FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC)
BARCELONATECH

MASTER THESIS

A tool for integrating dynamic healthcare data sources

MASTER IN INNOVATION AND RESEARCH IN INFORMATICS

DATA SCIENCE

Author: Meysam Zamani Forooshani

Tutor: Oscar Romero Moral
Department of Service and Information System Engineering (ESSI)

Advisor: Oscar Flores Baquero
CEO at Made of Genes (Genomcore)

Co_Advisor: Petar Jovanovic
Department of Service and Information System Engineering (ESSI)

June 22, 2020

Abstract

Currently, there is a growing interest in finding a comprehensive method in ETL process technology that can deal with healthcare data formats. Healthcare centers are very concerned about their sensitive information which mostly has patients' data inside. they avoid using the market-driven ETL tools because they need higher performance and adaption for complex security treatment. On the grounds of this, they are in great need of a stable ETL tool that does their transformation for them privately. One of the common data formats used by various healthcare providers for the exchange of clinical and administrative data among software applications is Health Level Seven (HL7). HL7 refers to a series of international standards that are used to move clinical and administrative data between applications by different healthcare providers. Typically, hospitals and other healthcare providers have several different data structures that are used for everything from billing reports to monitoring clinical data for patients. Standardized HL7 format is one of the widely used cases, with information about the patient identification and results of patient tests. Data analysts in the bioinformatics department typically need to analyze and review these kinds of information carefully. HL7 has a standard format for describing each of this information but it is a time-consuming process to understand and prepare them for analysis.

In this thesis, a powerful ETL tool named `ETL_for_HL7` is proposed to deal with this type of data. Usually, HL7 data is made available in the XML format file by the laboratory. The main objective of this thesis is to understand the HL7 data format, design a proper data model, create a valid conceptual schema and convert it to a physical data warehouse, implement an ETL tool in the way that the laboratory saves it and sends it to the data management department, transform it into analysis-ready formats (e.g., DW schema) and finally load it into the data warehouse.

The tool starts working by getting the patient's ID and the date of patient's test as parameters. It selects the relevant XML file from the input folder. In the second phase, it starts transforming it from encoded XML that has an HL7 data format within to JSON format file and then extracts two types of information from the entire response. They are patient identification and bio values of the patient's test result. The third phase starts when the results are ready in JSON format. The data can now load into the data warehouse that we previously implemented. The data is able to be used by any analysis department and this was our main objective.

Acknowledgements

First of all, I would like to express my sincere thanks of gratitude to Prof. Petar Jovanovic for being my master thesis advisor. I am very grateful for his guidance, advice, suggestions, warnings and support that he has given me during the last six months when I have been developing this thesis.

I would also like to acknowledge Prof. Oscar Romero for being my master thesis tutor and I am gratefully indebted to his for his very valuable comments on this thesis.

Secondly, I would also like to thanks to all my family for their support during this time. Especially to my parents for their love, understanding and continuous support.

Finally, last but by no means least; also to everyone in the Genomecore (Made Of Genes) company. it was great sharing premises with all of you during the last year.

Contents

List of Figures

List of Tables

List of Abbreviations

API Application Programming Interface.

BI Business Intelligence.

CDA Clinical Document Architecture.
CEO Chief Executive Officer.
CSV Comma-separated values.

DBMS Database Management System.
DNI Documento Nacional de Identidad.
Dr Doctor.
DW Data Warehouse.

EHR Electronic Health Record.
ELGA Electronic Health Records System.
EM TEP Emergency Management Tracking of Emergency Patient.
ETL Extract, Transform, Load.

FHIR Fast Healthcare Interoperability Resources.

HL7 Health Level Seven.

i2b2 Informatics for Integrating Biology and the Bedside.

JSON JavaScript Object Notation.

NCPDP National Council for Prescription Drug Programs.

Prof Professor.

UML Unified Modeling Language.

XML Extensible Markup Language.

Chapter 1

Introduction

In this chapter, it is explained what this thesis is about, the main objective and motivations of developing it, and the general structure of this document.

1.1 Context

Healthcare systems need to exchange information outside and across boundaries to provide better care for patients. With the development of our healthcare systems, sharing patient information between different institutions is much more important.

The Health Level Seven International (HL7) specification which is standard for the exchange of medical electronic data, helps us solve this issue. It specifies how healthcare systems can share medical information electronically. The original purpose of the HL7 standard is to provide a framework for developing medical messaging systems, but there is an increasing number of initiatives applying HL7 methodology and structure to their entire medical information system [9].

In a centralized electronic health record system (ELGA[1]), HL7 is used to exchange medical records with key components to identify patients, health care providers, and authorization management. There are currently four different forms of records available in this system (i.e. Discharge Summary of a Physician, Nursing Discharge Summary, Laboratory Report and Diagnostic Imaging Report) [10]. For this thesis, HL7 version 2 based on the Extensible Markup Language (XML) is considered to be a data sample. You can find full detail on the various versions of HL7 in section 2.2.

One of the main healthcare challenges is to provide a comprehensive analysis of the clinical care process that involves the integration of clinical information provided by heterogeneous information systems established using various technologies, for different disciplines and purposes, and by various organizations [11]. This requires the implementation of a process entirely devoted to extract data from heterogeneous data sources, clean and restructure as per the required standard, and finally load them into a DW optimized for data analysis purposes [1]. This is a well known process called ETL (Extract Transform Load). In the Business Intelligence (BI) workflow, the ETL process is a vital procedure which "is the most challenging aspect of BI, requiring about 80 percent of the time and effort and generating more than 50 percent of the unexpected project costs" [12].

Health level 7 (HL7) has already addressed the issue of standardizing data in healthcare, which provides standards for enhancing communication between healthcare organizations [1]. However, There is no full implementation of an ETL method in the healthcare systems, using physical DW approach, for answering HL7 version 2 messages. Just a few researchers have put

[1]ELGA stands for "elektronische Gesundheitsakte" (electronic health records). ELGA is an information system that simplifies the process of accessing your health records for you and your doctors, as well as other health care professionals at hospitals, care facilities and pharmacies. Health data such as a patient's test results are generated by a variety of health institutions. ELGA networks all of them and makes them available digitally by means of a link. Website: http://www.elga.gv.at

some work into effect. In this thesis, it is desired to give an implementation that can be used for any HL7 version 2. Laboratory report is one of the use cases that usually transfer with HL7 messages. Laboratories use it to send patient identification and test results to analysis departments. Analysts need to extract the patient's bio values from the HL7 message represented in the XML file format. They have to spend a lot of time getting details through a long HL7 message that corresponds to the patient. This process can be performed by an ETL tool for transforming HL7 data and make it accessible in a data warehouse.

Data is typically transferred from routine systems to a data warehouse using an ETL process. [13, 14]: (1) data is extracted from source systems, (2) cleaned, harmonized and transformed into a form suitable for analyses, and (3) loaded into the data warehouse. To manage the complexity of such processes, they are often implemented using unique environments that provide connector libraries to different types of sources, transformation operators that make our information meaningful, and finally connects to different types of solutions for data storage.

To that end, we can use market-driven ETL tools or create our ETL process. ETL tools have not been commonly adopted by healthcare centers with large datasets. Each ETL tool has its own unique logic, and perhaps some applications may find it restrictive. But hand-coded programming comes with limitless possibilities, and that way is very flexible. on the other hand, market-driven ETL tools are often not dynamic enough to tackle complicated processing requirements, improve the performance of the process and adaption for complex security treatment. The clarity of the corresponding SQL commands performing the transformation is often missing, making it difficult to resolve transformation errors [15].

That being the case, we would create an ETL process, implemented in python3 programming language, that can use in healthcare systems for managing with distributed clinical data in HL7 format. Data scientists and engineers can use the results of this study and efficiently generate their new version of the ETL process.

1.2 Motivation

I did 1 year internship at the bioinformatics company called Genomcore[2]. During my time there, I found that they were spending a lot of time to understand and transform HL7 messages and to prepare them for analysis. We were thinking about creating an ETL process to avoid data analysts wasting 80 percent of their time on pre-processing data and constantly performing the same transformations [16]. One of the most common data, they use for part of their analysis is HL7 messages related to patient's test results. HL7 messages are sent to them in an encoded XML format by the laboratory. There is a gap between receiving data from laboratories and using it by analysts. A proper ETL process created specifically for managing HL7 data can fill that gap. I choose this topic as my final master thesis on the basis of three main reasons:

- First of all, since the outcome of this master thesis can be used in any healthcare data management department to provide patient information almost ready for analysis without even opening HL7 data.

- Secondly, the subject has been studied very well before and it shows that it may be an academically successful subject. You can find some of these mentioned works in the chapter 3.

- Thirdly, since it relates to healthcare services, my feeling that I am doing this project is something that can indirectly help people, and it makes me happy and motivated to complete this project and make it available for future work. Especially these days, more than before, because of the Coronavirus outbreak, we need a faster healthcare system.

[2]Genomcore is a company that provides an interoperable framework between diagnosis providers, healthcare profesionals and end-customers. Website: https://genomcore.com

1.3 Objectives

The main objective of this thesis project is to develop a methodology for the design of Extract, Transform and Load (ETL) process in a clinical data warehouse system based on the Electronic Healthcare Record (EHR[3]) which it potentially would be an ETL tool by adding other response type which I have listed them in Figure 2.1. This approach take advantage of EHR architecture as one of the main sources of information to feed data warehouse, also taking into account that clinical documents provided by heterogeneous legacy systems are structured using the HL7 standard. As you can see in Figure 2.1, there are different kinds of queries that EHR can request and obtain a response in the HL7 message. In this thesis, we are focusing on clinical laboratory results of patients, but being extensible to include other types of HL7 response to the proposed ETL tool as a future work.

At the end of this thesis project, we will have an ETL tool named `ETL_for_HL7`, to transform HL7 messages based on the Extensible Markup Language (XML) into an analysis-ready format structure and load it into a Data Warehouse. This goal is very general and to accomplish it, it has been divided into several concrete objectives:

1. Find and document the HL7 segments definition that can help us to understand the HL7 functionality better.

2. Model the data and design the schema for creating a physical data warehouse.

3. Conceptually model the ETL process. The conceptual design should be modeled at two different levels: overview of whole process and ETL process itself. Such representations would conceptually provide a full description of the data flow and control flow.

4. The ETL process implemented by python programming language that meets all nun-functional requirements defined in section 1.3.1.

5. Build a wrapper script that would allow the user to run the ETL tool to request specific patient observation results.

6. Validate the ETL tool with sample data from Genomcore company (the data are fake because of the sensitivity of the patient's test results).

1.3.1 Non-functional requirements

The professional design of an ETL process needs to rely on the design of underlying processes from an end-user perspective that represents business requirements. In Table 1.1, we have proposed a list of characteristics for ETL process quality that coherently accumulate concepts from the fields of data warehousing, ETL, data integration, and software engineering which describes the non-functional requirements specific for our system [6].

Table 1.1: Non-functional requirements - source [6]

Name	Requirement
Flexibility	The ability of the ETL flow to provide alternative options and dynamically adjust to environmental changes
Reusability	The ability to use the ETL process for the operations of other processes.

[3]An electronic health record (EHR) is the systematized collection of patient and population electronically-stored health information in a digital format. These records can be shared across different health care settings. Website: https://www.healthit.gov

Understandability	The clearness and self descriptiveness of the ETL process model for (non-technical) end users.
Maintainability	The ability of effectiveness and efficiency with which the ETL process can be modified to implement any future changes.
Testability	The ability to which the process can be tested for feasibility, functional correctness, and performance prediction.

1.4 Initial planning

The thesis project has been performed during 5 months, between January 2019 and June 2020. 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 1.2, main activities that I have initially considered.

Table 1.2: Initial planning

Activity	Start Date	End Date
Study the initial material and get familiar with the HL7 messages.	20-January	5-February
Implement structure and prepare an environment in the python programming language.	5-February	15-February
Complete implementation of extraction and transformation steps.	15-February	10-March
Improve the ETL design in transformation, especially in validation step.	10-March	25-March
Start data modeling, and create target DW schema in a UML diagram.	25-March	5-April
Install PostgreSQL locally and create a physical data warehouse in it.	5-April	15-April
Loading data into PostgreSQL and perform several tests.	15-April	25-April
Start the thesis writing process.	25-April	25-May
Apply additional changes suggested by supervisor and finalize the thesis writing.	25-May	10-June

1.5 Document structure

My master thesis is organized in the following way:

- **Chapter 1: Introduction.** In the first chapter, the problem that we are going to study and the main objectives of this thesis as well as motivation are explained.

- **Chapter 2: Preliminaries.** In the second chapter, some of the preliminaries concepts used in this thesis are explained. They should be understood in order to fully understand this document.

- **Chapter 3: State-of-the-Art.** It is explained the current literature of the ETL process for distributed data, in particular, the healthcare data and HL7 messages.

- **Chapter 4: General overview of the project.** It is presented the design of the project and the methodology used.

- **Chapter 5: Implementation and Use.** It is given an overview of how it was implemented and how to use the created ETL tool.

- **Chapter 6: Results.** In the result chapter, the loaded data obtained with the ETL tool are presented and explained.

- **Chapter 7: Conclusion.** It is explained the main conclusions of this thesis.

- **Chapter 8: Future work.** Finally, there are some of the future works that could be done after this thesis.

Chapter 2

Preliminaries

In this section, the HL7 message structure (syntax) and content (semantics) related to this thesis are briefly explained. Additionally, an interchange format and the encoding rules for HL7 version 2.x message instances based on the Extensible Markup Language XML will thoroughly be explained. These concepts are essential and should be understood in order to understand the full description of this document. Firstly, the idea of the HL7 standard is explained. Secondly, it is briefly introduced differences between HL7 versions.There are mainly 4 versions that are version 2 messaging, version 3 messaging, Consolidated-Clinical Document Architecture (CDA), and Fast Healthcare Interoperability Resources (FHIR). The version used in this thesis is version 2 that I am going to explain in detail.

2.1 The Health Level Seven International (HL7)

For most health systems, the electronic health record (EHR) is the primary base for citizens' wellbeing. The development of electronic health records at national level is faced with problems such as the expansion and inconsistency of quality of record data, difficulty in creating a specified and standard record structure due to variance and multiplicity of structure, lack of specific medical terminology, and challenges relative to privacy providing answers to such problems is very critical and requires a type of electronic health record design (architecture). As a matter of fact, the architecture of the electronic record is required to establish a structure for the effective implementation of electronic government in health sector. This is one of the essential types of architecture that related to electronic health records applied schedule and technologies. José Maldonado et al, stated that various international bodies have focused on the concept of electronic health record architecture, resulting in attempts to define such standards of architecture as the Health Level Seven International (HL7) [17]. HL7 is an international standard that has been established to enable the interoperability of health information technology. It initially focused on the compatibility within large hospitals among information systems. The organization also started to concentrate on integration between systems in various organizations, including public health. [18].

The organization's early efforts included improving communications terminology and a structured vocabulary. It is not the only method used to transmit data relating to health. Many entries include:

- NCPDP[1] (National Council for Prescription Drug Programs) for ordering medications.[19]
- EM TEP[2] (Emergency Management Tracking of Emergency Patients) for tracking health information for patients in transport.[20]

[1]NCPDP Standards create and promote data interchange standards for the pharmacy services sector of the healthcare industry. Website: https://ncpdp.org

[2]Tracking of Emergency Patients (TEP) Messaging Standard assists emergency responders, coordinators and administrators in the exchange of emergency patient and EM tracking information, thus increasing the effectiveness of emergency medical management, patient tracking, care management and family notification. Website: https://www.oasis-open.org

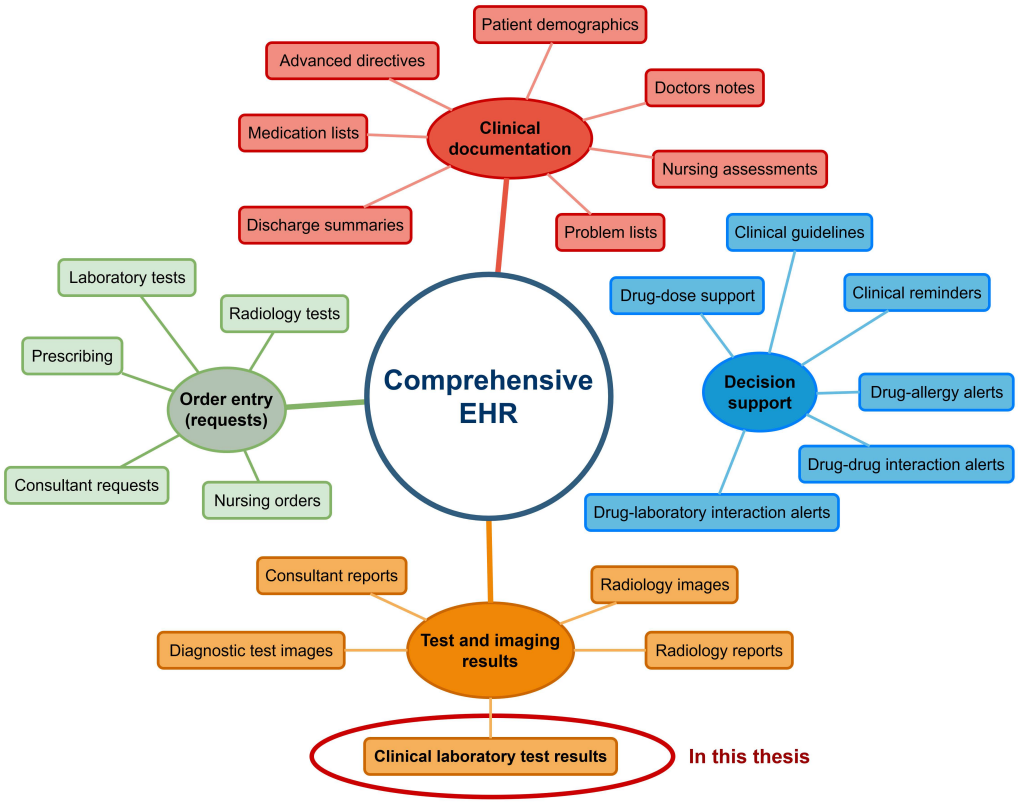

Figure 2.1: Various kinds of queries that EHR can request

Although both NCPDP and EM TEP are different specifications, the definitions can be mapped to HL7. The HL7 is considered the standard for health data communication. It has gone through rigorous validation and approval process as a standard accepted by ANSI (American National Standards Institute). HL7 standards continue to grow for the transmission, management and integration of information on electronic healthcare. With the implementation of the "Meaningful Use" system, the use of HL7 received a significant boost. This included the use of particular HL7 implementation guides for lab results, cancers, syndromes, and immunizations produced by the public health.

HL7 is responsible for interoperability, both semantic and syntactic. Semantic criteria refer to terminology used to describe elements of data such as events, test results, and parameters. Syntactic criteria refer to how these data and related words are prepared by the sender system before sending it.

Additionally, the transport layer corresponds to the protocol that connects one system to another. Transport standards are distinct from semantic and syntactic criteria, and the transportation layer is "agnostic" of the transported data semantics and syntax (the content and format). Looking at the seven steps described below, you see how the ETL tool in this system can be beneficial and can impact progress.

1. In the real world, something triggers the cycle to begin. It is called an event activation. The cause may be any number of activities, such as a person from analysis department pressing a button on the system that orders a query from another platform to collect medical data such as results of patient blood tests.

2. Firstly, the sender prepares the transportation details. This means they collect the data they need and prepare it for transportation. They apply the required HL7 standard to

explicitly and predictably construct this package of data.

3. Next the sender connects via the transport layer to the recipient (First step of the ETL process). This step involves authenticating the data can be sent by the sender.

4. Third, the receiver collects and parses the data packet (translates it from HL7 to an internal format).

5. In the fourth step, the receiver processes the data, applying local business rules and data safety. It is an important part of the process and can cause issues if not clearly documented by the receiver and communicated to the sender.

6. The receiver then transformed the data into DW, which can be easily queried for analysis.

7. Finally, and crucially, the available data will use for departmental analysis, biomedical analysis, and exploratory analysis, which is the last step of whole process.

2.2 Multiple HL7 versions

As the use and experience for HL7 continues to evolve, several HL7 versions are now available.

- Decades ago, the HL7 organization released version 2 messages. This edition is now modified and improved and commonly used globally. In addition to the existing specifications for use of version 2, version 2.x allows unnecessary changes such as new information that is submitted to a case study. In addition, version 2.x supports query and response. For instance, the Electronic Health Record(EHR) system requests and receives a registry record of immunization. In Figure 2.1, you can find different kinds of queries that EHR can request and then obtain query response in HL7 messages.

- HL7 version 3 messaging has been widely applied internationally, but not everywhere.

- Clinical Document Architecture (CDA) is widely adopted in the U.S., but not in all other parts of the world And is in use with Consolidated-Clinical Document Architecture (C-CDA). HL7 developed the CDA on the HL7 version 3 data system.

- Fast Healthcare Interoperability Resources (FHIR) is just evolving, but it seems easy to incorporate and could be the future wave.

2.3 HL7 version 2.x messages represented in XML file

As we will implement an ETL tool for HL7 version 2, you can now assume that HL7 is identical to version 2 of HL7.

Many XML encoding could serve as an alternative syntax for HL7 messages. We deal primarily with the conversion between standard HL7 encoded and XML encoded, and define the underlying rules and principles.

There are some benefits of using XML as format for interchange. You can find a part of HL7 message example represented in XML interchange format, in appendix C. This sample is received from Genomcore company and will be the main sample data in this thesis project.

2.3.1 HL7 Hierarchical Message Structure

A specific HL7 message is a hierarchical structure and is established by a trigger, representing a real world event. A message is the atomic unit of data transmitted between systems and consists of a group of segments in a given sequence. Messages begins with the Message Header

Segment MSH and shall be defined by the type of message and the event to be initiated. In every message a three-character code identifies their type. For example, the form of ADT message is used for transmitting portions of patient administration (ADT) data from one system to another.

HL7 describes the message content as an abstract collection of data elements in data segments. Segments are ordered field sequences and can be declared optional or recurring as needed. Every segment starts with a literal three-character value defining it within a message (segment identifier). For example, the ADT message can contain the following segments: Message Header (MSH), Event Type (EVN), Patient ID (PID) and Patient Visit (PV1). You can see the order of some commonly used segments in Figure 2.2. Also included in Appendix A as additional details regarding each segment and its fields.

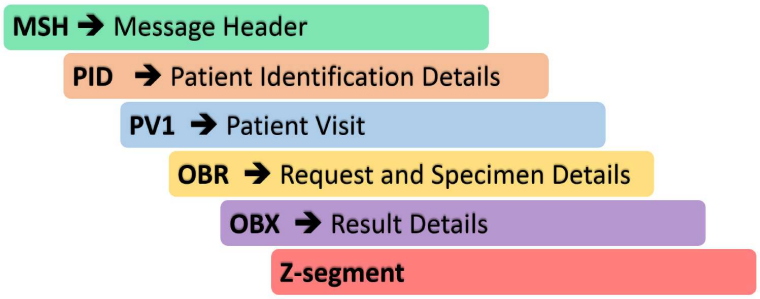

Figure 2.2: Some of commonly used segments

The semantic content of a message is passed to the segment fields. Fields can be of varying length. The contents of the fields may be required or optional, individual fields may be repeated. The individual data fields in the message are defined by their location within their related segments. Multi-component fields are used for further subdivision of the field and promote the transmission of locally relevant semantic information.

A data type is specified for every field or field portion. Complex data types consist of two or more elements. Examples are the CE data type (coded elements) of which the components are coded value, code designator and code system or the family name, which consists of several sub-components to denote the different parts of a person's name. There are a variety of data types specified in HL7 [21].

2.3.2 Messages identifications and trigger events

A key position performs the ID of the message structure That is specified in the concept of an abstract message and also given in the MSH-9 field of the message header segment. This field contains the message type, trigger event, and The message structure ID of the message (You can find complete detail of MSH segment in Appendix A under section A.1).

- The first component is a message type code that contains values such as ACK, ADT, ORM, ORU etc.
- The second component is the event trigger code with values such as A01, O01, R01 etc. Refer to the HL7 standard document for complete listing.
- The abstract message structure identification is the third component. All messages with a structural ID are conceptually the same.

Example of a Message Header segment, with message type, trigger event, and the message structure ID repeated in MSH-9:

```
1 MSH|^~\&|ADT1|MCM|LABADT|MCM|198808181126||ADT^A04^ADT_A01|M|P|
    2.4|
```

Messages use the message structure ID as a root element for the XML instance documents. The corresponding XML message fragment is shown below as an example.

```
<ADT_A01>
...(segment elements)
</ADT_A01>
```

The element <ADT_A01> carries the segment elements (see following section) as child elements.

2.3.3 Segments

Message structures include segments which are also defined as XML elements. Segments are-ordered sequences of fields. Each segment starts with a literal value of three characters, which identifies it within a message (segment identifier). For example, a segment with MSH has <MSH> as the name of the XML element, a segment with PID <PID> etc.

Considering the ADT_A01 example above, the corresponding XML message fragment is shown below.

```
<ADT_A01>
<MSH>
...(MSH field elements)
</MSH>
<EVN>
...(EVN field elements)
</EVN>
<PID>
...(PID field elements)
</PID>
...(other segment elements)
</ADT_A01>
```

2.3.4 The XML representation of an HL7 message

The XML representation discussed here represents structures of HL7 messages as elements of XML. Message structures include segments which are often represented as XML elements. Segments contain fields, which are again defined as XML elements. The data type of a field is stored as a fixed attribute in the field attribute list, while the content model of the field specifies the data type components. Various fixed attributes are used to extend the abbreviations and to indicate requirements on the HL7 table attribute. As an example, a simple message in the syntax of the standard encoding rules can be seen as bellow.(Also you can find a sample of this representation signed in Figure 2.3.

```
MSH|^~\&|LAB^foo^bar|767543|ADT|767543|19900314130405||ACK^|XX3
    657|P|2.3.1
MSA|AA|ZZ9380
```

Here is the same message in the syntax of the recommended XML encoding rules:

```
<ACK>
  <MSH>
   <MSH.1>|</MSH.1>
   <MSH.2>^~\&amp;</MSH.2>
   <MSH.3>
     <HD.1>LAB</HD.1>
     <HD.2>foo</HD.2>
     <HD.3>bar</HD.3>
   </MSH.3>
   <MSH.4>
     <HD.1>767543</HD.1>
```

```
    </MSH.4>
    <MSH.5>
      <HD.1>ADT</HD.1>
    </MSH.5>
    <MSH.6>
      <HD.1>767543</HD.1>
    </MSH.6>
    <MSH.7>19900314130405</MSH.7>
    <MSH.9>
      <CM_MSG_TYPE.1>ACK</CM_MSG_TYPE.1>
    </MSH.9>
    <MSH.10>XX3657</MSH.10>
    <MSH.11><PT.1>P</PT.1></MSH.11>
    <MSH.12>
      <VID.1>2.3.1</VID.1>
    </MSH.12>
  </MSH>
  <MSA>
    <MSA.1>AA</MSA.1>
    <MSA.2>ZZ9380</MSA.2>
  </MSA>
</ACK>
```

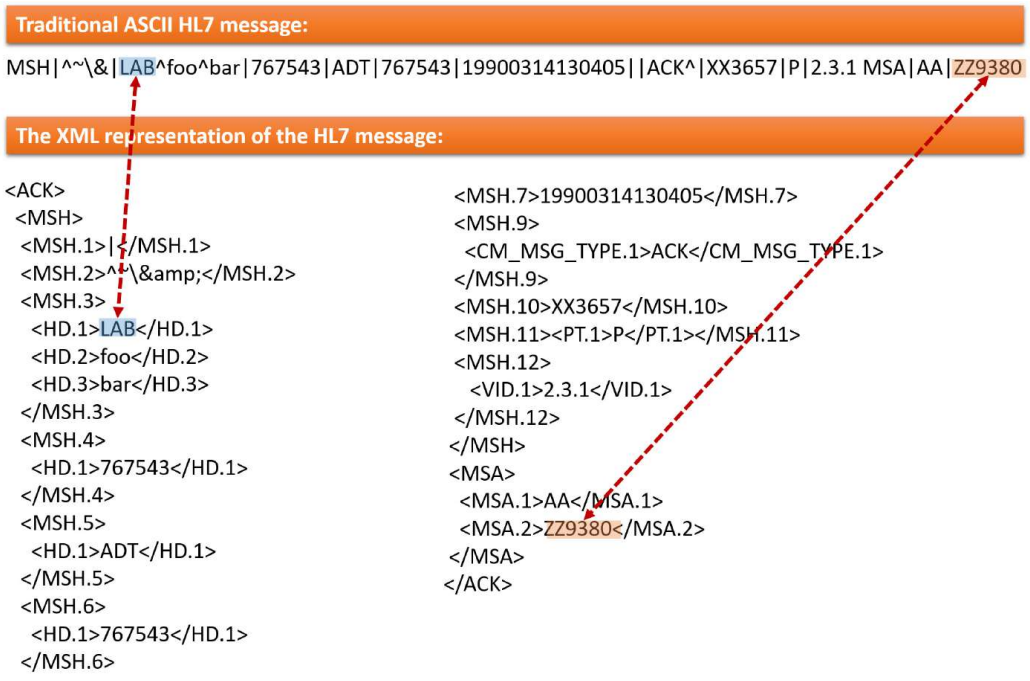

Figure 2.3: XML representation of an HL7 message

To that end, we can now understand the available data sample which is HL7 represented in XML file and make it more understandable for the entire project.

Chapter 3

State-of-the-Art

In this chapter, it is briefly explained the current literature of the ETL process for distributed data, in particular, the healthcare data and HL7 messages.

The ETL method is not a new field of data science research, quite the opposite. It has been used for years in data engineering tasks such as integration, and the use of the ETL method in health data is very widespread and plays a significant role in this area. In this thesis, we will concentrate on the ETL process applied to distributed healthcare data, in particular, the Health Level Seven International (HL7). As already stated, HL7 messages are often expressed in the XML format file that is used for transferring data between centers and healthcare departments.

The use of ETL for distributed data in healthcare systems is much less explored (less researched and used) in the data science world. One of the advantages of using the ETL tool would be helping scientists to work with data in analytics-ready format rather than using row data. In general, "the ETL process is the most time-consuming part of the data analysis process, even mentioned that the number could rise to 80% of the total project development time" [16]. Very few researchers have tried to introduce HL7 version 2 (V2) and to specify the ETL process for it. On the other hand, there is more focus on HL7 CDA (V3 standard) and HL7 FHIR in research areas and that is the reason for Gap-Meaning in the integration layer for transforming HL7 version 2. In this section, we will go through some of the previous research classified on the basis of the HL7 CDA and HL7 FHIR and briefly explain their approach for the development of the integration layer. In the third section of this chapter, we will see some previous research done on HL7 version 2, which is the focus of this thesis project, And finally, the last section presents the summary of the contributions of this thesis project.

3.1 Method for transforming HL7 CDA

The CDA is part of the HL7 version 3 series of standards. This family includes both CDA and the evolving version 3 message standards represented in Extensible Markup Language (XML)[22].

In [1], a methodology for the design of Extract , Transform and Load (ETL) tools in a clinical data warehouse architecture based on the HL7 CDA standard is proposed. As you will see in section 2.2, there are some main differences between different versions of HL7 messages. The data warehouse dimensional model based on the CDA specification can be extracted from:

- "HL7 Hierarchy defined by the triple <Participation, Role, Entity> that models subjects/objects involved in the process as well as the role played by them within the action."[1]
- "CDA Backbone composed by the specializations of the different Act classes as well as their relationships that models the actions documented in the CDA."[1]
- "Attributes of the Act class that identifies the type of action observed using vocabularies."[1]

The CDA constructs presented in [1], were used to model the logical data map shown in 3.1 where, for each business rule, the columns of each data warehouse schema table are mapped with the attributes of the corresponding CDA class. The schema is represented by the three facts reported in dark grey, surrounded by four conformed dimensions reported in light gray as well as three individual dimensions reported in white.

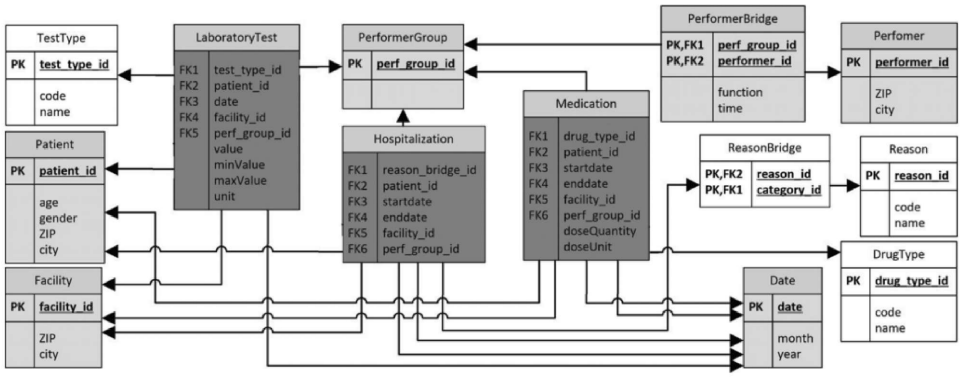

Figure 3.1: Constellation schema modeling the hospitalization, laboratory test and drug therapy business processes. - source: [1]

There are some other works that they focused on HL7 CDA standard but not Version 2 and since the data hierarchy structure is quiet different, it is not reproducible for HL7 version 2. The transformation of the HL7 version 3 standard is more complicated but easier to understand compared with HL7 version 2 [10]. The CDA document has a header and a body. The header expresses the context in which the document was created and the body contains the information. The objective of the header is to make it possible for clinical documents to be transferred across and within healthcare organizations; to improve the management of clinical documents; and to facilitate the collection of individual patient clinical documents in electronic health records [22].

3.2 Method for transforming HL7 FHIR

Whereas HL7 version 2 messaging followed an ad hoc design process and version 3 followed a tightly established stepwise model-driven process called heterochronous dataflow model (HDF[1]), FHIR uses a systematic and iterative approach for accelerating progress [23].

HL7 Fast Healthcare Interoperability Standard (FHIR) aims to fix their shortcomings in the earlier versions HL7 version 2 and version 3. It specifies the resources of distinct healthcare organizations. Based on the principles of `RESTful` architecture, it allows easier, simpler and more stable adoption of HL7. Figure 3.2 shows an example of FHIR 'Patient' resource as JSON object [2].

In [2], they proposed a system that uses HL7 FHIR standard for maintaining maternal health records as FHIR resources. `MongoDB`, a `NoSQL` data store serves to enable powerful data record manipulation. `NoSQL` data stores are schema free. However, to store the resources as JSON documents in `MongoDB`, an underlying structure (Schema) had to be defined. In this example, a database created to contain a set of collections and each collection stores multiple JSON documents. For example, the Patient collection holds FHIR Patient resource documents. To access the prototype system, a `RESTful` framework is developed. This would establish the basis for a structured information system on maternal and child health care. The proposed architecture is represented in Figure 3.3.

[1]The Heterochronous Dataflow (HDF) domain, created by Ye Zhou, is an extension of the Synchronous Dataflow (SDF) domain. In SDF, the set of port rates (called rate signatures) of an actor are constant. In HDF, however, rate signatures are allowed to change between iterations of the HDF schedule. Website: https://ptolemy.berkeley.edu

```
{
  "name": [{ "use": "official", "text": "Sarah", "family": ["Bor"], "given": ["Sarah Olaf"]}],
  "gender": { "coding": [{ "system": "http://hl7.org/fhir/v3/AdministrativeGender", "code": "F",
"display": "Female" }] },
  "telecom": [{ "system": "phone", "value": "+31612345678", "use": "mobile" },
        {"system": "phone",  "value": "+31201234567", "use": "home" }],
  "birthDate": "1960-03-13",
  "address": [{ "use": "home", "line": ["Bos en Lommerplein 280"], "city": "Amsterdam", "zip":
"1055RW", "country": "NLD" }],
  "managingOrganization": { "reference": "Organization/1", "display": "Burgers University Medical
Centre" },
  "active": true
}
```


Figure 3.2: FHIR patient JSON example - source: [2]

Data Access Objects (DAOs) provide an architecture for the `MongoDB` server that characterises it. In order to store the JSON-formatted strings received from clients, the DAOs validate it, add identifiers and transform them for storing in `MongoDB`. DAOs also map the requests to queries on the `MongoDB` database and return the results. The Repository of `MongoDB` is the back-end storage system. The JSON objects received from the client side are converted into JSON documents by the DAOs, transformed and finally loaded to the `MongoDB` repository for storing in the appropriate collection (Figure 3.3).

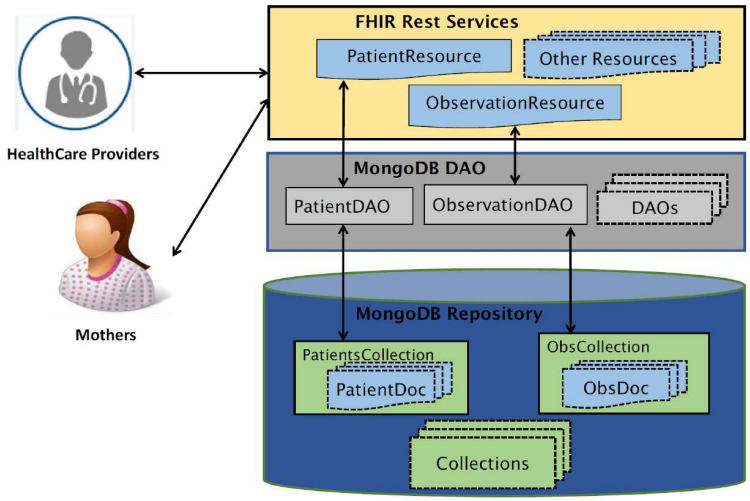

Figure 3.3: Proposed maternal health record system architecture - source: [2]

An ETL method based on the HL7 FHIR format was also proposed in [3], by identifying common similarities between various health data entities and HL7 FHIR resources. In short, the preferred method implies implementing ontology transformation operations in various HL7 FHIR datasets and storing the results in a knowledge base in the form of triple information. As a result, a direct mapping is performed between the attribute data in the source dataset and the HL7 FHIR resources by performing matching operations running either in series or parallel to discover the semantics and/or the nature of the data elements. Several methods have also been implemented to facilitate ETL over healthcare data, to gather the results of the ETL process, to make decisions quicker and more efficient. The structural mapping process consists of four different layers, following a waterfall approach as presented in Figure 3.4.

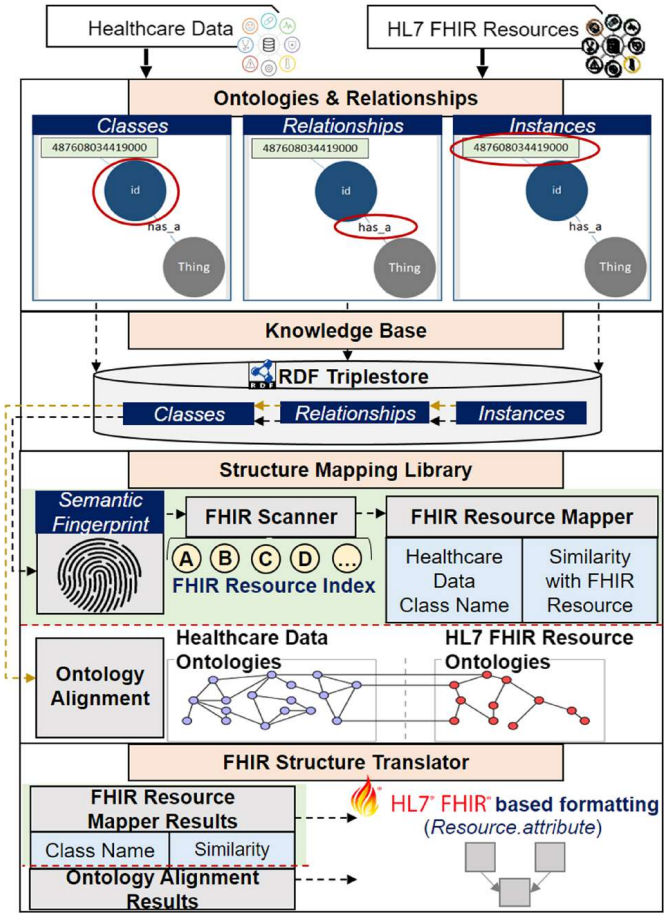

Figure 3.4: Layers of the structure mapping mechanism - source: [3]

The first layer is Ontologies and Relationships. The creation of ontologies would be the same as deriving the relationships, classes, instances, and transferring this to triples. The second layer is the Knowledge Base that provides a data store based on relationships to store the ontologies and relationships defined from the previous layer. More specifically, the triple-store RDF is implemented that can store the identified: (i) relationships, (ii) classes, and (iii) instances of the created ontologies, making it easier to perform queries through the gathered data which may contain information about one or more of the stored data. The third layer is the Structure Mapping Library, which provides a method that offers the capacity to recognize and interpret the semantic context of the various classes already stored in the Knowledge Base layer. In more detail, according to its particular instance and the relationship that occurs between the class and its instance, each different class is related to a specific semantic meaning based on their semantic fingerprint. The FHIR Structure Translator, more precisely, obtains the classes along with their identified HL7 FHIR resources and converts them into the correct HL7 FHIR type. The mechanism obtains the HL7 FHIR resource and the specific attribute to which the class may belong and translates it, using formatting based on the HL7 FHIR [3].

3.3 Previous research on HL7 version 2 (V2)

The HL7 messaging standard version 2.x (V2) is now the most widely used solution for clinical domain data exchange. Whereas the HL7 version 3 and FHIR version cover a very small fraction of real-world interfaces [23].

In [4], a Semantic-driven engine called SeDIE proposed for semantic integration of heterogeneous and distributed health data sources. SeDIE enables the integration of data from different healthcare data sources in order to reformulate a patient's entire medical record and to efficiently query patient data. SeDIE 's efforts are summarized in the following points:

- To integrate structured and unstructured data, it identifies patterns from the HL7 version 2 segments to be filled with information extracted from a free medical text.

- An algorithm proposed and implemented that efficiently transforms data from HL7 version 2.x messages into RDF that integrates data efficiently by combining data from patients including data from illness, medical, and environmental.

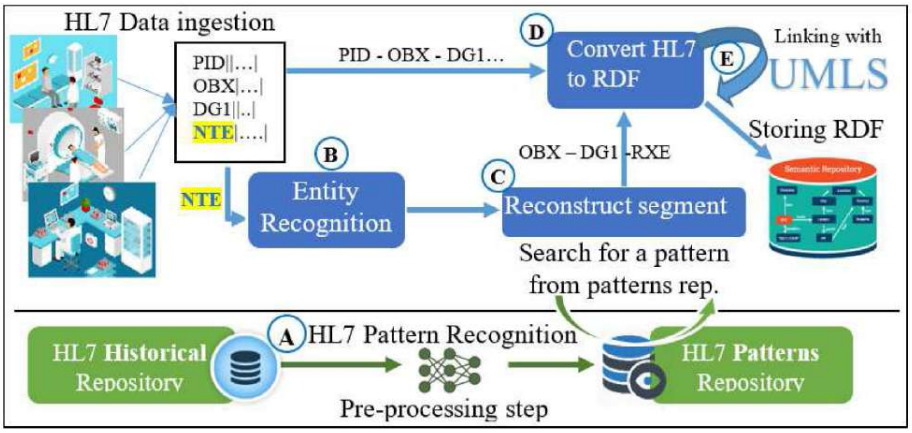

Figure 3.5: SeDIE – A semantic-driven engine for health data integration - source: [4]

Various approaches have been applied to extract RDF from unstructured data. SeDIE consists of several components programmed to handle various functions such as pre-processing and data processing. SeDIE high-level architecture is shown in Figure 3.5 that performs pattern recognition tasks (Figure 3.5 A) during the preprocessing phase of the data. The three components used to perform tasks that are: the HL7 message repository, the pattern recognition engine, and the HL7 pattern repository. The data ingestion system allows data to be consumed into various modules, such as the entity recognition module and the HL7 message converter. Messages are consumed in text format and can involve structured data such as patient diagnosis, test results, etc., and unstructured data such as medical records, pathology reports, radiology imaging. The entity recognizer (Figure 3.5 B) identifies healthcare entities in the messages. A search interface (Figure 3.5 C) is provided to allow a query pattern in repository patterns and to reconstruct a structured segment. The converter (Figure 3.5 D & E) transforms the HL7 message segments into RDF triples and consolidates all the coding used in the messages according to a single standard coding scheme. In the end, the RDF triples are stored in the triplestore. The new representation of HL7 data and triplestore enables semantic querying and intelligent data retrieval in research-oriented scenarios [4].

In [5], a real-time data import processing of HL7 messages proposed for the i2b2[2] data warehouse. The proposed real-time HL7 data import module is developed using the Java SE 6 development kit. The local integration service provides all relevant information via a network stream using the basic layer protocol as defined by the HL7 standard. A specific network port is used by the developed software to obtain all messages that are then passed through a "clean up" filter. By using regular expression search and substitution functions, arbitrary message segments are omitted and proprietary fields are corrected. In the next stage, standard HL7-compliant messages are forwarded to the HAPI parser that facilitates semantically enhanced access to individual information. By using the HAPI module, multi-valued concepts are aggregated and translated to Java objects. Finally, a database layer uses the JDBC library to load semantically improved observations into the i2b2's Oracle database. one HL7 message contains one or more clinical facts, each of which results in one or more rows in i2b2's central `observation_fact` table. Since discovered messages may contain clinical concepts (e.g. diagnoses, laboratory values)

[2]Informatics for Integrating Biology & the Bedside - The i2b2 tranSMART Foundation is a member-driven nonprofit foundation developing an open-source / open-data community around the i2b2, tranSMART and OpenBEL translational research platforms. Website: https://www.i2b2.org/

that were previously unknown to the data warehouse, the system identifies new concepts and inserts relevant rows into the `concept_dimension` and ontology table. Consequently, a received fact will be made available to the end user immediately via the query interface of the i2b2. The presented information flow is represented in Figure 3.6.

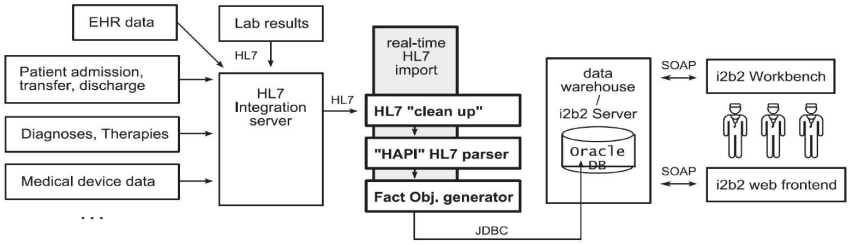

Figure 3.6: Flow of information from healthcare systems to DW's end users. - source: [5]

3.4 Contributions

In general, all research follows the design of [4], where the ETL is a two-layer implementation in which the first layer proposed the extract, transformation, and load for HL7 messages, and the second layer validates HL7 messages with Pattern Recognition system and repairs it in pre-processing step. This method is a kind of semantic-driven approach for healthcare data sources and my approach uses such a method in this project (in section 4.3, the ETL process methodology is explained in full details). The ultimate goal of this project was to help the analyst of the healthcare systems, by developing an efficient ETL process in order to prepare patients' data for exploitation purposes like aggregated studies (for example to explore normality levels, population trends, etc). Therefore, initially, I had to take into consideration the information that has been previously gathered. This information has helped me to define my starting point in this thesis project. Referring to the previous researchers' approaches, we are going to define our contributions in this work:

Firstly, we have decided to design a DW using relational DBMS (you can find all assumptions about decisions made in chapter 4.1), while some previous works rely more on NoSQL and, in particular, graph-based approach. There have been several other works that have agreed to apply their ETL method for the physical DW architecture and, in particular, with DBMS. Since our data model will not change a lot in the future due to the predefined segment structure we have in HL7 format files, we decided to use a data warehousing approach that brings us some higher performance because of strong query optimization ability that it has for the future works.

Secondly, many of the works did not use the market-driven approach to the creation of ETL tools. Succinctly, ETL tools have not been commonly adopted by healthcare centers with large datasets. Each ETL tool has its own unique logic, and some applications may find it restrictive. In many cases, it is difficult to create our solutions with the standard components included in the ETL tools. Construction of a solution based on (complicated) combinations of the supplied components or the integration of custom-coded components into the ETL program is quite time-consuming. For other issues, it can also be much easier to express the required operations in some lines of code rather than drawing flows and setting items in dialog boxes [24]. They are quite often not dynamic enough to tackle complicated processing requirements, enhance process performance, and adapt to complex security treatment. However hand-coded programming comes with limitless potential, and this way is very flexible and extensible to other requirements or other data sources that may come from EHR.

From this standpoint, we are going to improve the previous approaches by designing an ETL process, specific to work with EHR data (focusing on the HL7 format) in order to transform, integrate, and prepare such data for the final analysis. Besides, we design an innovative DW schema that enables multidimensional analysis of the patient's data.

Chapter 4

General Overview of the Project

Initially, one of the main aims of this project was to support data analysts from various health centers to analyze patient's test results, working with HL7 to analyze their data. As I mentioned in section 3.4, We are going to create a hand-coded ETL tool that meets their requirement in healthcare systems.

This chapter provides descriptions of the various processes involved in the project. This includes three main sections:

- ***Section 4.1*** provides an overview of the entire process. In this section, I will explain the whole process that shows the data movement on which it is.
- ***Section 4.2*** explaining the methodology used for data modeling, how the schema is established, and the creation of the physical data warehouse.
- ***Section 4.3*** explain the ETL process methodology and all the steps of `ETL_for_HL7` which is the main purpose of this thesis.

4.1 Overview of the whole process

Within this section, I will explain the overview of the entire procedure. As you will see, I modeled such a process in a diagram that you can find in Figure 4.1. I designed it in three main layers, the Medical Laboratory (Data Source Layer), Health Data Management (Integration Layer), and Medical Analysis (Exploitation layer). A brief description of each step presented below.

Medical Laboratory: This is the phase at which the source data is produced. Depending on the advice of your doctor and what's going on with your health, your doctor can ask for one or more medical tests that may include one or more particular patient samples. Patients go to a medical laboratory to take a blood test. In general, a clinical/medical laboratory is the place where clinical specimen tests are conducted. Such tests shall be carried out in order to obtain such clinical information as they may correspond to the diagnosis, treatment and prevention of a patient's illness.

Healthcare Data Management: Healthcare data management is a complex process. It consists of several core components such as data management, data integration, data enrichment, data extraction, data transformation, data storage, and data security. Healthcare initiatives have led to the implementation of electronic health records (EHRs) that can help clinicians and health care providers manage data more effectively and conveniently in the HL7 format. On the other hand, there has been an exponential increase in the volume of information and a need for analytical, clinical, and business intelligence resources to turn the data into meaningful information.

Medical Analysis: And the third phase is medical analysis. Data analysts are query the target schema for exploitation purposes, not the source schema. That is why we removed the heterogeneity of the sources (HL7 messages) by providing a single source of truth.

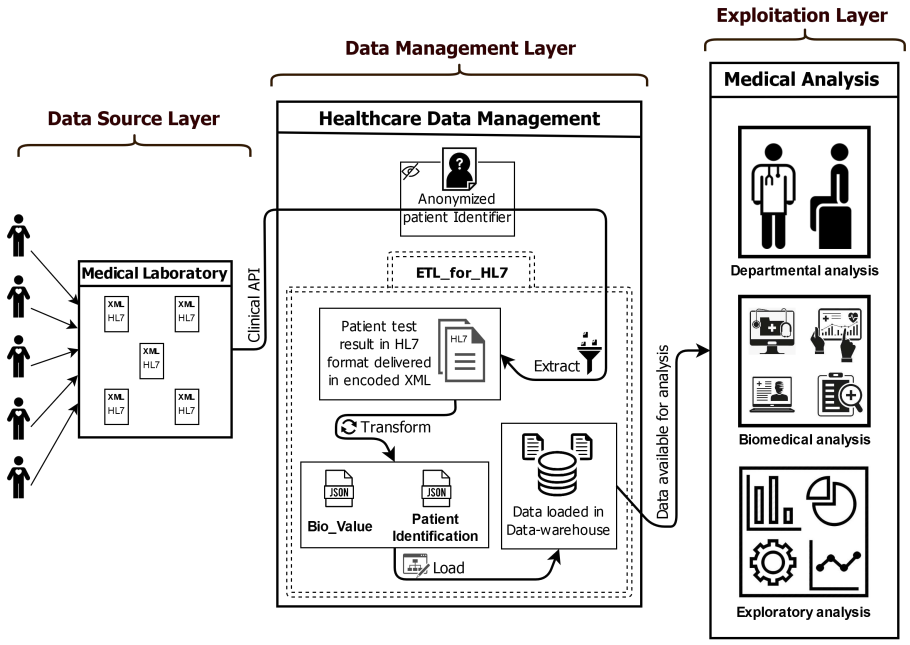

Figure 4.1: Overview of the whole process

4.1.1 Medical Laboratory (Data Source Layer)

The patient's blood test is normally collected from a medical laboratory and provided for analysis. In this procedure, the patient must first go to the medical laboratory to conduct a blood test. Medical laboratories often use the HL7 format file to save their blood test results as well as patient identification information (Figure 4.2). For the transmission of such HL7 messages to the analysis, they mostly use the XML representation of HL7 messages which we explained in detail in the chapter 2. They have to send this XML file to the analysis department, but to make this data ready for analysis, they first send it to the healthcare data management via an API.

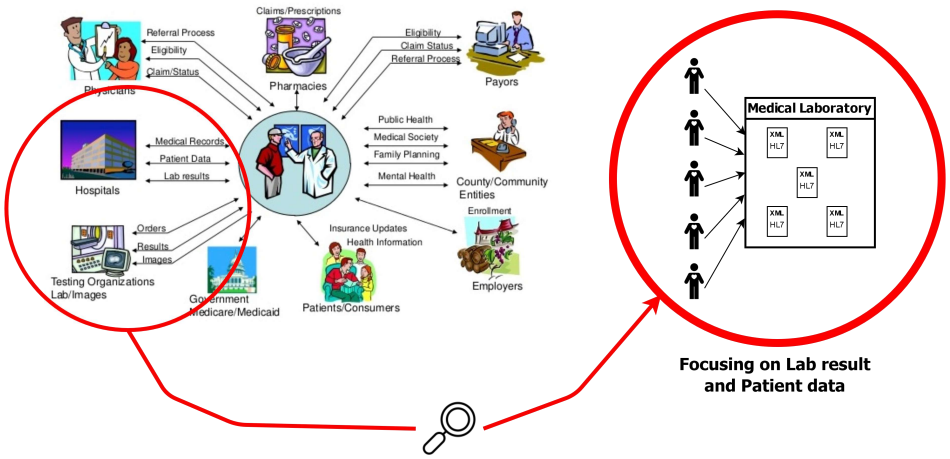

Figure 4.2: Data source layer in healthcare system

4.1.2 Healthcare Data Management (Data Management Layer)

Now the data is in healthcare data management and ready to received by the ETL tool that we are implementing in this thesis. Before proceeding via the ETL tool, there is another stage to

guarantee security and privacy called the anonymized patient identifier. The data must pass this protection step to guarantee the security of patient's data. The confidentiality of patient health information is important in healthcare Data Management process.

Anonymisation or de-identification are referred to the mechanisms by which a data custodian creates, manages and distributes a data set not containing individually identifiable information to the data receiver (Table 4.1). De-identification of medical record data refers to the removal or replacement of personal identifiers so that a link between the individual and his/her data would be difficult to reestablish. While a de-identified data set may contain an encrypted patient identifier with which authorized individuals may re-link a patient to their data set. This data set must not contain data that will enable an unauthorized individual to infer a patient's identity from the existing data element. Anonymisation refers to the permanent elimination of the link between the individual and his/her medical record to the point that re-establishment of the link would be virtually impossible [7]. In addition to anonymization and de-identification, which are the most widely used cases for the protection of patient medical information, there are various data protection methods that we can implemented for this thesis project(Table 4.1).

Table 4.1: Definitions of various data protection methods- source [7]

Data Protection	Description
Anonymization	Irreversible removal of the link between the individual and his or her medical record data to the degree that it would be virtually impossible to reestablish the link.
Augmentation	Augmentation Often achieved by generalization, in which each record is indistinguishable from another shared record.
Binning	Data pre-processing technique used to reduce the effects of minor observational errors; the original data values which fall in a given small interval (i.e., a bin) are replaced by a value representative of that interval.
Cell Suppression	Blanking certain fields in a data table in such a way that no entry (row) in the table is unique.
Censoring	Value of a measurement or observation is only partially known.
De-identification	Removal or replacement of personal identifiers so that it would be difficult to reestablish a link between the individual and his or her data.
Depersonalization	Process of identifying and separating personal from other data.
Disambiguation	Process to provide clarity when a term is ambiguous.
Encryption	Process of transforming data using an algorithm to make it unreadable except to the intended recipient.
Eponyms	Words that could be both clinical terms used in a report or proper names (e.g., Parkinson's disease).
Exclusion	Prohibition of specific data elements.
Generalization	Process of creating successive layers of summary data in a database.
Hash Function	Algorithm that converts a large data set into a small datum, usually a single integer that may serve as an index to an array; typically resulting in an anonymous code, same for a given individual, but impossible to retrieve the identity.

Hiding Function	Methodology that makes information invisible except to the intended recipient.
Obfuscation	Concealment of intended meaning in communication, making communication confusing, intentionally ambiguous, and more difficult to interpret.
Pseudonymization	Identification data is transformed and then replaced by a specifier that cannot be associated with the data without knowing a certain key.
Transformation	Conversion of data from a source data format into destination data.

To make it simpler, we are going to concentrate on the last item in Table 4.1 which is Transformation itself. In section 4.3.2, we will see how `patient_identification` is segregated from `bio_values`. So that re-establishing a link between the individual and his/her observation data would be difficult and it would be our method for protecting patient medical information. Please take into account that anonymization or de-identification would be more reliable methods but we keep it simpler to reduce the complexity of the project and concentrate on implementing the whole procedure. The development of a more complex data protection approach might be a reasonable choice for future work in this thesis project.

The goal is to get useful patient information and his/her test result, loaded into a data warehousing system in an analysis-ready format. We need to study the HL7 message very carefully and understand the significance of each segment to achieve this goal. Details about each segment and sub-segments of HL7 message can be found in Appendix A. Conceptual design of ETL processes is shown in the Figure 4.1 separated by dashes named `ETL_for_HL7`. The reason for using conceptual design to represent these procedures is to correlate the steps of data integration with the business rules to establish which information is needed and how this information can be effectively transformed and loaded into the data warehouse. In section 4.3 of this thesis, it will be explained in more detail.

4.1.3 Medical Analysis (Exploitation layer)

The data is available for analysts in the final phase of the whole process and it is ready for analysis purposes such as exploratory, departmental, and biomedical analysis(Figure 4.1 - exploitation layer). As an example, in Genomecore company, there are 6 people who were working in Analysis department and they need `patient_identification` and `bio_values` which we extracted , transformed and loaded in DW and that was the basis of the idea for my final master thesis.

My approach for storing patient observation from their test result comes from several meetings that I have done with CEO of Genomecore company , Dr. Oscar Flores. According to those meeting, I come up with the idea of using some specific fields in our data model. These are the fields that we will store them as `Bio_values`: Title of the analysis, code, group, min, max and units. You will find complete explanation for each of these fields in section 4.2.2.

The use of our tool for medical analysis could be: 1) For transactional purpose, as we need to store somewhere the data of a single biomedical analysis to include it on the reports and 2) Aggregated studies, for example to explore normality levels, population trends, etc. A common aggregated analysis can be, for example, check which are the normal levels of, for instance, cholesterol (or any other biomarker). To do so we would like to get the 90% confidence interval from all observations segregated by sex <`Male, Female`> and age range <`young, middle_aged, old`>. Other analysis can be to check correlations between some levels and anothers (for example, we can check which is the correlations between cholesterol and triglycerides levels).

Identifying potential health concerns before they become serious is an essential part of healthcare organizations. Despite appropriate information, healthcare providers do not have the indicators or knowledge required to avoid health emergencies, but data analytics may provide patient

health tracking to identify these issues. With this methodology, healthcare providers will control patient statistics and vital features and focus on preventive treatment to keep patients out of the hospital. This can also avoid the creation or continuation of such diseases by delivering the right treatment at the right time and ensuring stronger overall health. The analysis team would be able to make sense of the data once it is in the DW. There is so much knowledge obtained in health care, and not all of them are important for an analysis to derive improvements. An analytics team member would also be able to present data in a manner that is understandable for non-technical users. The visual representation must be easy to comprehend by a general person. The analysis of the data starts after the relevant data has been collected, taken to a single source of truth (SSOT[1]) and grouped together. The analysis process consists of several parts:

- Data quality evaluation: Analysts need to comprehend the data by taking time to analyze the data. They would need to consider their assessment method, which they will refer to when discussing their results with the viewer.

- Data discovery: Analysts should take time to explore the data and search for relevant capabilities and patterns.

- Interpretation: The interpretation of blood test results typically might become exceedingly challenging when multiple parameter values are outside the normal range or when all values are within the normal range.

- Presentation: Presentation is quite important. Now with all the progress that has been done bringing the data to this stage, the analyst needs to tell the story with the data in a consumable, easy way that provides services to the audience.

Visualization tools and analytics approaches can be used to model patient observations and to highlight the correlation between different analytical measures to create an appropriate dashboard.

4.2 Methodology for modeling data and physical data-warehouse design

In this section, I will clarify all decisions I have made about modeling data, creating the target schema, and converting it to a physical data warehouse. As you can see in 4.3, we have three steps to create a data warehousing system. A brief description of each step presented below.

- ***Data modeling:*** Data modeling is the initial step in the data warehouse design process. This step is often seen as a high-level and theoretical design phase by choosing a data-driven or demand-driven approach. It could be that the efficiency of the work is extremely changing. The demand-driven approach will be used in this project, which I will explain it in detail in the section 4.2.1.

- ***Create conceptual schema:*** Creating a conceptual schema is required to make sure that we have the logical structure of the entire database. Designing a schema depends on the business rules that we have and the data modeling approach that we choose. You can find a complete explanation of my approach for designing the conceptual schema as well as the graphical representation of it in section 4.2.2.

- ***Convert schema to physical data warehouse:*** Finally, when the visual representation of the schema is prepared, we are going to convert it to a physical data warehouse. The physical database model shows all table structures, including column name, column data type, column constraint, primary key , foreign key, and table relationship. A full description

[1]In information systems design and theory, single source of truth (SSOT) is the practice of structuring information models and associated data schema such that every data element is mastered (or edited) in only one place. SSOT systems provide data that are authentic, relevant, and referable. Source: [25]

of this section can be found in section 4.2.3.

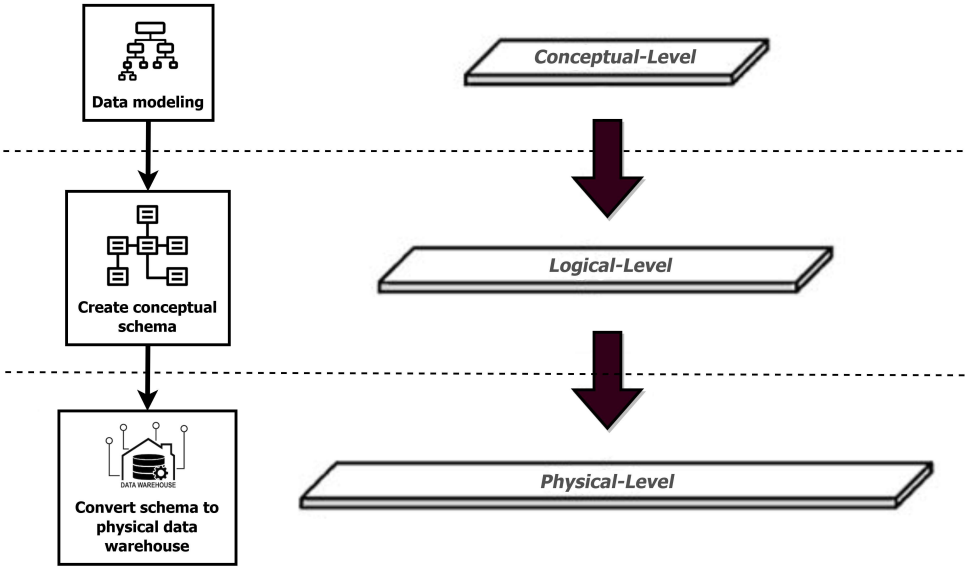

Figure 4.3: Data warehouse design

4.2.1 Data modeling

The data modeling process starts by considering the statement of requirements. When designing a proper data model, we are able to gather requirements. It is important to communicate the requirements with the analysis department. Data modeling is a process by which data is structurally stored in database accordance with a variety of business rules. Keeping the data in the proper format ensures that we can get responses to the business rules quicker and more effectively. It is important to model the data correctly in order to properly and efficiently access the data. However, the entire data modeling process is not as simple as that. We need to provide a clear understanding of the organizational structure and then come up with a solution that meets its final goals and is sufficient to achieve the desired objectives.

Consistent data representation makes it easier to analyze data efficiently. It gives a straightforward picture of the data that analysts can use to better understand the data. This goal can only be achieved if we know the needs of the analysis department correctly. I had some individual discussions with the CEO of Genomecore, who is head of the data analysis department at this company. We also defined the requirements for the Data Warehousing System that we need to manage health data sources. I am going to go through them in this section and clarify our decisions in the initial phase of data modeling.

This process starts with the recognition of business rules by consumers. As a data designer, you need to understand all the data so that you can build a proper data warehousing system. Based on the needs of data analysts, the first model to be established is a conceptual model. While data models provide the data structure, business rules are often used to tell how the data can and should be used [26]. We apply those constraints to ensure that the database follows the business rules of the consumer. Our constraints on business rules fall into two categories: 1- The constraints on data (Table 4.2). 2- The constraints on relationships (Table 4.3). Business rules will become constraints at the database level in section 4.2.3.

Table 4.2: Data constraints

Num	Business-Rules
1	Every patient must have a name (e.g., Meysam), a first surname (e.g., Zamani), date_of_birth (e.g., 15/03/1989), an age (e.g., 31), age-range (e.g., young), sex (e.g., male) and dni number (e.g., Y5694768M).
2	Patient range age have just three type: Young, Middle_aged, and Old.
3	A patient can have a second surname (e.g., Forooshani).
4	Each patient should have unique identifying number.
5	Each sample date/time should have the sample timestamp as unique identifier (e.g., 2019-11-22 09:57:19).
6	Sample timestamp should clearly show the date and time of an observation.
7	Every sample date/time must have a day, a month, and a year defined separately (e.g., year:2019, month:11, day:22).
8	Each analysis type should have the code of analysis type as unique identifier (e.g., HB: Hemoglobin).
9	Every analysis type must have a name (e.g., Hemoglobin), a Unit of measurement (e.g., g/dl), and Min and Max value (e.g., Min value for Hemoglobin analysis is 12.5 and Max value is 17.2).
10	Every observation result should have consequence[2] (e.g., The normalcy range for Cholesterol total(CT) is between 0.0 and 200.0 and the result value is 247 which is higher than normalcy range and the consequence would be "H" refers to "Above normal high" value).
11	Every observation result can have the relative discrepancy from the normal range for each observation result. It should be in the range of 0 to 1 and shows relative discrepancy in case that the observation result be higher than Max value or lower than Min value. (e.g., relative discrepancy for CT-Colesterol Total is 0.23 due to the result value equal 247 and normalcy range of 0.0 to 200.0).
12	Each group analysis should have unique identifying number.
13	Each observation must have a result value (e.g., Hemoglobin analysis result for our data sample is 15.5).

Table 4.3: Relationship constraints

Num	Business-Rules
1	An observation result must be associated with one and only one patient.
2	A patient can be associated with one or many observation result.
3	An observation result must be associated with one and only one sample date/time.
4	A sample date/time can be associated with one or many observation result.
5	An observation result must be associated with one and only one analysis type.
6	An analysis type can be associated with one or many observation result.
7	An analysis type must be associated with one and only one group analysis.

[2]Consequence refers to Abnormal flags that identifies the normalcy status of the result - Source [8].

8	A group analysis can be associated with one or many analysis type.
9	A patient cannot have more than one observation result for an analysis type at the same time.

After we have defined all business rules we will decide on the data warehousing system structure. The building of a data warehouse (DW) is a challenging issue. Current methods for developing DW can fall within three main approaches: data-driven, demand-driven and hybrid approach.

- ***Data-driven approach:*** Also known as supply-driven, the requirements are the last thing to be considered in the data-driven approach. Organizational goals and user requirements are not reflected at all. Moreover, this approach risks wasting resources by handling many unneeded information structures [27].

- ***Demand-driven approach:*** Also known as requirement-driven or goal-driven, this approach is proposed to establish a first model based on the needs of the company. The Analysis Department defines the objectives and gathers, prioritizes, and defines the business rules that support these goals. Subsequently, business rules are prioritized and the most relevant business rules are specified in order to identify data elements terms [28].

- ***Hybrid approach:*** There are a few works implement combining the two previous approaches. In general, these approaches begin with a demand-driven stage to identify facts of interest and then identify their dimensional concepts through a data-driven stage [29].

After talking to Dr. Oscar Flores Baquero, Head of the Analysis Department and CEO at Genomecore, we came up with the idea that the demand-driven would be the right approach for blood test descriptive analysis purposes derived from the HL7 response. That is because basically, HL7 messages have lots of segments that they have never used and had no sense to store in DW, on the other hand, the initial purpose why we are going to provide this ETL tool is to make it easier for analysts to access the relevant information.

Data modeling can be done in various ways. In this project, we have decided to use Relational Data Modeling, which is a Data Warehousing solution and reduces complexity and provides a clear overview of the data compares to the hierarchical model (XML in this project). It is used to store large data for supporting the company to perform data analysis. A data warehouse needs to be modeled before it is created in the Database Management System (DBMS). Data warehouse modeling can be performed in different ways, such as normalization, dimensional, and the most recent one commonly used is the combination of normalized and dimensional that we are going to use it in this project[13]. Some examples of notations for data warehouse conceptual modeling are the Entity-Relationship Model and the Unified Modeling Language (UML) that we are going to use UML notations to represent dimensional modeling in the next section (Section 4.2.2). There are three forms of dimensional modeling, which are the snowflake schema, fact constellations, and star schema [30]. By considering the questions below, we choose what scheme to use in Dimensional model development:[31]

- What kind of analysis do analysts attempt to perform on the data?

✓ Aggregated studies, for example to explore normality levels, population trends, etc. A common aggregated analysis can be check which are the normal levels of, for instance, cholesterol (or any other biomarkers). To do so they would like to get the 90% confidence interval from all observations segregated by sex <Male, Female> and age range <young, middle_aged, old>. Other analysis can be to check correlations between some levels and anothers (for example, we can check which is the correlations between cholesterol and triglycerides levels).

- Which are the other purposes for storing data into the data warehouse?

- ✓ For transactional purpose, as they need to store somewhere the data of a single biomedical analysis to include it on the reports.

- Which are the requirements and restrictions for the analytics?

- ✓ In addition to all business rules specified in table 4.2 and 4.3, they requested to facilitate group analysis filter over the whole analysis type.

- What tool do they intend to use for BI? While different tools may appear to demonstrate the same sort of data and results, they may be very different under the surface, and for the best results depend on a specific scheme.

- ✓ Pentaho Business Analytics (BA)[3] and also ScaiPlatform[4] from scaidata[5].

We chose snowflake schema by answering the above questions after several meetings with the head of data analysts at Genomecore company. The schema will have one main fact table with three dimension tables connected to it. It goes a step further to the star scheme, with the fact table surrounded by one denormalized dimension linked to another dimension table, and two other dimensions that have normalized hierarchies. Each level in the dimension hierarchy would be its dimensional table with parent keys that are provided to link the hierarchical structure. The fact-table collects the foreign keys to the lowest dimensions hierarchy level. Some BI tools are explicitly designed to use snowflake schemas. To generate reports and queries, the tools use metadata definitions about the dimensional snowflake model [31].

By choosing the Demand-driven approach, Relational Data modeling, the Unified Modeling Language (UML) graphical notation, and snowflake schema, we will create a graphical representation of the conceptual schema in the next section. (Section 4.2.2).

4.2.2 Create conceptual schema

Conceptual schema design is the process of creating a conceptual representation of the database content in logical terms, which is clear and straightforward in order to create a physical data warehouse in the next step. The method takes the input data requirements for the tool that uses the database and creates a scheme represented in the conceptual modeling notation [32].

The snowflake schema will be used in this project, as discussed in section 4.2.1. A snowflake schema is a normalization form of a star scheme in which dimensional tables can be connected to other dimensional tables in addition to the fact table. It expressed in the UML that you can find it in Figure 4.4 by applying all business rules defined in section 4.2.1.

Keep checking our data model before proceeding to the next phase. For example, if we have to choose the primary key to properly identify each record in the dataset, we have to make sure we pick the right attribute. We will go through them in each table and identifying them.

As you can see in Figure 4.4, in our relational data model, we have 3 main dimension tables which are `Dim_Patient`, `Dim_Type_Analysis` and `Dim_Date`. On the other hand, we have an aggregated dimension table, named `Dim_Group` that allows quicker access to commonly used data like group of analysis time. By doing that, the `Dim_Group` table is rolled up along `Dim_Type_Analysis` table. And finally the last table is the main fact table in the schema named `Fact_Observation` with a connection to `Dim_Patient`, `Dim_Type_Analysis` and `Dim_Date` table.

[3] Pentaho Business Analytics (BA) The Pentaho Business Analytics (BA) platform enables you to securely access, integrate, manipulate, visualize, and analyze your big data assets.

[4] ScaiPlatform combines the power of business intelligence and data management. It seamlessly integrates with your SQL databases and your data warehouses with real-time data access.

[5] Scaidata is the real-time analytics platform for business intelligence, data management and automation.

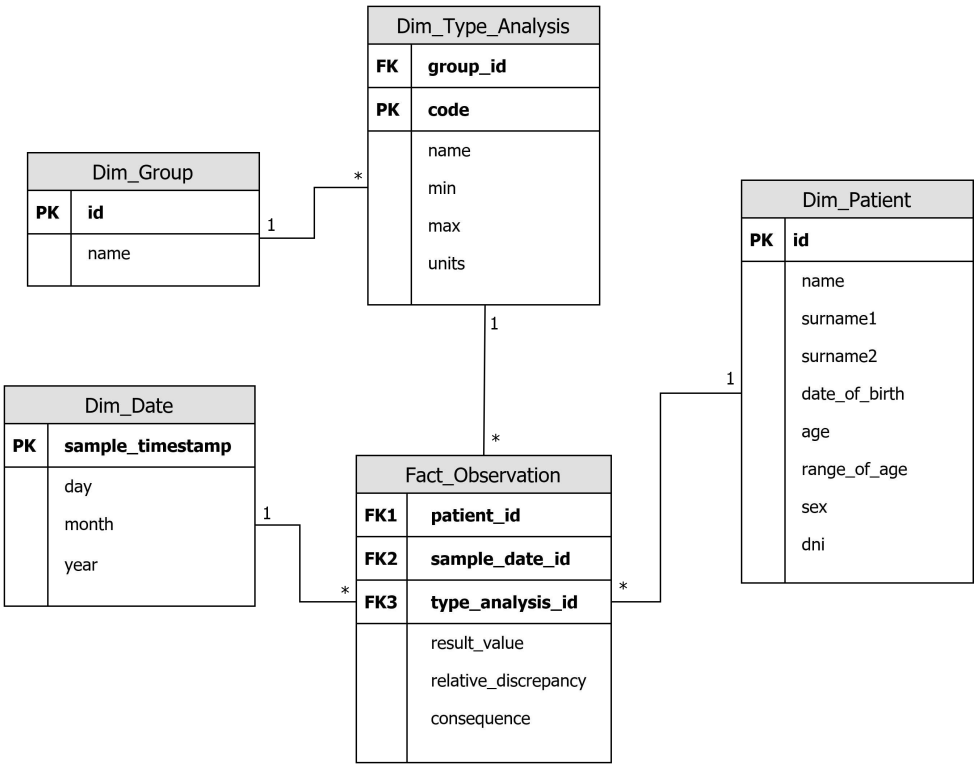

Figure 4.4: Conceptual design of the target DW schema

In this section, we are going to show you all attributes I have chosen to put in each table and also represent the relationships between tables. In order to organize the content of this section, I have specified the name of each part as the table's title.

4.2.2.1 Dim_Patient:

The `Dim_Patient` table consists of patient data such as **name** of patient, **first surname** and **second surname** of patient, **date of birth**, **age**, **rane of age**, **gender(sex)** and **DNI**. There is also a primary key, as you can see in Figure 4.5, which perform connection between `Dim_Patient` table and the fact table (`Fact_Observation`).

First I defined the patient's `DNI` as the primary key, but after discussing it with my supervisor, I notice that patient's `DNI` is a kind of sensitive data that would be better not highlighted as the primary key. And that is why we decided to define the primary key of `Dim_Patient` as a `SERIAL` number to ensure it's unique to each patient.

As we defined in business rules and you can see in Figure 4.5 The relation between `Dim_Patient` and `Fact_Observation` is one to many. This means that a patient can be associated with one or many observation results, while an observation result must be associated with one and only one patient.

4.2.2.2 Dim_Date:

The `Dim_Date` table consist of data regarding sample date/time, such as **day**, **month** and the **year** of a sample test. There is also a primary key, as you can see in Figure 4.6, which perform connection between `Dim_Date` table and the fact table (`Fact_Observation`). I set the complete sample timestamp as the primary key of the `Dim_Date` table to make it unique for the `Fact_Observation` reference.

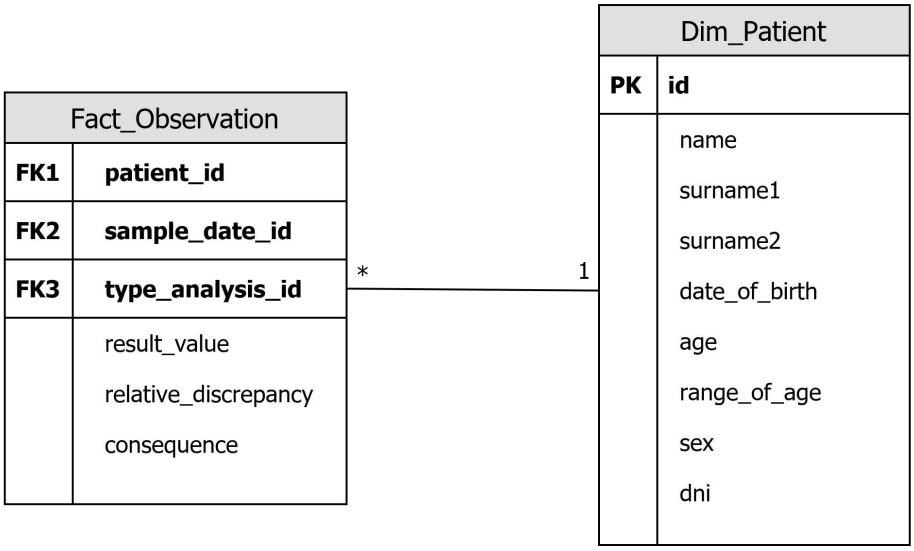

Figure 4.5: `Dim_Patient` table

As we defined in business rules and you can see in Figure 4.6 The relation between `Dim_Date` and `Fact_Observation` is one to many. This means that a **sample_timestamp** can be associated with one or many observation results, while An observation result must be associated with one and only one **sample_timestamp**.

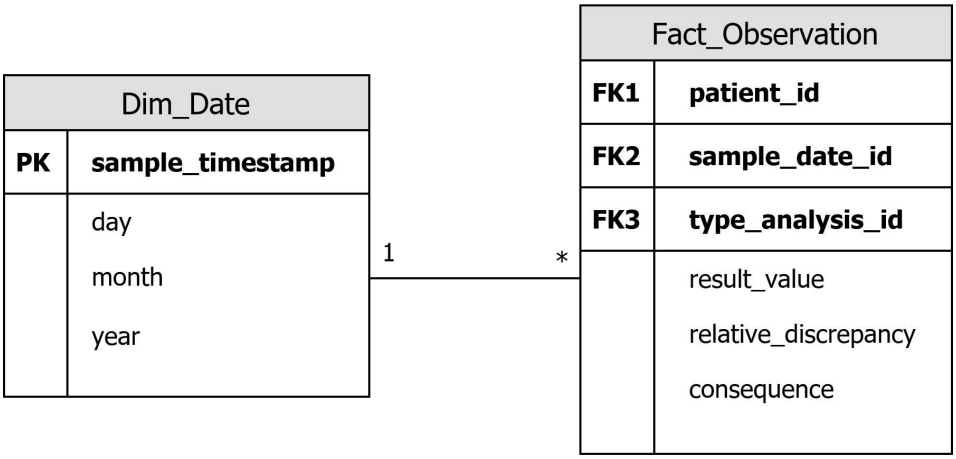

Figure 4.6: `Dim_Date` table

4.2.2.3 Dim_Type_Analysis:

The `Dim_Type_Analysis` table consists of data regarding analysis type which has attributes such as **name** of analysis, **min** and the **max** which are shown normality range of analysis and the **units** that has to define in order to know the measure of result value.

There is also a primary key, as you can see in Figure 4.7, which perform connection between the `Dim_Type_Analysis` and `Fact_Observation`. I defined the **code** of analysis as primary-key in the `Dim_Type_Analysis` table to make it unique for `Fact_Observation` reference.

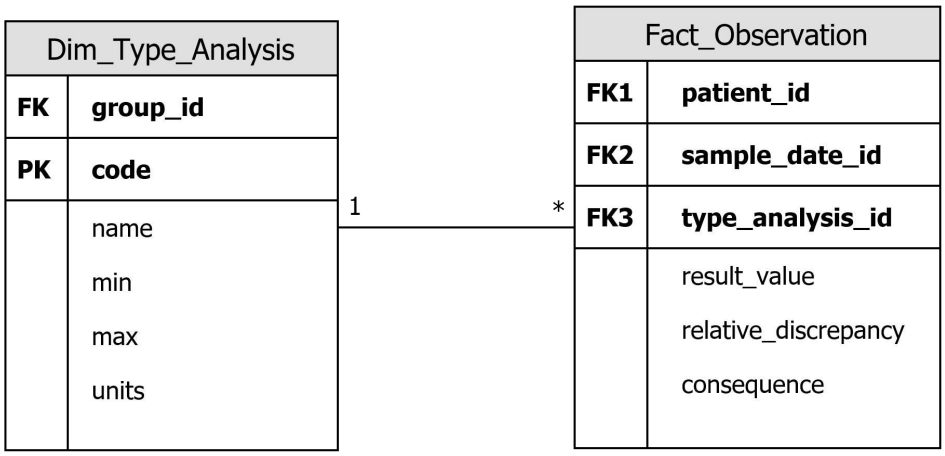

Figure 4.7: `Dim_Type_Analysis` table

As we defined in business rules and you can see in Figure 4.7 The relation between `Dim_Type_Analysis` and `Fact_Observation` is one to many. This means that an analysis type can be associated with one or many observation results, while An observation result must be associated with one and only one analysis type. There is also a foreign-key in the `Dim_Type_Analysis` table which performs the connection between `Dim_Group` and `Dim_Type_Analysis`.

4.2.2.4 Dim_Group:

The `Dim_Group` table consists of data regarding the group of analysis, which has **name** of group analysis as an attribute. There is also a primary key, as you can see in Figure 4.8, which perform connection between `Dim_Group` table and and `Dim_Type_Analysis` table. As It mentioned in the previous section, there is a foreign-key in the `Dim_Type_Analysis` table that has reference to this primary key in the `Dim_Group` table.

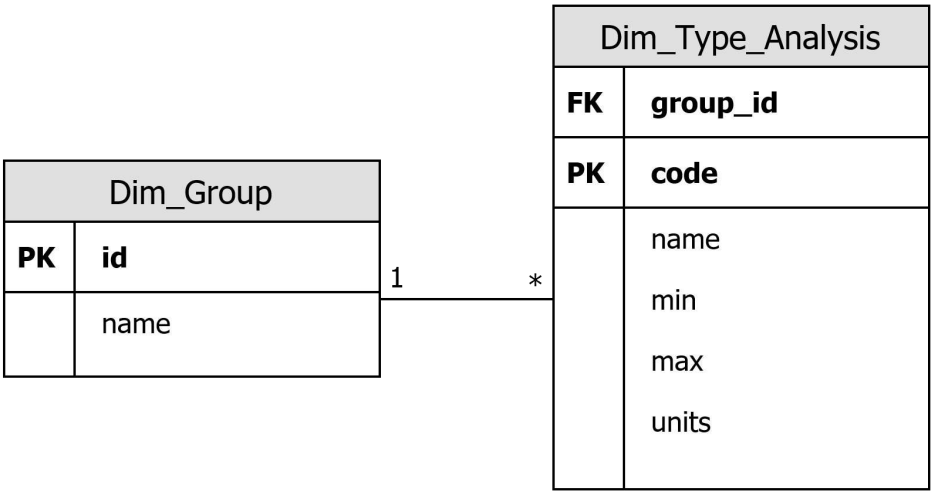

Figure 4.8: `Dim_Group` table

As we defined in business rules and you can see in Figure 4.8 The relation between `Dim_Group` and `Dim_Type_Analysis` table is one to many. It means that a group analysis can be associated

with one or many analysis type, while an analysis type must be associated with one and only one group analysis.

The table Dim_Group is the use of aggregate tables to allow quicker access to commonly used data while keeping the power to address any user query. basically we are interested in analyzing data per each type or per group analysis and for doing that, we are going to design an aggregated table for the group analysis. By doing that, the Dim_Group table is rolled up along the table Dim_Type_Analysis.

4.2.2.5 Fact_Observation:

The Fact_Observation table is the main fact table in our design. It is connected to three dimensional tables, which are: Dim_Patient, Dim_Date and Dim_Type_Analysis. As you can see in Figure 4.9, this table contains three foreign keys which are **patient_id**, **sample_date_id** and **type_analysis_id** that perform connection between fact and dimension tables like as bellow:

- The relation between Fact_Observation and Dim_Patient is performed by **patient_id**, which is foreign-key in Fact_Observation table and its the reference of **id** in Dim_Patient as a primary-key.
- The relation between Fact_Observation and Dim_Date is performed by **sample_date-_id**, which is foreign-key in Fact_Observation table and its the reference of **id** in Dim_Patient as a primary-key.
- The relation between Fact_Observation and Dim_Type_Analysis is performed by **type-_a nalysis_id**, which is foreign-key in Fact_Observationtable and its the reference of **code** in Dim_Type_Analysis as a primary-key.

On the other hand, there are three measures named **result_value**, **relative_discrepancy**, and **consequence** That we are going through them one by one to define them and see how they potentially aggregate if we want to use ROLLUP operation:

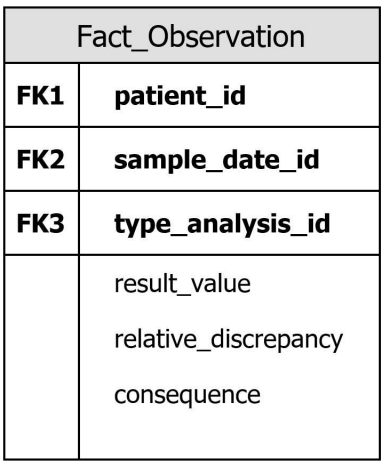

Figure 4.9: Fact_Observation table

- The first one is **result_value** that has result value of each observation related to a specific type of analysis in Dim_Type_Analysis. That means, we have to find the number of records created in Fact_Observation, equal to the total number of type analysis record for each insertion of HL7 response.

Now we are going to see what kind of aggregation function and with which attributes can be performed to see the potential exploitation that consumers (data analysts) might want to take advantage of them for the **result_value**.

An example of aggregation function would be the `AVERAGE` of the **result_value** by rolling up the patient age, patient range_of_age, his/her gender, and also for the specific day, month or year related to a specific type of analysis.

We can not use the **result_value** for rolling up the group analysis because each of the analysis types may have different units of measurement and we can not group them while they contain values from different units. Even it would not possible to group those analysis type that has the same units of measurement because their normality range is different and grouping them for the **result_value** will not provide us any useful information. The way that we can use aggregation functions for the group analysis will be normalizing the **result_value** in a way to have them unitless. That is the point, we decided to derive a new measure that we will see it in the next point.

- The second one is **relative_discrepancy** which is an example of deriving new measures and it created to show a relative discrepancy in case that the observation result is higher than `Max` value or lower than `Min` value. To do so, first, we have to calculate the absolute value of the differences between the `Min`/`Max` value and the result value of an observation. It would be expressed with the following equation:

$$AbsoluteDiscrepancy = \begin{cases} |MaxValue - ResultValue|, \\ \text{if the result value be higher than Max value} \\ \\ |MinValue - ResultValue|, \\ \text{if the result value be lower than Min value} \end{cases}$$

The absolute discrepancy is expressed in the same unit as the result value and `Min`/`Max` values. We are going to make it unitless in order to can take advantage of aggregation function for the group of analysis. In the second step, we will calculate the relative discrepancy which is expressed with the following equation:

$$RelativeDiscrepancy = \begin{cases} \frac{AbsoluteDiscrepancy}{MaxValue}, \\ \text{if the result value be higher than Max value} \\ \\ \frac{AbsoluteDiscrepancy}{MinValue}, \\ \text{if the result value be lower than Min value} \end{cases}$$

For the calculation of `relative_discrepancy`, the denominator will be `Max` value, if the result value is higher than the normality range, or will be `Min` value, if the result value is lower than normality range.

Now we are going to see what kind of aggregation function and with which attributes can be performed to see the potential exploitation that consumers (data analysts) might want to take advantage of them by the `relative_discrepancy`.

An example of aggregation function would be the `AVERAGE` of the `relative_discrepancy` by rolling up the group analysis. The `relative_discrepancy` is signless and its because we will recognize the lower or the higher by consequence variable that identifies abnormal flag and it extracted to give us information regarding the normality status of the result that we will go through it in the next point.

- The last one is **consequence** that refers to the abnormal flags that identify the normality status of the result. The complete list of consequences is presented in table 4.4. For example, when the laboratory is able to determine a textual report's normal status, these should be identified as N when normal, and A when abnormal.

Table 4.4: Concept code & concept name of abnormal flags

Code	Value
L	Below normal low
LL	Alert low
<	Panic low
H	Above normal high
HH	Alert high
>	Panic high
A	Abnormal
N	Normal
AA	Very abnormal

As you can see, since the **consequence** is not a numeric variable, initially, we can not perform any aggregation function on it. However, these abnormal flags are ordered from very low to very high and in this sense, we can convert the **consequence** to numerical variables by assigning value numbers from 1 to 9 to each of abnormal flags. by doing that, we can take advantage of aggregation functions for this measure alike.

4.2.3 Convert conceptual schema to physical data warehouse

The next step in the DW design process would be to convert the snowflake schema data model to a physical data warehouse. In principle, there are five steps to build the physical model from the snowflake schema model, namely:

1. Finding fact table.
2. Finding dimension tables.
3. Finding dimension tables that are in different levels of granularity.
4. Changing dimension tables to relational tables.
5. Changing fact table to relational table.

As I explained in section 4.2.2, we have 3 main dimension tables named `Dim_Patient`, `Dim_Date` and `Dim_Type_Analysis`. The only fact table is `Fact_Observation`. additionally, we have the `Dim_Group` table which is in the second hierarchy level connected to the `Dim_Type_Analysis` table. Now, in order to convert this Snowflake schema to a physical data warehouse, we need to translate UML diagram (Figure 4.4) with key constraints to relational tables.

4.2.3.1 Convert dimension tables to physical data warehouse

Identifying the dimension tables begins by looking at all the tables at the logical level. The constraint will be checked for each attribute. Also find the secondary dimension table called `Dim_Group` (a dimension that is directly linked to the first level dimension table called `Dim_Type_Analysis`). We will go through them and translate each into SQL query language.

- ***Dim_Patient:*** The first table I would convert in the physical data warehouse is `Dim_Patient`. It has patient identification information that we will go through them:

  id: The id is the `PRIMARY_KEY` for the `Dim_Patient` table. By assigning the `SERIAL` type to the id column, first, create a sequence object and set the next value generated by the sequence as the default value for the column. Second, add a `NOT NULL` constraint to the id column because a sequence always generates an integer, which is a non-null value. And Third, assign the owner of the sequence to the id column; as a result, the sequence object is deleted when the id column or table is dropped. It is important to note that the `SERIAL` does not implicitly create an index on the column or make the column as the primary key column. However, this can be done easily by specifying the `PRIMARY KEY` constraint for the `SERIAL` column.

  name: The name column will be a character string column with a `VARCHAR` type which does `NOT` accept the `NULL` value because according to the rule number 1, in the table 4.2, every patient must have a name. We are assuming that the `name` of patient will not be more than 100 character.

  surname1: The surname1 column will be character string column with the `VARCHAR` type which does `NOT` accept the `NULL` value because according to the rule number 1, in the table 4.2, every patient must have a first surename. We are assuming that the first surename of patient will not be more than 100 character.

  surname2: The surname2 column will be character string column with the `VARCHAR` type which accepts the `NULL` value because according to the rule number 3, in the table 4.2, a patient can have a second surename and it is not mandatory. We are assuming that the second surename of patient will not be more than 100 character.

  date_of_birth: The `date_of_birth` column will have date type which has only the date component. The default literal string format of a `DATE` value is `YYYY-DD-MM` which `YYYY` is four digits that represent the year with the range of 0001 to 9999, `MM` is two digits that represent the monthwith the range of 01 to 12, and DD is two digits that represent the day of the specified month with the range of 01 to 31. This column does `NOT` accept the `NULL` value because according to the rule number 1, in the table 4.2, every patient must have a `date_of_birth`.

  age: The age column will have the `SMALLINT` type. `SMALLINT` is sufficient and it is not necessary to use the Int type instead. This column does `NOT` accept a `NULL` value because according to the rule number 1, in the table 4.2, every patient must have an `age`.

  range_age: The `range_age` column will be the character string column with the `VARCHAR` type which does `NOT` accept the `NULL` value because according to the rule number 1, in the table 4.2, every patient must have an `range_age`. As the values of this field are predefined and the max length of its value, reference to the longest predefined `range_age`, can be 11 character, we will limit it by 15 characters per value.

  sex: The sex column will be a character string column with the `VARCHAR` type which does `NOT` accept the `NULL` value because according to the rule number 1, in the table 4.2, every patient must have a `gender`. As the values of this field are predefined and the max length of its value, have to be just 1 character, since the acceptable values for this field are: A for Ambiguous, F for Female, M for Male, N for Not applicable, O for Other, U for Unknown. We will limit it by 10 characters.

  dni: The dni column will be character string column with `VARCHAR` type which does `NOT` accept the `NULL` value because according to the rule number 1, in the table 4.2, every patient must have a `dni`. We will limit it by 15 characters per value because the `DNI`s typically will not occupied more than 15 characters.

After defining all data types and constraints for each attribute, the SQL query for creating the Dim_Patient table can be performed as follows:

```
CREATE TABLE Dim_Patient
(
    id SERIAL NOT NULL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    surname1 VARCHAR(100) NOT NULL,
    surname2 VARCHAR(100),
    date_of_birth DATE NOT NULL,
    age SMALLINT NOT NULL,
    range_age VARCHAR(15) NOT NULL,
    sex VARCHAR(10) NOT NULL,
    dni VARCHAR(15) NOT NULL
)
```

- ***Dim_Date:*** The Dim_Date table has the sample date/time information. By keeping records on the sample date information, we will access a patient's specific blood test result that we will go through them:

  sample_timestamp: The sample_timestamp is the PRIMARY_KEY for the Dim_Date table. The sample_timestamp column will have the timestamp type. The timestamp data type allows you to store both date and time. This column does NOT accept NULL value since it would be PRIMARY_KEY of the table and according to the rule number 3, in the table 4.3, an observation result must be associated with one and only one sample date/time which it performs through the PRIMARY_KEY.

  day: The day column will have the SMALLINT type. SMALLINT is sufficient and it is not necessary to use the Int type instead. This column does NOT accept the NULL value because according to the rule number 7, in the table 4.2, Every sample date/time must have a day, defined separately.

  month: The month column will have the SMALLINT type. SMALLINT is sufficient and it is not necessary to use the Int type instead. This column does NOT accept the NULL value because according to the rule number 7, in the table 4.2, Every sample date/time must have a month, defined separately.

  year: The year column will have the SMALLINT type. SMALLINT is sufficient and it is not necessary to use the Int type instead. This column does NOT accept the NULL value because according to the rule number 7, in the table 4.2, Every sample date/time must have a year, defined separately.

  After defining all data types and constraints for each of attributes, we can perform a SQL query for creating table Dim_Date as bellow:

```
CREATE TABLE Dim_Date
(
    sample_timestamp timestamp NOT NULL PRIMARY KEY,
    day SMALLINT NOT NULL,
    month SMALLINT NOT NULL,
    year SMALLINT NOT NULL
)
```

- ***Dim_Group:*** The Dim_Group table has the list of group analysis inside. This table is defined as the second hierarchy level after Dim_Type_Analysis table. This normalization has some benefits for consumers(Data Analysts) as we discussed in section 4.2.1.

Dim_Group just have one attribute and a primary key that we will go through them:

id: The id is The PRIMARY_KEY for the Dim_Group table. As I explained before, by choosing the SERIAL data type, first, create a sequence object and set the next value generated by the sequence as the default value for the column. Second, add a NOT NULL constraint to the id column because a sequence always generates an integer, which is a non-null value. And Third, assign the owner of the sequence to the id column. As a result, the sequence object is deleted when the id column or table is dropped.

name: The name column will be a character string column with the VARCHAR type which does NOT accept the NULL value. As the values of this field is just a name, first we assigned character limitation of 100, but after checking the available sample data, we recognize that this value can be longer than the one assigned before, and because of that we will assume that the max length of its value can be up to 300 character.

After defining data types and constraints for those attributes, we can perform a SQL query for creating table Dim_Group as bellow:

```
CREATE TABLE Dim_Group
(
    id SERIAL NOT NULL PRIMARY KEY,
    name VARCHAR(300) NOT NULL
)
```

- ***Dim_Type_Analysis:*** The Dim_Type_Analysis table has the list of analysis type which contains information related to each type of analysis. There are some attributes in this table that we will go through them:

  code: The code is the PRIMARY_KEY for the Dim_Type_Analysis table. This code is the abbreviation of analysis name and will be character string column with VARCHAR type which does NOT accept NULL value because according to the rule number 8, in the table 4.2, each analysis type should have the code of analysis type as unique identifier. As the values of this field is just a code name, consist of abbreviation of each specific analysis type, we will assume that this column can store up to 10 characters.

  group_id: The group_id is a Foreign-Key in the Dim_Type_Analysis table which performs the connection between Dim_Group and Dim_Type_Analysis and its reference is id in Dim_Group table. group_id column will have the SERIAL type like the id in the Dim_Group table. This column does NOT accept the NULL value because according to the rule number 7, in the table 4.3, An analysis type must be associated with one and only one group analysis.

  name: The name column will be a character string column with the VARCHAR type which does NOT accept the NULL value because according to the rule number 9, in the table 4.2, every analysis type must have a name. We are assuming that the name of each analysis will not be more than 200 character.

  min: The min column will be FLOAT type column which does NOT accept the NULL value because according to the rule number 9, in the table 4.2, every analysis type must have a min value.

  max: The max column will be FLOAT type column which does NOT accept the NULL value because according to the rule number 9, in the table 4.2, every analysis type must have a max value.

  units: The units column will be character string column with the VARCHAR type which does NOT accept the NULL value because according to the rule number 9, in the table 4.2, every analysis type must have a units value. As the characters for defining each units

identifier are limit, we will assume that his column can store up to 20 characters.

After defining all data types and constraints for each of attributes, we can perform a SQL query for creating table Dim_Type_Analysis as bellow:

```
CREATE TABLE Dim_Type_Analysis
(
    code VARCHAR(10) NOT NULL PRIMARY KEY,
    group_id SERIAL NOT NULL REFERENCES Dim_Group (id),
    name VARCHAR(200) NOT NULL,
    min FLOAT NOT NULL,
    max FLOAT NOT NULL,
    units VARCHAR(20) NOT NULL
```

4.2.3.2 Convert fact table to physical data warehouse

The Fact_Observation table is the main fact table in our design which is connected to three-dimension tables which are: Dim_Patient, Dim_Date, and Dim_Type_Analysis. The primary key of the fact table is the collection of foreign keys from dimension tables associated with the fact table. There are three foreign keys and three measures that we will go through them:

patient_id: The patient_id is the foreign key in the Fact_Observation table which performs the connection between Dim_Patient and Fact_Observation and its reference is id in Dim_Patient table. patient_id column will have the SERIAL type like the id in the Dim_Patient table. This column does NOT accept the NULL value because according to the rule number 2, in the table 4.3, a patient can be associated with one or many observation result.

sample_date_id: The sample_date_id is the foreign key in the Fact_Observation table which performs the connection between Dim_Date and Fact_Observation and its reference is sample_timestamp in Dim_Date table. sample_date_id column will have timestamp type like sample_timestamp in the Dim_Date table. This column does NOT accept the NULL value because according to the rule number 4, in the table 4.3, a sample date/time can be associated with one or many observation result.

type_analysis_id: The type_analysis_id is the foreign key in the Fact_Observation table which performs the connection between Dim_Type_Analysis and Fact_Observation and its reference is code in Dim_Type_Analysis table. type_analysis_id column will be character string column with VARCHAR type which does NOT accept the NULL value because according to the rule number 4, in the table 4.3, an analysis type can be associated with one or many observation result. Since we use the code of each analysis type as unique identifier, the number of characters that it can store will be the same as the one that we defined in Dim_Type_Analysis table which is up to 10 characters.

result_value: The result_value column will be a FLOAT type column which does NOT accept the NULL value because according to the rule number 13, in the table 4.2, each observation must have a result value.

relative_discrepancy: The relative_discrepancy column will be a FLOAT type column which accepts the NULL value because according to the rule number 11, in the table 4.2, Every observation result can have the relative discrepancy.

consequence: The consequence column will be a character string column with the VARCHAR type which does NOT accept the NULL value because according to the rule number 11, in the table 4.2, every observation result should have a consequence. As the value of this field is just an abbreviation of each specific abnormal flag, we will assume that it can store up to 5 characters.

After defining all data types and constraints for attributes, the SQL query for creating `Fact_Observation` table can be performed as bellow:

```
CREATE TABLE Fact_Observation
(
    patient_id SERIAL NOT NULL REFERENCES Dim_Patient (id),
    sample_date_id timestamp NOT NULL REFERENCES Dim_Date (
   sample_timestamp),
    type_analysis_id VARCHAR(10) NOT NULL REFERENCES
   Dim_Type_Analysis (code),
    result_value FLOAT NOT NULL,
    relative_discrepancy FLOAT,
    consequence VARCHAR(5)
)
```

4.3 ETL process methodology

In this section, I will explain the ETL process methodology that I used to build `ETL_for_HL7` and address all the decisions I have made in each stage. As you can see, I modeled this process in a diagram that you can find in Figure 4.10. I designed it in three main layers which are Extract, Transform, and Load.

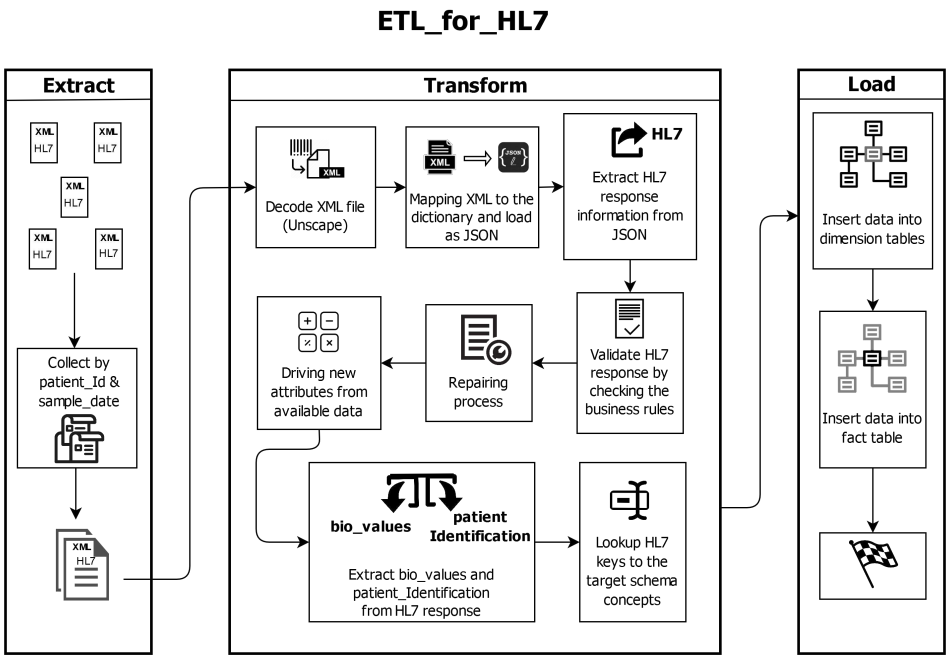

Figure 4.10: Conceptual design of `ETL_for_HL7`

Extraction step: This is the phase in which the ETL process receives the source data. Depending on how the data can be obtained, we may require a different extraction method. For instance, if the data is collected through an API, the extraction method can be different from the way we have the data available on the application. I will describe the approach that I used for extraction, in Section 4.3.1, and I will also address other ways that data can be extracted.

Transformation step: The transformation stage of the ETL process is the most critical aspect of the tool. It consists of a variety of functional processes for transforming data into

meaningful information and getting prepared for the loading phase. It is a process that can be different from one system to another, and there is no particular way to get a final result in this phase. The way I have transformed the HL7 messages can also be used for other HL7 files in any other framework but with some slight changes that I will mention in section 4.3.2. The transformation stage in `ETL_for_HL7` is composed of eight separate subsections listed bellow:

- Decode XML file (Unscape)
- Mapping XML to the dictionary and load as JSON
- Extract HL7 response from JSON
- Validate HL7 response by checking the business rules
- Repairing process
- Deriving new attributes from available data
- Extract `bio_values` and `patient_Identification` from HL7 response
- Lookup HL7 keys to the target schema concepts

Each of these subsections needs to be explained in detail in order to understand the process that I used for dealing with HL7 messages. I'll clearly describe it in section 4.3.2.

Load step: And finally , the last step is to load the transformed data into the data warehouse that we have created in section 4.2. First, we will insert the data into the dimension tables, and then in the next step, we will insert records into the fact table. You can find a comprehensive explanation of the loading step in section 4.3.3.

4.3.1 Extraction

Extraction in the ETL process can be done with internal or external sources. If the source data comes from an external system, then the extraction can be done through an API, for instance in Genomcore company, they used an API to extract data from a laboratory. It usually happened by sending a request like a query to ask for specific patient's test results that have identified with patient ID and sample date. The request sends these two parameters and then the laboratory response through the API with a single XML file that has an HL7 message inside. You can see the representation of this step in Figure 4.11.

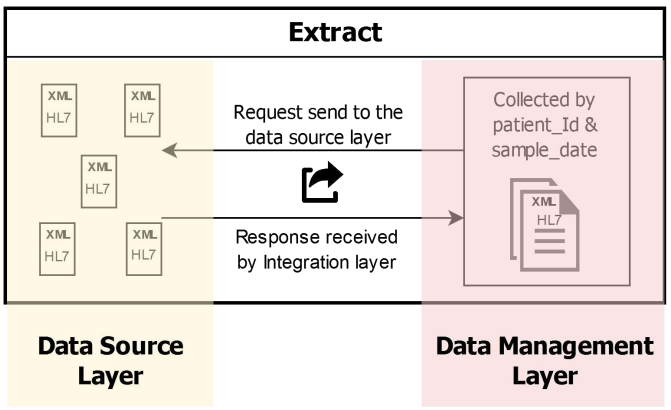

Figure 4.11: Extraction: Collecting data by patient ID and sample date

On the other hand, if the source data is available in the system (Internal Sources), then we will only need a true file naming system for XML files and extract it by specifying the patient ID and sample date required to get the relevant file and start the procedure. That is exactly what we have done for this thesis project by extracting the relevant XML response, from the internal source. The implementation of this step can be found in the section 5.4.2

4.3.2 Transformation

The data extracted from the laboratory is raw and not ready for analysis. Therefore it needs to be cleansed, mapped, and transformed to be prepared for analysis. In fact, this is the key step where the ETL process adds value and changes data such that insightful BI reports can be generated and it is an important step in the analytic workflow. In the transformation step, you can perform customized operations on data. Therefore, we are going to apply a set of functions on HL7 messages represented in the XML file.

Transformation in `ETL_for_HL7` consists of 8 different subsections that we are going through them and we will see the method that I used for transforming the HL7 file. Also, you can find them all in Figure 4.10, in the "Transform" stage of the workflow.

4.3.2.1 Decode XML file (Unscape)

The source data is a decoded XML file which has XML format, but it needs to be encoded in order to fix unescape characters to XML punctuation characters (Figure 4.12).

```
&lt;ORU_R01.PATIENT_RESULT&gt;
    &lt;ORU_R01.PATIENT&gt;
      &lt;PID&gt;
        &lt;PID.2&gt;
          &lt;CX.1&gt;&lt;/CX.1&gt;
        &lt;/PID.2&gt;
        &lt;PID.5&gt;
          &lt;XPN.1&gt;
            &lt;FN.1&gt;Zamani&lt;/FN.1&gt;
          &lt;/XPN.1&gt;
          &lt;XPN.2&gt;Forooshani&lt;/XPN.2&gt;
          &lt;XPN.3&gt;Meysam&lt;/XPN.3&gt;
        &lt;/PID.5&gt;
        &lt;PID.7&gt;
          &lt;TS.1&gt;19890315&lt;/TS.1&gt;
        &lt;/PID.7&gt;
        &lt;PID.8&gt;M&lt;/PID.8&gt;
      &lt;/PID&gt;
    &lt;/ORU_R01.PATIENT&gt;
&lt;/ORU_R01.PATIENT_RESULT&gt;
```

Row format

```
<ORU_R01.PATIENT_RESULT>
    <ORU_R01.PATIENT>
      <PID>
        <PID.2>
          <CX.1></CX.1>
        </PID.2>
        <PID.5>
          <XPN.1>
            <FN.1>Zamani</FN.1>
          </XPN.1>
          <XPN.2>Forooshani</XPN.2>
          <XPN.3>Meysam</XPN.3>
        </PID.5>
        <PID.7>
          <TS.1>19890315</TS.1>
        </PID.7>
        <PID.8>M</PID.8>
      </PID>
    <ORU_R01.PATIENT>
</ORU_R01.PATIENT_RESULT>
```

Unescape format

Figure 4.12: Instance of unescaping row XML format in PID segment

Unescapes an XML file means removing traces of offending characters that might be misinterpreted as markups. The characters in the 4.5 table are reserved in XML and have to be replaced by their respective XML entities. This step is mandatory to have the correct XML format and helps us to continue the transformation in the next step.

Table 4.5: XML Escape/Unescape characters

Char	Escape String
<	<
>	>
"	"
'	'
&	&

4.3.2.2 Mapping XML to the dictionary and load as JSON

The XML file is converted to a dictionary format, that at the highest level, would be JSON array mapping to a list. Dealing with the dictionary and JSON format, in general, makes it much easier to parse the data and extract useful information (Figure 4.13).

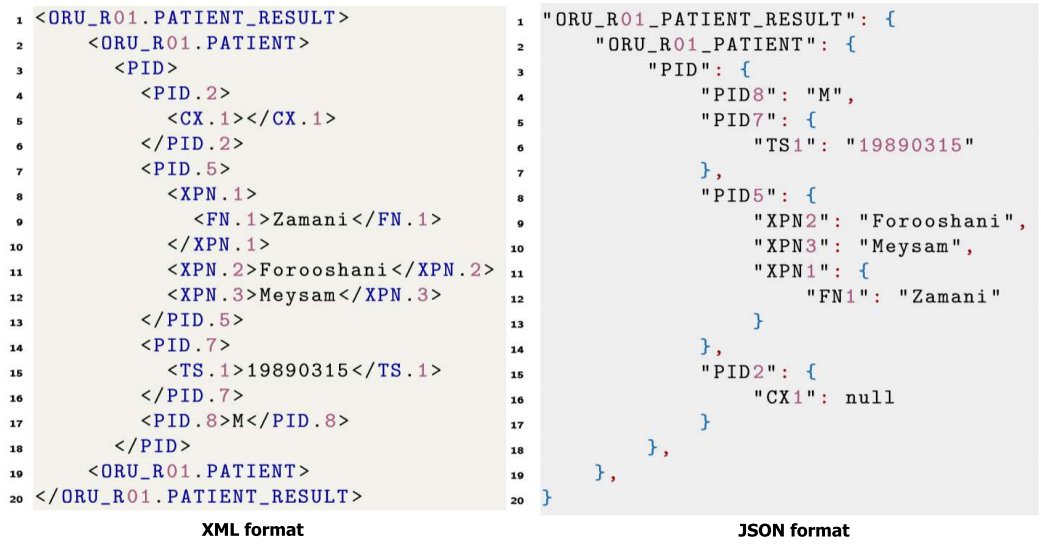

Figure 4.13: Instance of mapping XML format to JSON in PID segment

4.3.2.3 Extract HL7 response from JSON

For the XML data, there are usually a collection of namespaces given by the XML file, which specifies the scope of data. The first three hierarchy levels are for those namespaces and the HL7 response is in the 4th level. Since we only need the main HL7 response body, we can separate the 4th hierarchy level from the entire one and save it as the principal answer. Henceforth the main response is the HL7 message itself, and we can address it directly.

4.3.2.4 Validate HL7 response by checking the business rules

In addition to the business rules that we defined in section 4.2.1, we need some validation rules for data sources. Data validation is a method of checking the accuracy and quality of data sources. It can also be considered a form of data cleansing. Determine the overall health of the HL7 data and the changes required from the source data to match the schema in the target we defined in the 4.2.2 section. Then search for incongruous or incomplete counts, incorrect formats, and null field values. For example, The Age cannot be more than two digits, or the `Min/Max` value can not be empty (null). Data validation is typically performed using a scripting language such as Python to compose validation scripts.

Business rules are shown on table 4.2 are specified to control record insertion throughout the loading process. In the validation step, the same rules will be taken as consideration for validating the data source. in such a way, we will have two steps to validate the HL7 data during the process. The first is the validation of HL7 data sources, that it goes through all HL7 segments. And Second, it would be applying business rules for creating the physical data warehouse.

4.3.2.5 Repairing process

As a part of our assessment of the HL7 data, we can classify which errors can be repaired at the source data. This step is combined with the part of validation. The tool shows the validation results. If the data source is invalid, we should resolve the problem, and check the data source

again. It is a cycle that needs to fulfill the target scheme by including necessary repair methods to provide a complete and acceptable data source for feeding the single source of truth.

For example, if the values `Min` or `Max` are `NULL` in the analysis then mapping it to 0 for that field. By doing so, we are going to make sure, our pipeline will receive a number as it defined `FLOAT` type in the target schema. You can see this method is applied, in section 5.4.2.

4.3.2.6 Deriving new attributes from available data

One of the key tasks of the ETL process is to derive new attributes from the initial response. This process will be performed by applying the business rules to the HL7 response and extracting new measured values from existing data. The process will start by parsing the main HL7 response and extracting those values which can help us to define new attributes. Such new attributes might be new numerical attributes such as calculating the patient's age from `Date_Of_Birth` or non-numeric attributes such as classification of the patient's age and resulted in `Range_Of_Age` in the final result. In some cases aggregating data is beneficial. It will boost reporting performance, allow business logic to be applied to calculated measures, and make it easier for developers to understand the data. An example of deriving new measures is `relative_discrepancy` which we have defined it in section 4.2.2.

In general, any information that can satisfy our target data model needs to be extracted and given for the next step. In the next step, we will gather the information we have now prepared for it. Therefore, It is very relevant to seek a relationship between available data and trying to extract any useful information that could satisfy our final model.

4.3.2.7 Extract bio_values and patient_Identification from HL7 response

In this step, we will extract whatever data we need from each segment. The full description of each segment's element can be found in Appendix A. In this project, I split up the target attributes into two parts that we need for the final data model and save them in JSON responses:

- ***Patient_Identification:*** Here we have all the patient identifying details under the `patient` table that we need to create his/her record in the physical target data warehouse. This information is extracted from the `PID` segment of the HL7 response. (Details on the `PID` segment of HL7 response can be found in appendix A under section A.2).

- ***Bio_Values:*** In `Bio_Values`, we will save the results of patient observation extracted from the `OBX` segment of the HL7 response. (Details on the `OBX` segment of HL7 response can be found in Appendix A under section A.6).

4.3.2.8 Lookup HL7 keys to the target schema concepts

In this step, we would give a standard name with one standard definition for each specific data element. It can be done by changing the name of each segment's element, to the meaningful names after getting all the required information in the `Patient_Identification` and `Bio_values` responses.

Table 4.6 is shown that information from HL7 response, extracted from the `PID` and `ORC` segments, and corresponding new names specified for each of them:

Table 4.6: Lookup HL7 keys for `Patient_Identification` response

New_Name	Segment location in HL7 response
Name	[ORU_R01_PATIENT_RESULT] -> [ORU_R01_PATIENT] -> [PID] -> [PID5] -> [XPN3]
First Surname	[ORU_R01_PATIENT_RESULT] -> [ORU_R01_PATIENT] -> [PID] -> [PID5] -> [XPN1] -> [FN1]
Second Surname	[ORU_R01_PATIENT_RESULT] -> [ORU_R01_PATIENT] -> [PID] -> [PID5] -> [XPN2]
Date_Of_Birth	[ORU_R01_PATIENT_RESULT] -> [ORU_R01_PATIENT] -> [PID] -> [PID7] -> [TS1]
Sex	[ORU_R01_PATIENT_RESULT] -> [ORU_R01_PATIENT] -> [PID] -> [PID8]
DNI	[ORU_R01_ORDER_OBSERVATION] -> [0] -> [ORC] -> [ORC3] -> [EI1]

On the other hand, table 4.7 is shown that information from HL7 response, extracted from the `OBX` segment and corresponding new names specified for each of them:

(**Note:** All the observations in HL7 response are under `[ORU_R01_PATIENT_RESULT] -> [ORU_R01_ORDER_OBSERVATION]` that is called obs in the table 4.7.)

Table 4.7: Lookup HL7 keys for `Bio_Values` response

New_Name	Segment location in HL7 response
Title	obs -> [ORU_R01_OBSERVATION] -> [OBX] -> [OBX3] -> [CE2]
Code	obs -> [ORU_R01_OBSERVATION] -> [OBX] -> [OBX3] -> [CE1]
Group	obs -> [ORU_R01_OBSERVATION] -> [OBX] -> [OBX3] -> [CE3]
Min	obs -> [ORU_R01_OBSERVATION] -> [OBX] -> [OBX7]
Max	obs -> [ORU_R01_OBSERVATION] -> [OBX] -> [OBX7]
Units	obs -> [ORU_R01_OBSERVATION] -> [OBX] -> [OBX6] -> [CE2]
Value	obs -> [ORU_R01_OBSERVATION] -> [OBX] -> [OBX5]
Consequence	obs -> [ORU_R01_OBSERVATION] -> [OBX] -> [OBX8]

4.3.3 Load

This section describes the loading method used to load HL7 data into the data warehouse. loading data is the process that requires taking the transformed data and loading it where the consumers can access it.

Insert operation is involved in the loading of the data into the warehouse. In DW, we can insert new data or insert newer dimension data versions. To do this, we use the `INSERT` statement.

There are two primary methods of loading data into a DW: **Full load** method is to dump the entire dataset and then replace it entirely with the new, modified dataset. On the other hand, **Incremental load** method is to periodically apply the ongoing changes. In this thesis project, the incremental loading method is performed to apply continuous changes immediately and automatically after the transformation step and will be directly loaded into the data warehouse.

For incremental loading, we should review the whole ETL process to ensure that all process steps support incremental loading. How data is received is extremely important, since we may receive duplicate data, it will be essential to use a method to determine each entry in order to prevent duplicate records in our final result. Our approach for dealing with duplicate data will be a little complicated. In order to do this, we would try to select the unique record identifier that we want to insert into the specific table. The next step will be a restriction to check whether or not the length of the answer to the query is greater than zero or not. If it is greater than zero, it means that the current record exists in the target table and as such the function is passed without any insertion. In the case that the length of the answer to the query is equal to zero, it means, the given record does not exist in the target table and that the record can be entered in the corresponding table.

As you have seen in section 4.2, the Snowflake schema is provided as our data modeling solution and is ready to be fed with the result of the ETL tool that was developed in section 4.3. Essentially, we have two kinds of tables: Fact Table and Dimension Table. Foreign keys and primary keys defined to make such references between fact table and dimension tables. We always have to load data into dimensional tables first and then fact table. It is because if we first try to load data into the fact table, then primary key references from the connected dimensions can't be found. By first loading dimensional tables and then the fact table, we will guarantee that in fact table, foreign keys would find corresponding primary keys, from dimension tables. And we will successfully load the data into the target DW. Full details about each table insertion can be found in section 5.4.2.

Chapter 5

Implementation and Use

This chapter briefly describes how I built the `ETL_for_HL7` tool. I have decided to use the **python3** language because it is widely used in the implementation of ETL tools and fits the purpose of the thesis.

A tool is created that tries to encapsulate all the functionalities of an ETL process. The objective is to build an application that has all the functions of the ETL process to satisfy the main objectives of the thesis project. Data scientist only needs to use the final wrapper script to run the tool. We are going to see how this tool can be used and how the ETL process is created and adapted. The complete code of the ETL tool is accessible through the Git repository link below:

https://github.com/meysam24zamani/ETL_For_HL7

Requirement analysis is the first phase in designing the `ETL_for_HL7` tool. In this phase, we set the initial standards and conditions that the ETL tool has to satisfy. At a high level, the main requirement for the tool is to allow the user to easily execute the ETL tool and see the results in the output folder as well as loaded data into a data warehouse.

5.1 Purpose

The `ETL_for_HL7` is a tool that is provided for data analysts from different healthcare centers, in order to facilitate data preparation step for them for dealing with HL7 messages. Therefore we can assume that the final users of the application are the data analysis from different healthcare centers and laboratories. The tool would allow the user to input the relevant data sources (HL7 represented in the XML format file) and the database connection parameters. After running the tool, As a consequence, corresponding records should be created in the database. Alternatively, a folder will be created inside the `output` directory of the tool with the naming system of patient DNI, followed by related sample test date. Then inside would be two output JSON files named `Patient_Identification` and `Bio_Value` with the result values that we discussed in section 4.3.2. The implementation of this ETL tool has been entirely performed during the development of this thesis project. However, the idea and initial plan for having such a tool comes from the time that I worked in Genomecore company as an internship program.

5.2 Technology used and functional requirements

During this section I will detail the main functional requirements that the tool has to respect. The main technology used for creating `ETL_for_HL7` is Python. This Python application will use some packages that are not included in the standard library of python 3 programming language. Each of these libraries requires a specific version of the libraries. The solution for this problem is to build a virtual environment that includes a Python installation for a particular Python version, plus required packages.

```
# Creat a virtual envirnoment with all dependencies installed in python version 3.7
python3.7 -m venv env && source env/bin/activate && pip install --upgrade pip &&
pip install -r requirements.txt
```

Figure 5.1: Create a virtual environment with all dependencies

In this project, we have used `venv` that essentially allows us to create a "virtual" isolated Python installation and install packages into that virtual installation. The command to build `venv` for this project can be found in Figure 5.1. Also, you can find The list of required python packages used for creating this tool in table 5.1.

Table 5.1: Requiered Python packages in requirements.txt file

Package	Version
certifi	2019.11.28
chardet	3.0.4
idna	2.8
marshmallow	2.20.2
psycopg2	2.8.5
requests	2.22.0
setuptools	39.0.1
urllib3	1.25.8
xmltodict	0.12.0

5.2.1 Main user interface

5.2.1.1 Data source files

The user should be able to input the corresponding data source files by placing them in the `input_files` directory (Figure 5.2). The input files should be in XML format and the naming rule for the input files should be the patient's DNI following by the sample date/time. The filename example would be the same as `Y5694768M_2019-11-22T09-57-19.xml`

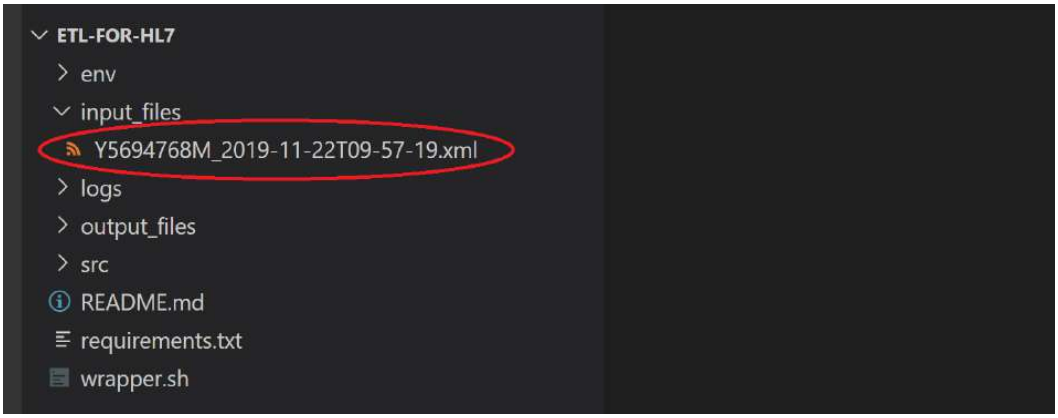

Figure 5.2: Input_files directory

My approach to the extraction of the source files would be to collect patient sample test by DNI of the patient and sample date/time. Nevertheless, it can be applied in other ways, such as via an API linked to a real laboratory, as discussed in the 4.3.1 section.

The data source available for this thesis project is generated by a real laboratory in Barcelona. The Genomcore company provided us this sample data and agreed that the name of the laboratory not be reported in this study. The available data gathers all the results of a patient's examination with fake information due to privacy issues requested by Genomcore company. A part of the sample data is provided in appendix C. However, you can fond the complete file in `this link`.

5.2.1.2 Input main parameters

The tool should allow the user to enter the main parameters to the tool. These parameters refer to the following details: patient ID(DNI) and sample date/time. They will be fetched by the `wrapper.sh` as arguments in the code. In the current implemented process, These two parameters would be passed to the `launch.py` by defining in the `wrapper.sh` script (Figure 5.3).

wrapper.sh

```
#!/bin/bash
set -euo pipefail

# Activate python environment
source env/bin/activate

# Set environment variables
export SAMPLE_DATE='2019-11-22T09:57:19'
export SAMPLE_ID='Y5694768M'
export MAIN_SCRIPT="src/main.py"
export PYTHONPATH=$PWD/src:$PWD/src/tools

python src/launch.py \
    --sample-date ${SAMPLE_DATE} \
    --sample-id ${SAMPLE_ID} \
```

Figure 5.3: Input main python script parameters

The reason I decided to implement it with passing parameters is that I found it user-friendly for those consumers who want to implement the same kind of tool in a web application. Then, through a web application, it would be possible for the user to enter the corresponding parameters, by placing them inside a simple text box, and by clicking the post button, such parameters would be passed to the `wrapper.sh` file.

5.2.1.3 Input DB parameters

The user will add the connection parameters of the existing database to the `connection.py` file. These parameters refer to the following details: `<DB_NAME, DB_USER, DB_PASS, DB_HOST, and DB_PORT>` and will be fetched by the `connection.py` that are using the connection to the database.

5.2.1.4 Run the ETL_for_HL7 tool

Once the above steps have been completed, the user will be able to run the ETL tool by running the wrapper script. Running `wrapper.sh` will start the tool by running the `launch.py` with defined arguments. Later in section 5.4, I will describe the entire process after running this script.

```
./wrapper.sh
```

5.2.2 Output interface

5.2.2.1 By running schema creation script(schema.py)

After running `schema.py`, the terminal window should display information regarding the Success/Failure status for creating schema with all Dimension/Fact tables into the database (Figure 5.4). These information shows that the schema is created into the DW. It is supposed to run this script once and before start using the main script.

```
(env) meysam24zamani@meysam24zamani-N501VW:~/Workspace/TFM/ETL-for-HL7$ python src/database/schema.py
Database connect successfully...
All tables are deleted successfully
---------------------------------
---------------------------------
Table -> Dim_Patient is created successfully
Table -> Dim_Date is created successfully
Table -> Dim_Group is created successfully
Table -> Dim_Type_Analysis is created successfully
Table -> Fact_Observation is created successfully
(env) meysam24zamani@meysam24zamani-N501VW:~/Workspace/TFM/ETL-for-HL7$
```

Figure 5.4: Terminal view after running the schema creation script

5.2.2.2 By running main script(wrapper.sh)

After running `wrapper.sh`, the terminal window should display information regarding the `Success/Failure` status for connecting and inserting records into the database as well as final completion status (Figure 5.5). This information indicates that the respective records are successfully created in the database. You can find complete information about final results in chapter 6.

```
meysam24zamani@meysam24zamani-N501VW:~/Workspace/TFM/ETL-for-HL7$ source env/bin/activate
(env) meysam24zamani@meysam24zamani-N501VW:~/Workspace/TFM/ETL-for-HL7$ ./wrapper.sh
Database connect successfully...
2020-05-12 01:22:42 [INFO] -- ETL_for_HL7
Record created successfully in Dim_Patient table
Record created successfully in Dim_Date table
Record created successfully in Dim_Group table
Record created successfully in Dim_Type_Analysis table
Record created successfully in Fact_Observation table
2020-05-12 01:22:43 [INFO] -- Complete successfully
(env) meysam24zamani@meysam24zamani-N501VW:~/Workspace/TFM/ETL-for-HL7$
```

Figure 5.5: Terminal view after running the main script

5.3 Non-functional requirements satisfaction

As we defined in section 1.3.1, we have proposed a list of characteristics for ETL process quality and now we are going through them to see if we meet all requirements to satisfy those special characteristics or not. Showing that, we will repeat table 1.1 and add the behaviour of created ETL tool in table 5.2.

Table 5.2: Non-functional requirements satisfaction

Name	Requirement	Available system behaviour
Flexibility	The ability of the ETL flow to provide alternative options and dynamically adjust to environmental changes	-The tool can expect new endpoint easily due to defining connections separately from the main workflow. -The system has a workflow engine with dynamic rules. -The system actively listens to the business rules, at the initial design phases.
Reusability	The ability to use the ETL process for the operations of other processes.	-The source code of the system is reusable due to the standard architecture that used for defining functionalities. In section 5.4, you will see, how different functionalities implemented in user-friendly structure and the way we connected them together in order to provide the ability to change segment mapping without touching all the code. Any new segment mapping structure can define easily and it brings reusability characteristic for our ETL tool.
Understandability	The clearness and self descriptiveness of the ETL process model for (non-technical) end users.	-The user can navigate through the source files easily as you can see the user-friendly structure files in Figure 5.6. -The code is commented for all functionalities and methods.
Maintainability	The ability of effectiveness and efficiency with which the ETL process can be modified to implement any future changes.	-Due to the explanation in **Reusability**, The system is designed in a way that it is easy to add new functionalities. -The system easily can cater to future changes through flexible architecture, design, and implementation. It is because as you can see in Figure 5.6, we defined the main aspects of the process in different files that it makes it flexible for future changes.
Testability	The ability to which the process can be tested for feasibility, functional correctness, and performance prediction.	-The system can be test through the validation step thanks to the Marshmallow package of python3 programming language. -The system is recording all failures in a persisted environment.

5.4 Implementation steps

This section briefly explains how I implemented the `ETL_for_HL7` tool. As I said before, it is an easy-to-use implementation attempt, and for that, I have been trying to build ETL steps in a simple structure. The file structure of the tool is presented in Figure 5.6. There are mainly 5 key directories and a bash script called `wrapper.sh` in the root of the application. The directories are **env**, **input_files**, **logs**, **output_files**, and **src**. The **env** folder is created when we run the code of Figure 5.1. **input_files** directory is the location where we have to put whatever input files, we have. The **logs** folder has `log_main.log` file inside which is responsible for maintaining execution logs. The **output_files** directory is the place where the output is stored in JSON format. This alternative result consists of a folder with the execution timestamp in the folder name for each execution, and there are two JSON files inside. One of which is `Patient_Identification` and the other is `Bio_value` for a specific patient's test results. Finally, the last directory that would be the main one with all the functional python files is the **src** folder. There are mainly three folders inside the **src** which are 1-database, 2-etl, and 3-tools that I will go through them and explain what their functionality is and what they have inside. Besides, the complete code of each file can be accessed through the Git repository provided for this project (https://github.com/meysam24zamani/ETL_For_HL7).

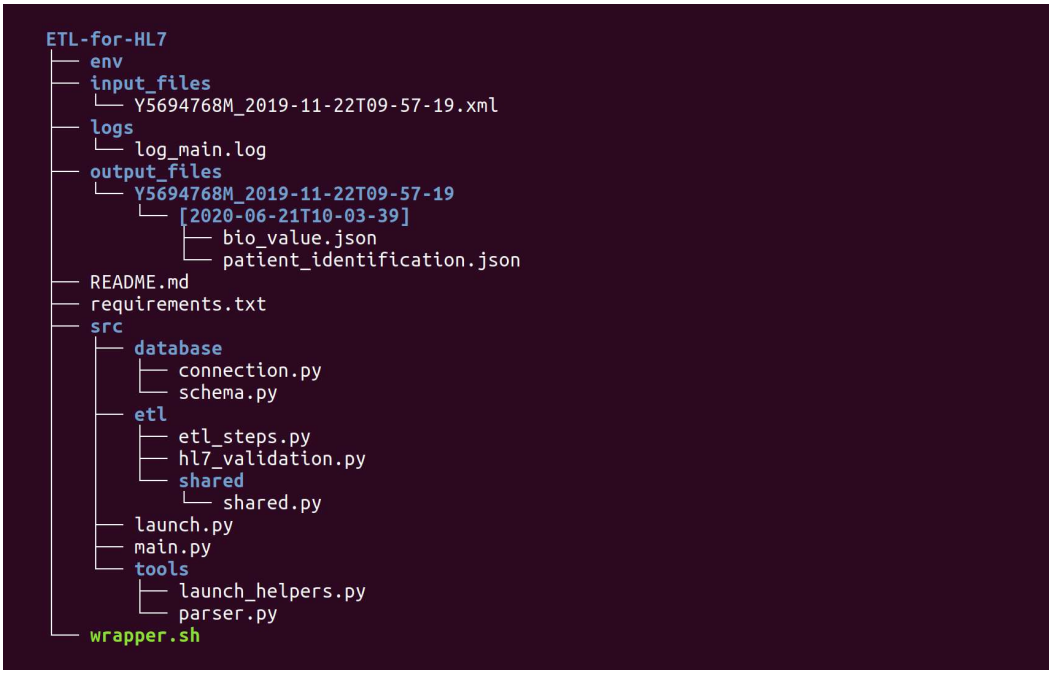

Figure 5.6: Effective folder structures to organize files in `ETL_for_HL7`

5.4.1 PostgreSQL as a data warehousing solution

In this thesis project , I have decided to use `PostgreSQL` for creating the data warehouse. `PostgreSQL`, as one of the most advanced open-source database, is so adaptable that it can represent as a simple, efficient and cost-effective data warehousing solution. It provides powerful SQL query capabilities, and other tools can use `PostgreSQL` to provide business intelligence, analytics , and data mining. You can also integrate it with a variety of analytical methods. Some advantages of using `PostgreSQL` as a data warehouse are listed in Table 5.3.

Table 5.3: Benefits of using PostgreSQL as a data warehouse

Property	Description
Cost	If you're using an on-prem environment, the cost for the product itself will be free.
Scale	You can scale reads it in a simple way by adding as many replica nodes as you want.
Performance	With a correct configuration, PostgreSQL has a really good performance on different scenarios.
Compatibility	You can integrate PostgreSQL with external tools or applications for data mining, OLAP and reporting.
Extensibility	PostgreSQL has user-defined data types and functions.

The `connection.py` file is for connecting the database with the tool. As you can see in te code, I used `psycopg2` python package to make a connection between the tool and PostgreSQL. As explained in the 5.2.1.3 section, there are 5 input parameters for a database connection that I have listed in table 5.4.

Table 5.4: Database connection parameters

Property	Description
DB_NAME	The DATABASE NAME
DB_USER	The USERNAME of the user who have access to the database
DB_PASS	The PASSWORD of the user who have access to the database
DB_HOST	The HOST IP ADDRESS
DB_PORT	The PORT of the host

Now we use `schema.py` to create tables, and for that, we use the creation queries explained in section 4.2.3. Note that such a python script will be executed once, in order to create the schema in PostgreSQL with all the necessary tables.

5.4.2 ETL process

In this section we will review all I have done in the ETL process and see the corresponding code related to each part. As I said before, a user-friendly implementation would be attempted. Therefore, The ETL steps are implemented in a single file named `etl_steps.py`. Of course, some cases have been implemented in separate files that I will mention them respectively.

`etl_steps.py` is the main file containing all steps of extraction, transformation, and load inside. As I said before, all input files should be in the `input_files` directory, and the name of the files should have a patient ID and a sample date separated by an underscore. thus, we will collect each source file by two parameters in the extraction phase (Figure 5.7). The first is the patient id named `sample_id` in the code and the second is the patient test result named `sample_date_name`.

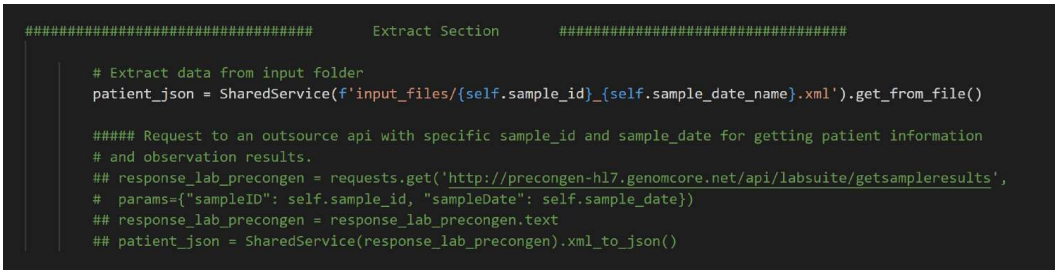

Figure 5.7: Extraction step in the code

As I said earlier, the alternate method for collecting data would be through an API from laboratories. There is a commented part in Figure 5.7 that contains the code for extracting data through an API from the Genomecore company.

After receiving the HL7 file (HL7 represented in XML file format), the transformation phase begins. The first step would be to unescape the decoding XML file with `xml.sax.saxutils` python package. This step is followed by Mapping XML content to the python dictionary and loading the code as JSON format. These two steps are implemented in the `shared.py`. You can see the related function in Figure 5.8.

```
def get_from_file(self):

    if self.format_of_the_file == 'xml':
        data = self.xml_to_json_from_file()
    else:
        with open(self.response) as json_file:
            data = json.load(json_file)
    return data

def xml_to_json_from_file(self):
    patient_xml = json.loads(json.dumps(xmltodict.parse(unescape(open(self.response).read()))))
    return patient_xml
```

Figure 5.8: Unescape XML code, Mapping to dict and load as JSON format

The next step will be to extract HL7 response information from the prepared JSON response. Looking at the JSON response, you would see three inessential hierarchy level just at the beginning of the XML file. Since the main response is the HL7 part, we need to get the 4th hierarchy level that has all the HL7 message inside. To have this response accessible for all other functions, we will pass it to an object named self.`labsuite_json` which makes it available in the whole EtlService class (Figure 5.9).

```
# Getting main HL7 response which is in level 4 of hierarchy result
xml_inside_json = patient_json.get('soap:Envelope',
                                   {}).get('soap:Body',
                                           {}).get('ObtenerResultadosMuestraResponse',
                                                   {}).get('ObtenerResultadosMuestraResult',
                                                           {}).get('ORU_R01',
                                                           {})
analysis = schema.load(xml_inside_json).data
self.labsuite_json = analysis
return analysis
```

Figure 5.9: Getting HL7 response from 4th hierarchy level

Since we have the whole HL7 response available in a Python object, Now we can start the validation step. In Python, there is a library called `marshmallow` that its main functionality is the validation of Python objects. Schema function in `marshmallow` will be used and keeps track of the data format. The schema function is used for dictating the data format that is sent to

the ETL tool. In order to validate the entire HL7 response, we need to go through all segments once and validate it by previously defined business rules. The validation part is implemented in `hl7_validation.py` file.

When the HL7 response be validated, we need to carefully review all business rules described by the consumers (data analysts who will be the final consumers of the ETL result) in order to derive new attributes and measures from the available data. As explained in the section 4.3.2.6, `Age` and `Range_Of_Age` are two examples of such attributes and `relative_discrepancy` is an example of derive a new measure to satisfy the target schema.

The last step of transformation will be to collect such information and prepare them for the next step of the process. We also like to divide those information into two parts and save them in separate JSON responses. one of which is `Patient_Identification` and the other is `Bio_value` for a specific patient's test results. You will find full details about the attributes in section 4.3.2. Since these data are defined in a deep hierarchical level of HL7 response, we are going to lookup HL7 keys to the target schema concepts as you have seen them in tables 4.6 and 4.7. The corresponding code for the last two transformation steps are available in Figures 5.10 and 5.11.

```
observations = analysis['ORU_R01_PATIENT_RESULT']['ORU_R01_ORDER_OBSERVATION']

patient_Identification = {}

patient_Identification['name'] = analysis['ORU_R01_PATIENT_RESULT']['ORU_R01_PATIENT']['PID']['PID5']['XPN3']

patient_Identification['surname1'] = analysis['ORU_R01_PATIENT_RESULT']['ORU_R01_PATIENT']['PID']['PID5']['XPN1']['FN1']

patient_Identification['surname2'] = analysis['ORU_R01_PATIENT_RESULT']['ORU_R01_PATIENT']['PID']['PID5']['XPN2']

patient_Identification['sex'] = analysis['ORU_R01_PATIENT_RESULT']['ORU_R01_PATIENT']['PID']['PID8']

patient_Identification['dni'] = observations[0]['ORC']['ORC3']['EI1']

patient_Identification['dateOfBirth'] = analysis['ORU_R01_PATIENT_RESULT']['ORU_R01_PATIENT']['PID']['PID7']['TS1']


current_year = int(strftime("%Y", gmtime()))
patient_birth_year = int(patient_Identification['dateOfBirth'][:4])
age = current_year - patient_birth_year
if age < 40:
    range_age = "young"
elif  40 <= age < 60:
    range_age = "middle_aged"
else:
    range_age = "old"

patient_Identification['age'] = age

patient_Identification['range_age'] = range_age
```

Figure 5.10: Corresponding code for creating `Patient_Identification` response

Now we have all the data available in two main python objects that are `Patient_Identification` and `bio_value`. We can start loading data into the `PostgreSQL` database that we built in section 5.4.1. Before that, I would like to save `Patient_Identification` and `Bio_value` in to the `output_files` directory. You can find corresponding JSON files in 6.1.

In order to load transformed data into `PostgreSQL`, we need to start with dimension tables as we stated in section 4.3.3. It is obligatory to start with the dimension tables because we will use the primary keys to reference them in the fact table. we are going to show you the corresponding code I have used to create records in each table and perform the relationships between tables. In order to organize the content of this section,I have specified the name of each part as the table's title.

```
for obs in observations:
    bio_value = {}

    range_valid = obs['ORU_R01_OBSERVATION']['OBX']['OBX7']
    range_valid = range_valid.split(' - ')

    bio_value['title'] = obs['ORU_R01_OBSERVATION']['OBX']['OBX3']['CE2']
    bio_value['code'] = obs['ORU_R01_OBSERVATION']['OBX']['OBX3']['CE1']
    bio_value['group'] = obs['ORU_R01_OBSERVATION']['OBX']['OBX3']['CE3']

    stringmin = range_valid[0]
    bio_value['min'] = float(stringmin.replace(",","."))

    stringmax = range_valid[1]
    bio_value['max'] = float(stringmax.replace(",","."))

    bio_value['units'] = obs['ORU_R01_OBSERVATION']['OBX']['OBX6']['CE2']

    stringvalue = obs['ORU_R01_OBSERVATION']['OBX']['OBX5']
    bio_value['value'] = float(stringvalue.replace(",","."))

    if bio_value['value'] > bio_value['max']:
        absolute_diff = abs(bio_value['max'] - bio_value['value'])
        relative_diff = absolute_diff / bio_value['max']
        relative_diff = round(relative_diff, 2)
    elif bio_value['value'] < bio_value['min']:
        absolute_diff = abs(bio_value['min'] - bio_value['value'])
        relative_diff = absolute_diff / bio_value['min']
        relative_diff = round(relative_diff, 2)
    else:
        relative_diff = 0
    bio_value['relative_discrepancy'] = relative_diff

    bio_value['consequence'] = obs['ORU_R01_OBSERVATION']['OBX']['OBX8']

    bio_value_list.append(bio_value)
```

Figure 5.11: Corresponding code for creating `bio_value` response

5.4.2.1 Insert data into Dim_Patient

`patient_Identification` is the first information I am going to insert into the `Dim_Patient` table. As you can see in Figure 5.12, the first step is to pass the corresponding `patient_Identification` data into new variables with the **"value_"** prefix. In the second step, we must ensure that the patient is unique and that there is no pre-insert with the same patient identification number (DNI in this project). I have to make sure there is not the same patient in the table, and if there is one, just pass the code to next step. The final step is to create a patient record in the `Dim_Patient`, and then we can print a positive message to the terminal for the first insertion.

```
#Query1: Insert data into Dim_Patient table.

value_name = str(response['patient_Identification']['name'])
value_surname1 = str(response['patient_Identification']['surname1'])
value_surname2 = str(response['patient_Identification']['surname2'])
value_date_of_birth = str(response['patient_Identification']['dateOfBirth'])
value_age = str(response['patient_Identification']['age'])
value_range_age = str(response['patient_Identification']['range_age'])
value_sex = str(response['patient_Identification']['sex'])
value_dni = str(response['patient_Identification']['dni'])


cur.execute("SELECT dni FROM Dim_Patient WHERE dni = %s", (value_dni,))
if len(cur.fetchall()) > 0:
    pass
else:
    query1 = "INSERT INTO Dim_Patient (name, surname1, surname2, date_of_birth, age, range_age," \
    "sex, dni) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
    cur.execute(query1, (value_name, value_surname1, value_surname2,  \
    value_date_of_birth, value_age, value_range_age, value_sex, value_dni))
print("Record created successfully in Dim_Patient table")
conn.commit()
```

Figure 5.12: Insert data into `Dim_Patient` table

5.4.2.2 Insert data into Dim_Date

The Dim_Date table has information about the date/time of observation that is one of the main parameters in the tool. There were essentially two key parameters in the extraction process, one of which is the sample_date. The sample_date type is timestamp and needs to be stored separately in a dimention table connected to the main fact table. For loading the sample_date into the Dim_Date table, the sample_date needs to be parsed and then extract the year, the month, and the day. Like the previous insertion of the Dim_Patient table, I called new variables with the **"value_"** prefix and will use them for the insertion query.

```
#Query2: Insert data into Dim_Date table.

value_sample_date = str(response['bio_value']['section_bio_values']['sample_date'])
value_sample_timestamp = str(response['bio_value']['section_bio_values']['sample_timestamp'])
value_date = value_sample_date[:8]
value_day = value_date[6:8]
value_month = value_date[4:6]
value_year = value_date[0:4]


cur.execute("SELECT sample_timestamp FROM Dim_Date WHERE sample_timestamp = %s", (value_sample_timestamp,))
if len(cur.fetchall()) > 0:
    pass
else:
    query2 = "INSERT INTO Dim_Date (sample_timestamp, day, month, year) VALUES (%s, %s, %s, %s)"
    cur.execute(query2, (value_sample_timestamp, value_day, value_month, value_year))
print("Record created successfully in Dim_Date table")
conn.commit()
```

Figure 5.13: Insert data into Dim_Date table

Inserting duplicate data is not permissible since the timestamp for each observation must be unique. The sample_date in timestamp format is the primary_key and must be unique. The way I used to check whether or not the time stamp exists in the Dim_Date table is like the method I used for the patient ID number. It will occur by selecting timestamp from the current sample_date, and if any record can be found with the same timestamp in Dim_Date table, it will ignore it to avoid duplicate information created in the database. You will find the corresponding code in the Figure 5.13.

5.4.2.3 Insert data into Dim_Group

The Dim_Group table has a list of group analysis and is in the first level granularity of the Dim_Type_Analysis table. In order to insert all existing group analysis from bio_values, I need to create a loop that can pass through each of the analysis in a particular observation and collect all the group analysis. We have to take into account that the duplicate name of the group is not acceptable, and to monitor it for the duplicate group name we have to enforce a script like the previous insertion. The corresponding code for this section can be found in Figure 5.14.

5.4.2.4 Insert data into Dim_Type_Analysis

The Dim_Type_Analysis table has a list of information for each analysis type. Loading data into the Dim_Type_Analysis is almost the same as the Dim_Group table insertion. The only difference from the previous one is the connection between Dim_Type_Analysis and Dim_Group table. this connection will be perform by the id of the Dim_Group table as primary key and group_id of the Dim_Type_Analysis as the foreign key. For the rest of the attributes, we need to create a loop in order to pass through each of the analysis types in a single observation and collect all the values related to each attribute. The duplicate analysis type is not acceptable and we need to implement the code such that avoid insertion of the duplicate type of analysis. You can see the corresponding code for this section in Figure 5.15.

```
#Query3: Insert data into Dim_Group table.

value_all_values = response['bio_value']['section_bio_values']['value']

for key in value_all_values:
    value_group_name = str(key['group'])
    cur.execute("SELECT name FROM Dim_Group WHERE name = %s", (value_group_name,))
    if len(cur.fetchall()) > 0:
        pass
    else:
        query3 = "INSERT INTO Dim_Group (name) VALUES (%s)"
        cur.execute(query3, (value_group_name,))

print("Record created successfully in Dim_Group table")
conn.commit()
```

Figure 5.14: Insert data into `Dim_Group` table

```
#Query4: Insert data into Dim_Type_Analysis table.
for key in value_all_values:
    value_group_name2 = str(key['group'])
    cur.execute("SELECT id FROM Dim_Group WHERE name = %s", (value_group_name2,))
    rows = cur.fetchall()
    for y in rows:
        value_group_id = str(y[0])

    value_name = str(key['title'])
    value_code = str(key['code'])
    value_min = str(key['min'])
    value_max = str(key['max'])
    value_units = str(key['units'])

    cur.execute("SELECT code FROM Dim_Type_Analysis WHERE code = %s", (value_code,))
    if len(cur.fetchall()) > 0:
        pass
    else:
        query4 = "INSERT INTO Dim_Type_Analysis (code, group_id, name, min, max, units)" \
        "VALUES (%s, %s, %s, %s, %s, %s)"
        cur.execute(query4, (value_code, value_group_id, value_name,  \
        value_min, value_max, value_units))
print("Record created successfully in Dim_Type_Analysis table")
conn.commit()
```

Figure 5.15: Insert data into `Dim_Type_Analysis` table

5.4.2.5 Insert data into Fact_Observation

Finally, in the last step, we will insert the main values of each analysis type into the `Fact_Observation` table followed by `consequence` attribute and also `relative_discrepancy` which is a new derived measure in our final model. `Fact_Observation` is the main fact table of the data model and all `primary_keys` from dimension tables must be referenced to corresponding foreign keys in this table. For `Dim_Patient`, we need `patient_id` foreign key, for `Dim_Date`, we need `sample_date_id` foreign key, and for `Dim_Type_Analysis`, we need `type_analysis_id` foreign key. You will see the corresponding code for this section in Figure 5.16.

5.4.3 Tools for running the main script

In this section, we will see how the `main.py` script works and the related tools that help with this execution. We are going to use the result of section 5.4.2 and place it into the execution process. As you can see in `main.py`, I imported **EtlService** from `etl_steps.py` and used it in

```python
#Query5: Insert data into Fact_Observation table.
for key in value_all_values:
    value_result = str(key['value'])
    value_relative_discrepancy = str(key['relative_discrepancy'])
    value_code = str(key['code'])
    value_consequence = str(key['consequence'])

    cur.execute("SELECT id FROM Dim_Patient WHERE dni = %s", (value_dni,))
    rows = cur.fetchall()
    for r in rows:
        value_Patient_id = str(r[0])

    cur.execute("SELECT patient_id, sample_date_id, type_analysis_id, " \
    + "result_value FROM Fact_Observation WHERE patient_id = %s AND sample_date_id = %s AND " \
    + "type_analysis_id = %s AND result_value = %s", (value_Patient_id, value_sample_timestamp, \
    value_code, value_result))
    if len(cur.fetchall()) > 0:
        pass
    else:
        query5 = "INSERT INTO Fact_Observation (patient_id, sample_date_id, type_analysis_id, " \
        "result_value, relative_discrepancy, consequence) VALUES (%s, %s, %s, %s, %s, %s)"
        cur.execute(query5, (value_Patient_id, value_sample_timestamp, value_code, value_result, \
        value_relative_discrepancy, value_consequence))
print("Record created successfully in Fact_Observation table")
conn.commit()
```

Figure 5.16: Insert data into `Fact_Observation` table

`run_tool` function. The `run_tool` function used, in order to execute the ETL process, and if all goes well, we can see the "Complete successfully" log in the terminal.

Eventually, in `launch.py`, we are going to run the `main.py` with patient ID and sample date/time as parameters for selecting the appropriate XML file from the `input_files` directory and run the whole process. As a result, the corresponding records will be created in `PostgreSQL` that I am going to explain it in chapter 6. In addition, as I said before, you can find the result in the `output_files` directory in JSON format. In the end, I would like to mention that in order to prepare logs for the main script running, we will use `launch_helpers.py` and `parser.py` to manage `STDERR` and `STDOUT` with the Python `subprocess` library and save them in `log_main.log`.

Chapter 6

Results

The results obtained after applying the described practical techniques in section 4 and the implementation described in section 5 are detailed in this chapter.

It is important to highlight that the `ETL_for_HL7` have been tested using the sample data which Genomecore company gave it to me, so the documentation would be based on their sample data that you can find it in Appendix C.

`ETL_for_HL7` is an ETL tool for managing HL7 messages that usually come from health centers and laboratories. The tool consists of the full implementation of the Extraction, Transformation, and Loading steps for this type of information. After testing the tool, the expected result was obtained and I am going to explain it in the following sections.

6.1 Output in JSON format

As shown in Figures 5.10 and 5.11, I saved the final result in two Python objects. One of them contains `patient_Identificaiton` and the other contains `bio_value` information. Both responses were converted into JSON files and saved into the `output_files` directory.

The idea of saving them in JSON format, before inserting them into the data warehouse, is that some supplementary analysis pipelines, need input data in JSON format. with this manner, we will reduce the cost of implementation process for those pipelines as I had this experience in Genomecore company. The JSON responses of the sample test can be found in Appendix B under section B.1 and B.2.

6.2 Final result in PostgreSQL

In section 5.4.2, I have shown you the way I extract, transform , and load data into `PostgreSQL`. In this section, I am going to show you the exact results that I obtain in each step of the loading process. `PgAdmin` is the tool that I used to show the final results. `PgAdmin` is a widely used database management tool for the `PostgreSQL` community. It helps to simplify the creation, maintenance, and use of database objects by providing a clean and user-friendly interface.

The first table that is receiving the corresponding record is the `Dim_Patient`. Figure 6.1 shows the record created in the `Dim_Patient` table after the steps in section 5.4.2 have been completed.

	id [PK] integer	name character varying (500)	surname1 character varying (500)	surname2 character varying (500)	date_of_birth date	age integer	range_age character varying (200)	sex character varying (70)	dni character varying (200)
1	1	Meysam	Zamani	Forooshani	1989-03-15	31	young	M	Y5694768M

Figure 6.1: Created record in `Dim_Patient` table

The Dim_Date is the next table that is received the corresponding record. As you can see in Figure 6.2, by using the sample_date/time and converting it to the timestamp format, It is used as the Primary key of the created record. And of course, we have the Year, the Month and the day, placed separately in the following columns.

	sample_timestamp [PK] timestamp without time zone	day integer	month integer	year integer
1	2019-11-22 09:57:19	22	11	2019

Figure 6.2: Created record in Dim_Date table

The Dim_Group is the next table where the data inserted into it. As you can see in Figure 6.3, there are 5 different group analysis in the available sample data.

	id [PK] integer	name character varying (200)
1	1	Hematología y Hemostasia
2	2	Bioquímica
3	3	Endocrinología (suero)
4	4	Inmunología
5	5	Monitorización de Fármacos y Toxicología

Figure 6.3: Created record in Dim_Group table

id is The primary key in the Dim_Group table that has integer type assigned to each group analysis. This primary key is the group_id reference in the Dim_Type_Analysis table which is located in the second column of the Dim_Type_Analysis table (Figure 6.4).

	code [PK] character varying (200)	group_id integer	name character varying (200)	min double precision	max double precision	units character varying (200)
1	HEM	1	Hematíes	4.1	5.75	x10&6/mm³
2	HB	1	Hemoglobina	12.5	17.2	g/dL
3	HCTO	1	Hematocrito	36.5	50.5	%
4	VCM	1	VCM	78	99	fL
5	HCM	1	HCM	26	33.5	pg
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
34	TES	3	Testosterona total	2.6	10	ng/mL
35	COR	3	Cortisol	5	23	µg/dL
36	INSU	3	Insulina	2	25	mU/L
37	PCU	4	Proteína C Reactiva en suero (...	0	3	mg/L
38	ZN	5	Zinc en suero	60	120	µg/dL

Figure 6.4: Created record in Dim_Type_Analysis table

As you can see in Figure 6.4, there are 38 different types of analysis for each patient observation. In Dim_Type_Analysis table, we used the code of each type analysis as the primary key and referred it to the type_analysis_id, in the Fact_Observation table. The Fact_Observation table is the main fact table in the target data model.

And finally, the main result's value of the analysis are located in the `Fact_Observation` table. As you can see in Figure 6.5, there are three foreign keys, each of which has a reference from one of the related dimension tables.

- The first foreign key is `patient_id`, which indicates the corresponding patient Id number from the `Dim_Patient` table.
- The second foreign key is the `sample_date_id`, which has the data/time of the sample test. The first two columns will have a fixed value for all 38 types of analysis that inserted for each patient's test result, because all of them come from one observation in a specific date/time, and from a particular patient.
- The third foreign key is `type_analysis_id`, which has a code reference for each type of analysis from `Dim_Type_Analysis`.

There is a result value for each type of analysis that we have in the `result_value` column of the `Fact_Observation` table followed by `consequence` value that indicates the abnormal flag and also `relative_discrepancy` which has calculated value, following the formula represented in section 4.2.2. (Figure 6.5).

	patient_id integer	sample_date_id timestamp without time zone	type_analysis_id character varying (200)	result_value double precision	relative_discrepancy double precision	consequence character varying (200)
1	1	2019-11-22 09:57:19	HEM	4.83	0	N
2	1	2019-11-22 09:57:19	HB	15.5	0	N
3	1	2019-11-22 09:57:19	HCTO	44.3	0	N
4	1	2019-11-22 09:57:19	VCM	92	0	N
5	1	2019-11-22 09:57:19	HCM	32.1	0	N
6	1	2019-11-22 09:57:19	CHCM	35	0	N
7	1	2019-11-22 09:57:19	RDW	12.8	0	N
8	1	2019-11-22 09:57:19	LEUC	4.9	0	N
9	1	2019-11-22 09:57:19	NEUT%	51.8	0	N
10	1	2019-11-22 09:57:19	LINF%	34.2	0	N
11	1	2019-11-22 09:57:19	MONO%	10.6	0.12	H
12	1	2019-11-22 09:57:19	EOS%	2.7	0	N
13	1	2019-11-22 09:57:19	BASO%	0.7	0	N
14	1	2019-11-22 09:57:19	NEUT	2.5	0	N
15	1	2019-11-22 09:57:19	LINF	1.7	0	N
16	1	2019-11-22 09:57:19	MONO	0.5	0	N
17	1	2019-11-22 09:57:19	EOS	0.1	0	N
18	1	2019-11-22 09:57:19	BASO	0	0	N
19	1	2019-11-22 09:57:19	PLQ	182	0	N
20	1	2019-11-22 09:57:19	VMP	10.3	0	N
21	1	2019-11-22 09:57:19	GS	92	0	N
22	1	2019-11-22 09:57:19	HG	5.3	0	N
23	1	2019-11-22 09:57:19	HGIF	34	0	N
24	1	2019-11-22 09:57:19	CT	247	0.23	H
25	1	2019-11-22 09:57:19	CH	59	0	N
26	1	2019-11-22 09:57:19	TG	105	0	N
27	1	2019-11-22 09:57:19	MG	2.15	0	N
28	1	2019-11-22 09:57:19	LDH	314	0	N
29	1	2019-11-22 09:57:19	CPK	93	0	N
30	1	2019-11-22 09:57:19	FERRI	210	0	N
31	1	2019-11-22 09:57:19	CAT	9.4	0	N
32	1	2019-11-22 09:57:19	HOMOC	12.38	0	N
33	1	2019-11-22 09:57:19	25OH	19.2	0.36	L
34	1	2019-11-22 09:57:19	TES	3.08	0	N
35	1	2019-11-22 09:57:19	COR	6.38	0	N
36	1	2019-11-22 09:57:19	INSU	5.74	0	N
37	1	2019-11-22 09:57:19	PCU	0.5	0	N
38	1	2019-11-22 09:57:19	ZN	79	0	N

Figure 6.5: Created record in `Fact_Observation` table

Chapter 7

Conclusion

In this chapter, the main conclusions of the thesis are presented and it is reviewed if the initial objectives were fulfilled. In this document, I have designed and implemented an ETL tool for dealing with HL7 messages represented in XML format file.

First of all I would like to point out the advantages which I have noticed while using a conceptual approach for modelling the ETL process. I have managed to conceptually model the operations during the extraction, transformation and insertion stages. As paper [33] indicates, there are two perspectives for the representation of ETL process: the control flow view and the data flow view. In this project, I have used such definitions to represent the ETL workflow at two different levels. First, by modeling the ETL flow at a higher conceptual level, I was able to determine how the processes interact with each other. Second, at the dynamic level, I have been able to understand how data flows between components and better handle the different stages during the data integration process.

`ETL_for_HL7` is a tool designed for health data sources. I have started by defining the HL7 message format from a semantic point of view. The role of each segment has been explained and the XML representation of the HL7 message has been provided. I clarified that some other versions of HL7 are available, but we were focusing on HL7 version2 in this project. Last year, I worked as a trainee in Genomcore company, and I found an idea to create an ETL tool for HL7 messages. The sample data used in this thesis project was given by Dr. Oscar Flores (CEO of Genomcore company).

The tool starts to work by collecting data using the patient ID and sample date/time. Then, in the next step, the available data, which is the XML representation of the HL7 message, enter to the transformation process. In the transformation step, several control flows were designed and the one that performs better was the one I presented in 4.10. The transformation process starts with unescaping XML text in order to remove traces of offending characters that could be incorrectly interpreted as markup. Follow this by mapping XML to the dictionary and loading it as a JSON format for a more comfortable parse of the data. Since the HL7 message starts from the fourth hierarchy level of the available data, the first step for parsing data would be to extract HL7 response information from the entire response. After getting the HL7 response, validate the data by reviewing the business rules and adding a control section for each segment of the HL7 message. The repair process is done at the same time as the validation step, and now the data is ready for deriving new attributes depending on target schema. The `Age` and `Range_Of_Age` are two examples of such attributes and `relative_discrepancy` is an example of derive a new measure to satisfy the target schema. Since the data model that we decided to choose is a demand-driven approach, in the final transformation step, I decided to create two main categories that we need to feed the DW target. `Patient_Identification` and `Bio_Values` are two responses extracted from the entire HL7 response. Before I moved to the loading section, I have created a lookup table for HL7 keys and assigned them to the correspondences field in conceptual DW schema by very similar name like as the ones we have in target schema.

In the end, by inserting transformed data into the target physical DW, we have the final result of the ETL process. I have chosen `PostgreSQL` to create the data warehousing system because It is a free and open-source relational database management system that emphasizes flexibility and SQL compliance. To query the final result, I used `pgAdmin4`, a general-purpose tool for developing, maintaining, and managing `PostgreSQL` databases.

Review of the thesis objectives

At the beginning of this document, there is a list of the main objectives of this thesis. Let us review if they have been accomplished:

1. Find and document the HL7 segments definition that can help us to understand the HL7 functionality better.

✓ In chapter 2, the structure and segments definition of HL7 messages is explained completely.

2. Model the data and design the schema for creating a physical data warehouse.

✓ In chapter 4.2, There is a full explanation of the methodology that we used to model the data, create the target schema, and convert it to the physical data warehouse.

3. Conceptually model the ETL process. The conceptual design should be modeled at two different levels: overview of whole process and the ETL process itself. Such representations would conceptually provide a full description of the data flow and control flow.

✓ In chapter 4, the conceptual design of all aspects of the project is explained in detail.

4. The ETL process implemented by python programming language that meets all nun-functional requirements defined in section 1.3.1.

✓ As I stated in section 5.3, we met all the nun-functional requirements defined in table 5.2. The complete code of the ETL tool in the python programming language can be accessed through the Git repository link below:

https://github.com/meysam24zamani/ETL_For_HL7

5. Build a wrapper script that would allow the user to run the ETL tool to request specific patient observation results.

✓ In section 5.2, the functional requirements for running the tool is explained. After running the tool, the final result would be ready that explained in chapter 6.

6. Validate the ETL tool with sample data from Genomcore company (the data are fake because of the sensitivity of the patient's test results).

✓ In section 6.1 and 6.2, the result of testing sample data from the Genomecore company is presented (You can find the sample data itself in Appendix C).

Chapter 8

Future Work

Finally, a list of tasks is proposed that would continue with the work done in this document, in terms of better automating the extraction part, improve the stability of the system and reduce the probability of encountering an error during the execution. As we have seen throughout the thesis project, my work was based mainly on creating an ETL tool for dealing with patients' data in HL7 files coming from healthcare centers. Below I will describe some of the major improvements that could enrich the user experience:

- First of all, the tool could be designed and executed with more HL7 messages with different purposes. The current sample data is based on the laboratory blood test but as you can see in Figure 2.1, several other forms of EHRs can be requested.

- The validation step could place all segments of the HL7 message. In this thesis due to limitations of the time, only intended those segments that have been used in most frequent HL7 messages, but it would be possible to add all segments, in order to add this potential for extracting whatever information we want from the source data.

- The ETL tool could be extended to allow further extension of the file during the extraction process. Currently, in this thesis project, I focused on the HL7 message represented in XML format file, but might be possible that a center expects to integrate their available patient data from CSV, JSON, or even other databases with available HL7 file.

- Implementation of a more comprehensive data security approach for the safety layer. Table 4.1 provides a list of other potential data security methods.

- The marshmallow package of python that we used for validating HL7 messages has a set of attributes that can tune the quality of each feature of the dataset. I used some of them in this thesis project, but it would possible to add more attributes to define more specific quality measures for available data.

- In this thesis, it was designed a demand-driven approach for data modeling. Another future-work would be to design it using a data-driven approach.

- The ETL tool proposed in this thesis were implemented in Python programming language. This code could be refined and added to a package in order to be available for the community.

- Also related to the implementation, it can be connected to an interface of an application implemented by HTML, CSS, JavaScript, or any other web programming languages. It may facilitates interaction between tool and consumers (data analysts).

These are some of the possible future-works of this thesis, but as previously stated, there is a lot of work that can be done in the research area of ETL methods for distributed data in the healthcare systems.

Bibliography

[1] Fabrizio Pecoraro, Daniela Luzi, and Fabrizio L. Ricci. Designing ETL Tools to Feed a Data Warehouse Based on Electronic Healthcare Record Infrastructure. *Studies in Health Technology and Informatics*, 210:929–936, 2015.

[2] Saadia Ismail, Majed Alshmari, Usman Qamar, Wasi Haider Butt, Khalid Latif, and Hafiz Farooq Ahmad. HL7 FHIR Compliant Data Access Model for Maternal Health Information System. (October), 2016.

[3] Athanasios Kiourtis. Structurally Mapping Healthcare Data to HL7 FHIR through Ontology Alignment. 2019.

[4] Houssein Dhayne and Rima Kilany. SeDIE : A Semantic-Driven Engine for Integration of Healthcare Data. (December), 2018.

[5] Raphael W Majeed. Automated Realtime data Import for the i2b2 Clinical data Warehouse : Introducing the HL7 ETL cell. (August), 2012.

[6] Vasileios Theodorou, Alberto Abelló, and Wolfgang Lehner. Quality measures for ETL processes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8646 LNCS:9–22, 2014.

[7] Clete A. Kushida, Deborah A. Nichols, Rik Jadrnicek, Ric Miller, James K. Walsh, and Kara Griffin. Health Record Data for Use in Multicenter Research Studies. *Med Care*, 50(July), 2012.

[8] Practice Fusion and Electronic Health Record. HL7 Lab Results API Developer Guide – 2.3. pages 1–42.

[9] Wang Xing, Tang Xiaoying, Yin Licheng, and Liu Weifeng. HL7 and the Transmission of Dynamic Signal in HL7 Standard. *2013 ICME International Conference on Complex Medical Engineering*, (Dmim):124–127, 2013.

[10] Yeb Havinga. Adding HL7 version 3 data types to PostgreSQL. (March 2010), 2014.

[11] Nilmini Wickramasinghe and Jonathan L. Schaffer. Creating knowledge-driven healthcare processes with the Intelligence Continuum, 2006.

[12] Hugh J. Watson and Barbara H. Wixom. The current state of business intelligence. *Computer*, 40(9):96–99, sep 2007.

[13] William H. Inmon. *Building the data warehouse*. Wiley Pub, 2005.

[14] Michael J Denney, Dustin M Long, Matthew G Armistead, Jamie L Anderson Rhitchts-im, and Baqiyyah N Conway. International Journal of Medical Informatics Validating the extract , transform , load process used to populate a large clinical research database. *International Journal of Medical Informatics*, 94:271–274, 2016.

[15] Toan C. Ong, Michael G. Kahn, Bethany M. Kwan, Traci Yamashita, Elias Brandt, Patrick Hosokawa, Chris Uhrich, and Lisa M. Schilling. Dynamic-ETL: A hybrid approach for health data extraction, transformation and loading. *BMC Medical Informatics and Decision Making*, 17(1), sep 2017.

[16] Ardhian Agung Yulianto. Extract transform load (ETL) process in distributed database academic data warehouse. 4(2):64–71, 2019.

[17] Maryam Jahanbakhsh, Reza Rabiei, Farkhondeh Asadi, and Hamid Moghaddasi. Electronic Health Record Architecture: a Systematic Review. *Journal of Paramedical Sciences*, 7(3):29–36, 2016.

[18] Joint Initiative Council, South America, and Functional Model. HEALTH LEVEL SEVEN ® INTERNATIONAL The Worldwide Leader in Interoperability Standards HEALTH LEVEL SEVEN ® INTERNATIONAL The Worldwide Leader in Interoperability Standards. 2011.

[19] Clement J. McDonald, J.Marc Overhage, William M. Tierney, and Dexter. The Regenstrief Medical Record System: a quarter century experience. *International Journal of Medical Informatics*, 54(3):225–253, jun 1999.

[20] K. Denecke T. Bürkle, M. Lehmann. *Healthcare of the Future: Bridging the Information Gap*. 2019.

[21] Paul V Biron, Kai U Heitmann, and Mike Henderson. v2.xml. pages 1–45, 2002.

[22] P A U L V B Iron, S Andra L E E B Oyer, and D Aniel E Ssin. The Practice of. pages 552–570.

[23] Duane Bender and Kamran Sartipi. HL7 FHIR: An agile and RESTful approach to healthcare information exchange. *Proceedings of CBMS 2013 - 26th IEEE International Symposium on Computer-Based Medical Systems*, (March):326–331, 2013.

[24] Christian Thomsen and Torben Bach Pedersen. Pygrametl: A powerful programming framework for extract-transform-load programmers. *International Conference on Information and Knowledge Management, Proceedings*, (November):49–56, 2009.

[25] Candy Pang and Duane Szafron. Single Source of Truth (SSOT) for Service Oriented Architecture (SOA). *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8831:575–589, 2014.

[26] Issn No and Abhijeet Raipurkar. Available Online at www.ijarcs.info Business Rules in DBMS. 3(3):693–696, 2012.

[27] Matteo Golfarelli, Dario Maio, and Stefano Rizzi. Conceptual design of data warehouses from E/R schemes. In *Proceedings of the Hawaii International Conference on System Sciences*, volume 7, pages 334–343. Institute of Electrical and Electronics Engineers Computer Society, 1998.

[28] Paul Westerman. *Data Warehousing: Using the Wal-Mart Model*. Morgan Kaufmann, 2001.

[29] Oscar Romero and Alberto Abelló. A framework for multidimensional design of data warehouses from ontologies. *Data and Knowledge Engineering*, 69(11):1138–1157, 2010.

[30] Chuck Ballard, Dirk Herreman, Don Schau, Rhonda Bell, Eunsaeng Kim, and Ann Valencic. Data modelling techniques for Data Warehousing. *Redbooks.Ibm.Com*, 1998.

[31] Rick Sherman. Dimensional Modeling. In *Business Intelligence Guidebook*, pages 197–235. Elsevier, jan 2015.

[32] John Mylopoulos Alexander Borgida. Conceptual Schema Design. In *Encyclopedia of Database Systems*, pages 438–442. Springer US, 2009.

[33] Zineb El Akkaoui and Esteban Zimányi. Defining ETL worfklows using BPMN and BPEL. *International Conference on Information and Knowledge Management, Proceedings*, pages 41–48, 2009.

Appendix A

Segments in the HL7 Message

This appendix section includes extensive information about the segments in the ORU_R01 message. The highest hierarchy level of the HL7 message is ORU_R01. The ORU message is for transmitting observational results, including lab, clinical or other observations, to other systems. Each item in the list below is a segment under the ORU hierarchy [8].

- Message Header (MSH)
- Patient Identification (PID)
- Patient Visit (PV1)
- Order Common (ORC)
- Observation Request (OBR)
- Observation Result (OBX)
- Note (NTE)

A.1 Message Header (MSH)

The MSH segment is the first segment for every HL7 message and identifies the source, purpose, destination, and certain syntax specifics such as the delimiters and character sets used in the message. The MSH segment is required and may appear only once. All fields in this segment can be found in Table A.1.

Table A.1: Description of fields in MSH segment - Source [8]

Seq	Name	Value
MSH-0	Segment identifier	MSH
MSH-1	Field delimiter	| (the "pipe" character)
MSH-2	Encoding characters	sample
MSH-3	Segment identifier	Identifies your application
MSH-4	Sending facility	Identifies the sending laboratory or reporting site
MSH-5	Receiving application	Identifies Practice Fusion as the destination
MSH-6	Receiving facility	Identifies the healthcare organization for which the message is intended
MSH-7	Message date and time	Identifies the date and time the message was created
MSH-8	Security	Used in some implementations for security features
MSH-9	Message type	ORU_R01

MSH-10	Message control ID	Contains the value the system uses to associate the message with the response to the message
MSH-11	Processing ID	P for "in production" D for "in debugging" T for "in training"
MSH-12	HL7 version	2.3
MSH-13	Sequence number	A non-null value in this field indicates that the sequence number protocol is in use
MSH-14	Continuation pointer	Contains the value used by a system to associate a continuation message with the message that preceded it.
MSH-15	Accept acknowledgement type	AL: Always require NE: Never require SU: successfully transmitted ER: Event of an error
MSH-16	Application acknowledgement type	AL: Always NE: Never SU: successfully transmitted ER: Event of an error
MSH-17	Country code	HL7 recommends values from ISO table 3166
MSH-18	Character set	Valid character set codes are defined in HL7 table 0211
MSH-19	Principle language of message	HL7 recommends values from ISO table 639

A.2 Patient Identification (PID)

The `PID` segment is used by all applications as the primary means of communicating information on patient identification. This segment contains patient identification and demographic information that is not likely to change frequently. The `PID` segment is required and may only appear once. All fields in this segment can be found in Table A.2.

Table A.2: Description of fields in `PID` segment - Source [8]

Seq	Name	Value
PID-0	Segment type ID	PID
PID-1	Sequence number	Identifies the number of the PID segment in circumstances where the message contains multiple patient reports
PID-2	External patient ID	Unique identifier for the patient; retained for backward compatibility
PID-3	Patient identifier list	Uniquely identifies the patient using values such as a medical record number, billing number, birth registry, and so forth

PID-4	Alternate patient ID	Contains alternate, pending, or temporary optional patient identifiers to be used, such as a social security number, a visit date, or a visit number; it has been retained for backward compatibility
PID-5	Patient name	Patient's first, last, and middle name
PID-6	Mother's maiden name	Maiden name of mother
PID-7	Patient date of birth	Patient date of birth
PID-8	Patient gender	Valid gender codes are defined in HL7 table 0001
PID-9	Patient alias	Patient alias
PID-10	Patient race	Valid race codes are defined in HL7 table 0005
PID-11	Patient address	Patient address
PID-12	Patient county code	Valid county codes are defined in HL7 table 0289
PID-13	Patient home phone number	Patient phone number

A.3 Patient Visit (PV1)

The `PV1` segment contains information regarding a particular patient visit. This segment can be used to send multiple-visit statistical records to the same patient account or single-visit records to more than one account. All fields in this segment can be found in Table A.3.

Table A.3: Description of fields in `PV1` segment - Source [8]

Seq	Name	Value
PV1-0	Segment identifier	PV1
PV1-1	Sequence number	1
PV1-2	Patient class	Valid patient classes are defined in HL7 table 0004
PV1-3	Assigned patient location	Identifies the patient's initial assigned location or the location to which the patient is being moved.
PV1-4	Admission type	Valid admission type codes are defined in HL7 table 0007
PV1-5	Pre-admit number	Identifies the patient's account prior to admission
PV1-6	Prior patient location	Identifies the prior location of the patient when being transferred.
PV1-7	Attending provider	Attending provider's name and unique identifier.

A.4 Order Common (ORC)

The optional `ORC` segment describes the basic details on the order for the sample to be tested. This segment includes identifiers of the order, who placed the order, when it was placed, what action to take regarding the order, and so forth. All `ORC` segment fields can be found in Table A.4.

Table A.4: Description of fields in ORC segment - Source [8]

Seq	Name	Value
ORC-0	Segment type	ORC
ORC-1	Order control	Specifies the code in HL7 table 0119 that identifies the action to be taken for the order
ORC-2	Placer order number	Identifies the application requesting the order
ORC-3	Filler order number	The order number of the application filling the order
ORC-4	Placer group number	Used by the application placing the order to group sets of orders together and identify them
ORC-5	Order status	Specifies the code in HL7 table 0038 that identifies the status of the order.
ORC-6	Response flag	Specifies the code in HL7 table 0121 that allows the placer application to determine the amount of information to be returned from the filler.

A.5 Observation Request (OBR)

The OBR segment is used to transmit information related to a medical test or evaluation, physical exam or assessment order. It determines the characteristics of a particular diagnostic service order. All fields in this segment can be found in Table A.5.

Table A.5: Description of fields in OBR segment - Source [8]

Seq	Name	Value
OBR-0	Segment type	OBR
OBR-1	Sequence number	These values should be a numeric sequence.
OBR-2	Placer order number	Identifies the application requesting the order
OBR-3	Filler order number	Contains a permanent identifier for an order and its observations.
OBR-4	Universal service ID	Specifies the code for the requested observation.
OBR-5	Priority	Specifies the priority of the request.
OBR-6	Requested date and time	Specifies the date and time of the request.
OBR-7	Observation date and time	Identifies the clinically-relevant date and time of the observation.
OBR-8	Observation end date and time	Identifies the end date and time of a study.
OBR-9	Collection volume	Specifies the collection volume of a specimen.
OBR-10	Collector identifier	Identifies the individual, department, or facility.
OBR-11	Action code	identifies the action to be taken.
OBR-12	Danger code	Contains the code, or text, or both that indicate any known or suspected patient.

OBR-13	Relevant clinical information	Contains additional clinical information about the patient or specimen.
OBR-14	Specimen received date and time	Identifies the date and time a diagnostic service receives the specimen.
OBR-15	Specimen source	Identifies the site where the specimen should be obtained.
OBR-16	Ordering provider	Identifies the individual that requested the order or prescription.
OBR-17	Order callback phone number	Identifies the phone number to call for clarification of a request.
OBR-18	Result reported date and time	Identifies the date and time when the results are entered in a report.
OBR-19	Charge to practice	Contains the charge to the ordering entity.
OBR-20	Diagnostic serv sect ID	identifies where the observation was performed.
OBR-21	Test status	identifies the status of results.

A.6 Observation Result (OBX)

The `OBX` segment is used for the transmission of a single observation or observation fragment. It is the smallest indivisible unit of an analysis and is intended to provide information on the observations in the report messages. All fields in this segment can be found in Table A.6.

Table A.6: Description of fields in `OBX` segment - Source [8]

Seq	Name	Value
OBX-0	Segment type	OBX
OBX-1	Sequence number	These values should be a numeric sequence.
OBX-2	Value type	Contains the format of the observation value.
OBX-3	Observation identifier	Contains a unique identifier for the observation.
OBX-4	Observation sub-id	Contains a unique identifier for each OBX segment.
OBX-5	Observation value	Contains the value observed by the producer.
OBX-6	Result units of measurement	Specifies the ISO value of the units for the measurement.
OBX-7	Result unit reference range	Specifies lower limits, upper limits, or both for result values.
OBX-8	Abnormal flags	Identifies the normalcy status of the result.
OBX-9	Probability	Identifies the probability of the result being true.
OBX-10	Nature of abnormal test	identify the nature of an abnormal test.
OBX-11	Observation result status	identifies the current completion status of the observation result.
OBX-12	Effective date of last normal observation	Contains changes in the observation methods.

OBX-13	User-defined access checks	Permits the producer to record results.
OBX-14	Observation date and time	Identifies the physiologically-relevant date and time of the report.
OBX-15	Producer's id	Contains the unique identifier of the responsible producing service.
OBX-16	Responsible observer	Contains the unique identifier of the individual responsible for performing or verifying the observation.
OBX-17	Observation method	Identifies the method or procedure by which an observation was obtained.

A.7 Note (NTE)

The `NTE` segment includes notes and comments and can be added to the `PID`, `ORC`, `OBR`, and `OBX` segments. All fields in this segment can be found in Table A.7.

Table A.7: Description of fields in `NTE` segment - Source [8]

Seq	Name	Value
NTE-0	Segment type	NTE
NTE-1	Sequence number	These values should be a numeric sequence.
NTE-2	Comment source	Identifies the source of the comment.
NTE-3	Comment	Contains the comments entered by the source.

Appendix B

Output in JSON Format

B.1 Patient_Identification.JSON


```
{
    "name": "Meysam",
    "surname1": "Zamani",
    "surname2": "Forooshani",
    "sex": "M",
    "dni": "Y5694768M",
    "dateOfBirth": "19890315",
    "age": 31,
    "range_age": "young"
}
```


B.2 Bio_Value.JSON


```
{
    "section_bio_values": {
        "title": "Patient bio values",
        "sample_date": "20191122095719",
        "sample_timestamp": "2019-11-22 09:57:19",
        "value": [
            {
                "title": "Hemoglobina",
                "code": "HB",
                "group": "Hematología y Hemostasia",
                "min": 12.5,
                "max": 17.2,
                "units": "g/dL",
                "value": 15.5,
                "relative_discrepancy": 0,
                "consequence": "N"
            },
            {
                "title": "Hematocrito",
                "code": "HCTO",
                "group": "Hematología y Hemostasia",
                "min": 36.5,
                "max": 50.5,
                "units": "%",
                "value": 44.3,
                "relative_discrepancy": 0,
```

```
                "consequence": "N"
            },
            {
                "title": "VCM",
                "code": "VCM",
                "group": "Hematología y Hemostasia",
                "min": 78.0,
                "max": 99.0,
                "units": "fL",
                "value": 92.0,
                "relative_discrepancy": 0,
                "consequence": "N"
            },
            {
                "title": "HCM",
                "code": "HCM",
                "group": "Hematología y Hemostasia",
                "min": 26.0,
                "max": 33.5,
                "units": "pg",
                "value": 32.1,
                "relative_discrepancy": 0,
                "consequence": "N"
            }
            .
            .
            .
            .
            .
            .
            {
                "title": "Insulina",
                "code": "INSU",
                "group": "Endocrinología (suero)",
                "min": 2.0,
                "max": 25.0,
                "units": "mU/L",
                "value": 5.74,
                "relative_discrepancy": 0,
                "consequence": "N"
            },
            {
                "title": "Proteína C Reactiva en suero (Ultrasensible)",
                "code": "PCU",
                "group": "Inmunología",
                "min": 0.0,
                "max": 3.0,
                "units": "mg/L",
                "value": 0.5,
                "relative_discrepancy": 0,
                "consequence": "N"
            },
        ]
    }
}
```

Appendix C

Sample Data

The sample data used in this thesis project is presented in this section. It is an XML file that includes an HL7 message within the file. This is the sample data that we received from the Genomcore company. Patient identity is fake, due to the privacy issues related to the actual health data sources.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <ObtenerResultadosMuestraResponse xmlns="http://lumensoft.net/">
   <ObtenerResultadosMuestraResult><ORU_R01 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="urn:hl7-org:v2xml">
  <MSH>
   <MSH.1>|</MSH.1>
   <MSH.2>^~\&amp;</MSH.2>
   <MSH.3>
    <HD.1>ETL_FOR_HL7</HD.1>
   </MSH.3>
   <MSH.5>
    <HD.1>MadeGenes.Ws</HD.1>
   </MSH.5>
   <MSH.7>
    <TS.1>20191126163122</TS.1>
   </MSH.7>
   <MSH.9>
    <MSG.1>ORU</MSG.1>
    <MSG.2>R01</MSG.2>
   </MSH.9>
   <MSH.10>20191126163122262</MSH.10>
   <MSH.11>
    <PT.1>P</PT.1>
   </MSH.11>
   <MSH.12>
    <VID.1>2.5</VID.1>
   </MSH.12>
  </MSH>
  <ORU_R01.PATIENT_RESULT>
   <ORU_R01.PATIENT>
    <PID>
     <PID.2>
      <CX.1></CX.1>
```

```
        </PID.2>
        <PID.5>
          <XPN.1>
            <FN.1>Zamani</FN.1>
          </XPN.1>
          <XPN.2>Forooshani</XPN.2>
          <XPN.3>Meysam</XPN.3>
        </PID.5>
        <PID.7>
          <TS.1>19890315</TS.1>
        </PID.7>
        <PID.8>M</PID.8>
      </PID>
      <ORU_R01.VISIT>
        <PV1>
          <PV1.3>
            <PL.1>PVV</PL.1>
            <PL.3>LAB URGELL-BARCELONA (PVV)</PL.3>
          </PV1.3>
          <PV1.4 />
          <PV1.17>
            <XCN.1 />
            <XCN.2>
              <FN.1 />
            </XCN.2>
            <XCN.3 />
            <XCN.4 />
          </PV1.17>
          <PV1.50>
            <CX.1 />
          </PV1.50>
          <PV1.52>
            <XCN.1>SC</XCN.1>
            <XCN.3>SC-ACT A TRASPASAR/SIN CARGO</XCN.3>
          </PV1.52>
        </PV1>
      </ORU_R01.VISIT>
    </ORU_R01.PATIENT>
    <ORU_R01.ORDER_OBSERVATION>
      <ORC>
        <ORC.1>XO</ORC.1>
        <ORC.2>
          <EI.1 />
        </ORC.2>
        <ORC.3>
          <EI.1>Y5694768M</EI.1>
        </ORC.3>
        <ORC.5>A</ORC.5>
        <ORC.9>
          <TS.1>20191122095719</TS.1>
        </ORC.9>
      </ORC>
      <OBR>
        <OBR.1>2</OBR.1>
        <OBR.2>
          <EI.1>Y5694768M</EI.1>
        </OBR.2>
        <OBR.4>
```

```
          <CE.1>LIFE</CE.1>
          <CE.2>PERFIL LIFE</CE.2>
        </OBR.4>
      </OBR>
      <FT1>
        <FT.11>
          <CP.1>
            <MO.1 />
          </CP.1>
        </FT.11>
      </FT1>
      <ORU_R01.OBSERVATION>
        <OBX>
          <OBX.1>2</OBX.1>
          <OBX.2>NM</OBX.2>
          <OBX.3>
            <CE.1>HB</CE.1>
            <CE.2>Hemoglobina</CE.2>
            <CE.3>Hematolog a y Hemostasia</CE.3>
          </OBX.3>
          <OBX.5>15,5</OBX.5>
          <OBX.6>
            <CE.2>g/dL</CE.2>
          </OBX.6>
          <OBX.7>12,50 - 17,20</OBX.7>
          <OBX.8>N</OBX.8>
          <OBX.14>
            <TS.1>20191122</TS.1>
          </OBX.14>
          <OBX.16>
            <XCN.1></XCN.1>
          </OBX.16>
        </OBX>
        <NTE>
          <NTE.3 />
        </NTE>
      </ORU_R01.OBSERVATION>
    </ORU_R01.ORDER_OBSERVATION>
    <ORU_R01.ORDER_OBSERVATION>
      <ORC>
        <ORC.1>XO</ORC.1>
        <ORC.2>
          <EI.1 />
        </ORC.2>
        <ORC.3>
          <EI.1>Y5694768M</EI.1>
        </ORC.3>
        <ORC.5>A</ORC.5>
        <ORC.9>
          <TS.1>20191122095719</TS.1>
        </ORC.9>
      </ORC>
      <OBR>
        <OBR.1>3</OBR.1>
        <OBR.2>
          <EI.1>Y5694768M</EI.1>
        </OBR.2>
        <OBR.4>
```

```
          <CE.1>LIFE</CE.1>
          <CE.2>PERFIL LIFE</CE.2>
        </OBR.4>
      </OBR>
      <FT1>
        <FT.11>
          <CP.1>
            <MO.1 />
          </CP.1>
        </FT.11>
      </FT1>
      <ORU_R01.OBSERVATION>
        <OBX>
          <OBX.1>3</OBX.1>
          <OBX.2>NM</OBX.2>
          <OBX.3>
            <CE.1>HCTO</CE.1>
            <CE.2>Hematocrito</CE.2>
            <CE.3>Hematolog a y Hemostasia</CE.3>
          </OBX.3>
          <OBX.5>44,3</OBX.5>
          <OBX.6>
            <CE.2>%</CE.2>
          </OBX.6>
          <OBX.7>36,50 - 50,50</OBX.7>
          <OBX.8>N</OBX.8>
          <OBX.14>
            <TS.1>20191122</TS.1>
          </OBX.14>
          <OBX.16>
            <XCN.1></XCN.1>
          </OBX.16>
        </OBX>
        <NTE>
          <NTE.3 />
        </NTE>
      </ORU_R01.OBSERVATION>
    </ORU_R01.ORDER_OBSERVATION>
    <ORU_R01.ORDER_OBSERVATION>
      <ORC>
        <ORC.1>XO</ORC.1>
        <ORC.2>
          <EI.1 />
        </ORC.2>
        <ORC.3>
          <EI.1>Y5694768M</EI.1>
        </ORC.3>
        <ORC.5>A</ORC.5>
        <ORC.9>
          <TS.1>20191122095719</TS.1>
        </ORC.9>
      </ORC>
      <OBR>
        <OBR.1>4</OBR.1>
        <OBR.2>
          <EI.1>Y5694768M</EI.1>
        </OBR.2>
        <OBR.4>
```

```
          <CE.1>LIFE</CE.1>
          <CE.2>PERFIL LIFE</CE.2>
        </OBR.4>
      </OBR>
      <FT1>
        <FT.11>
          <CP.1>
            <MO.1 />
          </CP.1>
        </FT.11>
      </FT1>
      <ORU_R01.OBSERVATION>
        <OBX>
          <OBX.1>4</OBX.1>
          <OBX.2>NM</OBX.2>
          <OBX.3>
            <CE.1>VCM</CE.1>
            <CE.2>VCM</CE.2>
            <CE.3>Hematolog a y Hemostasia</CE.3>
          </OBX.3>
          <OBX.5>92</OBX.5>
          <OBX.6>
            <CE.2>fL</CE.2>
          </OBX.6>
          <OBX.7>78,00 - 99,00</OBX.7>
          <OBX.8>N</OBX.8>
          <OBX.14>
            <TS.1>20191122</TS.1>
          </OBX.14>
          <OBX.16>
            <XCN.1></XCN.1>
          </OBX.16>
        </OBX>
        <NTE>
          <NTE.3 />
        </NTE>
      </ORU_R01.OBSERVATION>
    </ORU_R01.ORDER_OBSERVATION>
    <ORU_R01.ORDER_OBSERVATION>
      <ORC>
        <ORC.1>XO</ORC.1>
        <ORC.2>
          <EI.1 />
        </ORC.2>
        <ORC.3>
          <EI.1>Y5694768M</EI.1>
        </ORC.3>
        <ORC.5>A</ORC.5>
        <ORC.9>
          <TS.1>20191122095719</TS.1>
        </ORC.9>
      </ORC>
      <OBR>
        <OBR.1>5</OBR.1>
        <OBR.2>
          <EI.1>Y5694768M</EI.1>
        </OBR.2>
        <OBR.4>
```

```
          <CE.1>LIFE</CE.1>
          <CE.2>PERFIL LIFE</CE.2>
        </OBR.4>
      </OBR>
      <FT1>
        <FT.11>
          <CP.1>
            <MO.1 />
          </CP.1>
        </FT.11>
      </FT1>
      <ORU_R01.OBSERVATION>
        <OBX>
          <OBX.1>5</OBX.1>
          <OBX.2>NM</OBX.2>
          <OBX.3>
            <CE.1>HCM</CE.1>
            <CE.2>HCM</CE.2>
            <CE.3>Hematolog a y Hemostasia</CE.3>
          </OBX.3>
          <OBX.5>32,1</OBX.5>
          <OBX.6>
            <CE.2>pg</CE.2>
          </OBX.6>
          <OBX.7>26,00 - 33,50</OBX.7>
          <OBX.8>N</OBX.8>
          <OBX.14>
            <TS.1>20191122</TS.1>
          </OBX.14>
          <OBX.16>
            <XCN.1></XCN.1>
          </OBX.16>
        </OBX>
        <NTE>
          <NTE.3 />
        </NTE>
      </ORU_R01.OBSERVATION>
    </ORU_R01.ORDER_OBSERVATION>
    .
    .
    .
    .
    .
    .
    <ORU_R01.ORDER_OBSERVATION>
      <ORC>
        <ORC.1>XO</ORC.1>
        <ORC.2>
          <EI.1 />
        </ORC.2>
        <ORC.3>
          <EI.1>Y5694768M</EI.1>
        </ORC.3>
        <ORC.5>A</ORC.5>
        <ORC.9>
          <TS.1>20191122095719</TS.1>
        </ORC.9>
      </ORC>
```

```
      <OBR>
        <OBR.1>36</OBR.1>
        <OBR.2>
          <EI.1>Y5694768M</EI.1>
        </OBR.2>
        <OBR.4>
          <CE.1>LIFE</CE.1>
          <CE.2>PERFIL LIFE</CE.2>
        </OBR.4>
      </OBR>
      <FT1>
        <FT.11>
          <CP.1>
            <MO.1 />
          </CP.1>
        </FT.11>
      </FT1>
      <ORU_R01.OBSERVATION>
        <OBX>
          <OBX.1>36</OBX.1>
          <OBX.2>NM</OBX.2>
          <OBX.3>
            <CE.1>INSU</CE.1>
            <CE.2>Insulina</CE.2>
            <CE.3>Endocrinolog a (suero)</CE.3>
          </OBX.3>
          <OBX.5>5,74</OBX.5>
          <OBX.6>
            <CE.2>mU/L</CE.2>
          </OBX.6>
          <OBX.7>2,00 - 25,00</OBX.7>
          <OBX.8>N</OBX.8>
          <OBX.14>
            <TS.1>20191122</TS.1>
          </OBX.14>
          <OBX.16>
            <XCN.1></XCN.1>
          </OBX.16>
        </OBX>
        <NTE>
          <NTE.3 />
        </NTE>
      </ORU_R01.OBSERVATION>
    </ORU_R01.ORDER_OBSERVATION>
    <ORU_R01.ORDER_OBSERVATION>
      <ORC>
        <ORC.1>XO</ORC.1>
        <ORC.2>
          <EI.1 />
        </ORC.2>
        <ORC.3>
          <EI.1>Y5694768M</EI.1>
        </ORC.3>
        <ORC.5>A</ORC.5>
        <ORC.9>
          <TS.1>20191122095719</TS.1>
        </ORC.9>
      </ORC>
```

```
      <OBR>
        <OBR.1>37</OBR.1>
        <OBR.2>
          <EI.1>Y5694768M</EI.1>
        </OBR.2>
        <OBR.4>
          <CE.1>LIFE</CE.1>
          <CE.2>PERFIL LIFE</CE.2>
        </OBR.4>
      </OBR>
      <FT1>
        <FT.11>
          <CP.1>
            <MO.1 />
          </CP.1>
        </FT.11>
      </FT1>
      <ORU_R01.OBSERVATION>
        <OBX>
          <OBX.1>37</OBX.1>
          <OBX.2>NM</OBX.2>
          <OBX.3>
            <CE.1>PCU</CE.1>
            <CE.2>Prote na C Reactiva en suero (Ultrasensible)
  </CE.2>
            <CE.3>Inmunolog a</CE.3>
          </OBX.3>
          <OBX.5>0,50</OBX.5>
          <OBX.6>
            <CE.2>mg/L</CE.2>
          </OBX.6>
          <OBX.7>0,00 - 3,00</OBX.7>
          <OBX.8>N</OBX.8>
          <OBX.14>
            <TS.1>20191126</TS.1>
          </OBX.14>
          <OBX.16>
            <XCN.1></XCN.1>
          </OBX.16>
        </OBX>
        <NTE>
          <NTE.3 />
        </NTE>
      </ORU_R01.OBSERVATION>
    </ORU_R01.ORDER_OBSERVATION>
    <ORU_R01.ORDER_OBSERVATION>
      <ORC>
        <ORC.1>XO</ORC.1>
        <ORC.2>
          <EI.1 />
        </ORC.2>
        <ORC.3>
          <EI.1>Y5694768M</EI.1>
        </ORC.3>
        <ORC.5>A</ORC.5>
        <ORC.9>
          <TS.1>20191122095719</TS.1>
        </ORC.9>
```

```
      </ORC>
      <OBR>
        <OBR.1>38</OBR.1>
        <OBR.2>
          <EI.1>Y5694768M</EI.1>
        </OBR.2>
        <OBR.4>
          <CE.1>LIFE</CE.1>
          <CE.2>PERFIL LIFE</CE.2>
        </OBR.4>
      </OBR>
      <FT1>
        <FT.11>
          <CP.1>
            <MO.1 />
          </CP.1>
        </FT.11>
      </FT1>
      <ORU_R01.OBSERVATION>
        <OBX>
          <OBX.1>38</OBX.1>
          <OBX.2>NM</OBX.2>
          <OBX.3>
            <CE.1>ZN</CE.1>
            <CE.2>Zinc en suero</CE.2>
            <CE.3>Monitorizaci n de F rmacos y Toxicolog a</
   CE.3>
          </OBX.3>
          <OBX.5>79</OBX.5>
          <OBX.6>
            <CE.2> g /dL</CE.2>
          </OBX.6>
          <OBX.7>60,00 - 120,00</OBX.7>
          <OBX.8>N</OBX.8>
          <OBX.14>
            <TS.1>20191126</TS.1>
          </OBX.14>
          <OBX.16>
            <XCN.1></XCN.1>
          </OBX.16>
        </OBX>
        <NTE>
          <NTE.3 />
        </NTE>
      </ORU_R01.OBSERVATION>
    </ORU_R01.ORDER_OBSERVATION>
  </ORU_R01.PATIENT_RESULT>
</ORU_R01></ObtenerResultadosMuestraResult>
        </ObtenerResultadosMuestraResponse>
    </soap:Body>
</soap:Envelope>
```