ETL Testing Analyzer

-MASTER THESIS-

In Partial Fulfillment

of the Requirements for the Degree

Information Technologies for Business Intelligence

by

Claudiu Ionut Murar

December, 2014

Author: Claudiu Ionut Murar
Supervisors: Oscar Romero, Alberto Abello
ABSTRACT

ETL testing techniques are nowadays widely used into the data integration process. These techniques refer to the ability of being able to check whether the information that is loaded into the data warehouse has correctly followed all the transformations steps. Due to the errors that might occur during the extraction, transformation and load stages, there is a need to monitor and handle the errors that can cause severe data quality issues into the data warehouse.

This thesis paper is based on the information gathered from a previous project that was performed at UPC. The main goal of this project was to help the professors from UPC who are teaching the “Database” course, to better analyze the performance of the students. Therefore, an ETL system was implemented with the goal of extracting student information from multiple sources, transform and load it into the data warehouse. This information can refer to the student’s personal data, the exercises they are performing, etc. The initial ETL design was based on the creation of the data warehouse schema containing the main dimensions and fact tables. The main issue is that it did not present any monitoring and error handling functionalities, even though the system was generating several errors every time the ETL was executed. The steps I have followed while working on this thesis project have been to model the initial ETL process using a BPMN representation, include error handling and monitoring functionalities and ultimately re-design the initial ETL processes using a chosen tool. Although the initial processes were modelled using Pentaho Kettle, due to the new requirements regarding the error handling and monitoring capabilities I had to perform a comprehensive ETL tool comparison to check what is the tool that can better answer the requirements of this project.

The last part of my thesis was to build a web application that can allow the final user of this system to interactively run the ETL from the application interface. The user should be allowed to input the necessary data sources that will be fetched by the ETL extraction components and insert the necessary database connection parameters. Moreover, after the user runs/execute the ETL flow the application should display information about the existing errors that might have occurred during the execution, in a user friendly and understandable manner.
I am grateful to all the people who supported me throughout my thesis work. I would like to express my sincere gratitude to my advisors Oscar Romero and Alberto Abello for the continuous support of my study and research, for their patience, motivation and enthusiasm. Their guidance helped me during all the time of research and writing of this thesis. I would also like to thank the rest of my professors for their encouragement, insightful comments, and all the help they have provided me. Last but not the least, I would like to thank my family: my parents Marian Murar and Titiana Murar, for supporting me spiritually throughout my life. I am deeply grateful to my sister as well, who has been advising and morally supporting me during the last 6 months.

Barcelona, December 2014
MURAR CLAUDIU IONUT
# TABLE OF CONTENTS

Abstract ...................................................................................................................... ii

Acknowledgments .................................................................................................. iii

List of Tables .......................................................................................................... vii

List of Figures ......................................................................................................... viii

1. INTRODUCTION .............................................................................................. 1

1.1 BACKGROUND ............................................................................................... 1

1.2 CONTEXT ........................................................................................................ 2

1.3 MOTIVATION ................................................................................................... 2

1.4 PROBLEM AREA .............................................................................................. 3

1.5 OBJECTIVES ................................................................................................... 4

1.6 INITIAL PLANNING ......................................................................................... 4

1.7 STRUCTURE OF THE REPORT ........................................................................ 5

2. RESEARCH BACKGROUND .............................................................................. 7

2.1 ETL TESTING TECHNIQUES ........................................................................ 7

2.2 ETL TESTING CHALLENGES ......................................................................... 7

2.3 ETL TOOLS ................................................................................................... 8

2.3.1 General Information ................................................................................... 8

2.3.2 ETL tools comparison criteria ................................................................. 9

2.4 CONCEPTUAL DESIGN OF ETL PROCESSES ...................................... 10

3. ETL CONCEPTUAL DESIGN USING BPMN ................................................ 12

3.1 INITIAL ETL DESIGN ................................................................................... 12

3.1.1 Data Sources .............................................................................................. 12

3.1.2 Main dimensions and Fact Tables ........................................................... 14

3.2 ETL DESIGN ERRORS .................................................................................. 15

3.2.1 ETL failing causes .................................................................................... 16

3.2.2 Design consistencies ................................................................................ 17

3.2.3 Missing functionalities ............................................................................. 17

3.3 BPMN REPRESENTATION: STATIC LEVEL ............................................. 17

3.3.1 Create Monitor/Error logging tables ...................................................... 17

3.3.2 Generate ETL ID ....................................................................................... 20

3.3.3 Insert Candidate Dimension ................................................................. 20

3.3.4 Insert Delivery System Dimension ......................................................... 22

3.3.5 Insert Experiment Dimension ............................................................... 23

3.3.6 Insert Item Dimension ............................................................................ 23

3.3.7 Insert Thematic Dimension .................................................................... 25

3.3.8 Insert Request Time Dimension ............................................................ 25

3.3.9 Create/Update Temporary Tables .......................................................... 26

3.3.10 Experiments Fact Table ......................................................................... 26

3.3.11 Invalid Fact Table .................................................................................. 27

3.3.12 Valid Fact Table ..................................................................................... 27

3.3.12.1 Update response processing ............................................................ 29

3.4 BPMN REPRESENTATION: DYNAMIC LEVEL ................................... 29

3.4.1 CONTROL FLOW ...................................................................................... 29
3.4.2 Insert Candidate Dimension ................................................................. 31
3.4.2.1 Compute course and semester information.......................................... 32
3.4.3 Insert Delivery System Dimension ...................................................... 33
3.4.4 Insert Experiment Dimension ............................................................. 34
3.4.5 Insert Item Dimension ........................................................................... 34
3.4.6 Insert Thematic Dimension .................................................................... 35
3.4.7 Insert Request Time Dimension ........................................................... 35
3.4.8 Create/Update Temporary Tables ....................................................... 36
3.4.9 Experiments Fact Table ......................................................................... 37
3.4.10 Invalid Fact Table ................................................................................ 38
3.4.11 Valid Fact Table .................................................................................. 39
3.4.11.1 Update response processing ............................................................. 41

4. ETL TOOLS .................................................................................................. 42
4.1 TOOLS COMPARISON .............................................................................. 43
4.2 TOOL SELECTION JUSTIFICATION ....................................................... 45

5. ETL DESIGN USING TALEND OS ............................................................... 46
5.1 CONTROL FLOW ..................................................................................... 46
5.2 CREATE LOGGING TABLES ................................................................. 47
5.2.1 Monitoring tables ................................................................................. 47
5.2.2 Error logging tables .............................................................................. 49
5.3 ETL ID GENERATION ............................................................................... 50
5.4 MONITORING PROCEDURE DESIGN ................................................... 51
5.5 ERROR HANDLING DESIGN ................................................................. 54
5.5.1 Create/Update Temporary tables ......................................................... 55
5.5.1.1 Missing file for updating the temporary tables .................................... 55
5.5.2 Missing attributes from data sources .................................................... 56
5.5.3 Fact tables error handling ................................................................. 56
5.5.4 ETL Rollback operation (Updates) ....................................................... 57

6. ETL ANALYSER WEB APPLICATION ....................................................... 59
6.1 REQUIREMENTS SPECIFICATION ......................................................... 59
6.2 PURPOSE .................................................................................................. 59
6.3 FUNCTIONAL REQUIREMENTS ............................................................. 59
6.3.1 Main interface ....................................................................................... 59
6.3.2 Output interface ................................................................................... 61
6.4 NON FUNCTIONAL REQUIREMENTS ..................................................... 61
6.5 USE CASES .............................................................................................. 61
6.6 MONITOR/LOGGING TABLES ............................................................... 63
6.6.1 Monitor logging insertion table ........................................................... 64
6.6.2 Monitor logging extraction table ......................................................... 65
6.6.3 Error logging table ............................................................................... 66

7. FUTURE WORK ............................................................................................ 68
7.1 ETL LEVEL ............................................................................................... 68
7.1.1 Design features and functionalities ...................................................... 68
7.1.2 Error Handling/Monitor logging .......................................................... 68
7.1.3 Rollback Update Operation ................................................................. 69
7.1.4 Improve system’s performance ............................................................. 69
7.2 WEB APPLICATION .................................................................................. 69

8. SUMMARY .................................................................................................. 70
8.1 ETL design improvements........................................................................................................... 70
  8.1.1 ETL - BPMN consistency ......................................................................................................... 70
  8.1.2 Monitor/Error Logging ........................................................................................................... 71
  8.1.3 ETL tools summary .................................................................................................................. 72
8.2 ETL analyzer web application ..................................................................................................... 73
8.3 General remarks .......................................................................................................................... 74
8.4 Final planning .................................................................................................................................. 75

9. References ......................................................................................................................................... 76

Appendix A: Additional design features and functionalities .............................................................. 78
  A.1 Extra functionalities ....................................................................................................................... 78
  A.2 BPMN consistency ....................................................................................................................... 78
  A.3 Checking for duplicate values ..................................................................................................... 79
    A.3.1 Using tMatchGroup component ....................................................................................... 80
    A.3.2 Using tUniqueRow component ......................................................................................... 81

Appendix B: ETL tools information ................................................................................................... 83
  B.1 Pentaho Kettle ............................................................................................................................. 83
    B.1.1 Connectivity ......................................................................................................................... 84
    B.1.2 Operations ............................................................................................................................. 85
    B.1.3 Debugging .............................................................................................................................. 87
    B.1.4 Monitoring ............................................................................................................................ 87
    B.1.5 Error Handling ...................................................................................................................... 88
    B.1.6 Integration capabilities ......................................................................................................... 89
  B.2 Talend Open Studio ..................................................................................................................... 89
    B.2.1 Connectivity .......................................................................................................................... 89
    B.2.2 Operations ............................................................................................................................. 90
    B.2.3 Debugging .............................................................................................................................. 92
    B.2.4 Monitoring ............................................................................................................................ 92
    B.2.5 Error Handling ...................................................................................................................... 93
    B.2.6 Integration capabilities ......................................................................................................... 94
  B.3 Informatica PowerCenter ............................................................................................................ 94
    B.3.1 Connectivity .......................................................................................................................... 94
    B.3.2 Operations ............................................................................................................................. 94
    B.3.3 Debugging .............................................................................................................................. 95
    B.3.4 Monitoring ............................................................................................................................ 95
    B.3.5 Error Handling ...................................................................................................................... 95
    B.3.6 Integration capabilities ......................................................................................................... 96

Appendix C: Preconditions, post conditions ..................................................................................... 97
  C.1 Initial requirements ...................................................................................................................... 97
  C.2 Operations ..................................................................................................................................... 98
    C.2.1 Extract ................................................................................................................................... 98
    C.2.2 Select ................................................................................................................................... 99
    C.2.3 Create .................................................................................................................................. 102
    C.2.4 Insert ................................................................................................................................... 103
    C.2.5 Rollback ................................................................................................................................ 105
    C.2.6 Update .................................................................................................................................. 105
LIST OF TABLES

Table 1. Initial planning. ........................................................................................................... 4
Table 2. Tools comparison. ........................................................................................................ 43
Table 3. Non Functional requirements. ..................................................................................... 61
Table 4. Final planning................................................................................................................ 75
Table B.1 Extract Operation: Pentaho Kettle. .......................................................................... 84
Table B.2 Join Operation: Pentaho Kettle .............................................................................. 84
Table B.3 Insert operation: Pentaho Kettle ............................................................................. 85
Table B.4 Extract Operation: Talend OS ................................................................................. 90
Table B.7 Join Operation: Talend OS ..................................................................................... 91
Table B.8 Insert Operation: Talend OS .................................................................................. 91
Table C.1 Parameters of the system ...................................................................................... 97
Table C.2 Existing tables in DB ............................................................................................. 97
LIST OF FIGURES

Figure 3.1.1 .TXT File Extract. ..........................................................13
Figure 3.1.2 .XML File Extract. ..........................................................13
Figure 3.1.3 Existing Database tables. ..................................................14
Figure 3.3.1 Control Flow: Static level. .................................................18
Figure 3.3.2 Create Monitor/Error logging tables. ..................................20
Figure 3.3.3 Candidate Dimension: Static Level ....................................21
Figure 3.3.4 Delivery System Dimension: Static Level ............................23
Figure 3.3.5 Experiment Dimension: Static level ......................................23
Figure 3.3.6 Item Dimension: Static level ..............................................24
Figure 3.3.7 Thematic Dimension: Static level .........................................25
Figure 3.3.8 Request Time Dimension: Static level .................................25
Figure 3.3.9 Create/Update Temporary tables: Static level ........................26
Figure 3.3.10 Experiments Fact Table: Static level ...................................27
Figure 3.3.11 Invalid Fact Table: Static level ..........................................27
Figure 3.3.12 Valid Tact Table: Static level ............................................28
Figure 3.4.1 Control Flow: Dynamic level ..............................................30
Figure 3.4.2 Candidate Dimension: Dynamic Level ................................31
Figure 3.4.4 Delivery System Dimension: Dynamic Level .........................33
Figure 3.4.5 Experiment Dimension: Dynamic level ................................34
Figure 3.4.6 Item Dimension: Dynamic level ........................................35
Figure 3.4.7 Thematic Dimension: Dynamic level ....................................35
Figure 3.4.8 Request Time Dimension: Dynamic level ............................36
Figure 3.4.9 Create/Update Temporary tables: Dynamic level ....................37
Figure 3.4.10 Experiments Fact Table: Dynamic level ..............................38
Figure 3.4.11 Invalid Fact Table: Dynamic level .....................................39
Figure 3.4.12 Valid Tact Table: Dynamic level .......................................40
Figure 5.1.1 High level flow of the ETL processes ....................................47
Figure 5.3.1 ETL ID generation ..........................................................50
Figure 5.3.2 Dimensions Group .........................................................51
Figure 5.4.1 Candidate Dimension Monitoring procedure ........................52
Figure 5.4.2 Monitored information .....................................................53
Figure 5.5.1 Errors catching ...............................................................54
Figure 5.5.2 Check for missing files .....................................................55
Figure 5.6.1 Fact Table Creation .........................................................56
Figure 6.3.1 Main User Interface ..........................................................62
Figure 6.6.1 Monitor Logging Insertion ................................................64
Figure 6.6.2 Monitor Logging Extraction ...............................................65
Figure 6.6.3 Error Logging ...............................................................66
Chapter 1. Introduction

1.1 Background

ETL testing concepts appeared for the specific challenges that can occur during the data integration stage. These challenges can refer to: integrating data from multiple sources and populating a data warehouse with information. When dealing with ETL operations, the first step is to extract data and prepare it for further transformations. The transform and load steps are normally applying a set of rules to the extracted information in order to successfully load the data from source to target destination [11]. During these steps there is a need to consider the data quality issues that might generate inconsistencies in the data warehouse. Generally, during the ETL process data is coming from multiple sources, in some cases unreliable sources. Due to this reason there is a need to perform comprehensive error handling procedures in order to see what are the issues that have appeared during the execution. Implementing an ETL design that can support error handling techniques, is often considered one of the most difficult stages of a DW project [12]. Fortunately, nowadays there are many ETL tools on the market that can solve these data integration issues. As a general definition, an ETL tool is a software responsible for gathering data, cleansing, customizing and inserting it into the target data warehouse [15].

The project that I will be referring throughout this thesis paper describes an ETL system that was previously built with the specific purpose of helping the professors from UPC who are teaching the ‘Database’ course, to gather information about students’ performance, details about the exercises they are performing, etc. This project has been started a few years ago and it contained the initial data warehouse design of the main dimensions and fact tables that are forming the system. The main problems of the initial ETL design is the fact that it contained errors that were causing the system to fail. Moreover, the design is not fully consistent with the final users’ requirements, reason why there is a need to redesign the flow and ensure a proper consistency level with their requirements.

The first step of this thesis paper was to conceptually model the ETL operations. There are several approaches that can be followed when conceptually modelling the ETL processes of a system. Paper [7] introduces an UML based approach for modelling ETL processes in a data warehouse. The authors provide a set of standardization techniques and mechanisms that can be applied to the representation of different ETL operations. On the other hand, as we will see in Chapter 2.4 several papers have introduced the concept of using a BPMN representation to model the ETL processes. Eventually, together with the final users of the system I have decided to use BPMN for conceptually designing the ETL processes and insuring a thorough consistency between their requirements and the actual design implementation. During the dynamic modelling level of the operations, I have included the error handling and monitoring components, which I have later on integrated into the ETL design.
The next step after modelling the processes in BPMN, was to build and implement the new design using an ETL tool. In the specific case of our project, the initial ETL design was modelled using the Pentaho Kettle ETL tool. Due to the new functionalities and improvements that I have added during the conceptual modelling stage, there is a need to make a comparison between different ETL tools and decide which tool can better satisfy the requirements of our project.

Therefore, a different step before the implementation phase was to perform a tool comparison between Pentaho Kettle, Talend Open Studio and Informatica Power Center. After choosing the right tool to implement the ETL design, I had to ensure a good level of consistency with the BPMN conceptual representation of the processes.

The final part of this thesis project was to build a web application that can allow the user to input the necessary data sources, database connection parameters, and run the ETL flow from the front end interface of the web application. As a result this application should display information about the errors that might have appeared during the execution and about the operations that have been monitored.

1.2 Context
This thesis paper is based on the information gathered from a previous project that was performed at UPC, entitled “Implementing a Data Warehouse ETL tool and performing analysis using an OLAP tool”. The ultimate goal of this project was to help the teachers of the “Database” subject from the UPC department, to analyze the performance of the students. Therefore, initially I had to take into consideration the information that has been previously gathered. This information has helped me to form a starting point for my thesis project, so that later on I could perform comprehensive tests for testing, monitoring and handling the errors that might occur during the ETL execution process.

1.3 Motivation
At a personal level, my motivation for tackling this topic for my thesis project is applying the business intelligence and data analytics knowledge that I gathered during the last two years of studies. Throughout the implementation of this project I have used several tools and got more experienced in performing advanced ETL operations. Moreover, by integrating the ETL job within the web application, I have improved my web development skills in using technologies such as HTML and PHP.

Regarding the academic part of my research I am positive that the results of my project can provide good insights and help the professors from UPC to better analyze and test the information coming from the ETL process.
1.4 Problem Area

Data integration stage plays a significant role in ensuring that the business information is consistent, exact and reliable [10]. Handling and tracking the errors that might occur during this stage is important because it gives the user a better perspective over the reasons why the quality issues appear. Generally, when errors occur during the ETL stages we should check whether data is missing from the data sources, or if the information gets incorrectly transformed. In paper [11], the authors suggest that the issues that can generally cause errors in the ETL execution are influenced by several factors, such as:

a. **Missing requirements information**: Business requirements should specifically state how the system needs to behave, what are the things that it has to do and should not do under different circumstances. Moreover the requirements should cover each and every aspect of the ETL system. Due to this reason, initially there is a need to conceptually represent the ETL processes and ensure the fact that the user requirements have been correctly understood.

b. **Data is coming from multiple sources**: In the case of this project, we are using multiple sources of information, such as: data coming from the existing database tables, information coming from external files (.txt, xls,.xml),etc. Therefore, it is important to monitor and track this information that is being extracted and make sure that all this data gets processed consistently and that later on is connected with each other according to the join attributes.

c. **Incomplete data**: One of the issues that is causing failing errors within the system are the missing attributes from the data sources. For example, let us assume that we are using a special attribute to join two data sources together. In case this attribute is missing from one of the two sources, the ETL flow execution will be affected. This scenario is commonly encountered in our context.

d. **Incompatible and duplicate data**: The entries that are generated by a system multiple times could influence the accuracy and consistency of the ETL process analysis. In our case, checking for duplicate entries within the source files is not considered a priority. It is assumed that the information which is generated by the learning platforms (Raco, Moodle) does not contain duplicate values. However, this issue will be tackled during the design implementation, Appendix A contains a detailed description w.r.t this issue.

e. **High volume of data**: Due to the high volume of data that might need to be processed, the performance of the system that generates the ETL processes can have a negative impact over the consistency of the data. Although in our context the volume of data from the data sources is not considered to be big, this performance issue can still affect the overall performance of the system. As a solution the queries used during the fact tables creation can be tested and if necessary can be optimized.
Another challenge that this thesis paper tackles, is the ability to execute the ETL flow from a more interactive and user friendly environment. In some cases the final user of the system might not have the necessary technical experience to execute the processes from inside the ETL tool. Although the existing ETL tools are able to execute the extraction, transformation and loading procedures, the logical flow of the ETL design can be hardly understood by users who are not familiar with the these kind of tools. Therefore, there is a need to integrate the ETL job within a Web application interface that can offer an understandable and easy way to navigate. The interface should allow the tester to automatically input the chosen data sources, input the database connection parameters and finally run/undo the ETL. As a result it should display relevant information w.r.t to the errors that might have appeared during the execution of the ETL processes.

1.5 Objectives
The main objectives of this thesis paper are:
   a. Find and document the initial design errors that might have caused an incorrect functionality of the ETL processes
   b. Conceptually model the ETL processes using a BPMN representation. The conceptual design should be modeled at two different levels: static and dynamic. The dynamic level should conceptually contain the newly added functionalities, which are used to implement the error handling and monitoring procedures.
   c. Perform an ETL tools comparison, to find the right tool that can better answer the specific requirements of this project.
   d. Remodel and implement the new design by using the selected ETL tool
   e. Build a functional application that would allow the user to input the necessary data sources, input the database connection parameters and run the ETL flow from the front end interface of the application. As a result, the application should display information about the errors that might have occurred during the execution and information about the operations which have been monitored.
   f. Relate the contribution of this thesis paper to the project that has been previously performed

1.6 Initial planning
The thesis project has been performed during 3 months, between September and December. Due to the limited amount of time, the initial planning of the thesis has been condensed in specific tasks which helped me to fulfill the requirements of this project. We can notice in the table below the main activities that I have initially considered.
<table>
<thead>
<tr>
<th>Activity</th>
<th>Start Date</th>
<th>End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Study the initial material</td>
<td>15-September</td>
<td>25-September</td>
</tr>
<tr>
<td>-Get familiar with the initial ETL design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Create a BPMN representation of the ETL processes: static and dynamic level</td>
<td>25-September</td>
<td>08-October</td>
</tr>
<tr>
<td>-ETL tools installation and familiarization</td>
<td>08-October</td>
<td>18-October</td>
</tr>
<tr>
<td>-ETL tools comparison</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Finalize the BPMN conceptualization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Redesign the ETL using the chosen tools</td>
<td>18-October</td>
<td>25-October</td>
</tr>
<tr>
<td>-Build the Monitoring/Errors logging tables</td>
<td>25-October</td>
<td>02-November</td>
</tr>
<tr>
<td>- Improve the ETL design errors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Build the Web application front end interface</td>
<td>02-November</td>
<td>15-November</td>
</tr>
<tr>
<td>-Integrate the ETL job with the web application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Present the initial thesis outline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Create dashboards to present the testing results from the logging tables</td>
<td>15-November</td>
<td>22-November</td>
</tr>
<tr>
<td>-Start the thesis writing process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Finalize the thesis writing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Apply additional changes suggested by the final users</td>
<td>22-November</td>
<td>05-December</td>
</tr>
</tbody>
</table>

| Table 1. Initial Planning |

1.7 Structure of the report
The rest of the report is organized as follows:
Chapter 2 contains information about the research background that I have gathered during the last months. In Chapter 3 I will explain the initial stage of the ETL design. Moreover, this section displays the final version of the BPMN conceptual representations, which I have used as a basis to redesign and implement the ETL workflow using a selected tool. Chapter 4 presents the ETL tools comparison and the justification of my choice when selecting the tool. In Chapter 5 I have documented the implementation steps that I have followed while implementing the new ETL flow. Chapter 6 emphasizes the details of the web application that I have built for running the
ETL flow. Chapter 7 contains information regarding the future work that can be achieved. Chapter 8 presents the summary of the work I have done so far and highlights the contributions of my paper. Moreover, I have included in the paper three different appendixes which I consider to be useful and informative. Appendix A presents a set of functionalities that can be later on integrated in the system. The information from appendix B can be used to back up my justification for using the ETL tool. Furthermore, Appendix C contains the precondition and post conditions which I have followed while modelling the ETL processes.
Chapter 2. Research Background

2.1 ETL Testing Techniques
ETL testing plays a significant role in ensuring that the information is exact, consistent and reliable. According to article [13], we can consider two high level approaches when performing the testing methods:

- **Approach I**: Validate the information that is extracted from the data sources after it is outputted into the DW. This approach will validate if the data which is present into the DW has been transformed according to the correct rules or not. The disadvantage in using this approach is the fact that we have no control over the transformation process that occurs between the source and the target system.

- **Approach II**: Validate the data through each step of the transformation phase, including the final insertion in the DW. For example, in stage one we can verify if the sources from where we are extracting information are available or not. During this stage we can also verify the consistency of the information inside the data source. For the second stage we can apply the transformation rules and monitor the execution of each step in order to track and understand how the ETL system works. During the last stage we could test the insert operation and the actual load of data into the DW.

For the specific case of my thesis project I have decided to use the second approach since it presents a more detailed method of how to monitor all the steps that take place during an ETL execution (extract, transform, load).

2.2 ETL Testing Challenges
As stated in paper [14], we first need to understand the ETL logical flow before checking for the errors that might appear. The author refers to different issues that are generally the cause for ETL execution failures, such as:

- **Missing Values**: A missing record may have certain dependencies with some other fields in some another tables. This issue can further aggravate data quality problems. In our context, the data sources which are used to extract information can contain missing attributes. According to the author, these missing attributes are affecting the dependencies between the fact tables and the dimensions.
Normalization issues: When we are merging data from different sources, we can end up in de-normalizing the target database. If the database if poorly designed it may become difficult to use and can even fail to work properly.

Furthermore, article [13] suggests that in order to perform comprehensive monitoring, testing and error handling techniques, we first need to consider the following questions:

- **a. Has all the expected data been loaded from the source file?** In order to check if all the required data has been loaded from the data sources, we need to monitor this step and log the entries’ information.
- **b. Did all the records load into the right place?** According to the author we need to verify and validate if each attribute that is extracted from each source is loaded in the correct field.
- **c. Are there any duplicate values?** A checkup for duplicate values could be performed in order to see if the data sources contain any duplicate entries.
- **d. Is data being corrected and rejected correctly?** Sometimes the information may be incomplete or unclear. This data must be logged and stored for further clarification before it is passed to the next step.

The solutions offered by the author of the article are the following: verify that the information is transformed correctly according to the requirements, check the DBMS constraints, verify that the data transformation process from source to target executes according to the initial rules, check if the expected data is added in target system, verify that the DB fields are loaded and do not contains errors.

### 2.3 ETL Tools

#### 2.3.1 General Information

The main functionality of an ETL tool is to extract, transform and load data from the source into the target system. The advantage of using ETL tools is that they can easily extract information from multiple sources such as flat files or different databases, while modifying and moving the data [15].

In our specific context, the ETL tool that I am going to use to redesign and implement the processes, should answer the specific requirements of this project. Initially the ETL design was modelled using Pentaho Kettle. Since the design did not contain any error handling and monitoring components, there is a need to perform a comprehensive comparison and decide which is the best tool that we can use to implement these new functionalities.

Different articles [15][16] have listed some of the main functionalities that an ETL tool has to offer:

- The ability to deal with multiple input data formats (such as flat files, SQL databases, SAS data sets, etc.)
• Support of a GUI interface that allows the connection of data elements from source to destination
• Built-in data analysis functionality which can allow us to examine the information from different aspects
• The ability to perform error tracking and monitor logging functionalities

2.3.2 ETL Tools Comparison Criteria
Some of the most popular ETL tools that exist on the market are: Pentaho Kettle[17], Talend Open Studio[18], Informatica PowerCenter[19], Data Migrator[20], DataFlow Manager, etc. Several comparisons between different types of ETL tools have been previously performed in [21],[22]. Some examples of the most common benchmarks that were used in previous papers include factors, such as: User interface, Infrastructure, Speed, Performance, Scalability, Data Quality/Profiling, Debugging facilities, etc.

In the specific case of our project we are not interested in categories such as speed and scalability due to the fact that the volume of information inside our data sources is not big. On the other hand it is highly important that the ETL tool should have the debugging and error handling functionalities. Another aspect that we are taking into account is the user-friendliness of the tool, in order to ease the work of the developer who might deal with the data integration implementation phase.

a. User Interface
According to article [23], there are basically two different families of ETL tools: Script-based tools and Graph-based tools. Nowadays script-based tools are considered to be outdated although they are thought to be more versatile. The conclusion of the author is that nowadays it is more difficult to perform data transformations using script-based ETL tools, such as: SAS Base, Scriptella, etc. On the other hand graph-based tools are more intuitive and offer a great range of functionalities for representing the ETL transformations during the data integration stage. The most popular tools that are using a graph based user interface are Pentaho Kettle[17], Talend OS[18], Data Migrator[20], etc.

b. Debugging facilities
In paper [24] the authors present a framework criteria for ETL tools evaluation. The comparison is made from different perspectives including debugging support facilities. The tools mentioned and tested in the paper are: Microsoft SQL Integration Service, Pentaho Kettle and ETI High Performance Data Integration. According to the authors of the paper one of the most basic debugging functionalities that an ETL tool has to provide, is the possibility to add breakpoints and pause the transformations to specific points during the execution. These pause intervals can help the user to better analyze the execution of the ETL steps. In the paper mentioned above, the
tools are compared from many different perspectives, reason why I will not make any generalization based on this specific debugging support criteria presented in the paper. However, as we will see later the debugging functionalities are a decisive differentiation metric between the tools that I have compared.

c. Error handling/Monitoring
As stated before one of the goals of this thesis project is to include into the design of the ETL, error handling and monitoring components that can catch the errors and track the specified information. Paper [25] presents an evaluation framework for data quality tools. The authors of the paper are stating that it is very important to measure the quality of data, hence to handle the errors that can occur throughout the ETL process. The tools that are compared in the paper are mainly commercial ETL tools which are nowadays used by organizations for detecting data quality issues that can appear in their CRM systems.

2.4 Conceptual Design of ETL processes
Conceptual design of ETL processes is part of a more general picture of conceptualizing the data warehouse design. This notion was initially introduced by Golfarelli in [4]. The reason behind using conceptual design for representing these processes is to correlate the data integration steps with the business requirements, in order to identify which information is required and how to efficiently transform and load this information in the data warehouse [1]. Although a lot of research has been already done w.r.t different ETL issues, there are only few papers that are tackling the conceptual representation of the ETL process. One of the papers which approaches the conceptual modelling of these processes is presented in [10]. The author displays an UML based approach for modelling ETL processes in a Data Warehouse environment. This paper provides the specification mechanisms for the most common ETL operations which are encountered during the data integration stage of a DW project. Furthermore, paper [2] presents a high level approach for the representation of the ETL processes, based on the Business Process Management Notation. The BPMN standardization is used to build a high level framework of the business processes, regardless of the tools that are being used. One of the advantages of using BPMN is the ability to provide a standard notation that can be understood by users having different backgrounds: business analysts, business managers, technical developers, etc. [2] The paper mentioned above presents a BPMN metamodel for representing the ETL processes. The metamodel is based on the classification of different ETL objects, derived from certain tools which are commonly used for data integration. The approach of the authors was to define two different perspectives for characterizing the ETL processes, i.e.: control flow view and data process view. The control flow perspective allows the user to have a high level overview of the ETL processes, while the data process view captures the real nature of these processes. Another similar approach is presented in [1], where the authors
display a method to design the ETL workflows using BPEL (standing for Business Process Execution Language) and BPMN. The knowledge that I got after reading papers [1],[2] has helped me to understand the usefulness of the BPMN representations. I have used this knowledge to design the main operations of the ETL processes from our system. Therefore, during the conceptual modelling of the ETL steps I have used BPMN to represent the workflow at a static and at a dynamic level.
3.1 Initial ETL Design

Initially, one of the goals of this project was to help the professors from UPC who are teaching the database course, in their decision making with regard to the exercises that are performed by students and to allow them to analyze students’ performance w.r.t their time invested for solving certain exercises. The initial design of the ETL was realized in Pentaho Kettle and incorporated the creation of the main dimensions and fact tables that are forming the system. Unfortunately the ETL design did not contain any components for catching eventual errors or for monitoring certain steps during the execution of the ETL processes. The problem is that although the execution of the ETL was sometimes failing, the user had no possibility to handle and track the errors except for the debugging panel of the Pentaho Kettle which was displaying the outputted error messages. Moreover, the design presented certain inconsistencies with the user requirements. I have noticed and corrected these issues from the initial design while conceptually representing the ETL processes using BPMN.

At the fact tables’ level, most of the errors that have occurred were due to some missing attributes from dimensions which are used as foreign keys. On the other hand at the dimensions level, the errors occur during the data extraction and insertion stage, since the data sources used in the project are unreliable and sometimes unavailable, i.e.: missing data source (.XLS file containing calendar university year information). Since data is coming from multiple sources and joined together according to certain attributes, there was a need to ensure that data is not being lost. Below I will describe the main data sources that are used throughout the development of this project.

3.1.1 Data sources

a. Student information (generated by two different learning platforms Raco and Moodle): This data is gathered in different external sources (.xls,.txt, .xml files, etc.)
   - .TXT file: In this flat file we collect information about the candidate (student) such as: DNI, username, city, country, the program degree that the student is studying, etc. Below we can see a a fragment of this .txt file:
• **XML file**: This file is automatically generated by the Moodle learning platform and contains personal information of the student, such as: *username, user id, mail, password*, etc.

```
<WWW-Crom>
  <!-- Taula mdl_user -->
  <mdl_user>
    <id>2</id>
    <auth>manual</auth>
    <confirmed>1</confirmed>
    <policyagreed>0</policyagreed>
    <deleted>0</deleted>
    <mnethostid>1</mnethostid>
    <username>claudiu.murar</username>
    <password>000000</password>
    <idnumber/>
    <firstname>Claudiu</firstname>
    <lastname>Murar</lastname>
    <email>claudiu.murar@est.fib.upc.edu
```

• **XLS file**: It contains the DNI information of the student and the mark(*nota*) that he has been assigned.

b. **Calendar university year information**:

• It refers to special dates within the calendar of the university year, such as: free days, holiday periods, etc. This temporary information is gathered into an (.XLS) file and should be inserted at the beginning of each university year in two temporary tables (*calendar_fib, class_day*). It is important to mention that one of the reasons which is causing the ETL to fail is the absence of this external source file.
c. Existing databases from the university department:

- Table staging_area_resultats: This table provides information about the student performance, such as: the time a student has spent to perform an exercise, what exercise he has done, how long the correction has lasted, the insertion moment, etc.
- Table t_questions: It provides information about the different exercises that are performed in the current semester and course. Within this table we find data such as the author, theme, difficulty degree of the exercises, etc.
- Table t_jocsproves: In this table we can find information about the different tests and experiments that were performed in different time periods.
- Table t_tematiquesquestions: It provides information about the correspondence between the topics and the exercises.

Figure below presents a more detailed description of the attributes that are contained into this tables:

<table>
<thead>
<tr>
<th>t_questions</th>
<th>staging_area_results</th>
<th>t_jocsproves</th>
<th>t_tematiques</th>
<th>t_tematiquesquest</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>serial</td>
<td>id</td>
<td>nom</td>
<td>id</td>
</tr>
<tr>
<td>autor</td>
<td>text</td>
<td>ordre</td>
<td>text</td>
<td>nom</td>
</tr>
<tr>
<td>titol</td>
<td>text</td>
<td>nominp</td>
<td>text</td>
<td>int</td>
</tr>
<tr>
<td>enunciati</td>
<td>text</td>
<td>momentoinser</td>
<td>text</td>
<td>text</td>
</tr>
<tr>
<td>tipussolucio</td>
<td>integer</td>
<td>duradacorrec</td>
<td>text</td>
<td>descripcio</td>
</tr>
<tr>
<td>difficultat</td>
<td>real</td>
<td>interval</td>
<td>text</td>
<td>text</td>
</tr>
<tr>
<td>esquema</td>
<td>text</td>
<td>resultatcodi</td>
<td>interval</td>
<td>netaja</td>
</tr>
<tr>
<td>tipusquestic</td>
<td>int</td>
<td></td>
<td></td>
<td>interval</td>
</tr>
<tr>
<td>usuariurl</td>
<td>text</td>
<td></td>
<td></td>
<td>pes</td>
</tr>
<tr>
<td>netaja</td>
<td>text</td>
<td></td>
<td></td>
<td>real</td>
</tr>
</tbody>
</table>

Figure[3.1.3]. Existing sources in the department

3.1.2 Main dimensions and Fact Tables

The initial state of the project included the load of the main dimensions and fact tables from the system, together with the corresponding data warehouse design. During this chapter I will not go through the data warehouse design, since it has been previously created and detailed. However, figure below presents a zoom inside the attributes of the main dimensions and fact tables.
When executing the ETL flow it is assumed that the dimension tables and fact tables have already been created in the database.

The initial state of the design contained errors which did not allow the user to successfully execute or run the ETL flow. Some of the initial errors which I have noticed are presented in the next section.

### 3.2 ETL Design Errors

In this section I will mention the main issues which I have initially encountered in the ETL design. I will emphasize three types of issues:

a. **ETL failing causes**: These types of errors were directly influencing the successful execution of the ETL. Due to this issues whenever the ETL flow was executed, the debugging panel of the ETL tool was outputting several error messages that were occurring during the execution.

b. **Design inconsistency issues**: Regarding these inconsistencies, I have mainly identified them throughout the iterations that I had with the product owners (my advisors) while setting up the requirements. By using BPMN to conceptually represent the ETL processes, I have tried to ensure a maximum consistency between the final users’ requirements and the actual implementation, as well as a productive communication.

c. **Missing functionalities**: The initial design did not contain any components for monitoring the ETL operations and handling the errors. Even though the system...
was not working properly and there were several outputted errors every time the system was executed, these issues were only reported in the debugging panel of the ETL tool.

3.2.1 ETL failing causes

a. Missing information: As stated before one of the first steps in our system is to extract data from multiple sources and prepare it for further processing. Due to the data quality issue from the data sources, the ETL system is prone to errors. One of the scenarios that has previously occurred, is w.r.t the missing attributes which were selected from the data sources. Their absence was causing the execution of the ETL to stop. We can mention here that one of the transformation steps within our ETL flow is to join two data source together according to a specified attribute. In some cases this attribute might be missing. This example was causing several failing errors and was stopping the execution of the ETL flow. Although, we cannot directly solve the problem of the missing attributes (since the information is automatically generated by the student learning platforms) we can at least monitor the ETL operations and handle the errors in such a way that the user will know what ETL step has failed and why it has failed. We can extrapolate the problem of missing attributes and mention that the execution of the fact tables is also affected by this issue. The successful execution of the fact tables is based on the existence of certain attributes from the dimensions, which are considered foreign keys. Due to these failing errors that were occurring in the initial ETL design, there is a need to perform a thorough error handling/monitoring procedure in order to help the user to better track the main ETL operation and handle the errors within the system.

b. Missing data sources: Another common issue that was causing the ETL to fail is the absence of one of the data sources. Specifically this data source (.XLS file) refers to the information about the calendar university year, which should normally be inserted into the temporary tables (calendar_fib and class_day). The absence of this file was initially generating errors and was stopping the ETL execution. After discussing with the product owners of this system, I have been suggested that the ETL flow should not stop in case this file is missing, but should rather continue with additional consequences which will be detailed later on in the thesis paper.

c. Additional causes:
   • Creating temporary tables: In the initial design, the temporary tables were created at the beginning of the ETL flow and were dropped at the end of flow. This might seem logical since the temporary nature of these tables implies the fact that every time the ETL processes are finished, ultimately we should also drop the tables. Due to the previous design model, in case the ETL execution was stopped before the final step of the flow, the temporary tables would not be dropped. After rerunning the processes a different error message was occurring suggesting that the tables have already been created.
• **Item, Candidate Dimensions:** In the initial design, the type of certain attributes which were inserted into the database did not map with the corresponding field from the database table. This minor inconsistency issue was causing the ETL execution to fail. The same issue appeared for the Candidate Dimension as well, when the *nota* (mark) attribute was not correctly mapped to corresponding field into the database table.

### 3.2.2 Design inconsistency issues

While conceptually representing the ETL processes using BPMN, I have found several inconsistencies within the initial design. Although these issues were generally not causing failures into the execution, they are important because it can affect the performance of the system. One of the examples from the initial design is that some operations were unnecessarily used, i.e.: computing *course* and *semester* information for the Experiments dimension, although there is no need to store the *course/semester* information into the database table.

### 3.2.3 Missing functionalities

- **Error handling/Monitoring:** The initial ETL design did not present any components for catching the errors that might occur during the execution and for tracking certain ETL operations.
- **Web application:** The web application has only been a requirement for my thesis project, therefore its implementation has been started during this project.

### 3.3 BPMN representation: Static Level

In order to achieve the objectives of this project, firstly I needed to understand the main requirements and the main reasons why the ETL was failing. First step was to conceptually re-design the ETL using a BPMN representation. During the iterations that I had with the final users of this system, I have noticed that using BPMN to display the ETL processes has eased the communication process and has ensured a maximum consistency between my implementation work and their requirements. In order to better understand the functionalities of the ETL operations, I have firstly documented a set of preconditions and post conditions. (See Appendix C).

It is important to mention that although I have tried to follow the logical flow of the initial ETL design, the BPMN diagrams which I have represented in this chapter are actually the final representations of the improved ETL design. Therefore, while redesigning and implementing the new ETL flow in Talend Open Studio, I have used the same steps and logic that is displayed in the BPMN representations from this chapter.
Control Flow
The conceptual representation of the control flow is displayed in Figure 3.3. As we can see, it contains different structures and group, such as:

a. **Create Monitor/Error logging tables:** There are three different tables were we store the logged information. I will briefly enumerate these tables, although this step will be detailed later on in the chapter.
   - *monitor_logging_insertion:* This table contains the monitored information after each *insert* operation is performed.
   - *monitor_logging_extraction:* It contains the monitored information after each *extract* and *join* operations are performed.
   - *error_logging:* This table displays the errors that have occurred throughout the execution of the system.

b. **Generate ETL Id:** Every time the user runs the ETL flow we are generating a new ID. This is useful in case the user wants to differentiate the new entries from the old entries. Moreover, in case the user wants to rollback his actions, we need to keep track of the past executions ID.

c. **Dimensions Group:** During this step we are creating the dimensions that are later on used for the fact table execution. It is important to mention that the temporary tables are created during this stage, since one of the dimensions (*RequestTime* Dimension) is directly depended of the information contained into these tables. The other dimensions are not depended of the temporary tables, as we can see in the figure below.

d. **Fact Tables Group:** In this phase we are loading the fact tables with information coming from the database dimension tables. Moreover, during this stage we are performing certain *update* operations which we will describe later on in the paper.
3.3.1 Create Monitor/Error Logging Tables

During this step I am creating the tables that are used to store the details about the operations which are being monitored. These details refer to parameters, such as: **Start Date, End Date, Lines read/written**, etc. The reason why we need to create these tables at the beginning of the ETL flow is that all the other steps are directly depended on these tables existence.

**a. Monitor Logging:** As we will see during the dynamic level of the representation there are different ETL operations which are being monitored, such as: **Extract, Join, Insert** operations. The reason behind dividing the monitored information and storing it into two different tables is for a better understandability when displaying the data in the main interface of the application. As we will later on see, the information from these tables has to be displayed in the main UI of the application. Initially, I have used only one table to keep track of the monitored information, but while designing the front end side of the application I have decided to better reorganize the information. Therefore, the monitoring information is divided into the following two tables:

- **Monitor_logging_insertion:** In this table I am mainly storing data that is generated after the **insert** operations. For example if we are interested to know how many rows have been inserted in the one of the dimensions or fact tables, we have to look into this table.

- **Monitor_logging_extraction:** This table contains the monitored information that is generated after **extraction** and **join** operations. For example if we are interested to see how many entries have been read(extracted) from any of the data sources, or how many entries have resulted after the join operation, this is the table where we are storing this parameters.

**b. Error logging:** This table is created in order to record the errors that are appearing during the ETL flow execution. The name of the table is **error_logging**.

A more detailed description about the three tables mentioned above will be presented in the following chapters. Figure below displays the conceptual steps of the ETL processes for creating the logging tables:

![Create Monitor/Error logging tables](image-url)
3.3.2 Generate ETL ID

The reasons why we need to generate a different execution ID every time we are running the system are the following:

- **Rollback action** (Functional Requirement): One of the features that is interesting to offer in the web application is a special ‘Undo’ button that would allow the user to roll back the operations that have been performed. The system has to roll back the main operations (insertions, updates), therefore there is a need to record a specific ID for each execution.

- **Understandability** (Non-functional requirement): In case the user executes the ETL flow multiple times from the Web application, it will be hard to differentiate the newly inserted logged steps from the old ones. Therefore the logging tables displayed in the Web application should have an ETL ID column, which can show the corresponding number of each execution.

In order to generate the ID I am firstly creating a new column into the database tables (ETL ID column). Afterwards I am selecting the maximum number from the values inside the column and incrementing it with one (+1) every time I am running the ETL flow, either from the tool or from the web application.

Dimensions Group

3.3.3 Insert Candidate Dimension

During this stage we are extracting student information data from different data sources, transforming and inserting it into the database (dim_candidate).

As we can see in Figure 3.3.3 there are a series of steps that are performed inside the Candidate Dimension.
Figure [3.3.3]. Candidate Dimension: Static level

a. Extraction:
The ‘student’ information is extracted from different sources, such as:

Data Source Files
- .TXT file: The attributes selected from this flat file are: DNI, Username, Titol, Repetidor, Group, etc.
- .XML file: The attributes selected are: Username, User Id, etc.
- .XLS file: The attributes selected are: DNI, Nota

Database Table
- staging_area_results (SAR): The selected attribute from the table is Idusuari

b. Join:
- Join between the .TXT <-> .XML files according to the Username attribute
- Join between the .XML File <-> staging_area_results (SAR) table according to the User ID attribute
- Join between the .XLS <-> .TXT files according to the DNI attribute
c. Union
This operation is bringing together data regarding the students and the information related to the course and the semester. Course/semester information is computed by using the momentinsert attribute from the database table staging_area_results (SAR).

d. Insert
The information that has been extracted and transformed until this final step is inserted into the database table dim_candidate.

e. Additional transformations:
- **Identify students that are not repeating the course:** Attribute (repetidor) contains information about whether or not a particular student has repeated the course. This attribute is selected from the (.txt) file. Unfortunately, due to data quality issues some rows inside this column contain null values. The students who have repeated the course have ‘[R]’ values inside the row field, while for the students who have not repeated the course the corresponding rows sometimes contain ‘null’ values. These null values should be replaced with [NR] (non-repeating).
- **Truncate nota:** The nota(mark) attribute from the (.xls) file contains non-truncated values. A transformation step should be applied for this attribute in order to truncate the values inside.
- **Get ‘subject’ field:** This variable is introduced into the system, in order to offer more detailed information regarding the student’s profile.
- **Rename attributes to English notation:** Some of the attributes should be mapped and renamed to their English notation for a better understanding.

3.3.4 Insert Delivery System Dimension
This dimension is used to collect the distinct[moodle] attribute from the staging_area_result (SAR) table. The attribute extracted from SAR table is renamed to its English notation and loaded into the dim_delivery_system database table.

a. Extraction: The distinct values from the moodle attributes are selected from SAR database table.

b. Insertion: The values are inserted into the dim_delivery_system table.

c. Additional transformations: Map the moodle attribute to new identifier [name].

The BPMN representation of the ETL processes inside the *Delivery System* Dimension is presented in the figure bellow.
3.3.5 Insert Experiments Dimension
In this dimension we store the information that we want to examine w.r.t different test kits that are being performed. Data inserted in this dimension table is related to information such as: the identifier of the test, identifier of the exercise, name of the test set, etc. Figure 3.3.5 displays the BPMN representation of the ETL processes inside the Experiments dimension.

The conceptual operations and processes inside this dimension are:

a. **Extraction**: Information is gathered from t_jocsproves database table. The attributes that are selected from this table are: id, nomjp and ordre

b. **Insertion**: Information is inserted into the database table, dim_experiments.

c. **Additional transformations**: The selected attributes are renamed to their English language notation.

3.3.6 Insert Item Dimension
During this stage we gather information about different exercise that the student must perform. Data is extracted from the t_questions database table and is related to information about: exercise difficulty, type of exercise, type of response, author, etc. Moreover course and semester
information is computed by using the \textit{momentinsert} attribute from SAR table. Figure below displays the BPMN representation of the ETL processes inside this dimension.

![BPMN diagram](image)

\textit{Figure}[3.3.6]. \textit{Item Dimension: Static level}

The conceptual operations and processes inside the \textit{Item} dimension are:

\begin{enumerate}
  \item {\textbf{Extraction}}: Information is extracted from the \texttt{t_questions} database table. The selected attributes are: \texttt{id, autor, titol}, etc. Moreover the course and semester information is computed using the \texttt{momentinsert} attribute from the SAR table
  \item {\textbf{Union}}: Information regarding the exercises is merged with the \texttt{course} and \texttt{semester} information.
  \item {\textbf{Insertion}}: After extracting and transforming the information, data is inserted into the \texttt{dim_item} table
  \item {\textbf{Additional transformations}}: The selected attributes from the \texttt{t_questions} table are renamed to their English language notation.
\end{enumerate}

\textbf{3.3.7 Insert Thematic Dimension}

This dimension is dependent of the \textit{Item dimension} because it uses the \texttt{ID} attribute from the \texttt{dim_item} table. The attributes referred in the thematic dimension are: \texttt{id}, \texttt{item} and \texttt{theme}. Figure 3.3.7 displays the BPMN representation of the ETL processes within this dimension.
The conceptual operations and processes inside this dimension are:

a. Extraction: Information is extracted from the database table `t_tematiquequestions`. The selected attribute is `nom`. Moreover during this step we are using the `ID` attribute from the `dim_item` table.

b. Insertion: Information is outputted into the database table `dim_thematic`.

### 3.3.8 Insert Request Time Dimension

In this time dimension we are using the `momentinsert` attribute from the SAR table in order to compute the information about the `year`, `month`, `week`, `day`, `course` and `semester`. This is done by using one script which according to the insertion moment can calculate the `semester` value (1 or 2) and the `course` year. Figure 3.3.8 displays the BPMN representation of the ETL processes from this dimension:
The conceptual operations and processes inside this dimension are:

a. **Extraction:** The `momentinsert` attribute from the SAR table is used to compute the `course,semester,year,month,week,day`. The `real_day` attribute from `calendar_fib` temporary table is being used, reason why we need to create the temporary tables before loading this dimension.

b. **Union:** Information regarding the `course` and `semester` is merged with the time information.

c. **Insert:** After the data is being computed we will insert it into the database table `dim_request_time`.

d. **Additional transformations:** Renaming the attributes to a more understandable format.

### 3.3.9 Create/Update Temporary Tables

The temporary tables are created inside the dimensions group due to the fact that the `Request Time` dimension depends on one of the attributes from the tables. Therefore we are creating the temporary tables before loading the `Request Time` dimension. For inserting data into the temporary tables we are using an (.xls) file that contains information about the calendar university year. Figure 3.3.9 displays the processes that take place inside this task.

**Figure 3.3.9. Create/Update temporary tables**

- **a. Create:** The `calendar_fib` and `class_day` tables are created into the database
- **b. Insert/Update:** Information is being inserted into the two temporary tables

### Fact Tables Group

#### 3. 3.10 Experiments Fact Table

The fact table pulls together information about the different test kits that have been executed, their results, the date when they were executed, etc. It uses information from certain dimension and the SAR table which is later on introduced into the *Experiments* fact table. Figure 3.3.10 displays the ETL processes inside the *Experiments* fact table:
3.3.11 Invalid Fact Table

In this fact table we are gathering information about the exercises, whether the exercises have been performed during class hours or not, date and time, etc. This fact table pulls together information from the dimensions and from the SAR database table, which is later on introduced into the invalid_response_processing table. Figure 3.3.11 better displays the ETL processes inside the Invalid fact table:

3.3.12 Valid Fact Table

This fact table contains information about the valid responses given by the student, whether the answer was given within the class hour interval, if it has been his last attempt, similar exercises to the one performed, date and time, etc. It pulls together information from the dimensions and from other database tables. As we can see in the figure below the input and the output of the updated columns is the same, valid_response_processing table. Moreover, we are also
updating different fields inside the `valid_response_processing` table. Figure 3.3.12 better displays the ETL processes inside `Valid fact table`:

![Diagram of ETL processes in the Valid fact table]

*Figure[3.3.12] Valid Fact Table: Static level*
### 3.3.12.1 Update response_processing

![Diagram](image)

Figure [3.3.13] Update response processing

*Response_processing* table contains different columns which are inherited by the fact tables. Some of these columns are *collisions* and *closed_session*. *Collisions* fields indicate if there are executions that have been coincided in the same interval. *Closed_session* attribute answers the question whether if an exercise has been done during class hours or not. As we can see in the figure above, *closed_session* attribute depends on one of the attributes from the temporary tables. In case the temporary tables have not been updated with information, the *closed_session* update will not be realized either.

### 3.4 BPMN representation: Dynamic level

In order to show the data flow within the activities and to perform a comprehensive monitoring and error handling procedure for the ETL processes, firstly we need to conceptually add this components to our BPMN static representation. Therefore, the next step is to zoom into the dynamic level of the design.

#### 3.4.1 Control Flow

Figure below is a high level representation of the ETL processes.
Compensation activities: Compensation activities are used for performing the rollback operation in case the user decides to undo his actions. As we can see in the figure above, we have attached a compensation activity to each structure where we need to undo certain operations. For example, in case a specific type of error occurs we will need to delete the new insertions and undo the updates. This is one of the reasons why we have previously generated an ETL ID, so we can keep track of the new executions every time we run the flow. Moreover, in the fact tables group we will need to undo the updates that have been performed during the last execution. As we can see for the ‘Create/Update temporary tables’ job, even if an error occurs (i.e.: .xls file is missing) we will continue with the ETL flow. The consequence is that later on the system will not update the closed_session column which is dependent on the existence of the temporary tables. However, this issue will be detailed in the next subsections. In the current stage of the implementation, in case the user decides to undo the operations, the changes will apply to both the dimensions group and fact tables group. As a future work the cancelation event could take place separately for the dimensions and fact tables and the user should have the possibility to choose to undo either the fact tables or the dimension tables, depending on the type of error that has occurred. I have detailed this approach in the future work chapter.
3.4.2 Insert Candidate Dimension

Figure below displays a high level dynamic representation of the ETL processes within the Candidate Dimension.

Figure [3.4.2] Candidate Dimension: Dynamic level

a. Error handling

Due to the fact that the information extracted from the source files might present data quality issues and inconsistencies, there is a need to perform a comprehensive error handling testing procedure especially at this stage. Let us consider the following operations:

**Extraction:** If the attributes that are being selected from the sources are also used to perform the join operation between two data sources, the existence of these attributes within the sources (files or database table) is considered highly important. These attributes that are extracted and which are later on used for the join operation are:

- Files: .TXT file (DNI, username), .XLS(username, userid), .XLS (DNI)
- Database: SAR table (idusuar)
In case any of this attributes is missing we will log the error and compensate the execution of the ETL flow. On the other hand if the (joining) attributes mentioned above exist, but there are other attributes selected from the data sources, which are missing, the solution I have found together with the final users of this application was to fill these missing attributes with null values. For example, a common scenario that is likely to appear is the absence of the `nota` attribute from the (.xls) file. In this case we need to fill the `nota` attribute with null values. Moreover, we will log the error and output it into the database table, `error_logging`. As we will see in Chapter 4, this table contains more details regarding the logged information such as: `date and time`, `error type`, `error message`, etc.

b. Monitor logging

As I have stated previously the error and monitor logging tables will be displayed within the web application interface after the user runs the ETL flow. For a better understandability I have divided the monitoring information into two different tables, w.r.t the type of operation that is being monitored:

- **Extract, Join**: I am monitoring these operations from the `Candidate` dimension in order to allow the user to answer questions like: How many rows were read (extracted) from a specific data source file? How many lines were outputted after a specific join operation was performed? The monitoring details are logged and outputted into the `monitor_logging_extraction` table.
- **Insert**: In some occasions the user might be interested to know how many rows were written to the `dim_candidate` database table. Therefore the monitoring details of this operation are stored into a second table entitled `monitor_logging_insertion`.

### 3.4.2.1 Compute course and semester Information: Dynamic level

One of steps inside the `Candidate` Dimension is to compute the `course` and the `semester`. Although this computation activity appears in other dimensions as well (`Item`, `Request Time`), we will conceptually represent it during this step.

Figure 3.4.3 displays the BPMN representation of the ETL processes within this step:
**Error handling**

As we can see, we are using the `momentinsert` attribute from the SAR table to compute the corresponding `course` and the `semester`. One of the errors that might occur during this step, is the absence of the `momentinsert` attribute. Specifically for the `Candidate` and `Item` dimension, this issue can be solved by filling the missing attributes with null values and continuing the ETL flow. Ultimately, the `dim_candidate` table will contain only information regarding the students, but no information about the `course`/`semester`. This scenario has been discussed with the final users of the system and is considered acceptable. I would emphasize again the utility of using a conceptual BPMN representation for the ETL processes, since the operation presented above is a specific example of how this conceptual representation can help in taking safe decisions which can satisfy the requirements of the final users.

### 3.4.3 Insert Delivery System Dimension

Figure 3.4.3 displays the BPMN dynamic representation of the ETL processes within this dimension:

![BPMN diagram](image)

**Figure [3.4.4] Delivery system Dimension: Dynamic level**

**a. Error handling**

**Extract:** In some cases the `moodle` attribute can be missing from the SAR table. As a result we will need to cancel the process. This is due to the fact that the fact tables depend on some of the attributes from this dimension. In case the attribute exists we will continue with the ETL flow and rename it to a new identifier.

**b. Monitor logging**

**Insert:** The monitoring details regarding this operation are stored into the `monitor_logging_insertion` table.
3.4.4 Insert Experiment Dimension

Figure 3.4.4 displays the BPMN dynamic representation of the ETL processes within this dimension:

![BPMN diagram](image)

**Figure 3.4.5.** Experiment Dimension: Dynamic level

a. **Error handling**

*Extraction:* In case the selected information from the t_jocsproves table are missing, we will use a cancelation event.

b. **Monitor logging**

*Insert:* The user might be interested to know how many rows were written to the database table dim_experiments. Due to this reason the monitoring details regarding this operation are stored into the monitor_logging_insertion table.

3.4.5 Insert Item Dimension

Figure 3.4.6 displays the BPMN dynamic representation of the ETL processes within this dimension. The error handling and monitor logging procedures are performed similarly as for Experiment dimension. We can notice that during the execution of this dimension we are also computing the course/semester values, by using the same logic as presented before in figure 3.4.2.1. As in the other case, we can continue the ETL flow even if the moment_insert attribute is missing from the SAR table. The union step is being performed between the information about exercises and course/semester data.
3.4.6 Insert Thematic Dimension

Figure 3.4.7 displays the BPMN dynamic representation of the ETL processes within this dimension. The error handling and monitor logging procedures are performed similarly as for Delivery System dimension.

3.4.7 Insert Request Time Dimension

Figure below displays the BPMN dynamic representation of the ETL processes within this dimension:
Figure[3.4.8] Request Time Dimension: Dynamic level

**a. Error handling**

This time dimension is dependent on the existence of the `momentinsert` attribute from the SAR table. Therefore, while computing the `year, month, week, day` values by using this attribute, in case an error occurs we should first check if the attribute is present in the SAR table. In case it is not we will cancel the process, or contrary we will continue the ETL flow if the attribute exists. On the other hand, this operation is not entirely depended on the temporary table, `calendar_fib`. To better explain this issue, since we have already created the temporary tables before the load of the Request Time Dimension we are sure at this step that these tables exist. However, in case the temporary tables exist but are not updated with information (i.e.: `.xls` file is missing), the execution of this computation will not be stopped. Furthermore, the other computation activity that we see in the figure above (getting course/semester values) is in the same scenario as the previously explained case. If the `momentinsert` attribute is missing, the execution is either ways going to be canceled (while computing the `year, month, week, etc.`), so there is no need use another cancelation event.

**b. Monitor logging**

The information inserted into the database table `dim_request_time` is monitored and the details are stored into the `monitor_logging_insertion` table.

### 3.4.8 Create/Update temporary tables

Figure below displays the BPMN dynamic representation of the ETL processes within this task:
a. Error handling

**Create:** One of the main causes of error in the initial design of the ETL was the fact that the tables were only dropped at the end of the ETL flow. Due to this issue, in case the execution was failing before the tables were dropped and the ETL flow was executed again, the debug panel of the tool would display a different error message such as ‘**tables already exist in the database**’. There is a need to check if tables already exist in the database and in case they exist we should drop them during this phase.

**Insert/Update:** It is important to mention that after the discussion I had with the final users of the ETL application, they have implied the fact that the .xls file which contains information about the calendar university year might not always be available. Therefore, while the tester uses the web application to input the source files, the system should continue working even if this file is not uploaded. This will affect the Request Time dimension and **Closed_Session** column from the **response_processing** table, but the rest of the ETL processes should continue. However, a flag will be raised to pass an ‘alarm’ that will be caught later on.

b. Monitor logging

The information inserted into the temporary tables is monitored and the details are stored into the **monitor_logging_insertion** table.

**Fact Tables group: Dynamic level**

3.4.9 Experiments Fact table

Figure below displays the BPMN dynamic representation of the ETL processes within the **Experiments** fact table:
3.4.10 Experiments Fact table: Dynamic level

a. Error handling

Select: One of the main issues that is causing the ETL to fail is the absence of certain attributes from the dimensions, which are used as foreign keys while loading the fact tables. As we can see in the figure above we are pulling together information both from dimensions and some of the database tables. In case the select operation is failing the consequence is to log the error and cancel the process.

Insert: While performing the error handling procedure we should check if the foreign key attributes that are filtered are null or not. In case an attribute is null, the solution is to record the missing attribute and end the process. On the other hand if the attribute is not null we will continue with the insert operation. If an issue occurs during the insertion stage, the error will be logged and stored into the error_logging table. This scenario will be replayed for all the filtered foreign key attributes. (Multi Instance sequential loop).

b. Monitor logging

The information inserted into the fact tables is monitored and the details are stored into the monitor_logging_insertion table.

3.4.10 Invalid Fact Table

Figure below displays the BPMN dynamic representation of the ETL processes within the Invalid fact table. The error handling and monitor logging procedures are performed similarly as for Experiment fact table, therefore I will not go again into the details of the main operations. (Select, Insert)
3.4.11 Valid Fact table

Figure 3.4.12 displays the BPMN dynamic representation of the ETL processes within the Valid fact table. The error handling and monitor logging procedures are performed similarly as for the other two fact table. This is due to the fact that the most common type of error that we are encountering during this stage is the absence of one or more attributes from the dimensions which are used as foreign keys in the fact tables. As we can see in the figure below, the input and the output of the updated columns is the same, valid_response_processing table. Moreover, while updating the columns we are using information from other data source tables as well, i.e: dim_item, dim_tematiquequestiones, etc. I will shortly enumerate the columns that are being updated while loading this fact table: succeeded experiments, conducted experiments (total number of experiments that have been conducted), Outcome (succeeded experiments/conducted_experiments), LastTrial (whether the last trial corresponds to the latest attempt carried out by the candidate), PreviousTrial (nr. of previous attempts of the candidate for each exercise), PreviousRelatedItems (nr. of previous attempts related to the exercises’ themes, TriedAllThematics( whether the student has tested or not all the topics associated with the exercise), FirstTrialOutcome, PreviousTrialOutcome.
Figure [3.4.12] Valid Fact Table: Dynamic level
3.4.11.1 Update Response processing

![Diagram](attachment:image.png)

**Figure [3.4.12] Update response processing: Dynamic level**

**a. Error handling**

**Update:** As stated before, collisions and closed_session fields are part of the response_processing table. As we can see in figure above the update operation for the closed_session field is dependent on the information from the temporary tables, reason why initially we are checking the flag which has been raised during the ‘Create/Update temporary tables’ sub-job. In case the temporary tables exist but have not been updated, the closed_session field will not be updated either. On the other hand, in case the temporary tables have been updated successfully, but there is any other error occurring during the update operation, the process will be canceled. This scenario applies for the collisions field as well.

**a. Monitor logging**

During this step we are interested if the update operation took place (start date, duration, etc) and how many rows have been updated. The information will be stored in the monitor_logging_insertion table.
Chapter 4. ETL Tools

One of the goals of this project was to find an alternative ETL tool that can better answer the new design improvements performed during this project. The initial ETL design was modeled using Penatho Kettle, but due to the fact that I have made several changes and included additional functionalities, I had to take into consideration redoing the entire design by choosing an ETL tool that can better solve these new challenges. Therefore in this chapter I am going to present the steps that I have performed in choosing the ETL tool. Appendix B contains the detailed description of the steps that I have conducted during this step.

I took into consideration three of the most popular ETL tools existing on the market: Pentaho Kettle, Talend Open Studio and Informatica PowerCenter. The comparison between these three tools was done by applying different criteria specific for our project:

a. **Connectivity:** Ability to connect to the existing data sources (external files, PostgreSQL database)

b. **Easy-of-Use:** Overall difficulty in performing different actions and solving certain requirements, such as: debugging, monitoring, error handling procedures, integration, etc.

c. **Debugging:** Although it is expected that all the selected tools are offering debugging capabilities, still it is interesting to check this feature in terms of understandability (*How easy is to deduce the type of error which is displayed in the debugging panel?*) and in terms of functionalities (Possibility to edit the unstructured error message displayed in the debugging panel of the tool and insert this message in the corresponding error_logging table field, i.e.: *type of error*)

d. **Monitoring/Error Handling:** As stated before one of the main requirements of this thesis project was to redesign the initial ETL flow and include several monitoring and error handling components. Due to this reason I had to perform a comprehensive check among the selected ETL tools w.r.t the implementation of these new functionalities. Although nowadays most of the ETL tools are offering basic error handling capabilities, it is interesting to check which of the selected tools can better answer the requirements of this specific project.

e. **Integration Capabilities:** Testing the possibility to deploy and the easiness to integrate the ETL job within a web application.

In the next sections of this chapter I will directly present the tools comparison table and I will justify my choice for choosing one ETL tool to implement the new design, out of the three choices mentioned above. The tools comparison table is based on the information gathered in Appendix B.
4.1 COMPARISON BETWEEN TOOLS

The selected ETL tools should be compared w.r.t to different criteria specific for our project. As we can see in the table below, the ETL tools are very close in terms of the functionalities that they are being offered. However, the requirements of our project state the fact that the tools have to support comprehensive debugging, error handling and monitoring procedures. The tool has to allow the developer to implement the needed functionalities, in order to be consistent with the BPMN representation. Moreover, the integration of the ETL job within a web application is another important aspect that should be taken into account. All the three tools offer integration capabilities, therefore regarding this factor the difference is made in details such as: easiness to perform the integration, cost, etc. Table below presents a comparison between the ETL tools based on the information displayed in Appendix B.

<table>
<thead>
<tr>
<th>Tool/Feature</th>
<th>Pentaho Kettle</th>
<th>Talend Open Studio</th>
<th>Informatica</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connectivity</td>
<td>Database connection, i.e.: PostgreSQL</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>Additional data sources connections, external files (.txt,.xml,.xls)</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Ease of Use</td>
<td>Difficulty in performing ETL operations (Extract, Join, Insert)</td>
<td>Ease</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Difficulty in performing the Debugging/Monitoring/Error Handling</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Difficulty in integrating the ETL job within an external web application</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Code-driven approach vs Metadata driven approach</td>
<td>Metadata</td>
<td>Code driven (based on a Java platform)</td>
</tr>
<tr>
<td></td>
<td>Overall usage difficulty of the tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debugging</td>
<td>Main Debugging Interface</td>
<td>Execution Results</td>
<td>Debug mode</td>
</tr>
<tr>
<td></td>
<td>Understandability of the error messages displayed in the debugging panel of the tool</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Debugging functionalities</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Adding breakpoints (step-by-step execution) & Editing the debug message & Pausing the debugger -set conditions- & Using code driven approach to perform complex tasks i.e.: hardcoding the error type in the logging table field.

**Monitoring**

- Possibility to output the monitor logging results to an external target table from our DB
- Monitoring Metrics existence **Yes/No**
- Step-by-Step monitoring execution
- Performance monitor
- Statistics Monitor

**Error Handling**

- Monitor logging components existence
- Implementing Per/Row error handling possibility
- Edit error message
- Use external libraries that can improve the error handling

<table>
<thead>
<tr>
<th><strong>Integration capabilities</strong></th>
<th>Plugin/Tool/Feature used for performing the integration or deployment</th>
<th>Cost/Pricing</th>
<th>Data Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pentaho Carte</td>
<td>Talend ESB</td>
<td>Informatica Cloud</td>
</tr>
<tr>
<td></td>
<td>Free version available</td>
<td>Free Version available</td>
<td>Requires a regular subscription</td>
</tr>
<tr>
<td></td>
<td>Data Cleaner</td>
<td><strong>Talend Open Studio for DataQuality</strong></td>
<td><strong>Informatica Data Quality</strong></td>
</tr>
</tbody>
</table>

*Table [11]. Tools comparison*
4.2 Tool selection justification

After looking at the capabilities offered by each ETL tool, my choice has been to use Talend OS for redesigning the ETL flow. Even though it might be more difficult to use due to the possibility of having to write Java code while performing certain tasks, the advantage is that Talend provides more flexibility in answering the requirements of this thesis project, in terms of error handling, monitoring, debugging and integration. As an example, the code driven approach of Talend OS (which indeed can be considered a disadvantage from the *easiness of use* perspective), has helped me to easily perform specific tasks, such as hard-coding the error message displayed in the debug panel or checking for missing attributes in the data sources. Throughout the implementation stage of the ETL, I have realized that using Talend OS has been the right choice. Although most of the *’drag and drop’* components that are offered by Talend for solving data integration challenges, have a correspondence in the other ETL tools as well, the code driven approach of Talend OS makes the tool harder to use (*learning curve is steep*) but in the same time very powerful to use in terms of functionalities. We can also take into consideration the following aspects:

a. **Debugging/Monitoring/Error Handling**: Although both *Pentaho Kettle* and *Informatica* offer solutions to for this functionalities, the difference is in terms of debugging understandability. The errors messages from the debugging panel of *Kettle* and *Informatica* are hard to be deciphered, since they are unstructured and in some cases unreadable. On the other hand the possibility to hard-code the error message from the debugging panel of the tool to a specific string, i.e.: *’Missing file’, ‘Missing attribute’,etc.* , has proved to be much more harder to implement than by using Talend OS.

b. **Integration**: The tool has to allow an easy integration of the ETL job within a web interface. Based on the research that I have done, all the three tools offer a solution for this integration. In the current stage of this project these solutions prove to be enough, but for the future we might consider using cloud based services for the ETL job. To my knowledge, *Pentaho Kettle* and *Talend OS* are open source tools and do not require additional costs for this task, whereas *Informatica Cloud* requires a certain subscription price for this service. However, this issue has not been a differentiation factor since all the three tools can accomplish the integration requirements in the current state of the project.

c. **Ease of use**: This feature is important in order to ease the work of the ETL developer, but it is not a primarily differentiation factor. Therefore, after testing the three ETL tools, in my opinion *Pentaho Kettle* and *Informatica* are easier to use than the third tool. Talend OS uses a code-drive approach, since it is build on a Java platform. This means that for performing certain advanced tasks there is a need for writing or generating Java code.
Chapter 5. ETL Design using Talend OS

As I have previously stated Talend Open Studio is an Eclipse-based graphical environment that allows us to perform data integration tasks. Throughout the ETL re-designing phase, the functionalities provided by this tool proved to be sufficient in answering the main requirements of our project. While redesigning the ETL processes I have added several functionalities in order to perform a comprehensive monitoring and error handling procedure. It is important to state again that the purpose of using a BPMN representation for the ETL processes was to get a deeper insight into the details of the system and to ensure a good level of compatibility between the product owner’s requirements and the actual implementation. Therefore, while re-designing the system using Talend OS I have tried to follow the same conceptual steps which were previously represented in the BPMN diagrams. In the following subchapters I will present the levels of the ETL implementation, starting from a higher level and ending with a more detailed description of the monitoring and error logging procedures inside the dimensions and fact tables.

5.1 Control Flow
At a conceptual level the control flow is structured into three major blocks. From a theoretical perspective this structures refer to the conceptual order in which the ETL flow is executed.

a. Prerequisites: In this initial phase we are creating the tables in the database, which will be used to store the logging information. These logging tables need to be created before the execution of the dimensions and fact tables, since their successful execution is conditioned by the existence of these tables. Moreover, we can conceptually include in this phase the generation of the ETL ID. This activity task comes after the creation of the logging tables due to the fact that we first need to create an ID column into the logging tables to store the corresponding value. Although it is obvious that these two jobs (Creation of the logging tables and ETL ID generation) have different purposes, at a purely conceptual level I have included them both into the prerequisites block.

b. Dimensions Group: As we can see in the BPMN representation of the control flow, an important phase of our design is to parallelize the dimension jobs. In order to be consistent with the BPMN design, inside the group I have represented the load of the dimensions in parallel. Due to the fact that the Request Time Dimension is dependent of the information gathered from the temporary tables, the dimensions group also contains a sub-job entitled ‘Create/Update Temporary tables’.

c. Fact Tables Group: The last step of our ETL design is to create the fact tables and load them with data information gathered from the dimensions. This group contains the parallel execution of the existing fact tables.
Figure bellow presents a higher level insight into the main jobs of the ETL flow.

![Diagram of ETL processes](image)

*Figure [5.1.1]. High level flow of the ETL processes*

From a practical perspective, in Talend OS the groups presented above can be technically defined as main jobs. In the following paragraphs I am going to zoom in into each main job, while emphasizing the consistency between the ETL design and the BPMN conceptual representation.

### 5.2 Create Logging Tables

This main job contains the creation of the monitoring and error logging tables. Moreover, during this step I am also including an ETL ID column into the logging tables, dimension tables and fact tables.

#### 5.2.1 Monitoring tables

Although initially I have created one single table to store the details about the operation which is monitored, later on I have decided to split the information and include it into two separate monitoring tables. This decision has been based on the fact that there is a need to differentiate the records that are being monitored. For example some of the records can refer to monitoring the dimension insertion step (i.e. *How many lines were written into a specific dimension/fact table?*), while other monitored records can refer to the extract or join operations (i.e. *How many lines were read from a specific data source file? How many lines were read after performing the join step between two data sources?*). Bellow I will present a more detailed description of these two tables:

a. **Monitor_logging_insertion**: In this table we store the monitoring parameters which are generated after each *insert* operation either from the dimensions or fact tables. These parameters can refer to the *start date, start time, end time, duration time* of the execution and the number of *lines written* to the database table. Figure 6.6.1 displayed in the next chapter presents a first insight into the interface and the main fields of this table. However, below I will enumerate the main attributes which are stored into the table:

- **ETL ID**: In this column we store the ID of each execution. Each time the ETL flow is executed, this *ID* will be incremented for each group of logged records. A more detailed description about this attribute will follow up in the next section.
- **StartDate**: It contains the date/time information when the monitored step was executed.
- **StartTime/EndTime**: These attributes generate information regarding the starting time and ending time of the monitored operation.
- **Duration**: Represents the time period in between starting and ending the job. In some specific scenarios the user might be interested to monitor the duration of the job execution for further performance analysis. For including the duration attribute I have used the `tChronometerStart` and `tChronometerEnd` components.
- **Lines_write**: One of the factors that the user might be interested to see is the number of rows that have been inserted into the database table. Therefore this attribute contains the number of rows written to the table.

It is important to mention that some other attributes were included in the `monitor_logging_insertion` database table, with the specific purpose of offering more useful details to the user. This descriptive attributes are exemplified bellow:

- **PID**: When executing the ETL flow each monitored step receives a particular PID key, which can later on be used for differentiating the monitored records. Although this attribute along with the other attributes have been stored in the PostgreSQL database, it will not be displayed in the web application interface. The reason behind this choice is that for the specific purpose of our project I do not consider it necessary to display the PID information in the UI.
- **Project**: In this column we can store the name of the project that includes the ETL flow.
- **Job**: It represents the name of the job that has been monitored. For example if we are monitoring the insertion step from the Item dimension, the name of the job will be `insert_dim_item`.

b. **Monitor_logging_extraction**: In this table we store the information monitored after the *extraction* and *join* operations. Figure 6.6.2 displayed in the next chapter presents a first look into this table. The `monitor_logging_extraction` table contains the following attributes:

- **ETL ID**: In this column we will store the ID of each execution.
- **Project, Job, Start Date**: The purpose of using this descriptive attributes is the same as for the previous table
- **Component Origin**: It presents the name of the ETL component that is being monitored. This attribute is useful for the user to better visualize in the application interface the step that has been monitored and tracked.
- **Label**: This attribute displays the label of the data source that has been monitored. For example if we are extracting data from the (.xls) file the label value will be `.xls`. On the other hand if we are monitoring the *join* operation between the .txt and .xml files, the *label* field will have the name of the (joining ), i.e.: `[username_join]`.
- **Count**: One of the interesting factors that the user might be interested to see, is the number of rows read after the monitored operation is performed. In this way the user can
answer questions such as: *How many rows were extracted from the (.txt) file? How many rows were read after the join operation was performed between two specific data source?*

### 5.2.2 Error_logging table

This table contains information about the error handling procedures that I included in the design. As we can see in the BPMN diagrams, there are several errors that can occur and badly influence the successful execution of the ETL flow. Figure 6.6.3 displayed in the next chapter presents a first look of this table’s interface. The attributes that I have considered useful to display when performing the error handling procedure are the following:

- **ETL ID, Project, Job, StartDate**: The purpose of using these attributes is the same as for the monitoring tables.
- **Error Type**: In most of the cases when the system generates an error, the user might be interested to see more specific details w.r.t the type of error that has occurred, not just an unstructured message that can be hard to decipher. After taking into consideration the errors that can occur throughout the execution of the ETL flow, I have hard-coded this attribute to receive specific values depending where and why the error has occurred. Therefore this attribute can display string values such as: “*Dimension attribute does not exist*”, “*File Missing*”, “*Missing attribute from .txt*”, “*Missing attribute from .xls*”
- **Component origin**: By using the information from this column we can offer a more detailed information about the error type and its occurring reason. The information displayed by in this attribute is related to the error type information. For example one of the errors that might occur is the absence of the *nota* value in the (.xls) data source file. In this case while the error type will be “*Missing attribute from .xls*”, the component origin will be labeled as [‘*nota*’]. On the other hand if the error that occurs is related to the absence of the calendar year information file, the component origin will display the string [‘Checking if the .xls file exists’]. Component origin is the specific component from the design that performs the operation.
- **Message**: As stated before, when the ETL tool generates an error, the message that is being displayed in the debugging interface is unstructured and hard to decipher. Therefore this attribute has been hard-coded as well so that whenever a different type of error occurs, the string values could be sentences such as : [‘*Attribute is missing from the data source*’].

For creating the logging tables using Talend OS I have used the **tPostgreSQLInput component**. The major function of this component is to execute an inputted SQL query and create new tables into the database.
5.3 ETL ID Generation

Before zooming into this job we have to mention some of the reasons why there is a need to generate an ETL ID.

- **Understandability**: If we are running the ETL flow many times it can be hard to differentiate the newly inserted records from the old records. Therefore there is a need to structure the monitoring and error logging executions, by generating an incremental ID for each execution.

- **Functionality**: One of the interesting features that can be included in our system is to allow the user to perform an *Undo/Rollback* operation. In order to implement this task we first need to differentiate the ETL executions.

Using Talend OS I have managed to design this step by choosing the maximum value from the ID column of the logging tables, dimensions and fact tables. The next step is to increment it with one (+1) every time the ETL is executed.

As we can see in Figure 5.3.1, I have used three different components for performing the execution of this task.

- **tFileInput**: This component passes the database connection details to the ETL_LoadID. One of the functionalities that this system provides to the user, is the ability to upload his database connection details from the web application interface. The main usability of this component is to pass the database connection details.

- **tContextLoad**: The usage of this component is related to the previous one. I have used it to dynamically modify the values of the incoming flow, into two separate columns (key and value). The key attribute represents the name of the DB parameters (*server, host, login, password*) while the value information is provided by the user for each parameter.

- **ETL_LOAD**: This *tPostgreSQLInput* component is used to generate and increment the ETL ID for the tables. This task is performed by using a query that selects the maximum value of the existing ID in the logging table and increments this value with (+1) after each execution.

![Figure 5.3.1: ETL ID generation](image-url)
Dimensions Group
This main job of our ETL design contains the parallel execution and insertion of data into the dimension tables. A parallel execution of these jobs is modelled in order to ensure a consistent design with the BPMN representation. Using Talend OS there are two main ways for creating a parallel execution of the sub-jobs:

- **Using tParalelize component**: The best way to realize a parallel execution design is to use this component. The main issue comes from the fact that its usage is only available in the commercial version of the tool. I have exemplified in Appendix .A how this component can be used and implemented for future improvements.
- **Selecting the Multi Thread execution tab**: This simplistic approach managed to solve our problem for designing the dimensions in parallel. Although there are not too many settings that can be applied to the dimensions execution order, this feature proved to be sufficient enough for solving the ETL-BPMN consistency requirements.

Figure 5.3.2 presents the design of the Dimension Group. In order to be consistent with the BPMN representation, the job *Create/Update temporary tables* is linked with the dimension *Request Time* and therefore is part of the Dimensions Group.

![Figure 5.3.2. Dimension Group](image)

5.4 Monitoring procedure design
In this section I will zoom into the implementation of the components which were used for monitoring certain processes, during the execution of the ETL. Since the processes that are monitored might be different in some dimensions and fact tables, I will only present the designs which I considered to be more special.
Figure below displays the ETL components that are used for performing the monitoring procedure in the *Candidate* Dimension. I have chosen to display the design for this dimension because it contains references to all the three logging tables.

![Diagram of ETL components](image)

*Figure [5.4.1] Candidate Dimension monitoring procedure*

The components used in the design have the following functions:

a. tFlowMeterCather (Catch Flow): This component is used to catch the flow outputted by the tFlowMeter component. Each monitored process is followed by a tFlowMeter that is capturing certain details such as: number of rows extracted, number of rows written, etc. The tFlowMeter component is not shown in the figure above but is present throughout the design of dimensions and fact tables. Therefore, we are using tFlowMeterCather to catch the information flow that was outputted before.

b. tBufferOutput (Buffer Output 1): During this step we are storing the information in a buffer until the execution flow of processes inside *candidate* dimension is ended.

c. tBufferInput: In this buffer we are preparing the monitored information that will be used by the corresponding mapping components.
d. **tMap**: It is used to map the information from the input to the output table. As we can see the first `tMap` component connects the information with the `monitor_logging_insertion` table, while the second `tMap` component connects the information with the `monitor_logging_extraction` table.

e. **tFilterRow** (Filter the insert operations): Due to the fact that `tFlowMetter` component monitors the `extract, join` and `insert` operations, we need to filter the `insert` information which is mapped and stored into the `monitor_logging_insertion` table.

f. **tUniqueRow** (Select the attributes that are mapped): During this step we are choosing the fields outputted by the mapping component, which will be introduced into the corresponding database table. For example the `monitor_logging_extraction` table does not contain descriptive information regarding the `start date/end date`, such as the other monitoring table, reason why we need to choose which fields will be displayed in the table.

For monitoring the operations from other dimensions and the fact tables we have used a slightly different design. The ETL components displayed in figure 5.4.2 applies to other tables as well. As we can see, the logical order of the processes is very similar with the previous design, reason why I will not go into the details of each component. The main difference is that in this case we are not monitoring the `extract` or `join` operations, only the insertions.

![Figure 5.4.2 Dimensions monitored information](image)

The logical order for implementing the monitoring procedure is the following: 

- Catch the monitored flow of the corresponding operation
- Store it into a buffer until the process flow ends (i.e: In order to obtain the `duration` time we need to wait until the sub-job’s execution has finished)
- Map the information to the database table (select the parameters which will be included in the table, i.e.: `duration, start date`, etc.)
- Insert information into the corresponding logging table.
5.5 Error handling design

In this section I will zoom into the implementation of the components that are normally generating errors during the execution of the ETL.

At a general level in order to catch the errors that occur during the execution of the dimensions and fact tables I am using the structure displayed in the figure below.

![Figure 5.5.1. Error handling: Catching procedure](image)

- \textbf{tLogCather}(CatchError): In case there are any errors occurring during the execution of the ETL this component will catch that error
- \textbf{tMap}(Map error logs to DB output): The error information parameters are being mapped to the output table
- \textbf{tPostgreSqlOutput}(error_logging table): During the execution of this step the logged error information is loaded into the \textit{error_logging} table

5.5.1 Create/Update temporary Tables

As stated before each time we execute the ETL there are two temporary tables that should be created, calendar\_fib and class\_day. Normally, we should extract data from the calendar university year (.xls) file and insert it into the two tables.

\textbf{a. Create Temporary tables}

The table creation is done with the help of a \texttt{tPostgreSqlRow} (Create tables) component which executes the necessary scripts for creating these tables.

\textbf{b. Drop temporary tables}

One of the issues that was previously encountered, was the fact that after the temporary tables were created they were dropped only at the end of the ETL flow. Due to this reason, when running the ETL jobs individually, the system was constantly throwing errors, such as “\textit{Tables already exist in the database}”. One of the special features that Talend allows us to do, is to use a special action when creating a table and select the \texttt{Drop if tables already exist}. This feature has allowed us to eliminate the issue which in the previous design was generating this type of error.

\textbf{c. Update temporary tables:}

After we have created the temporary tables we need to update them with information coming from the calendar university year (.xls) file. We should take into account the fact that this file
might not be always available. Therefore, when performing the error handling procedure we have to consider this problem. The components that are used when modelling this technique are displayed in Figure 5.5.2.

### 5.5.1.1 Missing file for updating the temporary tables

As stated before one of the causes the system has been failing in the past is the absence of the data source (.xls) file. The solution I have found together with the final users of the application, was to log the errors and continue the ETL flow, in case this file is missing.

![Figure 5.5.2. Check for missing file](image)

The components that are used in the design are:

- **tFileExist (Check if file exists):** This component is used to see if the (.xls) file has been uploaded or not by the user in the web application’s interface.

- **tRowGenerator (Generate input row):** This component performs no explicit action but is used to generate an input for the **tMap**. It is important to remind again that the **tMap** needs special types of components for the input and for the output.

- **tMap (Map input information to output table):** Connecting the input information with the output table. During this step I am also hard-coding the message that the logging error should display in the database table. For example in case the file is missing the **error type** would be hard-coded to ['Missing File'].

- **tPostgreSQLOutput (error_logs):** During this step I am outputting the error information into the **error_logging table**
5.5.2 Missing attributes from data sources
While checking for the missing attributes inside the data sources I have used the following logic. Let us consider for example that we want to see if the (.txt) file containing student information, has the selected attributes or not:

- Use a tFileInputDelimited component to input the .txt file
- Break down this .txt file by using another tFileInputDelimited component where we individually store the attributes from the previous inputted .txt file
- Use a tMap component to check if the DNI and username fields are found within the attributes
- Use another tMap component to select the error logging fields which will be outputted in the error_logging table. Also during this step hard-code the message that should be displayed in the logging table, i.e.: ‘Missing attribute from the .txt file’
- Insert information into the error_logging table.

5.5.3 Fact Tables error handling
The operations performed inside the fact tables are highly depended on the dimension tables that have been created. Below I will describe the main fact tables that are created in our system together with the catch of the error that might occur during the execution.

In order to extract information from the dimensions and load it into the fact tables I have used the tPostgreSQLRow component for executing the scripts. The error that might occur during the load is fetched during the error handling ETL structure at the bottom of the figure and inserted into the error_logging database table.

5.5.4 ETL Rollback action (Updates)
In the current state of the implementation the user can undo only the insertions from the database by pressing the Undo button. Rolling back the update operations is not fully functional, therefore
the feature has not been fully integrated into the system. However, I will detail bellow the steps which I have performed until this moment into implementing this functionality. I believe that the steps presented below can offer an inspiration for the person who continues the implementation for this project.

The component provided by Talend to perform the commit and rollback actions is called tSqlConnection. While using this component the new data information will not be automatically committed to the DB, unless we select the Auto Commit action found on the Advanced settings tab. On the other hand we can choose to roll back the update operation in case some specific errors occur by using the tSqlRollback connection. Figure 5.5.4 presents the logical flow of this operation.

Let us consider for example the scenario in which we need to insert or roll back the update operations. In this case the error that might generate such a scenario would be the absence of one of the dimension attributes which is used as a foreign key inside the fact tables.

Although the implementation is not fully functional, I have presented in the figure below how this scenario can be tackled by using the functionalities provided in Talend. The design contains several components such as:

a. tLogCather (Catch error): In case any error occurs this component has the function to catch the error

b. tFilterRow (Filter by Null)

c. tSqlConnection (Check if attribute is null): This component is used to check if the attribute selected is null or not.

d. tSqlCommit (Insert data into the fact table): Depending on the output of the previous step (On Component Ok), it performs the insertion to the fact table

e. tSqlRollback (Rollback if error): Depending on the output of the previous step (On Component Error)

f. tPostgreSQL_Output (error_logs): In case we roll back the process, it is indicated to record the error into the error_logging table so that we can later on display it in our web application interface.
As a future work this functionality can be included into the web application and allow the user to undo his actions from the main user interface.

Figure [5.5.5] Rollback/Insert
Chapter 6. ETL analyzer Web application

6.1 Requirement specification
Requirement analysis is the first phase in designing the ETL analyzer web application. In this phase, we set the initial standards and conditions that the web application has to satisfy. At a high level the main requirement for the web application is to allow the user to easily execute the ETL job and see the results of the logging tables inside the front end interface.

6.2 Purpose
The ETL Analyzer is a web application that is provided to the teachers from UPC who are teaching the ‘Database’ class, in order to interactively analyze and test the information generated by the ETL job. Therefore we can assume that the final users of the application are the university teachers or university administrators. The application should allow the user to input the necessary data sources, database connection parameters and Run/Undo the ETL. As a result it should display information about the monitored operations or the errors that might have occurred during the execution of the ETL.

The implementation of the web application has been entirely performed during the development of this thesis project.

6.3 Functional Requirements
During this sub chapter I will detail the main functional requirements that the UI of the application has to respect.

6.3.1 Main User Interface

a. Choose Data Source Files: The user should be allowed to input the corresponding data source file, by clicking on the Choose File button. We are assuming that the files regarding the student information are always available, although the information inside these sources might not always be complete. On the other hand, the user has the option to input/or not to input the other data source file containing the calendar university year information. In case the user will not input any file the system should continue running.

b. Upload Data Source Files: The user should be allowed to upload the inputted source files by clicking on the Upload button. (Files will be uploaded to a specific folder and are automatically fetched by the ETL components that using this information)

b. Input DB connection parameters: The UI should allow the user to add the connection parameters of the existing database. These parameters refer to the following details: DB name, DB schema, localhost, username, password, and will be fetched by the ETL components that are using the connection to the database.
c. Run the ETL flow: The user should be able to run the ETL from the main user interface of the application, after the above steps have been accomplished, by clicking on the Run ETL button.

d. Undo the ETL flow: The user should have the possibility to undo the main operations after an ETL execution, by clicking on the Undo button

Figure 6.3.1 displays the functionalities of the main interface.

![ETL Analyzer Interface](image)

*Figure [6.3.1] Main User Interface*

The main technologies that have been used to design the user interface were HTML and PHP.

- **HTML**: For creating the front end interface of the application, starting from the information organization, page style, buttons, etc.

- **PHP**: I have used PHP for integrating the ETL executable file that is generated by Talend inside the web application. Moreover when uploading the source files from the web application into a specific folder, I have used certain (.php) scripts that can perform this action.

- **Bootstrapping techniques**: Bootstrapping is a combination of CSS, HTML and Javascript code which is designed to build user interface components. I have used this technique for a better organization and style of the logging tables.
6.3.2 Output interface

a. Monitoring and error logging tables: After running the ETL flow, the web application should display information regarding the three logging tables.

b. Additional functionalities of the logging tables:
   - **Search Box**: The interface should have a Search box, which can give the user a better flexibility in finding specific information. For example if the user is interested to search only relevant information w.r.t the insert operation, he will have to type the “insert” message into the Search box.
   - **Sorting the information**: The UI should allow the user to sort the information inside the tables (i.e.: according to the ETL Load Id). For example the user should be able to check for the newest or the oldest records inside the logging tables.

6.4 Non-functional Requirements

The table below describes the non-functional requirements specific for our system.

<table>
<thead>
<tr>
<th>Name</th>
<th>Requirement</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>The application should be available 99.9% of the time</td>
<td>-The user should be able to access the web application from any place.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-The system must be available 24/7 with no downtime</td>
</tr>
<tr>
<td>Supportability</td>
<td>The webpage should be supported by different browsers</td>
<td>-The system must be tested on different browsers before the application is published.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-The user should not experience different displays or non-compliance structure for the application when using different browsers.</td>
</tr>
<tr>
<td>Extensibility</td>
<td>The system should be easily extendable</td>
<td>-The system should be designed in a way that it should be easy to add new functionalities.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-The source code of the system should be reusable.</td>
</tr>
<tr>
<td>User friendly</td>
<td>The system should be responsive to user queries</td>
<td>-The user should be able to navigate through the web application easily and intuitively</td>
</tr>
</tbody>
</table>

Table [13]. Non Functional requirements
6.5 Use cases

Use case #1 (RUN ETL- All the data sources are available)

**Main actor:** User (UPC administrator)

**Precondition:** The data source files exist and are available for usage. The database already contains all the required tables used for outputting information (logging tables, dimensions and fact tables).

**Main success scenario:**

a. User upload the data source files (.xls, .xml, .txt) by clicking on the corresponding ‘upload’ button

b. User uploads the calendar information file (.xls) by clicking on the corresponding ‘upload’ button

c. User inserts the DB connection details into a file and uploads the file into the system by clicking on the corresponding upload button

d. User clicks on the RUN ETL button

**Result:** The web application will display the three monitor/error logging tables containing the logged information about the execution.

Use case #2 (RUN ETL- Not all the data sources are available)

**Main actor:** User (UPC administrator)

**Preconditions:** The data source files regarding the student information exist and are available for usage. Calendar university year (.xls file) information is not being inputted. The database already contains all the required tables used for outputting information (logging tables, dimensions and fact tables).

**Main scenario:**

a. User selects the data source files (.xls, .xml, .txt) by clicking on the corresponding ‘upload’ button

b. User inserts the DB connection details into a file and uploads the file into the system

c. User clicks on the RUN ETL button

**Result:** The web application will display three monitor/error logging tables with the needed information about the ETL execution. The error_logging table should have a record showing that the information regarding the calendar university year is missing.

Use case #3 (Run ETL-Database connection issues/Database missing tables issues)

**Main actor:** User (UPC administrator)

**Preconditions:** Data source files are uploaded successfully but the database connection parameters are incorrect, i.e.: the database password or login is incorrect. Moreover, we also
assume that the database does not contain all the required tables used for outputting information (dimensions and fact tables).

**Main scenario:**
- a. User selects all the data source files (xls, xml, txt) by clicking on the corresponding ‘upload’ button
- b. User inserts the DB connection details into a file and uploads the file into the system
- c. User clicks on the RUN ETL button

**Result:** The ETL executable file will not be able to run reason why no data will be inserted into any of the tables. Therefore, the web application will only display empty tables but no new information is being added. As a future improvement an error message should be generated on the main interface of the application showing the cause of the problem.

**Use case #4 (Undo ETL)**

**Main actor:** User (UPC administrator)

**Preconditions:** Data source files are uploaded successfully and the database connection parameters are correct

**Main scenario:**
- a. User selects all the data source files (xls, xml, txt) by clicking on the corresponding ‘upload’ button
- b. User inserts the DB connection details into a file and uploads the file into the system
- c. User clicks on the RUN ETL button
- d. User clicks on the UNDO button

**Result:** The new insertions are being deleted from the database tables. As a future work, the new updates from some of the columns should be also rolled back.

### 6.6 Monitor/Error logging tables

Below I will display a first look into our logging tables. We can see that the UI of these tables includes most of the functional and nonfunctional requirements that I have mentioned before. The three logging tables are: *monitor_logging_insertion, monitor_logging_extraction* and *error_logging*
6.6.1 Monitor_logging_insertion table

In this table we are storing the monitored records that are generated after the insert operations. As we can see the table displays the following information:

- **Project, job name** (dimension/fact table name) that has been monitored
- Execution parameter details such as **Start Date/Start Time/Stop Time, Duration**.
- ETL ID (Load id) which is incremented in the first column. As we see it is the fifth time I have executed the ETL flow, reason why the maximum ID is 5. The user has the possibility to sort/order the information ASC/DESC, to see the newest added information in the logging table by double clicking on the LoadId column tab.
- Lines written: The number of lines that have been written to the database tables after performing the insertion

**Descriptive information:**

- **Show [#] entries**: The user has the possibility to choose between displaying the first [10, 25, 50, 100] entries
- **Search Box**: If the user is interested to check for specific information he can use the search box. The string sentence can be type into the search box, i.e.: insert, candidate, delivery, etc.
- The message displayed at the bottom of the table emphasizes how many entries are being displayed
- **Page view**: The user can navigate through the table entries by clicking on the <Preview, Next> tabs.
6.6.2 Monitor_logging_extraction Table

In the monitoring extraction table we are storing the monitored records that are generated after the extraction and join operations from our tables. As we can see the table displays the following information:

- **Project, job name** (dimension name) that has been monitored
- **Load ID** which is incremented in the first column.
- **Component origin**: Displays the origin of the component that has been monitored (i.e. extract-'Load component' or join-'Join Datasources')
- **Label**: This attribute displays more specific information about the extract operation which has been monitored (i.e.: .xls, .xml, etc) or information regarding the joining attribute.
- **Count**: This attribute is similar with the Lines_written attribute from the previous table, since it does a count of the number of rows that are outputted after each monitored operation.
**Descriptive attributes:**
The table contains the same features as we have previously described in the other monitoring table. The user is able to search within the table’s search box, show more table entries, go to next/previous pages, etc.

### 6.6.3 Error logging table

<table>
<thead>
<tr>
<th>LoadID</th>
<th>Project</th>
<th>Job Name</th>
<th>Start Date</th>
<th>Error Type</th>
<th>Component Origin</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CLAU</td>
<td>update_temporary_tables</td>
<td>2014-12-02 20:17:20</td>
<td>FILE MISSING</td>
<td>Checking if .xls file exist</td>
<td>fout.xls was missing in source ETL job</td>
</tr>
<tr>
<td>1</td>
<td>CLAU</td>
<td>test_schema</td>
<td>2014-12-02 20:17:20</td>
<td>Missing Attribute from tbl</td>
<td>-</td>
<td>Attribute was missing in source ETL job</td>
</tr>
<tr>
<td>2</td>
<td>CLAU</td>
<td>test_schema</td>
<td>2014-12-02 20:17:30</td>
<td>Missing Attribute from tbl</td>
<td>-</td>
<td>Attribute was missing in source ETL job</td>
</tr>
<tr>
<td>3</td>
<td>CLAU</td>
<td>test_schema</td>
<td>2014-12-02 20:17:40</td>
<td>Missing Attribute from tbl</td>
<td>-</td>
<td>Attribute was missing in source ETL job</td>
</tr>
<tr>
<td>4</td>
<td>CLAU</td>
<td>test_schema</td>
<td>2014-12-02 20:17:50</td>
<td>Missing Attribute from tbl</td>
<td>-</td>
<td>Attribute was missing in source ETL job</td>
</tr>
<tr>
<td>5</td>
<td>CLAU</td>
<td>test_schema</td>
<td>2014-12-02 20:17:50</td>
<td>Missing Attribute from tbl</td>
<td>-</td>
<td>Attribute was missing in source ETL job</td>
</tr>
</tbody>
</table>

*Figure [6.6.3]. Error Logging Table*

In the error handling table we are storing the records that are generated whenever a specific type of error occurs into our system. As we can see the table displays the following information:

- **Project, job name, Start Date**
- **Load ID** which is incremented in the first column. As we have seen in the previous case, the user can order the ID value ASC or DESC to difference the new insertions from the old ones.
- **Component origin**: Displays the origin of the component which has been logged due to the error that has occurred.
- **Error Type**: This attribute details the type of error that has occurred, and helps the user to better perform future corrections, i.e.: In case the .xls temporary data source is missing, the error type will be ‘File Missing’.
• *Message:* The hard-coded message generated by specific types of errors is displayed in this field.
Chapter 7. Future Work

The system presents some improvements that can be considered for the future, in terms of better automating the error handling/monitoring functionalities, improve the stability of the system, increase the performance and overall to reduce the probability of encountering an error during the execution. As we have seen throughout the thesis project, my work was based mainly on catching the errors that might occur during the execution time and monitoring the information about certain ETL operations. As a future work, we can consider fixing the errors that are appearing in the logging table so that eventually the error_logging table should contain as few logged errors as possible. Below I will also describe some of the major improvements that could enrich the user experience and the web application functionality. First of all, I will enumerate the features that are presented in Appendix A (Additional design features and functionalities). The implementation of this features has been started but until the submission moment they have not been fully integrated.

7.1 ETL level

7.1.1 Design features and functionalities (See Appendix A)

a. **ETL BPMN design consistencies**: In case the commercial version of the tool will be installed, it will be possible to use the tParallelize component for parallelizing the tasks execution. This can be interesting in case the developer wants to obtain a maximum visual consistency with the BPMN representations. Although, the solution I have found for paralyzing the ETL jobs has worked as well, I consider that having a special component that can perform this action can be beneficial from the visual perspective.

b. Add ‘Level’ column to the logging tables to help the user better understand the connection between the logged operation and its corresponding BPMN conceptual task.

c. **Check for duplicates feature**: The possibility to add a button on the web interface that will allow the user to automatically check for duplicate values within the inputted sources. At the ETL level I have shown in Appendix A how we can use the tMatchGroup and tUniquesRow components from Talend OS to perform this action.

7.1.2 Error handling/Monitor logging

*Error handling*: The error handling functionalities can be improved especially at the fact table level. In the current state of the project, the system is able to capture and display the errors that might occur during the execution of the fact tables, but it is not fully stable. Currently, if there is an issue during the load of the fact tables an error message will be stored in the logging table, but without any further details. Therefore, we should display the hard-coded error message into the logging table with the specific type of the error. This task can be performed similarly as for the dimensions, when we have hard-coded the corresponding message for each error type.
7.1.3 Roll-back Update operations

Although chapter 5.5.4 contains the steps I have achieved into implementing this functionality, there are still improvements that can be done. I consider that the steps which I have followed and previously explained, can be a starting point for finishing implementing this functionality. An interesting feature that could be added at the ETL level is giving the user the possibility to undo the operation from the dimensions or fact tables individually, according to the type of error that has occurred. Sometimes there might not be necessary to undo the operations from both the fact tables and dimensions. Therefore, we should have two undo buttons on the front end interface for allowing the user to roll back either the operations from dimensions or from the fact tables. A solution for this would be to generate two different ID columns for the dimensions and fact tables. Whenever the user clicks on the corresponding undo button, the system will roll back the corresponding operations.

7.1.4 Improve system’s performance

The queries used while creating the fact tables can be analyzed and if necessary they should be optimized. In order to achieve this we can use the EXPLAIN utility from PostgreSQL. This feature is used to examine the execution of a given query. For example, it can be used as a query debugger to see how much time it takes to get every row of data from the dimensions. Although in the specific case of our project, the data we are extracting is not big, as a future work we can take into account the query performance optimization for the fact tables. Moreover the overall stability of the system can be increased and if needed the automation of new functionalities.

7.2 Web application

Regarding the interface of the application we can consider enriching it with new functionalities, such as:

- Adding a “Duplicate Check-Up” button, so that if necessary the user can examine the duplicate values inside the data sources.
- Better re-organize the logging tables in order to divide the logged information about the dimensions and about the fact tables. Therefore, two additional buttons such as ‘Dimensions’ and ‘Fact tables’ would allow the user to see the information which interests him more.

For a more interactive and user-friendly application, the front interface can also be restructured in the future.
Chapter 8. Summary

8.1 ETL Design improvements

By using Talend Open Studio, it has been possible to redesign and execute the ETL processes, starting with the data extraction from the external sources, transforming and loading it into the DW. As stated in Chapter 3.1, the initial state of the ETL design contained multiple errors that were not tracked nor recorded by the system. Therefore, the initial design of the ETL processes was considered to be incomplete due to its lack of providing monitoring and error logging capabilities. Before going into the specific design improvements that have been performed during this project, it is important to state again the main factors that were taken into consideration when redesigning the ETL workflow. This factors that I am referring to are:

a. ETL--BPMN Consistency: The ETL flow should be consistent with the BPMN diagrams in order to ensure a maximum compatibility between the system’s functionalities and the product owners’ (UPC professors) requirements.

b. Error/Monitor Logging: The system should allow the user to test and monitor the main ETL processes that occur during the data extraction and data insertion stages. In order to implement these tasks, new components had to be added in the initial design.

8.1.1 BPMN-ETL Consistency

After a series of discussions with the final users of the system, I have designed the main processes that should occur during the extraction, transformation and loading phases. In order to conceptually design the ETL operations in a logical manner I have used BPMN (standing for Business Process Management Notation).

a. BPMN static level: The first step was to design a higher level view of the main ETL steps inside the system. The initial representation of the control flow included the main dimensions and fact tables that are forming the ETL structure. Moreover, by conceptually modeling the ETL operations that occur inside the dimensions and fact tables I have been able to maintain an efficient communication with the final users of the system, with whom I had a series of discussions throughout the modeling phase. Modelling the initial ETL design using BPMN has helped me to better understand the errors that might occur during the data integration phase of the project.

b. BPMN dynamic level: During this modeling stage I have added more dynamicity to the conceptual design by including certain functionalities such as error handling, monitoring, etc. The initial ETL design did not present any monitoring/error handling capabilities. Therefore, during this stage I have included these new functionalities into the conceptual model of the design. I have used different BPMN notations in order to represent the dynamicity of the system, such as: signal, error, cancel or compensation events. For a better understanding of these notations I have used the following book: BPMN Method and Style from Bruce Silver [3]. In conclusion, when modeling the BPMN representations of the ETL processes I considered the
main requirements that the system should accomplish w.r.t the error handling and monitoring functionalities.

c. **BPMN to ETL:** By using Talend OS I have managed to complete the cycle of redesigning and implementing the ETL processes based on the new added functionalities, while ensuring a good level of consistency with the BPMN conceptual representations. Although, some of the steps inside the newly redesigned ETL flow do not concur with the BPMN notation, this is due to the fact that the tool I have been using is capable of performing two conceptual tasks by using only one component, i.e.: selecting and renaming attributes from one of the data sources can be done by using a single component. Moreover, the initial design of the ETL, which was previously performed in Pentaho Kettle, presented a lack of understandability regarding the flow of the ETL processes. As an example, part of the components that formed the initial ETL design have been chaotically ordered and labeled. Considering the future work that will be performed for this project, in order to improve the communication between the users and the technical developer I have also redesigned the flow by following a logical order of the components, together with a consistent labeling.

**8.1.2 Monitor/Error Logging**

As stated in the first chapter of this paper, one of the main objectives has been to offer users the possibility to perform a comprehensive error handling and monitoring procedure w.r.t data extraction, transformation and load steps.

a. **Monitoring:** In some cases the user might be interested in checking the execution of the ETL flow. By monitoring the ETL operations, the user is able to have a deeper insight inside the system. Therefore, by using Talend OS I have managed to include in the ETL design certain components with the specific purpose of monitoring different sub-steps that occur during the data extraction and data insertion phases. Implementing the ETL monitoring capabilities will allow the system to answer some of the main questions that the user might be interested in, such as:

- **Data extraction:** How many rows have been extracted from a specific data source? How many rows resulted after performing the join operation between different data sources? When was the data extraction step performed (date and time)?

- **Data load:** How many rows have been inserted into a specific database table? When was the data insertion step performed (date and time)?

b. **Error Handling:** During this modeling and implementation stage I have considered the main scenarios that might cause errors into the system. By allowing the user to perform a comprehensive error handling procedure for the data insertion and data extraction phases, the system can later on be tested and eventually be improved. The ETL tool which I have used for redesigning the ETL flow allowed me to include most of the functionalities that could help the user in performing the error handling operations.
• **Data Extraction:** As I have presented in the third chapter of this paper, several errors can occur during the data extraction stage. For example, the data source external files (.txt,.xls,.xml files) containing student information might have one or more missing attributes. A typical scenario would be regarding the (.txt) file which might not contain the ‘username’ or ‘DNI’ attribute. As a result the joining step between the information coming from the (.txt) file and the information coming from the (.xml) file might not work properly, since the join operation is done according to the ‘username’ attribute. Another scenario would be regarding the (.xls) file which might not contain the ‘nota’ field. The system I have designed is able to handle these errors and store a message about them in the error_logging table. Although there is no join operation between the data sources w.r.t this attribute, the correct solution for this error is to fill the missing attribute with null values. Furthermore, regarding the information about the calendar university year, in some cases the (.xls) file might not be available. After discussing this problem with the final users of this system, it has been concluded that this scenario can be commonly encountered. However, the error generated by this missing file should not stop the ETL flow, but rather should allow the ETL flow to continue while partially influencing the update operation of a certain column(closed_session). The examples presented above emphasize the need of allowing the user to perform a comprehensive error handling procedure of the ETL processes. By using Talend OS I was able to deal with the scenarios mentioned above and give the user an overview over the errors that might occur during the data extraction phase.

• **Data Load:** It is likely for errors to occur during the loading stage. These errors are common especially while loading data into the fact tables. This is due to the fact that the errors that could occur inside the dimension tables have a negative impact over the selection and insertion operation of the fact tables. A typical scenario that can generate these errors could be the missing attributes inside the dimensions. By using BPMN to represent the ETL steps inside the selection and insertion phases of the Fact Tables, I have managed to model the behavior of the system and find the cases in which the errors are more likely to occur. Talend OS has allowed me to implement the error handling procedure at the dimension level and fact table level. Although there are many improvements that can still be done in order to improve the stability of the system and the error handling functionalities especially at the fact tables level, I consider that the work I have performed so far can offer good perspectives w.r.t this project continuation.

8.1.3 **ETL Tools selection summary**
One of the goals of this project was to perform a comprehensive ETL tools comparison, in order to find the right tool that could answer the requirements of this project. As we saw in the fourth chapter, the ETL flow was initially designed using Pentaho Kettle. Although this tool offers good capabilities for designing and monitoring ETL processes, it lacks in offering comprehensive debugging and error handling functionalities. Moreover, regarding the integration
of the ETL job within a web interface, the procedure is not that well documented as it is for other tools. Therefore, while redesigning the ETL flow there was a need to consider other tools as well. I have chosen *Talend Open Studio* and *Informatica* as alternatives to using *Pentaho Kettle*. The tools comparison presented in the fourth chapter of this paper was based on different criteria such as: connectivity, debugging, monitoring, error handling and integration capabilities. Although all the chosen tools offer the functionalities that cover the above mentioned criteria, the choice between the tools has been justified due to certain requirements specific for our project. For example, *Talend OS* allows us to perform comprehensive ETL error debugging and monitoring, due to its code driven approach. *Talend OS* is built over a Java platform which makes it easy to integrate and debug Java code within the tool. Moreover, the functionalities offered by *Talend OS* proved to be very efficient when performing the error handling and debugging steps. Another reason for choosing *Talend OS* as the main tool to redesign the ETL flow is due to its integration capabilities within a Web interface. Using *Talend ESB* we can easily build a job and export the executable file which can later on be used for integration purposes. Overall the three tools are very similar in terms of functionalities, but as stated before the difference between them was made by small details based on the requirements of our project.

### 8.2 ETL analyzer Web application

The final goal of this thesis project was to build a web application that should allow the final user to interactively interact with the system.

**a. Input:** The web application that I have built allows the user to input the necessary data source files containing student and calendar university year information. Also, the user can input the database connection parameters from within the interface, and run the ETL. The undo button has also been implemented, but in this current state of the implementation the user is able to undo only the new insertions from the database, except for the updates. For rolling back the update operations, especially at the fact table level, I have detailed the steps which I consider should be followed, in the previous chapters. Moreover, chapter 6 contains the main functionalities of the interface. Even though the interactivity of the interface was not a priority for this project, I have tried to give the web application a pleasant work-around environment.

**b. Output:** After clicking on the run button, the application displays the logged information from the monitoring and error handling tables. Due to the fact that the information inside the monitoring table can be chaotically ordered, I have created two separate tables in order to divide the information according to the main operations that are monitored(extract, join, insert).

By using the ETL analyzer web application the user is able to answer questions like:
• Monitoring: How many rows were extracted from a specific data source?; How many tuples were joined between two specific data sources? (i.e.: join between .TXT and .XLS datasources); When did the execution of the monitored processes took place? (Start Date attribute)

• Error handling: What are the missing attributes from the data sources? What is the type of error that occurred? What are the missing attributes from the dimensions? (fact tables error handling)

8.3 General remarks

As we have seen in the previous sections from this chapter, a big portion of my thesis paper has been based on improving the functionalities from a specific project. However, I can make some general remarks based on the work I have achieved and the experiences I have encountered while redesigning the ETL processes. First of all I would like to point out the advantages which I have noticed while using a conceptual approach for modelling the ETL processes. By following the BPMN notations, displayed by Bruce Silver in the book “BPMN Method and Style”[3], I have managed to conceptually model the operations during the extraction, transformation and insertion stages. Moreover, as paper [1] suggests there are two perspectives when representing the ETL processes: control flow view and data flow view. During this project I have used these notions while representing the ETL workflow at two different levels. Firstly by modelling the ETL flow at a higher conceptual level I was able to deduct how the processes interact with each. Secondly, at the dynamic level I managed to understand how the data flows between the components and better handle the errors that are appearing at different stages during the data integration process. From the communication point of view I have noticed that using BPMN has efficiently increased the compatibility between the product owners’ requirements (UPC professors) and the implementation work. The alternative of using BPMN would have been to use an UML based modelling approach of the ETL processes. Although this approach has been taken into account in the beginning of the modelling phase, I consider that using BPMN to solve these tasks is a good solution since it addresses to more categories of users with different backgrounds.

Furthermore, allowing the user to handle the errors that could occur during the execution of the ETL and to track/monitor the main operations from the design, is an important aspect that should be taken into account during the data integration stage of any data warehouse project. We have seen that offering the user the possibility to track the source of errors, monitor the ETL operations and visualize the results in a web application, can bring a qualitative benefit during the data integration stage.
## 8.4 Final planning

Due to the limited amount of time for performing this project, the final planning slightly differs from the initial planning that I considered at the beginning of the project. Table below presents the final timetable according to the work I have achieved for this thesis project.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Start Date</th>
<th>End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Study the initial material</td>
<td>15-September</td>
<td>25-September</td>
</tr>
<tr>
<td>- Get familiar with the initial ETL design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Create the BPMN representation of the ETL processes: Static level</td>
<td>25-September</td>
<td>08-October</td>
</tr>
<tr>
<td>- Pre/Post conditions document</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Initial Design Errors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Create a BPMN representation of the ETL processes: Dynamic level</td>
<td>08-October</td>
<td>28-October</td>
</tr>
<tr>
<td>- ETL tools installation and familiarization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Apply corrections for the Pre/Post conditions document</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- ETL tools comparison</td>
<td>28-October</td>
<td>05-November</td>
</tr>
<tr>
<td>- Finalize the BPMN conceptualization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Redesign the ETL static flow using Talend OS</td>
<td>05-November</td>
<td>12-November</td>
</tr>
<tr>
<td>- Start the Web application implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Improve the ETL dynamic design in Talend OS by adding new functionalities: error handling/monitoring</td>
<td>12-November</td>
<td>22-November</td>
</tr>
<tr>
<td>- Create the front end interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Trial run for the ETL job within the Web application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Start Thesis Writing Document</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Focus on the Web application output: Monitoring/Error logging tables. Add details inside the tables, i.e.: error type, monitored process name, start date, etc.</td>
<td>22-November</td>
<td>02-December</td>
</tr>
<tr>
<td>- Correct the BPMN Diagrams: Notational issues</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Present the initial thesis outline, continue the writing process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Correct the BPMN Diagrams: Notational issues</td>
<td>02-December</td>
<td>12-December</td>
</tr>
<tr>
<td>- Thesis Writing finalization</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table [14] Final planning
9. References


APPENDIX A. Additional design features and functionalities

This appendix has been referred in the future work chapter of the paper. Here I will present the functionalities that I have been recently working on, but due to the limited amount of time these features are not fully functional. However, I will enumerate these features and present the current work that has been done so far, regarding their implementation. I consider that this information is useful for the future improvements that can be included in the implementation process.

A.1 Extra functionalities

**Level column**: In order to show the connection between the ETL monitored operation and the BPMN design, during the implementation we can add a new column to the monitoring table. Each ETL step where the monitoring operation is performed corresponds to a specific value, following the next order: Control flow [level1], Dimensions and Fact Tables [level 2], Sub-steps inside the dimension/fact tables [level 3]. The benefit of having this column is to ease the work of the user, in case there is a need to check for consistency between the BPMN design and the ETL processes.

In order to perform this task, Talend OS provides the tMap component which could be inputted after each of the steps that are being monitored. In this way we will be able to assign different default level values for each ETL monitored operation. Therefore, the level column can help the user to cross-check the ETL monitored processes with the BPMN conceptual design. Below I will present an example of how this task can be performed using the Talend OS tool. Let us consider for example that we want to add the corresponding level value for each of the data source extraction operation inside the Candidate dimension. In order to perform this task the steps that we should follow are:

- Drop a new tMap component after the monitored process.
- Add a tPostgreSQLOutput component (monitor_logging tables)
- Besides the already existing columns inside the logging table, add a new column entitled ‘Level’
- In the output table component, click on ‘Edit Schema’ in order to ensure that the level column is present
- Add the default Level value attribute for each monitored process. In our case, since the monitored process is an sub-step extract operation inside the Candidate dimension, the corresponding level value would be ‘3’
A.2 ETL design consistency

In order to increase the understandability between the ETL flow and the conceptual BPMN design, Talend OS provides a series of components that can be useful w.r.t this manner. For example, while looking at the control flow diagram presented in the previous chapters we can see that the dimensions are connected in parallel by using an AND gateway. The name of the component provided by Talend OS for representing tasks that are running in parallel is tParallelize. The solution I have found for parallelizing the dimensions and fact tables execution from the design point of view was to use a special action tab, that Talend is offering as an alternative to using the tParallelize component. The reason why I have not represented the dimensions in parallel by explicitly using this component, is due to the fact that it is only available in the commercial version of the tool. However, I found a different solution for performing this task by using the Multi Thread execution tab. The tParallelize component can be taken into account for the future in order to ensure a visual consistency with the BPMN representation.

In order to represent the parallel execution of the insertion dimensions (insert dim_item, update_candidate, insert_dim_delivery_system, etc.) we can follow the next steps:

- Drop a tParallelize component in the design workspace, from the orchestration family components (considering that the commercial version of the tool is already installed)
- Connect the tParallelize component with the insertion dimensions using a Parallelize outgoing link

A.3 Checking for duplicate values

Generally during the data extraction phase of an ETL testing procedure, it is recommended to check for duplicate records. In the specific case of our project, this functionality is not a priority since we consider that the system which generates the data source files containing student information is not supposed to produce duplicate records. However the system is prone to errors, so as a future work we could take into consideration the errors that might be generated by the existence of duplicate records. For example if the data source file containing the student DNI
information contains records about two different students having the same DNI value but different usernames, this records should be inspected. Catching these duplicate records will not necessarily mean that there is a system error, but still this scenario should be inspected. Moreover, this duplicates checking feature can be integrated in our web application and allow the user to easily check for duplicate into the inputted data sources. For implementing it at the ETL level by using Talend OS, we need to emphasize the following possibilities. If we are using the commercial version of the tool we can take advantage of the tMatchGroup component, while if we are using the open source version of the tool we can take advantage of the tUniqueRow component.

A.3.1 Using **tMatchGroup component**

This component allows us to create a certain group of similar data records using different matching rules. The table below presents the main properties of the component.

<table>
<thead>
<tr>
<th>Component Family</th>
<th>Data Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>The main function of this component is to encounter duplicate records and to group these records together for further analysis.</td>
</tr>
</tbody>
</table>
| Settings         | The following parameters are being defined in the output schema:  
  - GID: The ID value represents the identifier of the group  
  - GRP_SIZE: Counting the number of records that are present in the group  
  - MASTER: Allows us to choose the matching algorithm between records  
  - SCORE: The score value shows us the similarity degree between attributes, w.r.t the matching algorithm |

*Table[A.3.1]. tMatchGroup component properties*

In the specific case of our project let us consider a scenario in which we are interested to see if the data source files generated by the system contain duplicate records. In this particular case we would like to group the duplicate records, and insert them into one table that can be used for further inspection analysis. The figure bellow presents an example of how this component can be mapped to our existing ETL flow.
As we can see in the figure above, I am using two special components:

a. tMatchGroup(\text{Group\_records}) component for gathering and grouping duplicate records based on specific inputted attributes. In the scenario described above we can use different algorithms (i.e.: Jaro-Winkler matching method), w.r.t different attributes from the data source files.

b. tLogRow (\text{Matches}, \text{Suspects}, \text{Uniques}) components for analyzing the information inside the table. For analyzing the information we can use three different output flows w.r.t the threshold that can be defined in the confidence threshold field. Therefore, we can use the output flows to analyze the information by taking into consideration several parameters.

A.3.2 Using \text{tUniqueRow} component

This component is generally used for solving quality issues inside an ETL flow. The differences between using this component or using the \text{tMatchGroup} component for solving the issues generated by duplicate records are:

a. \text{Availability}: This component can be found in the open source version of the tool, while the \text{tMatchGroup} component can only be found in the commercial version.

b. \text{Settings}: \text{tUniquesRows} is used for splitting the entry rows into two categories: unique rows and duplicate rows. On the other hand, \text{tMatchingGroup} is mostly used to group the suspected duplicate records and offers the possibility to analyze the information from this group.

Let us use the same example as I have previously mentioned, a scenario in which we are interested to see if the data source files generated by the system contain duplicate records. Using the \text{tUniqueRows} component we are only able to split the unique and duplicate rows, and store the information into different external sources (Files/DB tables). The figure below presents an example of how this component can be mapped to our existing ETL flow.
As we can see in the figure above, the components I am using are:

a. tUniqueRows (Deduplicate) component for gathering and grouping duplicate records based on specific inputted attributes.

b. tLogRow (Unique,Duplicate) components having two different flows for the unique and duplicated records.
APPENDIX B: ETL Tools Information

The information from this appendix has been referred in the fourth chapter of the paper (ETL Tools comparison). This section contains a detailed description for each of the selected ETL tools, w.r.t different factors such as: connectivity(ability to connect to PostgreSQL database), operations(difficulty in performing the Extract, Join, Insert operations), debugging (understandability and functionalities of the debugging panel), monitor/error logging functionalities, integration capabilities.

B.1 Pentaho Kettle

Pentaho is a commercial BI suite that provides solutions for data integration, data visualization, reporting, etc. The product provided by Pentaho for performing data integration tasks is called Kettle. Before going into more specific details, we can mention some of the high level characteristics that define this product which are: meta-data driven approach, easy-to-use and active community for the whole Pentaho suite.

B.1.1 Connectivity

*Pentaho Kettle* offers several solutions for connecting with the existing databases and for performing particular operations( extract, join, insert)

**PostgreSQL database connection**

Setting up the connection in Pentaho Kettle is done by right clicking on the Database Connections→New tab and choosing the corresponding database. In our case, for setting up a connection to PostgreSQL  we need to input the following parameters: *Host Name, Database Name, Port number, User Name, Password*

B.1.2 Operations

**Extraction**

Data extraction phase from multiple source files can be easily performed using Pentaho Kettle tool (See Table 2):

- File type: Represents the type of files that are loaded throughout the ETL process
- Components: Represents the name of the component offered by Pentaho Kettle which is used to extract information from the corresponding source file
Extracting data from the .xls, .xml, .txt files is performed by using the above mentioned tasks. While executing this phase there are a number of steps that need to be followed:

a. Import the files in Kettle from your local computer by using the corresponding component for each type of file

b. Set up the file parameters and fields that should be displayed:
   - .txt file: One of the parameters that Kettle provides when extracting data from a .txt file is to select the ‘No empty rows’ tab. For extracting the right content from the .txt file we can select the Fields ➔ Get Fields tab. This tab can be found inside the Text File Input task.
   - .xls file: After importing the .xls file into Kettle we need to select the ‘Sheets’ tab. For extracting the right content from the corresponding .xls sheet, we can select Fields ➔ Get Fields from header row. This tab can be found inside the Microsoft Excel Input task.
   - .xml file: For extracting the right content from the .xml file we can select Fields ➔ Get Fields. This action can be found inside the Get Data from XML tab.

Data extraction from a table is done by using the Table Input task. When performing this extraction there are a number of steps that need to be followed:

- Make sure that the ODBC drivers are installed
- Set up the connection to the DB table (i.e.: For PostgreSQL database we need to set up the Host name, DB name, port, login and password)
- Write the corresponding SQL query, to SELECT the needed attributes FROM the DB table
- Optional step: Preview the data that is extracted before actually loading it our system

Depending on the type of data that is being extracted, additional component can be used such as: Fixed File Input, JSON Input, CSV File Input
Join

The join operation between the data sources is done according to certain attributes. Pentaho Kettle allows us to easily perform this operation using the following component.

<table>
<thead>
<tr>
<th>Join Operation</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Join between .txt file → .xml file</td>
<td>Join rows -condition-</td>
</tr>
<tr>
<td>.xml file → DB table</td>
<td></td>
</tr>
<tr>
<td>.txt file → .xls file</td>
<td></td>
</tr>
</tbody>
</table>

*Table [B.2] Join Operation: Pentaho Kettle*

In our specific case we are using certain steps in order to join the data sources according to specific attributes, i.e.:

- The Join operation between .txt file → .xml file is done according to the `username` attribute
- The Join operation between .xml file → DB table data is done according to the `userId` attribute
- The Join operation between .txt file → .xls file is done according to the `DNI` attribute

The above conditions are inputted inside the `Join rows` component. Moreover, when adding the `course` and `semester` columns obtained from the DB table to the student information, this component can act as an `Union` operator. Additional tasks can be performed in Kettle depending on the type of `Join` that is needed: Merge Join, Merge Rows, Multi way Merge Join.

Insert

Inserting data into the DB using the Pentaho Kettle tool (See table 3):

<table>
<thead>
<tr>
<th>Database</th>
<th>Connection to DB</th>
</tr>
</thead>
</table>
| PostgreSQL | a. Connection to PostgreSQL is done from inside the Insert Output component  
b. Set up the connection parameters to the database table (i.e.: Host name, DB name, port login and password)  
c. Test connection to the DB |

*Table [B.3] Insert Operation: Pentaho Kettle*
When inserting data into a table we can map the fields that we need to insert/update with the corresponding table attributes. This operation is done by selecting the Edit Mapping tab, from inside the Input Output task.

In our specific case, when inserting data into the fact tables we use an SQL task in order to write the corresponding insertion script. A good example is the three tasks that can be found in the Update Fact Tables transformation from our ETL, i.e.: Update Experiments, Update Valid table, Update Invalid table.

B.1.3 Debugging

Pentaho Kettle allows us to execute a transformation or a job from within the Spoon design environment on our local machine. This is ideal for debugging and testing transformations or ETL activities.

a. Execution results panel

The debugging can be performed from the Execution Results panel:

- In the Spoon graphical interface, click Run This Transformation. The Execute a transformation dialog box appears.
- Click Launch. The transformation executes. Upon running the transformation, the Execution Results panel opens below the canvas.

Understandability: While debugging the initial state of the ETL design I have encountered several design errors that were throwing exceptions. For example some of the attributes selected from the data source files were either missing or contained null values. Whenever an error was issued, the ‘execution results panel’ of the tool would display the error message. These error messages that I was receiving were hard to decipher due to the unstructured nature of the displayed information. Therefore the time spent for finding the cause of the errors, while deciphering the information inside the execution panel was longer in comparison with the other tools.

Functionalities: One of the features that I had to include while performing the error handling steps was to configure and rename the message displayed by the debugger. For example if the (.xls) file containing calendar year information is missing, the type error displayed in our database table should be ‘Missing file’. In order to perform this action there is a need to debug the error message displayed in the ‘execution results panel’ and change this message to the desired format. This action proved difficult to be solved in Pentaho Kettle due to the inability to hard-code the message.
b. Debugging settings

The following table provides a detailed description of the debug settings provided by Pentaho Kettle. These settings refer to the ability of performing certain actions such as adding breakpoints, pausing the debugger, etc.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step List</td>
<td>This panel allows us to configure different steps inside our ETL flow. For example this list can be used if I want to debug and configure the joining steps inside the Candidate Dimension.</td>
</tr>
<tr>
<td>Number of rows to retrieve</td>
<td>Configure the number of rows we want to preview for the selected step. This setting has proven to be useful when trying to preview how many rows were read from the source files (.txt,.xls,.xml)</td>
</tr>
<tr>
<td>Pause Transformation (Set condition)</td>
<td>Pause the debugging process if one of the conditional break-points is activated. In our specific case I have used this setting to debug the join operations inside the Candidate Dimension.</td>
</tr>
<tr>
<td>Break-point/Pause condition</td>
<td>Possibility to configure the break-points conditions</td>
</tr>
</tbody>
</table>

*Table[B.4] Debugging settings: Pentaho Kettle*

Depending on the context, other settings can be used for debugging in Pentaho Kettle such as *Analyze the impact of this transformation* on the database, *Preview/Replay/Verify* transformation.

**B.1.4 Monitoring**

a. Monitoring settings

Pentaho Kettle allows us to monitor the execution of each step from inside a transformation, by using the *Step Metric, Execution History* and *Performance Graph* tabs from the Execution Results panel. Moreover, Pentaho Kettle allows the monitoring information to be outputted into an external database logging table.

- *Step Metrics*: It provides statistics for each step in our transformation including how many records were read, rows written, processing speed (rows per second) and more. If any of the steps caused the transformation to fail, they would be highlighted in red. In the specific case of our project this functionality can be used to check the number of rows read from the data source files and the number of rows inserted in the database table. This
monitoring information can be later on recorded and stored in the corresponding logging table from the database.

- **Performance Graph:** It allows us to interactively analyze and visualize the performance graphs based on a variety of metrics including how many records were read, rows written, processing speed (rows per second). Although creating reports and dashboards based on the ETL process performance has not been a requirement for our specific purpose, as a future work is important to mention this interesting capability offered by Kettle.

- **Execution History:** It provides us information about the previous executions of the transformation. This functionality is useful when performing the rollback operation of some of different steps. We can use this feature to look at the details of the information that has been previously inserted.

### b. Output the monitored information into an external table

In Pentaho Kettle each task information can be monitored and the result outputted either in the Execution Results panel, or in an external log file or database table. This operation can be performed by following the next steps:

1. Select the **Transformation setting** ➔ **Logging tab**
2. Connect to the DB table where you want the information to be outputted (i.e.: PostgreSQL parameters: Host name, DB name, port, login and password)
3. Choose between **Transformation** or **Steps** tab, depending what we decide to log. For example if we are interested in the sub steps performed inside dimensions we should choose the **Steps** tab
4. Choose the fields that we want to appear in the DB: LOG_DATE, TRANSFORMATION_NAME, LINES_READ, LINES_WRITTEN, etc.

### B.1.5 Error Handling

Transformation steps in Pentaho Kettle may encounter errors at many levels. They may encounter unexpected data errors, or problems with the execution environment. Depending on the nature of the error, the system can decide to stop the transformation by throwing an exception, or support the PDI Error Handling feature, which allows us to divert bad rows to an error handling step.

- **Check for missing files and attributes:** One of the error handling steps that we had to perform during the development of this thesis was to check for missing attributes or missing data source files. Checking for missing files can be done by using the **File exists** component.

- **Change error type and message:** Kettle displays an error messages whenever a certain type of error occurs. As stated in the debugging subsection in order to hard-code the
name of this message to a particular string is a difficult task to perform in Pentaho Kettle.

- **Throwing a KettleException: Calling a Hard Stop:** If a step encounters an error during row processing, it may log an error and stop the transformation. This is done by calling setErrors, stopAll(), setOutputDone(). Alternatively, this step can throw a KettleException, which also causes the transformation to stop.

- **Implementing Per-Row Error Handling:** We may want to divert bad rows to a specific error handling step. This capability is referred to as the Error Handling feature. The Error handling tab in Pentaho Kettle can be accessed from inside each transformation. Moreover we could use the Execution Results → Logging tab to display the logging details for the most recent execution of the transformation. Error lines are highlighted in red.

### B.1.6 Kettle Integration and deployment

- **Pentaho Kitchen** – It is an application which helps the execution of ETL jobs in a batch mode, usually using a schedule which makes it easy to start and control the ETL processing.

- **Pentaho Carte** – It is a web server which allows remote monitoring of the Pentaho Data Integration ETL processes by a using a web browser.

Both features offered by Pentaho could be used to integrate the ETL job within a web application. One of the drawbacks of using Pentaho Kitchen and Pentaho Carte is that they are not very well documented.

### B.2 TALEND OPEN STUDIO

#### B.2.1 Connectivity

Talend OS offers a solution to connect with different databases and to perform particular operations (extract, join, insert) with the data coming from different sources.

**PostgreSQL database connection:**

- In the Repository expand the Metadata option
- Right-click on Db Connections.
- In the menu, click Create connection to open the Database Connection wizard.

In our case, for setting up a connection to the PostgreSQL DB we need to input the following parameters: *Host Name, Database Name, Port number, User Name, Password*
B.2.2 Operations

Extraction
Data extraction from source files using Talend Open Studio (See Table 6):

- Task Name: Represents the name of the task that is used in our ETL

<table>
<thead>
<tr>
<th>File type</th>
<th>Task Name used</th>
</tr>
</thead>
<tbody>
<tr>
<td>.xls file</td>
<td>tFileInput Excel</td>
</tr>
<tr>
<td>.xml file</td>
<td>tFileInput XML</td>
</tr>
<tr>
<td>.txt file</td>
<td>tFileInput FullRow</td>
</tr>
</tbody>
</table>

Table [B.6] Data extraction: Talend OS

Extracting data from the (.xls,.xml,.txt) files is performed by using the above tasks. When performing this extraction there are a number of steps that need to be followed:

a. Import the files in Talend OS from your local computer by using the corresponding task for each type of file

b. Set up the file parameters and fields that should be displayed:
   - .txt file: For extracting the right content from the .txt file we can select the Edit schema tab. This tab can be found inside the tFileInput FullRow task. This is useful when we are selecting only part of the attributes from our data source, i.e.: DNI, username, group, repetidor, etc.
   - .xls file: Importing the .xls file into Talend and selecting the ‘All Sheets’ tab, from inside the task. For extracting the right content from the corresponding .xls sheet, we can select First Column → Last Column tab. This tab, can be found inside the tFileInput Excel task. As well, we can selected the advanced setting tab if we prefer more advanced options, such as: trimming the columns.
   - .xml file: For extracting the right content from the .xml file we can select the tab Mapping tab. This tab, can be found inside the tFileInput XML task. As well, we can selected the advanced setting tab if we prefer more advanced options, such as: ignoring the DTD file or ignoring the namespaces of the XML file.

The extraction of data from a table is done by using the tFileInput task. When performing this extraction there are a number of steps that need to be followed:

a. Install ODBC drivers
b. Set up the connection to the DB table (i.e.: For PostgreSQL set up the Host name, DB name, port and PostgreSQL password)

Additional tasks can be performed in Talend Open Studio, depending on the type of data that is being extracted, i.e.: tFileInput MailFile, tFileInput JSON, etc. Although there are not used in the context of our project, it is interesting to mention this new capabilities.
Join
Joining data sources according to certain attributes using the Talend Open Studio tool (See table 7):

- Task Name: Represents the name of the task that has been used in our ETL
- Additional types of joins: Tasks that could be helpful for joining data in different ways.

<table>
<thead>
<tr>
<th>Join between</th>
<th>Task Name used</th>
</tr>
</thead>
<tbody>
<tr>
<td>.txt file → .xml file</td>
<td>tJoin</td>
</tr>
<tr>
<td>.xml file → DB table</td>
<td>tJoin</td>
</tr>
<tr>
<td>.txt file → .xml file</td>
<td>tJoin</td>
</tr>
</tbody>
</table>

Table[B.7]. Join operation: Talend OS

In our specific case we are using the Join operation in order to merge the data sources according to certain attributes, i.e.: username, userId, DNI. Moreover, when adding the ‘course’ and ‘semester’ columns obtained from the DB table staging_area_results to the student information, Talend OS provides a task for performing this operation called tUnite. Additional tasks can be performed in Talend OS, depending on the type of Join that is needed: tMap, tReplace, tConvert.

Insert

<table>
<thead>
<tr>
<th>Database</th>
<th>Task Name Used</th>
<th>Connection to DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostgreSQL</td>
<td>tPostgreSQL Output</td>
<td>Steps:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a. Connection to PostgreSQL is done from DB connection tab, displayed in the main user interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b. Set up the connection to the DB table (i.e.: Host name, DB name, port and PostgreSQL username--password)</td>
</tr>
</tbody>
</table>

Table[B.8]. Insert operation: Talend OS

- When inserting data into a table we can map the fields that we need to insert/update with the corresponding table attributes.
- This operation is done by selecting the tMap component. By using this component we can map the attributes regarding our information, with the corresponding attributes from our database.
- To check and test if the information has been inserted in the DB, we can use the tRow component

When inserting data into the fact tables we can use the query option from the Basic setting tab and write the corresponding SQL query.
B.2.3 Debugging

a. Debug run panel
The debug can be realized from the interface panel, after clicking on the Run Job tab. To access the Debug mode, we need to follow two easy steps:

➔ Click the Run view to access it.
➔ Click the Debug Run tab to access the debug execution modes.

Understandability: While debugging the initial state of the ETL design I have encountered several design errors that were throwing exceptions. The message displayed in the debug panel of the tool whenever a specific type of error occurs, is intuitive and can help us to track the cause of the error easily.

Functionalities: Due to the Java platform on which Talend is based, the debugging can also be performed from the source code of the job. This ensures a maximum flexibility for the developer while performing certain tasks. For example, the technical names of some ETL components need to be hardcoded to specific strings that can be outputted in the error logging table field “Origin”, i.e. [tFlowMeterCather= ’Checking the number of rows’], etc.

b. Debugging options

Before running the Job in Debug mode, we can add breakpoints to the major steps of our ETL Job flow. This will allow us to get the Job to automatically stop at each selected breakpoint. In this way, components and their respective variables can be verified individually and debugged if required. For adding breakpoints to a certain component we can follow the next steps:

➔ Right-click on the design workspace, and select Add breakpoint on the contextual menu.
➔ To switch to debug mode, click the Java Debug button on the Debug Run tab of the Run panel. Talend Studio's main window gets reorganized for debugging.
➔ We can then run the Job step by step and check each breakpoint component for the expected behavior

B.2.4 Monitoring

a. Traces Debug mode

When monitoring the results of the ETL inside the Talend tool, we can use the Traces Debug mode. The traces feature allows us to monitor the data processing when running a Job. It provides a row by row view of the component behavior and displays the dynamic result next to the Row link on
the design workspace. This functionality is useful while monitoring the insertion and the extraction operations from our ETL flow.

Moreover we can activate or deactivate Traces or decide what processed columns to display in the traces table that is displayed on the design workspace when launching the current Job.

In order to activate the Traces debug mode we need to follow the next steps:

- Click the Run view button
- Click the Debug Run tab to access the debug and traces execution modes.
- Click the down arrow of the Java Debug button and select the Traces Debug option. An icon displays under every flow of your Job to indicate that process monitoring is activated.
- Click the Traces Debug to execute the Job in Traces mode.

b. Advanced monitoring execution settings:

- tChronometerStart-tChronometerOutput components are used to check for the duration of the process. The duration values is later on included in the monitor logging tables.

b. Components used for monitoring

Talend OS allows us to monitor the execution of each step by using several components:
- tLogRow: Displays the flow content information (rows) of the process
- tFlowMetterCather. tFlowMetter – For throwing and catching the specified monitored flow, after each extraction or insertion operation.
- tChronometer: Logs can be outputted to a file or to an external relational table

c. Output the results in the DB table

The output of each transformation can be logged into additional files or tables. This operation can be performed by following the next steps:

- tPostgreSQL Output→Map the ETL flow with the output logging table from the database
- Connect to the DB table where you want the information to be outputted (i.e.: PostgreSQL: set up set up the Host name, DB name, port and PostgreSQL password)

B.2.5 Error handling

Error handling components

- tAssertCather-This component will be invoked whenever an error occurs in the dimensions
- tDie – ‘Die on error’ basically stopping the ETL flow whenever a fatal error occurs. In our specific case, let us consider an example when we try to select attributes from the dimension tables and insert data into the fact tables. In case any of the attributes from the dimension tables does not exist, we can use this component to cancel the ETL flow.
Although in our case we will continue the process, for a proof of concept this component could be used for future improvements

- tFileExist - Checks the availability of an existing file (.XLS file containing the calendar year information)

The components presented above can be used in different circumstances such as: checking if the calendar year information exists and has been uploaded in the web application (tFileExists), checking if any attribute is missing in the data source files (tLogCather, tAsserCather), checking if the fact tables generate errors (tFlowMetterCather).

**B.2.6 Integration and deployment**

Talend ESB provides the ability to deploy a Job as a Web service in order for the Job to be called by other applications. With Talend ESB, any Job can be exported and exposed as a Web service. Since Talend jobs are built over a Java platform, some of the libraries available in Java can make the ETL jobs executable and easy to integrate on the server side.

**B.3 INFORMATICA POWER CENTER**

**B.3.1 Connectivity**

Informatica PowerCenter allows connectivity to most of the existing databases. In order to set up a connection we need to follow the next steps:

- Launch the Informatica Workflow Manager.
- Connect to the repository.
- Select Connections--Relational
- Click Add, and then select ODBC
- Choose the required database

**B.3.2 Operations**

Informatica Developer contains the components specific components for extracting, joining and inserting information.

**B.3.3 Debugging**

**a. Informatica Debugger:**

Informatica Workflow Debugger is used for testing the data loading into target database tables or just to debug the errors without loading. The main purpose of debugger is to view Transformation-by-Transformation to trace out any logical error.
Understandability: While debugging the initial state of the ETL design I have encountered several design errors that were throwing exceptions. Although the understandability of the error message displayed in the debug panel is more intuitive than Kettle’s debugging panel, it is not as clear as in Talend OS.

b. Debugger options
In order to efficiently debug the transformations performed in Informatica we can use certain features, such as adding breakpoints. These features could be used when searching for error that might occur during the extraction phase of our project.

- Open the ETL Mapping workflow.
- Define different transformations as breakpoints where the Debugger has to stop to show the errors. We can add as many breakpoints as required.
- Select a transformation from the list
- Decide DATA or ERROR for stoppage
- Define number of records (for DATA) or number of errors (for ERROR) to be skipped before debugger stops to show the data.

Informatica allows us to debug the error message from inside the component and make the necessary changes.

B.3.4 Monitoring

a. Informatica Workflow Monitor panel
The Monitor Workflow feature allows the user to view details about the ETL workflow in different interactive ways. Moreover it is possible to run, stop, or resume workflows from the Workflow Monitor. We can monitor the descriptive parameters of the execution such as start date, execution time etc.

b. Output results in the DB table
Informatica allows the output of the monitoring results to be stored in an external DB table. In order to perform this task we need to input the details of our target database.

B.3.5 Error Handling

Error handling components:
To capture the errors at the transformational level in our target database table we can enable the field error_log type and afterwards perform the following steps:

- Choose the option "Relation Database"
- PMERR_DATA. Corresponding source row of error. Can be used while debugging the extraction, join and insert operation
- PMERR_MSG. Data about the error message. Can be used to store the message type, i.e.: Missing file, missing attribute etc.
• PMERR_TRANS. Stores information about when a transformation error occurs.

B.3.6 Informatica Integration and deployment

Informatica ETL jobs can be run as a web service from other applications. *Informatica Cloud* provides integration and deployment capabilities for the ETL jobs created in Informatica. One of the main disadvantages of using this integration and deployment feature, is due to cost since it requires regular subscription.
Appendix C. Preconditions & Post conditions

This appendix has been referred in the third chapter, while detailing the static level of the ETL conceptual processes. It contains the conditions and the requirements information which I have followed during the BPMN modelling stage. The pre/post conditions have been documented during the initial iterations I had with the product owners of the application, w.r.t to the main operations that are executed inside the ETL flow: CREATE, INSERT, UPDATE, ROLLBACK, JOIN, SELECT

C.1 Initial requirements

<table>
<thead>
<tr>
<th>Parameters of the system(introduced by the user)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data sources</strong></td>
</tr>
<tr>
<td><strong>DB connection information</strong></td>
</tr>
<tr>
<td><strong>The data sources and the database connection parameters are introduced from the interface of the web application, therefore is necessary for the user to upload this information before running the ETL.</strong></td>
</tr>
</tbody>
</table>

Table[C.1]. Parameters of the system

<table>
<thead>
<tr>
<th>Existing tables in the database</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data source tables</strong></td>
</tr>
<tr>
<td><strong>Dimension tables</strong></td>
</tr>
<tr>
<td><strong>Fact tables</strong></td>
</tr>
</tbody>
</table>

Table[C.2]. Existing tables in the DB
## C.2 Operations

### C.2.1 EXTRACT

<table>
<thead>
<tr>
<th>Task name</th>
<th>Input</th>
<th>Output</th>
<th>Precondition</th>
<th>Postcondition</th>
<th>Error Handling</th>
<th>Monitor</th>
</tr>
</thead>
</table>
| Load Student Information (dim_candidate) | a. Data is gathered from external files (.txt,.xml,.xls files) | Information is extracted from the external files.                     | The `.txt,.xml,.xls` data sources files containing the necessary student information, exist and are available for usage | In case data sources exist and contain the selected attributes, the join operation between data sources is going to be processed | a. Check if the files and DB table contain the necessary attributes used for joining the data sources. In case any of this attributes are missing the process should be canceled  
`.TXT` file ➔ *Username, DNI*
`.XML` file ➔ *userid, username*
`.XLS` file ➔ *DNI*
*SAR* table ➔ *UserId*  
b. In case the (joining) attributes mentioned above exist, check if there are other missing attributes selected from the data sources, | monitor_logging extraction table:  
a. Monitor how many rows have been extracted from a specific file.  
b. Monitor other descriptive parameters such as: *Start Date* of the process, *end date*, *duration*, *ETL execution ID*, etc. |
In case there is a missing attribute fill it with null values, but do not stop the ETL flow.

### C.2.2 SELECT

<table>
<thead>
<tr>
<th>Task name</th>
<th>Input</th>
<th>Output</th>
<th>Precondition</th>
<th>Post condition</th>
<th>Error Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT Attribute</td>
<td>Attributes are selected from the DB tables: t_questions, t_jocsproves, t_tematiquesqu estions Staging_area_results</td>
<td>Gathering data from the DB tables by selecting the needed attributes</td>
<td>b. The database tables presented in the input column, contain the needed attributes that are being selected</td>
<td>Information is selected from the DB tables, and the ETL transformation stage will begin</td>
<td>Check if the selected attributes from the tables mentioned in the input, exist.</td>
</tr>
</tbody>
</table>

**Tasks that are affected by this operation**

<table>
<thead>
<tr>
<th>Task name</th>
<th>Input</th>
<th>Output</th>
<th>Precondition</th>
<th>Post condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT UserID (Dim Candidate)</td>
<td>staging_area_results table</td>
<td>Information is selected from the database table, and ready to be joined with the .xml file information</td>
<td>staging_area_results table exists in the database, and contains the selected attribute</td>
<td>The join operation between the .XML file and the DB source is successfully performed</td>
</tr>
<tr>
<td>SELECT id, autor ,titol, etc.</td>
<td>t_questions table</td>
<td>Information is selected from the</td>
<td>t_questions table exists in the</td>
<td>Information is being further</td>
</tr>
<tr>
<td><strong>(Dim Item)</strong></td>
<td>database tables</td>
<td>database and contains the selected attributes</td>
<td>processed/transformed and later on inserted in the dim_item table</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>SELECT moodle</strong>&lt;br&gt;<strong>(Dim Delivery System)</strong></td>
<td>staging_area_results table</td>
<td>staging_area_results table exists in the database and contains the moodle attribute</td>
<td>Information is being further processed/transformed and inserted in the dim_delivery_system table</td>
<td></td>
</tr>
<tr>
<td><strong>SELECT id,nomjp,ordre</strong>&lt;br&gt;<strong>(Dim Experiments)</strong></td>
<td>t_jocsproves table</td>
<td>t_jocsproves table exists in the database and contains the selected attribute</td>
<td>Information is being further processed/transformed and inserted in the dim_experiments table</td>
<td></td>
</tr>
<tr>
<td><strong>SELECT momentinsert, real_day</strong>&lt;br&gt;<strong>(DimRequestTime)</strong></td>
<td>staging_area_results, calendar_fib tables</td>
<td>Information is selected from database table and temporary table</td>
<td>Information is being further processed/transformed and inserted in the dim_request_time table</td>
<td></td>
</tr>
<tr>
<td><strong>SELECT nom,id</strong>&lt;br&gt;<strong>(Dim Thematic)</strong></td>
<td>t_tematiqueque questions, dim_item tables</td>
<td>Information is selected from the database tables</td>
<td>Information is being further processed/transformed and inserted in the dim_thematic table</td>
<td></td>
</tr>
<tr>
<td><strong>SELECT attributes</strong>&lt;br&gt;<strong>(Fact Table Experiments,V)</strong></td>
<td>Dim_candidate, Dim_Item, Dim_experiment</td>
<td>Information is selected from the dimensions and the database</td>
<td>Information is being inserted in the corresponding fact table</td>
<td></td>
</tr>
<tr>
<td>alid, Invalid</td>
<td>Dim_delivery_system, Dim_request_time, SAR, t_questions</td>
<td>table</td>
<td>dimensions which are used while loading the fact table, exist</td>
<td></td>
</tr>
</tbody>
</table>

### C.2.3 CREATE

<table>
<thead>
<tr>
<th>Task name</th>
<th>Input</th>
<th>Output</th>
<th>Precondition</th>
<th>Post condition</th>
<th>Error Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CREATE temporary tables</strong></td>
<td>--- (It is considered that the tables do not exist in the DB)</td>
<td>Class_day, Calendar_fib temporary tables</td>
<td>DB does not contain any tables with the same name as the temporary tables</td>
<td>The two temporary tables are created and are present in the PostgreSQL database</td>
<td>Check if the database contains any other tables with the same name as the new temporary tables. In case it exists tables with the same name → Drop tables</td>
</tr>
<tr>
<td><strong>CREATE Monitor/Error logging tables</strong></td>
<td>--- (It is considered that the tables do not exist in the DB)</td>
<td>monitor_logging_Insertion, Monitor_logging_Extraction, Error_logging Tables</td>
<td>DB does not contain any tables with the same name as the logging tables</td>
<td>The three logging tables are created and are present in the PostgreSQL database</td>
<td>Check if it exists other tables with the same name in the DB. <strong>Solution:</strong> The creation of the tables is done only once, at the first execution of the ETL flow. After this, the option of 'Create only if tables do not exist' is enabled. Therefore the tables will not be overwritten.</td>
</tr>
</tbody>
</table>
### C.2.4 INSERT

**a) Safe_Insertion** - We assume that the information coming from the following data sources exist and that the extraction and transformation stages have been successfully executed:

- **DB Tables**: staging_area_results, t_questions, t_jocsproves, t_tematiquequestions, etc
- **Files**: .xml, .txt, .xls

**b) ErrorProne_Insertion** - We assume that some of the attributes which are used as foreign keys in the fact tables may not exist. Moreover, one of the data sources used for updating the temporary tables may not exist as well.

- Conceptually, the process should be canceled during the extraction stage in case some of the selected attributes which are necessary for the dimensions to load correctly do not exist. In case there are other attributes which are missing, but that are not essential for the execution flow (i.e.: *nota* attribute from the .xls file) the ETL flow will continue, while the missing attribute will be filled with null values and inserted in the DB.
- Error_prone_insertion can also be caused by a missing data source file (.XLS containing calendar university year information)
- Error_proneinsertion can cause the fact tables not to load properly in case some of the attributes from dimensions are missing.

#### a. Safe_Insertion

<table>
<thead>
<tr>
<th>Task name</th>
<th>Input</th>
<th>Output</th>
<th>Precondition</th>
<th>Post condition</th>
<th>Error Handling</th>
<th>Monitor</th>
</tr>
</thead>
</table>
| INSERT    | Data gathered from different data sources | Information which is inserted appears in the corresponding DB table | a) The data sources exist and contain the selected attributes  
b) The tables corresponding to each dimension/fact | The database tables corresponding to the dimensions or fact tables contain new information | This scenario applies for a safe_insertion, where we assume that the tables and data sources exist and contain all the selected attributes. Although | monitor_logging_insertion:  
a. Monitor how many rows have been inserted in the corresponding database table.  
b. Monitor other descriptive attributes, |
table where the data will be inserted, have already been created.
c) The connection to the PostgreSQL database is successful.
d) The fields from the DB table have the same data type as the values that are being inserted.

we do not consider the case, it is important to mention that errors could appear during the transformation steps between extract-insert operations. These transformation errors are not considered in the current stage of this project.

such as: start date, end date, duration, etc.

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Input</th>
<th>Output</th>
<th>Precondition</th>
<th>Post condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT Student Information into Dimension Candidate</td>
<td>-.txt,.xml,.xls files -SAR table</td>
<td>Dim_candidate table</td>
<td>Necessary attributes: Username, userId, DNI</td>
<td>Information is being inserted into the DB table, dim_candidate</td>
</tr>
<tr>
<td>INSERT into Delivery System Dimension</td>
<td>staging_area_results table</td>
<td>Dim_delivery_system table</td>
<td>Necessary attributes: moodle</td>
<td>Information is being inserted into the DB table, dim_delivery_system</td>
</tr>
<tr>
<td>INSERT Into Item Dimension</td>
<td>SAR, t_questions tables</td>
<td>Dim_item table</td>
<td>Necessary attributes: -SAR table →momentinsert -t_questions →id,autor,etc</td>
<td>Information is being inserted into the DB table, dim_item</td>
</tr>
<tr>
<td>INSERT into experiments</td>
<td>t_jocsproves table</td>
<td>Dim_experiments</td>
<td>Necessary attributes: -t_jocsproves: →id,nomjp,order</td>
<td>Information is being inserted into the DB table,</td>
</tr>
</tbody>
</table>
104

<table>
<thead>
<tr>
<th>Dimension</th>
<th>INSERT into Thematic Dimension</th>
<th>Dim_thematic table</th>
<th>Necessary attributes: Id, id_item from selected tables</th>
<th>Information is being inserted into the DB table, dim_thematic</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT into Experiments Fact Table</td>
<td>Dim_candidate, Dim_item, Dim_experiment Staging_area_ Results tables</td>
<td>Experiments table</td>
<td>Foreign key attributes from the selected dimensions, exist</td>
<td>Information is being inserted into the fact table, experiments</td>
</tr>
<tr>
<td>INSERT into Invalid Fact Table</td>
<td>Dim_candidate, Dim_item, Dim_delivery_system, Dim_experiment Staging_area_ Results tables</td>
<td>Invalid_Response_processing table</td>
<td>Foreign key attributes from the selected dimensions and tables, exist</td>
<td>Information is being inserted into the fact table, invalid</td>
</tr>
<tr>
<td>INSERT into Valid Fact Table</td>
<td>Dim_candidate, Dim_item, Dim_delivery_system, Dim_experiment Staging_area_ Results tables</td>
<td>Valid_Response_processing table</td>
<td>Foreign key attributes from the selected dimensions and tables, exist</td>
<td>Information is being inserted into the fact table, valid</td>
</tr>
</tbody>
</table>

b. ErrorProne_Insertion

<table>
<thead>
<tr>
<th>Task name</th>
<th>Input</th>
<th>Output</th>
<th>Precondition</th>
<th>Post condition</th>
<th>Error Handling</th>
<th>Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT Student Information into Dimension Candidate</td>
<td>.xls, .xml, txt files</td>
<td>Dim_candidate table</td>
<td>The file exists although it may not contain all the selected information, i.e. nota attribute from the .XLS file</td>
<td>Table dim_candidate contains newly inserted rows</td>
<td>Check if the files contain a missing attribute that is selected. Fill the missing attribute with null values</td>
<td>Monitor the error type, i.e.: ‘Missing attribute’</td>
</tr>
</tbody>
</table>
**INSERT into temporary tables**

| INSERT into temporary tables | Information regarding the calendar university year, coming from an external .XLS file | calendar_fib, class_day, tables | The .XLS file exists | Tables calendar_fib, class_day is filled with information | -Check if the .XLS file has been uploaded by the user in the web application. Although it has not been uploaded, continue the ETL flow. | Monitor_loggin g_Insertion: How many rows have been inserted into the temporary tables? |

**INSERT into request_time Dimension**

| staging area_results table | Dim_request_Time table | Necessary attributes: -real_day attribute from calendar_fib, -momentinsert from staging_area_results | Information is being inserted into the DB table, dim_request_time | Check if the temporary tables are loaded with information and contain the attribute real_day | Monitor_loggin g_Insertion: How many rows have been inserted into the DB table dim_request_time? |

---

**C.2.5 Rollback**

<table>
<thead>
<tr>
<th>Task name</th>
<th>Input</th>
<th>Output</th>
<th>Precondition</th>
<th>Post condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rollback Insertions from dimensions and fact tables</td>
<td>-Dimension Tables -Fact Tables</td>
<td>-Dimension Tables -Fact Tables</td>
<td>-ETL ID has been previously generated -New insertions have been added to the dimensions and fact tables -User clicks on the undo button from the front end interface of the application</td>
<td>The new insertions are deleted from the tables according to the maximum ETL ID</td>
</tr>
<tr>
<td>Rollback Updates (future work)</td>
<td>-Valid fact table update columns</td>
<td>-response_processing</td>
<td>- User clicks on the undo button from the</td>
<td>The updates are undone from the corresponding tables</td>
</tr>
</tbody>
</table>
from fact tables and dim_item dimension
- -response_processing
difficulty column from dim_item
- Valid fact table
front end interface of the application
- Old insertions from the tables are recorded (future work)

<table>
<thead>
<tr>
<th>Task name</th>
<th>Input</th>
<th>Output</th>
<th>Precondition</th>
<th>Post condition</th>
<th>Error Handling</th>
<th>Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATE</td>
<td>Response_processing, Valid fact table</td>
<td>Tables are updated with information</td>
<td>Dimensions or fact tables where the information is being updated exist. Dimensions have been loaded correctly</td>
<td>Attributes are updated in the valid, Response_processing tables</td>
<td>Check if foreign key attributes from the dimensions, which are used during loading the fact tables exist. Solution: Record the missing foreign key attributes from the</td>
<td>Monitor_loggingInsertions: How many rows have been updated in the fact tables?</td>
</tr>
<tr>
<td>Table</td>
<td>Tables:</td>
<td>The corresponding column is being updated in the valid response time table</td>
<td>Column</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>UPDATE SET</strong></td>
<td><strong>staging_area_resultat_sw , dim_delivery_system dim_candidate dim_item, dim_request_time</strong></td>
<td>a) Tables mentioned in the input section already exist b) Foreign key attributes used while loading the fact tables exist</td>
<td><strong>Conducted Experiments</strong> from valid_response_processing table is updated with information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Conducted Experiments</strong></td>
<td><strong>staging_area_resultat_sw , dim_delivery_system dim_candidate dim_item, dim_request_time</strong></td>
<td>a) Tables mention in the input already exist b) Foreign key attributes used while loading the fact tables exist</td>
<td><strong>Succeeded Experiments</strong> from valid_response_processing table is updated with information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Succeeded Experiments</strong></td>
<td>-Succeeded Experiments, -Conducted Experiments</td>
<td>a) columns succeeded_experiments and conducted experiments have been loaded correctly</td>
<td><strong>Outcome</strong> from the valid_response_processing table is updated with information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Outcome</strong></td>
<td><strong>valid_response_processing</strong></td>
<td>a) valid_response_processing table exists b) Foreign key attributes used while loading the fact tables exist</td>
<td><strong>LastTrial</strong> from valid_response_processing table is updated with information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>UPDATE SET</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LastTrial</strong></td>
<td><strong>valid_response_processing</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tasks that are affected by this operation (‘Valid’ Fact Table)**
<table>
<thead>
<tr>
<th><strong>UPDATE</strong></th>
<th><strong>SET</strong></th>
<th><strong>valid_response_processing</strong></th>
<th><strong>The output represents the number of previous trials that have been performed</strong></th>
<th><strong>Column</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Previous_Trials</strong></td>
<td></td>
<td></td>
<td>a) valid_response_processing table exists</td>
<td>Previous_Trials from valid_response_processing table is updated with information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>UPDATE</strong></th>
<th><strong>SET</strong></th>
<th><strong>valid_response_processing</strong></th>
<th><strong>Numeric/Percentage value</strong></th>
<th><strong>Column</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FirstTrial_Outcome</strong></td>
<td></td>
<td></td>
<td>a) valid_response_processing table exists</td>
<td>First Trial Outcome from valid_response_processing table is updated with information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>UPDATE</strong></th>
<th><strong>SET</strong></th>
<th><strong>valid_response_processing</strong></th>
<th><strong>Numeric/Percentage value</strong></th>
<th><strong>Column</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PreviousTrial_Outcome</strong></td>
<td></td>
<td></td>
<td>a) valid_response_processing table exists</td>
<td>PreviousTrial Outcome from the valid_response_processing table is updated with information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>UPDATE</strong></th>
<th><strong>Tables:</strong></th>
<th></th>
<th>a) Tables</th>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET</td>
<td>Previous Related_Items</td>
<td>Table: valid_response_processing, dim_item, tematiquesque s tions</td>
<td>The output suggests if all the thematic for the exercises have been tried or not</td>
<td>previous_related_item from the valid_response_processing table and is updated with information</td>
</tr>
<tr>
<td>UPDATE SET</td>
<td>Thematic Tried</td>
<td>Table: Tematiques questions, Valid_response_processing</td>
<td>a) Table Tematiquesquestions already exists b) Foreign key attributes from the dimensions exist c) The data type of the values are boolean</td>
<td>Column all_thematics from valid_response_processing table is updated with information</td>
</tr>
<tr>
<td>UPDATE SET</td>
<td>Collisions</td>
<td>Table: Response_processing</td>
<td>a) response_processing table already exist b) Foreign key attributes used while loading the fact tables, exist</td>
<td>Column collisions from response_processing table is updated with information</td>
</tr>
<tr>
<td>UPDATE SET</td>
<td>Difficulty</td>
<td>Table: Dim_Item</td>
<td>a) dimension item has been created and contains the difficulty column</td>
<td>Column difficulty is updated in the dim_item table</td>
</tr>
</tbody>
</table>
b. ErrorProm_Updates

<table>
<thead>
<tr>
<th>Task name</th>
<th>Input</th>
<th>Output</th>
<th>Precondition</th>
<th>Post condition</th>
<th>Error Handling</th>
</tr>
</thead>
</table>
| UPDATE SET      | Table: Response_processing, calendar_fib, class_day                  | Response_processing       | a) response_processing table already exists and contains the closed_session column  
| Closed_session  |                                                                      |                           | b) XLS file containing calendar university year information exists            | In case the information exists, the closed_session update will be performed in the response_processing table | Check if the .XLS file containing 'calendar year' information exists. In case it does not exist the updated will not be realized. |
| UPDATE Temporary tables | Information coming from the .xls file | Calendar_fib, Class_day tables | Information from the .xls file containing data about the calendar university year may not exist | In case the information does not exist, the temporary tables will not be updated |                                                                    |