Universidade do Minho
Escola de Engenharia

Vasco Nuno Caio dos Santos

A Relational Algebra Approach to ETL Modeling

The MAP-i Doctoral Programme in Informatics, of the Universities of Minho, Aveiro and Porto

universidade de aveiro

Universidade do Minho

U.PORTO

July 2015

Vasco Nuno Caio dos Santos **A Relational Algebra Approach to ETL Modeling**

UMinho | 2015

Universidade do Minho
Escola de Engenharia

Vasco Nuno Caio dos Santos

A Relational Algebra Approach to ETL Modeling

The MAP-i Doctoral Programme in Informatics, of the Universities of Minho, Aveiro and Porto

universidade de aveiro

Universidade do Minho

U.PORTO

supervisor:
Professor Doutor Orlando Manuel de Oliveira Belo

July 2015

STATEMENT OF INTEGRITY

I hereby declare having conducted my thesis with integrity. I confirm that I have not used plagiarism or any form of falsification of results in the process of the thesis elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, 03 July 2015

Full name: Vasco Nuno Caio dos Santos

Signature: Vasco Santos

To those who wish me well.

Acknowledgments

I would like to express my appreciation to all those that in some way supported and helped during the long years that took me conclude this project, and in particular to:

- Professor Orlando Belo for his supervision, guidance and stimulus, and permanent companionship during this project.
- Polytechnic of Porto and School of Management and Technology of Felgueiras (IPP.ESTGF) for the support and conditions provided to conclude this project.
- My IPP.ESTGF colleagues who supported and encouraged me in a daily basis.
- My family, with special attention to my wife Marta, who not also encouraged but also demanded my total commitment in this project.
- All those that in some way were deprived of my company and have suffered from my absence.

Abstract

Information Technology has been one of drivers of the revolution that currently is happening in today's management decisions in most organizations. The amount of data gathered and processed through the use of computing devices has been growing every day, providing a valuable source of information for decision makers that are managing every type of organization, public or private. Gathering the right amount of data in a centralized and unified repository like a data warehouse is similar to build the foundations for a system that will act has a base to support decision making processes requiring factual information. Nevertheless, the complexity of building such a repository is very challenging, as well as developing all the components of a data warehousing system. One of the most critical components of a data warehousing system is the Extract-Transform-Load component, ETL for short, which is responsible for gathering data from information sources, clean, transform and conform it in order to store it in a data warehouse. Several designing methodologies for the ETL components have been presented in the last few years with very little impact in ETL commercial tools. Basically, this was due to an existing gap between the conceptual design of an ETL system and its correspondent physical implementation. The methodologies proposed ranged from new approaches, with novel notation and diagrams, to the adoption and expansion of current standard modeling notations, like UML or BPMN. However, all these proposals do not contain enough detail to be translated automatically into a specific execution platform. The use of a standard well-known notation like Relational Algebra might bridge the gap between the conceptual design and the physical design of an ETL component, mainly due to its formal approach that is based on a limited set of operators and also due to its functional characteristics like being a procedural language operating over data stored in relational format. The abstraction that Relational Algebra provides over the technological infrastructure might also be an advantage for uncommon execution platforms, like computing grids that provide an exceptional amount of processing power that is very critical for ETL systems. Additionally, partitioning data and task distribution over computing nodes works quite well with a Relational Algebra approach. An extensive research over the use of Relational Algebra in the ETL context was conducted to validate its usage. To complement this, a set of Relational Algebra patterns were also developed to support the most common ETL tasks, like changing data capture, data quality enforcement, data conciliation and integration, slowly changing dimensions and surrogate key pipelining. All these patterns provide a formal approach to the referred ETL tasks by specifying all the operations needed to accomplish them in a series of Relational Algebra operations. To evaluate the feasibility of the work done in this thesis, we used a real ETL application scenario for the extraction of data in two different social networks operational systems, storing hashtag usage information in a specific data mart. The ability to analyze trends in social network usage is a hot topic in today's media and information coverage. A complete design of the ETL component using the patterns developed previously is also provided, as well as a critical evaluation of its usage.

Keywords: Data Warehousing Systems, Extract-Transform-Load Processes, Relational Algebra, ETL Conceptual and Logical Modeling, ETL Patterns, BPMN.

Resumo

As Tecnologias da Informação têm sido um dos principais catalisadores na revolução que se assiste nas tomadas de decisão na maioria das organizações. A quantidade de dados que são angariados e processados através do uso de dispositivos computacionais tem crescido diariamente, tornando-se uma fonte de informação valiosa para os decisores que gerem todo o tipo de organizações, públicas ou privadas. Concentrar o conjunto ideal de dados num repositório centralizado e unificado, como um *data warehouse*, é essencial para a construção de um sistema que servirá de suporte aos processos de tomada de decisão que necessitam de factos. No entanto, a complexidade associada à construção deste repositório e de todos os componentes que caracterizam um sistema de *data warehousing* é extremamente desafiante. Um dos componentes mais críticos de um sistema de *data warehousing* é a componente de Extração-Transformação-Alimentação (ETL) que lida com a extração de dados das fontes, que limpa, transforma e concilia os dados com vista à sua integração no *data warehouse*. Nos últimos anos têm sido apresentadas várias metodologias de desenho da componente de ETL, no entanto estas não têm sido adotadas pelas ferramentas comerciais de ETL principalmente devido ao diferencial existente entre o desenho concetual e as plataformas físicas de execução. As metodologias de desenho propostas variam desde propostas que assentam em novas notações e diagramas até às propostas que usam notações *standard* como a notação UML e BPMN que depois são complementadas com conceitos de ETL. Contudo, estas propostas de modelação concetual não contêm informações detalhadas que permitam uma tradução automática para plataformas de execução. A utilização de uma linguagem *standard* e reconhecida como a linguagem de Álgebra Relacional pode servir como complemento e colmatar o diferencial existente entre o desenho concetual e o desenho físico da componente de ETL, principalmente devido ao facto de esta linguagem assentar numa abordagem procedimental com um conjunto limitado de operadores que atuam sobre dados armazenados num formato relacional. A abstração providenciada pela Álgebra Relacional relativamente às plataformas de execução pode eventualmente ser uma vantagem tendo em vista a utilização de plataformas menos comuns, como por exemplo *grids* computacionais. Este tipo de arquiteturas disponibiliza por norma um grande poder computacional o que é essencial para um sistema de ETL. O particionamento e distribuição dos dados e tarefas pelos nodos computacionais conjugam relativamente bem com a abordagem da Álgebra Relacional. No decorrer deste trabalho foi efetuado um estudo extensivo às propriedades da AR num contexto de ETL com vista à avaliação da sua usabilidade. Como complemento, foram desenhados um conjunto de padrões de AR que suportam as atividades mais comuns de ETL como por exemplo *changing data capture, data quality enforcement, data conciliation and integration, slowly changing dimensions* e *surrogate key pipelining*. Estes padrões formalizam este conjunto de atividades ETL, especificando numa série de operações de Álgebra Relacional quais os passos necessários à sua execução. Com vista à avaliação da sustentabilidade da proposta presente neste trabalho, foi utilizado um cenário real de ETL em que os dados fontes pertencem a duas redes sociais e os dados armazenados no *data mart* identificam a utilização de *hashtags* por parte dos seus utilizadores. De salientar que a deteção de tendências e de assuntos que estão na ordem do dia nas redes sociais é de vital importância para as empresas noticiosas e para as próprias redes sociais. Por fim, é apresentado o desenho completo do sistema de ETL para o cenário escolhido, utilizando os padrões desenvolvidos neste trabalho, avaliando e criticando a sua utilização.

Palavras-Chave: Sistemas de *Data Warehousing*, Processos de Extração-Transformação-Alimentação, Álgebra Relacional, Modelação Concetual e Lógica de ETL, Padrões ETL, BPMN

Table of Contents

List of Figures

List of Tables

Acronyms

3NF	Third Normal Form
BPEL	Business Process Execution Language
BPMN	Business Process Modeling Notation
CDC	Change Data Capture
DBMS	Database Management System
DCI	Data Conciliation and Integration
DQE	Data Quality Enforcement
DSS	Decision Support Systems
DW	Data Warehouse
DWS	Data Warehousing System
ER	Entity–relationship
ETL	Extract-Transform-Load
IT	Information Technology
LDL	Logic Data Language
ODS	Operational Data Store
OLTP	On-Line Transaction Processing
QoX	Quality of eXperience
RA	Relational Algebra
RC	Relational Calculus
RCD	Rapid Changing Dimensions
SCD	Slowly Changing Dimensions
SKP	Surrogate Key Pipelining

SQL	Structured Query Language
UML	Unified Modeling Language
VSN	Virtual Social Network
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 Context Overview

It is commonly witnessed in today's enterprises to an unbridled search for new solutions sponsored by management to improve the products or services offered in order to distinguish themselves in a competitive market. Enterprises are making significant efforts in the search of new markets, domestic and abroad, to maximize exposure and contribute to a sustained and persistent growth. The development of new and better products and services and the expanding of activities to new markets have contributed to a significant increase in the quantity of information created, treated and stored. However, in most of the cases, the bigger the quantity of data to treat the higher are the resources consumed in the process and the slower is the response time to typical daily requests. Consequently, one of the solutions enterprises adopt and invest is the computerization and automation of business processes.

Information Technology (IT) revolutionized the way enterprises do business, starting with automating processes, and then simplifying tasks like registering data, printing documents and controlling machines. With the intensive development of IT, new businesses and working methods have revolutionized the day-by-day operations having a clear impact on cost reduction. However, these new technologies, which involve frequently high investment costs, are not always fully exploited due to the lack of knowledge endangering consequently enterprise's future. Using IT in the most adequate manner can benefit the enterprise by conceding leverage over its direct competitors due to product cost reduction or service improvement expanding therefore its customer base and market share, or even a higher rate of return. With this in mind, throughout the years, enterprises have been investing in software applications in order to potentiate business, gathering and treating pertinent information.

The proliferation of software applications, primarily transactional systems, has resulted in a continuous accumulation of data, which does not necessarily reveal an increase of knowledge about the business. One major reason has to do with the fact that these applications have been developed independently, and were not built with a global data model in mind which provokes redundancy and inconsistency of information. The latter case is the main cause for not exploiting the potential of IT. It is a real problem for business as it puts into question the entire operational support for decision-making processes. Another reason is that transactional systems, by definition, are oriented exclusively to serve corporate operational needs, recording and processing the company's day-by-day operations that guarantee a smooth running for the company. However, by nature, these systems are not very suitable for carrying out data analysis tasks, data correlation or even reporting business evolution between different time periods.

The emergence of *Decision Support Systems* (DSS) came to fill some gaps providing the ability for exploring data stored in existing transactional systems. Usually, DSS are developed in querying oriented platforms, having objectives such as helping or counseling decision-making process, providing a specialized data repository, and facilitating querying development. However, the development of DSS within enterprises did not follow a data integration structured plan, which caused many of these systems to be based on only a portion of the existing data of the enterprise, thus once again the existence of different "truths" about the same subject. This incongruity between reports or analysis made in different DSS only revealed that these systems are not intended to support the decision making process of managers of an enterprise.

One way to solve the data integration from different systems or applications is the development and implementation of *Data Warehousing Systems* (DWS) in an enterprise. DWS emerged as an natural addition to the operational systems of the companies, since they are specifically oriented to data analysis, providing an integrated repository of historical data - a data warehouse - instead of the operating systems that are oriented for transaction processing and business activity support. Thus, a DWS allows for receiving large volumes of information from different sources to be extracted, processed, and stored in a single data repository hence enabling the development of faster and more effective analysis processes.

A DWS is intended primarily for users with decision-making capacity - the decision makers - about the future of the company, which with the help of specialized tools can analyze patterns and variations in the data, thereby extracting the information needed to respond in timely basis to the various issues and develop the basis for the daily support of its decision-making processes. Since these systems store data from various sources of information, you can also correlate the various types of data that so far, only with transactional systems, were a little difficult to correlate in the information system. Given the characteristics of a DWS, enterprises began to think about the adoption of such a system as an investment in the future. Some advantages are pretty obvious, namely:

- having the potential to serve as decision support systems;
- allowing for the creation of a centralized and conformed data storage system;
- providing a robust basis for data analysis tools.

However, investing in this type of system requires large financial resources since there are components of the system that are expensive. This cost is sometimes a factor quite inhibitor in the adoption of these systems by small and medium-sized enterprises, because they have less money available to consider such prohibitive investment. On the other hand, large enterprises are also faced with problems affecting the development of these systems, which are time and the volume of data to be processed. As the volume of information to be processed periodically increases, the *Extract-Transform-Load* (ETL) task consumes more and more resources. Since there may be limits on the time frame reserved for this task, the solution is to increase the computing power so that you can run the task in the time required. The increased computing power has been achieved usually by acquiring computers with increased performance which are even more expensive. However, this solution may not be feasible for all enterprises. One possible solution to overcome the problems listed above is to take advantage of the computing power of the computers on the local networks of enterprises which in the vast majority of the time are available and untapped. If used, this computational power will not increase significantly the cost of the system and might in turn even reduce it since these computers were already acquired by the enterprise. The use of these computers with parallelism might also contribute to reduce the time required to execute the ETL. However, commercial ETL tools are not prepared to use local network computers to execute ETL tasks, since they are not able to split and distribute data and tasks over existing computing nodes, even when their modeling capabilities are becoming a handicap due to the use of proprietary notations.

1.2 Motivation

The use of proprietary notations in modeling ETL systems is discouraged. With time it will act as a reason to block changes in the ETL system. ETL commercial tools are expensive and their adoption should be highly weighted, especially in their ability to accept and import diagrams made in standard languages. It is our belief that design and execution should be different platforms in a choice for more flexibility in the development and maintenance of an ETL system. Bearing in mind that the cost of developing a DWS is still prohibitive for some enterprises, mainly for small and mid-sized ones, and that the greater amount of resources is spent in the development of the ETL system, we propose a different approach for its development. The idea of minimizing the cost of this critical DWS component will in turn contribute to the adoption of such systems by financially constrained enterprises which keep postponing their DWS projects due to their exacerbated cost.

In order to develop an ETL system that will be flexible, scalable and reliable, and above all, cheap, the use of the computational resources already present in the enterprise is of superior interest. Concepts like Grid computing (Foster et al., 2001) and Cloud computing (Vouk, 2008) are still in the order of the day. Yet, both approaches have the same goals - distribution and resource sharing - with the focus shifting from a technology point of view to a service oriented philosophy. The use of grid environments in data processing intensive tasks has been studied in academic and scientific institutions already for a long time. Commercial organizations are now adopting this approach to help the mitigation of the impact of the increased amount of data gathered by their enterprise information systems that needs to be analyzed, through testing grid middleware software. The basic idea is to take advantage (or to attenuate the effect) of the inactivity of their computing devices during a regular day helping them to perform those tasks. A grid based approach maximizes the investments already made and postpones some other expensive computer acquisitions (Demiya et al., 2008).

One of the objectives of an enterprise information system is to support management's decision-making processes through the analysis of large amounts of data stored in them. Since data increases normally through time, the processing power needed to analyze it in a useful time also increases undermining the infrastructure available. A grid environment might be a suitable solution to this kind of problem, due to the easiness and inexpensiveness of adding new processing nodes to the infrastructure (Poess and Nambiar, 2005, Wehrle et al., 2005). The challenge, however, is to bridge the gap between this kind of execution platform and the ETL design phase. The absence of a DBMS to execute normal queries that transforms data or the heterogeneity of the computing nodes present in a Grid environment are also challenges that must be taken into account in the development of the ETL system. By using a scalable technical architecture like a Grid architecture and a modeling approach that uses a standard notation/language, all ingredients are set for a Low Cost ETL System (Santos et al., 2014).

In short, the motivation behind this work is based in the need to find a different approach to the logical design of the ETL System that could serve two purposes: to bridge the gap between conceptual modeling and execution platforms, and to be platform independent thereby opening the possibility of different technological architectures as execution platforms.

1.3 Scope of the Thesis

Although in the previous section a possibility was stated as a hypothesis for the technological infrastructure of an ETL System, a Grid architecture, the challenges that exist in the use of this kind of environment to execute a critical DWS component like the ETL system are many and varied.

Outside the scope of this thesis are problems related to the Grid architecture like for instance node communication, bandwidth management, node evaluation and control of task flows. However, and considering that such architecture is available and valid, taking advantage of it to execute the ETL process is also a challenge mainly due to the question "what are we going to send to the computing nodes?"

Due to the inexistence of a centralized or local DBMS, the use of SQL is not recommendable, therefore data transformations must be made using a different grain. The idea is to use the finest grain available when dealing with data stored in a relation, which is a *Relational Algebra* (RA) operation. These operations can be distributed and executed over sent data and once the transformation concludes the results are sent back to the central computational node. Bear in mind that the RA language is part of the foundation of the Relational Model (Codd, 1970), is a widespread acknowledged standard that has been used very successfully over the last 40 years in the database community. Moreover, one of the primary characteristics of RA Language is the fact that it is a procedural language. Operations are executed in a pre-determined order which is essential when dealing with flows of data transformations which is the base for ETL processes. One of the advantages of this characteristic is the ability to analyze the flow of operations generated in search for possible optimizations and therefore minimize processing time and data. Another advantage is the ability to narrow the gap between the modeling phase and execution phase, since the interpretation and translation of RA operations into execution platforms is easy and common. RA language is comprised of a limited set of unary and binary operators with a clear logic of use thus easing the translation process into the execution platform.

With this in mind, the challenge shifted to the modeling capabilities of the RA language. If we were to use RA as a modeling language for the ETL system, the major ETL tasks should be studied and modelled as patterns that could be instantiated when needed. Patterns will be larger modeling entities in terms of grain, which will simplify ETL models and encapsulate process and task complexity. As such, the scope of this thesis is to propose RA patterns to model common ETL tasks, analyze their usability in a real ETL scenario, and evaluate the feasibility of the proposal.

However, there are already some proposals for ETL modeling worth studying, both in the conceptual phase of the design and also in the logical phase. These proposals base themselves either in new methodologies and diagrams or in standard modeling languages that are extended or adapted. Nevertheless the majority of the proposals fail at identifying in detail the operations needed to execute each ETL task, especially at the logical level design, leaving a gap between the modeling phase and execution phase that must be addressed.

1.4 Main Objectives

An ETL system is a crucial component of a DWS, and therefore subject of intensive research by the data warehousing community. There are already several ETL commercial tools available in the market, which in turn are becoming even more efficient and capable. Nevertheless their modeling capabilities tend to be supported by proprietary notations with limited or no ability to interpret and import different diagrams, even from standard notations. Moreover, the academic community has also been presenting ETL modeling methodologies and notations that in fact are not supported by commercial tools. The gap between conceptual methodologies and ETL commercial tools lies in the lack of a more formal approach in the logical phase of the design. Common ETL tasks are not formally defined in the logical perspective of the flow of operations and the use of RA language can bridge the existing gap. Therefore, the main objective of this thesis is to model patterns of common ETL tasks using an extended RA language. To achieve this, an in-depth study of the most cited ETL modeling methodologies will be conducted, with particular emphasis on the conceptual and logical phases of the ETL system without disregarding the physical one. In addition, an evaluation of the feasibility of the RA operators over common ETL tasks will also be studied to better comprehend the application of each RA operator in this specific domain. The combination of these studies will allow for a formal specification of the ETL tasks, more specifically, to model some common ETL tasks as patterns, namely:

1. *Changing Data Capture* (CDC) as the primary operation to represent source data in the Data Staging Area;
2. *Data Quality Enforcement* (DQE) focusing in data transformations required to clean, conform and standardize data;

3. *Data Conciliation and Integration* (DCI) that conforms information obtained from heterogenic data sources with misrepresented values, corrects and integrates them in into the repository's tables;
4. *Slowly Changing Dimensions* (SCD) as a means to preserve history in a centralized repository;
5. *Surrogate Key Pipelining* (SKP) to finalize the process of loading fact data into the data warehouse.

Finally, to demonstrate the application of all the aspects related to the proposed ETL patterns, an application over a real ETL scenario was conducted to better assess the merits of the proposal made on this thesis, and reveal their strengths and weaknesses.

1.5 Organization of the Thesis

In order to expose and discuss appropriately the most relevant scientific and technological aspects covered in this work, we structured the remaining part of this thesis as follows:

- Chapter 2 – **ETL Modeling.** In this chapter it is presented an in-depth study of the most cited ETL modeling proposals for conceptual, logical and physical design of an ETL system. A case scenario was used to demonstrate the use of each proposal with a final comparison of all the approaches presented and discussed.
- Chapter 3 – **Relational Algebra Approach**. This chapter contains an in-depth study about Relational Algebra, and its extended version. In this study, several excerpts of the case scenario presented in chapter 2 were used to demonstrate the use and application of the RA language in common ETL tasks.
- Chapter 4 - **ETL standard processes specification**. As a result of the research made in the previous two chapters, several ETL sub processes are studied and modeled as RA expressions patterns.
- Chapter 5 - **Specifying a Real ETL System**. The application of the patterns proposed previously over a real ETL scenario is conducted in this chapter. Two heterogenic sources

and one data mart are used to demonstrate the feasibility of the patterns proposed in chapter 4, i.e, the instantiation of each pattern to a real ETL modeling system.

- Chapter 6 - **Conclusions and Future Work**. To end this work, a critical analysis of this thesis' proposal is presented in comparison with the most cited ETL modeling proposals for a low cost ETL approach, presenting when appropriate some of the publications accomplished in several of the topics approached in this thesis. Finally, we suggest some possible future lines for future researching and application.

Chapter 2

ETL Modeling

Designing and implementing a *Data Warehousing Systems* (DWS) is a complex and burdening task, prone to errors, and with an overwhelming failure rate. These facts lead to the presentation of several methodologies in order to minimize the risk of failure and maximize the projects outcome. Inmon, who coined the term *Data Warehouse* (DW), presented in 1994 his view of a DWS development (Inmon and Hackathorn, 1994, Inmon, 2005), whose concept was extended latter to a higher degree with its *Corporate Information Factory* (Inmon et al., 2001). This approach focused in the design and implementation of a single data warehouse for the entire company, using the 3NF as the basis to database design. Later, Kimball et al. (1998) presented another perspective for data warehousing development, referred in the area as *"a data warehouse bus architecture"*, defending the design and implementation of distinct parts of a DW, as autonomous components (data marts), covering each one different business and decision-making processes. Together, such components will form the company's DW. In this methodology, it was also presented the foundation of dimensional modeling as the basis to define a way to store data in a data warehouse. Nevertheless both authors clearly define that a DWS is independent from any other *Online Transaction Processing* (OLTP) system. Usually, OLTPs act as information sources providing data to populate a DW. This process, also called Extract, Transform and Load (ETL) process, is one of the most critical components of a DWS, consuming about 60-70% of the time spent in the development of the entire DWS.

The data warehousing research community has been focusing their attention in the different components of the development of a DWS, with particular emphasis in the final stages of the DWS, i.e. data analysis, data mining and data querying. Nevertheless, other DWS issues have also been studied and developed, like the ones related to dimensional modeling and management or ETL processes specification and validation. This chapter presents an in-depth study and analysis about ETL specification and validation, using for that purpose a very typical ETL task as working example: the Slowly Changing Dimension case.

2.1 Managing the ETL process

In Inmon's *Corporate Information Factory* (Figure 1), the data warehouse is mainly served with integrated data treated by the Integration/Transformation layer. This layer is responsible for accessing and extracting data from various source systems (or applications), combining and transforming it (when necessary) into corporate data, loading it into an *Operational Data Store* (ODS) for further analysis before being transferred to the data warehouse.

Inmon characterizes this layer has an "unstable set of programs" mainly due to the fact that any DWS is an evolving system that integrates an ever-increasing amount of data from evolving sources and applications. Nevertheless, one of the most important tasks assigned to this layer is the correct representation of Metadata, i.e., the steps and transformations done to the data, giving the possibility to analysts to understand the correct meaning and origin of the data being analyzed. The management of this layer should be attributed to a data warehouse administrator, being this the best profile to handle all the particularities associated with the I&T layer. On the other hand, Inmon identifies two types of ETL tools: 1) those that produce code and can access legacy data on its own; and 2) those that produce a parameterized run-time module that can only access flattened legacy data in order to clean, conform and integrate data into the ODS.

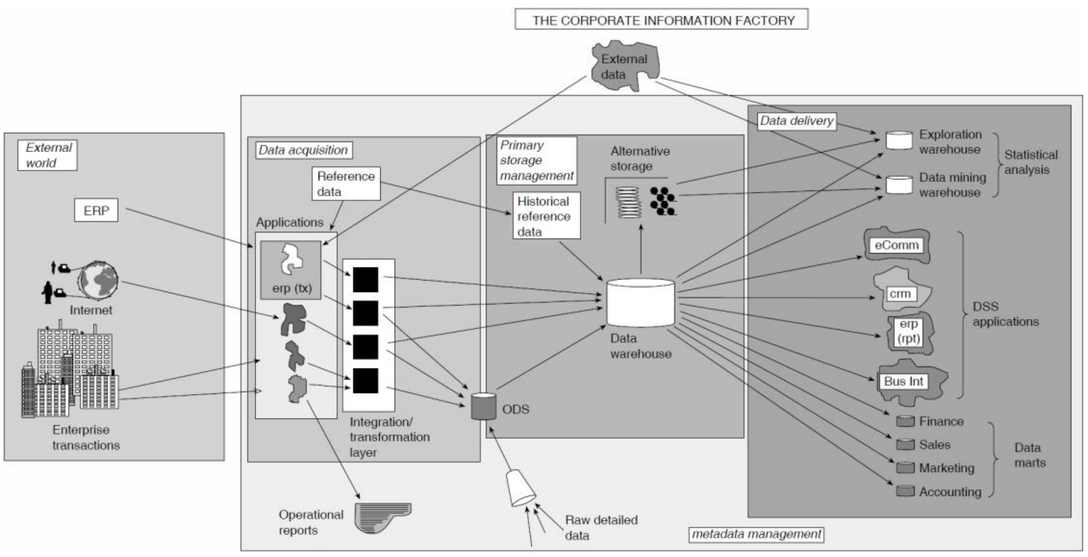


Figure 1 - Inmon's *Corporate Information Factory* architecture - extracted from (Inmon et al., 2001)

Ralph Kimball, another recognized data warehouse practitioner's guru, proposes an integrated approach for DWS design and development. In his point of view, a data warehouse is the "sum" of all the data marts developed over a common ground of requirements, referring such view as *"a data warehouse bus architecture"*. In order to succeed in this approach, Kimball also proposed a management methodology (Figure 2). This methodology, the *Business Dimensional Lifecycle*, has three parallel branches: the top one deals with the DWS' infrastructure; the middle branch approaches the design and development of the data warehouse and all tasks necessary to populate it; and the bottom branch is related to the design and development of the application's interface for users. All tasks in every branch are supported by a diversified group of tabular forms identifying the role and tasks of every stakeholder involved, providing more control in the DWS project management.

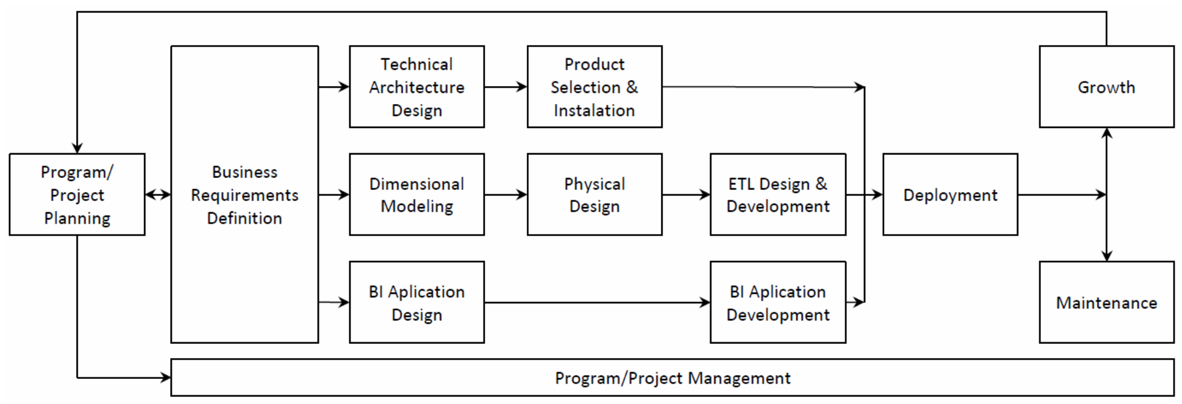

Program/ Project Planning
Business Requirements Definition
Technical Architecture Design
Product Selection & Instalation
Dimensional Modeling
Physical Design
ETL Design & Development
BI Aplication Design
BI Aplication Development
Deployment
Growth
Maintenance
Program/Project Management

Figure 2 - Kimball's *Business Dimensional Lifecycle* diagram - extracted from (Kimball et al., 1998)

He used the same approach for the ETL systems design and development with his proposal of the *ETL Design & Development* task (Kimball and Caserta, 2004). Kimball advises on a series of steps, aggregated in two main threads, i.e., the Planning and Design Thread (Figure 3) and the Data Flow Thread (Figure 4), which are both developed in parallel. The former thread is dedicated to the correct management of the ETL process by dedicating special attention to the requirement analysis, system definition and implementation and finally testing, documenting and system tuning. The latter one details the usual steps taken in the design and development of the ETL tools. Summing up, the planning thread is mainly comprised of schemas and maps that represent the ETL process by identifying data sources, transformations and destinations. The data flow thread, builds and tests each component of the ETL process, giving particular emphasis to SCD problems and surrogate key lookup and substitution.

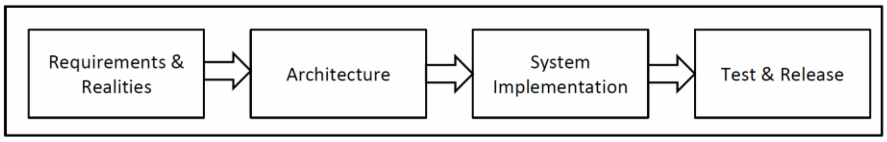


Figure 3 - The Planning and Design thread - extracted from (Kimball and Caserta, 2004)

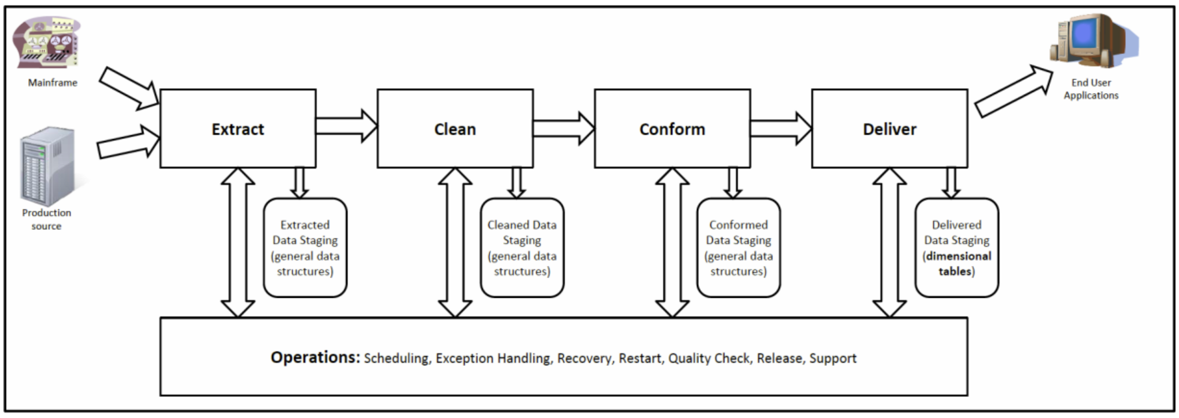

Mainframe
Production source
Extract
Clean
Conform
Deliver
End User Applications
Extracted Data Staging (general data structures)
Cleaned Data Staging (general data structures)
Conformed Data Staging (general data structures)
Delivered Data Staging (dimensional tables)
Operations: Scheduling, Exception Handling, Recovery, Restart, Quality Check, Release, Support

Figure 4 - The Data Flow thread - extracted from (Kimball and Caserta, 2004)

Although both Inmon and Kimball advise on how to manage and build a DWS, the ETL component is dealt frequently using ad hoc tools and a significant lack of formal design and methodology application. The definition and development of an ETL component is responsible for 60 to 70% of the total DWS development efforts, which makes it quite critical. The majority of organizations that have a DWS (Mannino and Walter, 2006), identify the ETL component has a critical DWS component due to its impact in the remainder of the system and constantly in need of optimizations to reduce the time consumed in the refreshment process. With this in mind, a closer look to the proposals presented by the data warehousing research community during the last years will be made, following a chronological approach in order to reflect better the evolution made until now.

2.2 ETL Modeling – Industry Practitioners

In recent years, the design of an ETL component for a DWS has become a strong focus of attention by an increasing number of researchers encouraged by the lack of standards for its development. Another reason stands with the fact that an ETL system is one of the most important components of a DWS thus affecting greatly the success of the DWS itself. Let's first analyze two of the most known data warehousing field practitioners, Inmon and Kimball, and afterwards some proposals coming from important academic researchers in the field.

- **The Inmon's Approach**

Chronologically speaking, only after coining the term Data Warehouse, Inmon started presenting his view and approach to the development of a system that would support its new concept. The mechanisms used to gather, clean and conform data for deliver to the data warehouse were called, as stated previously, ETL tools or processes. However, no design methodology was specified. It was considered, that these tools, or pieces of software, were developed as needed, specifically oriented to its application field. Many of these procedures were then simple database queries.

- **The Kimball's Approach**

On the contrary, Kimball proposed a very complete and extensive methodology for the design and implementation of a DWS, with some particular focus on the design of ETL processes (Kimball and Caserta, 2004). Nevertheless, his approach for ETL design is very ad-hoc, without using any formality or known notation. Figure 5 presents an adapted diagrammatic example of the high-level design of an ETL process. If necessary, an in-depth design might be elaborated in order to best represent each source-to-target flow.

Kimball then resumes the design of an ETL process to a table filled with properties of the tools used in the process in addition to the identification of all objects involved. It also gives a series of best practices, tips and tricks, about how to develop better tools for ETL processes.

2.3 ETL Conceptual Modeling

One of the first proposals for ETL modeling was based on the independence between different layers. First we develop the conceptual design then the logical design and finally the physical design, as it is normally used in database design methodologies. Here, the most important proposals of conceptual models from data warehousing researchers are presented in a chronological order, using as a working example a part of an ETL process that loads some data into a dimension table with history, representing customers (Figure 6) of an OLTP system (Microsoft's Adventure Works Sample Database)[1].

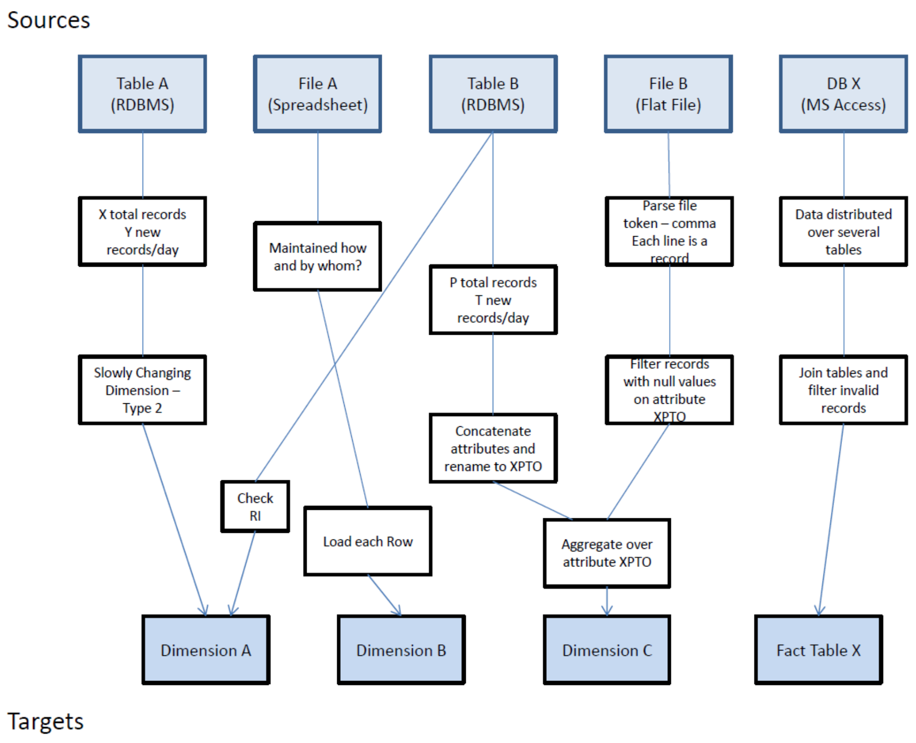

Sources
Table A (RDBMS)
File A (Spreadsheet)
Table B (RDBMS)
File B (Flat File)
DB X (MS Access)
X total records Y new records/day
Maintained how and by whom?
P total records T new records/day
Parse file token – comma Each line is a record
Data distributed over several tables
Slowly Changing Dimension – Type 2
Filter records with null values on attribute XPTO
Join tables and filter invalid records
Concatenate attributes and rename to XPTO
Check RI
Load each Row
Aggregate over attribute XPTO
Dimension A
Dimension B
Dimension C
Fact Table X
Targets

Figure 5 - ETL high-level design - adapted from (Kimball et al., 1998)

1 http://msftdbprodsamples.codeplex.com

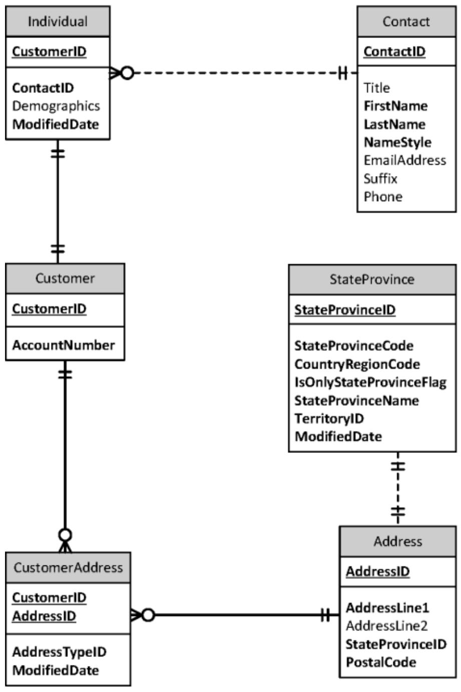


Figure 6 – An excerpt of the Adventure Works Database focusing on the table "Customer" and its related tables

The dimension *Customers* is typical recognized as a SCD in which changes to OLTP customer's data must be reflected into the data warehouse with or without preserving their history. In this example, Customers' dimension (Figure 7) data comes from several OLTP tables. We assumed that a dimension integrates two identical tables: one storing current data (the dimension table itself) and the other storing historical data (the historical table). This approach will offer us the ability to preserve all history without the overburden of extending the actual size of current data or adapt the dimension table structure, since historical data is stored in a separate table, having the advantage of requiring only a simplified task for detecting and storing changed data (Santos and Belo, 2011a). Ordinary browsing data operations over the dimension table works quite well once we keep historical data separated from regular dimension data. This does not happen in a SCD type 2.

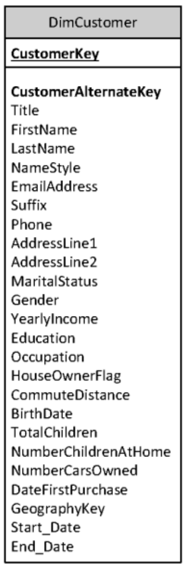


Figure 7 - Data Warehouse Dimension Customer

2.3.1 A Novel Proposal from Vassiliadis and Simitsis

Over the last ten years or so, Vassiliadis et al. (2002a) have been researching and evolving a design approach for the ETL component of a DWS. Their first proposal was based on the need to correctly identify the mapping between attributes of data sources and attributes of data warehouse tables. Therefore, they define a graphical notation (Figure 8) and the corresponding *Unified Modeling Language* (UML) metamodel with the aim of documenting and formalizing the ETL process in a customizable and extensible approach. The primary reason for this approach, instead of using *Entity–relationship* (ER) modeling or conventional UML modeling, was the fact that, in the ETL environment, attributes needed to be treated as *first-class citizens*. In ER or UML modeling the attributes are part of an entity or class, and this approach is not compatible with the transformations needed to be done at the attribute level in an ETL environment.

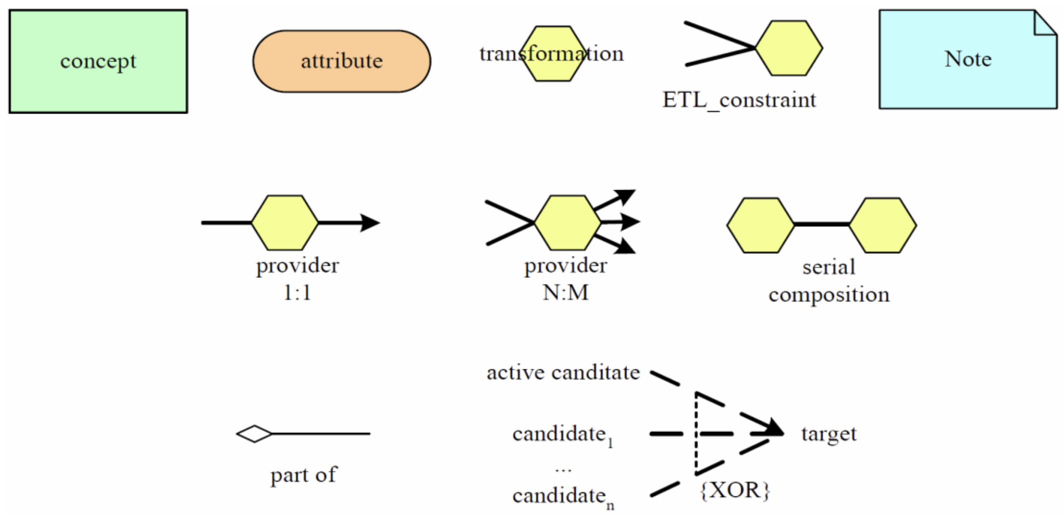


Figure 8 - Vassiliadis' notation for the representation of the conceptual model of ETL activities - extracted from (Vassiliadis et al., 2002a)

In subsequent work, Simitsis and Vassiliadis (2003) detailed a methodology for the usage of the previous conceptual model. This methodology is comprised in four steps.

1. Identification of the proper data stores;
2. Candidates and active candidates for the involved data stores;
3. Attribute mapping between the providers and the consumers;
4. Annotating the diagram with runtime constraints.

The first two steps are normally executed in the requirements and analysis stage. The ETL designer knows the data warehouse in detail and therefore must correctly identify adequate data stores, and if many arise as complimentary, he must, in step 2, choose which ones to use and how. Step 3 is the most burdening task, since the designer needs to represent the mappings between source attributes and destination attributes. This task may involve source administrators with the purpose of eventually clarifying source data codes, their access restrictions and their meaning. Finally, in the last step, all information particular to the execution and restrictions of the ETL activity must be annotated to better document the process. It also serves the purpose of feeding information to the next step of the modeling phase.

In Figure 9 we present a high-level example of the use of this notation, where we identify the main data sources used to load the data warehouse dimension *DimCustomer*.

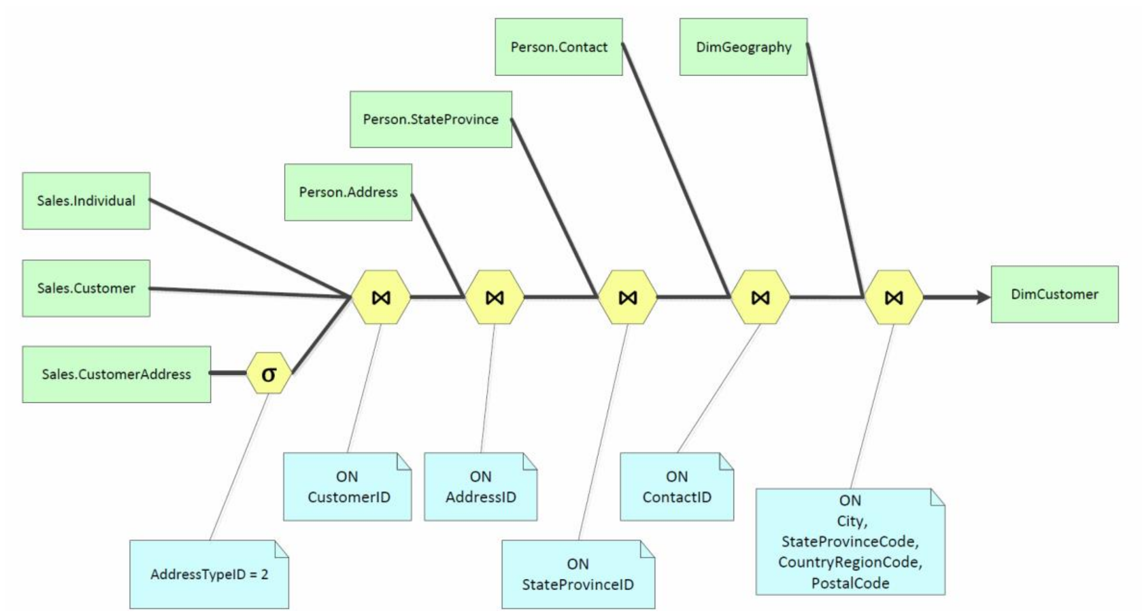


Figure 9 - High-level example of ETL activities represented using Vassiliadis' notation

A lower-level detail of this diagram (Figure 10) presents mappings between source and target attributes as well as correspondent transformations. Notice the denormalization of an XML attribute into several attributes of the dimension *DimCustomer*, several data type conversions and the use of a function to timestamp the date from which the tuple is valid.

As already referred, the approach to store data in a dimension with history is obtained generically through the use of two base tables. One stores current data, which we normally address as the dimension itself, and another table that stores historical data, which represents the old values of current data. In Vassiliadis' proposal there exists a transformation that typically applies to attributes identified as SCD Type 1, 2 or 3. However for the example chosen there is no clear graphical notation to use, mainly because the steps needed to maintain the dimension and its history up to date are less conceptual and more logical.

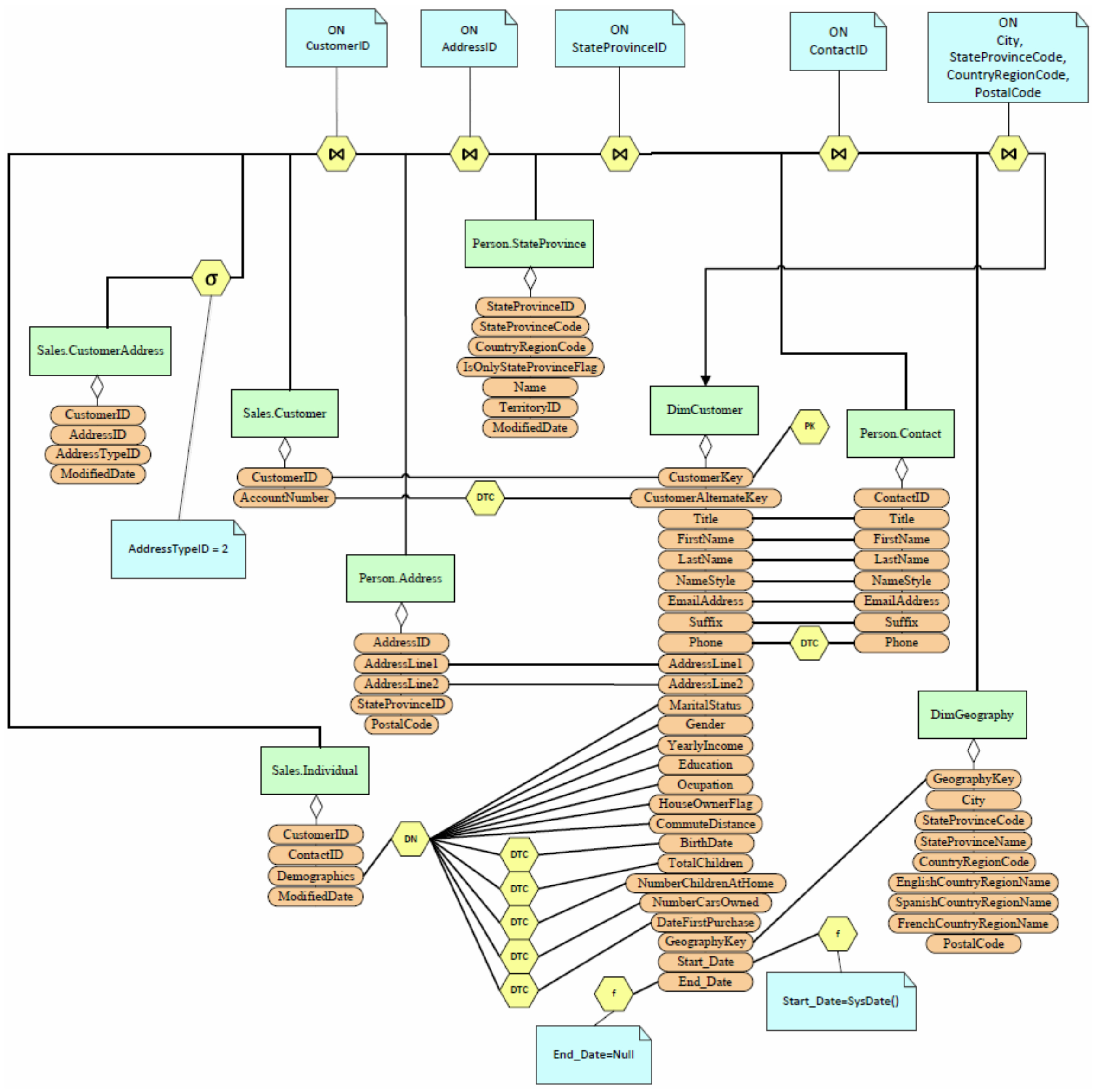


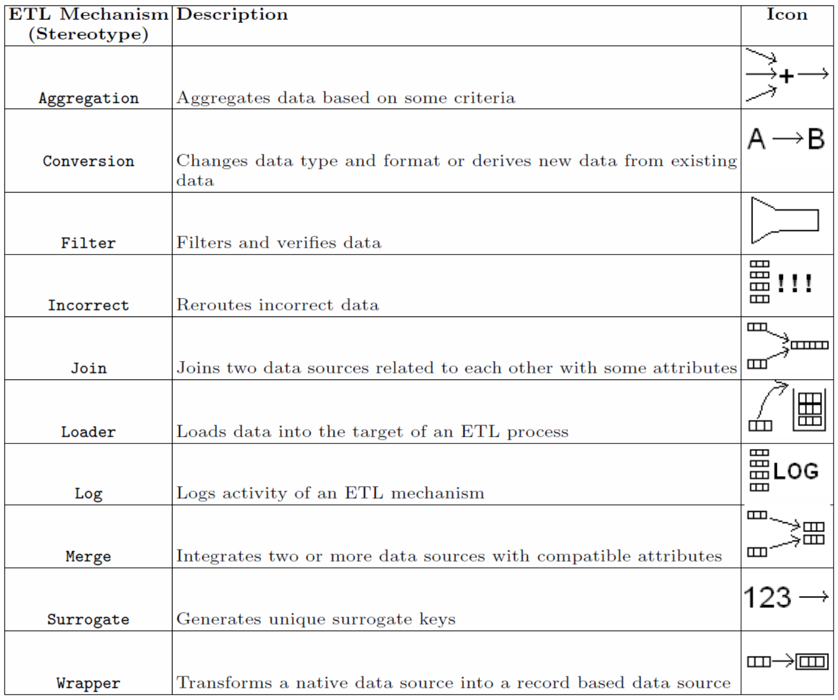
Figure 10 - Low-level example of ETL activities represented using Vassiliadis' notation

2.3.2 The Trujillo and Luján-Mora UML approach

Other data warehousing researchers that presented some work over the ETL modeling phase were Juan Trujillo and Sergio Luján-Mora. Their view over the conceptual modeling of the ETL process was opposite to the Vassiliadis' view. As studied previously, Vassiliadis proposed a novel approach to the design of the conceptual model of the ETL process, particularly presenting a new graphical notation. Trujillo on the other hand used UML and extended it with mechanisms and icons (Figure 11) to better represent the ETL model (Trujillo and Luján-Mora, 2003). The basis to this approach was to use a world accepted notation, known by most software and database designers, and therefore remove the need to learn a new notation and reduce the time spent on the design of the ETL.

As it can be seen in Figure 11, Trujillo's UML extension proposal is based in common ETL activities. The filtering, conversion, aggregation and loading of data are some of the tasks performed frequently in an ETL process. In Figure 12 it is presented the first part of how the working example chosen is represented in UML using the proposed mechanisms and icons. This part gathers the data from different table sources, joins it and transforms it into a final table, identified as *Customer_Final*.

This table is then used to load data into Customer's dimension using two separate operations (Figure 13). The *Insert* operation only loads new data, discarding unchanged and updated data to an auxiliary table. This auxiliary table is used with the *Update* operation that only updates existing data. The primary problem with the use of this notation is that it is not possible to represent the necessary operations to maintain a historical table of a dimension as needed by the working example. There is no graphical notation to represent the transfer of old data (previous values of update data) into the historical table. Also, Trujillo's approach does not provide any graphical representation for the other, more common, SCDs which are one of the most common data warehouse requirements.



ETL Mechanism (Stereotype)	Description	Icon
Aggregation	Aggregates data based on some criteria	+ →
Conversion	Changes data type and format or derives new data from existing data	A → B
Filter	Filters and verifies data	
Incorrect	Reroutes incorrect data	!!!
Join	Joins two data sources related to each other with some attributes	
Loader	Loads data into the target of an ETL process	
Log	Logs activity of an ETL mechanism	LOG
Merge	Integrates two or more data sources with compatible attributes	
Surrogate	Generates unique surrogate keys	123 →
Wrapper	Transforms a native data source into a record based data source	

Figure 11 - Trujillo's UML ETL mechanisms and icons – extracted from (Trujillo and Luján-Mora, 2003)

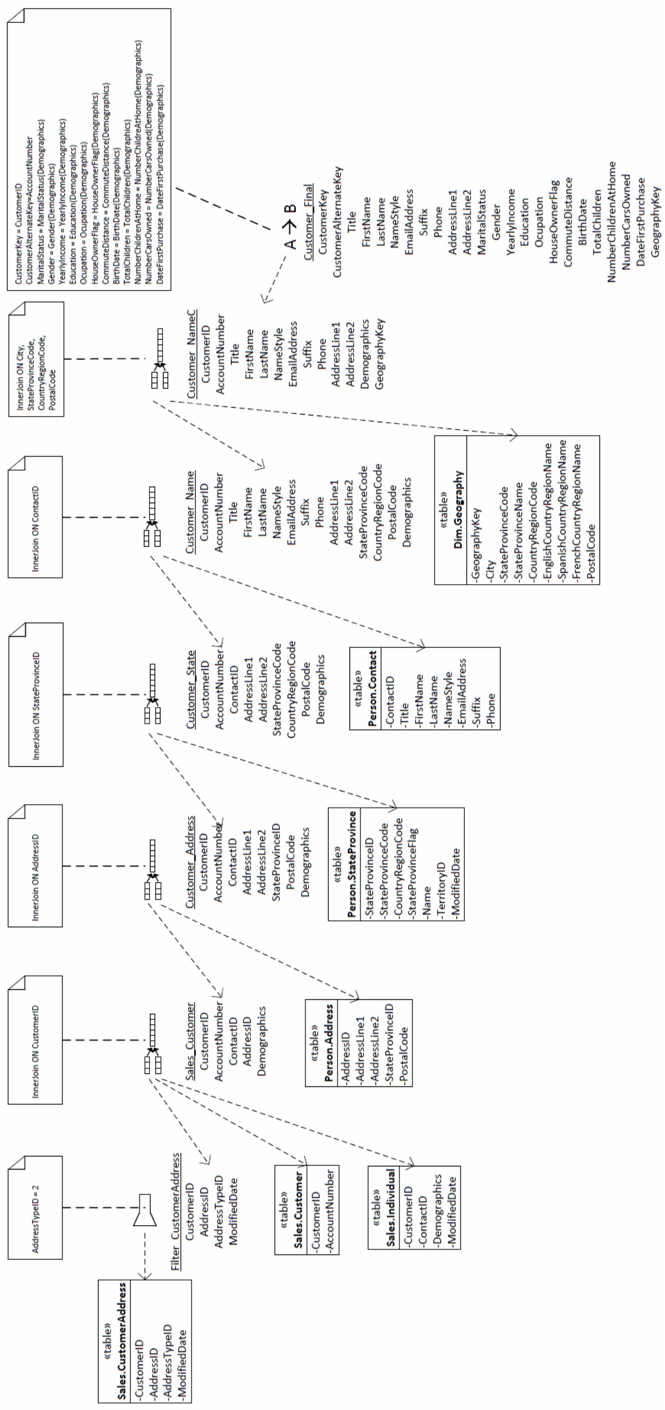


Figure 12 - Working example of Trujillo's proposed UML extension (Trujillo and Luján-Mora, 2003)

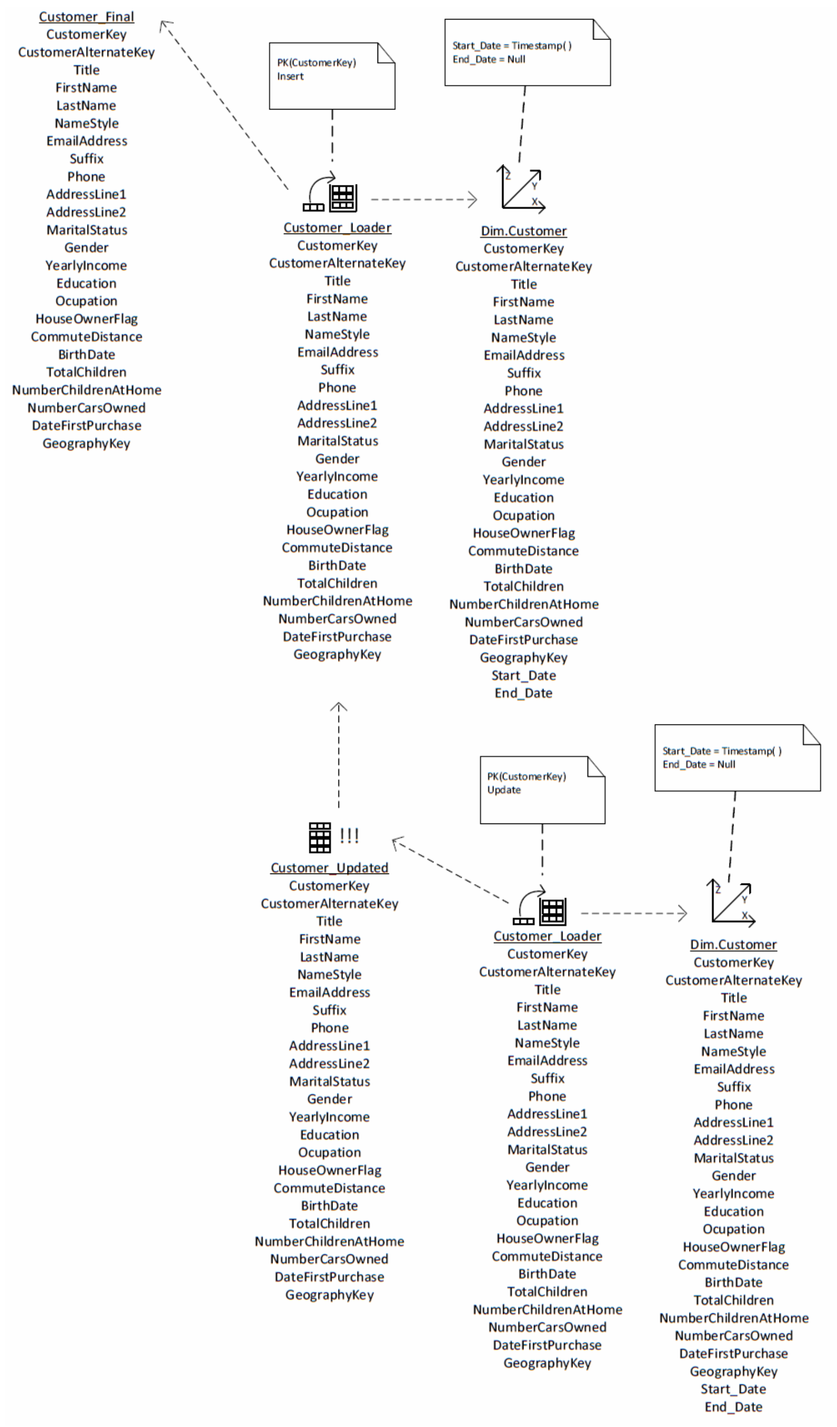


Figure 13 - Loading data into the dimension "*Customer*"

2.3.3 The Mixed Approach from Luján-Mora, Vassiliadis and Trujillo

As a direct consequence of a research partnership, Luján-Mora, Vassiliadis and Trujillo proposed a framework for modeling ETL processes using extended UML supporting attributes as first-class citizens (Luján-Mora et al., 2004). The idea is to represent the process as a multi-level UML diagram that can be drilled-down to more detailed levels, extending the representation of data mappings between attributes presented in the ETL process. The concept of data mapping is used to represent shared information at the various levels (database, table or attribute), detailing the transformations necessary between source data and their target. Figure 14 presents an example of the detail of each of the four levels proposed. In the first level, Level 0, source schemas and target schemas are identified along with a high level data mapping package. In the second level, Level 1, the data mapping package is detailed to represent each target table data flow. In the third level, Level 2, the dataflow level is detailed to represent how each target table relates to the source tables in terms of data transformations. In the fourth level, Level 3, the data transformations are detailed to the attribute level identifying the sequence of transformations and cleaning tasks to be performed.

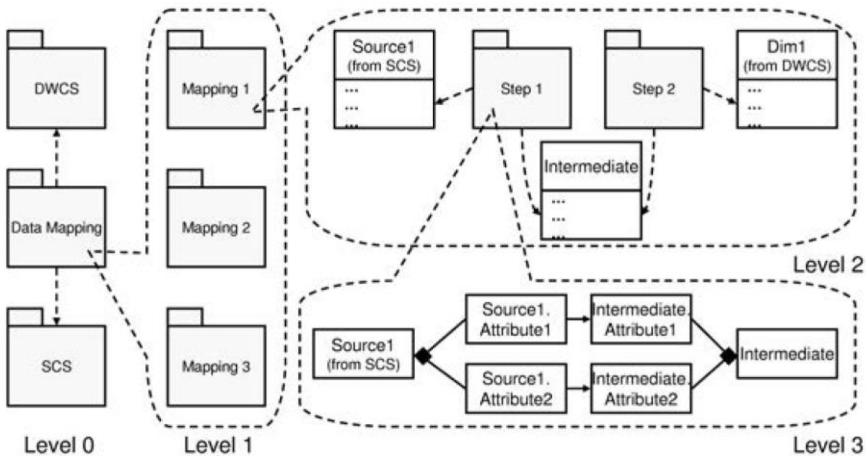


Figure 14 - Data mapping levels – extracted from (Luján-Mora et al., 2004)

2.3.4 Conceptual Modeling using UML Activity Diagrams

Following the proposal of modeling ETL processes using UML (Trujillo and Luján-Mora, 2003), Lilia Muñoz develops the concept and characterizes a lower level of this approach by defining the dynamic aspects and behavior of the ETL processes (Muñoz et al., 2008). The proposed modeling framework has a higher level where the elements of the ETL process are specified, like sources, targets and the ETL process itself. At the lower level, the ETL process is specified in terms of flows of activities using metaclasses of UML Activity Diagrams (Figure 15).

In this proposal all ETL mechanisms used by Trujillo (Trujillo and Luján-Mora, 2003) are characterized to present a set of customizable templates to be used in the conceptual model. Figure 16 presents a generic model of a reusable activity in a ETL process.

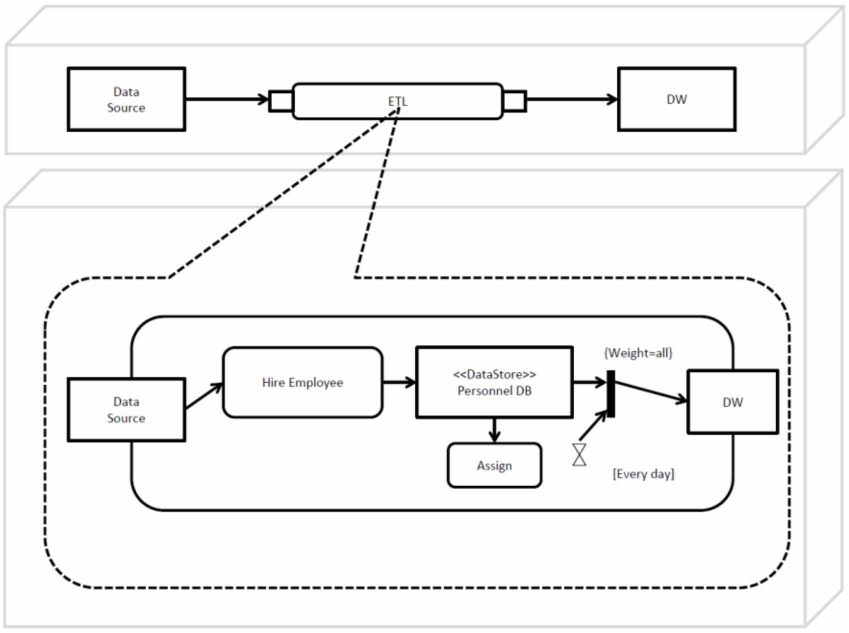


Figure 15 - Muñoz' two level modeling framework using UML AD – extracted from (Muñoz et al., 2008)

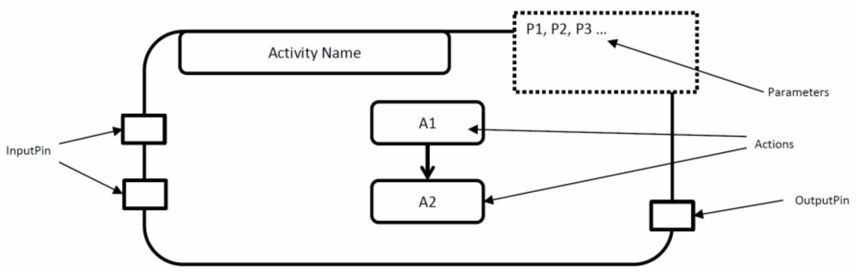


Figure 16 - Template proposed to represent ETL activities - extracted from (Muñoz et al., 2008)

Figure 17 presents the first two ETL operations displayed in Figure 12 using UML Activity Diagrams templates proposed in (Muñoz et al., 2008). The first operation is a filter applied to table *Sales.CustomerAddress* and the second operation is a three table join operation over the common attribute *CustomerID*.

Since this approach uses UML Activity diagrams, it is possible to design the steps needed to correctly maintain a dimension with history thus filling the gap present in the previous approach, however, as we can observe in Figure 17, a simple join operation requires a rather complex diagram. Since normal ETL processes tend to gather information from several OLTP systems, with many tables and attributes, the size and complexity of the ETL conceptual diagram based on this framework would be overwhelming.

2.4 ETL Logical Modeling

Logical modeling is perceived as the second phase of the ETL modeling task. This phase is normally a continuation of the previous one, working on its output to generate an input for the following phase, i.e., the physical design. In the previous section a study of the different proposals for ETL conceptual modeling was made. Some of them used new models, and others used extensions to standard modeling languages. Although there were various proposals to conceptual modeling, the logical modeling phase is somewhat more uniform since there is a common view that the logical design of the ETL should be represented as a workflow of operations (Bouzeghoub et al., 1999). Following the same methodology as in the previous section, let’s study and analyze the ETL logical modeling proposals, again in chronological order.

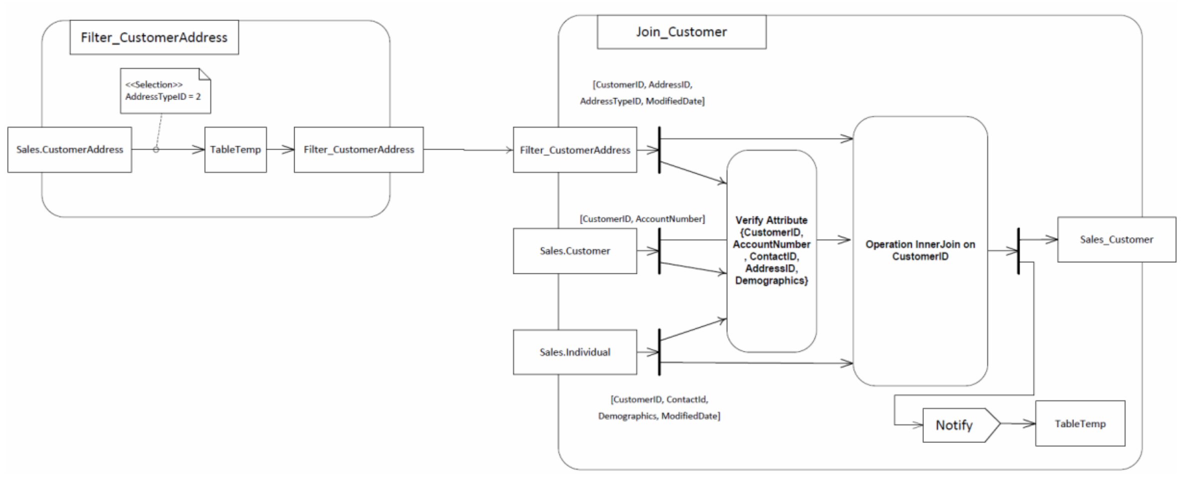

Filter_CustomerAddress
<<Selection>> AddressTypeID = 2
Sales.CustomerAddress
TableTemp
Filter_CustomerAddress
Join_Customer
[CustomerID, AddressID, AddressTypeID, ModifiedDate]
Filter_CustomerAddress
[CustomerID, AccountNumber]
Sales.Customer
Verify Attribute {CustomerID, AccountNumber, ContactID, AddressID, Demographics}
Operation InnerJoin on CustomerID
Sales_Customer
Sales.Individual
[CustomerID, ContactId, Demographics, ModifiedDate]
Notify
TableTemp

Figure 17 - Partial ETL working example modeled using UML AD

2.4.1 Logical Modeling from the Perspective of Vassiliadis and Simitsis

Paired with the proposals already studied over conceptual modeling, Vassiliadis et al. (2002c) also proposed an approach to the logical modeling phase of the ETL process. The original idea is to model each activity using a graphical notation (Figure 18) and build the workflow as a series of activities particular to the ETL task. Activities work on data received from a set of inputs, it can also receive a set of parameters and produces two outputs, the transformed data and rejected data. The task or activity performed over the input data can be one of three major groups, i.e., filters, unary or binary transformations.

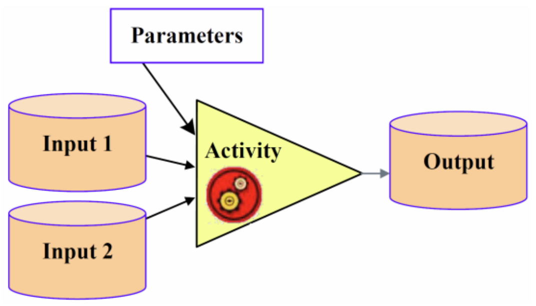


Figure 18- An elementary activity – extracted from (Vassiliadis et al., 2002c)

Using the graphical notation proposed, over the example of Figure 9, the workflow generated is presented in Figure 19. Here, data is extracted from different sources, joined and transformed so it can be loaded into the DW.

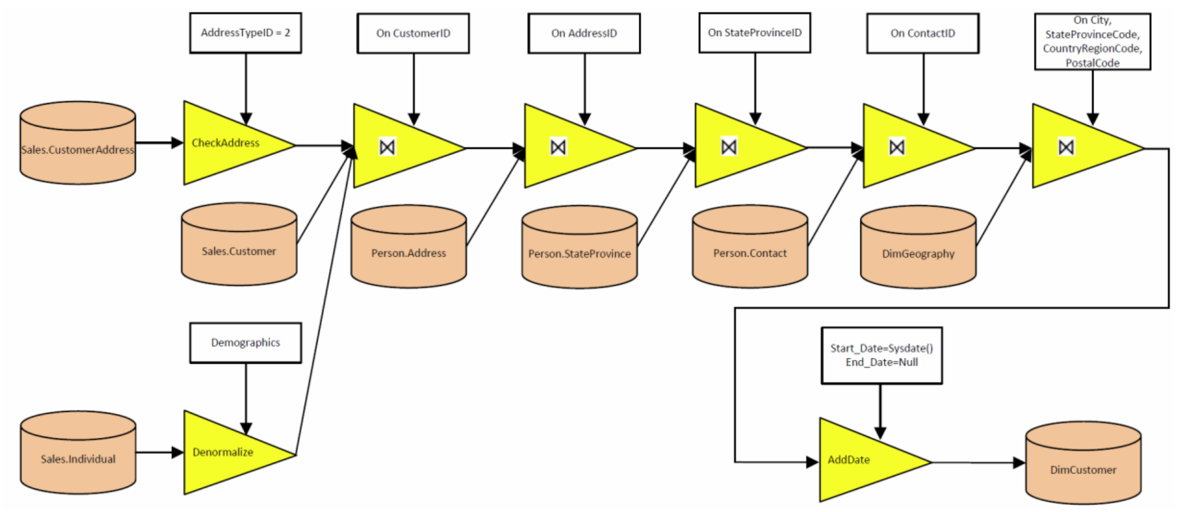
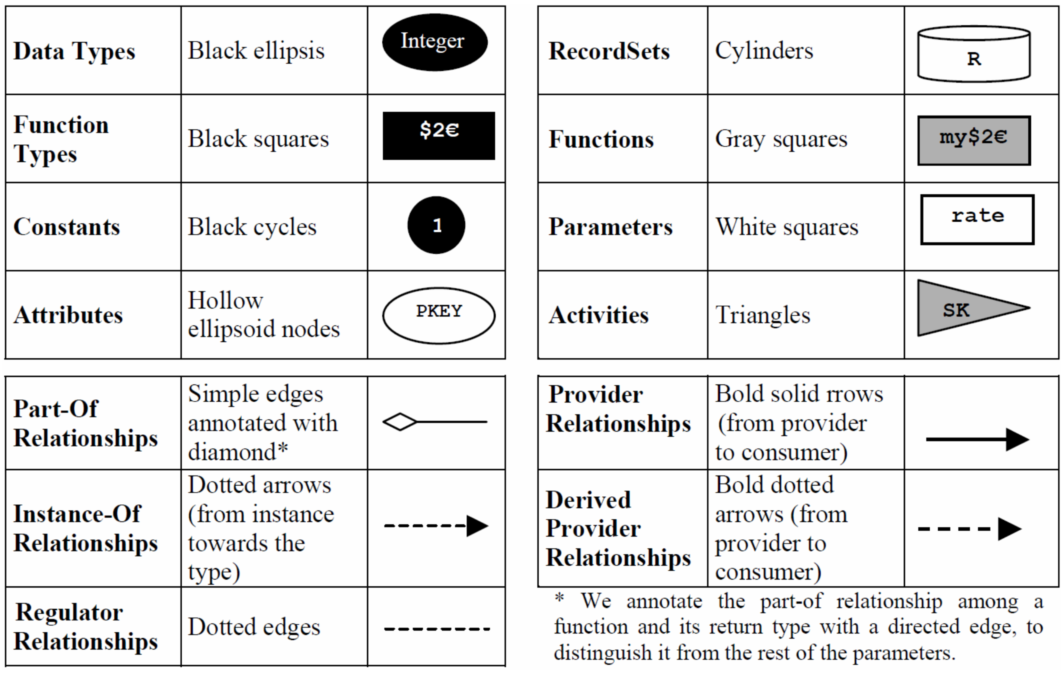


Figure 19 - The logical model of the conceptual model presented in Figure 9 (Vassiliadis et al., 2002b)

The similarity of the high level conceptual diagram presented in Figure 9 and the high level logical diagram presented Figure 19 is enormous. Such diagram represents a conceptual model. The notation used there already suggests a series of steps normally present in a logical design. Authors additionally state that the workflow can be modeled by a graph that they referred as *Architecture Graph*. The graphical notation used to build the graph is presented in Figure 20. The purpose of this extended notation is to clearly define all components and tasks of an ETL logical model, so it can be evaluated and optimized using proposed algorithms exploiting the graph and generating metrics that measure the vulnerability of the nodes through dependencies and responsibilities.

The diagram of Figure 21 presents an extract of the ETL logical model using the *Architecture Graph* notation. Based on the workflow presented in Figure 19, it focus on the join between 3 tables over attribute *CustomerID*, representing mappings between input attributes and output attributes with details over the join operation.



Data Types	Black ellipsis	Integer
Function Types	Black squares	$2€
Constants	Black cycles	1
Attributes	Hollow ellipsoid nodes	PKEY
RecordSets	Cylinders	R
Functions	Gray squares	my$2€
Parameters	White squares	rate
Activities	Triangles	SK
Part-Of Relationships	Simple edges annotated with diamond*	◇────
Instance-Of Relationships	Dotted arrows (from instance towards the type)	------▶
Regulator Relationships	Dotted edges	--------
Provider Relationships	Bold solid rrows (from provider to consumer)	────▶
Derived Provider Relationships	Bold dotted arrows (from provider to consumer)	----▶

* We annotate the part-of relationship among a function and its return type with a directed edge, to distinguish it from the rest of the parameters.

Figure 20 - The graphical notation of the Architecture Graph – extracted from (Vassiliadis et al., 2002b)

The logical modeling of the ETL is perceived as a workflow of activities. The order of the activities is relevant in the minimization of the workload needed to complete the process. As such, Simitsis et al. (2005b) based on the previously proposed ETL logical model graphical notation defined several other transitions that maintain the *Architecture Graph*'s state.

The Swap transition defines that it is equivalent to swap unary activities like selection, projection, null checking and primary key violation, etc. The Factorize/Distribute transition can be used to change the order of activities when using binary and unary activities together, for instance, when at least two unary activities perform the same task in different flows followed by a binary activity, it is equivalent to perform the binary activity first and then execute the unary activity over its result. The opposite is also true, it is possible to distribute an unary activity over the flows that converge into the binary activity. Finally, the Merge transition groups activities in order to force their execution together, the Split transition indicates that a grouped activity can be split.

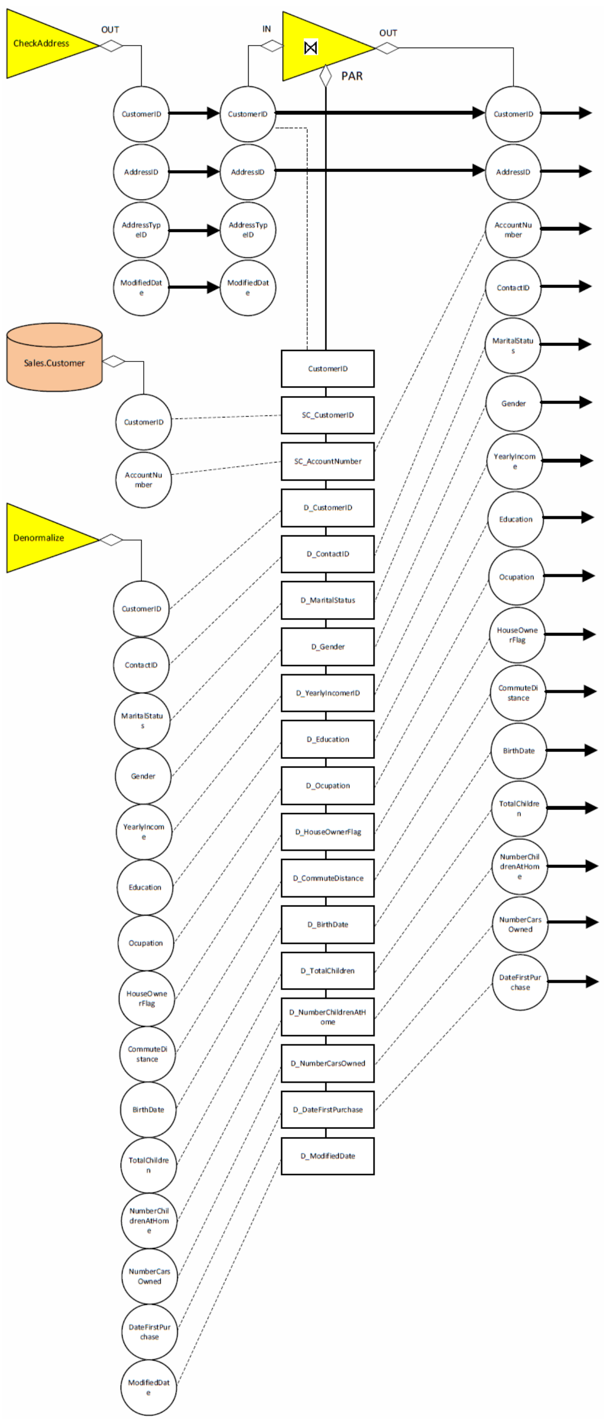


Figure 21 - A fragment of the Architecture Graph model (Vassiliadis et al., 2002b)

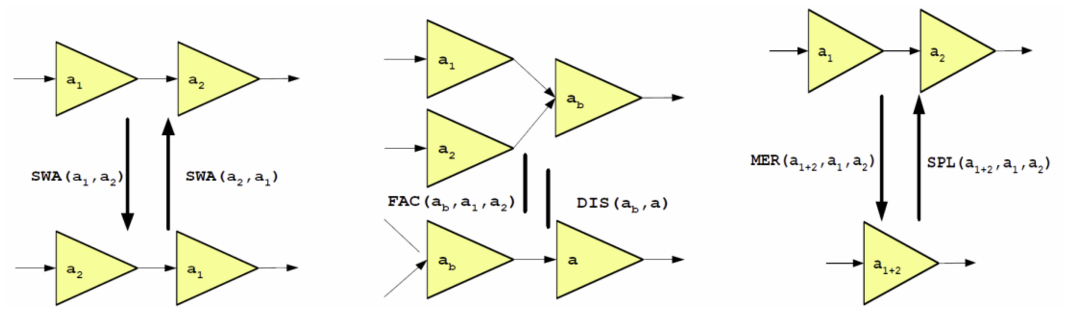


Figure 22 - Transition examples - Swap, Factorize and Distribute, Merge and Split – extracted from (Simitsis et al., 2005b)

These properties can be applied to the working example and several join operations may be swapped of order if the overall workload diminishes. The 3 table join operation might be split in two separate join operations if the result of the first join greatly reduces the result table's size.

The optimization of the logical workflow is based on the premises that the changes can be made without risking the workflow execution. In addition to the optimization process and to better represent the ETL, the authors use LDL++ language, an evolution of LDL (Naqvi and Tsur, 1989), to define templates of the semantics to use in the ETL logical design (Vassiliadis et al., 2003). According to them, since LDL++ is a logic-programming, declarative language that supports recursion, complex objects, negation, external functions, choice, aggregation and updates, it is easier to use, rewrite and compose statements in comparison to SQL.

Through the use of LDL++ a framework of templates for the common ETL activities is defined and those templates are then instantiated when designing the ETL. The templates proposed include selection, not null checking, domain mismatch, projection, surrogate keys, aggregation, etc.

After the presentation of the conceptual model and the logical model, the authors, through the use of LDL++ represent the semantics and present a semi-automatic method for the conversion of the conceptual model into the logical model (Simitsis, 2005, Simitsis and Vassiliadis, 2008, Vassiliadis et al., 2005). This method is composed of a set of rules, algorithms and steps that lead to the correct representation of the logical workflow of the ETL. As a consequence of previous research, the authors then propose a set of algorithms based on state space search that takes into account the workflow generated in LDL++ and tries to minimize the workload of the workflow by swapping, merging or splitting activities (Simitsis et al., 2005c).

As result of several years of research in the ETL design phase, the authors developed an ETL design tool that incorporates the graphical notations proposed – ARKTOS II (Vassiliadis et al., 2005), nevertheless the development of this tool was stopped by lack of funding.

2.4.2 Using BPMN and BPEL for ETL design

More recently, Akkaoui and Zimanyi (2009) have presented an approach to the ETL modeling phase based on *Business Process Modeling Notation*[2] (BPMN) which is a standard notation. BPMN has an intuitive set of conceptual tools to express business processes and is independent of the tools used. It also has the advantage of being able to be translated into execution languages like *Business Process Execution Language*[3] (BPEL), which is an execution language that uses web services to specify actions within business processes.

The authors consider the ETL process as a particular type of business process and therefore it should be modeled as such, using a proposed ETL palette in BPMN. The constructs proposed are divided in four categories: flow objects, artifacts, connecting objects, and swimlanes. Through the use of the proposed ETL palette, the working example that loads a dimension from source tables into the data warehouse would look like Figure 23.

[2] http://www.omg.org/spec/BPMN/2.0/
[3] http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

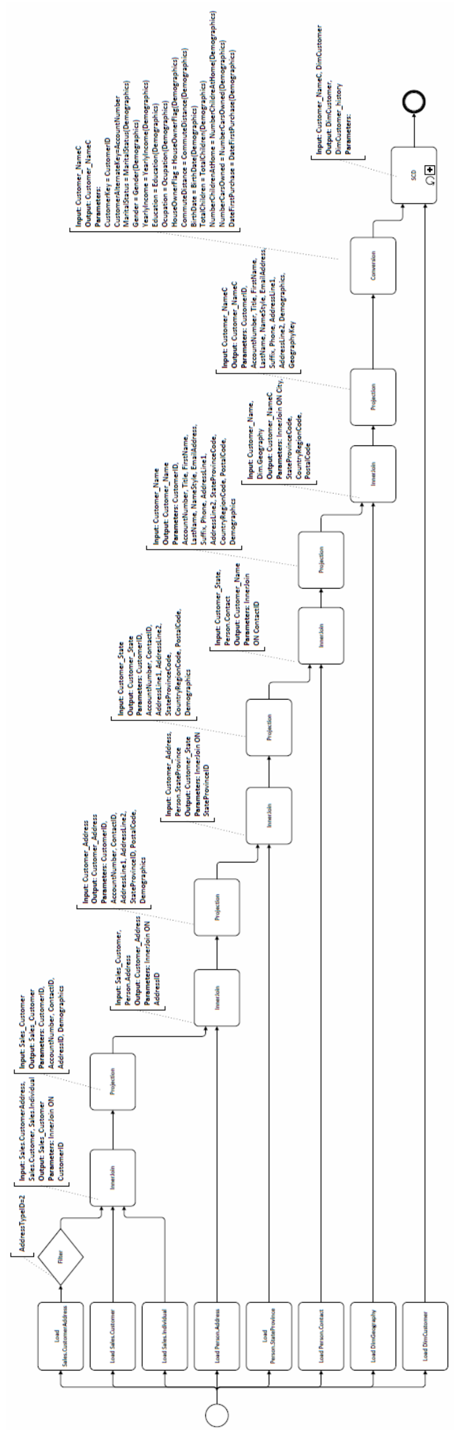


Figure 23 - Example of an ETL workflow using BPMN (Akkaoui and Zimanyi, 2009)

Zooming in in the collapsed task SCD, the steps needed to correctly maintain the dimension with history are presented in Figure 24.

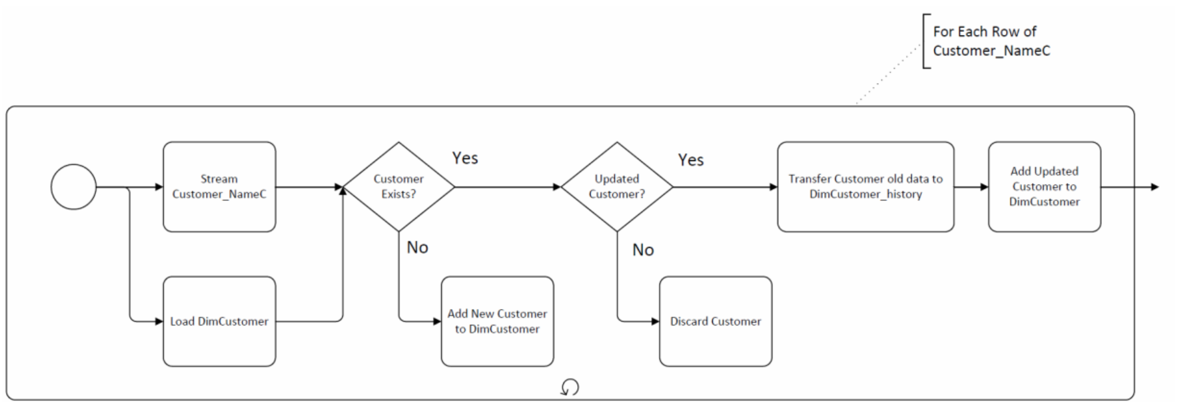

For Each Row of Customer_NameC
Stream Customer_NameC
Load DimCustomer
Customer Exists?
Yes
No
Add New Customer to DimCustomer
Updated Customer?
Yes
No
Discard Customer
Transfer Customer old data to DimCustomer_history
Add Updated Customer to DimCustomer

Figure 24 - Subtasks present on SCD collapsed task

After presenting the BPMN ETL palette, the authors also present a series of steps to translate the BPMN model into BPEL that is represented in XML language.

The use of BPMN in the design of ETL processes has the advantage of making the diagrams easier to understand to all stakeholders but due to the nature of the language designers have too much freedom to define the processes making this approach less formal, i.e., the same kind of operation can be design in different ways depending on the designer.

2.5 ETL Physical Modeling

The last phase of the ETL modeling task is dedicated to the physical model, where some physical options (and optimizations) are taken into account and complemented in the logical model previously designed. Most available ETL commercial tools adopt proprietary graphical notations to represent the workflow of tasks to be performed in the ETL process, through hand-coded routines, SQL statements, or other conventional ways for program execution, implementing the ETL process in order to populate the data warehouse adequately. Nevertheless, this approach is usually platform dependent and might constrain any future data warehouse developments (or changes), since the ETL process is ordinarily designed and executed by a specific tool. As we know, the use of standard notations instead of proprietary notations will reduce this risk and contributes to a global understanding of the standard notation. In addition, a common setback of these tools is the lack of automatic optimizations to the ETL workflow.

In the previous sections, we have seen proposals from (Simitsis et al., 2005b) that permit changes to the workflow in order to optimize it, and the use of LDL++ and state space search algorithms to reduce the workload of the ETL workflow (Simitsis et al., 2005c). In (Tziovara et al., 2007) an additional level of optimization is taken into account, applied mainly in the physical model. The idea is to use sorters between ETL activities in the workflow, taking advantage of sorted data to improve activities performance. The authors proposed an "Exhaustive Ordering Algorithm" that tests the insertion of sorters before a specific activity, and evaluates the impact on the overall workflow.

Recently, there has been done some efforts in the design of the ETL process in conjunction with QoX metrics (Dayal et al., 2009, Simitsis et al., 2009, Simitsis et al., 2010, Dayal et al., 2010). As the ETL process is a computational intensive set of tasks that executes periodically, it has to be designed to deal with failure inside a limited time window. Some of the QoX metrics in analysis are *Reliability*, *Maintainability*, *Freshness*, *Recoverability*, *Scalability*, *Availability*, *Flexibility*, *Robustness*, *Affordability*, *Consistency*, *Traceability* and *Auditability*. In order to maximize these metrics some tradeoffs must be made, since the use of some of them will undoubtedly hurt other metrics, for instance, increasing reliability through the use of redundancy (task replication) will increase the ETL time of completion endangering the time window available.

2.6 Methodologies Comparison

Inmon (1994, 2005) and Kimball (1998, 2004) considered by many the grand-parents of the Data Warehouse approach, presented respectively the data warehouse concept and the Dimensional model. Although they advocate different methodologies to the design and implementation of a DWS, Kimball's methodology is somewhat more defined and structured than Inmon's. Nevertheless both methodologies are quite weak in their formalization, using nonstandard diagrams and notations. In regard to SCD modeling, Kimball advises on a series of tips and best practices normally assigned to each type of SCD, on the other hand, Inmon bases his data warehouse approach on a 3NF relational model therefore a SCD dilemma is not present.

After studying in this chapter the most relevant proposals from data warehousing researchers for the conceptual and logical design of the ETL process and applying them to a SCD scenario, a brief tabular comparison of the referred approaches is presented in Table 1 and Table 2.

Table 1 - Comparison of conceptual design approaches.

Researchers	Approach	Notation	Complexity	Advantages	Disadvantages
Vassiliadis and Simitsis	Novel Approach	Graphical New Symbols	High	High and low level diagrams. Maps source attributes to data warehouse attributes.	New notation – non-standard. Large diagrams when dealing with high number of attributes. No representation to deal with dimensions with history.
Trujillo and Luján-Mora	Extended UML	Graphical UML + Stereotypes	High	Standard known notation.	Very large diagrams when dealing with high number of tables and attributes. No representation to deal with SCD.
Luján-Mora, Vassiliadis and Trujillo	Extended UML	Graphical UML	Very High	Standard known notation. Four level diagrams for better understanding. Maps sources to data warehouse (tables through attributes).	Extremely large diagrams when dealing with high number of tables and attributes.
Muñoz et al.	UML Activity Diagrams	Graphical UML- Activity Diagrams	Extreme	Standard known notation.	Extremely large and complex diagrams when dealing with high number of tables and attributes.

Table 2 - Comparison of logical design approaches.

Researchers	**Approach**	**Notation**	**Complexity**	**Advantages**	**Disadvantages**
Vassiliadis and Simitsis	Novel Approach	Graphical New Symbols	High	High and low level workflow diagrams. Workflow can be optimized through proposed algorithms.	New notation – non-standard. Extremely large diagrams when dealing with high number of attributes.
El Akkaoui and Zimányi	BPMN	Graphical BPMN	Average	Standard known notation. Easy to understand by stakeholders.	Very large diagrams when dealing with high number of tables and attributes. Loose standard contributes to lack of formality in language..

Chapter 3

Relational Algebra Approach

In 1970, Edgar F. Codd presented a research paper with the fundamentals of the relational model as an attempt to separate the logical from the physical design of a database (Codd, 1970). The basic principle that supports the relational model is the definition of *relation*, which is considered to be a finite set of n-tuples, where n is the degree of the relation, or number of attributes.

Definition - Consider a set of domains $D_1, D_2, \ldots, D_n$ not necessarily distinct and where $n>0$. R is a relation if it is a subset of the cartesian product of these set of domains. All members of the relation R are *n-tuples* where n is considered to be the degree of the relation.

Later, Codd redefined a few properties associated with the concept of relation (Codd, 1979), which could be represented in a tabular format, such as:

- There is no duplication of tuples;
- Row order is insignificant;
- Attribute order is insignificant;
- All table entries are atomic values.

Table 3 - Example of a relation represented in a tabular format

Number	Name	City
123	John	London
125	Ana	Lisbon
128	Peter	Miami

Along with the theory that supported the relation model, Codd presented also two data manipulation languages, called *Relational Algebra* (RA) and *Relational Calculus* (RC), that were able to access and query data stored in relations in order to answer to common queries.

In subsequent years, after the presentation of the relational model, several database management systems were developed based on this approach, like *System R* (Astrahan et al., 1976) and *INGRES* (Held et al., 1975, Stonebraker et al., 1976). Nevertheless, the users' need to access summarized data led to the development of aggregate functions that in turn needed the ability to store and represent duplicate tuples.

The theoretical foundations to support these aggregate functions were only formalized later in the work of (Klug, 1982) and developed by (Ozsoyoglu et al., 1987), and the use of duplicates was formalized through the use of Bag Theory (or multisets) instead of Set Theory in (Dayal et al., 1982). All authors support their proposals in Codd's Relational Model and extended the common operations of RA with new operations.

The difference between RA and RC lies in the fact that RA is a procedural language, where the order of each operation is crucial and RC is a non-procedural language, where the order of the operations is not important. Another way to understand the difference is that RA is preoccupied on how to get the data needed to answer the query instead of RC that works on what data to get.

Being a procedural language, the use of RA is normally associated with step by step operations and suits very well as a platform to model the logical phase of an ETL process. Using the example already presented previously (Figure 10), this chapter analyzes the primitive and extended RA operators and operations presenting their syntax and demonstrating their use.

The set of primitive RA operators include the selection and projection operations that were introduced in (Codd, 1970), the cartesian product, union and difference that are mathematical operations already defined in Set Theory and the rename operation that was presented in (Todd, 1976).

3.1 Selection (or Restriction)

The selection operator was first introduced by Codd as Restriction, however it is commonly known as Selection and acts as a horizontal filter over a relation, therefore this is a unary operator since it works over one relation. The relation that results from this operation has the same structure of the original relation, however the contents must satisfy the condition defined. This operator has the same behavior either if working with Sets or Bags (multisets), although algebraic expressions using the relation's attributes might be used in the definition of the condition in Extended Algebra. Nevertheless the result relation has the same structure of the original relation.

The operator is graphically represented by the Greek small letter sigma (σ) and the syntax for the operation is:

$$\sigma_F(R)$$

where F is a logical propositional expression made up of elementary algebraic conditions, and R is the relation over which the operation is applied.

The application of such operation in an ETL scenario is commonly associated with the need to filter data to load into a data warehouse. Figure 25 presents an excerpt from the low-level conceptual model of the example used in chapter 2, where selections and joins are made to gather the necessary data to load into the dimension Customer.

The conceptual model presented gathers only the customers whose addresses Type ID are 2. Using RA to specify this requirement, the expression would be:

RA expression: $\sigma_{(\,AddressTypeID=2\,)}\ (Sales.CustomerAddress)$

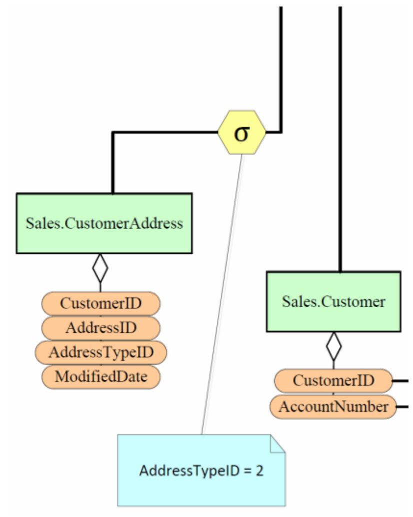


Figure 25 - An excerpt of the low-level conceptual model presented in Figure 10 focusing on the selection operation

Next, other common RA expressions are presented, focusing in their syntax, modus operandi and application in the selected ETL scenario.

3.2 Rename

The rename operation is used to change attributes names and also relations' name. It is normally used to facilitate the understanding of the contents of the attribute or relation, by changing its name to a more comprehensive one. Renaming an attribute is also common when the attribute is used in future RA operations that require identical attribute names to work with.

In an ETL scenario renaming attributes is very common since source systems can vary and their attribute naming conventions might be different from the data warehouse attribute naming.

The rename operator is also a basic RA operator, is graphically represented by the Greek small letter rho (ρ) and the syntax for the operation is:

$$\rho_{S(a_1,\dots,a_n)}(R)$$

where $a_1, \dots, a_n$ is the list of the names of the new attributes contained in the new relation S that should appear in the result relation.

Using the ETL scenario from Figure 26, a rename operation is used to transform one attribute of *Sales.Customer* relation into a matching name of an attribute of *DimCustomer*, i.e. *AccountNumber* needs to be renamed to *CustomerAlternateKey*.

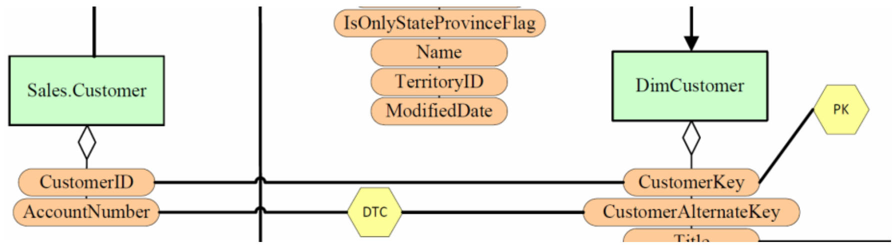


Figure 26 - An excerpt of the low-level conceptual model presented in Figure 10 focusing on data mappings between source relations and dimension relation

RA expression: $\rho_{SalesCust(CustomerID,CustomerAlternateKey)}\ (Sales.Customer)$

An alternative to the renaming operation is achieved by the use of the operation of Projection. This operation is used as a vertical filter over a relation, where only certain attributes appear in the result relation (with or without renaming) and therefore it is a unary operator as the previous operators.

3.3 Projection

The relation resulting from a projection operation contains only the attributes desired, but the contents depend on whether we use sets or bags as the basis for this operation. In Set Theory there are no duplicates therefore the result of the projection operation must not contain any duplicate tuples. In an extended version of RA, which is supported by Bag Theory, duplicates are allowed therefore not removed, and in addition new attributes may be added to the result relation as the result of any function that uses attributes from the original relation.

The operator is graphically represented by the Greek small letter pi (π) and the syntax for the operation is:

$$\pi_{a_1,\dots,a_n}\ (R)$$

where $a_1, \dots, a_n$ is the list of attributes contained in the relation R that should appear in the result relation (Set Theory).

The application of such operation in an ETL scenario is commonly associated with the need to filter unwanted attributes. In Figure 25 after the select operation only some attributes are needed, therefore the unneeded attributes are projected out with the following RA expression:

RA expression: $\pi_{(\,CustomerID,\ \ AddressID\,)}\ (\sigma_{(\,AddressTypeID=2\,)}\ (Sales.CustomerAddress))$

Notice that as the result of a RA expression is also a relation, it can be the operand of another RA expression, this property is called closure. Therefore RA operations can be nested producing expressions that are hard to analyze. In order to simplify these analyses, a RA expression can be fragmented into simpler operations that use temporary relations to store results that are later used in subsequent operation. This is possible due to the use of the assignment symbol ← .

Example :

$$CustType2\ \leftarrow \pi_{(\,CustomerID,\ \ AddressID\,)}\ (\sigma_{(\,AddressTypeID=2\,)}\ (Sales.CustomerAddress))$$

or

$$T_1\ \leftarrow \sigma_{(\,AddressTypeID=2\,)}\ (Sales.CustomerAddress)$$

$$CustType2\ \ \leftarrow \pi_{(\,CustomerID,\ \ AddressID\,)}(\,T_1\,)$$

As already stated, the projection operation can also be used to add attributes and to apply users' functions. Through the use of another excerpt from the conceptual model presented in the previous chapter (Figure 27), now focusing in a few data transformations and normalizations, the projection operation would be used to decompose the XML attribute *Demographics* into a set of attributes matching the Dimension's attributes.

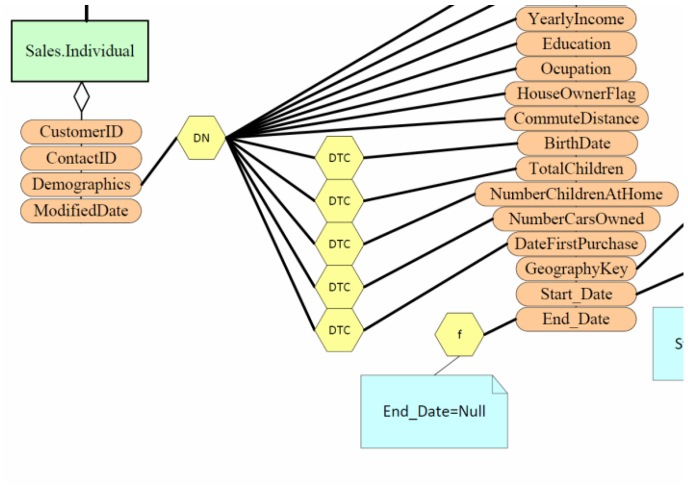


Figure 27 - An excerpt of the low-level conceptual model presented in Figure 10 focusing on the data normalization process

$$S\_Indiv \leftarrow \pi_{\substack{(CustomerID, ContactID, getXMLelem(Demographics,1) \rightarrow MaritalStatus \\ getXMLelem(Demographics,2) \rightarrow Gender, \\ getXMLelem(Demographics,3) \rightarrow YearlyIncome, \\ getXMLelem(Demographics,4) \rightarrow Education, \\ getXMLelem(Demographics,5) \rightarrow Ocupation, \\ getXMLelem(Demographics,6) \rightarrow HouseOwnerFlag, \\ getXMLelem(Demographics,7) \rightarrow CommuteDistance, \\ getXMLelem(Demographics,8) \rightarrow BirthDate, \\ getXMLelem(Demographics,9) \rightarrow TotalChildren, \\ getXMLelem(Demographics,10) \rightarrow NumberChildrenAtHome, \\ getXMLelem(Demographics,11) \rightarrow NumberCarsOwned, \\ getXMLelem(Demographics,12) \rightarrow DateFirstPurchase)}} (Sales.Individual)$$

After selecting the necessary data and filtering the attributes needed for future operations, data gathered from different relations is joined and matched for subsequent transformations. The join operation is based on the set based cartesian product, which combines two sets into a single set with all elements combined.

3.4 Cartesian Product

In the relational model, the cartesian product combines all the tuples of the first relation with all the tuples of the second relation; therefore it is a binary operation since it needs two operands (relations). The operator used is the same as in Math and is graphically represented by (X) and the syntax for the operation is:

$$R \times S = \{ (r,s) : r \in R \wedge s \in S \}$$

Although the cartesian product was defined to work with Sets, this same operation can be applied to bags (multisets). If the operands are bags, the result will also be a bag, therefore duplicates may exist and order is important. Another characteristic of the cartesian product is that it is a basic RA expression and rarely used alone due to its modus operandi. Nevertheless if a selection is used on the result of a cartesian product a Join is achieved. There are several kind of joins, with different purposes, some were presented with the relational model and others only appeared when RA was extended.

3.4.1 Natural Join

The natural join combines all the tuples of both relations over the set of common attributes. The result relation is composed of all attributes from both relations. The graphical symbol used for this operator is (⋈) and the syntax for the operation is:

$$R \bowtie S$$

The use of this operation is very common in an ETL scenario, since data gathered from source systems is normally dispersed over several relations and therefore must be joined for future integration. Using Figure 25 as example, three relations are joined over the common attributes.

$$\pi_{\substack{(CustomerID, ContactID, getXMLelem(Demographics,1) \rightarrow MaritalStatus \\ getXMLelem(Demographics,2) \rightarrow Gender, \\ getXMLelem(Demographics,3) \rightarrow YearlyIncome, \\ getXMLelem(Demographics,4) \rightarrow Education, \\ getXMLelem(Demographics,5) \rightarrow Ocupation, \\ getXMLelem(Demographics,6) \rightarrow HouseOwnerFlag, \\ getXMLelem(Demographics,7) \rightarrow CommuteDistance, \\ getXMLelem(Demographics,8) \rightarrow BirthDate, \\ getXMLelem(Demographics,9) \rightarrow TotalChildren, \\ getXMLelem(Demographics,10) \rightarrow NumberChildrenAtHome, \\ getXMLelem(Demographics,11) \rightarrow NumberCarsOwned, \\ getXMLelem(Demographics,12) \rightarrow DateFirstPurchase)}} (Sales.Individual)$$

$$\bowtie \left(\pi_{(\ CustomerID,\ \ AddressID\ )} \left(\sigma_{(\ AddressTypeID=2\ )} (Sales.CustomerAddress)\right)\right)$$

$$\bowtie \rho_{SalesCust(CustomerID, CustomerAlternateKey)} (Sales.Customer)$$

or, alternatively:

$$\mathrm{T}_1 \leftarrow \sigma_{(\ \mathrm{AddressTypeID=2}\ )} (\mathrm{Sales.CustomerAddress})$$

$$T_2 \leftarrow \pi_{(\ CustomerID,\ \ AddressID\ )}(\ T_1\ )$$

$$T_3 \leftarrow \pi_{\substack{(CustomerID, ContactID, getXMLelem(Demographics,1) \rightarrow MaritalStatus \\ getXMLelem(Demographics,2) \rightarrow Gender, \\ getXMLelem(Demographics,3) \rightarrow YearlyIncome, \\ getXMLelem(Demographics,4) \rightarrow Education, \\ getXMLelem(Demographics,5) \rightarrow Ocupation, \\ getXMLelem(Demographics,6) \rightarrow HouseOwnerFlag, \\ getXMLelem(Demographics,7) \rightarrow CommuteDistance, \\ getXMLelem(Demographics,8) \rightarrow BirthDate, \\ getXMLelem(Demographics,9) \rightarrow TotalChildren, \\ getXMLelem(Demographics,10) \rightarrow NumberChildrenAtHome, \\ getXMLelem(Demographics,11) \rightarrow NumberCarsOwned, \\ getXMLelem(Demographics,12) \rightarrow DateFirstPurchase)}} (Sales.Individual)$$

$$T_4 \leftarrow T_3 \bowtie T_2$$

$$T_5 \leftarrow \rho_{SalesCust(CustomerID, CustomerAlternateKey)} (Sales.Customer)$$

$$CustResult \leftarrow T_4 \bowtie T_5$$

In this particular case there are common attributes, but if the joining attributes had different names, a different type of join could have been used, the equijoin which is a particular case of the theta join.

3.4.2 Theta Join

It is not always possible to use the natural join operation, since it is necessary to have common attributes between the relations being joined or, if there is no interest in joining the relations over the common attributes. In these cases it is possible to associate a set of conditions to the join operation. The theta join uses the same graphical symbol as the natural join and the syntax is very similar, i.e., there is only the addition of a predicate to the join.

$$R \bowtie_F S$$

3.4.3 EquiJoin

The equijoin operation is a particular case of the Theta Join since the predicate associated with the operation is an equality. As in the Theta Join, the structure of the result relation is composed of all the attributes from both relations.

Using the previous example all natural join operations could have been written using the equijoin syntax.

$$T_4 \leftarrow T_3 \bowtie_{T_3.CustomerID=T_2.CustomerID} T_2$$

and

$$CustResult \leftarrow T_4 \bowtie_{T_4.CustomerID=T_5.CustomerID} T_5$$

The use of natural joins, when possible, makes the expression smaller but difficult to comprehend over which attributes the join operation is being done. If using equijoins, the joining attributes must be specified therefore it is easier to understand.

3.5 Attribute Extension

The ability to add new attributes to the result of a RA expression was introduced in the extension of RA (Baralis and Widom, 1994). The new attribute could be populated with values computed from other tuple's attributes or as a result of a user's function. This operation is similar to the extended projection where it was also possible to add new attributes to the result relation. The operator that adds attributes is represented by the Greek small letter epsilon (ε) and the syntax for the operation is:

$$\varepsilon_{(\mathrm{X} = \mathrm{expr})}(\mathrm{R})$$

where *X* is the new attribute's name, *expr* is an expression evaluated over each tuple of *R*.

Using the same running example as in previous demonstrations, Figure 28 shows the addition of two new attributes to dimension Customer. The attribute *Start_Date* is populated with the result of a function that determines the systems' date and the attribute *End_Date* is populated with special state called *Null*. The absence of information is very common in databases, either by being inapplicable in some property or unknown at present time (Vassiliou, 1979, Codd, 1979). Whenever this occurs the attributes in question store a special value, the *Null* value, commonly represented in RA as the Greek small letter omega (ω)

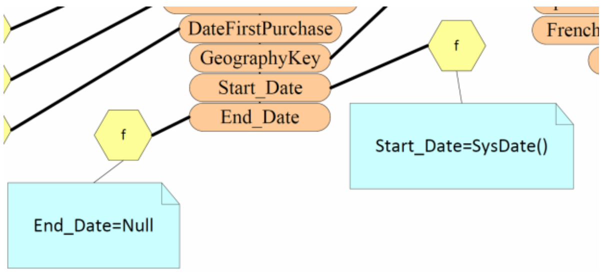


Figure 28 - An excerpt of the low-level conceptual model presented in Figure 10 focusing on the addition of new attribute

The RA expression for this example would have been

$$CustResult \leftarrow \varepsilon_{(Start\_Date=SysDate(\,),End\_Date\,=\,\omega)}(T_6)$$

3.6 Relational Algebra Trees

Another form of representing a RA expression is through the use of RA trees. A RA tree has several properties:

- Each leaf corresponds to the base relations used in the expression
- Non-leaf nodes are the result of a RA operation
- The root of the RA tree is the result of the entire RA expression
- The tree is traversed from leaves to root

The graphical representation of a RA expression that results from the use of a RA tree provides a better way to interpret and comprehend the complexity of the expression (

Figure 29). There are also a few techniques that can be used in order to optimize operations by changing and reordering operations and producing an equivalent optimized RA tree.

The running example used so far in this chapter to introduce and explain the RA operators is the result of the application of the conceptual model proposed by (Vassiliadis et al., 2002a) on a specific ETL scenario, a SCD with history preservation. Nevertheless, as already discussed in the previous chapter, this proposal does not support very well this specific scenario, therefore, and since some of the steps needed to maintain this dimension up-to-date are better modelled in a logical perspective rather than a conceptual one, the logical modeling proposal by (Akkaoui and Zimanyi, 2009) used in the previous chapter will be used to present the remaining RA operators. Figure 24, presented in the previous chapter, models, in BPMN, the steps needed to maintain the dimension up-to-date with history preservation by isolating new from already existing/updated customers. Figure 30 is an excerpt of Figure 24 that focuses on determining if a Customer is new or already existing in the dimension Customer. This goal can be achieved by the use of already presented RA operators in conjunction with the difference operator.

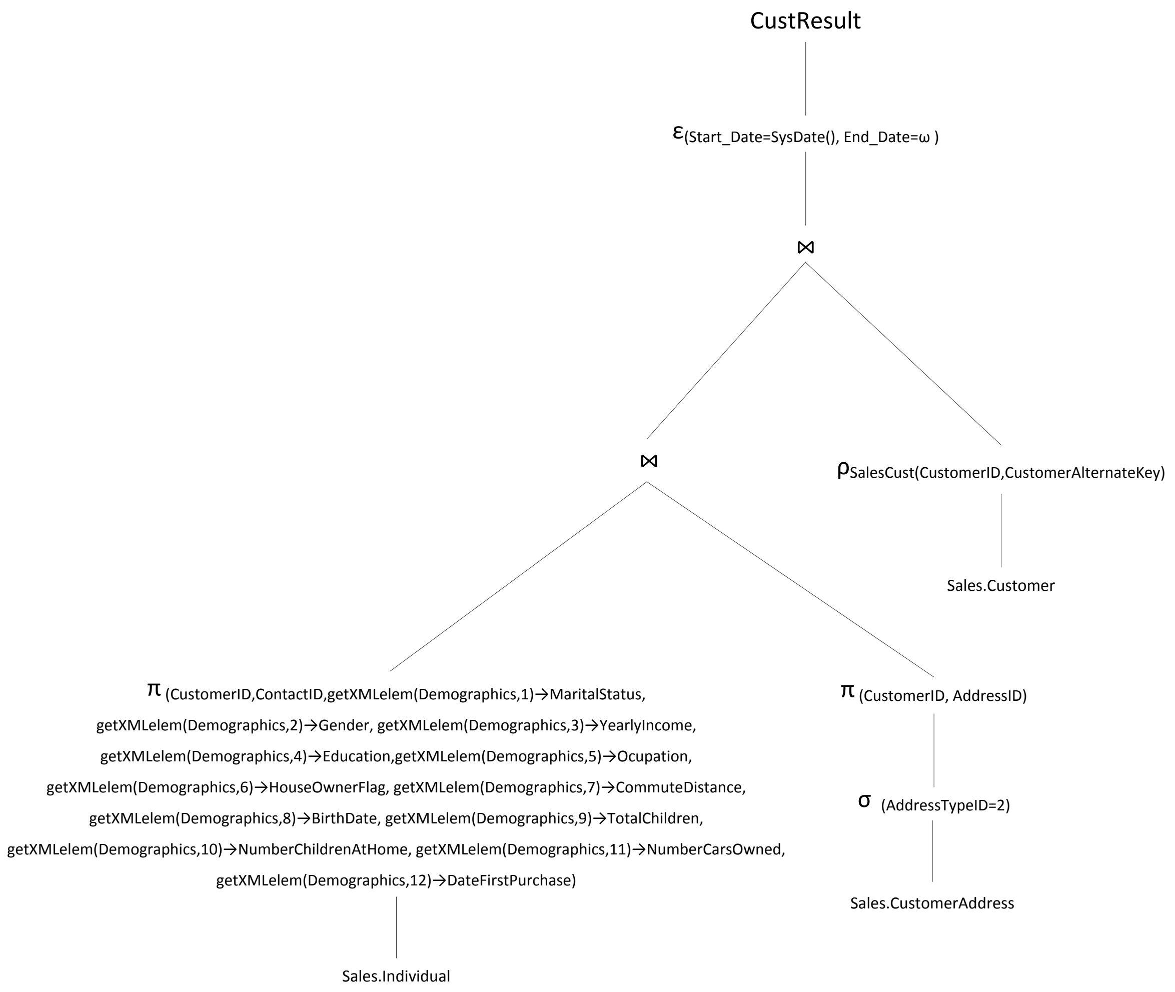

CustResult
ε(Start_Date=SysDate(), End_Date=ω)
⋈
⋈
ρSalesCust(CustomerID,CustomerAlternateKey)
Sales.Customer
π (CustomerID,ContactID,getXMLelem(Demographics,1)→MaritalStatus, getXMLelem(Demographics,2)→Gender, getXMLelem(Demographics,3)→YearlyIncome, getXMLelem(Demographics,4)→Education,getXMLelem(Demographics,5)→Ocupation, getXMLelem(Demographics,6)→HouseOwnerFlag, getXMLelem(Demographics,7)→CommuteDistance, getXMLelem(Demographics,8)→BirthDate, getXMLelem(Demographics,9)→TotalChildren, getXMLelem(Demographics,10)→NumberChildrenAtHome, getXMLelem(Demographics,11)→NumberCarsOwned, getXMLelem(Demographics,12)→DateFirstPurchase)
Sales.Individual
π (CustomerID, AddressID)
σ (AddressTypeID=2)
Sales.CustomerAddress

Figure 29 – RA tree with the representation of the RA expression presented in Section 3.4.1

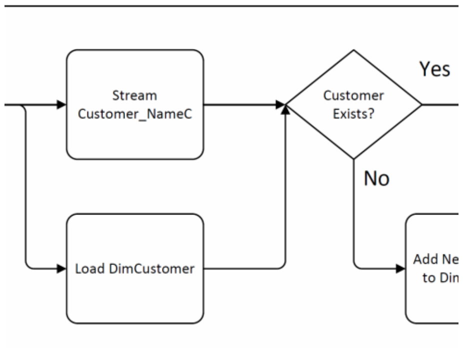


Figure 30 – An excerpt of the BPMN logical modeling presented in Figure 24 focusing in determining new/existing customers

3.7 Difference and Intersect

The difference is set operation that creates a set with the elements that are present in the first set that are not present in the second set and therefore it is a binary operation since it needs two operands (relations). The operator used is the same as in Set Theory and is graphically represented by (-) and the syntax for the operation is:

$$R - S = \{x: x \in \mathrm{R} \wedge x \notin \mathrm{S}\}$$

Assuming that only new and update customers are in the relation *Customer_NameC* the RA expression to determine the new customers would be

$$NewCustomers \leftarrow \pi_{CustomerID}(Customer\_NameC)\ - \pi_{CustomerID}(DimCustomer)$$

and in turn existing customers could be achieved by the intersect operator such as

$$ExistingCustomers \leftarrow \pi_{CustomerID}(Customer\_NameC)\ \cap \pi_{CustomerID}(DimCustomer)$$

Although the intersect operation is not a basic algebra operation, it can be implemented through the use of two difference operations. Nevertheless, the common intersect operation uses the common graphical set representation (∩) and the syntax for the operation is:

$$R \cap S = \{x: x \in \mathrm{R} \wedge x \in \mathrm{S}\}$$

This has the same result as:

$$R - (R - S)$$

These two operations have quite different outcomes when using bags instead of sets. Although the basis are the same, subtract from one relation the contents of the other or find the common tuples in both relations, the ability for these relations to hold duplicates change the outcome of the operation drastically when in comparison with set difference or set intersect. Assuming that there are duplicate tuples in both operand relations, the result relation has to take into account the occurrences of the tuples in each relation. The minimum cardinality of each duplicate tuple remains in the result relation.

Once the existing/updated customers are identified, the process to expire the data implies several steps (Figure 31): set the *End_Date* of the expired tuples and move them to the dimension that holds historical data. One approach to set the *End_Date* is through the use of the attribute extension operation.

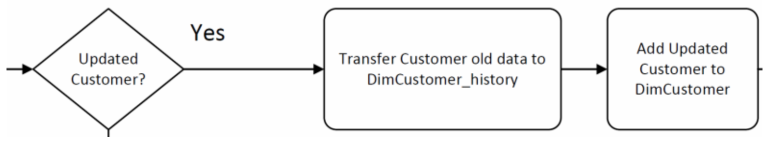

Updated Customer?
Yes
Transfer Customer old data to DimCustomer_history
Add Updated Customer to DimCustomer

Figure 31 - An excerpt of the BPMN logical modeling presented in Figure 24 focusing in the tasks to maintain dimension *DimCustomer* up-to-date

$$ExpiredData \leftarrow \varepsilon_{(End\_Date=SysDate()-1)}(\pi_{(*)}\ (ExistingCustomers\ \ \bowtie DimCustomer))$$

where the * in the projection means all attributes with the exception of *End_Date*. For easier understanding the expression could also be represented in a RA tree (Figure 32).

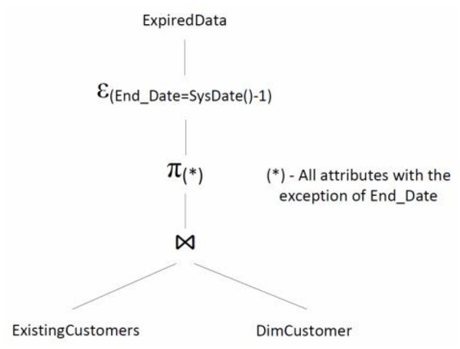


Figure 32 – RA tree determining expired data of the running example

After determining the expired data the next step would be to add these tuples into the dimension historical table through the use of the Union operator and the assignment symbol.

3.8 Union and Insert operation

The union is a set operation that creates a single set with all the elements from two compatible sets. In the relational model based on sets, the union operation combines all the tuples of the first relation with all the tuples of the second relation without duplicates; therefore it is a binary operation since it needs two operands (relations). The operator used is the same as in Set Theory and is graphically represented by (∪) and the syntax for the operation is:

$$R \cup S = \{x : x \in \mathrm{R} \vee x \in \mathrm{S}\}$$

Thus inserting the expired data in the dimension historical table would be:

$$DimCustomer\_history \leftarrow DimCustomer\_history \ \cup \ ExpiredData$$

The cardinality of the result relation of a union operation is at most the sum of the cardinality of both operands due to duplicate elimination. In extended RA the cardinality of the result relation is exactly the sum of the cardinality of both operands since there is no duplicate elimination.

The use of the assignment symbol has two different meanings. It can be used, as already presented, to decompose complex RA expressions using the left side of the assignment as a temporary relation or view. Or, if the left side is an already existing base relation, it can be used for insert, update and delete operations.

After inserting expired data in the historical table (*DimCustomer_history*), those tuples must be removed from *DimCustomer*.

3.9 Delete operation

Similar to the insert operation, the removal of tuples from a relation is obtained through the use of the assignment and difference operators.

$$R \leftarrow R - S$$

where *R* and *S* are union compatible relations.

Applying the syntax of this operation to the step needed in the running example, the expression would be as follows

$$DimCustomer \leftarrow DimCustomer - (ExistingCustomers \bowtie DimCustomer)$$

or in a RA tree

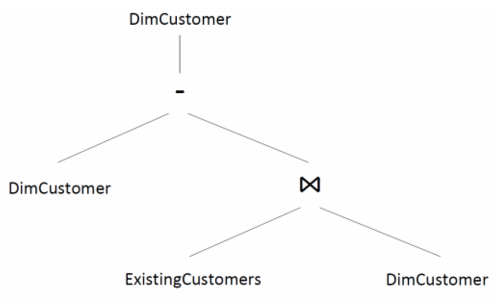


Figure 33 - RA Tree removing expired data from *DimCustomer*

Finally, and to end the case presented in Figure 33, updated customers are added to *DimCustomer*. In this case, and since expired data was already removed from dimension

DimCustomer , and that only new or existing customers were retrieved from the source systems, all data present in relation *Customer_NameC* can be added to *DimCustomer*. Let us assume that it would be better, for performance issues, that such tuples are ordered by the primary key before being added to the dimension. Such task can be achieved in RA through the use of specific operator that sorts tuples according to determined attributes.

3.10 Sorting

Although one of the properties of the relational model states that the order of the tuples is irrelevant, that property is not entirely followed by DBMSs for performance reasons, but is also inadequate for some query analysis. Once data is stored, common data analysis require that data is presented in an orderly fashion. The operator that allows for tuple sorting is graphically represented by the Greek small letter tau (τ) and the syntax for the operation is:

$$\tau_{\mathrm{L}}(R)$$

in which L is the list of attributes over which the relation should be ordered by. Once this operator is applied to a relation, the result is no longer a relation but a list.

Sorting and adding Customers to dimension *DimCustomer* in RA would be

$$DimCustomer \leftarrow DimCustomer \cup \left(\varepsilon_{\left(\substack{Start\_Date=SysDate(\,),\\ End\_Date=\,\omega}\right)}(\tau_{(CustomerID)}(Customer\_NameC))\right)$$

or in a RA tree

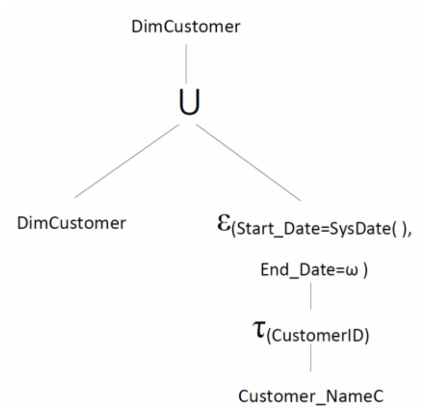


Figure 34 – RA Tree adding new and existing customers to dimension *DimCustomer*

3.11 Additional Operators

Based on the primitive RA operators, presented earlier, a series of auxiliary operators are presented next. These operators simplify the algebra expressions, but they can be rewritten using the primitive ones.

3.11.1 SemiJoin

The semijoin operation is used to identify the tuples of a relation (R) that have correspondent tuples in another relation (S). The result relation has the same structure of the first operand (R), nevertheless it can be expressed using a Natural Join followed by a projection over the first operands' attributes. The graphical symbol used for this operator is (⋉) and the syntax for the operation is:

$$R \ltimes_F S$$

$$R \ltimes_F S \Leftrightarrow \pi_{<Att_R>} (R \bowtie_{\mathrm{F}} S)$$

3.11.2 OuterJoins

Dealing with *Null* valued attributes in joining relations is sometimes necessary, and in those occasions natural joins and equijoins filter tuples that don't have a corresponding tuple due to the absence of related information (*Nulls*). Bearing this in mind, a set of extended RA operations were defined to deal with this special value.

The outerjoin is a variant of a join operation, therefore the result relation structure is the same as if using a theta join operation (all attributes of both relations), where non-correspondent tuples also appear in the result table, with the other relation's attributes Null (ω). The graphical symbol used for this operation varies according to specificity of the operation, since the outerjoin can be to the left, right or full. The syntax for the operation is:

Left Outer Join : $R ⟕_F S$

In this case, the result relation of this operation will be comprised of all the combining tuples of a typical join plus all tuples of *R* that do not have corresponding tuples in *S*. With this operation there is no joining loss of information regarding relation *R* because all tuples of *R* will be present in the result relation. In the result relation, those tuples that have no correspondence in relation *S* will have their S attributes with *Null* value.

Right Outer Join : $R ⟖_F S$

This variant of outer join will have the same behave in a similar way of the left outer join, but this time all tuples of relation *S* will be in the result relation regardless of having or not a corresponding tuple in relation *R*.

Full Outer Join : $R ⟗_F S$

Lastly, the full outer join will result in a relation with all the tuples of relation R and relation S. Tuples that match will be present like a normal join operation, unmatched tuples will also be present with the other relation's attributes with *Null* values.

3.11.3 Division

The division operation is used when there is the need to determine the tuples of a relation (*R*) that correspond (in pre-determined attributes) to all tuples in another relation (*S*). The graphical symbol used for this operator is the common sign for division (÷) and the syntax for the operation is:

$$R \div S$$

All attributes of relation *S* must be contained in relation *R*. The result relation will only have the attributes of relation *R* that are not present in relation *S*.

This operation can also be expressed with the basic RA operators.

Assuming that Att_A are the attributes of relation *R* that are not present in relation *S*

$$T_1 \leftarrow \pi_{<Att_A>}(R)$$

$$T_2 \leftarrow \pi_{<Att_A>}(((\mathrm{S} \times \mathrm{T}_1) - \mathrm{R})$$

$$T \leftarrow T_1 - T_2$$

3.12 Aggregation

The Relational Model revolutionized the way data was stored and accessed. Nevertheless the need to extract information from stored data was not satisfied by the use of simple selections, projections and joins. In order to compare data, to answers to particular questions, new operations were developed and implemented, first in database management systems and later formalized when Extended Relational Model was proposed (Dayal et al., 1982, Klug, 1982, Ozsoyoglu et al., 1987, Grefen and de By, 1994).

One set of operations developed in DBMS and later formalized accordingly to the extended relational model was aggregate functions. The idea is to compute and determine specific values applied to an attribute. There are five functions with special properties and characteristics:

- MAX – determines the maximum value of an attribute
- MIN – determines the minimum value of an attribute
- COUNT – counts how many values an attribute has
- SUM – sums all the values of an attribute
- AVG – calculates the average value of an attribute

The MAX, MIN and COUNT functions can be applied to every type of attribute; on the other hand SUM and AVG can only be applied to numeric attributes. The operator is graphically represented by the Greek small letter gamma (γ) and the syntax for the operation is:

$$\gamma_{\mathrm{L}}(R)$$

where *L* is the list of elements composed of individual attributes (grouping attributes), and/or elements of the form *θ(A)* where *θ* is the aggregate function and *A* the attribute to which the function is applied.

To demonstrate the use of this operator in an ETL scenario let's use an excerpt of Figure 17 which represents a three relation joining operation using UML Activity Diagrams proposed by (Muñoz et al., 2008). This operation was already used previously in this chapter to present the Natural Join and that operation will not be the primary focus of this section. The focus is the notification process and the ability to store aggregated data stored in *TableTemp*.

It is normal in an ETL process to log a series of data important for future analyses like errors, task completion times and records processed. Analyzing the quantity of data processed and time required for task completion will serve as valuable information for the ETL system monitoring phase.

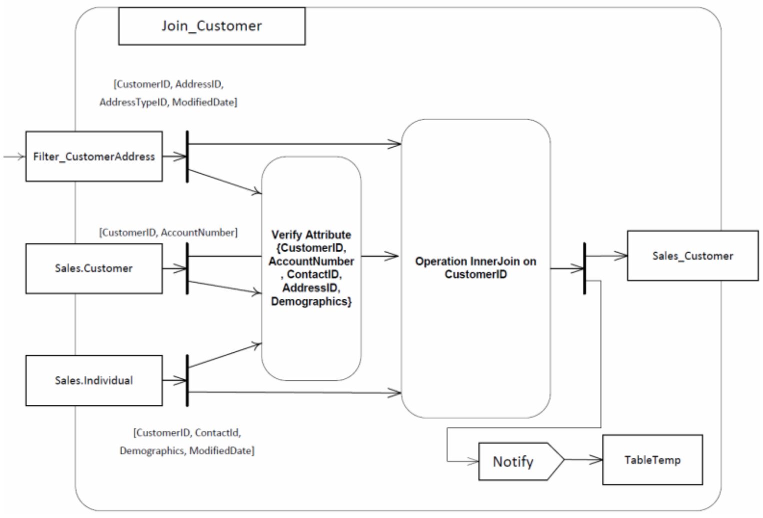

Join_Customer
[CustomerID, AddressID, AddressTypeID, ModifiedDate]
Filter_CustomerAddress
[CustomerID, AccountNumber]
Sales.Customer
Sales.Individual
[CustomerID, ContactId, Demographics, ModifiedDate]
Verify Attribute {CustomerID, AccountNumber, ContactID, AddressID, Demographics}
Operation InnerJoin on CustomerID
Sales_Customer
Notify
TableTemp

Figure 35 - An excerpt of the conceptual model presented in Figure 17, focusing in a three relation joining operation

Consider first that in the most critical ETL operations a notification process exists, similar to the presented, which upon task completion inserts a record in *TableTemp*. Let's assume that *TableTemp* structure is comprised of:

$$TableTemp = < OperationID, ProcessID, Date, Start, End, Duration, Records >$$

where *OperationID* identifies the ETL operation, *ProcessID* identifies a set of operations in the ETL system, *Date* is the day, *Start* is the time of the beginning of the operation, *End* is the time of completion of the operation, *Duration* is the amount of time needed to complete the operation and *Records* is the amount of tuples processed. Assume also that after each ETL run data is aggregated and disseminated over a few statistical tables and then the table is cleared for the next ETL run.

To properly maintain this table the ETL system must update it every time the operation concludes with data gathered from the ETL system configuration and from the operating system. Nevertheless the data needed to calculate the number of records processed is gathered with the use of a RA operation called aggregation. Using the COUNT function over a set of tuples the number of records is determined and used to populate the record to insert in *TableTemp*.

$$\gamma_{\mathrm{COUNT(CustomerID)}}(CustResult)$$

Once the ETL process terminates, several analyses could be triggered and statistical tables could be updated with information stored in *TableTemp*. The ETL manager can analyze data stored in *TableTemp* or in the statistical tables that could also be used as input for process optimization in future ETL runs. For instance, if the average time taken to complete an operation was an important input for optimization, aggregation and possibly sorting would be used in the relation algebra expression needed to identify such operations.

$$\tau_{AVG\_DUR}(\gamma_{\mathrm{(OperationID,AVG(Duration)\rightarrow AVG\_DUR)}}(TableTemp))$$

Let's suppose that two of the statistical tables present have the following structure:

$$Stats = < ProcessID, Date, SumDuration >$$

$$Warning = < ProcessID, Date >$$

and that the stored data is used to monitor each ETL run. *Stats* table stores the amount of time taken for each process in an ETL run. *Warning* table signals all processes that consumed double the average they normally take to complete the task. The RA expressions needed to keep these tables up-to-date would be:

$$Duration \leftarrow \gamma_{(ProcessID,Date,SUM(Duration)\rightarrow SumDuration)}(TableTemp)$$

$$AvgDuration \leftarrow \gamma_{(ProcessID,AVG(SumDuration)\rightarrow AvgDuration)}(Stats)$$

$$T_1 \leftarrow (Duration \bowtie AvgDuration)$$

$$T_2 \leftarrow \sigma_{(SumDuration > AvgDuration*2)}(T_1)$$

$$T_3 \leftarrow \pi_{(ProcessID, Date)}(T_2)$$

$$Warning \leftarrow Warning \cup T_3$$

$$Stats \leftarrow Stats \cup Duration$$

3.13 Duplicate Elimination

Extended RA works over bags and therefore duplicates are allowed, nevertheless there is an operator that transforms a bag into a set by removing duplicate tuples. The operator that removes duplicates is graphically represented by the Greek small letter delta (δ) and the syntax for the operation is:

$$\delta(R)$$

Using one of the previous statistical tables, table *Warning,* which is used to store processes that consume double the average time normally needed, let's suppose that the ETL manager wanted to know in which days those occurrences happened. The expression:

$$\pi_{(Date)}(Warning)$$

will return a list of dates, with possible duplicates that don't add any value to the answer, therefore the use of the duplicate elimination operator

$$\delta\left(\pi_{(\mathrm{Date})}(Warning)\right)$$

3.14 Algebraic Laws

User queries are often long and complex, being frequently difficult to process. When these queries are written or translated to RA expressions there is an optimization process that can simplify or improve the time taken to answering them. Query optimization is often based on the principles of the algebraic laws applied to sets and bags.

3.14.1 Commutativity and Associativity

Operations like the cartesian product, natural join, union and intersect uphold the commutative and associative laws, either using the Set or Bag Theory.

Commutative laws:

- $R \times S = S \times R$
- $R \bowtie S = S \bowtie R$
- $R \cup S = S \cup R$
- $R \cap S = S \cap R$

Associative laws:

- $(R \times S) \times T = R \times (S \times T)$
- $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$
- $(R \cup S) \cup T = R \cup (S \cup T)$
- $(R \cap S) \cap T = R \cap (S \cap T)$

Any other type of join operation besides natural join might not uphold the associative law. See, for instance, the example of a theta join:

- $(R \bowtie_c S) \bowtie_d T = R \bowtie_c (S \bowtie_d T)$

in which *c* and *d* are join conditions. If *d* condition refers to attributes of relation *R* the equality is not valid.

3.14.2 Laws for Selection

The selection operation maintains the structure of the base relation and filters the tuples according to the condition defined. Bearing this in mind there are a few equivalences that can be deducted and used in query optimization:

- $\sigma_{c \wedge d}(R) = \sigma_c \left(\sigma_d (R) \right)$
- $\sigma_{c \vee d}(R) = \sigma_c (R) \cup \sigma_d (R)$ - does not apply for Bags
- $\sigma_c \left(\sigma_d (R) \right) = \sigma_d \left(\sigma_c (R) \right)$
- $\sigma_c (R \cup S) = \sigma_c (R) \cup \sigma_c (S)$
- $\sigma_c (R - S) = \sigma_c (R) - \sigma_c (S) = \sigma_c (R) - S$

Additionally, and only if condition *c* can be applied to relation *R* alone, a few more equivalences can be deducted:

- $\sigma_c (R \times S) = \sigma_c (R) \times S$
- $\sigma_c (R \bowtie S) = \sigma_c (R) \bowtie S$
- $\sigma_c (R \bowtie_d S) = \sigma_c (R) \bowtie_d S$
- $\sigma_c (R \cap S) = \sigma_c (R) \cap S$

3.14.3 Laws for Projection

The projection operation is often used to filter out attributes that are not needed, either to answers the query or for future operations. Therefore it is a good practice to project out the unneeded attributes prior to any join operation. Some equivalences that allow to project prior to the join operation:

- $\pi_L(R \bowtie S) = \pi_L\left(\pi_M (R) \bowtie \pi_N (S) \right)$
- $\pi_L(R \bowtie_c S) = \pi_L\left(\pi_M (R) \bowtie_c \pi_N (S) \right)$
- $\pi_L(R \times S) = \pi_L\left(\pi_M (R) \times \pi_N (S) \right)$

In all these cases $\mathrm{L} \subseteq \mathrm{M}$ and $\mathrm{L} \subseteq \mathrm{N}$ and $\mathrm{M} \cap \mathrm{N} \neq \emptyset$, which means that the reduced attributes of relation R and relation S must contain their joining attributes in addition to the output attributes represented by L.

There are laws that can be applied to sets and cannot be applied to bags and vice versa. The projection operation in conjunction with other operations has that kind of problems. When using bags the following equivalence holds true but it does not for sets:

- $\pi_L(R \cup S) = \pi_L(R) \cup \pi_L(S)$

Additionally it is not possible to distribute (or push down) the projection operation over the intersection or difference operation.

3.14.4 Laws for Duplicate Elimination, Grouping and Aggregations

As extended RA was presented, algebraic laws that defined the behavior of each operation were also demonstrated. The duplicate elimination and grouping operators have, in one particular case, the same result since grouping by all attributes is the same has using duplicate elimination

$$\delta(R) = \gamma_A(R)$$

where *A* is the list of all attributes of relation *R*.

The algebraic laws when using aggregations and duplicate elimination are simple due to the behavior of grouping.

$$\delta\left(\gamma_L(R)\right) = \gamma_L(R)$$

where *L* is a list of grouping attributes and aggregate functions over relation *R*. There is no need to use duplicate elimination since the grouping/aggregation does not produce duplicates.

On the other hand, the removal of duplicates prior to the use of aggregate functions might affect the outcome depending on the aggregate function used. MIN and MAX functions are not affected by duplicate and in this case the following law holds true.

$$\gamma_L(R) = \gamma_L(\delta(R))$$

A small optimization that can also be adopted when using aggregations is to project out unneeded attributes has already stated.

$$\gamma_L(R) = \gamma_L(\pi_M(R))\ where\ L \subseteq M$$

Chapter 4

ETL standard processes specification

The ETL component of a data warehousing system is comprised of several tasks that are responsible for gathering data from information sources, transforming it (through cleaning, conforming and conciliating it), and finally loading it into the data warehouse (Kimball and Caserta, 2004). As already stated previously, the modeling phase of an ETL system is crucial to the success of its development, and therefore subject of intensive research by the data warehousing community. This chapter focuses on presenting a set of ETL patterns based on RA operations that were applied to common ETL activities. The goal of this approach was to achieve a standard and formal specification of ETL activities that still lacks in common ETL design methodologies, as already focused in chapter 2.

A typical ETL scenario integrates one or more data sources, the data providers, a staging area, where data, once extracted from sources, is cleaned, transformed and conformed accordingly to business necessities, and a data warehouse (or a data mart) to receive the data prepared to support decision-makers views (Figure 36).

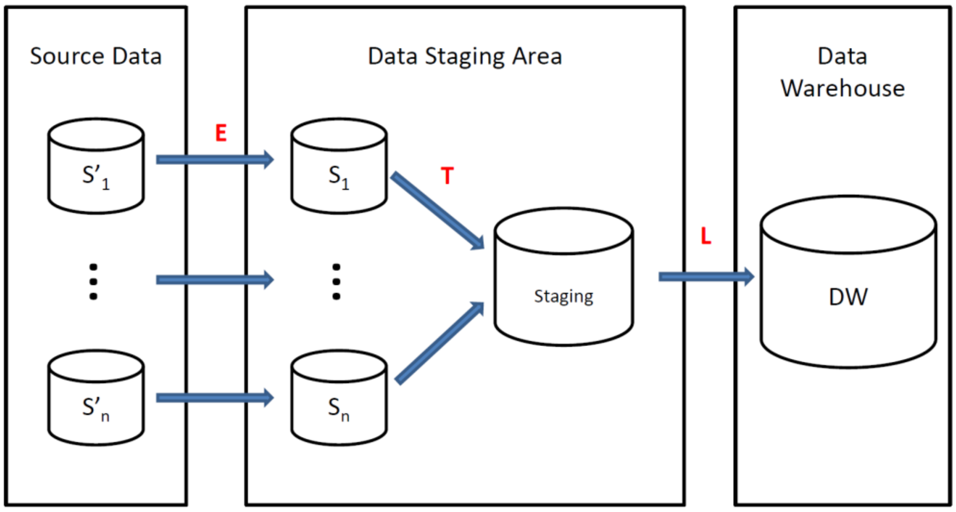


Figure 36 - A generic view of a data warehousing system functional architecture.

An ETL process integrates several tasks, mainly defined as a workflow process (Vassiliadis et al., 2002b, Simitsis et al., 2005a, Tziovara et al., 2007) that are normally executed in a dedicated environment. However, unusual DWS environments, such as grid environments, start to arise as an approach to deal with ETL performance bottlenecks when dealing with large amounts of data in a limited time window (Santos et al., 2012).

4.1 Changing Data Capture

Let us first assume that the data warehouse we will use is supported by a dimensional model (Kimball and Ross, 2002) and therefore data extracted from sources will be cleaned, transformed and integrated into dimensions and fact tables. Capturing changes in information sources is a critical operation that is required for the correct maintenance of a data warehouse. The challenge of this task varies according to the number and variety of the data sources involved with, which can be very diversified, and so is the primary challenge to access data. The data needed to populate a data warehouse might be stored in mainframes, flat files, XML files, DBMS, Web logs, etc., being its access sometimes complicated. Nevertheless, and exclusively for modeling purposes, let's assume that data is extracted from source systems and correctly represented in a relation's format. Let's also assume that the ETL process extracts all the data existent in a source system to load into a specific dimension, since it might not be possible to extract only new and changed data. We shall call the relation that stores the data extracted $src\_dimdata_{S'_x}$. This relation consists of a list of attributes that includes a business key and a finite set of additional attributes (Equation 1). The business key is normally identified as a primary key and may be a single attribute or a set of attributes.

$$src\_dimdata_{S'_x} = \langle BK_{S'_x}, Att_1, \dots, Att_p \rangle \tag{1}$$

where $BK_{S'_x}$is the business key of source S'_x and $1 \leq x \leq n$ and $n \geq 1$, and $Att_1, \dots, Att_p$ are additional data attributes needed for the dimension and $p \geq 1$.

Let's also assume that the previous extraction has been stored to facilitate the detection of new or changed tuples (Equation 2).

$$prev\_src\_dimdata_{S'_x} = \langle BK_{S'_x}, Att_1, \dots, Att_p \rangle \tag{2}$$

Now, to determine the new and changed tuples a subtraction is the only operation needed (Equation 3).

$$src\_dimdata_{S_x(BK_{S_x}, Att_1, \dots, Att_p)} \leftarrow src\_dimdata_{S'_x} - prev\_src\_dimdata_{S'_x} \tag{3}$$

To prepare for the next extraction, the previous extraction is discarded and the current extraction will be the next previous extraction (Equation 4).

$$prev\_src\_dimdata_{S'_x} \leftarrow src\_dimdata_{S'_x} \tag{4}$$

From this point on a series of transformations, mainly due to quality enforcement rules, are applied to the data extracted before integrating it into the DW.

4.2 Data Quality Enforcement

Usually, the information sources of a DWS contain inaccurate data, unknown or null data and sometimes inconsistent data that are spread generally by operational systems' objects, constraints or rules. This imposes the inclusion of cleansing tasks in data warehousing populating systems to detect and filter (and sometimes recover) potential data anomalies in the source's data before loading it into a DW. Often, these tasks are implemented and executed through the use of proprietary tools that analyze and transform the data accordingly to some predefined business requirements. Cleansing tasks are undoubtedly important. They are mainly concerned with identifying problems in metadata and data (Hernández and Stolfo, 1998, Rahm and Do, 2000, Lee et al., 1999), i.e., inconsistencies at attribute and row level, rather than defining a strategy to execute the procedures needed to deal with those problems.

4.2.1 Dirty Data and Data Quality

A data warehouse usually receives information from several sources, frequently of heterogeneous nature. These sources are responsible to support day-to-day business activities, storing and providing data to ensure common tasks in regular operational services. Nevertheless, these business activities not always provide such systems with accurate or even known data. It is common that during the normal use of an information system, data might not be available for input or data might be inaccurately introduced in the system due to some not expectable event like a human error. The balance between allowing attributes to be null or forcing their introduction is also a compromise to be made in the development of any information system. However, it is quite important to keep in memory that business cannot be stalled by overzealous requirements of an information system. When building a DW, several heterogeneous data sources are normally considered for integration and data can be presented in a lot of different formats, i.e., relational databases, structured or semi-structured files, or even free-form files. When analyzing source data, the more permissive the rules are the more difficult it is to validate the data that was stored.

Analyzing the types of data quality problems present in source data (Rahm and Do, 2000), as classified in Table 4, it is notorious that cleaning tasks complexity is far greater when several sources are involved. When dealing with a single source, the majority of the problems with dirty data are a reflex of lack of constraints, integrity constraints, domain constraints or even referential integrity. If the schema is permissive, it is also normal to find at the instance level data misspelled, redundant or attributes that contradict each other, such as the attributes age and date of birth, for instance. However, if multiple heterogeneous sources are used, the complexity of the problems associated with creating an integrated repository escalates to a higher degree of complexity. In this case we are dealing not only with data quality issues but also with data conciliation problems.

Table 4 - Data quality problems (Rahm and Do, 2000)

Single-Source Problems		*Multi-Source Problems*	
Schema Level (Lack of integrity constraints, poor schema design)	*Instance Level* (Data entry errors)	*Schema Level* (Heterogeneous data models and schema designs)	*Instance Level* (Overlapping, contradicting and inconsistent data)
- Uniqueness - Referential integrity	- Misspellings - Redundancy duplicates - Contradictory values	- Naming conflicts - Structural conflicts	- Inconsistent aggregating - Inconsistent timing

Although the problems associated with poor data quality are mostly identified, the tasks or procedures needed to deal with them are much diversified. In (Costa, 2006) a brief summary of actions is presented to solve data anomalies like the ones identified previously in Table 5. These tasks (Figure 37) are normally executed during the transformation phase of an ETL process.

Table 5 - Anomalies resolution strategy (Costa, 2006)

N.	Description
1	Data Decomposition – obtain atomic values
2	Transformations:
2.1	Standards (uppercase, lowercase, acronyms and abbreviations)
2.2	Normalization (i.e. enforce business rules)
2.3	Corrections
3	Correct null values
4	Referential integrity enforcement
5	Data enrichment
6	Duplicate data resolution
7	Expert intervention
8	Enforce Data Domains
9	Enforce Mandatory Data Entry
10	Change Data Type
11	Change Multidimensional Model

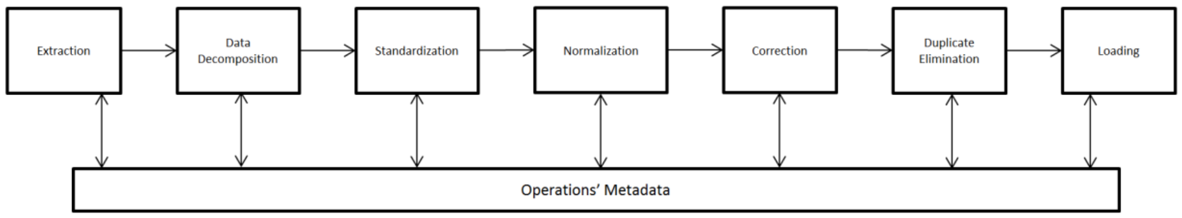


Figure 37 - Data transformation and cleaning tasks (Costa, 2006)

4.2.2 Modeling DQE Tasks with Relational Algebra

For demonstration purposes the dimension structure presented in Equation 3 will be used as the basis for the following transformations. The data stored in $src\_dimdata_{S_x}$ refers to new and changed data extracted from a source system.

Data Decomposition

Let us assume now that the attribute Att_p in Equation 3 is an alphanumeric attribute that was used to store information that according to the data warehouse structure is now stored in several different attributes, for example we will define three new attributes, as defined in Equation 5.

$$src\_dimdata\_d_{S_x} = \langle BK_{S_x}, Att_1, \dots, Att_p, Att_x, Att_y, Att_z \rangle \tag{5}$$

This is a common example of the use of data decomposition task. The contents of attribute Att_p should be decomposed into three new attributes Att_x, Att_y, Att_z. The RA expression that will allow us to obtain that decomposition is presented in Equation 6.

$$src\_dimdata\_d_{S_x} \leftarrow \varepsilon_{[Att_x=subs(Att_p,1),Att_y=subs(Att_p,2),Att_z=subs(Att_p,3)\,]}(src\_dimdata_{S_x}) \tag{6}$$

where the function *subs* is a user-defined function that extracts a set of characters from an attribute. The first argument is the attribute to be parsed and the second argument of the function defines which set of characters is intended (first, second, etc.). Generalizing, using the extend operator, ε (Baralis and Widom, 1994), that creates new attributes based on data present in other attributes, either by algebraic operations or user-defined operations, it is possible to decompose the contents of any attribute.

Standardization

The standardization task has the goal to conform data that came from distinct sources in different formats but that has the same meaning. This happens frequently in cases that are associated with the use of upper or lower case characters in attributes and with the use, or misuse, of acronyms and abbreviations. Although recognized as standardization, there are a lot of different approaches to deal with the problems raised by these cases. In the case of upper and lower case characters, a system (or an user-defined function) will suffice in conforming the data. However dealing with acronyms and abbreviations requires the use of auxiliary tables to match and substitute the attribute to be treated. Unfortunately, it happens often to be necessary to do the standardization task by hand.

For the upper and lower case standardization scenario, let us assume that attribute Att_x, which was previously generated by the standardization operation, should be in upper case letters. Therefore, Equation 7 presents the creation of a new attribute based on the Att_x attribute but with all letters in uppercase, and Equation 8 removes the old Att_x attribute from table preserving the new one. Finally, Equation 9 renames the new attribute to the old name restoring the original structure of the table.

$$Temp1 \leftarrow \varepsilon_{[Att_{x1}=upper(Att_x)\,]}(src\_dimdata\_d_{S_x}) \tag{7}$$

$$Temp2 \leftarrow \pi_{BK_{S_x},Att_1,\dots,Att_p,Att_{x1},Att_y,Att_z}(Temp1) \tag{8}$$

$$src\_dimdata\_u_{S_x} \leftarrow \rho_{BK_{S_x},Att_1,\dots,Att_p,Att_x,Att_y,Att_z}\,(Temp2) \tag{9}$$

With RA trees it’s quite simple to show how this set of operations works and understand clearly the sequence of operations followed. If there was a need to convert an attribute to a lower case then the approach to follow would be similar to the presented in Figure 38.

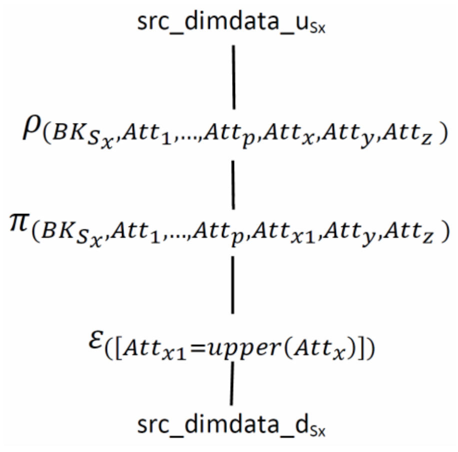
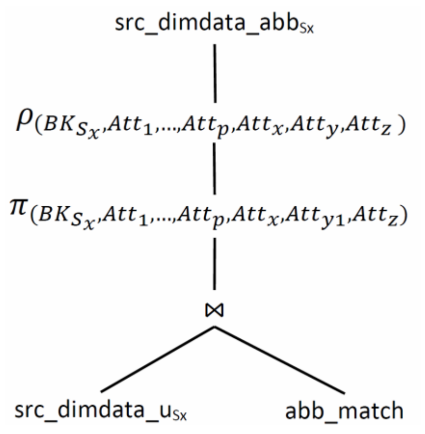

src_dimdata_u$_{Sx}$
$\rho(BK_{S_x}, Att_1, \ldots, Att_p, Att_x, Att_y, Att_z)$
$\pi(BK_{S_x}, Att_1, \ldots, Att_p, Att_{x1}, Att_y, Att_z)$
$\varepsilon([Att_{x1} = upper(Att_x)])$
src_dimdata_d$_{Sx}$

Figure 38 - RA tree representation for upper case transformations

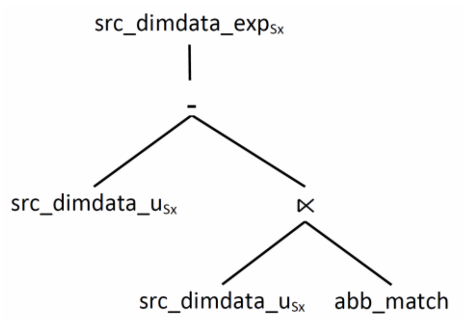
Another standardization scenario we have to deal with is the existence of acronyms and abbreviations. To conform an attribute that is supposed to store abbreviations/acronyms of some sort, first an auxiliary table must be created integrating all possible values encountered in the source table and their correspondence in order to perform transformations required. In this particular case, and since there is a lookup in an auxiliary table, new representations of the abbreviations/acronyms might appear in the source table having no correspondence in the auxiliary table. In this case those tuples must be stored in a temporary table that will require an expert intervention in order to establish the correct correspondence in the auxiliary table and rerun the transformation for these tuples. In this case the attribute Att_y stores abbreviations but previous data analysis reports that there are different abbreviations for the same meaning. Let us also assume that an auxiliary table was created to store all known correspondences (Equation 10).

$$abb\_match = \langle Att_y, Att_{y1} \rangle \tag{10}$$

The task for conforming data is basically represented by the result of a join operation identifying the conformed abbreviation that will be stored in the data warehouse (Figure 39 (a)), and afterwards the identification of tuples that have attribute values with no match in the auxiliary table thus in need of an expert attention (Figure 39 (b)). The proposed operations reflect the steps needed to conform abbreviations, but the approach to deal with acronyms is the same.


src_dimdata_abbSx
$\rho(BK_{S_x}, Att_1, \ldots, Att_p, Att_x, Att_y, Att_z)$
$\pi(BK_{S_x}, Att_1, \ldots, Att_p, Att_x, Att_{y1}, Att_z)$
⋈
src_dimdata_uSx
abb_match

a)



b)

Figure 39 - (a) Conforming abbreviations and (b) identifying non correspondences

The majority of the cleaning tasks needed to be done in the ETL process fall into the category of normalization and correction, where data was introduced in the OLTP systems with errors. The common approach to deal with these errors is the same has presented in the previous task. First, we need to identify variations of the correct value, build an auxiliary table with all valid correspondences (or mappings) between dirty values and clean values, and then conforming the data as necessary. Data not treated should be stored in a quarantine data staging table to be examined latter by an expert that will diagnose and correct (if possible) the problems, providing a correction patch that will be applied in future populating processes, i.e., insert all correspondences needed in the auxiliary table.

Duplicate data in source systems is common and identifying them is not an easy task. It is even more difficult to deal with these duplicates after detecting them. To avoid future problems, data in the source systems should be corrected in order to avoid future duplicates. However, in order to identify duplicates there must be an attribute that should be unique, but for some reason is not, that will allow us to determine these duplicates and deal with them.

Let us assume for our demonstration that the attribute Att_z stores information that should be unique in the dimension, i.e. the attribute Att_z, which was part of a larger attribute in the beginning of our demonstration, identifies each tuple. One method to identify duplicates is to find if a given data instance is present in more than one tuple and therefore violating the uniqueness of the value stored in Att_z (Figure 40). If such records exist then they must be stored apart for expert supervision.

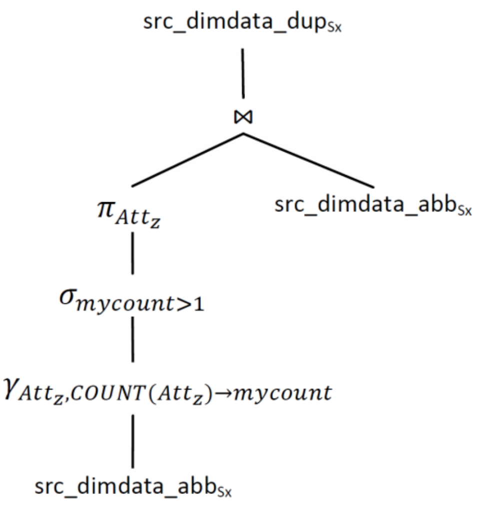


Figure 40 - Eliminating duplicates

After identifying the duplicates, they should be removed from the dimension table in order to proceed to the next task, probably data loading. All the duplicate detections will wait for expert intervention in a specific quarantine data staging (Equation 11).

$$src\_dimdata\_final_{S_x} \leftarrow src\_dimdata\_abb_{S_x} - src\_dimdata\_dup_{S_x} \tag{11}$$

4.3 Data Conciliation

In order to take full advantage of a DWS, all potential sources of data must be studied and analyzed for current or future integration in its DW. This multitude of sources increases the rate of failure if not dealt with proper care and attention.

In today's organization is common to find different transactional systems that evolved through different periods in the organizations' life, probably even in different departments or services units. Nevertheless, it is also likely to find several representations of common concepts in these systems.

For instance, if one system deals in some way with clients and another system deals with customers, several misrepresentations of data might occur, like different codes, names, addresses or other information for a same entity. The correct representation of this information in a data warehouse is often the most difficult tasks that one may find in an ETL system which aims at maintaining the data warehouse up-to-date and correct.

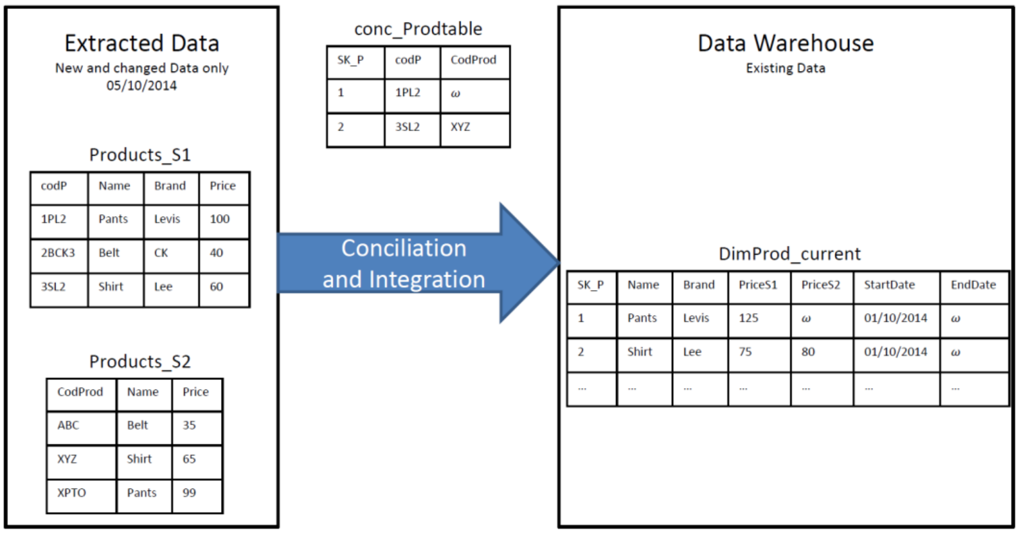
The problem of integrating data from heterogeneous sources is mainly categorized in two aspects: schema and semantics heterogeneity. Schema integration has been widely studied by the database research community (Lenzerini, 2002) and culminated in the presentation of several prototypes and algorithms (Rahm and Bernstein, 2001). Semantics integration is also a challenge that has been studied by several researchers and analyzes problems not only at schema-level but also at instance-level (Rahm and Do, 2000, Kedad and Métais, 1999). Nevertheless these approaches do not formalize any set of logical steps needed to maintain and integrate data coming from heterogeneous sources inside a DW.

Data, once extracted from a source S_x, has to be integrated, therefore transformed, in order to represent a unified and common view in the DW. The problem arises when these heterogeneous sources contain just partial information when in comparison to the data stored in the DW, and also when we have the need of storing historical information about changes that occur in the sources. This transformation phase is then comprised of several processes, mainly dedicated to the cleaning, conciliation and integration with or without dealing with changes.

4.3.1 Conciliation Phase

As already stated, the conciliation phase parses source data and matches it to existing integrated data already present in the DW. In this process two possibilities occur, data is matched and therefore integrated into a common representation in the DW, or data is unmatched, which means that it is new information that has no correspondence to existing data in the DW, and therefore it must be added to the data warehouse with the correct transformations which also means that all auxiliary conciliation support tables must be updated with this new information.

An example of a conciliation scenario is presented in Figure 41 where the extraction process detected new or changed tuples in the Products' relations from two sources that must be conciliated and integrated into the Data Warehouse. Therefore the assignment of the correct surrogate keys to the extracted data must be modelled in order to successfully integrate the data into the data warehouse and also to preserve history when changes in the source systems occur.



Products_S1

codP	Name	Brand	Price
1PL2	Pants	Levis	100
2BCK3	Belt	CK	40
3SL2	Shirt	Lee	60

Products_S2

CodProd	Name	Price
ABC	Belt	35
XYZ	Shirt	65
XPTO	Pants	99

conc_Prodtable

SK_P	codP	CodProd
1	1PL2	ω
2	3SL2	XYZ

DimProd_current

SK_P	Name	Brand	PriceS1	PriceS2	StartDate	EndDate
1	Pants	Levis	125	ω	01/10/2014	ω
2	Shirt	Lee	75	80	01/10/2014	ω
...	...	...	...	...	...	...

Figure 41 - Example of extracted data that must be conciliated and integrated into the data warehouse

Notice that the conciliation table is used to map business keys to surrogate keys, and that in the example shown only a few correspondences exist in comparison with data extracted from sources. That means that in the end of the conciliation process new surrogate keys have to be created and correspondences to current surrogate keys must be updated.

Continuing with the assumption that the data warehouse is based in the dimensional model proposed by (Kimball et al., 1998) which is basically organized as follows: fact tables that store events or facts and dimension tables that act as qualifiers of those facts. These tables are normally, in a simplified approach, organized in a star schema, where the fact table references the dimension tables. Therefore first we deal with dimension data and only after we deal with the facts. In both cases, for the conciliation phase, we need a conciliation table, for each dimension, that helps us with mapping business keys from the heterogeneous sources into the surrogate keys used in the DW. Surrogate keys are a normal approach to avoid using complex business keys in the DW, not only because of performance issues but also because it is not always possible to store business keys in the DW mainly when dealing with heterogeneous sources.

Consider a relation $conc\_dimtable$ as a list of attributes having a surrogate key, and a list of business keys (derived directly from the heterogeneous sources previously referred) – (Equation 12).

$$conc\_dimtable = \langle SK, BK_{S_1}, \dots, BK_{S_n} \rangle \tag{12}$$

in which SK is the surrogate key, normally a natural number, $BK_{S_1}, \dots, BK_{S_n}$ are the attributes that belong to the business key of a source S_x, where $1 \leq x \leq n$. A business key from a specific source might not be a single attribute but rather a set of attributes, which imposes that some additional adjustments must be made to the equation. The business keys will allow us to integrate data from different sources correctly, since they will map an instance, from a specific source, to a surrogate key that will in turn guarantee data integrity in the DW.

The common structure of source data after being extracted and cleaned was presented in (Equation 3).

A graphical approach to the conciliation process is presented in Figure 42, where for each source processed a matching task is performed to map the business keys into the data warehouse surrogate keys. Whenever a source tuple is unmatched, it is stored in a separate table for further revision since it will probably be new data that will need to be processed and updated with a matching correspondence in the conciliation auxiliary table ($conc\_dimtable$).

The matching process can be expressed in RA and is presented using RA trees in Figure 43. In Figure 43(a) the join operation between source data and the auxiliary table will result in determining the matching tuples and the correct correspondence to the surrogate key. In Figure 43(b) the subtract operation is used to remove from source data those tuples that were matched, therefore remaining those that do not have correspondence in the auxiliary table. These tuples must then be dealt with, normally through human interaction that determines the correct correspondence to a surrogate key (new or already defined) in the conciliation auxiliary table.

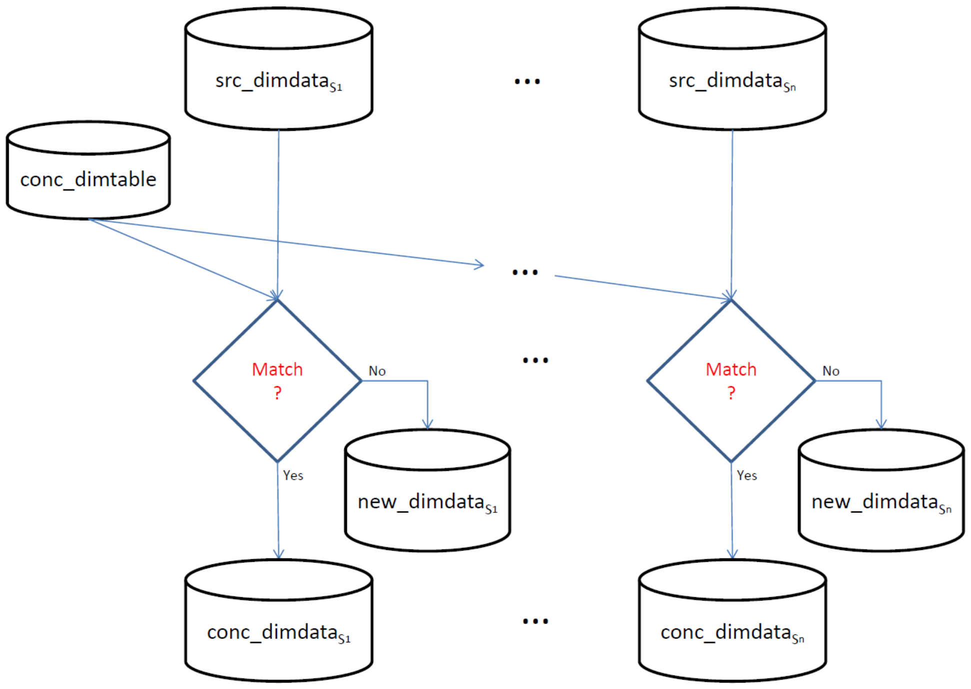


Figure 42 - ETL Conciliation process.

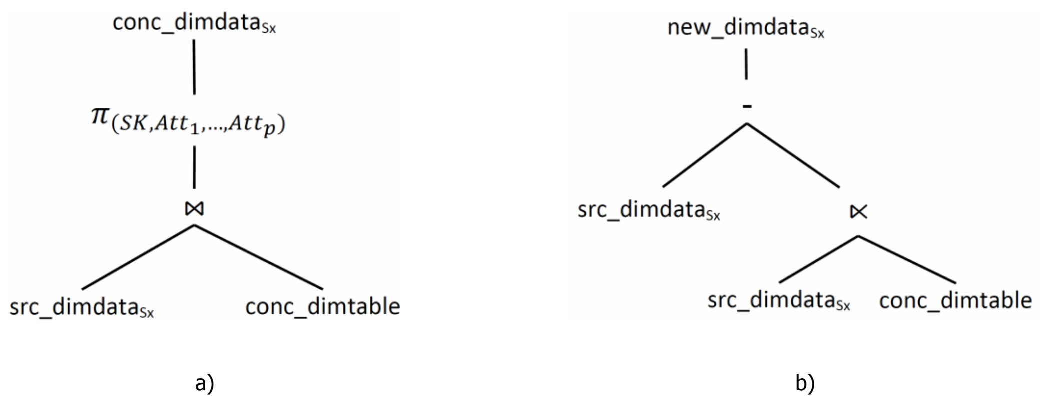


Figure 43 - (a) Conciliated Data; (b) Unmatched Data.

The structure of the result table after the conciliation phase is very similar to the source table, the only change is the presence of the surrogate key (SK) instead of the business key (BK_{S_x}) – (Equation 13).

$$conc\_dimdata_{S_x} = \langle SK, Att_1, \ldots, Att_p \rangle \tag{13}$$

After the conciliation phase, begins the integration phase, where data coming from different sources is integrated into the dimension table. In this phase, dealing with changes (Santos and Belo, 2011b), normally called changing dimensions, has to be adjusted to the fact that we are dealing with multiple sources.

4.4 Slowly Changing Dimensions

Perhaps one of the most important quests is to ensure consistency (and correctness) in the data warehouse's structures, especially at dimension table’s level. As is known, dimension tables are used to provide subject oriented information by providing data elements to filter, aggregate or describe facts in a data warehouse. Thus, it is quite important that dimensions remain consistent accordingly in some data warehouse's time frame even when some values of their attributes change over time. When this occurs, dimension tables are classified as SCD.

SCD (Kimball and Caserta, 2004) are not a new concept. Often, during the dimensional modeling phase of a data warehouse the term appears frequently, referring to dimension tables that have attributes whose values could change over time. Data warehouse designers sometimes select and integrate attributes in dimension tables that must have special populating procedures to ensure that their values keep consistency as time passes, ensuring that the DW, and in particular the schema where they are integrated, keeps an integrated state in all time frames considered in its own structure (Griffin et al., 2005). Even requiring such special procedures, SCD are very important to keep up-to-date dimension properties in a data warehouse - an address of a customer or supplier, a production standard cost of a product, an employee's salary, just to name a few, guaranteeing that facts have an up-to-date dimensional support. Special oriented ETL processes are responsible to maintain SCD, acting accordingly to the updating strategy previously defined.

The way they are implemented followed all the dimensional modeling requisites established for every dimension table classified as SCD; the same occurs if the dimension was classified as *Rapidly Changing Dimension* (RCD); its treatment is quite similar to SCD not differing in any relevant aspect, unless the form it varies in time.

Usually, every SCD process starts with a changing event occurrence in a specific information source (one or more) that provides the data to feed the dimension table. Usually, a CDC process detects the update event and records it accordingly to some predefined SCD requisites - temporal labels, metadata format, operation markers, operational systems application or user, etc. After, in a DWS staging area the remaining part of the SCD process is triggered and the SCD strategy applied. Although we always speak in terms of process, SCD strategies are applied at the attribute level, for a particular dimension table. For each dimension table, DWS designers use to define what kind of updates attributes will be modified over time. They must prevent all the possible cases of attribute changing, once it is not very recommendable to change dimension table structures when the DWS is already in production. Over a same SCD we could have to apply distinct updating strategies for a single record. Frequently, this involves different processes for different SCD strategies.

Based on the most traditional approaches founded in the literature to deal with SCD, the following types can be identified (Kimball and Ross, 2002, Kimball and Caserta, 2004, Kimball et al., 2008):

Type 1 - whenever occurs a change in an attribute classified as type 1, the change must be reflected in the dimension's attribute overwriting its previous value; no kind of history will be kept about the values changing.

Type 2 - this is the first SCD strategy that allows keeping historical data about changes in dimensions' attributes; every change is recorded as a new dimension record keeping the old one for historical reasons; all the records about a same entity are connected through conceptual pointers, which allow navigating among all the changes of a specific entity.

Type 3 - it allows also to maintain historical data, but only to some level of depth, which is the number of times that we must replicate the definition of the attribute for which we want to keep history; for instance, if we want to keep the last three addresses of a customer (depth 3), we must defined three attributes with the same definition to receive the last three address values.

Type 4 - this type considers the use of an auxiliary table especially oriented to keep the changes that dimension's attributes suffer over time, while the dimension table maintain the most recent records; this simplify a lot all the processes required to update the values of SCD attributes and keeps simple all data browsing tasks over the dimension.

Type 6 - basically, this updating strategy is a combination of the techniques used in SCD types 1, 2 and 3, with the objective to usufruct from the advantages of each one of them (Kimball and Ross, 2002, Manh Nguyen et al., 2007).

4.4.1 SCD Type 1

When an attribute is identified as a SCD type 1 attribute, the methodology to deal with the changes is simple, the attribute is simply updated. Since the old attribute value is lost, all future analysis refer to the new value, therefore history is not retained.

Let us assume that the SCD process follows the conciliation process and therefore use $conc\_dimdata_{S_x}$ (Equation 13) as the basis to modeling the SCD Type 1 task.

In a SCD Type 1 the typical structure of the dimension itself is presented in Equation 14 which is the same as Equation 13 therefore simplifying the task, since the only steps needed are the removal of the changed tuples from the dimension (Equation 15, Equation 16 or Figure 44) and the addition of the new and changed tuples that result from the conciliation process (Equation 17).

$$dimension = \langle SK, Att_1, \dots, Att_p \rangle \tag{14}$$

$$Old\_data \leftarrow dimension\_current \ltimes \pi_{(SK)}(conc\_dimdata_{S_x}) \tag{15}$$

$$dimension \leftarrow dimension - Old\_data \tag{16}$$

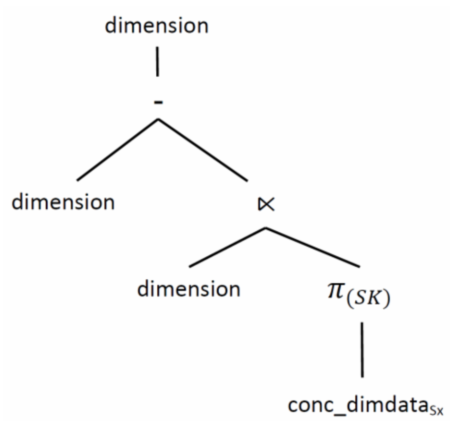


Figure 44 - Removal of changed tuples using a SCD Type 1 strategy

$$dimension \leftarrow dimension \cup conc\_dimdata_{S_x} \tag{17}$$

4.4.2 SCD Type 2

When an attribute is identified as a SCD type 2 attribute, the methodology to deal with the changes is complex. Since history is preserved in the dimension relation, additional attributes are needed to identify the timeframe in which the tuples are valid. Therefore the structure of the dimension relation (Equation 18) is somewhat different than in SCD type 1.

$$dimension = \langle SK, Att_1, \dots, Att_p, StartDate, EndDate, current\rangle \tag{18}$$

where $StartDate$ and $EndDate$ are two Data type attributes, $current$ attribute is typically a bit attribute that identifies if tuple is actual or not.

Let us consider that the type 2 attribute is one of the $Att_1,\dots,Att_p$ attributes. In order to process correctly the changes in the type 2 attribute, we first need to separate current from historical data

(Equation 19) since updates will force current data to become history by changing the $current$ and $DateTo$ attribute.

$$dimension\_current \leftarrow \sigma_{(current=1)}(dimension) \tag{19}$$

Expiring data are those tuples in which the SK is present in both $dimension\_current$ and $conc\_dimdata_{S_x}$ (Equation 20) and must be removed from the dimension (same expression as already defined in Equation 16). To complete the expiring process we add the changed data with $EndDate$ attribute set to the expiring date, normally two days prior to the day the ETL process is running, and $current$ attribute set to 0 which means expired data (Equation 21).

$$Old\_data \leftarrow dimension\_current \ltimes \pi_{(SK)}(conc\_dimdata_{S_x}) \tag{20}$$

$$dimension \leftarrow dimension \cup \pi_{(SK,Att_1,\dots,Att_p,StartDate,today()-2,0)}(Old\_data) \tag{21}$$

The entire process of expiring data can be presented in a RA tree as in Figure 45.

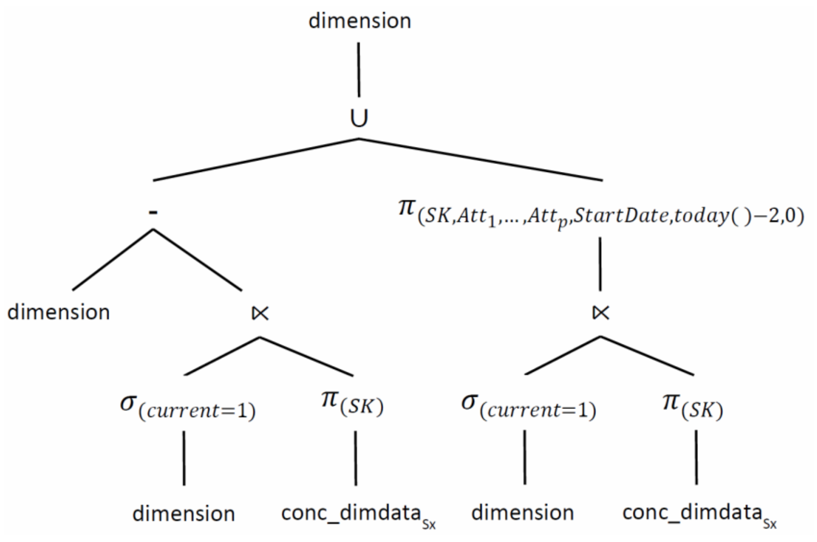


Figure 45 – Expiring data process using a SCD Type 2 strategy

Finally, and after the process of expiring data is concluded, all data is added to the dimension correctly time stamped and current (Equation 22 and Figure 46).

$$dimension \leftarrow dimension \cup \varepsilon_{(StartDate=today()-1, EndDate=\omega, current=1)}(conc\_dimdata_{S_x}) \tag{22}$$

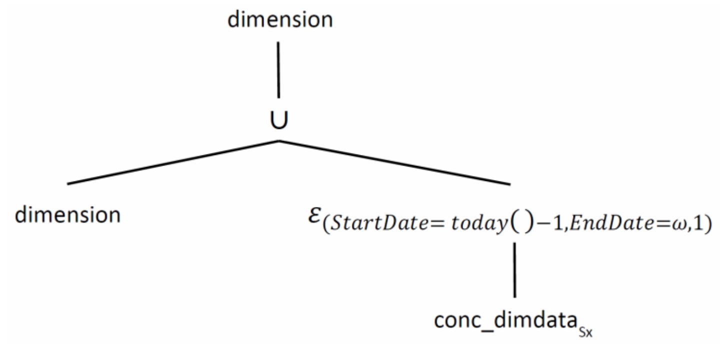


Figure 46 - Adding data using a SCD Type 2 strategy

4.4.3 SCD Type 3

As a compromise to the disadvantages present in the type 2 approach, type 3 is able to store historical changes to the degree needed. This is accomplished by the used of additional attributes that store previous values. Typically only one level is used, but it is possible, but not advisable, to use more. In order to process properly this approach we need to correctly identify the type 3 attribute, which in this case will be Att_p. The structure of the dimension relation therefore reflects this approach (Equation 23).

$$dimension = \langle SK, Att_1, \dots, Att_p, Att_{p\_previous} \rangle \tag{23}$$

in which Att_p is the type 3 attribute, and $Att_{p\_previous}$ is the attribute that will hold the previous Att_p value. This approach requires that whenever a change occurs, the previous value will be stored in attribute $Att_{p\_previous}$ and the current value of the attribute will be stored in Att_p. First step is accomplished through the use of Equation 15 and Equation 16 which removes from the dimension the changed tuples. Second step, in Equation 24 conciliated data is joined with old valued attributes to be added into the dimension.

$$dimension \leftarrow dimension \cup ((conc\_dimdata_{S_x}) \bowtie (\rho_{(SK,Att_{p\_previous})}(\pi_{(SK,Att_p)}(Old\_data)))) \tag{24}$$

Figure 47 represents in a RA tree all the operations needed to maintain a SCD Type 3 up to date.

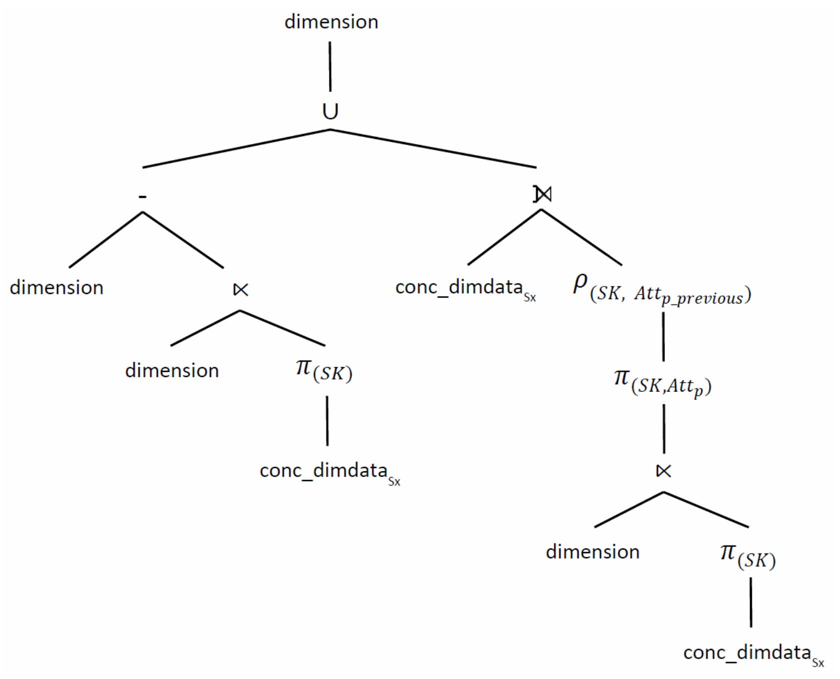


Figure 47 - SCD Type 3 operations

4.5 Data Integration on a SCD Type 4

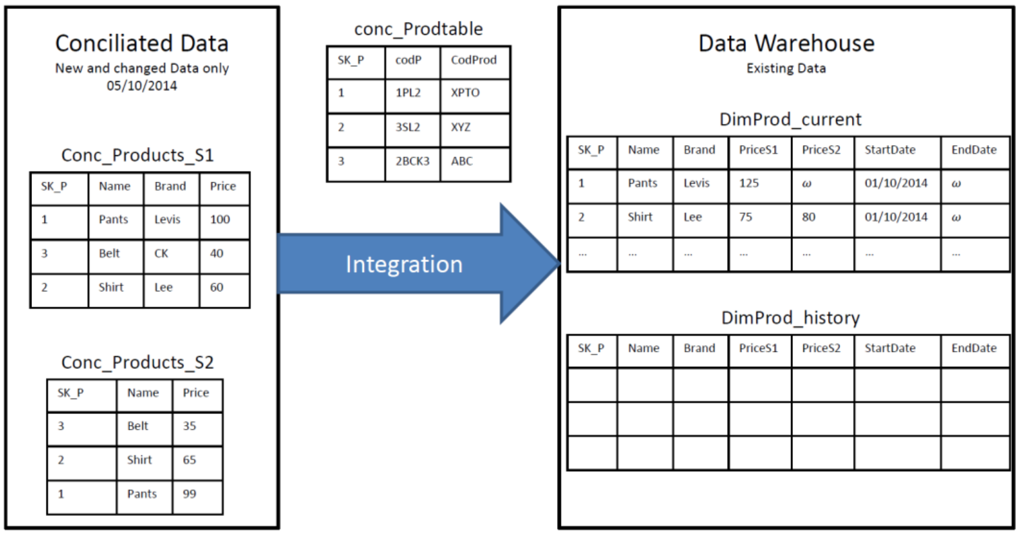
This phase is primarily responsible to load conciliated data into the DW. The challenge is to deal with changed source data.

Let us assume that in a data warehouse the structure of one of its dimensions is comprised of two identical tables (SCD Type 4): one storing current tuples and the other storing historic tuples (Santos and Belo, 2011a). The common structure of such a table is presented in (Equation 25).

$$dimension\_current = \langle SK, A_1, \dots, A_m, StartDate, EndDate \rangle \tag{25}$$

in which SK is the surrogate key, normally a natural number, $A_1, \dots, A_m$ are additional data attributes needed in the dimension and $m >= 1$. Since we are dealing with multi source data, a specific source might not contain all the information to correctly fill all dimension attributes, nevertheless each source attributes exist in the dimension's attributes - $\{ Att_1, \dots, Att_p \} \subseteq \{ A_1, \dots, A_m \}$.

Assuming that the extraction process (CDC) only provides new and changed tuples from source systems to the transformation phase, the methodology to deal with new tuples is somewhat different from the one used to deal with changed tuples, mainly due to the necessity of conforming data. Changed tuples are already integrated into the dimension so the only task needed is to preserve history, i.e., updates that occurred in source systems must be reflected in the data warehouse dimension without losing history, therefore we transfer old data to the historical table and insert changed tuples into the dimension that holds current values (Figure 48).



Conc_Products_S1

SK_P	Name	Brand	Price
1	Pants	Levis	100
3	Belt	CK	40
2	Shirt	Lee	60

Conc_Products_S2

SK_P	Name	Price
3	Belt	35
2	Shirt	65
1	Pants	99

conc_Prodtable

SK_P	codP	CodProd
1	1PL2	XPTO
2	3SL2	XYZ
3	2BCK3	ABC

DimProd_current

SK_P	Name	Brand	PriceS1	PriceS2	StartDate	EndDate
1	Pants	Levis	125	ω	01/10/2014	ω
2	Shirt	Lee	75	80	01/10/2014	ω
...	...	...	...	...	...	...

DimProd_history

SK_P	Name	Brand	PriceS1	PriceS2	StartDate	EndDate

Figure 48 - Example of conciliated data that must be integrated into the DW

In order to simplify the interpretation of the RA model, we present in Figure 49 the RA tree with the operations that identify expired data in the dimension.

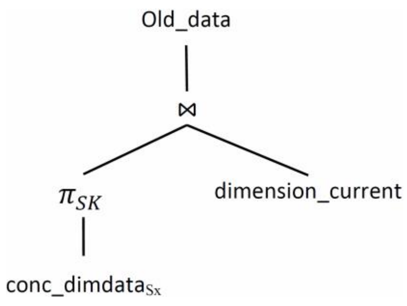


Figure 49 - A RA tree identifying expired data

Figure 50 presents the appropriate operations to transfer expired data from the dimension, since it is only supposed to store current data, to the historical table in order to preserve history.

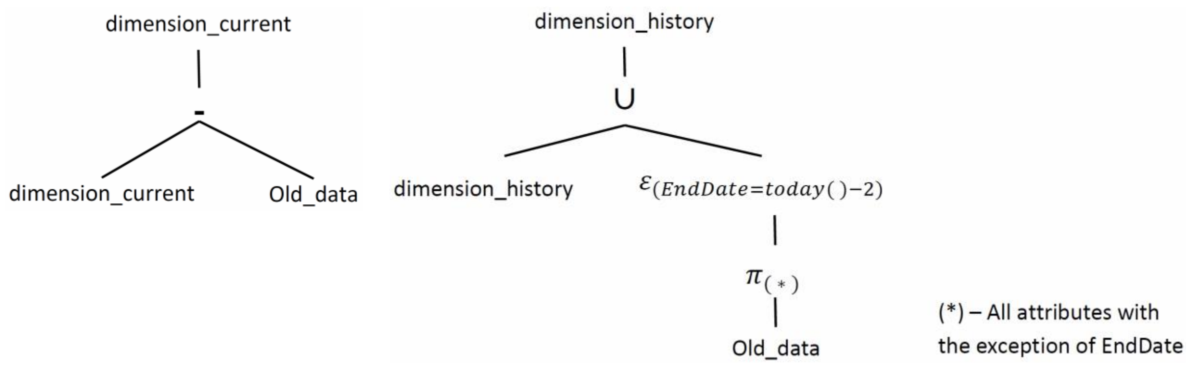


Figure 50 - Transferring expired data from a dimension table to a historical table.

After removing the expired data from the dimension, the last step needed is to add the updated values to the dimension without losing correspondence with data that was obtained from other sources (Figure 51).

Dealing with new data from $conc\_dimdata_{S_x}$ is based on the determination of tuples which surrogate keys do not have correspondence in the dimension. Those tuples are added to the dimension after being time stamped, identifying the validity of the information, with the particularity of leaving the dimension attributes that are not known in the current source in a *null* state – this is achieved by the left outer join (Figure 52).

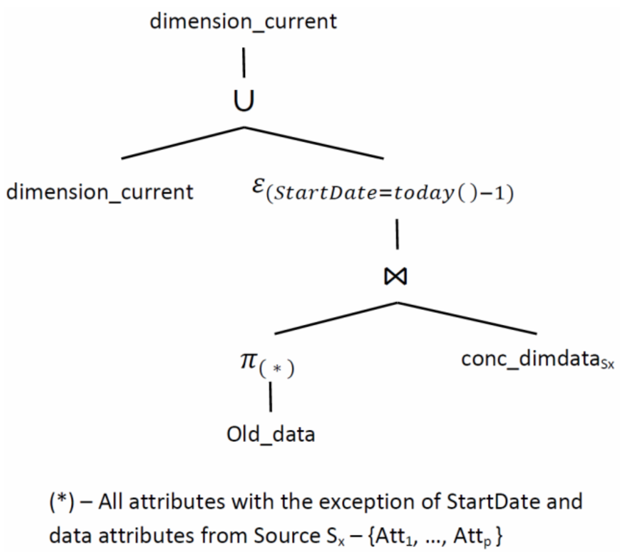

dimension_current
$\cup$
dimension_current
$\varepsilon_{(StartDate=today()-1)}$
$\bowtie$
$\pi_{(*)}$
conc_dimdata$_{Sx}$
Old_data
(*) – All attributes with the exception of StartDate and data attributes from Source S_x – {Att_1, ..., Att_p}

Figure 51 – Updating dimension with changed data

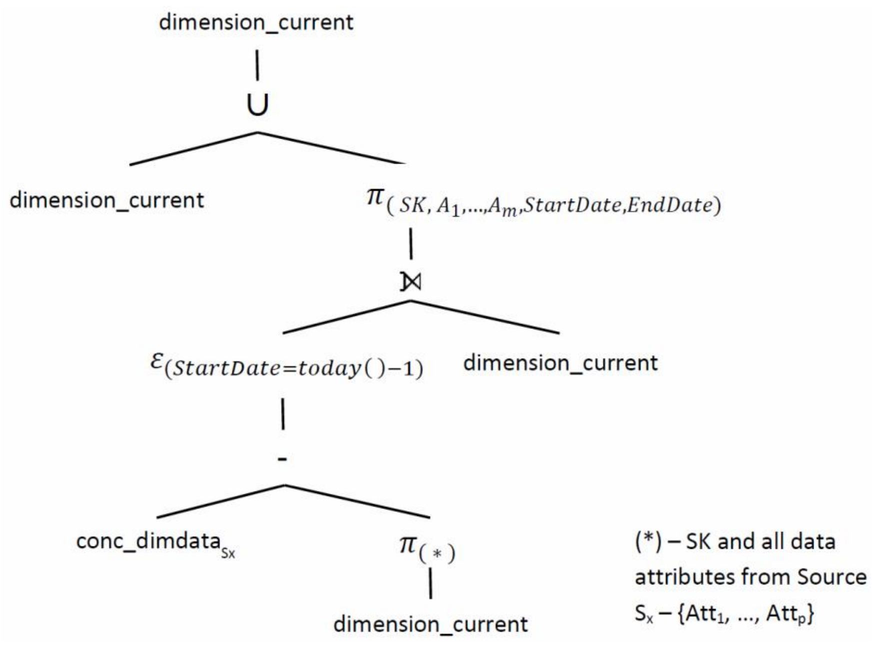
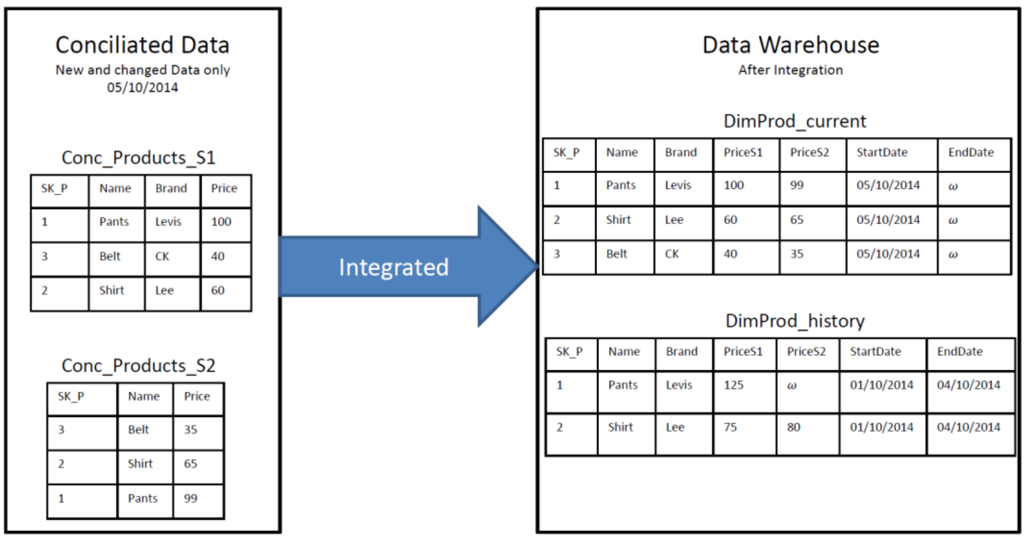

dimension_current
$\cup$
dimension_current
$\pi_{(SK, A_1,...,A_m,StartDate,EndDate)}$
$\rtimes$
$\varepsilon_{(StartDate=today()-1)}$
dimension_current
-
conc_dimdata$_{Sx}$
$\pi_{(*)}$
dimension_current
(*) – SK and all data attributes from Source S_x – {Att_1, ..., Att_p}

Figure 52 – The RA tree that adds new data into the dimension

The result of the integration process for the example shown in Figure 48, would be like the one presented in Figure 53.

4.6 Surrogate Key Pipelining

After dealing with dimension data, the process of loading facts into the data warehouse becomes simplified. The only task we need to perform now is the correct translation of the business keys to the correspondent surrogate keys. This process, also known as SKP, lookups up, in sequence, on the auxiliary conciliation tables for the correct surrogate key (Figure 54).



Conc_Products_S1

SK_P	Name	Brand	Price
1	Pants	Levis	100
3	Belt	CK	40
2	Shirt	Lee	60

Conc_Products_S2

SK_P	Name	Price
3	Belt	35
2	Shirt	65
1	Pants	99

DimProd_current

SK_P	Name	Brand	PriceS1	PriceS2	StartDate	EndDate
1	Pants	Levis	100	99	05/10/2014	ω
2	Shirt	Lee	60	65	05/10/2014	ω
3	Belt	CK	40	35	05/10/2014	ω

DimProd_history

SK_P	Name	Brand	PriceS1	PriceS2	StartDate	EndDate
1	Pants	Levis	125	ω	01/10/2014	04/10/2014
2	Shirt	Lee	75	80	01/10/2014	04/10/2014

Figure 53 - Example of the final result of the conciliation and integration process

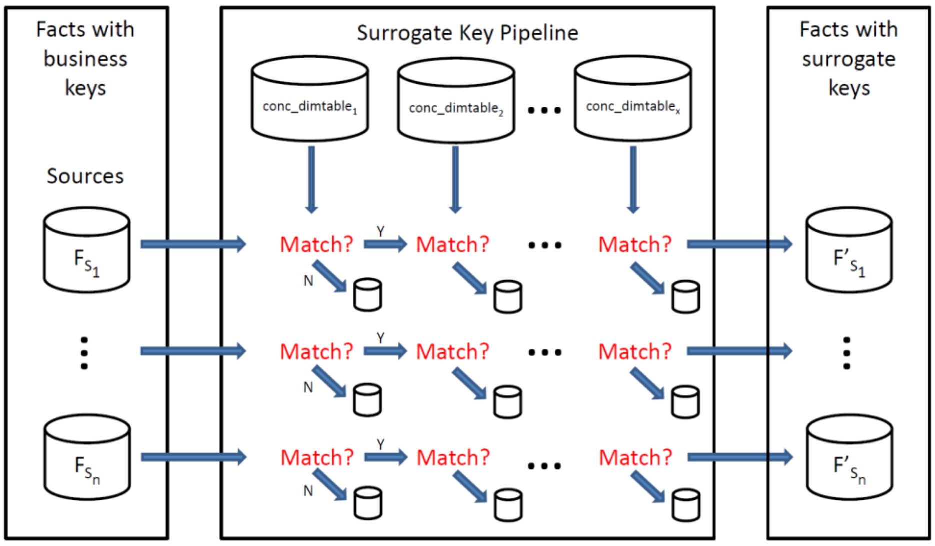


Figure 54 - SKP with heterogeneous sources

When processing facts from different sources, each tuple lookups for the dimension's correspondent surrogate key (given their business key), and proceeds in the same way for all the remaining dimensions. The tuple is ready to be loaded into the fact table when all processes conclude with success. Error occurrences or unmatched tuples must be signaled and dealt with in the recuperation phase that is normally a human interaction phase.

The SKP can also be modeled in RA, through a series of operations that prepare the fact table to be loaded into the data warehouse and separate the unmatched tuples for further revision.

Consider the following definition a common structure of a fact table from source before the SKP (Equation 26).

$$F_{S_1} = \langle BK_{dim_1}, \dots, BK_{\mathrm{dim}_z}, m_1, \dots, m_p \rangle \tag{26}$$

where BK_{dim_x} is the business key and $1 \leq x \leq z$ of the z dimensions present, and $m_1, \dots, m_p$ are the measurements present in the fact table.

and the structure of the fact table after the SKP (Equation 27)

$$F'_{S_1} = \langle SK_{\dim_1}, \dots, SK_{\dim_z}, m_1, \dots, m_p \rangle \tag{27}$$

where SK_{dim_x} is the correct surrogate key of each dimension and $1 \leq x \leq z$, and $m_1, \dots, m_p$ are the measurements present in the fact table.

The steps needed to substitute the business keys for the surrogate keys for each dimension are presented as a RA tree in Figure 55.

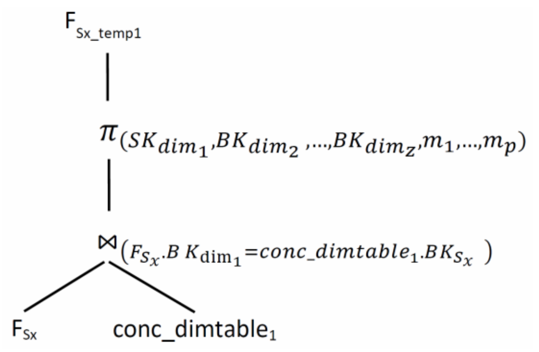

F_{Sx_temp1}
$\pi(SK_{dim_1}, BK_{dim_2}, \dots, BK_{dim_z}, m_1, \dots, m_p)$
$\bowtie(F_{S_x}.BK_{\dim_1} = conc\_dimtable_1.BK_{S_x})$
F_{Sx}
conc_dimtable$_1$

Figure 55 - Surrogate Key mapping for a dimension

All the following surrogate key substitutions follow the same approach with the necessary adjustments to the RA expressions.

The business keys from the fact table that for some reason don't find the correct correspondence in the conciliation table must be signaled for human revision therefore stored appropriately as presented in (Figure 54). The RA tree modeling this behavior is presented in Figure 56.

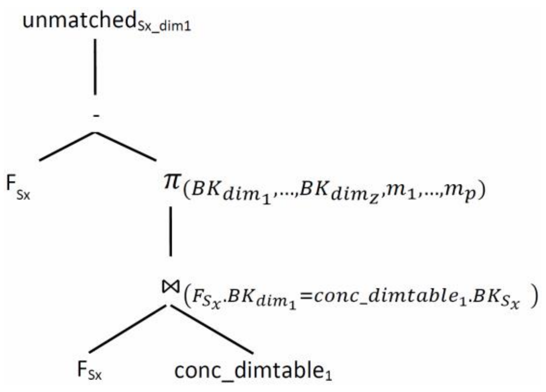

$unmatched_{Sx\_dim1}$
$-$
F_{Sx}
$\pi_{(BK_{dim_1},\ldots,BK_{dim_Z},m_1,\ldots,m_p)}$
$\bowtie_{(F_{S_X}.BK_{dim_1}=conc\_dimtable_1.BK_{S_X})}$
F_{Sx}
$conc\_dimtable_1$

Figure 56 - Unmatched Surrogate Key mappings

Chapter 5

Specifying a Real ETL System

Internet was mainly used to connect services, users and organizations through information exchange, both private or public information, by using email protocol (POP, IMAP, STMP), news transfer protocol (NNTP) or Internet websites (HTTP). This sort of communication was based in the need to make available information to interested parties. Information and news were broadcasted by organizations mainly through corporate websites and via email through mailing lists. Trending subjects were mainly determined by news companies, since the news broadcast choices were their responsibility.

This approach to news dissemination has somewhat changed since the appearance of multiple, and very successful, Virtual Social Networks (VSN) – also referred frequently as Online Social Networks. There are several popular and different VSN with different characteristics that nowadays have several million active users in a daily basis. According to (Pallis et al., 2011), VSNs are classified by their scope (entertainment or business), their data model (centralized or decentralized), their system model (web-based or cloud-based) and network model (user-oriented or content-oriented). The revolution in news dissemination comes from the use of these networks where users can broadcast, comment, publish and share information through their network of friends, colleagues and acquaintances that in turn can do the same thing (Kumar et al., 2010). Therefore news and trending subjects are no longer originating only from news corporations but also from influential individuals that use VSNs to spread their opinion (Mislove et al., 2007, Yu et al., 2011).

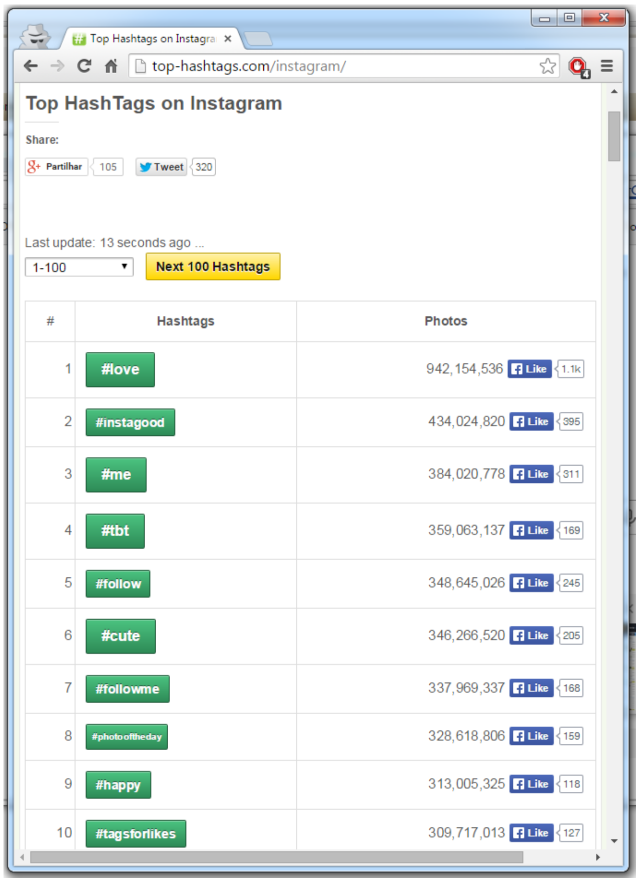
With millions of daily active users, VSNs are also an enormous source of data and information. VSN users post and comment over news available and/or trending subjects, thus making these networks a good source of information for users and also for search engines (Gloor et al., 2009). It is now common for users to search for topics of interest in their VSN instead of search engines too. Nevertheless VSNs are not search engines and seeking for correlated data in users' post messages was not an easy task. Metadata that classified each post would facilitate searching and indexing information, and as such the use of hashtags appeared and became popular, primarily in Twitter social network and afterwards in every post and message social network users felt the need for it.

As we know, a hashtag is a word or an unspaced phrase that starts always with the hash character (#). Using hashtags in a common post acts as some kind of metadata over its contents. In the last few years, hashtags have become very popular in social network sites and applications. They also become very useful for searching similar tagged messages, simply because search engines return messages based in those tags. Additionally, social networks have used hashtags to check for trending subjects in any temporal time window, as it happens with Instagram (Figure 57).

With this background in mind, we will demonstrate the application of the modeling proposal presented in the previous chapter over a case study involving the development of a centralized repository, and in particularly the modeling phase of the ETL component that will gather, clean, conform and integrate heterogenic data gathered from two different VSNs.

5.1 The Information Sources

The case study we selected includes two different VSNs. The first one virtual social network (SN_A) is mainly used as a network of users and organizations with specific professional skills, much like LinkedIn, where the primary purpose is to present, divulge and comment news and information about personal skills and organization achievements. The second virtual social network (SN_B) is a recreational network of users that communicate and share media contents with their family and friends, much like Facebook.



Top Hashtags on Instagram

top-hashtags.com/instagram/

Top HashTags on Instagram

Share:

Partilhar 105 Tweet 320

Last update: 13 seconds ago ...

1-100 Next 100 Hashtags

#	Hashtags	Photos
1	#love	942,154,536 Like 1.1k
2	#instagood	434,024,820 Like 395
3	#me	384,020,778 Like 311
4	#tbt	359,063,137 Like 169
5	#follow	348,645,026 Like 245
6	#cute	346,266,520 Like 205
7	#followme	337,969,337 Like 168
8	#photooftheday	328,618,806 Like 159
9	#happy	313,005,325 Like 118
10	#tagsforlikes	309,717,013 Like 127

Figure 57 – Top 10 HashTags – extracted from (Instagram, 2015)

The logical design of the first source of this case study is presented in Figure 58. This VSN defined in its business model that customers were able to create and own several login identifications (LoginID). This would allow each customer to assume different profiles and post accordingly to the profile used. For instance, a customer might have a private profile for family and friends and another profile that would act as his work profile for colleagues, partners, clients and acquaintances. Once logged in, a customer can post messages in the VSN and associate hashtags to the post. Hashtags in this VSN are not standardized and can be freely created by customers.

Another characteristic of this VSN is the ability to comment another users' post message and use hashtags on the comments. Parallel to posting and commenting, customers can browse other customers' messages and support or disapprove the message by creating an event that classifies the post accordingly. This kind of behavior is also common on VSN and can contribute to finding influential users. Finally, and to complement the classifying event, the owner of the post is notified whenever his post messages receive support or otherwise disapproval.

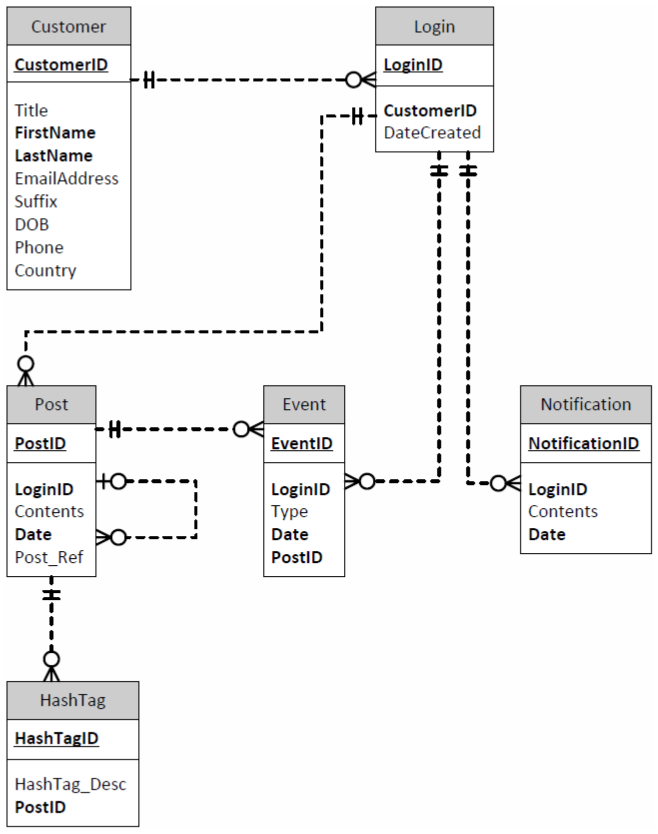


Figure 58 – The Logical Model of SN_A

The logical design of the second source of this case study is presented in Figure 59. Although the logical design of this VSN is somewhat more elaborated than the previous VSN, the working methodology is basically the same. The primary difference is the fact that each Client can only own one account or login. After logging in the VSN, a client can post new messages, and classify that message according to its type (text, video, photo, etc.) and add attachments to the message that are stored in the network. The client can also select from a predefined set of hashtags which ones are the most adequate to categorize his post message. Clients can also browse existing messages, reshare them, comment and tag them as they would normally do in a normal post message. Besides posting new messages and resharing other users' posts, clients can comment post messages and tag those messages with hashtags. It's clear for this VSN business that post messages are somewhat different from comments mainly by the fact that comments cannot have attachments and are limited to text only.

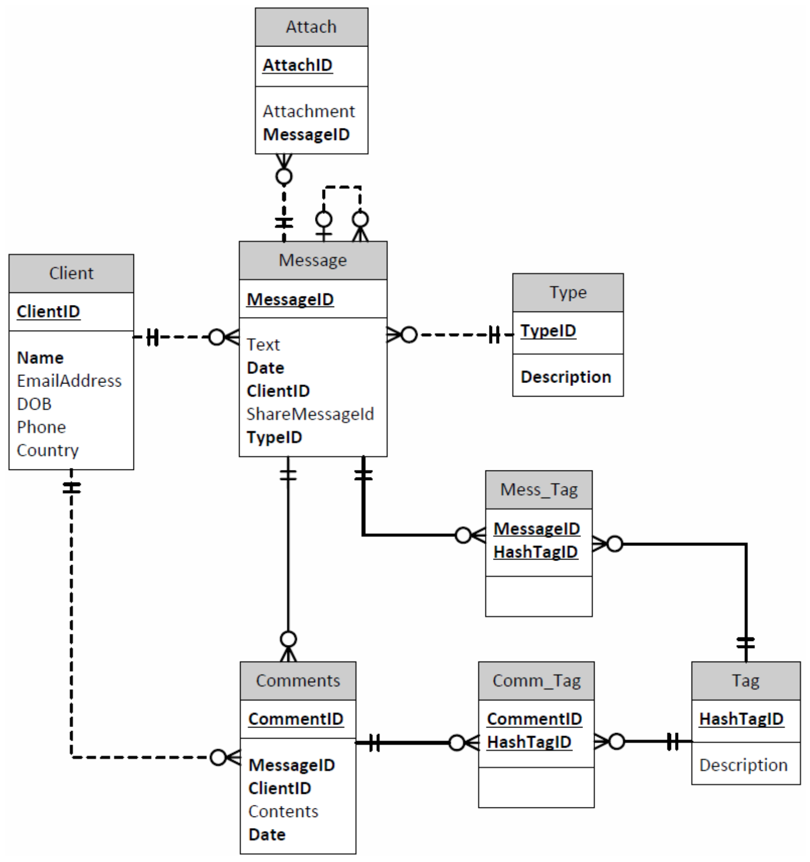


Figure 59 - The Logical Model of SN_B

5.2 The data mart *HashTags*

Users post and comment in both networks using hashtags to better characterize the message they wanted to transmit. After a real world event, users tend to opine heavily in social networks and trending subjects often arise very fast, like for instance when people made a stand against the terrorism attack in Paris. The hashtag #JeSuisCharlie rocketed to the top of hashtags used in January with a peak of six thousands post messages per minute in Twitter social network.

Being aware and detecting emerging trends through the analysis of hashtag content in post messages is critical to maintain a social network on top of things. Therefore, and since we are dealing with two heterogeneous sources, a common unified repository is needed to store the data from both sources. A data mart is a possible answer to this need, as it can serve as a data repository and data source for mining and reporting over common unified data. By taking into account the primary advantages of a data mart, we designed and implemented a specific one – "HashTags" – especially conceived to receive data coming from the two social networks information sources. Therefore, the dimensional model of this data mart (Figure 60) should support the previous identified needs to store the different hashtag data from the social networks. The grain of the fact table is the use of hashtags in comments or posts in the social networks that act as sources in a daily basis. However, if the period of analysis changes, from a daily basis to an hourly basis, or different, the changes in the model are restricted to the dimension time (dimTime) and corresponding relationship to the fact table. Since a hashtag can be used several times in the period chosen, the fact table must contain an attribute that counts the amount of times the hashtag was used in that day, in what is normally called measurement.

5.3 Modeling the ETL System with Relational Algebra

In this section we will present an instantiation of the RA patterns, proposed previously in Chapter 4, to model the populating process of the star schema represented in Figure 60. We considered that the dimension $dimTime$ is fully populated in advance and therefore will not be included in the modeling scenario. The dimension $dimSocialNetwork$ is also already populated with what categorizes the sources - Social Network A (SN_A) and Social Network B (SN_B) -, but is fully prepared to receive other additional sources. The remaining dimensions and the fact table need to be populated with data being extracted from both the Social Networks OLTP systems. Since we are dealing with two different sources, the primary challenge will be the conciliation of data extracted. Several conciliation tables will be needed mainly to standardize common concepts and to map business keys into the surrogate keys of the dimensions. After loading the dimensions appropriately, the fact table will also be populated with aggregated data, after the process of translating the business keys to the fact table primary keys, in what is normally called SKP process.

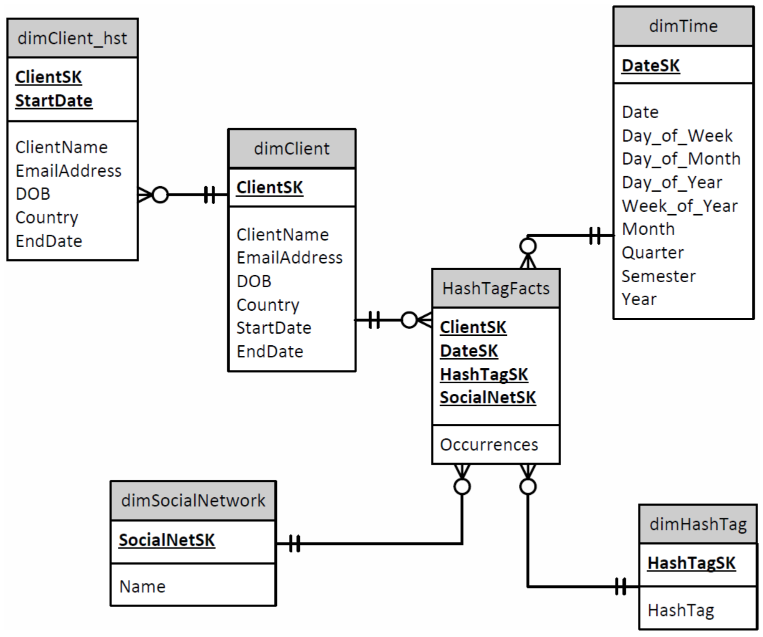


Figure 60 - The logical schema of the data mart "HashTags"

5.3.1 Populating the dimension table $dimHashTag$ with SN_A's data

The table dimension $dimHashTag$ is populated with data coming from both social networks' information sources, which imposes that the correspondent information flows must be modeled separately. In SN_A users can create hashtags freely, which originates several different hashtags for a same purpose. Therefore we need to make facts standard. Thus, the first process to run is to gather data from SN_A as we proposed before in Section 4.1. Then we continue the normalization (correction) of the hashtag values using an auxiliary conciliation table, as proposed in Section 4.2.2 in the topic Standardization.

The extraction process gathers all data from the table $HashTag$ and stores it into the table $src\_dimhashtag_{S'_a}$ (Equation 28) and through the use of the patterns proposed before in Section 4.1. New or changed data is stored in $src_dimhashtag_{S_a}$for future transformation (Equation (30).

$$src\_dimhashtag_{S'_a} = \langle HashTagID, HashTag\_Desc, PostID\rangle \tag{28}$$

$$prev\_src\_dimhashtag_{S'_a} = \langle HashTagID, HashTag\_Desc, PostID\rangle \tag{29}$$

$$src\_dimhashtag_{S_a(HashTagID,HashTag\_Desc,PostID)} \leftarrow src\_dimhashtag_{S'_a} - prev\_src\_dimhashtag_{S'_a} \tag{30}$$

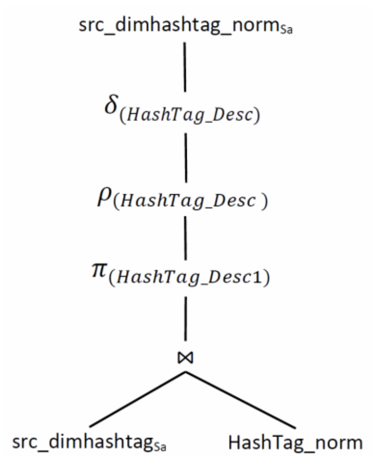
$$prev\_src\_dimhashtag_{S'_a} \leftarrow src\_dimhashtag_{S'_a} \tag{31}$$

The table $HashTag\_norm$ was used for conforming hashtag values (Equation 32), following the approach presented previously in section 4.2.2 in the topic Normalization and Correction, in which $HashTag\_Desc$ is the attribute value coming from the source and $HashTag\_Desc1$ is the conformed value of the attribute.

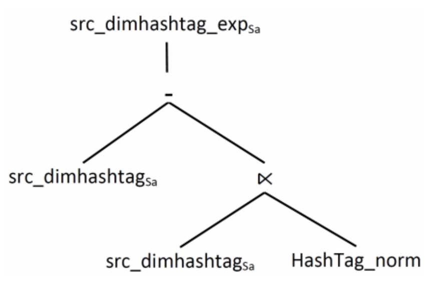
$$HashTag\_norm = \langle HashTag\_Desc, HashTag\_Desc1\rangle \tag{32}$$

The operations of Figure 61 were defined to conform the hashtag values into the table $src\_dimhashtag\_norm_{S_a}$, and unmatched tuples into the table $src_dimhashtag_exp_{S_a}$for expert supervision. This last table is a typical quarantine table.

After conforming the hashtag values for SN_A, the dimension table $dimHashTag$ of the data mart can now be populated following the conciliation process described before in Section 4.3.1..



(a)

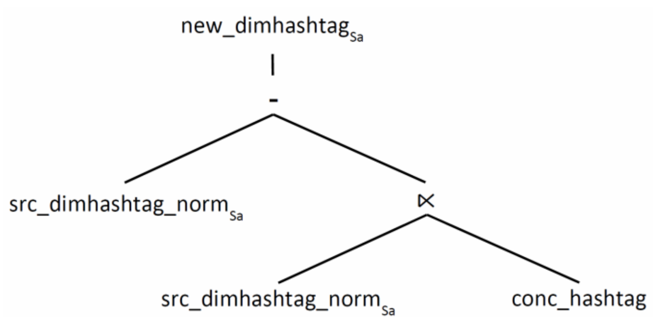


(b)

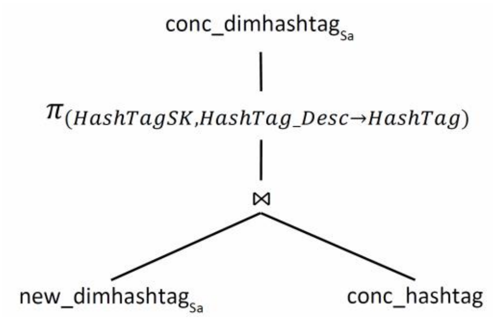
Figure 61 - (a) Conforming HashTag description (b) unmatched tuples

$$conc\_hashtag \ = \ \langle HashTagSK, HashTag\_Desc, HashTagID \rangle \tag{33}$$

The matching process will separate existing value from new values. Existing values are discarded and new values must be dealt with by expert supervision, mainly by updating the conciliation table and generating the correct Surrogate Key for those new values.



(a)



(b)

Figure 62 - (a) New hashtag data from SN$_A$ and (b) Surrogate key replacement for hashtag data

The integration of hashtag data from SN$_A$ into dimension $dimHashTag$ is achieved through Equation 34.

$$dimHashTag \leftarrow dimHashTag \cup conc\_dimhashtag_{S_a} \tag{34}$$

5.3.2 Populating the dimension table $dimHashTag$ with SN$_B$'s data

SN$_b$ has a different approach about how to store hashtag data. In this social network users associate predefined hashtags to their posts, instead of having the possibility to create their own. Nevertheless the steps required to populate correctly the table $dimHashtag$ involve to extract and conciliate OLTP's data.

$$src\_dimhashtag_{S'_b} = \langle HashTagID, Description \rangle \tag{35}$$

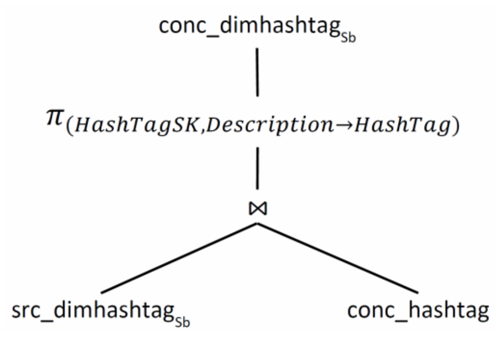
$$prev\_src\_dimhashtag_{S'_b} = \langle HashTagID, Description \rangle \tag{36}$$

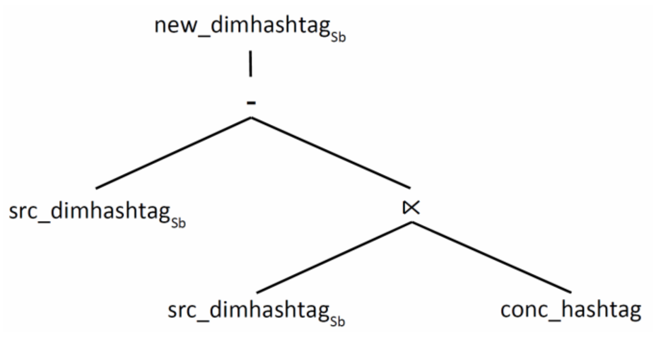
$$src\_dimhashtag_{S_b(HashTagID,Description)} \leftarrow src\_dimhashtag_{S'_b} - prev\_src\_dimhashtag_{S'_b} \tag{37}$$

$$prev\_src\_dimhashtag_{S'_b} \leftarrow src\_dimhashtag_{S'_b} \tag{38}$$

The table used to receive the conciliated data was already defined in Equation 33. The conciliation operations will be presented next. In Figure 63, the first RA tree substitutes a business key by its correspondent surrogate key, while the second RA tree determines new hashtag data coming from SN$_B$ with no correspondent surrogate key in the conciliation table. This tuples need expert intervention, i.e., assignment of the correct Surrogate Key and its update in the conciliation table.

Finally the integration of hashtag data from SN$_B$ into dimension $dimHashTag$ is achieved through Equation 39.





(a) (b)

Figure 63 – (a) Surrogate key replacement for hashtag data and (b) New hashtag data in need of surrogate key assignment

$$dimHashTag \leftarrow dimHashTag \cup conc\_dimhashtag_{S_b} \tag{39}$$

5.3.3 Populating $dimClient$ with SN$_A$ data

Populating the Customers/Client dimension from different sources usually leads to data quality and conciliation problems. Once those problems are addressed the main focus of attention turns to history preservation since data changes over time which can affect future analysis.

The extraction phase follows the patterns already defined and used in previous sections and presented next.

$$src\_dimclient_{S'_a} = \langle CustomerID, Title, FirstName, LastName, EmailAddress, Suffix, DOB, Phone, Country \rangle \tag{40}$$

$$prev\_src\_dimclient_{S'_a} = \langle CustomerID, Title, FirstName, LastName, EmailAddress, Suffix, DOB, Phone, Country \rangle \tag{41}$$

$$src\_dimclient_{S_a} \leftarrow src\_dimclient_{S'_a} - prev\_src\_dimclient_{S'_a} \tag{42}$$

$$prev\_src\_dimclient_{S'_a} \leftarrow src\_dimclient_{S'_a} \tag{43}$$

In this scenario, a normalization procedure must be applied to concatenate $FirstName$ and $LastName$ to fulfill the requirements of the $dimClient$ dimension and in addition project out unwanted attributes.

$$src\_dimclient\_norm_{S_a} \leftarrow \pi_{(CustomerID, FirstName+LastName \rightarrow ClientName, EmailAddress, DOB, Country)}(src\_dimclient_{S_a}) \tag{44}$$

After data quality issues are addressed, the conciliation phase begins by mapping the business key to the dimensions' surrogate key through the use of a conciliation table.

$$conc\_client = \langle ClientSK, CustomerID, ClientID \rangle \tag{45}$$

Again, and as already used in previous sections that deal with populating dimensions, the new tuples with no matching surrogate key must be dealt by expert supervision.

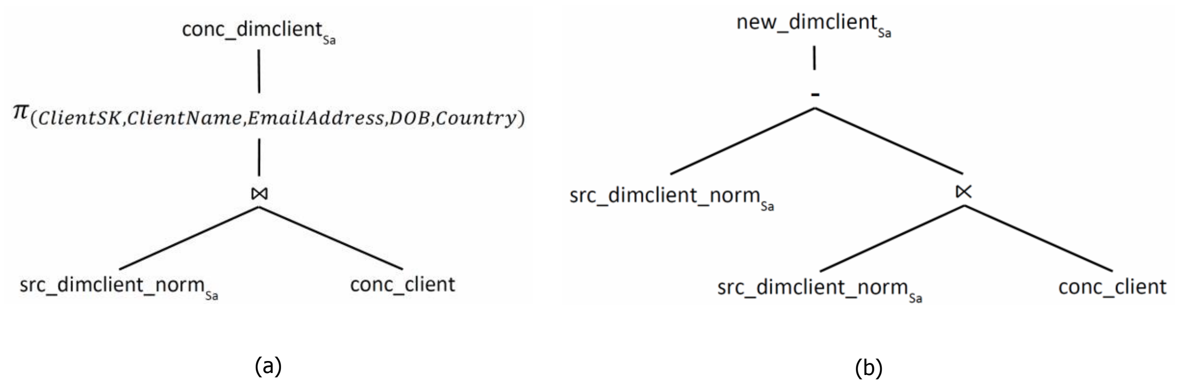

conc_dimclient$_{Sa}$
$\pi_{(ClientSK, ClientName, EmailAddress, DOB, Country)}$
$\bowtie$
src_dimclient_norm$_{Sa}$
conc_client
(a)
new_dimclient$_{Sa}$
-
src_dimclient_norm$_{Sa}$
$\ltimes$
src_dimclient_norm$_{Sa}$
conc_client
(b)

Figure 64– (a) Surrogate key replacement for client data and (b) New client data in need of surrogate key assignment

After conciliating data, history preservation is the next phase and is achieved through the use of a series of patterns that were previously defined in Section 4.5. The first step is the identification of expired data (Figure 65), then its transference from dimension $dimClient$ and to $dimClient\_hst$ (Figure 66).

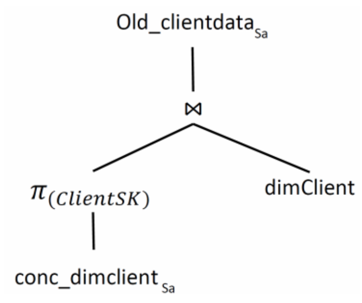
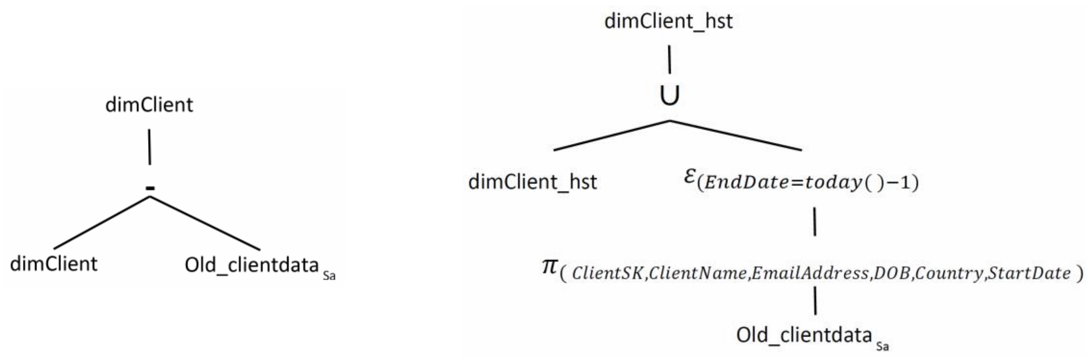


Figure 65 – Identification of changed and expired client data from SN$_A$



(a) (b)

Figure 66 – (a) Removing expired client data from $dimClient$ and (b) Adding removed client data to $dimClient\_hst$

After expired data is transferred, changed clients can be added into the dimension $dimClient$ (Figure 67) followed by new clients (Figure 68).

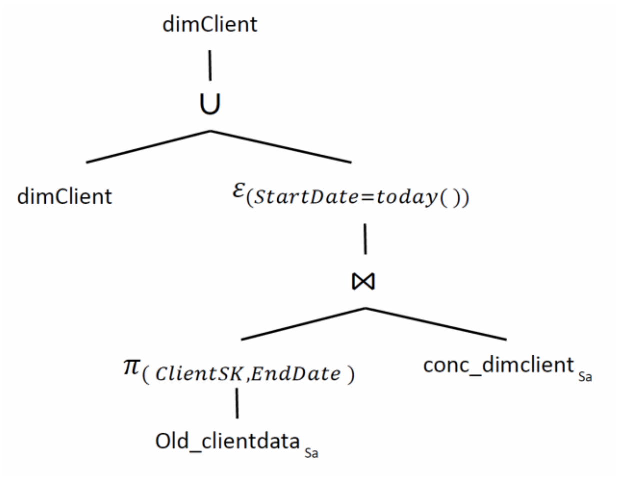

dimClient
∪
dimClient
$\varepsilon_{(StartDate=today())}$
⋈
$\pi_{(ClientSK,EndDate)}$
conc_dimclient Sa
Old_clientdata Sa

Figure 67 – Adding changed clients of SN_A to dimension $dimClient$

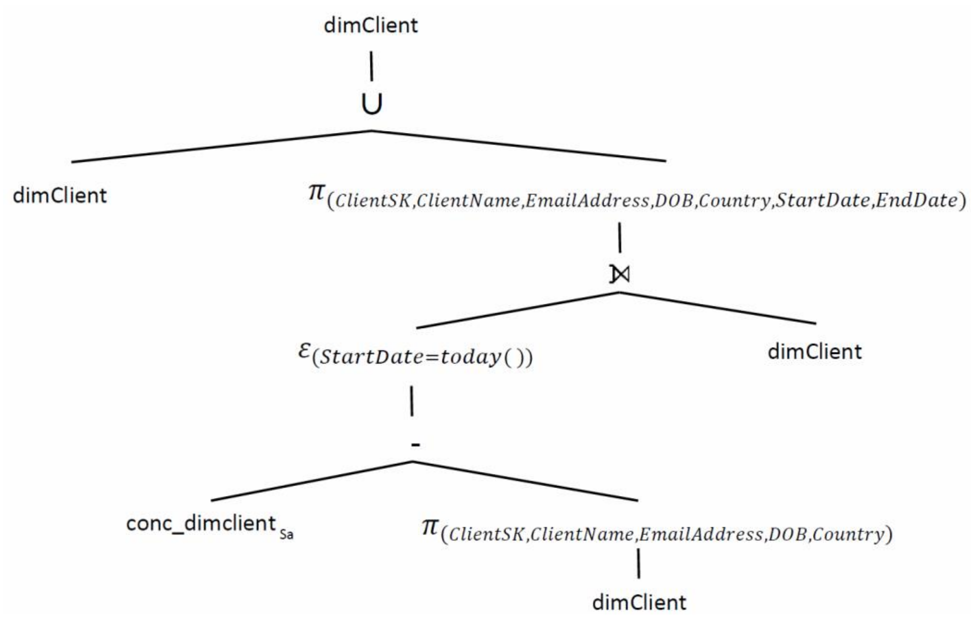

dimClient
∪
dimClient
$\pi_{(ClientSK,ClientName,EmailAddress,DOB,Country,StartDate,EndDate)}$
⋈
$\varepsilon_{(StartDate=today())}$
dimClient
-
conc_dimclient Sa
$\pi_{(ClientSK,ClientName,EmailAddress,DOB,Country)}$
dimClient

Figure 68 – Adding new clients of SN_A to dimension $dimClient$

Using these sets of operations we preserve the history of a client. The dimension $dimClient$ only maintains updated data.

5.3.4 Populating $dimClient$ with SN$_B$ data

The data about clients coming from SN$_B$ has a similar structure to the client's dimension in the data mart, therefore besides projecting out the unwanted attributes, the only task needed is the correct surrogate key attribution through the matching process that uses a conciliation table.

$$src\_dimclient_{S'_b} = \langle ClientID, Name, EmailAddress, DOB, Phone, Country \rangle \tag{46}$$

$$prev\_src\_dimclient_{S'_b} = \langle ClientID, Name, EmailAddress, DOB, Phone, Country \rangle \tag{47}$$

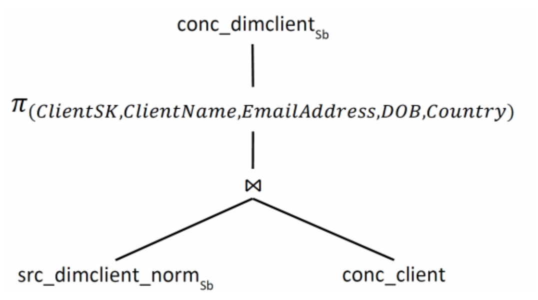
$$src\_dimclient_{S_b} \leftarrow src\_dimclient_{S'_b} - prev\_src\_dimclient_{S'_b} \tag{48}$$

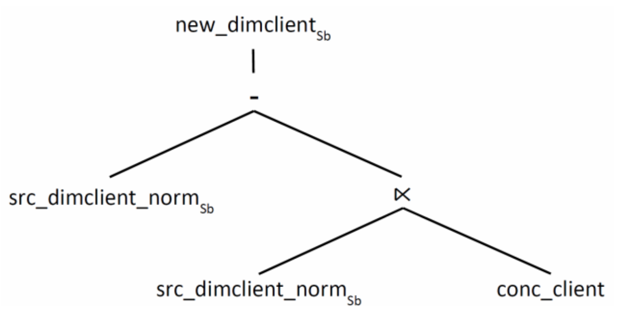
$$prev\_src\_dimclient_{S'_b} \leftarrow src\_dimclient_{S'_b} \tag{49}$$

After the execution of the extraction process, we proceed to filter out all the unwanted attributes renaming them accordingly to their corresponding attributes in dimension tables.

$$src\_dimclient\_norm_{S_b} \leftarrow \pi_{(ClientID, Name \rightarrow ClientName, EmailAddress, DOB, Country)}(src\_dimclient_{S_b}) \tag{50}$$

The conciliation table used to correctly assign the surrogate key was defined in Equation 45 and the conciliation process is almost identical to the one presented in the previous section.


conc_dimclient$_{Sb}$
$\pi_{(ClientSK, ClientName, EmailAddress, DOB, Country)}$
$\bowtie$
src_dimclient_norm$_{Sb}$
conc_client



(a) (b)

Figure 69 – (a) Surrogate Key replacement for client data and (b) New client data in need of surrogate key assignment

Since both sources have all the attributes necessary to populate the attributes of dimension $dimClient$ the history preservation phase is also almost identical to the previous section. If that was not the case only a few projections would have been different. Figure 70 through Figure 73 detail the operations of expiring changed clients data, and the addition of new clients to dimension $dimClient$.

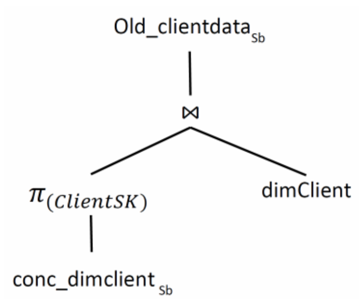
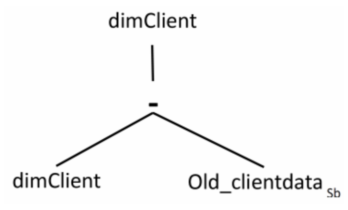
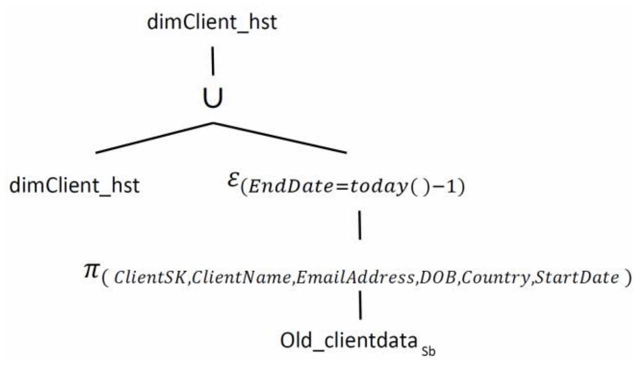


Figure 70 - Identification of changed and expired client data from SN_B




dimClient_hst
$\cup$
dimClient_hst
$\varepsilon_{(EndDate=today()-1)}$
$\pi_{(ClientSK,ClientName,EmailAddress,DOB,Country,StartDate)}$
$Old\_clientdata_{Sb}$

(a) (b)

Figure 71 – (a) Removing expired client data from $dimClient$ and (b) Adding removed client data to $dimClient\_hst$

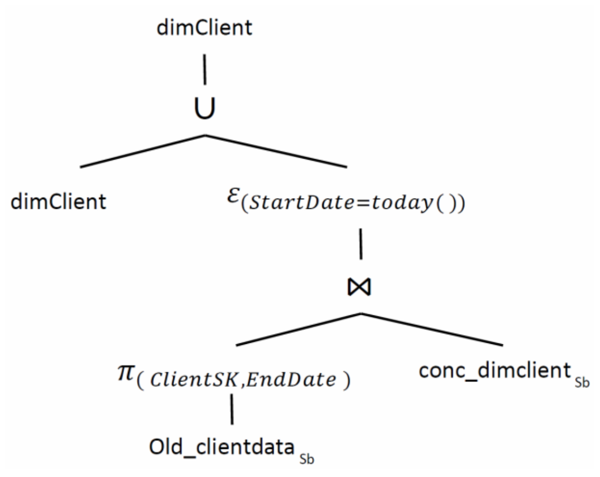


Figure 72 - Adding changed clients of SN_B to dimension $dimClient$

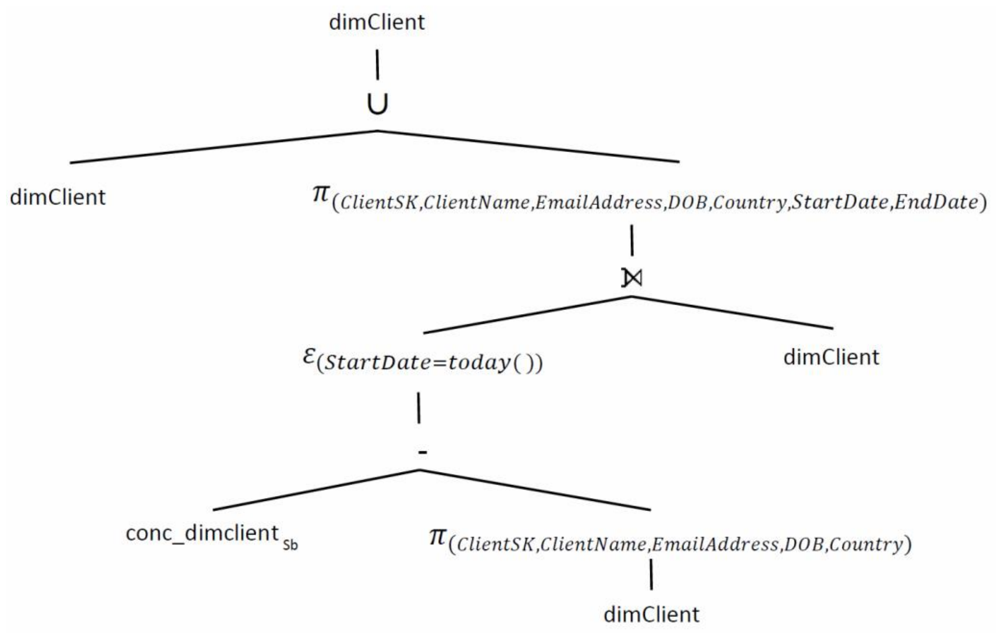

dimClient
∪
dimClient
$\pi_{(ClientSK,ClientName,EmailAddress,DOB,Country,StartDate,EndDate)}$
⋈
$\varepsilon_{(StartDate=today())}$
dimClient
-
conc_dimclient $_{Sb}$
$\pi_{(ClientSK,ClientName,EmailAddress,DOB,Country)}$
dimClient

Figure 73 - Adding new clients of SN_B to dimension $dimClient$

5.3.5 Populating $HashTagFacts$ with SN$_A$ data

Gathering hashtag usage from SN$_A$ implies filtering data from OLTP System tables, conform it and aggregate it according to previously defined goals. For this scenario the following RA expressions will accomplish the objective.

$$T_1 \leftarrow Login \bowtie \sigma_{(Date=today()-1)}(Post) \tag{51}$$

$$T_2 \leftarrow \pi_{(CustomerID,Date)}(T_1) \bowtie HashTag \tag{52}$$

Since clients could create their own hashtags freely in SN$_A$, the standardization process used for populating $dimHashTag$ (Equation 32) will also be used in the populating process of the fact table with data gathered from SN$_A$ (Equation 53).

$$HashTags_{S_a} \leftarrow \pi_{(CustomerID,Date,HashTag\_Desc1 \rightarrow HashTag\_Desc)}(T_2 \bowtie HashTag\_norm) \tag{53}$$

$$HashTags\_occur_{S_a} \leftarrow \gamma_{(CustomerID,Date,HashTagDesc,COUNT(HashTagDesc) \rightarrow Occurrences)}(HashTags_{S_a}) \tag{54}$$

$$HT\_Facts_{S_a} \leftarrow \varepsilon_{(SocialNetSK=1)}(HashTags\_occur_{S_a}) \tag{55}$$

Equation 54 aggregates the occurrences of hashtag data for each client and Equation 55 adds an attribute to properly identify the origin of the data (SN$_A$).

In order to load properly the fact data into the data mart, business keys will be substituted by surrogate keys, a process usually called Surrogate Key Pipelining (Figure 54). In this scenario the attributes $CustomerID$, $Date$ and $HashTagDesc$ need to be replaced by the corresponding surrogate keys.

Through the use of the client's conciliation table already defined in Equation 45, the RA tree presented in Figure 74 maps the $CustomerID$ from SN$_A$ to the correct surrogate key from the data mart.

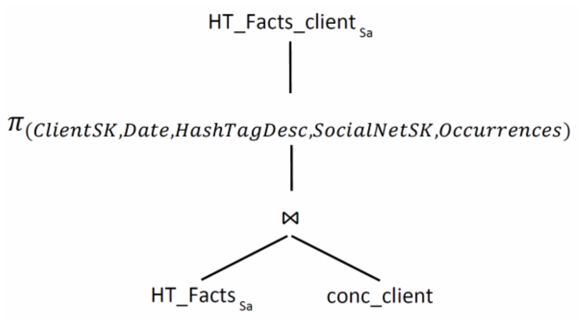


Figure 74 – Surrogate key assignment for customers from SN_A

Figure 75 represents the steps needed to identify unmatched customers in the assignment of a surrogate key and therefore in need of expert supervision.

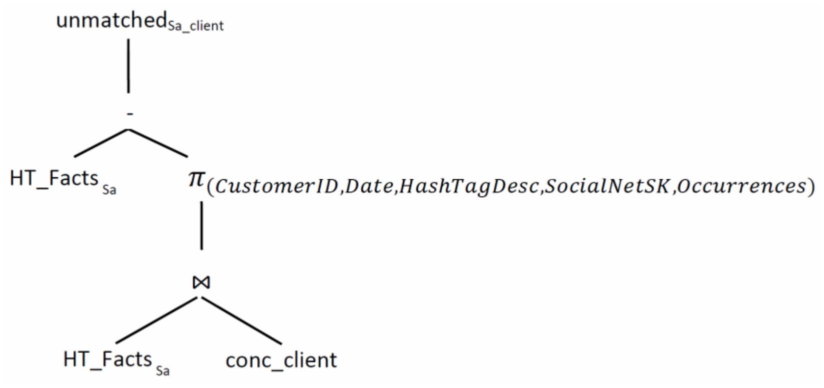

unmatched Sa_client
-
HT_Facts Sa
$\pi_{(CustomerID,Date,HashTagDesc,SocialNetSK,Occurrences)}$
⋈
HT_Facts Sa
conc_client

Figure 75 – Unmatched customers surrogate key assignment

The next surrogate key assignment is the transformation of the attribute $Date$ into the surrogate key used for dimension $dimTime$ and for this the conciliation table used is a projection of $dimTime$ itself (Equation 56). In this case there is no need to identify unmatched surrogate key assignments since all the dates extracted from the source have a correspondence in the conciliation table.

$$conc\_date \leftarrow \pi_{(DateSK,Date)}(dimTime) \tag{56}$$

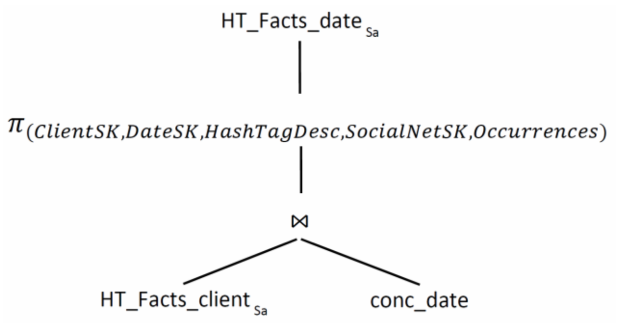

HT_Facts_date Sa
$\pi_{(ClientSK,DateSK,HashTagDesc,SocialNetSK,Occurrences)}$
⋈
HT_Facts_client Sa
conc_date

Figure 76 - Surrogate key assignment for dates from SN_A

To finalize the Surrogate Key pipeline process, the last business key (identifying the hashtags used) is mapped into the correct surrogate key in Figure 77.

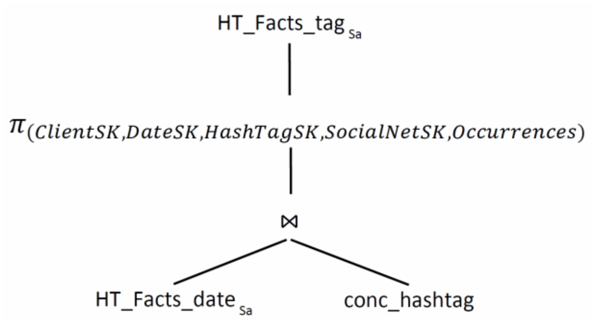


Figure 77 - Surrogate key assignment for hashtags from SN_A

The following RA tree (Figure 78) identifies the unmatched tuples in which there was no correspondence for $HashTagDesc$ when joined with the conciliation table and therefore in need of expert supervision.

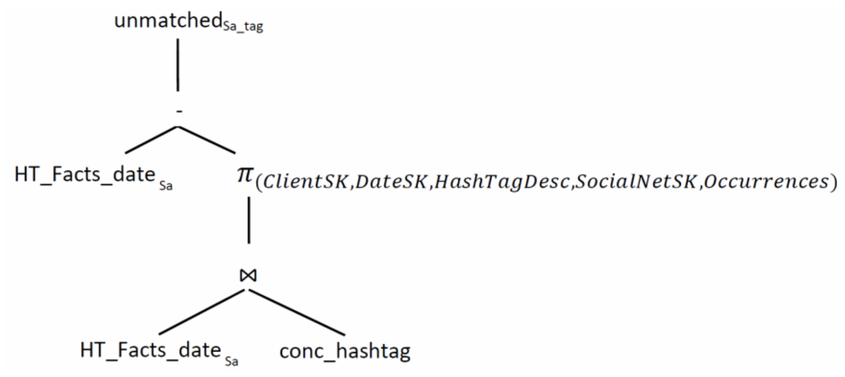


Figure 78 - Unmatched hashtag surrogate key assignment

The populating process of the fact table with data gathered from SN_A OLTP System is finalized after the SKP process and the addition of the result into the fact table (Equation 57).

$$HashTagFacts \leftarrow HashTagFacts \cup HT\_Facts\_tag_{S_a} \tag{57}$$

5.3.6 Populating $HashTagFacts$ with SN_B data

Although the OLTP diagram from SN_B appears to be more complicated than SN_A diagram, the fact that hashtags are predefined facilitates the process of gathering facts.

Equation 58 gathers hashtags used in posted messages and Equation 59 gathers hashtags used in comments, then all data is grouped to aggregate the number of occurrences a certain hashtag was used that day for each client. Finally, and before the surrogate key pipeline process starts, a new attribute is added to identify the source of the data (SN_B).

$$T_1 \leftarrow \pi_{(ClientID,Date,HashTagID)}(\,\sigma_{(Date=today(\,)-1)}(Message) \bowtie Mess\_Tag) \tag{58}$$

$$T_2 \leftarrow \pi_{(ClientID,Date,HashTagID)}(\,\sigma_{(Date=today(\,)-1)}(Comments) \bowtie Comm\_Tag) \tag{59}$$

$$HashTags_{S_b} \leftarrow T_1 \cup T_2 \tag{60}$$

$$HashTags\_occur_{S_b} \leftarrow \gamma_{(ClientID,Date,HashTagID,COUNT(HashTagID)\rightarrow Occurrences)}(HashTags_{S_b}) \tag{61}$$

$$HT\_Facts_{S_b} \leftarrow \varepsilon_{(SocialNetSK=2)}(HashTags\_occur_{S_b}) \tag{62}$$

The surrogate key pipeline process substitutes the business keys used in SN_B OLTP for the surrogate keys used in the data mart. Similar to the tasks presented in the previous section, the following RA tress represent the mapping procedures for attributes $ClientID$ (Figure 79), $Date$ (Figure 81) and $HashTagID$ (Figure 82).

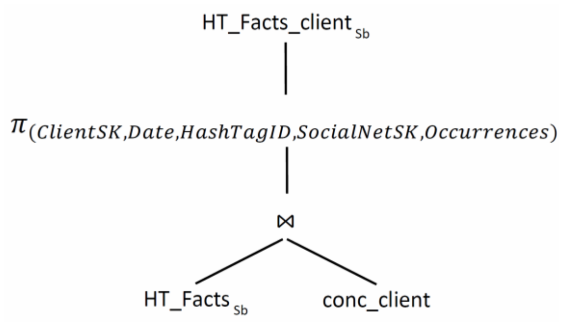


Figure 79 – Surrogate key assignment for clients from SN_B

Figure 80 identifies the unmatched clients in the mapping process subject to expert revision.

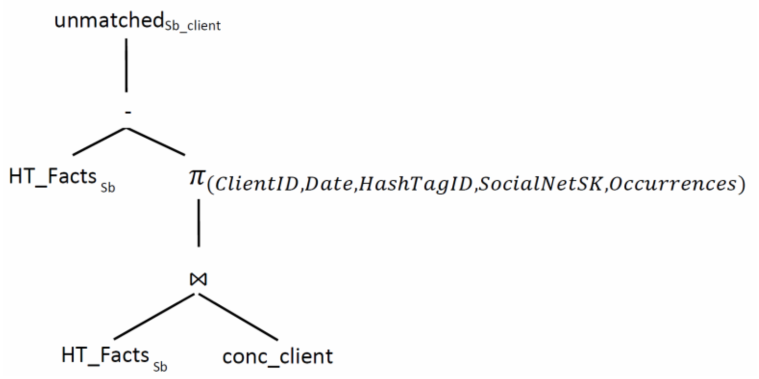


Figure 80 – Unmatched clients surrogate key assignment

In the RA tree presented in Figure 81, the date conciliation table used was previously defined in Equation 56.

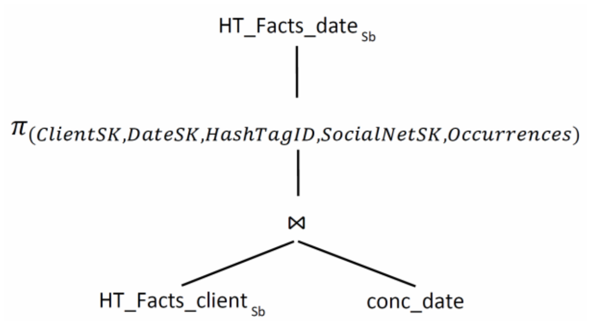


Figure 81 – Surrogate key assignment for dates from SN_B

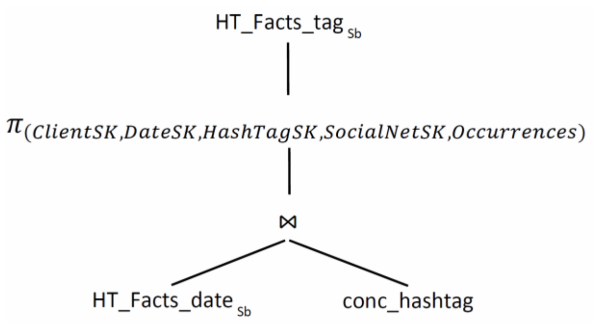


Figure 82 – Surrogate key assignment for hashtags from SN_B

Figure 83 is almost identical to Figure 78 and identifies tuples in need of supervision due to absence of correspondence in the hashtag conciliation table.

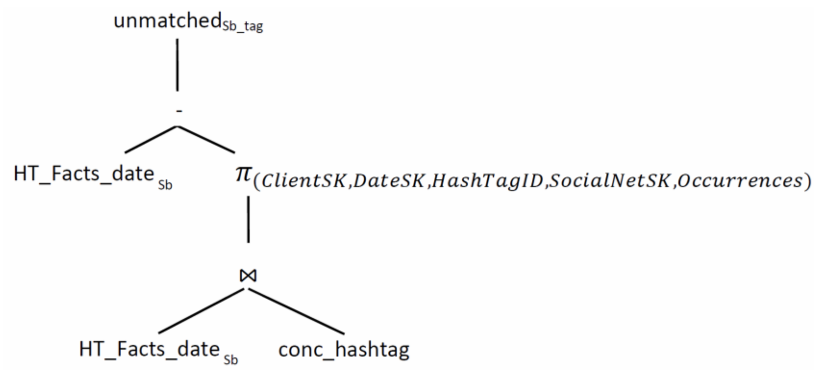


Figure 83 – Unmatched hashtag surrogate key assignment

The populating process of the fact table with data gathered from SN_B OLTP System is finalized after the SKP process and the addition of the result into the fact table (Equation 63).

$$HashTagFacts \leftarrow HashTagFacts \cup HT\_Facts\_tag_{S_b} \tag{63}$$

5.4 A Global View of the Populating Process

While the previous sections of this chapter have presented the set of steps needed to populate a data mart from two heterogenic OLTP sources (as an example of a Real ETL scenario), a generalized view of all the ETL is also significant to evaluate the modeling proposal presented. To represent and model this global view of the populating process we used BPMN[45]. The representation of the all modeling process in RA trees was also possible but the diagram would not be easy to read and interpret for the stakeholders. BPMN notation allows us to create a few more layers of abstraction that facilitates the interpretation of the activity flows. The idea was to encapsulate the main tasks that can be expanded to its sub processes that in turn contain the instantiation of the RA patterns proposed. In Figure 84 a higher level of the ETL process is presented with the main tasks, like for instance the Populating process of $dimHashTag$ with data from SN_A.

Following the characteristics of BPMN, in this diagram one pool was used, with three lanes, corresponding to the population of each dimension and fact table. Each collapsed process is further expanded for better understanding of the tasks involved.

Analyzing the collapsed process referent to Populating $dimHashTag$ with data from SN_A (Figure 85), the basic steps are the CDC, DQE and DCI tasks, as would normally be in the populating process of a typical dimension.

[4] http://www.bpmn-tool.com/en/tutorial/
[5] http://www.omg.org/news/meetings/workshops/SOA-HC/presentations-2011/14_MT-2_Brookshier.pdf

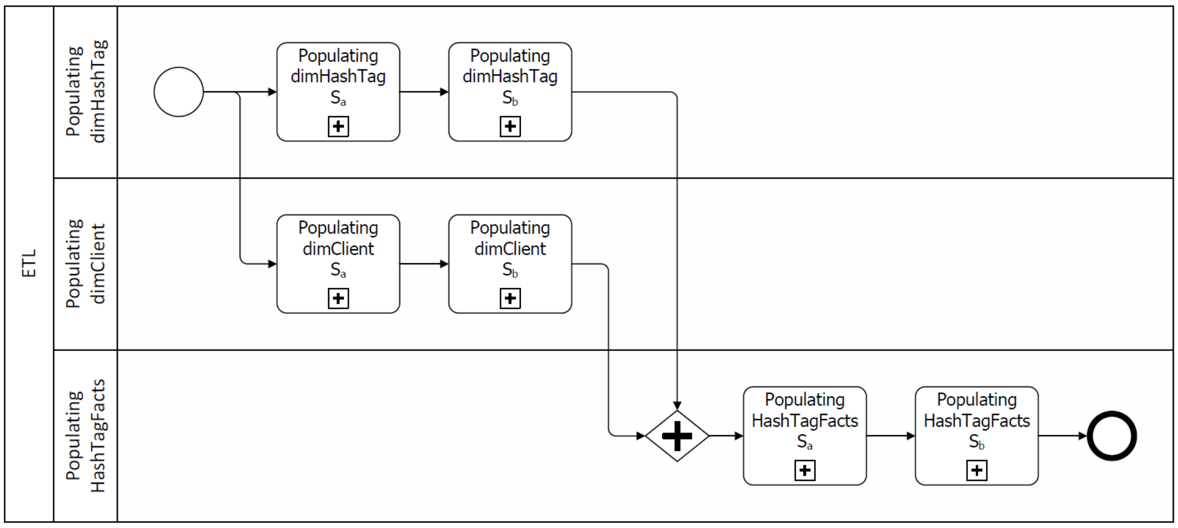


Figure 84 – BPMN high level model of the ETL for the scenario used

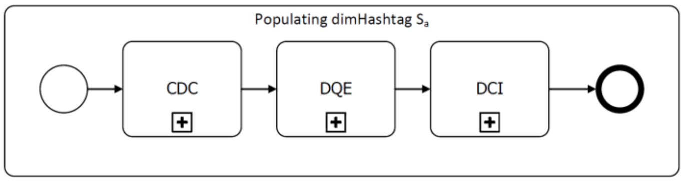


Figure 85 – Expanded process for Populating $dimHashTag$ with data from SN_A

Digging further into each collapsed process, it is now visible the sequence of operations defined in previous sections of this chapter, i.e., the RA expressions and trees used to process data from OLTP systems into the data mart (Figure 86, Figure 87 and Figure 88).

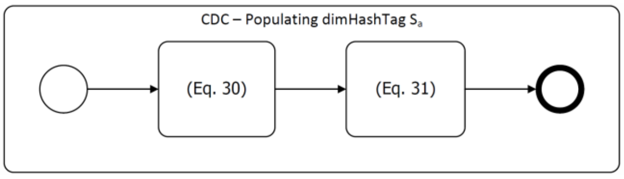


Figure 86 – Expanded process for the CDC used for populating $dimHashTag$ with data from SN_A

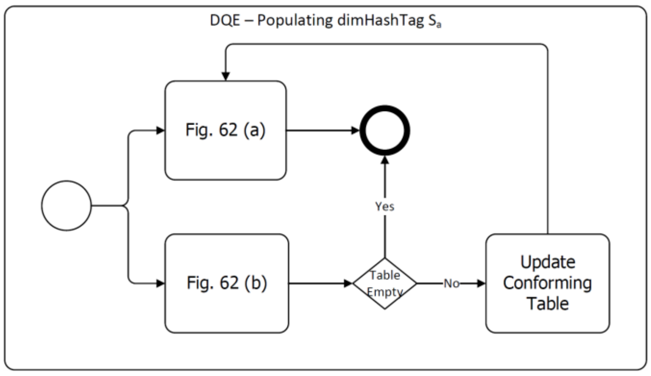


Figure 87 - Expanded process for the DQE used for populating $dimHashTag$ with data from SN$_A$

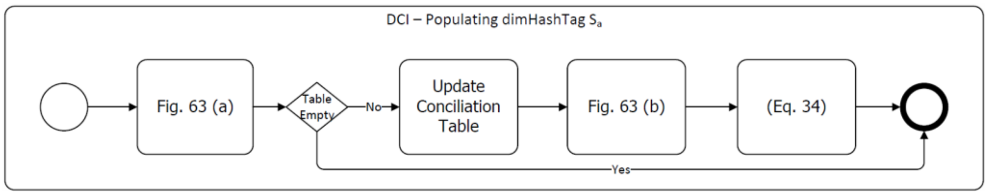


Figure 88 - Expanded process for the DCI used for populating $dimHashTag$ with data from SN$_A$

The other collapsed processes referent to the populating procedure of the dimensions are very similar to the one presented. In fact only a few particularities exist as it can be shown in the full diagram in Appendix.

Loading a fact table is nevertheless different, data has to be extracted and transformed from OLTP sources and submitted to the SKP process before loading it into the data mart as it can be seen in Figure 89.

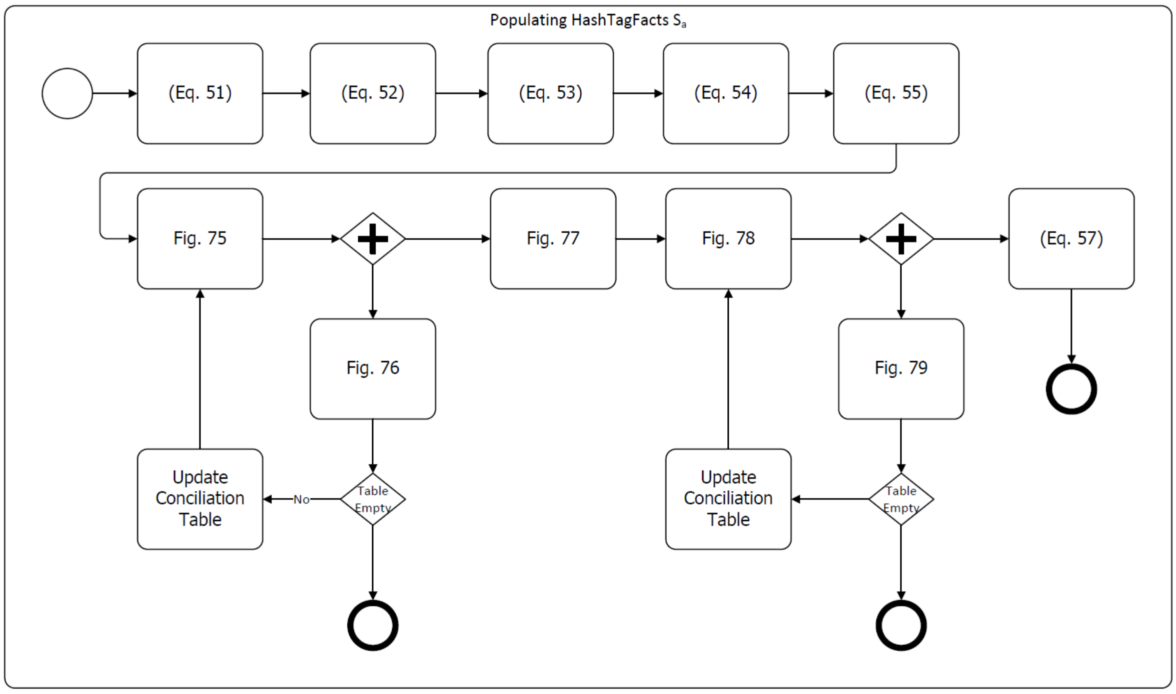


Figure 89 - Populating $HashTagFacts$ with data from the SN_A – expanded process.

The complete diagram of this Real ETL System Specification is presented in Figure 90.

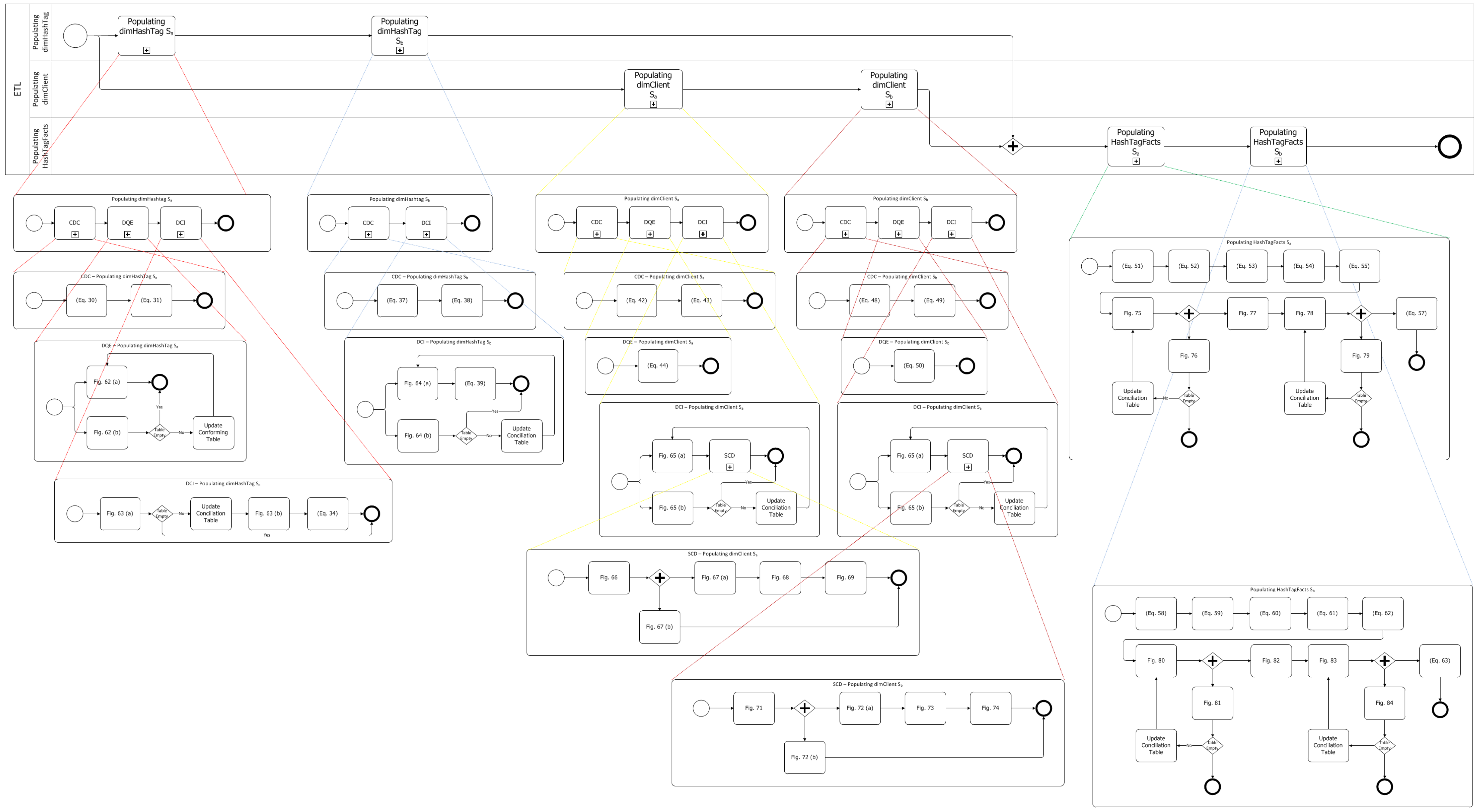


Figure 90 – The complete ETL model diagram for the data mart "HashTag

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In times of crisis, those that are best prepared tend to overcome, prosper and endure. This philosophy is essential for the survival of organizations and people. The global crisis that has affected the world in the last years has made some profound changes in market behavior, people choices with particular emphasis in financial and political choices, altering dramatically their consumption and priorities. These changes have a high impact on the normal business of organizations and enterprises, which must adapt themselves in order to survive and also search for opportunities to improve or diversify.

Detecting consumer changes, improvement opportunities or simply information to make a decision is not an easy task. This task would also have a higher rate of success if it could be made based on factual data instead of intuition. Therefore, having data to support management in their decision-making processes is essential for an organization that has as a primary goal growth and prosperity.

Ensuring the existence of factual data to support business decisions is no longer an issue, on the contrary, we now have access to an ever increasing amount of data from disparate sources of information – e.g. OLTP systems, personal working files, or Internet information. The problem now is to select which data is important, how to gather it, and above all how to store it in conjunction with all the data gathered from other sources. This centralized and unified repository, a data warehouse (Inmon and Hackathorn, 1994), will serve as an essential platform for querying and reporting that would in turn support management decision making processes. Such a repository and all the components necessary to load and maintain it is what we normally address as a DWS. One of its most critical components is the ETL component, which is responsible for extracting data from information sources, cleaning, transforming and conforming it and, finally, loading it into the data warehouse. Due to its complexity, this component normally takes the majority of the effort in its design and implementation.

ETL design has gained increased importance as means to alleviate the risk of managing the ETL process. Researchers and industry practitioners have proposed several approaches to the design and management of this critical DWS component. Inmon (1994, 2001, 2005) and Kimball (1998, 2004) proposed *ad-hoc* approaches that lack formality not contributing to a recognized and viable solution. In turn, Vassiliadis et al. (2002a, 2002b, 2002c, 2003, 2005, 2005c, 2007, 2008) proposed a three level architecture similar to the conventional database design composed of conceptual, logical and physical models, presenting a novel graphical notation for the conceptual model. In our view, this generates complex and rather difficult to read diagrams. The logical model is also based on a new graphical notation that specifies the workflow of tasks needed to populate a DW. The same problem applies to this approach, i.e., the diagrams produced are extensive and difficult for interpreting. Trujillo et al. (2003, 2004) proposed an extension to the UML notation to support the design of ETL tasks. The advantage of using a standard notation is rapidly undermined by the specificities of the ETL process, since several common tasks are not clarified in the extension proposed. UML activity diagrams (Muñoz et al., 2008) are used to minimize this problem but the consequence is a rather complex diagram even for the simplest tasks.

In the view of Akkaoui and Zimanyi (2009), the ETL process should be modeled has a business process instead of a database model as previous researchers have attempted to do. Therefore, they propose an extension to BPMN to deal with the specificities of common ETL tasks. This proposal has the advantage of using a standard modeling language that is easily translated into a programming language like BPEL. In our point of view, this approach generates diagrams that are easy to read and interpret but gives too much freedom to the designer due to some lack of rigor, i.e., a common task may be designed with different mechanisms depending on the designer.

Although several proposals have been made for the conceptual and logical design of the ETL process, there still has not emerged a standard notation commonly accepted and used in the development of the ETL component of a DWS. As a consequence, the physical model of the ETL process varies, with each ETL commercial tool proposing a proprietary design notation, which in turn does not help in the task of selecting a commercial tool due to the financial commitment enterprises need to make. We believe that a standard, widespread ETL modeling notation is still lacking and its existence would facilitate the task of selecting a commercial tool to implement the ETL component. The main advantage of such standard notation would be the separation of the design and implementation, contributing to the development and improvement of new modeling tools.

An ETL process is certainly complex because it comprises disparate tasks such as extraction and transformation of data to be loaded into a data warehouse. The transformation task is not a simple attribute mapping, data type conversion or attribute renaming one; it contains undoubtedly very important tasks of cleaning and conforming data to provide the data warehouse with correct data that faithfully reflects what actually happened in the operational systems. The data warehouse must use clean and conformed data to better support management decision-making processes with effectiveness. However due to the nature of many information systems that are supporting business activities – usually conventional OLTP systems -, data being gathered might not always be accurate and correctly stored. Therefore, the cleansing tasks of the ETL process must try to identify the data problems and inconsistencies, performing transformations in order to guarantee minimum data quality and prepare data for loading into the data warehouse. Data quality issues are increasingly more pertinent when dealing with multiple and heterogeneous sources, augmenting the risk of failure or, even worse, the risk of misrepresenting information in a system conceived for supporting management decision-making assessments. Another issue closely linked to heterogeneous sources, is the different representation of common concepts. To deal with these issues, conciliation tasks must be prepared to transform source data into a conformed and unified view of the business. And finally in order to correctly load cleaned and conformed data into the data warehouse, the surrogate key pipeline process must correctly attribute a surrogate key to each business key. These processes, CDC, DQE, DCI and SKP are commonly implemented in ETL commercial tools, but are poorly supported in proposed modeling methodologies and notations, especially in the logical design. Some of the existing conceptual modeling notations have diagrammatic representations of these processes but there is no logical implementation for them. This is a major handicap that does not contribute to reducing the gap between conceptual design and physical implementation of the ETL process (Oliveira et al., 2013).

A less common approach to the design of the ETL process, studied in this thesis, is through the use of RA operations and trees. The RA representation of the tasks in trees facilitates the understanding of how the flow of operations are organized, giving a clear picture for possible optimizations of the entire ETL process. This approach is well supported for uncommon DWS environments, where a database management system normally does not exist, mainly because the flow of operations is well defined. The RA operators are known and easily implemented in any execution language, which in turn will undoubtedly contribute to a higher degree of flexibility and usability.

In this thesis we proposed the use of patterns of RA operations to represent the most common ETL tasks, starting with CDC task that extracts source data and determines changes in the extracted data using mainly a RA subtract operator. DQE tasks are varied, but the primary concern is to represent correctly all data extracted. Several transformations might occur in this phase, mainly data corrections, or data normalizations and standardizations. These tasks were accomplished mainly through the use of user functions applied to attributes, or auxiliary tables that mapped incorrect to correct values (Santos and Belo, 2013). Conciliating Data from heterogeneous sources is a challenge mainly because of all the human interaction needed to maintain the conciliation tables up-to-date. Conciliation tables map business keys from source data to correspondent surrogate keys, which is of extremely importance when dealing with multiple sources due to the different possible representations common concepts can have in each source system. Besides it, the task itself is mainly achieved through join and subtract operations (Santos and Belo, 2014). Integrating conciliated data in dimensions depends on the selected strategy for history preservation (SCDs) (Santos and Belo, 2011b, 2011c), nevertheless we advocate that history preservation should be done through the use of an auxiliary table to ease the computational burden of the maintenance of the dimension (Santos and Belo, 2011a). Therefore this task is mainly achieved through an expiration procedure that transfers old and changed data to an historical table tagging it as expired, and adds the new updated records to the SCD. After extracting, cleaning, conforming and conciliating data, the loading process of the data warehouse must take into account that primary keys of the dimensions and fact tables are probably different from the OLTP systems where data was gathered. The process of replacing business keys by surrogate keys is normally called SKP and is achieved through the use of a set of tables that map the business key into the correct surrogate key through join and project operations.

Even though all the most common ETL tasks were modelled using RA operators, there are still a few common tasks in an ETL System that are beyond this approach. In a conceptual view, whenever a task fails it can be modelled to restart, or it can trigger a message event. These tasks or events are not modelled in RA operators, since the RA language is oriented to data manipulation and not to control flows of operations. Another particularity of the RA language is the fact that it does not take into account execution optimizations like the ones present in a DBMS. The concept of indexes and their impact on bulk loading or in referential integrity control is not present in a higher level language like RA. All other optimizations that commercial DBMS implement on their database engines are also not applied in RA language.

Nevertheless, the proposal merits and demerits were evaluated through the use of real ETL system specification that included two heterogeneous sources and one data mart. The patterns for the common ETL tasks proposed in chapter 4 were instantiated to the testing scenario and we can observe that common tasks once modeled are applied with no effort or ambiguity. To present the entire model in a user-friendly diagram (high level diagram), BPMN language was used to better represent the execution flows. In fact, BPMN was only used to be reader friendly and no special data warehouse artifacts or tasks were used in that notation. As a proof of concept we also tested successfully the modeling approach over an uncommon technological infrastructure composed of a few computers, a grid architecture, that were prepared to execute the ETL process without the need of a DBMS provided the nodes received the instructions and data stored in a XML format (Santos et al., 2011, 2012, 2014).

6.2 Future Work

Although the objectives planned for this work have been attained, further research and development can be achieved in this area of expertise by analyzing current trends in the Data Warehousing community with particular emphasis in the design of the ETL component. A new area of research called Semantic ETL, that reduces the initial work of discovering and identifying source attributes and their correct mappings to data warehouse attributes, is of significant value to minimize the initial effort of the ETL design phase.

Another area of interest that might influence the development of future DWS is the recent evolution of new DBMS, namely NoSQL and NewSQL database engines, which also support non-relational databases like hierarchical, graph and object-oriented databases. Data warehouses are commonly supported by the dimension model, which bases itself in the relational model, therefore other non-relational DBMS sources will augment the complexity of the ETL system, especially in the extraction and transformation processes therefore in the ETL design phase.

All modeling proposals presented in this work can also be improved through the use of the RA operations proposed by the scientific community – e.g. Carreira et al. (2007) or Kis and Buza (2009). In addition, other RA operators could be developed to provide a formal support to current DBMS operations, like the existence of auto-incremental attributes (also known as identity attribute), or hash-based attributes, just to name a few. The latter proposal would be very helpful in the modeling phase of a SKP process and also in the definition of surrogate keys that are frequently present in data conciliation tables that support conventional DCI processes.

Another line of research worth to be explored is the integration of the proposal we made on this thesis with other conceptual modeling proposals (Oliveira and Belo, 2013a, 2013b, 2014, Belo et al., 2014), as a continuous effort to minimize the gap between the conceptual and physical design of an ETL System. Just to end, it is important to refer we believe that the formalization of each ETL pattern with a regular standard language will improve the robustness of any ETL design process facilitating its conversion for a conventional ETL implementation and execution platform.

Bibliography

AKKAOUI, Z. E. & ZIMANYI, E. 2009. Defining ETL worfklows using BPMN and BPEL. *Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP.* Hong Kong, China: ACM.

ASTRAHAN, M. M., BLASGEN, M. W., CHAMBERLIN, D. D., ESWARAN, K. P., GRAY, J. N., GRIFFITHS, P. P., KING, W. F., LORIE, R. A., MCJONES, P. R., MEHL, J. W., PUTZOLU, G. R., TRAIGER, I. L., WADE, B. W. & WATSON, V. 1976. System R: relational approach to database management. *ACM Transactions on Database Systems,* 1**,** 97-137.

BARALIS, E. & WIDOM, J. An Algebraic Approach to Rule Analysis in Expert Database Systems. Proceedings of the 20th International Conference on Very Large Data Bases, 1994. 673002: Morgan Kaufmann Publishers Inc., 475 -- 486.

BELO, O., CUZZOCREA, A. & OLIVEIRA, B. Modeling and Supporting ETL Processes via a Pattern-Oriented, Task-Reusable Framework. Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on, 2014. IEEE, 960-966.

BOUZEGHOUB, M., FABRET, F. & MATULOVIC-BROQUÉ, M. Modeling Data Warehouse Refreshment Process as a Workflow Application. Proceedings of the International Workshop on Design and Management of Data Warehouses 1999 Heidelberg, Germany.

CARREIRA, P., GALHARDAS, H., LOPES, A. & PEREIRA, J. 2007. One-to-many data transformations through data mappers. *Data & Knowledge Engineering,* 62**,** 483-503.

CODD, E. F. 1970. A Relational Model of Data for Large Shared Data Banks. *Commununications of the ACM,* 13**,** 377-387.

CODD, E. F. 1979. Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems,* 4**,** 397-434.

COSTA, A. M. P. M. 2006. *A Gestão da Qualidade dos Dados em Ambientes de Data Warehousing na Prossecução da Excelência da Informação.* Mestrado, Universidade do Minho.

DAYAL, U., CASTELLANOS, M., SIMITSIS, A. & WILKINSON, K. 2009. Data integration flows for business intelligence. *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology.* Saint Petersburg, Russia: ACM.

DAYAL, U., GOODMAN, N. & KATZ, R. H. 1982. An extended relational algebra with control over duplicate elimination. *Proceedings of the 1st ACM SIGACT-SIGMOD symposium on Principles of database systems.* Los Angeles, California: ACM.

DAYAL, U., WILKINSON, K., SIMITSIS, A. & CASTELLANOS, M. 2010. Business Processes Meet Operational Business Intelligence.

DEMIYA, T., YOSHIHISA, T. & KANAZAWA, M. 2008. Compact grid : a grid computing system using low resource compact computers. *Int. J. Commun. Netw. Distrib. Syst.,* 1**,** 17.

FOSTER, I., KESSELMAN, C. & TUECKE, S. 2001. The Anatomy of the Grid : Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications,* 15**,** 200-222.

GLOOR, P. A., KRAUSS, J., NANN, S., FISCHBACH, K. & SCHODER, D. Web Science 2.0: Identifying Trends through Semantic Social Network Analysis. Computational Science and Engineering, 2009. CSE '09. International Conference on, 29-31 Aug. 2009 2009. 215-222.

GREFEN, P. W. P. J. & DE BY, R. A. A multi-set extended relational algebra: a formal approach to a practical issue. Data Engineering, 1994. Proceedings.10th International Conference, 14-18 Feb 1994 1994. 80-88.

GRIFFIN, D. A. J., GRIFFITHS, P. J. L., JUDGES, S. H., CAMPBELL, N. A. & ROBERTS, M. D. 2005. *Method of Managing Slowly Changing Dimensions*. United States of America patent application.

HELD, G. D., STONEBRAKER, M. R. & WONG, E. 1975. INGRES: a relational data base system. *Proceedings of the National Computer Conference and Exposition.* Anaheim, California: ACM.

HERNÁNDEZ, M. A. & STOLFO, S. J. 1998. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Mining and Knowledge Discovery,* 2**,** 9-37.

INMON, W. H. 2005. *Building the Data Warehouse*, Wiley Publishing, Inc.

INMON, W. H. & HACKATHORN, R. D. 1994. *Using the Data Warehouse*, Wiley-QED Publishing.

INMON, W. H., IMHOFF, C. & SOUSA, R. 2001. *Corporate Information Factory*, John Wiley & Sons, Inc.

INSTAGRAM. 2015. *Instagram Top Hashtags* [Online]. Available: http://top-hashtags.com/instagram/ [Accessed 26 June 2015].

KEDAD, Z. & MÉTAIS, E. 1999. Dealing with Semantic Heterogeneity During Data Integration. *In:* AKOKA, J., BOUZEGHOUB, M., COMYN-WATTIAU, I. & MÉTAIS, E. (eds.) *Conceptual Modeling — ER '99.* Springer Berlin Heidelberg.

KIMBALL, R. & CASERTA, J. 2004. *The Data Warehouse ETL Toolkit - Pratical Techniques for Extracting, Cleaning, Conforming, and Delivering Data,* , Wiley Publishing, Inc.

KIMBALL, R., REEVES, L., ROSS, M. & THORNTHWAITE, W. 1998. *The Data Warehouse Lifecycle Toolkit - Expert Methods for Designing, Developing, and Deploying Data Warehouses,* , John Wiley & Sons, Inc.

KIMBALL, R. & ROSS, M. 2002. *The Data Warehouse Toolkit - The Complete Guide to Dimensional Modeling,* , John Wiley & Sons.

KIMBALL, R., ROSS, M., THORNTHWAITE, W., MUNDY, J. & BECKER, B. 2008. *The Data Warehouse Lifecycle Toolkit - Practical Techniques for Building Data Warehouse and Business Intelligence Systems,* , Wiley Publishing, Inc.

KIS, P. B. & BUZA, A. Expansion of the Relational Algebra. 7th International Symposium on Intelligent Systems and Informatics, 2009. SISY '09, 25-26 Sept. 2009 2009 Subotica, Serbia. 127 -- 130.

KLUG, A. 1982. Equivalence of Relational Algebra and Relational Calculus Query Languages Having Aggregate Functions. *Journal of the ACM,* 29**,** 699-717.

KUMAR, R., NOVAK, J. & TOMKINS, A. 2010. Structure and Evolution of Online Social Networks. *In:* YU, P. S., HAN, J. & FALOUTSOS, C. (eds.) *Link Mining: Models, Algorithms, and Applications.* Springer New York.

LEE, M. L., LU, H., LING, T. W. & KO, Y. T. Cleansing Data for Mining and Warehousing. 10th International Conference on Database and Expert Systems Applications, August 1999 1999 Florence. 751-760.

LENZERINI, M. 2002. Data integration: a theoretical perspective. *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* Madison, Wisconsin: ACM.

LUJÁN-MORA, S., VASSILIADIS, P. & TRUJILLO, J. 2004. Data Mapping Diagrams for Data Warehouse Design with UML. *In:* ATZENI, P., CHU, W., LU, H., ZHOU, S. & LING, T.-W. (eds.) *Conceptual Modeling – ER 2004.* Springer Berlin / Heidelberg.

MANH NGUYEN, T., MIN TJOA, A., NEMEC, J. & WINDISCH, M. 2007. An approach towards an event-fed solution for slowly changing dimensions in data warehouses with a detailed case study. *Data & Knowledge Engineering,* 63**,** 26-43.

MANNINO, M. V. & WALTER, Z. 2006. A framework for data warehouse refresh policies. *Decision Support Systems,* 42**,** 121-143.

MISLOVE, A., MARCON, M., GUMMADI, K. P., DRUSCHEL, P. & BHATTACHARJEE, B. 2007. Measurement and Analysis of Online Social Networks. *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement.* ACM.

MUÑOZ, L., MAZÓN, J.-N., PARDILLO, J. & TRUJILLO, J. 2008. Modelling ETL Processes of Data Warehouses with UML Activity Diagrams. *In:* MEERSMAN, R., TARI, Z. & HERRERO, P. (eds.) *OTM 2008 Workshops.* Springer Berlin / Heidelberg.

NAQVI, S. & TSUR, S. 1989. *A logical language for data and knowledge bases*, Computer Science Press, Inc.

OLIVEIRA, B. & BELO, O. ETL process modeling using BPMN meta-models. Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on, 2013a. IEEE, 1-6.

OLIVEIRA, B. & BELO, O. Using REO on ETL conceptual modelling: a first approach. Proceedings of the sixteenth international workshop on Data warehousing and OLAP, 2013b. ACM, 55-60.

OLIVEIRA, B. & BELO, O. On the conceptualization of ETL patterns: a Reo approach. Proceedings of the 18th International Database Engineering & Applications Symposium, 2014. ACM, 348-351.

OLIVEIRA, B., SANTOS, V. & BELO, O. 2013. Pattern-based ETL conceptual modelling. *Model and Data Engineering.* Springer.

OZSOYOGLU, G., OZSOYOGLU, Z. M. & MATOS, V. M. 1987. Extending relational algebra and relational calculus with set-valued attributes and aggregate functions. *ACM Transactions on Database Systems,* 12**,** 566-592.

PALLIS, G., ZEINALIPOUR-YAZTI, D. & DIKAIAKOS, M. 2011. Online Social Networks: Status and Trends. *In:* VAKALI, A. & JAIN, L. (eds.) *New Directions in Web Data Management 1.* Springer Berlin Heidelberg.

POESS, M. & NAMBIAR, R. O. 2005. Large scale data warehouses on grid: Oracle database 10*g* and HP proliant servers. *Proceedings of the 31st international conference on Very large data bases.* Trondheim, Norway: VLDB Endowment.

RAHM, E. & BERNSTEIN, P. A. 2001. A survey of approaches to automatic schema matching. *The VLDB Journal,* 10**,** 334-350.

RAHM, E. & DO, H. H. 2000. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin,* 23**,** 3-13.

SANTOS, V. & BELO, O. No Need to Type Slowly Changing Dimensions. *In:* POWELL, P., NUNES, M. B. & ISAÍAS, P., eds. IADIS International Conference Information Systems 2011, 11 - 13 March 2011 2011a Avila, Spain. 129--136.

SANTOS, V. & BELO, O. Slowly Changing Dimensions Specification a Relational Algebra Approach. International Conference on Advances in Communication and Information Technology (CIT-2011), 2011-12-01 2011b Amsterdam, Netherlands. ACEEE, 50-55.

SANTOS, V. & BELO, O. Using Relational Algebra on the Specification of Slowly Changing Dimensions - A First Step. 6ª Conferência Ibérica de Sistemas e Tecnologias de Informação, 15 a 18 de Junho 2011c Chaves, Portugal.

SANTOS, V. & BELO, O. 2013. Modeling ETL Data Quality Enforcement Tasks Using Relational Algebra Operators. *Procedia Technology,* 9**,** 442-450.

SANTOS, V. & BELO, O. Modelling ETL Conciliation Tasks Using Relational Algebra Operators. 8th European Modelling Symposium on Mathematical Modelling and Computer Simulation - EMS 2014, 21 – 23 October 2014 Pisa, Italy. IEEE Computer Society, 275-280.

SANTOS, V., OLIVEIRA, B., SILVA, R. & BELO, O. Distribuição de tarefas ETL em ambientes GRID. 11ª Conferência da Associação Portuguesa de Sistemas de Informação (CAPSI 2011), 19-21 Outubro 2011 Lisboa, Portugal.

SANTOS, V., OLIVEIRA, B., SILVA, R. & BELO, O. 2012. Configuring and Executing ETL Tasks on GRID Environments - Requirements and Specificities. *Procedia Technology,* 1**,** 112-117.

SANTOS, V., SILVA, R. & BELO, O. 2014. Towards a Low Cost ETL System. *International Journal of Database Management Systems,* 6**,** 67.

SIMITSIS, A. 2005. Mapping conceptual to logical models for ETL processes. *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP.* Bremen, Germany: ACM.

SIMITSIS, A. & VASSILIADIS, P. A Methodology for the Conceptual Modeling of ETL Processes *In:* PERNICI, J. E. A. R. M. A. B., ed. The 15th Conference on Advanced Information Systems Engineering (CAiSE '03), 16-20 June 2003 2003 Klagenfurt/Velden, Austria. CEUR-WS.org, 305-316.

SIMITSIS, A. & VASSILIADIS, P. 2008. A method for the mapping of conceptual designs to logical blueprints for ETL processes. *Decision Support Systems,* 45**,** 22-40.

SIMITSIS, A., VASSILIADIS, P. & SELLIS, T. Logical Optimization of ETL Workflows. Proceedings of the 4th Hellenic Data Management Symposium, 2005a Athens, Greece. 55-65.

SIMITSIS, A., VASSILIADIS, P. & SELLIS, T. 2005b. Optimizing ETL Processes in Data Warehouses. *Proceedings of the 21st International Conference on Data Engineering.* IEEE Computer Society.

SIMITSIS, A., VASSILIADIS, P. & SELLIS, T. 2005c. State-space optimization of ETL workflows. *IEEE Transactions on Knowledge and Data Engineering,* 17**,** 1404-1419.

SIMITSIS, A., WILKINSON, K., CASTELLANOS, M. & DAYAL, U. 2009. QoX-driven ETL design: reducing the cost of ETL consulting engagements. *Proceedings of the 35th SIGMOD international conference on Management of data.* Providence, Rhode Island, USA: ACM.

SIMITSIS, A., WILKINSON, K., DAYAL, U. & CASTELLANOS, M. Optimizing ETL workflows for fault-tolerance. *In:* LI, F., MORO, M. M., GHANDEHARIZADEH, S., HARITSA, J. R., WEIKUM, G., CAREY, M. J., CASATI, F., CHANG, E. Y., MANOLESCU, I., MEHROTRA, S., DAYAL, U. &

TSOTRAS, V. J., eds. Proceedings of the 26th International Conference on Data Engineering, March 1-6, 2010 2010 Long Beach, California. 385-396.

STONEBRAKER, M., HELD, G., WONG, E. & KREPS, P. 1976. The design and implementation of INGRES. *ACM Transactions on Database Systems,* 1**,** 189-222.

TODD, S. J. P. 1976. The Peterlee Relational Test Vehicle&#x2014;a system overview. *IBM Systems Journal,* 15**,** 285-308.

TRUJILLO, J. & LUJÁN-MORA, S. 2003. A UML Based Approach for Modeling ETL Processes in Data Warehouses. *Conceptual Modeling - ER 2003.* Berlin Heidelberg: Springer -Verlag.

TZIOVARA, V., VASSILIADIS, P. & SIMITSIS, A. 2007. Deciding the physical implementation of ETL workflows. *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP.* Lisbon, Portugal: ACM.

VASSILIADIS, P., SIMITSIS, A., GEORGANTAS, P. & TERROVITIS, M. 2003. A Framework for the Design of ETL Scenarios. *In:* EDER, J. & MISSIKOFF, M. (eds.) *Advanced Information Systems Engineering.* Springer Berlin / Heidelberg.

VASSILIADIS, P., SIMITSIS, A., GEORGANTAS, P., TERROVITIS, M. & SKIADOPOULOS, S. 2005. A generic and customizable framework for the design of ETL scenarios. *Information Systems,* 30**,** 492-525.

VASSILIADIS, P., SIMITSIS, A. & SKIADOPOULOS, S. 2002a. Conceptual modeling for ETL processes. *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP.* McLean, Virginia, USA: ACM Press.

VASSILIADIS, P., SIMITSIS, A. & SKIADOPOULOS, S. Modeling ETL activities as graphs. *In:* LAKSHMANAN, L. V. S., ed. Design and Management of Data Warehouses 2002, Proceedings of the 4th Intl. Workshop DMDW'2002, 27 May 2002 2002b Toronto, Canada. CEUR-WS.org, 52-61.

VASSILIADIS, P., SIMITSIS, A. & SKIADOPOULOS, S. 2002c. On the Logical Modeling of ETL Processes. *Proceedings of the 14th International Conference on Advanced Information Systems Engineering.* Springer-Verlag.

VASSILIOU, Y. 1979. Null values in data base management a denotational semantics approach. *Proceedings of the 1979 ACM SIGMOD international conference on Management of data.* Boston, Massachusetts: ACM.

VOUK, M. A. 2008. Cloud Computing – Issues, Research and Implementations. *Journal of Computing and Information Technology,* 16**,** 235-246.

WEHRLE, P., MIQUEL, M. & TCHOUNIKINE, A. A Model for Distributing and Querying a Data Warehouse on a Computing Grid. Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on, 20-22 July 2005 2005. 203-209 Vol. 1.

YU, L. L., ASUR, S. & HUBERMAN, B. A. 2011. What Trends in Chinese Social Media. *CoRR,* abs/1107.3522.