. All these types of information are also presented in three versions: Actual, Budget, and Forecast. Budget data is available one month before the beginning month of a financial year and kept unchanged through the year. Forecast data is refreshed every quarter. New Forecast comes out one week before the quarter. Actual data is calculated monthly.

<table>
<thead>
<tr>
<th>#</th>
<th>Fact Table</th>
<th>Dimension Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HTX</td>
<td>Months or periods</td>
</tr>
<tr>
<td>2</td>
<td>Roster</td>
<td>Hierarchy or Org structure</td>
</tr>
<tr>
<td>3</td>
<td>HC, Salary, Vacancy, and LOA</td>
<td>Serial #</td>
</tr>
<tr>
<td>4</td>
<td>Executive</td>
<td>Reg or Temp</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Full or Part time</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>FLSA_Status</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Employee Type</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Versions.</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Measures</td>
</tr>
</tbody>
</table>

Table 4.2 Fact and dimension tables in Manpower Data Warehouse

In the example scenario, the actual data is from the data source named as O1, O2, T1, DSA and is in a format of details. Budget data and forecast data come from data source called WDS, in a more consolidated form. These three versions of data from different data sources in different time slots feed into same ETL system, and the outcome will populate the Manpower Data Warehouse.
In following sections, we will implement part of the ETL system in Manpower Data Warehouse – from data source (O1) to Data Staging Area (DSA) then to Manpower Data Warehouse (MDW).

4.2 Conceptual Model for ETL process in Manpower Data warehouse

The ETL process in Manpower Data Warehouse involves the propagation of data from the tables such as JOB and Personal_data. of source O1 as well as from the Exception_Data of source T1 to the data warehouse. Table MDW.HC and Salary Fact stores the information for available Employees (HeadCount) and Salary (Salary) of personals in each department (ORG). Practically, the two data sources O1 and T1 stand for the two types of employees of the data warehouse. The first data source is for employees in existing HRMS and the second is for those not, thus the data coming from the first source need to be aggregated to departmental level. This is done through staging table HC Actuals. Once the table HC Actuals is populated, data is further aggregated to populate the data marts HC and Salary Fact in the data warehouse. The conceptual model is graphically depicted in Figure 4.2.
Figure 4.2 Manpower Data Warehouse ETL Conceptual Model
In the conceptual model, the following transformations are involved.

1. First, data is retrieved from the snapshot of the sources O1.Job and O1.Employmentt. via effective date. This is done by applying Effective date to the D constrain.

2. The department in data source of O1.Job will not go to SDA.HC Actuals directly. The organization divisions in HRMS are different from these required for reporting in data warehouse. The mapping from department in O1.job to department in SDA.HC Actuals is done through a transformation S.

3. The name in data source O1.Personal Data needs similar transformation before feeding to SDA.HC Actuals. This is a format transformation called T. It is a kind of transformation like combining last name and first name into a single name.

4. In the SDA.HC Actuals, a SEQNO is required to identify a record uniquely. A kind of surrogate key is assigned on SEQNO. In the data warehouse context, it is common tactics to replace the keys of the production systems with a uniform key, which we call a surrogate key. The basic reasons for this replacement are performance and semantic homogeneity. Textual attributes are not good candidates for indexed keys and thus need to be replaced by integer keys. The surrogate key is produced by function SK.

5. FTE is calculated by using Std-hour divided by all-hour. This is done through a transformation function. In addition, this transformation should be after all attributes of SDA.HC Actuals are updated or inserted because it will use other attributes from the same table.

6. Another Transformation between the source tables and the staging table DSA.HC Actuals is transformation C. This transformation works as a constraint or flag to decide whether the source data go or not go to the destination. It affects all the fields in destination.
7. Other transformations or checks covered in the \( \Theta \) transformation include a test for NULL values, an insert of a DATE attribute with the value of system’s date, and a filter. The test verifies NULL values for the attributes of Jobcode and Position_nbr in O1.Job (activity NotNULL). The insert adds a DATE attribute with the value of system’s date, since source O1 does not contain date information (activity AddDate), while the filter selects only the records for which a available all_hours in O1.All_chk_hrs_em exists (all_hours >0), using activity CheckZero.

8. The transformation between O1.HC Actuals and MDS.Roster Fact is a typical example of concept level transformation. The Roster Fact is a kind of data warehouse table, each period a snapshot of data from HC Actuals is exported to here and tagged with a special period tag.

9. The transformation S is the most complex transformation in this practical example. The source data of employee level from DSA. HC Actuals will be aggregated into department level summaries.

4.3 Logical Model for ETL process in Manpower Data Warehouse

The logical model, as shown in figure 4.3, presents clearly the propagation of data from the table JOB, PERSONAL_DATA, EMPLOYMENT, HIERARCHY in source \( O_1 \) as well as from the table ALL_CHK_HRS_EM of source O1 to the data staging area (DSA). Table DSA.HC Actuals stores employee information, such as name, job type, employment type, and working hours.
(FTE).

![Diagram showing data flow and transformations]

Figure 4.3 Manpower Data Warehouse ETL Logical Model

The department in the source table O1.JOB is not in the same level and not standard as the department in hierarchy. Thus, the departments coming from the O1.JOB need to be converted to department values and formats from O1.Hierarchy. Once the DSA table of HC Actuals is populated, data is further aggregated to populate two data marts MDW.Roster Fact and MDW.HeadCount Fact in the data warehouse. The data mart MDW.Roster Fact stores historical information of employees and the data mart MDW.HeadCount Fact stores aggregated information of employees in each department. The data flow is depicted graphically in Figure 3.4 and involves the following transformations:
1. First, we transfer data of the snapshots from the sources O1.JOB, O1.EMPLOYMENT, O1.PERSONAL_DATA, O1.HIERARCHY to the table of DSA.HC Actauls of the DSA.

2. Before the data is transferred from data source O1 to the DSA. Some transformations must be done. These transformations include the department transformation which maps the department in job to standard department from hierarchy, name transformation which changes the name format to one that required by reporting, effdt transformation which filter out of data which is not effective, and employment transformation that filters non-active employees.

3. We assign a surrogate key on PKEY for data in DSA, before the data get into the DSA, or after the data feed into the DSA. In the data warehouse context, it is common tactics to replace the keys of the production systems with a uniform key, which we call a surrogate key. The basic reasons for this replacement are performance and semantic homogeneity. Textual attributes are not good candidates for indexed keys and thus need to be replaced by integer keys. At the same time, different production systems might use different keys for the same object, or the same key for different objects, resulting in the need for a global replacement of these values in the data warehouse. In our case, the activity that performs the surrogate key assignment for the attribute PKEY is SK. They both use the lookup table LOOKUP_PS.

4. For the data in DSA, we need to apply a function to calculate the FTE and keep the FTE in the DSA. This transformation is done through activity FTE.

5. We need to populate the two data marts of the warehouse. For the data mart MDW.Roster Fact, we simply append data to there. For the data mart MDW.HeadCount Fact, we will map the data to five different dimensions and aggregate the data to department level.

6. For several of the aforementioned activities, we need to trace the rows that do not pass the check or transformation performed. In this case, we employ the rejection schema of the activity to send these rows to their respective Log file.
4.4 ETL Process Implementation in Cognos DecisionStream

According to the logical model, we implement the whole ETL process in Cognos DecisionStream. For the transformations between data source and Data staging area, we create two Fact Builds: ToStagingArea-1 and ToStagingArea-2. We create two build instead of one because the transformation in ToStagingArea-2 have to be executed after all transformations in ToStagingArea-1 are done. Following figure 4.4 and figure 4.5 show the fact builds of ToStagingArea-1 and ToStagingArea-2.

Figure 4.4 Manpower Data Warehouse ETL Implementation - Fact Build 1
Figure 4.5 Manpower Data Warehouse ETL Implementation - Fact Build 2

For all transformation between Data Staging Area and Data Warehouse, we create another two Fact Builds: StagingToDW-1 and StagingToDW-2, because there are two different destinations in data warehouse and each has different requirements. The build to HeadCount Fact requires aggregation while the build to Roster Fact has no aggregation required. Following figure 4.6 and figure 4.7 show the fact builds of StagingToDW-1 and StagingToDW-2.
Figure 4.6 Manpower Data Warehouse ETL Implementation - Fact Build 3

Figure 4.7 Manpower Data Warehouse ETL Implementation - Fact Build 4

All these four Fact Builds are controlled and executed under an order set in Job Builder – ETL Process demonstrated in figure 4.8.
Other objects created for the implementation includes three functions which work as instances of transformation functions, five dimensions for the data warehouse, and three database connections to connect to source, staging, and destination data warehouse. Following figures 4.9, figure 4.10, and figure 4.11 show these objects.
Figure 4.9 Manpower Data Warehouse ETL Implementation – Database Connections

Figure 4.10 Manpower Data Warehouse ETL Implementation – Transformation Functions
4.5 Analysis and Comparison of ETL Implementation

Compare to traditional ad hoc program implementation of ETL process, above implementation clearly demonstrates some advantages.

The conceptual model clearly defines the concepts and attributes involving an ETL process. This not only provides a graphical presentation to improve the communication between users and modelers in the requirements or planning phase, but also lays a strong foundation for next phase – design phase. As a systematic methodology, the ETL conceptual model provides the ETL process a more system and logical design, which further make the ETL project a success. From the practical example, we see clearly that the conceptual model defines the major concepts and activities, while the logical model focuses on the data flow among the concepts.
The graph based conceptual model provides a formal and systematic method of ETL design and implementation. It greatly improves the maintenance and update process for the ETL system. Many errors, frequently met in the ad hoc program of ETL, include the functionality redundancy and data conflict. One example is that there are many date conversion programs in different part of the ETL process. ETL Conceptual Model avoids these withdraws by using activity instances.

The data modeling does brings many advantages, and improve ETL a lot in integrity, maintenance, reuse, reliability, extendibility, and capacity, thought there is no apparent demonstration of help in performance.

4.6 ETL Implementation Summary

Through the implementation of Manpower Data Warehouse in ETL conceptual model, we strongly experienced the advantages brought by the systematic modeling method – graphical presentation of logic and data flow, high reuse of transformation code, easy maintenance, and extendibility, and the integrity and reliability from a formal design.

However, we also felt that some parts of the model notation are not good or awkward to denote some situations from practical projects. One of these lacks is the there is no sequence or timing notation to express the sequence among all transformations. Only the direction of arrow is not enough in complex ETL process. In next chapter, we discuss three extensions to the ETL conceptual model notation.
Chapter 5 Analysis and Extension of ETL Conceptual Model

In this chapter, we will analyze the notation of ETL conceptual model proposed by Panos Vassiliadis. The clearly defined notation of the ETL conceptual model specified and demonstrated the major ETL concepts and objects. It also introduces a three-layer instantiation to provide a generic constructs and extensibility mechanism. However, how does the notation of this model meet the requirements in actual ETL projects? We try to answer this question in this chapter.

5.1 ETL Conceptual Model Analysis

Comparing to other conceptual models such as UML and dimensional model, ETL conceptual model is the first one specialized for ETL process. It explicitly demonstrates most of the ETL activities. However, some important ETL properties such as execution order or timing, some special transformations, and logging or other error processing can not be presented in the model notation. For such a conceptual model used to catch user requirements, lacking methods to catch these ETL properties certainly affects its applicability and functionality.

5.1.1 Execution Orders or Timing

ETL deals with the data transfer from sources to destinations. In practical environment, the data sources can be complex. A type of data may come from several different types of data sources, such as databases, text files, or spread sheet, and each could be available in different time slots. Also during the ETL process, some temporary or middle data is created and stored. These types of data probably need further processing, or it combine with other data to proceed to next point of ETL process. In other words, not all source data comes at the same time during ETL process. Some data may join in the middle of the process and need not much ETL processing. Based on
these reasons, the timing or sequence of an ETL activity is an important property of an ETL activity. In addition, this property is mentioned or proposed frequently during the initial process of analysis. It should be an official part of the ETL requirements and be defined properly in the ETL conceptual model.

With the development of data warehouse and on-line application, real time ETL has been proposed and suggested. An ETL conceptual model with strong timing or execution order design will be better prepared for future ETL design.

**Case 1:** In our practical example, there is a transformation, which calculates the FTE using hours data from the DSA database and then write the FTE, result of the transformation, back into the DSA database. This transformation must be done after other transformation because it uses data from the database which actually from other transformations. See following diagram in figure 5.1.

![Figure 5.1 Timing Order Case1](image-url)
**Case 2:** Sometimes in ETL process, the sequence among different activities is not because of the logical relationship among the data, but from consideration of data quality or availability. The users in the special situation manually or optimally set the order or timing sequence. For example, in the Manpower Data Warehouse, the salary data and headcount data are independent to each other. Salary data, which comes from paychecks, are fixed once they are available. However, the Headcount data is dynamic in some degree because of the delay of data entry. It is more accurate if it is given more time to become stable. Therefore, for the data quality we define the sequence that headcount data is retrieved and transformed after salary data.

In Panos Vassiliadis’s conceptual model, none of activity timing or sequence is mentioned or suggested. He leaves the activity execution sequence or timing all to the design phase and suggested to express them in notes [Vassiliadis and Skiadopoulos 2002]. This brings many problems. One problem is how effectively the sequence or time can be expressed in words. Another problem is how the model designer or analyst can understand it correctly or effectively. In some degree, the noted information may bring further confusing or wrong sequence or timing.

As far as the timing order or sequence is concerned, we suggest extending the notation set to include a special sequence symbol to express the sequence and timing of an ETL activity explicitly and graphically.

**5.1.2 Special Transformation**

In Panos Vassiliadis’s conceptual model, many transformations are covered and discussed. For example, he even gave a list of all transformations, categorizing all transformations into six types: Filters, Transfer Operations, Unary Transformation, File Operation, Binary
Transformations, and Composite transformation. These transformations do represent a wide range of ETL activities.

In the conceptual model, data are retrieved from the source and through these transformation processes, they are sent to the destination or staging data storage. In these cases, the transformation happens between the two data sources. However, in practical project, many transformations are done on the same data source, writing the transformed data back to same data source.

How does the conceptual model express these transformations efficiently?

**Case 1:** In our practical example, there are some transformations, which translate the digital expression of some information into meaningful text. This is a kind of basic requirement for data warehouse. These transformations get data from DSA database, translate the data into significant string, and then write back to the same database. See following diagram in figure 5.2.

Figure 5.2 Special Transformation case1
**Case 2:** In the process of ETL, this is also a situation, which happened often: a transformation gets source data from one or several attributes and writes the transformation result back to one or several attributes in the same concept (table).

### 5.1.3 Schedule, Logging, Trace, and Recovery

Requirements in Schedule, Logging, Trace, and Recovery are requirements to the Whole ETL system. However, lower level objects sometimes should be designed with a mechanism to implement these requirements, such a mechanism, or design is common to all the objects or modules. Therefore, in the conceptual model, such a requirement needs not to be repeated in each object or activities.

In Panos Vassiliadis’s conceptual model, such requirements are suggested to be specified in the notes. However, the notes are used popularly to specify requirements to single objects such as a data source or a transformation. This is a contradiction. In one aspect, the requirements in Schedule, Logging, Trace, and Recovery are on system level. In another aspect, these requirements are designated to be specified in object level. No mention whether it is meaningless to repeat such specification in each object note, the ETL designer or developer faces difficult to put such system requirement together from each object’s note.

**Case 1:** In the Panos Vassiliadis’s model, he mixes the summary notes and the detail notes and puts them in a same note box. This makes the important summary requirement being ignored, especially when project goes to a next phase or when doing maintenance according to the model document.
Case 2: There is no standard or strict logical definition for the language used in the notes. It is actually very dynamic depending on the writer’s expression habit and writing ability. Therefore, in the case of no explicitly identification of the notes, summary requirements are easily mixed as detail requirements.

5.2 ETL Conceptual Model Extension

For the three ETL functionalities mentioned above, the existing conceptual model lacks the proper mechanism or notation to express them. We can extend the conceptual model to cover these important functionalities. In this part, we analyze and compare the methods to implement execution orders, special transformation, and system function of Schedule, Logging, Trace, and Recovery. In a detail, we will discuss the extension solutions for ETL conceptual model.

5.2.1 Execution Orders or Timing

In UML, the Unified Modeling Language, there are many types of diagrams designed to explore the behavior of objects throughout a given period of time or to describe the sequence of messages or to show the interaction ordering among objects. These diagrams are called sequence diagram.

A sequence diagram shows, as parallel vertical lines, different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. Following is an example of sequence diagram.
A sequence diagram generally shows the interaction between objects over the progression of time. Thus, the first symbol required is that to represent an object. More correctly, a lifeline is drawn. It is a vertically dashed line, which has as its top a rectangle indicating the instance and class names in the standard UML format of instance name : class name.

In order to display interaction, messages are used. These are horizontal arrows with the message name written above them. Solid arrows with full heads are synchronous calls, solid arrows with stick heads are asynchronous calls and dashed arrows with stick heads are return messages. Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message.

In ETL logical model, a concept of stage is suggested to solve problems or questions concerning the execution order of the activities. Stage, a transformation stage (or a stage), is defined as a...
visual region in ETL dataflow in which a set of transformations that act within this region have no sequence order [Vassiliadis and Skiadopoulos 2002]. An ETL process can be divided into many transformation stages.

The transformations that belong to the same stage are called stage-equivalent transformations. All activities that stem from stage-equivalent transformations are swappable, thus there is no problem in their ordering. Two activities can be swapped in a logical ETL workflow, i.e., when we are able to interchange their execution priorities. The transformations that belong to the same stage, and whose respective activities in the logical design are swappable, are called order-equivalent transformations [Vassiliadis and Skiadopoulos 2002].

Combining the UML’s representation method of sequence diagram and the stage concept, we propose a notation to present sequence order of transformation during an ETL process. We called is time division line. It is a dotted vertical line with a dotted rectangle box on its top. The rectangle box is used to specify a time point. The accuracy of the time point is up to the ETL process. There are always a starting time line and an ending time line. If there is no sequence order among all transformation activities, these two time lines enclose a unique time slot for the whole ETL process. When the ETL activities need to be done in different priorities, more time division lines should be introduced to divide the space between the starting line and ending line into more sub-spaces. From the view of timing, more time slots are created. Thus, different activities with different priorities should be placed or drawn in different divisions. See following figure 5.4.
After applying above new notation into our practical example and implementation, we can achieve following new diagram in figure 5.5 for the ETL conceptual model. The whole process is divided into more sub-spaces. From the diagram, we can clearly find the transformations are in different time division, even they target to the same target database.
5.2.2 Special Transformation

To solve self-transformation, we can use existing diagram notation and draw a line from the data source and then transformation circle and again a line back the original data source. Figure 4.6 clearly demonstrates the notation. However, it does bring a new problem. If there is a self-transformation and a non-self-transformation from same data source, which one should go ahead first? This brings the priority problem.
The time division line suggested in prior section provides a simple solution to this problem. We can use time division line to present clearly the execution order of different transformations from same data source. In following figure 5.6, we demonstrate a scene of two transformations from a unique data source. In following figure 5.7, we applied the new self-transformation to the conceptual model of Manpower Data Warehouse.

Figure 5.6 Self-transformation diagram for ETL conceptual model
5.2.3 Schedule, Logging, Trace, and Recovery

There are two options to solve the problem that the note, defined in the conceptual model, is used for object level constraints or comments. One is to define a new notation symbol for system level notes such as execution schedule, logging or trace requirements, and recovery requirement. The other is to add a flag to the existing notation so that it can work for both of object level notes and system level summary notes.

We suggest this second solution because it will not bring a new notation symbol, which may also bring users the confusion when selecting the proper symbol.
Notes are still depicted as a rectangle with the top-right corner folded over. However, there is a connection line to connect it to the specific concept, attribute, transformation, or relationship, which this note is intended to. If there is no connection line to link the rectangle to any other model objects, this is a summary note for the whole diagram. Among these summary, if there are needs to comment on a group of objects, or other non-whole diagram levels, in the notes, the range should be explicitly defined. Any notes without any range defined in the summary note boxes are considered as to the whole diagram. See following diagram in figure 5.8 for the detail notes and summary notes notation.

![Summary Notes and Detail Notes](image)

Figure 5.8 New notes notation for summary notes and detail notes
Figure 5.9 The Manpower Data Warehouse ETL model with extended notation
Chapter 6 Result Analysis and Comparison

In prior chapter, we added three new notation symbols to the existing one to extend its functionality in expressing timing order, special transformation, and summary/detail notes. With these three new symbols, the ETL conceptual model has stronger capability to denote various practical ETL processes.

6.1 Analysis and Comparison

The new timing sequence notation not only clearly define sequence of the activities in ETL conceptual model, but also make the awkward and tedious ETL design process smooth and reasonable. We will make analysis about this in a detail in following part.

First, the timing order and sequence are important part of information that collected or created during the phase of requirement acquiring and existing system analysis. It is not natural and correct to ignore this part during the requirement and analysis phase. An ETL conceptual model without timing order and sequence is not a complete one. The further phases such as design and coding can not be done without them. In addition, the requirement acquiring and analysis phase in which the ETL conceptual model is created is the best period to get the activities’ sequence and timing order. As we have discussed in the other part of this study, sequence and timing order are not all completely logical. Sometimes, people manually set up the order to achieve the optimum outcome. In the phase of requirement acquiring and system analysis, modelers and users have many interactive communications such as meeting, presentation, and one-to-one discussion. Through these interaction and coordination, information (sequence and timing order) about activities can be easily acquired and presented in the ETL conceptual model.
Second, without sequence and timing order, the activities or transformation can be ambiguous. When actually there is an order between two transformation, it’s wrong to express them in the ETL conceptual model without any order. Because of lacking timing, the finally built ETL system may even not output correct results. In addition, the readers of the model diagrams will also feel confused by the group of activities without order.

Third, while the ETL conceptual model can still be created without timing order, the ETL logical model can not done without it. As we discussed, the ETL logical model focuses on the data flow of ETL process. Without sequence, how data can flow? As a result, the ETL conceptual model with timing order and sequence designed in it provides a strong foundation for the design of ETL logical model. The transition from conceptual model to logical model would be smooth and easy. However, in the original ETL conceptual model, since there is no sequence order and timing, when the project goes to the phase of Logical model design, it meet big problem. Either the logical modelers figure out the sequence among all the transformation activities, or they go back to the users for such information. It is awkward to find out the shortage of information during the design phase. There are many extra problems to face during this rollback process. For example, indefiniteness may occur. There may be integrity problems when introducing the timing in design phase. The logical modelers may not be familiar with the roles of requirement acquiring and analysis. There may be no enough time for logical modelers to make up the timing information.

The benefits brought by other two new notations are (1) that some special transformations from practical project can be denoted in the ETL conceptual model, and (2) that division of summary notes and detail notes makes all schedule, logging, and exception process can be clearly presented in the ETL conceptual model.
As a summary, we list all the benefits brought by the extended notation in following table 6.1.

<table>
<thead>
<tr>
<th>Compare</th>
<th>Extended Notation</th>
<th>Original Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model completeness</td>
<td>More complete</td>
<td>Lack timing order information</td>
</tr>
<tr>
<td>Model clarity</td>
<td>Activity order and notes are presented clearly</td>
<td>Confusing activity order and notes</td>
</tr>
<tr>
<td>Model transition to next phase</td>
<td>Smooth and easy</td>
<td>Awkward and tedious and easy to have errors</td>
</tr>
<tr>
<td>Model functionality</td>
<td>Able to present special transitions</td>
<td>Not able to present special transitions</td>
</tr>
</tbody>
</table>

Table 6.1 Comparison of extended notation and original notation
Chapter 7 Conclusions

In this study, we have implemented part of a practical Manpower Data Warehouse with formal ETL conceptual modeling, logical modeling, and the ETL tool of Cognos DecisionStream. These formal models are still in the theory or laboratory phase without popularly applied in industry. Upon the implementation experience, we strongly underwent the advantages brought by the formal modeling methodology such as maintainability, reuse, and extendibility. However, we also discovered some places in the notation needing improvement. We found that existing notations are awkward in denoting some practical situations or logics from practical projects. Based on these discoveries we suggested some extended notation for the ETL conceptual model. These extended notations help to express graphically the timing order or sequence logic in ETL conceptual model and to remove ambiguity in diagram notes.

As we introduced in the first part of this thesis, data warehouse projects have high failure rate. Also as an important major part of data warehouse building, ETL plays an important role in success of the data warehouse project. Traditional ETL system consists of ad hoc programs, or processes around ETL engine, which lack system and formal planning and design. These kinds of systems lack maintainability, reliability, extendibility, integrity, and reuse. In recent years, researchers realized these problems and began to explore formal modeling method for ETL system. Nevertheless, this methodology is still in research phase without popularly applied in industry.

Our implementation of the ETL conceptual model in a practical Manpower Data Warehouse demonstrates much improvement in system integrity, reuse, maintainability, and reliability to ETL system. For example, the ETL conceptual model defines clearly the attribute mapping from
source to target while local model focuses on the data flow. These two models provide strong tools for future maintenance and updates.

Implementation also suggested good feasibility in actual projects. All the notation symbols are popular shapes that can be easily found in various document tools such as MS VISIO or even MS Word. With these tools, these model diagrams can be easily maintained and output. However, if there are specially designed model diagram tools for their creation, maintenance, and output of ETL conceptual model and logical model, that would be better.

During our implementation, especially in the process to design the model diagrams using model notation, we discovered that some situations or cases from the practical project could not be presented clearly and properly in the existing notation set. One of these problems is that there is no method to express the timing order or activity sequence graphically in the existing notation set.

After the implementation, we extended the notation set so that it can deal with problems that are more practical. We added one new notation for timing order and updated other two to do self-transformation and summary notes. All these extended notations help to graphically and literally denote the ETL process. In addition, they help the building of ETL logical model a lot.

For further study or research, the tools to construct and maintain ETL models are important fields, which can help the industry to apply these methodologies in practical projects. Another direction is the automation of model transformation from ETL conceptual model to ETL logical model and from ETL logical model to ETL implementation tool. This automation process will combine the ETL model tools and ETL implementation tools into a single powerful automatic one.
References


