FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Monitoring Framework for Clinical ETL processes and associated performance resources

Margarida Abranches

DISSERTATION

Orientador da FEUP: João Correia Lopes

Co-orientador da B2F: Francisco Capa

October 31, 2020

Resumo

A arquitetura do sistema *Business Intelligence*, implementado no âmbito desta dissertação, está organizada em dois ambientes: *back-end* (ambiente de desenvolvimento) e *front-end* (ambiente de visualização). No *back-end*, o processo BI começa com a recolha de dados clínicos anonimizados, em formato de ficheiros CSV, a partir da versão demo MIMIC-III v.1.4 e subsequente extração, transformação e carregamento (ETL) dos dados para as dimensões e factos da *Data Warehouse* (DW) Clínical.

O processo ETL clínico foi implementado, on-premises, utilizando os módulos dos Serviços de Integração da Microsoft SQL Server (SS) e, no ambiente Microsoft Azure, foram utilizados os recursos *Azure Blob Storage* e da *Data Factory*.

Os sistemas de Business intelligence estão sempre associados a tecnlogias de visualização que criam *dashboards* interativos e de fácil utilização para analisar métricas e indicadores chave de desempenho. A Solução Visual Clínica foi desenvolvida no *Power BI Desktop* (PBID) e está dividida em cinco *Data Marts*: serviços de admissão hospitalar, medições da ficha eletrónica médica, intervenções médicas e testes laboratoriais.

As soluções clínicas proporcionam decisões rápidas e eficazes para profissionais de saúde e administradores, poupando tempo e recursos valiosos. Na solução desenvolvida é possível avaliar a carga de trabalho dos profissionais médicos, o número de intervenções médicas, medições da ficha eletrónica médica, e testes laboratoriais realizados por categoria médica, data (mês, trimestre e ano), profissional de saúde, e Unidade de Cuidados Intensivos (UCI). Também é possível analisar o tipo de admissão, a taxa de sobrevivência da admissão hospitalar, o tempo médio de espera da assistência médica do paciente (tempo entre a admissão e o primeiro teste realizado) e o tempo de hospitalização.

O DW clínico integra múltiplas fontes de dados heterogéneas no setor da saúde, fornecendo uma plataforma de informação otimizada e eficaz para os decisores de saúde. Assim, emerge a necessidade de construir uma plataforma centralizada para monitorizar todos os processos de ETL clínico, implementados no servidor on-premises ou na cloud.

A plataforma foi implementada no Ambiente Microsoft Azure para monitorizar mensagens de erro do ETL e as métricas de desempenho de recursos (por exemplo, CPU, RAM) de diferentes Soluções Clínicas. As informações de monitorização (erros do ETL e métricas de desempenho dos recursos) foram enviadas por e-mail, utilizando o serviço SendGrid Web API. A *Azure Logic Application* interpreta o corpo e assunto do e-mail para armazenar a informação de monitorização nas respetivas tabelas da base de dados do Azure. Finalmente, o utilizador final pode analisar o *dashboard* de monitorização e ser alertado se um ou mais projetos falharem ou estiverem perto de falhar.

Palavras-chave: Solução Clínica; ETL; *Data Warehouse*; *Business Intelligence*; *Framework* de monitorização;

Abstract

The dissertation's Business Intelligence Architecture is organized in two environments: back-end (development environment) and front-end (visualization environment). In the back-end, the process begins with the collection of clinical de-identified data, in CSV files format, from the MIMIC-III demo v.1.4 and subsequent extraction, transformation, and loading (ETL) of the data to the dimensions and facts of the Clinical Data Warehouse (DW).

The Clinical ETL process was implemented, on-premises, using Integration Services modules from the Microsoft SQL Server (SS) and, in the Microsoft Azure environment, using the Azure Blob Storage and Data Factory resources.

The Business intelligence systems are always associated with Reporting Technologies that create unified, user-friendly dashboards to analyze metrics and key performance indicators. The Clinical Visual Solution was developed in the Power BI Desktop (PBID) and is divided into five data marts: hospital admission services, electronic charted measurements, medical interventions, microbiology, and laboratory tests.

Clinical solutions provide quick and effective business decisions for medical and administrative professionals while saving valuable time and resources. In the developed solution, it is possible to evaluate the medical professionals' workload, the number of medical interventions, electronic charted measurements, microbiology and laboratory tests performed by medical category, date (month, trimester, and year), caregiver, and Intensive Care Unit (ICU). It is also possible to analyze the type of admission, the hospital admission's survival rate, and the patient's average medical assistance waiting (time between the admission and the first performed test) and hospitalization time.

The clinical DW integrates multiple heterogeneous data sources in the healthcare sector, providing an optimized and effective information platform for health decision-makers. Thus, emerging the necessity to built a central framework to monitor all the Clinical ETL processes, implemented on-premises or in the cloud.

The central framework was implemented in the Microsoft Azure Environment to monitor the ETL message errors and resources' performance metrics (e.g., CPU, RAM) of different Clinical Solutions. The monitoring information (ETL errors and the resources' performance metrics) were sent via e-mail, using the SendGrid Web API service. The Azure Logic Application interprets the e-mail body and subject to retrieve the monitoring information and store it into the respective Azure SQL database tables. Finally, the end-user can analyze the monitoring dashboard and be alerted if one or more projects fail or are close to failure.

Keywords: Clinical Solution; ETL; Data Warehouse; Business Intelligence; Monitoring Framework;

Agradecimentos

Em primeiro lugar queria agradecer ao meu orientador João Correia Lopes pela orientação prestada e por toda a disponibilidade e paciência durante este moroso percurso.

À B2F que me possibilitou a realização desta dissertação e que teve sempre disponível para me ajudar a ultrapassar os obstáculos encontrados pelo caminho. Queria agradecer, especialmente, ao meu co-orientador Francisco Capa pela amibilidade e disponibilidade ilimitada que demonstrou comigo.

À minha família pelo apoio emocional dado nestes últimos meses. Queria deixar um enorme obrigada à minha irmã por todas as mensagens de apoio, *playlists* e vídeos motivadores partilhados às 3 da manhã.

Aos reVOLTados pelas noitadas de partilha de *gossip* e pelas mensagens inspiradoras diárias. Um obrigado especial à Mariana e ao Rui que almoçaram comigo todos os dias para partilhar as angústias mais profundas.

À Simões que me fez companhia na escrita desta epopeia académica.

À Sara, Teresa, Cátia Vanessa e Rafaela por todas as conversas e pausas de café que ajudaram a manter a minha sanidade mental.

E, finalmente, um enorme obrigada à colega de casa que me viu passar todas as horríficas fases numa época de pandemia e que nunca me deixou ir a baixo por muito que eu tentasse. À minha querida amiga e psicóloga Mónica Filipa.

Margarida Abranches

Contents

List of Figures

List of Tables

Abreviaturas e Símbolos

API	Application Programming Interface
AWS	Amazon Web Services
BI	Business Intelligence
CIF	Corporate Information Factory
CPU	Central Processing Unit
CSV	Comma-Separated Value
DBMS	Database Management System
DE	Database Engine
DM	Data Mart
DSA	Destination Staging Area
DOD	Date of Death
DW	Data Warehouse
ETL	Extract, Transform and Load
FK	Foreign Key
GCP	Google Cloud Platform
HIPAA	Health Insurance Portability and Accountability Act
HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
ICU	Intensive Care Unit
IT	Information Technology
MIMIC	Medical Information Mart for Intensive Care
NoSQL	Non Relational Structured Query Language
OLAP	Online Analytical Processing
PaaS	Platforms as a Service
PBID	Power BI Desktop
PK	Primary Key
RAM	Random-access Memory
SaaS	Software as a Service
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SS	SQL Server
SSIS	SQL Server Integration Services
SSMS	SQL Server Management Studio
vCPU	Virtual Central Processing Unit

Chapter 1

Introduction

This document presents the dissertation of the Bioengineering Integrated Master carried out in a business environment at BUSINESSTOFUTURE (B2F).

In this introductory chapter, it will be explained in what context the project was developed, the description of the problem and its objectives.

1.1 Motivation

The Health industry generates and collects a huge amount of data every day (images, diagnosis, medical records, among others) from several isolated and dispersed information repositories [1].

Doctors assess the user's condition and decide on the respective treatment. Medical directors assess the clinical purposes, quality and cost of the health service provided. Administrators manage human resources and inventories. Executives make investment decisions in new lines of business, partnerships with other organizations and the removal of underutilized services. Collectively, these professionals need timely, accurate and relevant information in their decision-making process [2].

However, the accumulation of health data is exceeding the capacity of organizations to take advantage of data to improve the decision-making process of these professionals [3]. Thus, Business Intelligence (BI) systems are considered as an answer to these needs, allowing the management, transformation and integration of data and consequently improve in the different areas of performance of the organization.

Furthermore, databases, data extraction and treatment processes in health units have a fundamental role in the quality of services provided since the data needs to be permanently available and the information flow working correctly.

In this context, there is a need to prevent failures in ETL (Extract, Load, and Transform) processes implemented in hospital units, which consists in monitoring the ETL state (success or failure) and the performance measurements that can lead to systems deficiencies.

Thus, the aim of this work is to develop a centralized framework, in the Microsoft Azure environment, compatible with cloud and on-premises technologies, to actively monitor the Clinical Solutions and act proactively in case of error or malfunction detection.

Combined with all the steps necessary for implementing the monitoring framework, it will also be essential to develop visual and intuitive reports using the B2F's current visualization tools. These dashboards allow the healthcare administrators to provide a proactive and non-reactive monitoring towards the several implemented Clinical Solutions.

1.2 Objectives

To develop a centralized framework to be implemented in the health industry, it is necessary to proceed with the following three phases.

The first step is to develop an Extract, Transform, and Load process, using both on-premises and cloud environments, to promote effective clinical decisions by organizing and structuring clinical data that is usually ambiguous, incomplete and inconclusive. In this project, the data source is the MIMIC-III (Medical Information Mart for Intensive Care - III) Demo v.1.4. which comprises a range of data of 100 patients hospitalized in the emergency units of Beth Israel Deaconess medical center between 2001 and 2012. This freely-available online clinical database covers from demographic patients information to medical procedures and laboratory and microbiological test results.

The second phase consists of the clinical dashboard design, allowing hospital unit managers to have more efficient support during the analysis and decision processes. In this dashboard, the manager must view the following indicators interactively and clearly: emergency events, the volume of medical interventions, laboratory and microbiology tests organized by medical service, department category and date (month, trimester, and year).

Finally, associated with the development of monitoring processes and the use of cloud tools, it will be necessary to design and implement a standardized structure applied in the different databases of the existing health units. In addition to the mechanisms already mentioned, it will be essential to implement a model of logs together with a report developed in a visual tool so that it is possible to consult it quickly and effectively daily.

The proposed framework architecture and model should respond to the following points:

- Monitoring and alert of Extract, Transform and Load processes;
- Monitoring of server resources such as CPU, RAM, etc;
- Monitoring the use level of dashboards and reports;
- Monitoring the degree of reliability of the information; To ensure that the metrics with temporal context are with the correct value (for example, the number of clinical exams may not be well calculated because one of the data sources was down and it was not possible to extract and compile this information)

1.3 Document Structure

This document is divided into seven chapters.

The first one is the present introduction followed by the State of the Art. Chapter 2 is an overview of health systems monitoring published work and contains information about Business Intelligence (BI), Cloud Computing, and Reporting Tools.

The third chapter describes, in detail, the freely-available online clinical database, MIMIC-III Demo v.1.4., and the required adaptations to create the Clinical database and the associated relational model.

The Business Intelligence Architecture construction process to organize and structure the Clinical database both on-premises and cloud environments, is presented in Chapter 4.

Chapter 5 present the analysis and discussion of the clinical dashboard reports results.

Chapter 6 describes the necessary resources and steps to implement the proposed monitoring framework and its architecture and dashboard.

The conclusions and future work can be found in Chapter 7.

Chapter 2

Literature Review

This chapter presents the current state of research and development of the relevant topics for carrying out the proposed work, namely an overview of health systems monitoring published work and the technologies associated with Business Intelligence (BI), Cloud Computing, and Reporting Tools.

2.1 Health Information Systems

Computer technologies' constant evolution has led to the massive computerization of services in organizations in the most diverse areas such as education, commerce, and health. Simultaneously, the importance of information for organizations has increased considerably, and nowadays, the success of an organization is directly connected to how it manages its information.

In healthcare environments, information systems are responsible for optimizing information that exists in a healthcare unit. They proceed to collecting, processing, and data management of all stakeholders (patients, nurses, doctors) and all healthcare unit services [1].

Thus, they support health care working professionals' activities and must provide safe and consistent access, confidentiality, efficiency, and continuous availability at high performance. In particular, they should:

- Improve the quality of treatment by providing automatic alerts and checks on the consistency of information;
- Increase efficiency using better standards of clinical practice and registration;
- Reduce long-term costs in information management and carrying out complementary means of diagnosis, eliminating the need for redundant exams.

2.1.1 SONHO

SONHO system exists in 90% of SNS (National Health System) institutions in Portugal.

The main features include patient documentation, hospital administrative, and financial management using CID's (International Classification of Diseases) relationship codes with GDH (Homogeneous diagnostic groups). It is designated as an ADT system (Admission, Discharge, Transfer) that helps Portugal's hospital administrative work. It is the CORE system, inside the hospital, for other health systems because it has information about patients, such as user number, name, age, contact, clinical history, eliminating the possibility of duplicating patients in the database [4].

The database system is Oracle, version 7.3.4 (currently in version 12) with PL/SQL language (Oracle's procedural language extension to SQL). The database is manipulated using a connection, DBLink (database link), between two physical database servers that allow a client to access them as one logical database.

SONHO is divided into the following eight modules: patient registration, emergency, hospital internment, operating room, and external appointment calendar management, hospital admission, billing, and patient file archive [4].

SONHO's technology caused problems at application and infrastructure levels, such as authentication security, performance, and availability rate (the system fails several times a day in individual hospitals), among others.

Therefore, the system was gradually replaced by SONHO V2 and aims to respond to hospital units' technical problems resulting from SONHO's obsolescence. In order to guarantee scalability and the ability to evolve to new functionalities, SONHO V2 includes [4]:

- The technological migration to Oracle Forms and Reports Services 11g R2, which ensures greater alignment with most current applications and allows to build and design more complex dashboards that interact with an Oracle database;

- Development of a new service-oriented integration layer, the Local Interoperability Gateway For Healthcare (LIGHT);

- Provision of a Reporting Database, from which metrics, data and management reports can be obtained, removing cargo from the production database;

- Online Help tool that provides the user with support for navigation in the system, which can be updated and printed;

- Upload/Download of documentation, avoiding the paper file in the management of administrative documentation;

- Management Fees, which allows a global view of the information referring to the payment of user fees;

- Use of the citizen card, allowing fast and efficient identification of the user;

- Visualization of Daily and Statistical Maps;

- Access and schedule of several tasks simultaneously (appointments and medical tests).

2.1.2 LIGHT

One of the significant improvements with introducing the SONHO V2 version, in 2013, was adopting an interoperability platform, LIGHT, which consists of an integration layer, middleware, that mediates the exchange of information between the Health Ministry Shared Services (SPMS) products and external customers. Thus, SPMS intends to unify local systems at a national level so that health institutions communicate the same language and in a standardized way that will facilitate future international integrations [5].

This platform allows gradually to discontinue the SONHO V1's DBLink connections since they were developed individually for each institution and application (the same application often had different integration schemes in different institutions).

2.1.3 SClínico

The SClínico hospital information system is born from the fusion of the previous applications developed by SPMS to provide care to users, the Physician Support System (SAM), and the Nursing Practice Support System (SAPE), resulting in a single shared application to all health care providers, user-centered [6].

SClínico appeared as a strategy defined by the Ministry of Health to achieve uniformity of clinical records, increasing efficiency of health professionals and, consequently, saving resources and improving quality of care.

Unfortunately, although SClínico is used daily in most primary care health units, it is not yet used in all hospitals across the country. Some of the hospital records appointments are made at ALERT (system that forwards, automatically, a request for a new medical specialty or hospital). The lack of a return response when using ALERT is one of the most significant platform deficiencies because the family doctor, responsible for the request, does not receive clarification, and the information gets lost in the process [6].

Since 2017, only Centro Hospitalar de Leiria and Centro Hospitalar Distrital de Santarém have gradually implemented the SClínico/SONHO V2/LIGHT systems, and this platform still co-exists with a lot of other heterogeneous information systems [5].

2.1.4 Health Systems Monitoring

The health industry's database management systems consume large amounts of storage resources, processing, communication, and can not be subject to failures.

Failure monitoring consists of evaluating the system's behavior data and obtaining useful reports to avoid future failure circumstances. Therefore, the quality of hospital services provided to patients depends on failure prevention [7].

In many healthcare units, online fault-tolerant systems are used to ensure the permanent availability, reliability, and disaster recovery of data.

However, these mechanisms do not allow us to take preventive actions to avoid fault occurrence. Thus, the necessity of the development of faults prevention and prediction systems is emerging. These systems can predict faults with time in advance and allow taking early action to solve problems [7].

In the previous subsection, it was possible to conclude that are a lot of heterogeneous information systems so it is important to ensure a monitoring framework capable of monitoring all of these different systems.

There are several problems that influence database availability, such as the database can be inaccessible due to network problems or can be too slow and therefore not satisfy the users' requests.

According to the objective of preventing faults related to the resource limitation it has been selected the following statistics:

Processor utilization – The processor is one of the most important components, so it is necessary to constantly monitor its utilization by the user processes. Low values of processor utilization may indicate problems at the level of I/O. If the values are too high, it can compromise the functioning of the database.

Memory utilization – The memory is a key component to the speed of the database systems. The speed of access to data depends on the place where they are: memory or disk. If the data are in memory then access to them is faster.

DB time – This statistic gives information related to database response time. The response time is the period between an initial user request and the return of the results. In Oracle systems, this time is a sum of total time (including CPU time, IO time, Wait time). Therefore it is a good indicator of the workload of the system. Typically, this time increases with the number of simultaneous users or applications, but it also may increase due to large transactions.

Using a Business Intelligence tool, it is possible to analyze the data collected and create graphs that demonstrate the normal behavior of the database.

2.2 Business Intelligence

Business intelligence (BI) combines business analytics, data mining, data visualization, data tools, and the best practices to help institutions to make more data-driven decisions [8].

The concept of BI is comprehensive, but it can be defined as a combination of products, technologies, and methodologies that organize the critical information that a company needs to improve profits and its performance. It is considered the relationship between management and technology, where the raw material is information, and the final product is its knowledge [9].

This concept made possible the analysis and visualization of data available in real-time, previously considered useless and now considered essential assets to allow managers to have a clear perspective of the areas they must control and help the company make strategic decisions [10].

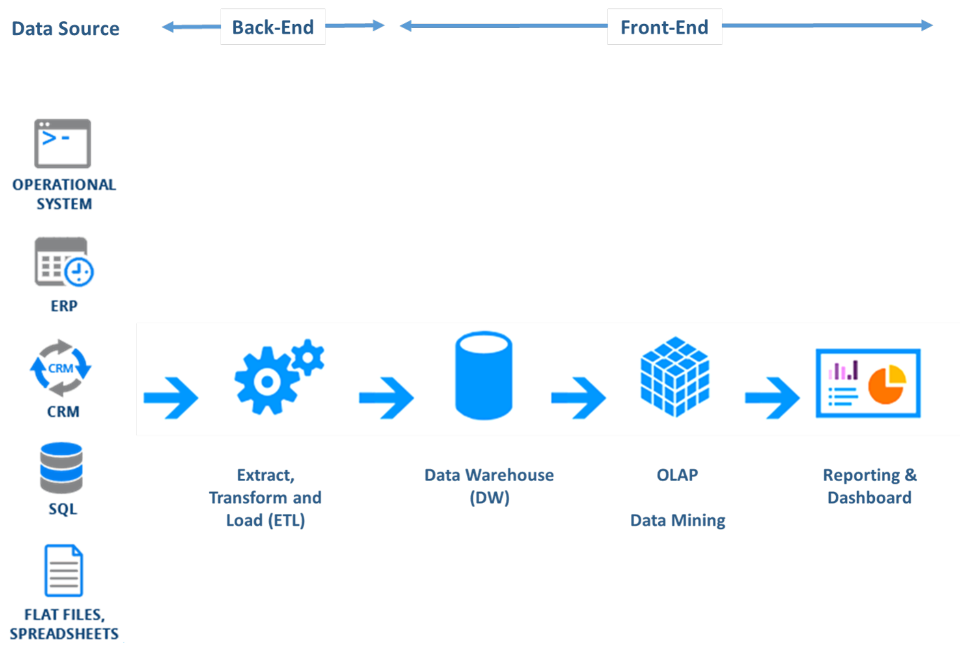


Figure 2.1: Business Intelligence Architecture

In practice, in modern business intelligence, there is a comprehensive view of an organization's data, which is used to drive change, eliminating inefficiencies, and quickly adapting to the market or supply changes [9].

BI has the task of extracting and processing data, depending on the project's requirements in question, to have it visualized, thus improving the availability and quality of information for decision making [11].

As illustrated in Figure 2.1, adapted from Kimball et. al. [11], there are five steps to consider in the BI architecture: operational source systems, ETL system, the data warehouse (DW), multidimensional database, and data visualization (reporting). Data and information are extracted from one or more sources, processed and transformed based on the requirements accordingly to the business needs, and are subsequently stored in a separate repository in a DW (Section 2.3). The described process is ETL – Extract, Transform, Load – covered in Section 2.4. The data are then prepared according to the project requirements, through Online Analytical Processing, OLAP queries and data mining applications, and, finally, submitted to the visualization phase, which communicates directly with the end-user.

2.3 Data Warehouse

In practice, business decisions are taken based on the analysis of past and current data, continuously collected during an enterprise's lifetime. A data analysis technology, widely accepted by research and industry, is based on data warehouse architecture [12]. In this architecture, data from multiple heterogeneous storage systems are integrated into a central repository, called a data warehouse (DW).

Data warehouses are mostly based on a so-called “multidimensional” data model, where essential events, e.g., medical interventions performed, are modeled as so-called facts, characterized by several hierarchical dimensions, e.g., time and patients, with associated measures, e.g., name and date of birth [12]. The multidimensional model is unique in providing a framework that is both intuitive and efficient, allowing data to be viewed and analyzed at the desired level of detail with excellent performance.

2.3.1 Kimball versus Inmon approaches

When a data warehouse is being designed for an organization, the two most common methods are the approaches proposed by Bill Inmon and Ralph Kimball.

In Bill Inmon’s data warehouse approach (the top-down design), the data is defined as a centralized repository for the entire enterprise. Since the data warehouse stores the “atomic” data at the lowest level of detail, it is necessary to separate and differentiate the data according to the different existing business areas to be analyzed using Data Mart’s after the DW is complete [13]. A Data Mart (DM) is a subsection of a Data Warehouse and is usually oriented towards a specific business area of the organization, and inside, the data are subdivided by areas (e.g., Finance, Human Resources, etc.). A Data Mart is created to complement a DW to satisfy the customer needs more effectively, restrict access, and, undoubtedly, reduce processing times, increasing the solution’s performance and speed.

Thus, in this approach, the data warehouse is at the center of the Corporate Information Factory (CIF), providing a logical framework for delivering business intelligence.

According to Inmon [13], the data warehouse is:

Subject-oriented: The DW data is organized so that all the data elements relating to the same subjects are linked together. Classical operations systems are organized according to the company’s applications (e.g., For an insurance company, the applications may be health and the insurance corporation’s major subject areas might be customer, policy, premium, and claim.)

Time-variant: The data alterations in the database are tracked and recorded so that reports can be produced showing changes over time. Time variance implies that all the values that a unit of data had, within a specific period, are stored in the data warehouse (data history)

Non-volatile: Data in the data warehouse is never overwritten or deleted. The data is loaded in a snapshot in a static format, read-only, and retained for future reporting.

Integrated: Data is fed from multiple sources into the data warehouse. As the data is fed, it is converted, reformatted, resequenced, summarized, etc. The result is a consistent data that has a single physical corporate image.

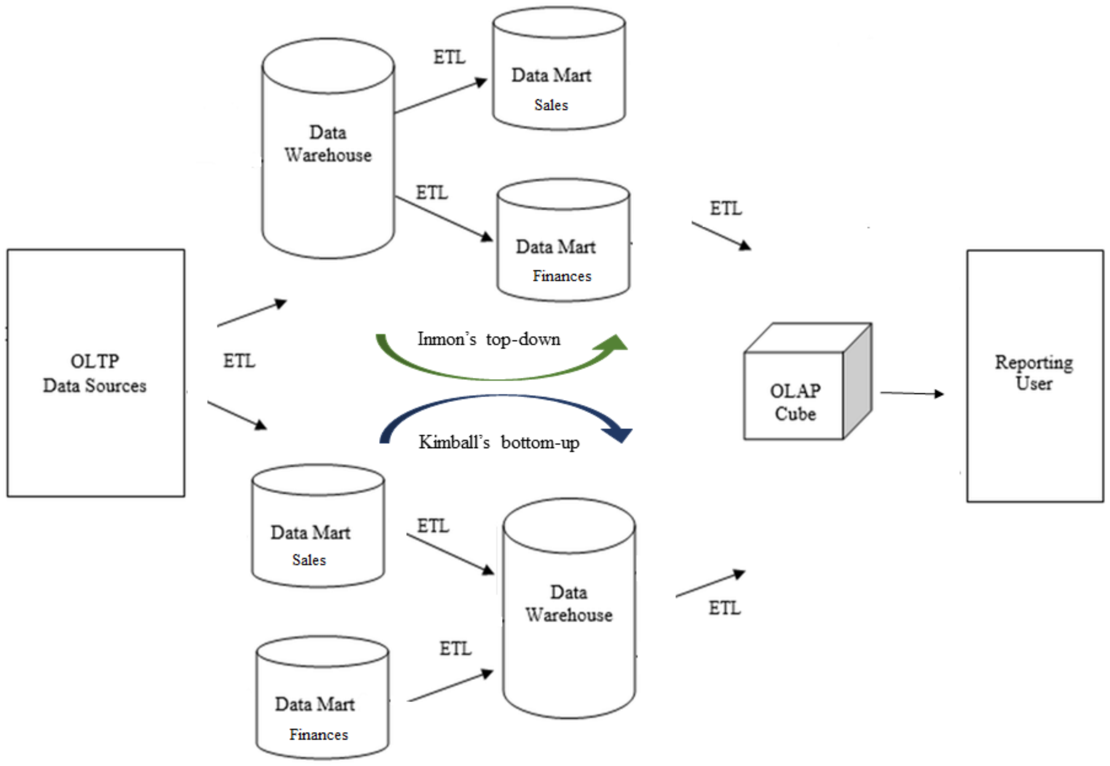


Figure 2.2: Inmon's top-down and Kimball's bottom-up approaches

In Ralph Kimball's dimensional design approach (the bottom-up design), the data marts facilitating reports and analysis are created first to provide a narrow view into the organizational data and, these can be combined into more massive data warehouse. In this approach, the first step to build the DW is to differentiate the data according to the business areas. Kimball defines the data warehouse as "a copy of transaction data specifically structured for query and analysis" [11].

On the other hand, Ralph Kimball's dimensional design approach (the bottom-up design), defending that DMs should be created initially, taking so that the first step in building the DW is to differentiate the data and according to with the surrounding business areas.

In Figure 2.2 adapted from Kimball et. al. [11], it is possible to compare both concepts behaviors in the architecture of a BI system and visualize Kimball and Inmon approaches.

2.3.2 Dimensional Modeling

Ralph Kimball defends the need for a logical design technique that seeks to present the data in an intuitive standard structure, allowing efficient and quick access. Thus, Kimball suggested a new methodology that would revolutionize the modeling of a Data Warehouse, dimensional modeling. This technique consists of a central table, called a fact table, and a set of surrounding tables: dimension tables. Each of the dimension tables has a primary key that corresponds precisely to one of the keys in the fact table (foreign keys) [14].

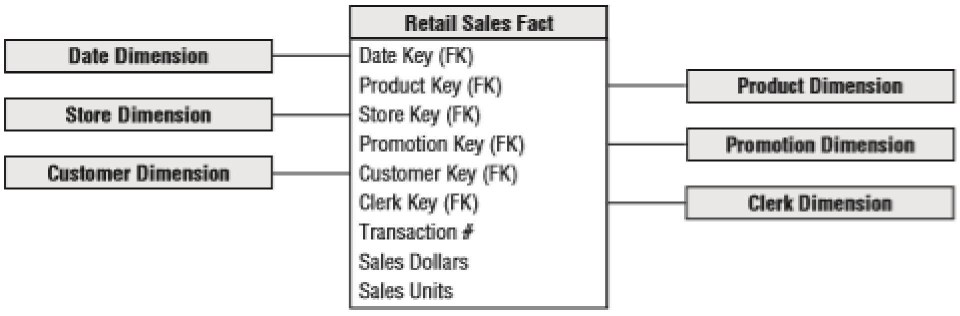


Figure 2.3: Star Schema

In order to build a Dimensional Model, it is necessary to identify the facts and dimensions for the construction of the respective tables. These are vital elements that define how the data is structured in the model in question.

There are two possible schemes in the dimensional model: Star schema and Snowflake schema represented in Figure 2.3 and 2.4 [11], respectively.

Star schema is a star shape model where each dimension table is linked to the central element: the fact table. In the case of the Snowflake schema, the dimension tables are related to other dimension tables to normalize the dimension information.

2.3.2.1 Fact tables

The fact tables store information on the organization's performance measures resulting from its events. Each table's row represents an event associated with a process, which contains the measurement information associated with that same event [14].

One of the core concepts of dimensional modeling is that all the measurement rows in a fact table have to be at the same level of detail (e.g., a laboratory test on laboratory measurements).

Thus, the facts contain crucial information for the analysis and consequent development of the company.

2.3.2.2 Dimension tables

Dimension tables are integral associations to a fact table and describe the objects involved in a BI project. The dimension tables store the textual context associated with a business process measurement event. While fact tables contain details about each instance of an object, the dimension table describes the "who, what, where, when, how, and why" allied with the event. These tables often have various columns or attributes (that consist of real words rather than abbreviations) and tend to have fewer rows than fact tables [14].

Each dimension is defined by a single primary key, PK, refer to the Product Key (notation in Figure 2.5), which serves as the basis to reference any attribute to a given fact table using a

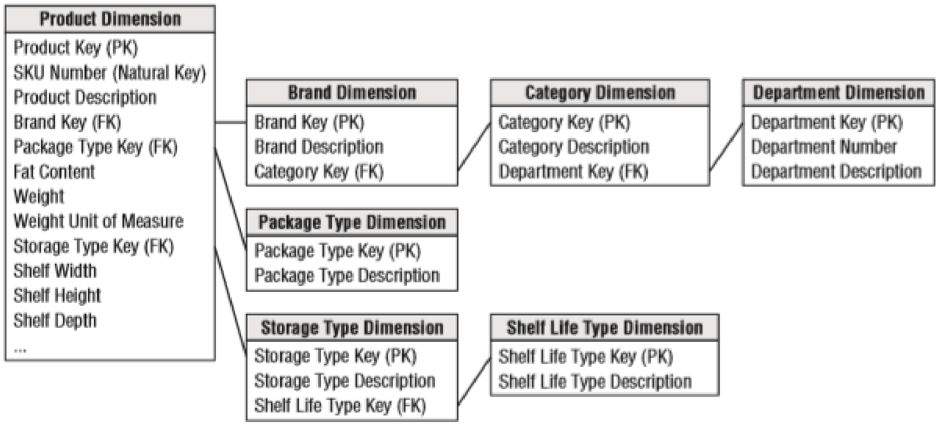


Figure 2.4: Snowflake Schema

foreign key (FK) because measurement data should not be repeated in multiple places for various organizational functions.

The data warehouse is as good as the dimension attributes since the BI environment's analytic power is directly proportional to the quality and extent of the dimension attributes [11].

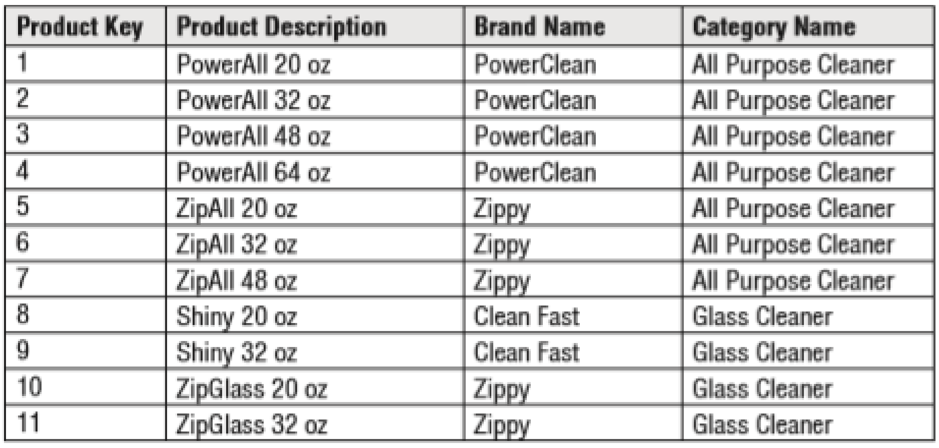
2.3.3 Data Warehouse Requirements

According to Kimbal, there are universal requirements for all data warehouse systems [11]. The DW must:

- make information easily accessible, fast, understandable and intuitive to the business user, not merely the developer;
- present information consistently that is carefully assembled from a variety of sources, cleansed, quality-assured, and released when it is fit for user consumption;
- adapt to user needs, business conditions and data to handle inevitable changes so that it does not invalidate existing data or applications;
- present information in a timely way. The DW team and business users need to have realistic expectations for data delivery if there is a short time to clean or validate it;
- be a secure bastion that protects the information assets and confidential data;
- serves as an authoritative and trustworthy foundation to build a decision support system;
- be accepted by the business community.

2.3.4 Clinical Data Warehouse

Traditional DW has as its primary objective, decision support, to increase productivity and efficiency in a business. These DWs are based on well-defined structures, data sources and objectives.



Product Key	Product Description	Brand Name	Category Name
1	PowerAll 20 oz	PowerClean	All Purpose Cleaner
2	PowerAll 32 oz	PowerClean	All Purpose Cleaner
3	PowerAll 48 oz	PowerClean	All Purpose Cleaner
4	PowerAll 64 oz	PowerClean	All Purpose Cleaner
5	ZipAll 20 oz	Zippy	All Purpose Cleaner
6	ZipAll 32 oz	Zippy	All Purpose Cleaner
7	ZipAll 48 oz	Zippy	All Purpose Cleaner
8	Shiny 20 oz	Clean Fast	Glass Cleaner
9	Shiny 32 oz	Clean Fast	Glass Cleaner
10	ZipGlass 20 oz	Zippy	Glass Cleaner
11	ZipGlass 32 oz	Zippy	Glass Cleaner

Figure 2.5: Sample rows from a dimension table

On the other hand, the main focus of a clinical DW is to facilitate the extraction of knowledge from clinical data. Traditional DWs are less focused on regulatory compliance compared to clinical DWs. The key feature of a clinical DW, as with all DWs, is to allow an organization to store and gain value from the data to support and improve decisions. The ability to standardize, group, analyze, explore and extract data from diverse and dispersed sources has been a challenge for the healthcare industry. While a traditional DW uses only a finite set of data sources, there are potentially hundreds of data sources in the industry [15].

In the Health sector, it is necessary to determine the requirements of clinical data, the purpose of the business, data integration, data quality and the ETL process to overcome the DW technologies' new challenges described below [15]:

Data format – The use of DW technology aims to define relationships in clinical data, discover disease trends, evaluate the performance of the different treatments and protocols used, improve the results of users and provide information to users from different areas (e.g., research, management). The medical data is collected during daily events are stored in various systems;

Business analysis – A clear understanding of the business purpose represents an important step in the clinical DW development process, since clinical DW does not fulfill its objectives without a clear definition of the business objective. In addition, medical data requirements are collected to understand the problem domain, in addition to determining the appropriate data model to design the clinical DW architecture;

Data integration – The data is integrated, transformed and consolidated to allow a unified view in the analysis of the data. The integration of data from different sources becomes a significant obstacle in situations of development of a clinical DW due to the complexity of the hospital environment, with various service practices, types of data and definitions. In addition, clinical data are integrated into various information systems, with incompatible and incomplete structures;

Data quality – In the health sector, the data are usually semi-structured, so it is necessary to assess their quality (relevance, consistency, validity, integrity). Data quality issues must be determined and resolved in order to determine the reliability of data for analysis, decision making and planning.

If these challenges are not addressed and managed carefully, can affect the quality of the data and, consequently, the Clinical DW.

2.4 Extract, Transform and Load

Extraction, Transformation, and Loading are procedures of a data loading technique responsible for extracting data from various sources, cleaning, optimizing, and inserting that data into a data warehouse [16]. This process is the most critical and time-consuming in building a DW.

ETL, as its name reveals, is based on the following three main phases.

Extraction, as its designation indicates, is the first ETL process, where the necessary information is extracted, from one or multiple sources (e.g., Flat Files, databases) available, to an intermediate storage space used between the data sources and the destination repository, called Data Staging Area (DSA) [17].

The second phase is the transformation responsible for manipulating the available information according to the business objectives and requirements. Tasks such as information cleaning (correcting misspellings, resolving domains conflicts, dealing with missing data, or parsing into standard formats), combining data from several sources, and duplicating data are examples used in the transformation process [11].

In the last phase, the previously treated information is loaded and aggregated into the DW [16].

2.4.1 Slowly Changing Dimension

Dealing correctly with the changes at the dimensions' data level is a critical success factor to the future of a Data Warehousing System.

Slowly Changing Dimension (SCD) is a dimension that stores and manages both current and historical data and is fundamental in tracking dimension records' history [18]. There are different approaches to deal with slowly changing dimensions to reduce the complexity of the ETL design, improve the ETL process performance, and minimize future data analysis computation.

In Table 4.2, it is summarized the several types of SCD identified by Kimball [18].

Table 2.1: Summary of the Slowly Changing Dimension Types

SCD Type	Methodology
0	Any detected change is disregarded and the DW is not updated.
1	The new dimension value overwrites the previous one, and there is no interest in maintaining any historical values of the modified attribute. This type is often used for processing data corrections.
2	All dimension history changes are preserved in the database using additional attributes to identify the record's validity. The standard method is to implement a binary active indicator (RecordActive), a start time (StartDate), and an expiration time (EndDate) to identify the active record period. The current record has the attribute End_Date as NULL and the RecordActive column set to 1. When the ETL process identifies a record with a different value in a Type 2 attribute, a new row is created in the dimension, with a new primary key but with the same identifier.
3	Only the current and previous values of the dimension's attribute are preserved.
4	The dimension is divided into two tables: one has all the current records (dimension_current), and the other stores all the updated records and, therefore, are expired (dimension_history).
6	Combines the approaches of types 1,2,3 (1+2+3=6).

2.5 Cloud Computing

Cloud computing provides services such as databases, analysis software and intelligence, storage, among others, through the Internet, to boost innovation, speed, and flexibility in accessing resources. This theme appeals to most companies and organizations today, as its traditional software brings high costs, and sometimes requires a high quantity and variety of hardware and software to be used. The cloud has the solution to these problems since the user only has to pay for the services he needs, and these are always just a click away [19].

There are several cloud models, types and services since the desired solution has to take into account the needs that the customer has. There are, therefore, three types of cloud implementations: public, private and hybrid.

2.5.1 Types of Clouds

- Public Cloud: In this case, the cloud service provider is the operator and owner of the public cloud, and it is who makes its resources available, such as servers and storage, through the Internet. Access to this type of cloud is done through the browser [20].

- Private Cloud: In the case of a private cloud, the organization/company is solely who provides and uses its resources. These clouds can be physically located in the data center of the company [20].

- Hybrid Cloud: This type of cloud, as the name implies, takes advantage of the public and private cloud, these are linked through a technology that allows data and applications to be shared between them. In this way, this solution provides greater flexibility implementation options, as it allows data movement and applications between public and private clouds.

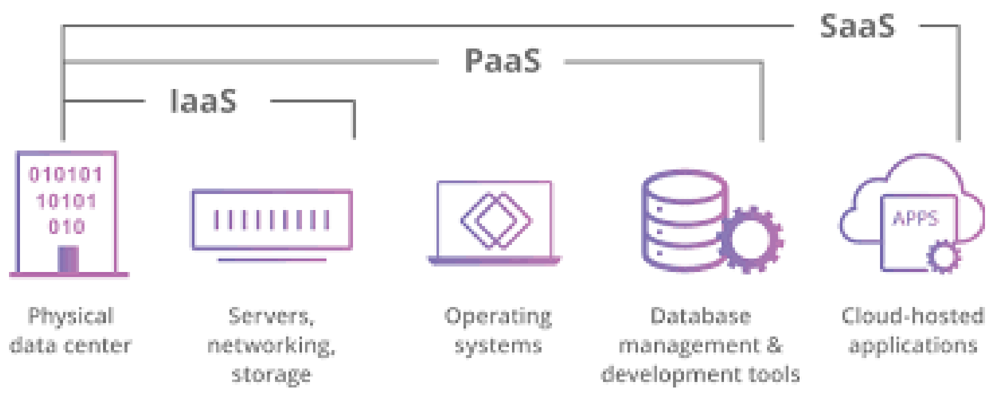

SaaS
PaaS
IaaS
010101
10101
010
Physical data center
Servers, networking, storage
Operating systems
Database management & development tools
APPS
Cloud-hosted applications

Figure 2.6: Types of Cloud Services

2.5.2 Types of Services

Most cloud services belong to one of these three different categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

The knowledge of these categories allows the user of cloud services to make better decisions taking into account your business needs and objectives as seen in Figure2.6.

- **Infrastructure as a Service**: IaaS is an infrastructure that provides services to the user integrated in data centers making it an automatic computing infrastructure, being managed with Internet use. Thus, the IaaS promotes a reduction in fixed costs to the extent that the user only pays for the services he uses, and avoids expenses on configuration and management of local data centers since they are virtual. It also offers advantages such as: continuous innovation, faster response to changes in the business, improvements in business continuity and disaster recovery, automatic backups, etc [21].

- **Platforms as a Service**: PaaS provides a wide range of features, from simple applications and cloud-based environments to sophisticated applications. It is the platform was initially designed to support the entire Web application development: construction, implementation, testing, management, and update. Thus, this platform offers advantages such as the reduction of time spent on the program as it provides pre-programmed application components, increase in development capacities without the need to increase teams, greater ease in managing the "life cycle" of applications and developing multiplatform (including mobile platforms), among others [22].

- **Software as a Service**: This last, more comprehensive service offers the user the possibility to connect and take advantage of cloud applications, over the Internet. In this way, the user rents the use of a cloud application for the respective organization and the organization's constituents connect to it, usually through the browser [23].

2.5.3 On-premises versus Cloud

With the evolution that the cloud has had over the years, and the advantages that come from it, it appears the need to understand the differences between these two approaches.

The main difference between cloud and on-premises is where the software is deployed.

On-premises means that the company has the software installed on their computers and servers, and it is managed and configured by them. On the other hand, in the cloud, the IT environment is located on servers outside the company's location and can be accessed through a web browser [24].

2.5.4 Cloud Computing Technologies

This Subsection presents the three leading Cloud Computing technologies (Amazon Web Services, Microsoft Azure and Google Cloud Platform) and its main features.

2.5.4.1 Amazon Web Services

Amazon Web Services (AWS) is a cloud computing platform that was launched by Amazon in 2006 and has been the leader in the market since.

Although it offers PaaS and SaaS services, its main features are Infrastructure as a Service, storage solutions, content migration as well as machine learning features. It is considered a service with good options for configuration, monitoring, security, and flexibility [25].

Unlike Microsoft Azure, it does not consider the use of private clouds.

2.5.4.2 Microsoft Azure

The scope of Microsoft Azure is similar to that of AWS. This platform was launched by Microsoft in 2010 and offers several features such as storage, databases, virtual machines, development and analysis tools, among others [26].

Azure is fully compatible with other Microsoft systems, such as Windows Server, System Center Active Directory.

2.5.4.3 Google Cloud

Unlike AWS and Azure, which entered this market as service providers based on IaaS, Google's first public cloud service was based on PaaS, the App Engine, launched in 2008. Although cloud computing surfaces like Azure or AWS have a broad service portfolio, Google Cloud Platform (GCP) has earned its reputation mostly due to its cloud computing services in the open source community, analysis tools, artificial intelligence, and machine learning [27].

2.5.4.4 Cloud Technologies Comparison

Cloud selection depends on each user's needs and it is common for companies to use several cloud service providers for different parts of the operation, designating them as semi-cloud. AWS, Microsoft Azure and Google Cloud offer very similar basic capabilities, such as flexible computing and storage. They share the same elements as a public cloud, such as self-service, instant provisioning, self-scaling, security and identity management tools. However, these differ from each other in relation to some characteristics, namely computing, storage, networking, databases, and security [28].

Computing: It is one of the main functions of the Infrastructure service. Processing and computing are the main capabilities of a computer, as such, it is also essential that the chosen supplier comes from capabilities that meet the needs of the customer. A suitable cloud provider can compute huge tasks in just minutes.

In the cloud computing market, selecting the best cloud is a difficult process, as it is necessary to consider CPU performance, memory, and computing price.

As we can see in Table 2.2, all three suppliers are a platform as a Service and include serverless functions. These are programmatic functions that are hosted on managed infrastructure, are invoked through the Internet, are hosted and maintained by cloud computing companies.

Storage: One of the most important capabilities of cloud services is storage. The exponential increase in communication and the consequent information growth, forces the presence of a service with an efficient storage power and a large volume.

Database: All the suppliers offer SQL and non-relational database (NoSQL) solutions when it comes to databases. In Table 2.2 it is easily noticeable that AWS and Microsoft Azure have characteristics with the same objectives. Concerning GCP, it appears that there are no database migration resources, which is important when more than one service is used. This feature allows you to migrate data from commercial data bases as long as these are open-source, such as MySQL, with the least possible downtime. The three competitors also offer management capabilities for the data warehouse which allows a quick, efficient and low cost analysis using BI and SQL tools.

Security: In this context, only two cloud suppliers provide security services that are one of the biggest attractions of cloud users. It is visible in the Table that Google is lacking in terms of security features. The Google environment does not consider cloud directory and does not provide firewall, which helps protect against common Internet attacks. Cloud directory services software is a modern implementation of identity management and directory solutions delivered through the cloud. These solutions provide many simple integrations to help expedite identity management operations across different networks and applications.

Costs: Considering the smallest instance that includes 2 virtual CPUs and 8 GB of RAM, AWS, Azure and Google Cloud will cost approximately \$69, \$70 and \$52 per month, respectively.

On the other hand, the largest instance that includes 3.84 TB of RAM and 128 vCPUs will cost around \$3.97/h, \$6.79/h, \$5.32/h in AWS, Azure and GCP respectively.

Table 2.2: Cloud Technologies Characteristics

Characteristics	Service	Amazon Web Service	Microsoft Azure	Google Cloud
Computing	PaaS	Elastic Beanstalk	Cloud Services	Google App Engine
	Serverless Functions	AWS Lambda	GCP Functions	Azure Functions, Azure Event Grid
Storage	Object Storage	Simple Storage Services	Blob Storage	Google Cloud Storage
Database	Managed relational database-as-a-service	RDS	SQL Database, Database for MySQL and PostgreSQL	Cloud Spanner and Cloud SQL
	NoSQL	Amazon DynamoDB	Azure Cosmos DB	Google Cloud Bigtable and Datastore
	Managed data warehouse	Amazon Redshift	SQL Data Warehouse	Google Cloud BigQuery
	Database Migration	AWS Database Migration Services	Azure Database Migration Services	–
Security	Web Firewall	AWS Web Application Firewall	Azure Application Gateway and Firewall	–
	Directory Services	AWS Directory Services	Azure Active Directory	–
Costs	Smallest Instance	$69/month	$70/month	$52/month
	Largest Instance	$3.97/h	$6.79/h	$5.32/h

Azure charges resource usage per minute while AWS charges per hour. This means that if a user uses the feature for 61 minutes on Azure they will pay 61 minutes while on AWS they will pay 2 hours [29].

2.6 Reporting Technologies

The Business intelligence systems offer features that analyze and report data for a streamlined presentation, creating a unified user-friendly dashboard showing analytics, metrics and key performance indicators.

Dashboards present interactive reports, with information typically relating to the obtained performance, aligning the objectives with the organization's strategy, and monitoring its progress.

Tableau, QlikView and Microsoft Power BI are some of the best BI systems available in the market.

2.6.1 Tableau

Tableau is a development platform, launched in 2003, that has a user-friendly interface allowing non-technical users to quickly and easily create customized dashboards to provide insight into a broad spectrum of business information. One of the key feature is to allow users to drag and drop data points into the visualization dashboard. The users can import all sizes and types of data from a range of sources like Oracle, MySQL, SAP, NoSQL, etc [30].

The main disadvantages include the lack of support for data encryption, the lack of multi-location support and not having a user-friendly interface for report-builder.

2.6.2 QlikView

QlikView is a data discovery and customer insight platform, launched in 1993 by Qlik, well-known for its data associations and relationship functionality, keeping data in-context automatically. It

Table 2.3: Report Technologies Characteristics

Characteristics		Tableau	Qlik	Power BI
Analytic Capabilities		Supports R and Python languages-based visualization	Does not support R or Python	Supports R language-based visualization
Visualization Capabilities		Perfect Graphics and visualization capabilities	Self-service Tool	Easy to use platform
Cloud Capability		It is compatible with Microsoft Azure	It offers a SaaS cloud	It is compatible with Microsoft Azure, Amazon Web Services, etc.
Ease of Learning		Requires a Data Science background	Requires a Data Science background	User-friendly due to the similarity of Excel interface and formulas
Storage Limit		100GB on cloud storage	500GB on cloud storage	10GB on cloud storage
Costs	Basic Version	Free but with limit features	Free but with limit features	Desktop version is free
	Advanced Desktop Version	35$per month/per user	20$/per month/per user	10$/per month/per user

offers several features, including its "pre-canned" dashboards which are dashboards that are pre-configured for data analysis and interpretation [31].

Similarly to Tableau, QlikView connects to the majority of data sources. It has a strong association engine built-in, which means that users can conduct direct and indirect searches across their data, or within a single field. All stored data is located in RAM, which means conversions, queries and searches can happen quicker and more efficiently [32].

However, it does not support online analytical processing (OLAP), lacks ad hoc reporting functionality and does not allow users to schedule receiving BI reports at specific times in a particular format.

2.6.3 Microsoft Power BI

Microsoft Power BI is a powerful tool, founded in 2011 by Microsoft, that helps users gather and interpret proprietary data to make better business decisions and it offers two solutions: Power BI Services, an SaaS deployment, and Power BI Desktop, the on-premises based version [33].

One of the biggest advantages of this tool is the development of dashboards that update in real-time from all data sources, so users will never be out of the loop about crucial business information. Users can also receive alerts, share insights, create reports and stay informed from any mobile device (compatible with Apple, Android, and Windows) via the application.

On the other hand, the graphical visualization is fairly limited as compared to other BI tools and is relatively difficult to work with huge data sets [32].

2.6.4 Reporting Technologies Comparison

Business analytics tools are used in business representation of data for the concerned stakeholders so it is necessary to take into consideration the clients needs.

Although Tableau, QlikView and Microsoft Power BI offer similar services, they have some differences between them that are presented in Table 2.3 [34].

Regarding the analytics capabilities, the three competitors provide features such as regressions, clustering, and predictive analytics but Tableau has the advantage of the drill-down and filtering options and Qlik lacks some analytics features [34].

Tableau and Power BI are both easy-to-use software, known for their perfect graphics and visualization capabilities. Qlik is a self-service analytics tool that provides good visualizations that are dynamic due to the in-memory engine.

Tableau is compatible with most cloud platforms such as Microsoft Azure, Amazon Web Services, etc. Qlik offers a SaaS cloud product. Power BI is compatible with Microsoft Azure which offers cloud-software called “Cloud-first”.

Power BI is an easy work tool due to the similarity of Excel interface, formulas, and other features. On the other hand, Tableau and Qlik require that the user have some programming skills knowledge.

The Qlik Sense Cloud Business, Tableau and Power BI subscription limits allow 500GB, 100GB, and 10GB, respectively, of cloud storage of data. It is possible to expand the data storage capacity by paying a fee.

In terms of cost Power BI leads the race because of less expensive membership fees and available features in the free version.

2.7 Microsoft Azure Resources

After analyzing Section 2.5.4.4 and Section 2.6.4, it is possible to understand the main differences between the cloud and reporting technologies. However, it is also possible to verify that the ranking criteria of this technologies depends entirely on the scope and objectives of the organization that the want to purchase. In the case of BUSINESSTOFUTURE (B2F), the company opted for Microsoft technologies, that is, Azure and Power BI as a cloud computing and reporting technology, respectively. In the project in question, the technologies adopted were Microsoft technologies stated above.

The present section contains an approach and definition of the Microsoft Azure resources operation currently used in the development of this dissertation solution.

2.7.1 Azure SQL database

Azure SQL Database is a scalable, cloud and managed relational database service that provides SQL Server compatibility [35].

The technology is based on a pay-per-month model and it has two purchasing packages: Database Transaction Unit (DTU)-Based and Virtual Core (vCore)-Based.

The Database Transaction Units (DTUs) represents a mixture of the CPU, Memory and Data I/O and Log I/O performance metrics as a single performance unit for Azure SQL Database. The DTU basic purchase model costs $4.8971/month and it has 5 DTUs and 2GB storage.

2.7.2 Azure Storage Account

The Azure Storage platform is Microsoft’s cloud storage solution for data storage scenarios, offering a massively scalable object store for text and binary data (Azure Blob), disk storage for Azure virtual machines (Azure Disks), a file system service for the cloud (Azure Files), a messaging store for reliable messaging (Azure Queues), and a NoSQL store (Azure Tables).

Data in Azure Storage is accessible from anywhere using HTTP or HTTPS. Microsoft provides client libraries for Azure Storage in a variety of languages, including .NET, Java, Node.js, Python, PHP, Ruby, Go, and others, as well as a mature REST API. Azure Storage supports scripting in Azure PowerShell or Azure CLI [36].

The cheaper billing plan includes locally redundant storage (LRS) Archive Block Blob with 3-year reserved capacity and it costs $0.00081/GB per month.

2.7.3 Azure Data Factory

The Azure Data Factory is a service designed to allow developers to integrate different data sources and to store relational and non-relational data. The role of Azure Data Factory is to manage a Cloud service that is built for complex hybrid extract-transform-load (ETL), extract-load-transform (ELT), and data integration projects [37].

This platform is able to store, process, analyze, and visualize data of any variety, volume, or velocity.

The technology adopted a pay-per-need model where the user can pay a fee per hour or for each activity that is executed.

2.7.4 Logic Application

Logic Apps is the Azure Integration software as a service (SaaS) solution from Microsoft that can connect applications, data, and devices anywhere on-premises or in the cloud by orchestrating calls to system APIs [38]. Each connector has a cost of $0.000125.

It is possible to access SQL Server Integration Services (SSIS) using a connector that supports the on-premises data gateway.

A logic app is comprised of a set of building blocks that work together to construct a process that orchestrates integration between various parts. Those building blocks are workflows, triggers, actions and flow controls.

A workflow is a sequence of API calls and flow control elements that defines a business process automation – each logic app defines a workflow. A trigger is the event that kicks off the workflow right before an action (HTTP call executing an operation against a System API) is executed. Flow controls control the flow of execution of the workflow, analyzing the messages from a trigger or previous actions.

The technology is based on a pay-per-execution model, where the user only pay for the logic apps steps that are executed every time a new logic app instance is invoked [38]. Each action has a cost of $0.000025.

2.7.5 Azure Functions

Azure Functions (AF) is a serverless compute service that allows the user to ran event-triggered script or piece of code without having to explicitly provision or manage infrastructure. Azure Functions support different languages: C#, F#, JavaScript, node.js.

Azure Functions consumption plan is billed based on per-second resource consumption and executions. Consumption plan pricing includes a monthly free grant of 1 million requests and 400,000 GB-s of resource consumption per month per subscription. If the user exceeds this grant, each execution time will cost $0.000016/GB-s and each million request will cost $0.20 [39].

2.8 Summary

The present Chapter is divided into seven sections. The first section gives the reader an overview of the currently existing Healthcare Information Systems in Portugal. The next three sections cover the main steps inherent to the Clinical Solution construction: Business Intelligence (BI), Data Warehouse (DW), and ETL (Extraction, Transformation and Loading). The fifth section approaches the several Cloud Computing Services and Technologies available in the market. The Business intelligence systems are always associated with Reporting Technologies that create a unified user-friendly dashboard to analyze metrics and key performance indicators. The sixth section presents the different Reporting Technologies available. Finally, the last and seventh section presents an overview of all the Microsoft Azure Resources used in this dissertation.

Chapter 3

Clinical Database

This chapter describes, in detail, the freely-available online clinical database, MIMIC-III Demo v.1.4., and the required adaptations to create the Clinical database. Also, it presents the Clinical Data Warehouse's five data marts (hospital admission services, electronic charted measurements, medical interventions, microbiology and laboratory tests) and the corresponding relational model.

3.1 MIMIC-III v1.4

MIMIC (Medical Information Mart for Intensive Care) integrates deidentified, comprehensive clinical data of patients admitted, from June 2001 to October 2012, to the Beth Israel Deaconess Medical Center in Boston, Massachusetts [40]. The latest version of MIMIC, MIMIC-III v1.4, was released on 2 September 2016 and comprised 61,532 intensive care unit stays: 53,432 stays for adults and 8,100 for neonatal patients [41].

The data's open nature supports a diverse range of clinical studies spanning epidemiology, clinical decision-rule improvement, and electronic tool development to be reproduced and improved in ways that would not otherwise be possible. Among these researches, the eICU program and the ICU (Intensive Care Unit) discharge delay analysis were both relevant papers to conduct the short report of ICU admissions and medical tests conducted, presented in Section 5.3.

A telehealth ICU, or teleICU, is a centralized monitoring framework that provides remote appointments and reactive alerts for ICU patients. The eICU Collaborative Research Database (eICU-CRD) resulted in a publicly available database sourced from the eICU program and the MIMIC-III database [42].

After implementing the eICU program, massive volumes of data were compiled and streamed for real-time monitoring, during the clinical routine, by a remote ICU unit.

eICU-CRD includes the APACHE IV system that predicts patient mortality using a set of parameters regarding the patients' first 24 hours: physiologic measurements, administrated treatments, and admission diagnosis. These data provide an informative estimate of the patient's medical condition severity on admission to the ICU, allowing hospitals to identify which policies may benefit patient outcomes [42].

According to Bose et. al. [43], while the economic implications of discharge delays (time between when a patient ready to be discharged and when they leave the ICU) are substantial, it did not significantly contribute to hospital mortality or increased the discharge length stay.

3.1.1 Methods

The MIMIC-III database was populated with data acquired during routine hospital care to not interfere with the caregivers' workflow. Data was acquired from several sources, including Social Security Administration Death Master File, hospital electronic health record databases, and archives from two critical care information systems: Philips CareVue Clinical Information System (models M2331A and M1215A; Philips Health-care, Andover, MA) and iMDsoft MetaVision ICU (iMDsoft, Needham, MA [41]. The information from CareVue and MetaVision systems was combined when building the events tables, except for the INPUTEVENTS_CV and INPUTEVENTS_MV tables, which store inputs for monitored patients. The mortality dates registered outside the hospital were collected using the Social Security Administration Death Master File [41].

3.1.1.1 Deidentification

Before data was incorporated into the MIMIC-III database, it was first deidentified following Health Insurance Portability and Accountability Act (HIPAA) standards using data cleansing and date shifting [41]. The deidentification process for structured data required removing all identifying data elements listed in HIPAA, including fields such as patient name, telephone number, address, dates, diagnostic reports, and physician notes. Regarding the dates, these were shifted into the future by a random offset for each patient consistently to preserve intervals, resulting in stays that occur between the years 2100 and 2200 [40]. Time of day, day of the week, and approximate seasonality were preserved during date shifting. Accordingly to HIPAA regulations, dates of birth for patients aged over 89 appear in the database with ages above 300 years to obscure their real age.

It was not necessary to require patient consent since the project did not impact clinical care, and all protected health information was deidentified [41].

3.2 Data Description

Table 3.1 summarizes the MIMIC-III v.1.4 relational database's 26 tables, divided into four categories: definition and tracking of patient stays, dictionary tables, data associated with critical care units, and hospital record system [41].

Generally, five descriptive tables define and track patient stays: ADMISSIONS, PATIENTS, ICUSTAYS, SERVICES, and TRANSFERS. Tables prefixed with "D_" are dictionaries and provide definitions for identifiers: D_CPT, D_ICD_DIAGNOSES, D_ICD_PROCEDURES, D_ITEMS, and D_LABITEMS. The other tables contain time-stamped patient care data, such as physiological measurements, medical interventions, laboratory, and microbiology test results, discharge

Table 3.1: An overview of the 26 data tables comprising the MIMIC-III (v1.4) critical care database.

Category	Table	Description
Patient Stay Description	ADMISSIONS	Every unique hospitalization for each patient in the database (defines HADM_ID).
	CALLOUT	Information regarding when a patient was cleared for ICU discharge and when the patient was actually discharged.
	ICUSTAYS	Every unique ICU stay in the database (defines ICUSTAY_ID).
	PATIENTS	Every unique patient in the database (defines SUBJECT_ID).
	SERVICES	The clinical service under which a patient is registered during their hospital stay.
	TRANSFERS	Patient movement from bed to bed within the hospital, including ICU admission and discharge.
Critical Care Unit Data	CAREGIVERS	Every caregiver who has recorded data in the database (defines CGID).
	CHARTEVENTS	All charted observations for patients. The electronic chart displays patients' routine vital signs and any additional information relevant to their care
	DATETIMEEVENTS	All recorded observations which are dates, for example time of dialysis or insertion of lines.
	INPUTEVENTS_CV	Intake for patients monitored using the Philips CareVue system while in the ICU.
	INPUTEVENTS_MV	Intake for patients monitored using the iMDSoft Metavision system while in the ICU.
	NOTEEVENTS	Deidentified notes, including nursing and physician notes, ECG reports, imaging reports, and discharge summaries.
	OUTPUTEVENTS	Output information for patients while in the ICU.
	PROCEDUREEVENTS_MV	Patient procedures for the subset of patients who were monitored in the ICU using the iMDSoft MetaVision system.
Hospital Record System Data	CPTEVENTS	Procedures recorded as Current Procedural Terminology (CPT) codes.
	DIAGNOSES_ICD	Hospital assigned diagnoses, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system.
	DRGCODES	Diagnosis Related Groups (DRG), which are used by the hospital for billing purposes.
	LABEVENTS	Laboratory measurements for patients both within the hospital and in out patient clinics.
	MICROBIOLOGYEVENTS	Microbiology measurements and sensitivities from the hospital database.
	PRESCRIPTIONS	Medications ordered, and not necessarily administered, for a given patient.
	PROCEDURES_ICD	Patient procedures, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system.
Dictionaries	D_CPT	High-level dictionary of Current Procedural Terminology (CPT) codes.
	D_ICD_DIAGNOSES	Dictionary of International Statistical Classification of Diseases and Related Health Problems (ICD) codes relating to diagnoses. These codes are assigned at the end of the patient's stay and are used by the hospital to bill for care provided.
	D_ICD_PROCEDURES	Dictionary of International Statistical Classification of Diseases and Related Health Problems (ICD) codes relating to procedures. These codes are assigned at the end of the patient's stay and are by the hospital to bill for care provided.
	D_ITEMS	Dictionary of ITEMIDs appearing in the MIMIC database, except those that relate to laboratory tests.
	D_LABITEMS	Dictionary of ITEMIDs in the laboratory database that relate to laboratory tests.

summaries, caregiver observations, billing information, medication records, electrocardiogram reports, and imaging studies. These classes are summarized in Table 3.2 [41].

The tables are connected by the following identifiers: SUBJECT_ID (unique patient), HADM_ID (unique admission to the hospital), ICUSTAY_ID (unique admission to an intensive care unit), CGID (unique caregiver), and ITEMID (unique measurement).

3.3 Data Access

MIMIC-III is provided as a collection of comma-separated value (CSV) files, with scripts to import the data into database systems, including PostgreSQL, MySQL, and MonetDB. Since the database contains sensitive information regarding patients' clinical care, researchers must formally request access via a process documented on the MIMIC website [41].

The access to the integral database is only granted if the researcher completes the "Data or Specimens Only Research" course and signs a data use agreement, which outlines appropriate data usage and security standards [41]. When the application is approved, the researcher receives the instructions to download the database from PhysioNet.

The MIMIC-III v.1.4 Clinical Database Demo contains all intensive care unit (ICU) stays for 100 randomly selected patients and excludes the NOTEEVENTS table [44]. These patients have a

Table 3.2: Classes of data available in the MIMIC-III critical care database.

Class of data	Description
Billing	Coded data recorded primarily for billing and administrative purposes. Includes Current Procedural Terminology (CPT) codes, Diagnosis-RelatedGroup (DRG) codes, and International Classification of Diseases (ICD) codes
Descriptive	Demographic detail, admission and discharge times, and dates of death
Dictionary	Look-up tables for cross-referencing concept identifiers (i.e., International Classification of Diseases (ICD) codes) with associated labels
Interventions	Procedures such as dialysis, imaging studies, and placement of lines
Laboratory	Blood chemistry, hematology, urine analysis, and microbiology test results
Medications	Administration records of intravenous medications and medication orders
Notes	Free text notes such as provider progress notes and hospital discharge summaries
Physiologic	Nurse-verified vital signs, approximately hourly (e.g., heart rate, blood pressure, respiratory rate)
Reports	Free text reports of electrocardiogram and imaging studies

date of death (DOD) but do not necessarily died during individual hospital admission or ICU stay.

Researchers can review the structure and content of MIMIC-III Demo and, posteriorly, determine whether or not to acquire the full dataset for a more thorough analysis.

The demo dataset latest version (v.1.4.) can be downloaded either as 25 comma-separated-value (CSV) files or as a single PostgreSQL database backup file (PostgreSQL 9.5) [44].

3.4 Limitations

Although MIMIC-III demo v.1.4 has useful information, the low number of patients registered (100 samples) and the random admissions time-shifting into the future difficulties the data analysis. Consequently, the results can only be interpreted by the year season and can not be linked to external factors (e.g., local epidemics or massive public transport accidents) that directly affect healthcare responses.

This database also contains dictionary tables with International Classification of Diseases Version 9 (ICD-9) codes of diagnoses, medical procedures, and drugs administrated used by the hospital billing system. However, the corresponding invoice costs are not available; therefore, it is impossible to evaluate the patient's expenses during his hospital stay.

In the demo version, all the listed patients are deceased, which makes difficult to assess the impact of the medical interventions, hospitalization period, and admission waiting time in the hospital survival rate.

Due to the storage's imprecision across the caregivers table, there is missing information about the professionals' occupation, and distinct caregivers with the same name may be considered the same caregiver.

3.5 Clinical Database

The Clinical Database resulted from the ETL (Extract, Transform and Load) process, described in the next chapter, using the MIMIC-III demo v.1.4 as the source.

The billing and the medications tables of the MIMIC-III demo v.1.4 are not present in the Clinical Data Warehouse.

Unfortunately, it was not possible to match the coded data (Current Procedural Terminology, Diagnosis-Related Group, and International Classification of Diseases codes) associated to the billing tables to health insurance and hospital costs, making it impossible to conduct a Hospital revenue analysis.

Drug products are identified and reported using a unique, three-segment number, called the National Drug Code (NDC), which serves as a universal product identifier for drugs. FDA publishes the listed NDC numbers and the information, including National Average Drug Acquisition Cost (NADAC), which is updated daily. However the medications records were associated with internal hospital codes, making it impossible to calculate the hospital orders and costs.

The Clinical Data Warehouse has nineteen dimension tables and five fact tables divided into five data marts: hospital admission services (Figure 3.1), electronic charted measurements (Figure 3.2), medical interventions (Figure 3.3), microbiology (Figure 3.4) and laboratory (Figure 3.5) tests.

Note that each table's first row corresponds to its primary key (PK), and each table contains a not null integer id except for DW_DIM_D_CARE_UNITS and DW_DIM_D_SERVICES that has as an identifier a not null acronym.

In the Slowly Changing Dimension type 2 (see Table 4.2), when the value of a determined attribute changes, the current record is closed (RecordActive is set to 0), and a new record is created with the adjusted data values (RecordActive is 1).

Each record contains the start time (StartDate) and the expiration time (EndDate) to identify the active record period.

In this model, DW_DIM_CAREGIVERS, DW_DIM_ADMISSIONS, DW_DIM_ICUSTAYS, DW_DIM_D_ITEMS_PROCEDURE, DW_DIM_D_ITEMS_CHART, and DW_DIM_D_LABITEMS have three additional attributes (RecordActive, StartDate, EndDate) to preserve the relevant historical data.

DW_DIM_PATIENTS

The DW_DIM_PATIENTS dimensional table defines a single patient, PatientId, and the patient's demographics: gender, date of birth, and death if applied. This table characterizes 100 subjects obtained from the CareVue and Metavision ICU databases.

DW_DIM_CAREGIVERS

The DW_DIM_CAREGIVERS table provides information regarding the type of caregiver represented by a unique identifier (CaregiverId), occupation (CaregiverLabel), and respective description (CaregiverDescription).

FlagActive is a binary integer that designates whether the caregiver is currently a hospital's employer. FlagActiveStartDate and FlagActiveEndDate define the period of the caregiver's work at the Beth Israel Deaconess Medical Center.

When a caregiver changes occupation, the respective record is closed, and RecordActive set to zero. It is created a new record, with the same CaregiverId updated CaregiverLabel and respective CaregiverDescription, and RecordActive equal to 1.

This dimensional table contains 7567 caregivers gathered from the CareVue and Metavision ICU databases.

Date Dimensions

Unlike the other dimensions, the date dimensions tables (DW_DIM_DATE_DISCHARGE, DW_DIM_DATE_ADMISSION, and DW_DIM_DATE_EVENTS) were built, using stored procedures, to cover the years from 2100 and 2200. These tables have the attributes DayId (in the format yyyymmdd), MonthId, Month description, TrimesterId, SemesterId, and YearId, allowing filtering the patients' admission, discharge, and medical event (e.g., procedure, laboratory and microbiology tests) by these time variables.

DW_DIM_ADMISSIONS

The DW_DIM_ADMISSIONS dimensional table defines all patient's hospital admission, AdmissionId, covering the period between 1 June 2001 and 10 October 2012. This dimension is characterized by:

- admission type (AdmissionType), which can be Elective (previously planned hospital admission), Urgent or Emergency (unplanned medical care);
- patient's admission and discharge date (PatientDoadmit and PatientDodisch, respectively) and time (PatientToadmit and PatientTodisch, respectively);
- a free text preliminary diagnosis (AdmissionDiagnosis) for patient's hospital admission;
- a flag (AdmissionExpireFlag) that indicates if the patient died within the given hospitalization (0 - survive; 1 - death) and the patient's death date (PatientDodeath) if applied;
- patient's id, PatientId, and the respective foreign key (FK), PatientKey, that corresponds to the DW_DIM_PATIENTS's PK.

When an incorrect patient characterizes an admission, the respective record is closed. It is then created a new record, with the same AdmissionId, with the updated PatientId and respective demographic information.

It is composed of 129 records retrieved from the Beth Israel Deaconess Medical Center.

DW_DIM_D_CARE_UNITS

DW_DIM_D_CARE_UNITS is a care unit dictionary described by a unique identifier, CareUnitId, and the respective description, CareUnitDescription. FlagActive is a binary integer that designates whether the care unit is discontinued. Consequently, the StartDate and EndDate represent the active period of the care unit. This dimensional table currently describes the Coronary care unit, Cardiac surgery recovery unit, Medical intensive care unit, Neonatal intensive care unit, Neonatal ward, Surgical intensive care unit, and Trauma/surgical intensive care unit.

DW_DIM_D_SERVICES

DW_DIM_D_SERVICES has an identical organization to the DW_DIM_D_CARE_UNITS table. The currently listed services range from Dental to Obstetrics.

DW_DIM_ICUSTAYS table

The DW_DIM_ICUSTAYS table represents each of the 136 patient's Intensive Care Unit (ICU) stays, IcustayId, registered in the hospital database, and compresses the following:

- patient's admission and discharge date (PatientDoadmit and PatientDodisch, respectively) and time (PatientToadmit and PatientTodisch, respectively);
- admission's id, AdmissionId, and the respective FK, AdmissionKey references the DW_ DIM_ADMISSIONS' PK;
- patient's id, PatientId, and the respective FK, PatientKey, references the DW_DIM_PATIENTS's PK;
- first and last care unit identifiers which the patient was cared for, FirstCareUnitId and LastCareUnitId, and the respective foreign keys, FirstCareUnitKey and LastCareUnitKey, that link to the DW_DIM_D_CARE_UNITS' primary key.

When an incorrect patient or admission characterizes an ICU stay, the respective record is closed. It is inserted a new record, with the same IcustayId, with the updated PatientId or AdmissionId.

DW_FACT_SERVICES

While a patient can be physically located at a given ICU, they are not necessarily being cared for by the same ICU team due to, for example, bed shortage. The DW_FACT_SERVICES table identifies the type of service a patient is receiving in the hospital. A patient's service is characterized by its identifier (PatientId), admission (AdmissioId), current service type (CurrServiceId), previous service type if applied (PrevServiceId), time of service transfer (ServiceToStart), and the ActiveServiceFlag. This binary integer flag is only positive for the last service that a patient was admitted. The fact table, shown in Figure 3.1, has the foreign keys PatientKey, AdmissionKey, CurrServiceKey and LastServiceKey, referencing the dimensional tables DW_DIM_PATIENTS, DW_DIM_ADMISSIONS, and DW_DIM_D_CARE_UNITS, respectively.

Dimension table normalization is referred to as snowflaking (as seen in Section 2.3.2, which is an extension of the dimensional model. Redundant attributes are removed from the flat, denormalized dimension table and placed in separate normalized dimension tables [11].

When a dimension table is normalized, low-cardinality attributes appear as secondary tables connected to the base dimension table by a key. Although the snowflake represents hierarchical data accurately, it is difficult for business users to understand and explore snowflakes. They may reduce the disk space consumed by dimension tables, but the savings are usually insignificant compared to the entire data warehouse environment and negative impact in query performance. In the case of Figure 3.1, the DW_FACT_SERVICES table could be connected to the admissions table, and the patients' table relationship removed since it is already referenced in the admissions table. Nonetheless, to increase query performance, both relationships were kept.

There are situations in which it is permissible to build an outrigger dimension that attaches to a dimension within the fact table's immediate halo, as illustrated in Figure 3.1. Outriggers are dimension tables joined to other dimension tables, but they distinguish themselves from the fully normalized snowflakes because it is one layer removed from the fact table [11].

In this example, the outrigger is two date dimensions snowflaked off a primary dimension (DW_DIM_ADMISSIONS). The outrigger date attributes are descriptively and uniquely labeled to distinguish the dates associated with the admission and discharge process, making it possible to simultaneously filter these dates.

DW_DIM_D_ITEMS_CATEGORY

DW_DIM_D_ITEMS_CATEGORY_CHART, DW_DIM_D_ITEMS_CATEGORY_PROCEDURE, and DW_DIM_D_ITEMS_CATEGORY_LABITEMS are similar structured dimensional tables that provide information about the data category of the patient's nurse-verified physiological measurements, medical interventions, and laboratory test results, respectively.

These dictionary tables are described by a unique integer identifier, ItemCategoryId, and the respective label, ItemCategoryLabel, and a binary flag, FlagActive, designates whether the item category is discarded. The StartDate and EndDate represent the active period of the item category.

Moreover, the DW_DIM_D_ITEMS_CATEGORY_CHART dimensional table also has a descriptive item category attribute, ItemCategoryDescription, that gives more detailed information about the patient's charted observations category.

DW_DIM_D_ITEMS_PROCEDURE, DW_DIM_D_ITEMS_CHART, and DW_DIM_D_LABITEMS define all items from the CareVue and Metavision ICU databases regarding clinical procedures (e.g., X-Ray, electroencephalogram, or dialysis), electronic charted nurse-verified measurements such as vital signs and laboratory measurements, respectively.

These dimensional tables are characterized by an integer identifier (ItemId or LabitemId), the item nomenclature (ItemLabel or LabitemLabel), the category identifier (ItemCategoryId or

LabitemCategoruKey), and the respective foreign key (ItemCategoryKey or LabitemCategoryKey) referencing the respective category items tables.

When an item changes category, the respective record is closed, and it is created a new record, with the same ItemId or LabitemId.

DW_FACT_CHARTEVENTS

The DW_FACT_CHARTEVENTS table registers all the events occurring on a patient chart (see Figure 3.2). A patient's chart event is characterized by its identifier (PatientId), admission (AdmissioId), intensive care unit stay (IcustayId), assigned caregiver (CaregiverId), item administrated (ItemId) and the corresponding result (ItemValue) and measurement unit (ItemUom), date and time of the observation, PatientDoStart, and PatientToStart, respectively.

This fact table has the foreign keys PatientKey, AdmissionKey, IcustayKey, CaregiverKey, ItemKey, PatiendDoStart, referencing the dimensional tables DW_DIM_PATIENTS, DW_DIM_ADMISSIONS, DW_DIM_ICUSTAYS, DW_DIM_CAREGIVERS, DW_DIM_D_ITEMS_CHART, and DW_DIM_DATE_EVENTS, respectively.

DW_FACT_PROCEDUREEVENTS

The DW_FACT_PROCEDUREEVENTS table lists all the events related to the patient's medical interventions. The structure of this table is similar to DW_FACT_CHARTEVENTS, as shown in Figure 3.3.

DW_DIM_D_ITEMS_MICROBIOLOGY

DW_DIM_D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM, DW_DIM_D_ITEMS_MICROBIOLOGY_SPECIMEN, and DW_DIM_D_ITEMS_MICROBIOLOGY_ORGANISM dimensional tables are dictionaries listing antibacterial antibiotics, type of samples (e.g., blood, pleural fluid, urine), and organisms (e.g., virus, bacteria, fungi), respectively. Each of these tables contains the attribute FlagActive, designating whether the item is discontinued.

The disk diffusion test or Kirby–Bauer test is an antibiotic susceptibility analysis to determine a pathogenic bacteria's sensitivity or resistance against a range of antimicrobial compounds. In this test, paper disks impregnated with antibiotics are placed on an agar plate to test the antibiotic efficiency to inhibit the organism's growth, assisting the physician in selecting treatment options for the patients' bacteria [45].

Fisher Scientific is an American distributor of scientific instrumentation, reagents and consumables, and software and services to healthcare and industry [46]. This company has a public online catalog of sensitive antibiotic disks produced by a global medical technology company BD [47].

Unlike the specimen and organism tables, the antibacterial one has the following attributes: antibiotic disk concentration (ItemConcentration), package's number of disks (ItemQuantity), and package price (ItemPrice). Note that all of the listed antibiotic prices refer to the current market values of the BD BBL^{TM} Sensi-Disc^{TM} brand available in the Fisher Scientific platform [47].

DW_FACT_MICROBIOLOGYEVENTS

DW_FACT_MICROBIOLOGYEVENTS comprises microbiology information, including cultures acquired, administered tests, and associated sensitivities/results, as illustrated in Figure 3.4. Similarly to the other event tables, a patient's microbiology event is described by its identifier (PatientId), admission (AdmissionId), specimen collected (SpecimenId), tested organism (OrganismId), and antibiotic (AntibioticId) and the corresponding result (AntibioticTestResult).

This fact table has the foreign keys PatientKey, AdmissionKey, SpecimenKey, OrganismKey, AntibioticKey and PatiendDoStart, referencing the dimensional tables DW_DIM_PATIENTS, DW_DIM_ADMISSIONS, DW_DIM_D_ITEMS_MICROBIOLOGY_SPECIMEN, DW_DIM_D_ITEMS_MICROBIOLOGY_ORGANISM, DW_DIM_D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM, and DW_DIM_DATE_EVENTS, respectively.

DW_FACT_LABEVENTS

The DW_FACT_LABEVENTS table registers all the patient's laboratory tests (see Figure 3.5), including out patient data (meaning do not have an admission at the Hospital). A patient's laboratory test is characterized by its identifier (PatientId), admission (AdmissioId), intensive care unit stay (IcustayId), assigned caregiver (CaregiverId), item tested (ItemId) and the corresponding result (ItemValue) and measurement unit (ItemUom), date and time of the observation, PatientDoStart, and PatientToStart, respectively.

This fact table has the foreign keys PatientKey, AdmissionKey, ItemKey, PatiendDoStart, referencing the dimensional tables DW_DIM_PATIENTS, DW_DIM_ADMISSIONS, DW_DIM_D_LABITEMS, and DW_DIM_DATE_EVENTS, respectively.

3.6 Summary

MIMIC (Medical Information Mart for Intensive Care) integrates deidentified, comprehensive clinical data of patients admitted, from 2001 to 2012, in the Beth Israel Deaconess Medical Center, Boston. This Chapter gives a detailed description of the this database's data, access and limitations.

The Clinical Database resulted from the ETL (Extract, Transform and Load) process that used the MIMIC-III demo v.1.4 as the data source. The Clinical Database is composed by nineteen dimension tables and five fact tables divided into five data marts: hospital admission services, electronic charted measurements, medical interventions, microbiology and laboratory tests. It is also presented this Clinical Solution's relational model, as well as, a detailed description of each table's attributes.

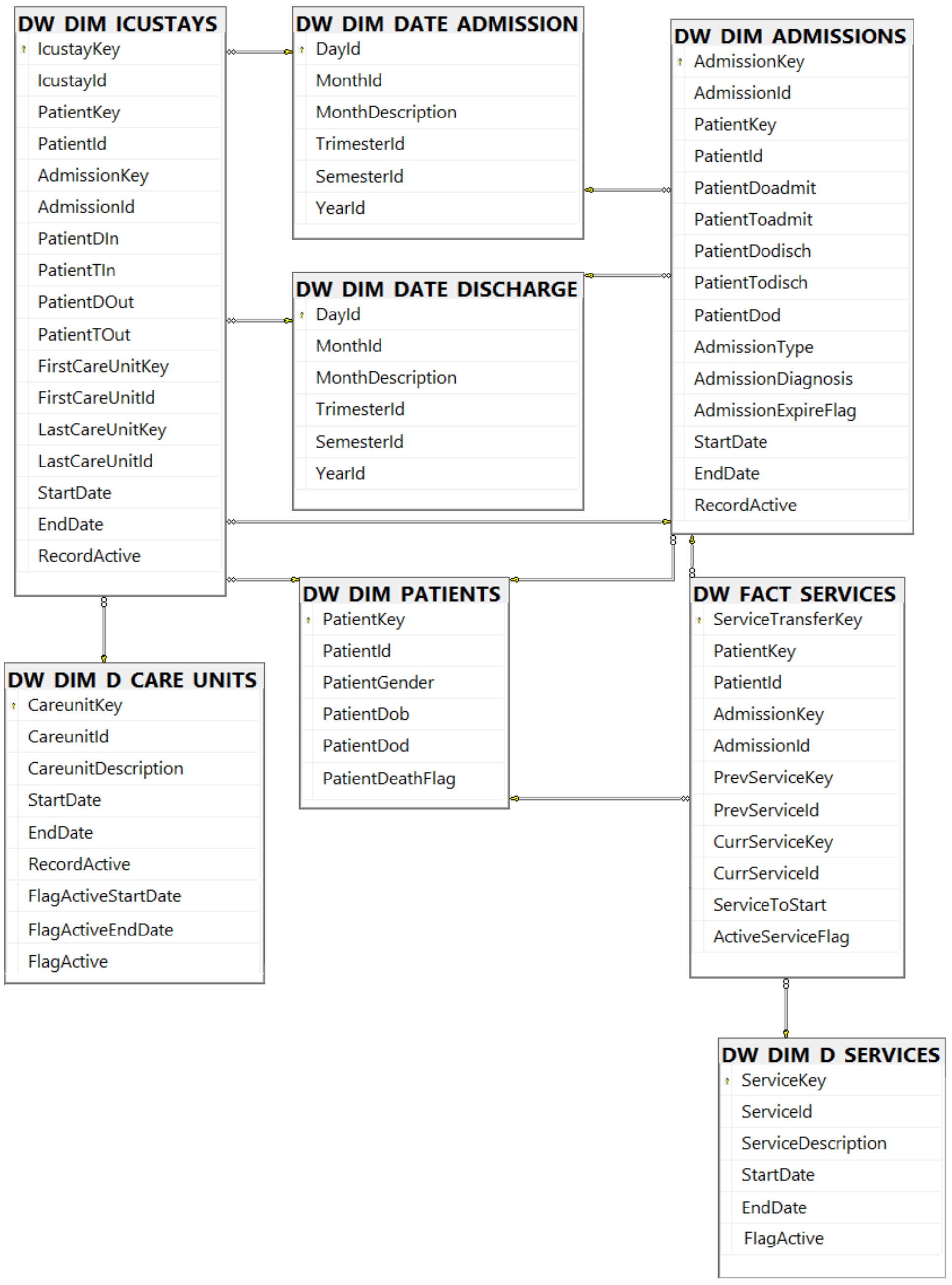


Figure 3.1: Overview of the Hospital Admission Services Data Mart Relational Model

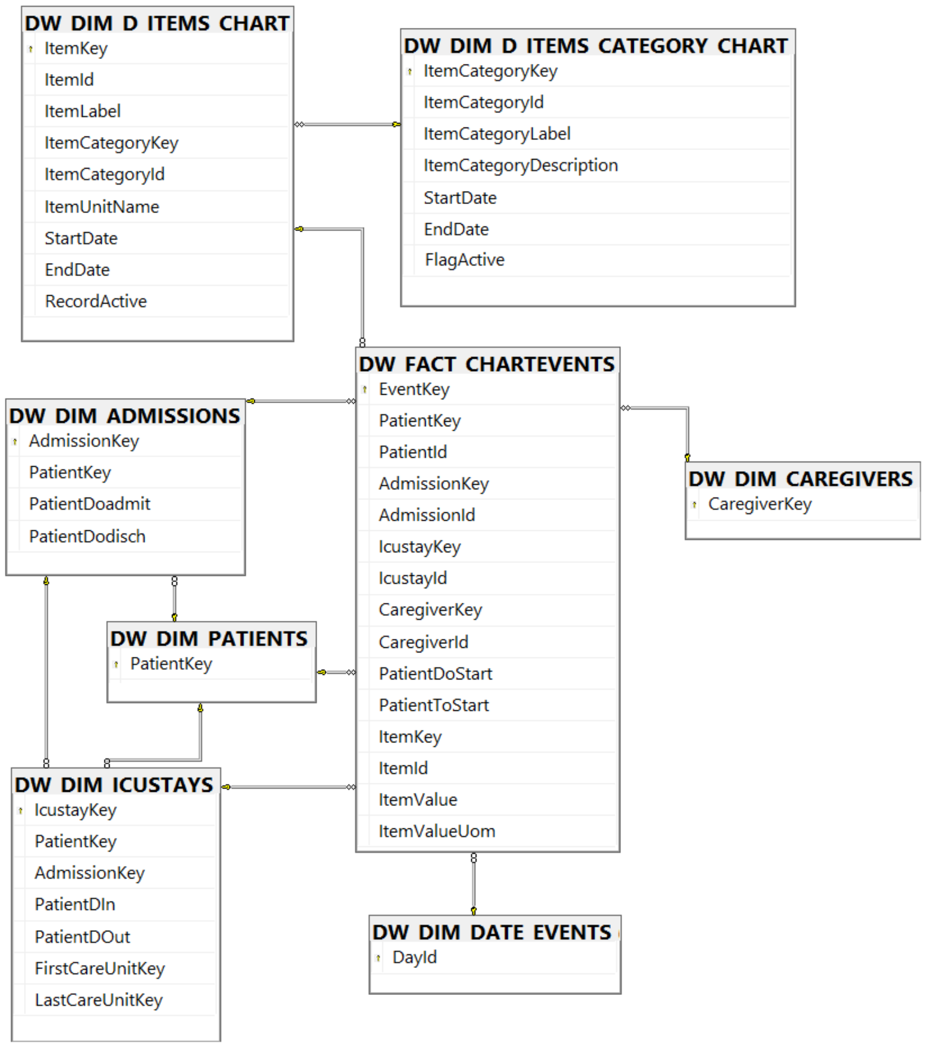


Figure 3.2: Overview of the Charted Measurements Data Mart Relational Model

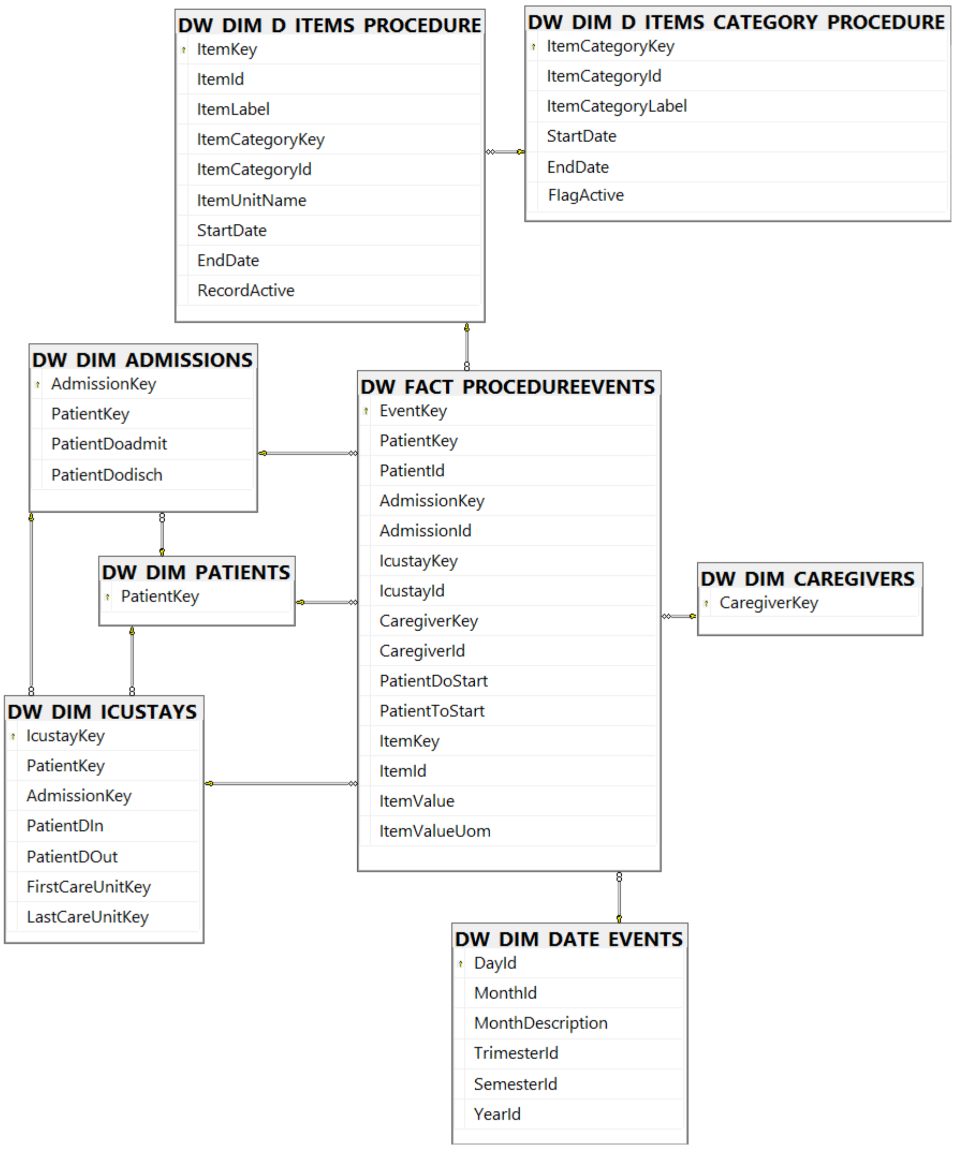


Figure 3.3: Overview of the Medical Interventions Data Mart Relational Model

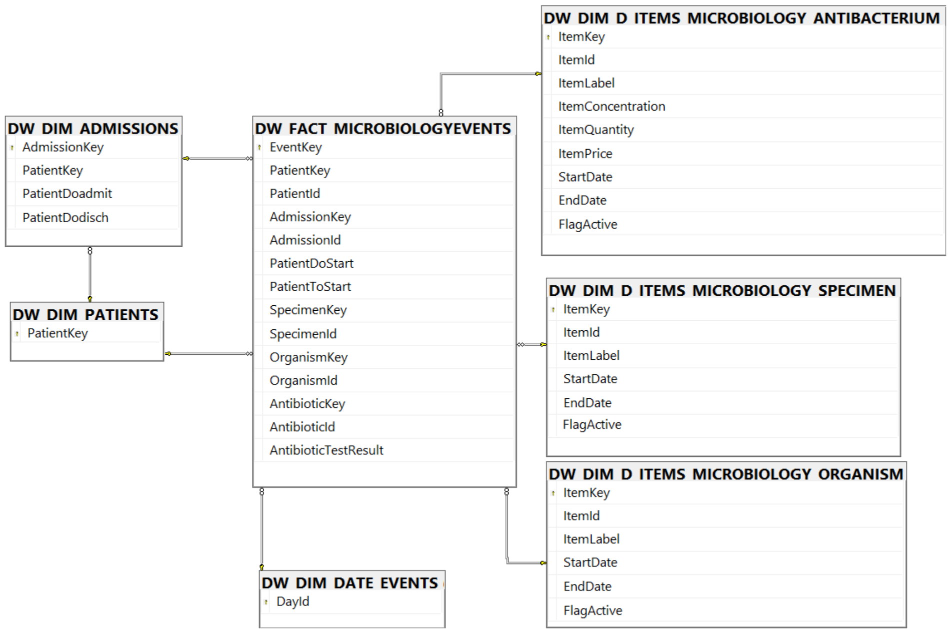


Figure 3.4: Overview of the Microbiology Tests Data Mart Relational Model

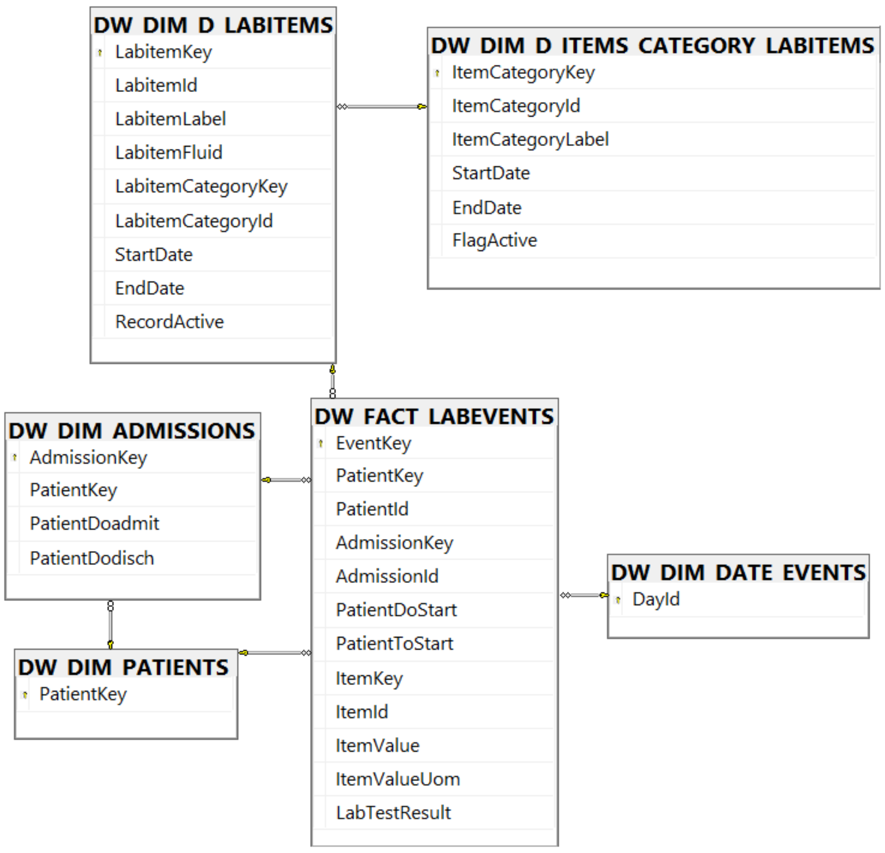


Figure 3.5: Overview of the Laboratory Tests Data Mart Relational Model

Chapter 4

Business Intelligence Architecture

This chapter dives into the Business Intelligence Architecture construction process. Firstly, there is the definition of the local and Microsoft Business Intelligence tools used to built the BI architecture in question. Then, the Clinical Extraction, Transformation and Loading steps to organize the Clinical database, both on-premises and cloud environments, is described.

4.1 Introduction

The increase in clinical records produced every day and the absence of analytical tools to handle all the information produced make it imperative to use Information Technologies (ITs) in the health sector, to support the decision-making process and reduce unpropitious events in a clinical setting.

The adoption of health information technologies is seen as an opportunity to improve not only the effectiveness, efficiency and quality of health services, but also to provide transparency about economic activities and the availability of information [2]. Healthcare organizations face a common problem with the high amount of gathered data from several relational tables, which normally are disorganized and unstructured, requiring computational time for data integration.

Despite the benefits inherent in the application of BI in Health Institutions, its adoption is not yet global and is lagging behind other sectors. Factors such as limited budgets, the absence of sponsors and qualified professionals, as well as data quality and interoperability issues between different information systems, have created difficulties in the implementation of BI solutions in health [48].

In the present context, the implementation of solutions based on the BI concept, at Beth Israel Deaconess Medical Center, promotes the creation of intensive care units' indicators and improves the clinical service provided through the representation of useful and interactive information. These solutions have the following advantages:

1. Organization and management of information from different sources, allowing relevant, agile, and transparent access to data;

2. Creation of reports, and clinical and administrative analysis in a fast and straightforward way;

3. Monitoring of hospital activities and processes to allow their improvement and increase user satisfaction;

4. Support and security of users' private data and control of information quality.

The data validation and importation from Database Management Systems (DBMS), the construction of mechanisms to automate procedures, and the creation of queries to consult indicators are tasks associated with the development of DW, subsequently allowing the representation of useful knowledge through the application of BI tools.

This chapter reflects the project's Business Intelligence process, which consists of the Extract, Transform and Load (ETL) of the data obtained by Beth Israel Deaconess Medical Center and the development of report and dashboards for viewing and analyzing medical information.

4.2 Business Intelligence Tools

The tools evaluation and selection follows the architecture definition since it is necessary first to define and understand what is necessary to accomplish. The tools selection focused on Microsoft products due to the existing partnership between Microsoft and B2F.

Figure 4.1 illustrates the tools selected for each stage of the on-premises and cloud BI's architecture, organized in two environments: back-end (development environment) and front-end (visualization environment).

In the back-end, the process begins with the collection of data, in CSV files format, from the MIMIC-III demo v.1.4 and subsequent extraction and transformation of the data to the dimensions and facts of the Staging Area and, finally, the loading of this information for the Data Warehouse (DW).

In this sense, SQL Server Management Studio (SSMS) was used to manage the Database Engine (DE), responsible for constructing the DW, where the dimension and fact tables are allocated. The DE used, both on-premises and in the cloud environment, was the Azure SQL Database.

On-premises, Visual Studio enabled the ETL process using Integration Services modules from the Microsoft SQL Server (SS). Each data extraction, transformation, and loading is built in one package using "Data Flow Task" and the "Execute SQL Task" tasks. The first one is responsible for moving and mapping data from a table or flat file (CSV file) source to the destination table using a Flat File and ADO.NET connection when sinking to the CSV File and the Microsoft Azure SQL Database, respectively. The second one is to run SQL statements to clean or modify data such as truncating or updating tables.

On the other hand, in the Microsoft Azure environment, the Azure Blob Storage and Data Factory are support resources to develop the ETL process. The Azure Blob Storage collects all the information of each CSV file, and the Data Factory facilitates the ETL construction using pipelines. Each pipeline is responsible for the extraction, transformation, or loading of one table, and contains a sequence of activity blocks that allow copying, mapping, and modifying data.

When there is a movement of data between two datasets, the 'Copy Data' activity connects to the table (Azure SQL Database service) or blob (Azure Blob Storage service) source and sinks and maps the columns to the destiny table. In this activity, it is possible to supplement with simple SQL scripts (e.g., truncate a table) or stored procedures (e.g., data transformation such as converting data types or semantic re-categorization of descriptive labels).

Using the Azure SQL Server in the cloud environment was sufficient to access the Azure SQL Database and run the Data Factory's ETL process. However, on-premises, it is mandatory to use two SQL Servers: Azure and the local machine (MSSQLSERVERMARGA) to access the database and run the developed ETL, respectively.

In the front-end, DW tables act as a source for the report and respective dashboards, providing the end user's optimized information, presented in the Power BI Desktop (PBID).

Once the on-premises tools (Visual Studio, SSMS and, Power BI Desktop) were selected, they were installed and tested on the prototype's local machine to ensure their operation.

Similarly, after creating a Microsoft Azure account, it was necessary to subscribe to the following resources: SQL Server, SQL database, Storage account, Data Factory (V2) and, Power BI.

In Section 2.7, the description of the tools incorporated in the prototype architecture is presented.

4.3 Extract, Transform and Load

The development of the necessary processes for the construction of the DW and the respective data marts modeled in the previous chapter depends on the execution of procedures and ETL functions, shown in the Figure 4.2. Note that all the tables resulting from the extraction, transformation, and loading have the prefixes "SA_CSV_"; "SA_DIM_" or "SA_FACT_"; "DW_DIM_" or "DW_FACT", respectively.

When building an ETL process is good practice to create two parameters tables: PARAMS_Control_Packages and PARAMS_Log_Packages. The first table contains all the created packages type (E – Extraction; T – Transformation; L – Loading), running order, creation and alteration dates and users name, and the binary flag RecordActive to indicate whether or not the package should be run. The Log table contains the running state information of all the run packages of an ETL project. Each row represents one package and includes the start and end running timestamp, a binary flag to indicate the ETL success or failure and the error message if applied.

In the Visual Studio, each package whether extracts, transforms or loads data from the source table/CSV file to the destiny table. However, the user running one package at a time is not an option. Therefore, it is created the "E_ALL", "T_ALL" and "L_ALL" packages that run all the extraction, transformation, and loading packages, respectively. The specific package running order is defined by the Control Table (see Table A.1 in Appendix). For example, the "L_ALL" is going

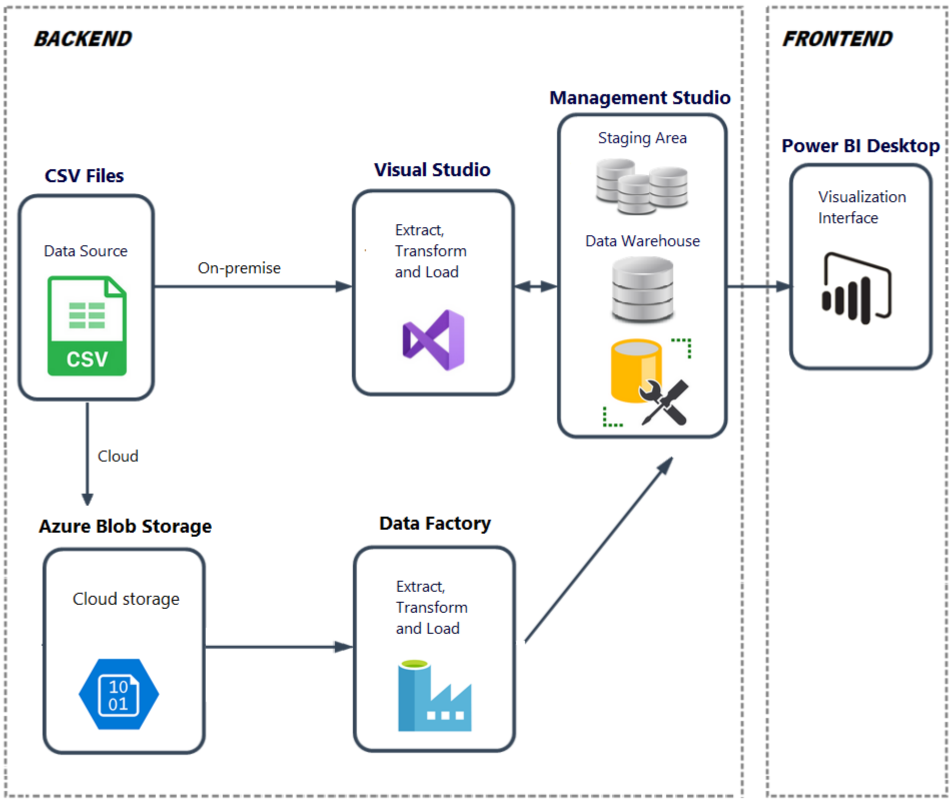


Figure 4.1: Clinical Business Intelligence Architecture

to load all the “L” type packages, starting with the dimensional tables first, followed by the fact tables. Finally, it is created the “ETL_ALL” package that runs, in this order, the “E_ALL”, “T_ ALL” and “L_ALL” packages.

On-premises, it was implemented a Clinical ETL with a total of 58 packages where the “ETL_ ALL” package runs sequentially the other 57 packages using a foreach loop.

Figure 4.3 represents the clinical ETL’s “ETL_ALL” package. The process begins with inserting a row into the PARAMS_Log_Packages with the “ETL_ALL” package name and start running time. Then, it will retrieve an array with all the packages that have a not null RecordActive and that the type “A” (“E_ALL”, “T_ALL” and “L_ALL” packages). In the foreach loop, it inserts a row into the PARAMS_Log_Packages with the “E_ALL” name and start running time, and executes this package. When the execution of the “E_ALL” is complete, it is updated the corresponding row with the end running time and with the error message(s) if it was encounter an error. Note that the “E_ALL” package execution is very similar to the one described before but instead of running the type “A” packages, it runs the type “E” packages. Finally the “ETL_ALL” loop repeats the process for the other two type “A” packages and updates the PARAMS_Log_Packages table’s “ETL_ALL” row. The two scripts “Send Mail on Failure” and “Send Mail on Success” are not components of the ETL process itself but have the purpose to monitor it. The monitoring process will be explained in Section 6.2.1.

On the other hand, in the Azure Data Factory, it was only implemented a short extraction

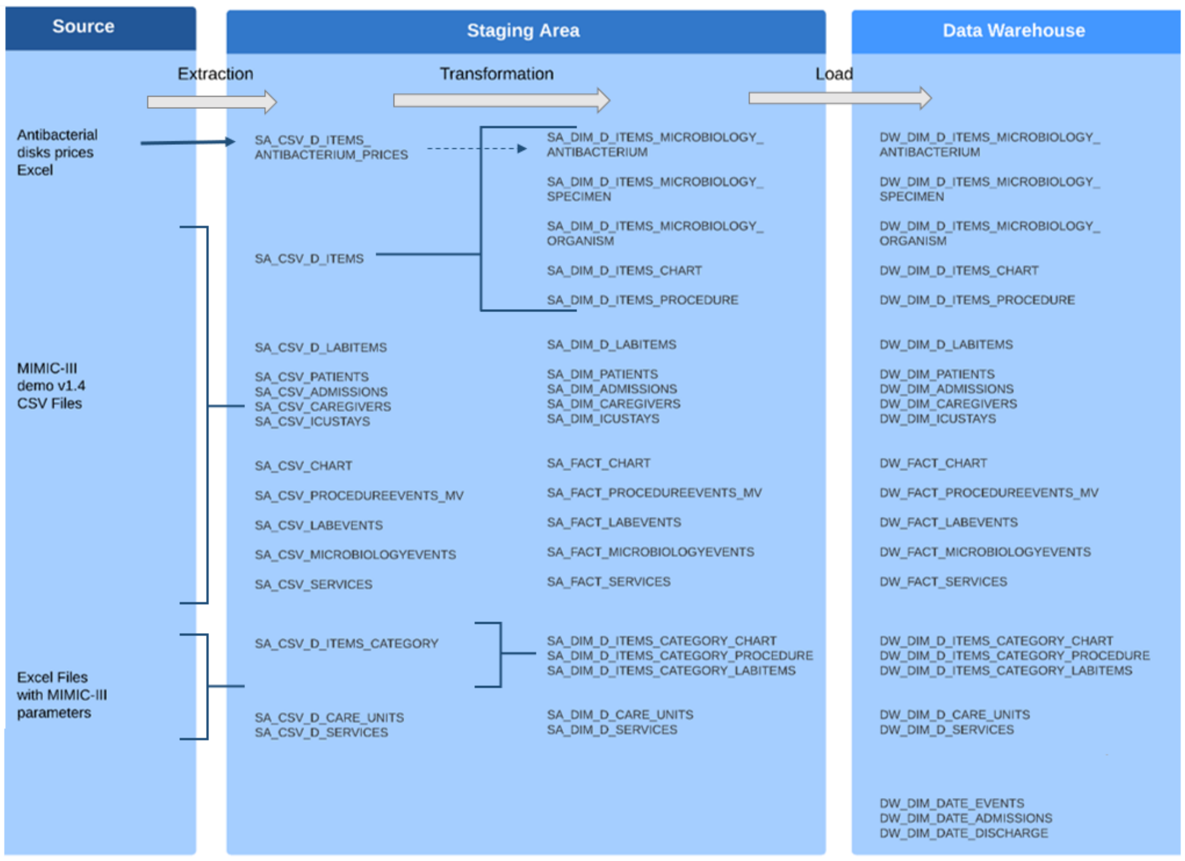


Figure 4.2: Clinical ETL procedures

pipeline which consisted into obtaining the listed antibacterial agents and their attributes from the Antibacterial disks prices excel and transferring this data into the SA_CSV_D_ITEMS_ANTIBACTERIUM_PRICES destiny table (see Figure 4.4). The two azure functions are not components of the ETL process itself but have the purpose to monitor it. The monitoring process will be explained in Section 6.2.1.

Since Antibacterial disks prices Excel was retrieved from Fisher Scientific platform [47] and not from the Hospital database, this information would be in a different server than the rest of the data. However, if it was necessary to replicate the exactly same clinical ETL in Azure, the implementation would be very similar with exception that it does not exist foreach loops in the Data Factory. The solution is to create a "ETL_ALL" pipeline with the sequenced "Execute Pipeline" activities to run multiple pipelines, in the desired order.

4.3.1 Extraction

In the information extraction phase, the following MIMIC-III tables were collected, in the form of CSV files:

- SA_CSV_ADMISSIONS
- SA_CSV_PATIENTS

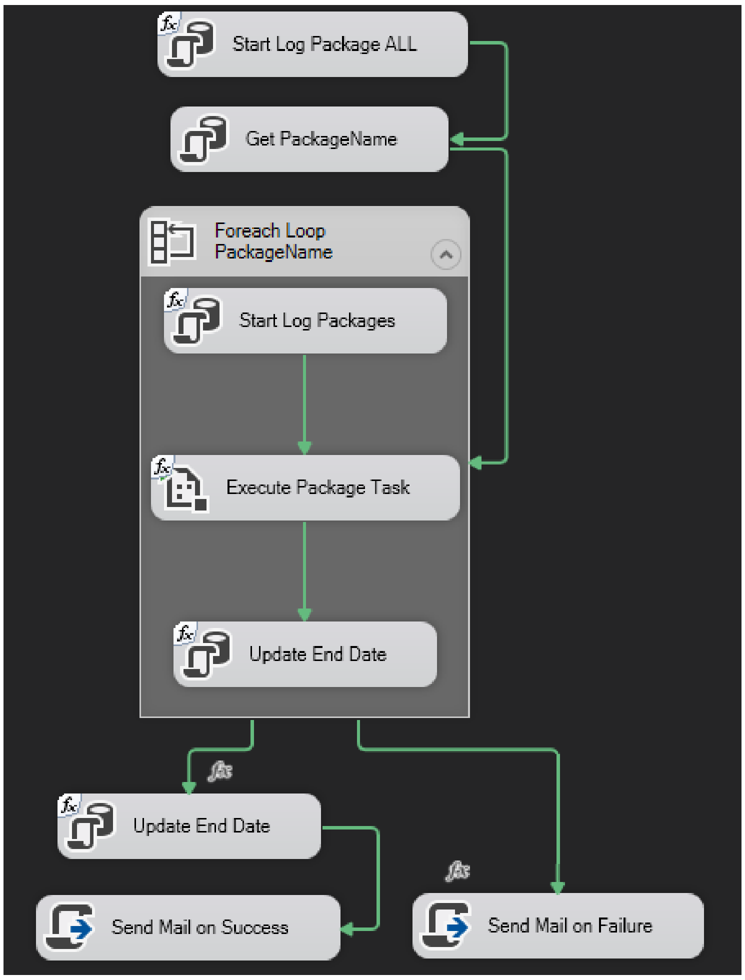


Figure 4.3: Clinical ETL's "ETL_ALL" package implemented in Visual Studio

- SA_CSV_CAREGIVERS
- SA_CSV_ICUSTAYS
- SA_CSV_SERVICES
- SA_CSV_D_ITEMS
- SA_CSV_D_LABITEMS
- SA_CSV_CHARTEVENTS
- SA_CSV_PROCEDUREEVENTS_MV
- SA_CSV_LABEVENTS
- SA_CSV_MICROBIOLOGYEVENTS

Furthermore, the Excel file (MIMIC_PARAMS) describes the microbiology concepts, the patient's nurse-verified physiological measurements, and medical intervention categories (SA_

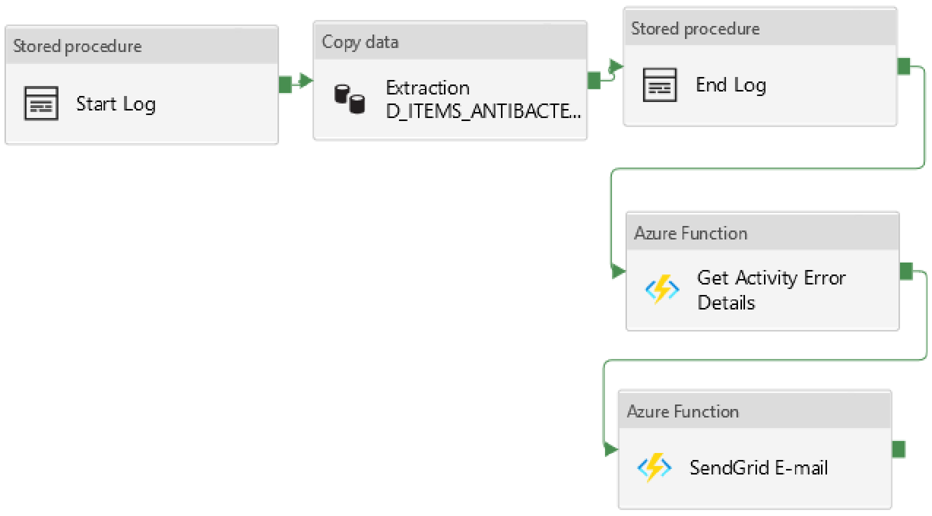


Figure 4.4: Antibacterial Disks Prices Extraction Pipeline implemented in Data Factory

CSV_D_ITEMS_CATEGORY), Critical Hospital Units (SA_CSV_D_CARE_UNITS), and Services (SA_CSV_D_SERVICES), generating three extraction new tables.

Finally, the listed antibacterial agents and their attributes (SA_CSV_D_ITEMS_ANTIBACTERIUM_PRICES) were collected from the Fisher Scientific platform [47].

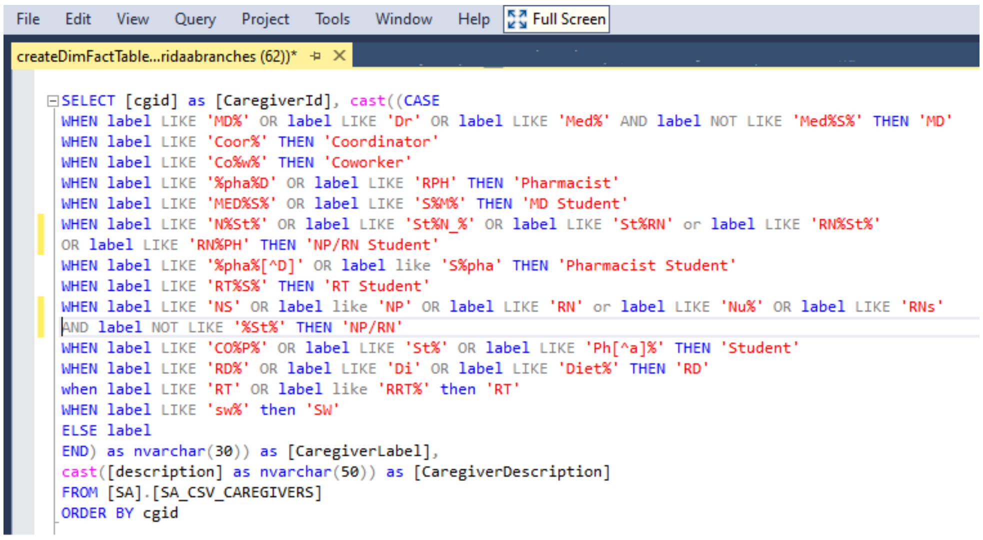
4.3.2 Transformation

The SA_CSV_D_ITEMS table has the columns "linksto" (e.g., microbiologyevents) and "category" (e.g., organism), allowing to filter the registered items by the associated fact table and category. Thus, originating the following five tables SA_DIM_D_ITEMS_CHART, SA_DIM_D_ITEMS_PROCEDURE, SA_DIM_D_ITEMS_MICROBIOLOGY_SPECIMEN, SA_DIM_D_ITEMS_MICROBIOLOGY_ORGANISM, SA_DIM_D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM. The last table resulted from the merging between the SA_CSV_D_ITEMS and SA_CSV_D_ITEMS_ANTIBACTERIUM_PRICES, using the antibiotic identifier code.

Similarly, the SA_CSV_D_ITEMS_CATEGORY column "linksto" filters the items' category by the linked fact tables, generating the tables SA_DIM_D_ITEMS_CATEGORY_CHART, SA_DIM_D_ITEMS_CATEGORY_PROCEDURE, and SA_DIM_D_ITEMS_CATEGORY_LABITEMS.

After assessing the quality of the attributes selected from the data sources provided by the MIMIC-III demo v.1.4, it was possible to identify the following inconsistencies: attributes with some null values, spelling errors, or identifier codes with different associated descriptors. Based on this assessment, proposals for corrective actions for attributes considered relevant to the final solution are presented in Table 4.1.

As noted in Table 4.1, the transforming actions consist of changing the data type, semantic recategorization of descriptive codes (see Figure 4.5), adding attributes directly linked to the existing



```sql
SELECT [cgid] as [CaregiverId], cast((CASE
WHEN label LIKE 'MD%' OR label LIKE 'Dr' OR label LIKE 'Med%' AND label NOT LIKE 'Med%S%' THEN 'MD'
WHEN label LIKE 'Coor%' THEN 'Coordinator'
WHEN label LIKE 'Co%w%' THEN 'Coworker'
WHEN label LIKE '%pha%D' OR label LIKE 'RPH' THEN 'Pharmacist'
WHEN label LIKE 'MED%S%' OR label LIKE 'S%M%' THEN 'MD Student'
WHEN label LIKE 'N%St%' OR label LIKE 'St%N_%' OR label LIKE 'St%RN' or label LIKE 'RN%St%'
OR label LIKE 'RN%PH' THEN 'NP/RN Student'
WHEN label LIKE '%pha%[^D]' OR label like 'S%pha' THEN 'Pharmacist Student'
WHEN label LIKE 'RT%S%' THEN 'RT Student'
WHEN label LIKE 'NS' OR label like 'NP' OR label LIKE 'RN' or label LIKE 'Nu%' OR label LIKE 'RNs'
AND label NOT LIKE '%St%' THEN 'NP/RN'
WHEN label LIKE 'CO%P%' OR label LIKE 'St%' OR label LIKE 'Ph[^a]%' THEN 'Student'
WHEN label LIKE 'RD%' OR label LIKE 'Di' OR label LIKE 'Diet%' THEN 'RD'
when label LIKE 'RT' OR label like 'RRT%' then 'RT'
WHEN label LIKE 'sw%' then 'SW'
ELSE label
END) as nvarchar(30)) as [CaregiverLabel],
cast([description] as nvarchar(50)) as [CaregiverDescription]
FROM [SA].[SA_CSV_CAREGIVERS]
ORDER BY cgid
```

Figure 4.5: Caregivers Table Transformation in the Staging Area

ones, and restructuring the dates to the ISO 8601 format "YYYYMMDD". All string data types were converted to nvarchar, storing Unicode characters of variable-length.

4.3.3 Loading

In all the tables, primary keys were automatically generated, which are referenced by other tables' foreign keys. Thus, at this stage, the connections of the relational model of the clinical database are built, as shown in Figures 3.1–3.5.

In this step, the parent dimensional tables are loaded first, followed by the child dimensional tables and fact tables accordingly to the foreign keys hierarchy.

Unlike the other tables, the date dimension was loaded through a stored procedure, which automatically generates the different attributes (year, month, description of the month, day, semester, trimester) of the date dimension when the solicited range of dates is provided.

When SA tables have new records that are not present in the respective DW tables, each row is loaded into the DW table using a LEFT JOIN.

Note that for each table's FK, the loading implies a LEFT JOIN between the SA table and the DW referenced parent table. For example, the PatientKey in the DW_DIM_ADMISSIONS results from the LEFT JOIN between the SA_DIM_ADMISSIONS and DW_DIM_PATIENTS (see Figure 4.6).

After inserting the new records into the DW tables, updating the dimensional tables' attributes is the next step. In this project, the implemented Slowing Changing Dimensions' types are 1, 2, and a combination of these two (see Figure 4.7).

The majority of the dimensional table's attributes are type 1, so they are overwritten when the ETL encounters a change. Three more columns are incorporated when there is at least one type 2 attributes in the dimensional tables: StartDate, EndDate, and RecordActive. StartDate and

Table 4.1: Summary of the transforming actions of the clinical database tables' attributes.

Table(s)	Attribute(s)	Transformation
PATIENTS	PatientDob	Convert the timestamp data type to the ISO 8601 'YYYYMMDD' date format.
	PatientDod	Convert the timestamp data type to the ISO 8601 'YYYYMMDD' date format.
CAREGIVERS	CaregiverLabel	Aggregate identical professionals label semantics (e.g., Med, Dr, and MD all refer to Medicine Doctor). It is assigned the value 'Unknown' to null caregivers identifiers.
ADMISSIONS	PatientDoadmit	New column. Convert the timestamp data type to the ISO 8601 'YYYYMMDD' date format.
	PatientDodisch	New column. Convert the timestamp data type to the ISO 8601 'YYYYMMDD' date format.
	PatientDod	Convert the timestamp data type to the ISO 8601 'YYYYMMDD' date format
ICUSTAYS	PatientDIn	New column. Convert the timestamp data type to the ISO 8601 'YYYYMMDD' date format.
	PatientDOut	New column. Convert the timestamp data type to the ISO 8601 'YYYYMMDD' date format.
	FirstCareUnitId, LastCareUnitId	It is assigned the value 'N/A' to null identifiers codes.
LABITEMS	LabitemFluid	Aggregate identical body fluid semantics (e.g., Cerebrospinal and CSF both refer to Cerebrospinal fluid (CSF))
D_ITEMS_CHART, D_ITEMS_PROCEDURE, D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM, D_ITEMS_MICROBIOLOGY_ORGANISM, D_ITEMS_MICROBIOLOGY_SPECIMEN, D_LABITEMS	ItemCategoryId	Convert the category name attribute from the extraction process to an integer identifier code (ItemCategoryId). This transformation results from a LEFT JOIN between the SA_CSV_D_ITEMS_ and SA_DIM_D_ITEMS_CATEGORY_ tables. Additionally, it is necessary to aggregate similar category names that correspond to the same category using SQL CASE statements. It is assigned the value -1 to the items that don't belong to a category.
SERVICES	PatientId, AdmissionId	It is assigned the value -1 to null identifiers codes.
	CurrServiceId	It is assigned the value 'N/A' to null identifiers codes.
	ServiceDoStart	New column. Convert the timestamp data type to the ISO 8601 'YYYYMMDD' date format.
PROCEDUREEVENTS_MV, CHARTEVENTS	PatientId, AdmissionId, IcustayId, CaregiverId, ItemId	It is assigned the value -1 to null identifiers codes.
	PatientDoStart	New column. Convert the timestamp data type to the ISO 8601 'YYYYMMDD' date format.
MICROBIOLOGYEVENTS	PatientId, AdmissionId, SpecimenId, OrganismId, AntibioticId	It is assigned the value -1 to null identifiers codes.
LABEVENTS	PatientId, AdmissionId, ItemId	It is assigned the value -1 to null identifiers codes.

Table 4.2: SCD types of all dimensional tables' attributes.

Table	Attribute(s)	SCD
PATIENTS	PatientGender, PatientDob, PatientDod, PatientDeathFlag	Type 1
CAREGIVERS	CaregiverLabel	Type 2
	CaregiverDescription	Type 1
D_SERVICES	ServiceDescription	Type 1
D_CARE_UNITS	CareunitDescription	Type 1
ADMISSIONS	PatientDoadmit, PatientToadmit, PatientDodisch, PatientTodisch, PatientDod, AdmissionType, AdmissionDiagnosis, AdmissionExpireFlag	Type 1
	PatientId	Type 2
ICUSTAYS	PatientDIn, PatientTIn, PatientDOut, PatientTOut, FirstCareUnitId, LastCareUnitId	Type 1
	PatientId, AdmissionId	Type 2
D_ITEMS_CATEGORY_PROCEDURE	ItemCategoryLabel	Type 1
D_ITEMS_PROCEDURE	ItemLabel, ItemUnitName	Type 1
	ItemCategoryId	Type 2
D_ITEMS_CATEGORY_CHART	ItemCategoryLabel, ItemCategoryDescription	Type 1
D_ITEMS_CHART	ItemLabel, ItemUnitName	Type 1
	ItemCategoryId	Type 2
D_ITEMS_CATEGORY_LABITEMS	ItemCategoryLabel	Type 1
D_LABITEMS	LabitemLabel, LabitemFluid	Type 1
	ItemCategoryId	Type 2
D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM	IemId, ItemLabel, ItemConcentration, ItemQuantity, ItemPrice	Type 1
D_ITEMS_MICROBIOLOGY_ORGANISM	ItemLabel	Type 1
D_ITEMS_MICROBIOLOGY_SPECIMEN	ItemLabel	Type 1

EndDate identify the active record period of each record. Table 4.2 shows identifies the SCD types of all attributes and the respective dimensional tables.

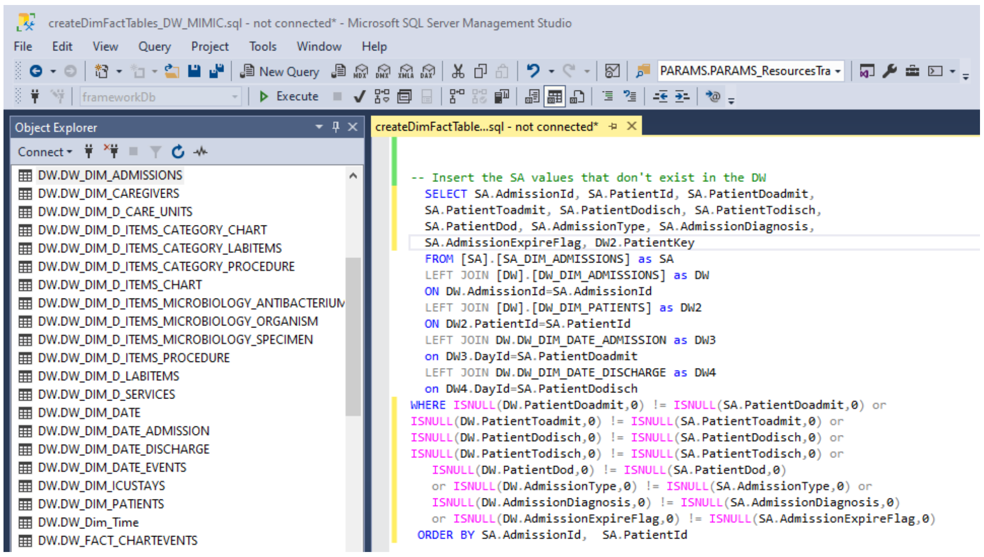
In all the dimensional tables, a record was added with primary key -1 and respective code -1 or 'N/A' depending on the type of variable (see Figure 4.8). Thus, it is guaranteed that if there is no correspondence between the identification codes (id) of the dimension and the fact tables that reference these keys, the fact table's foreign key is never null.

4.4 Report Visualization

Once the Power BI Desktop application is installed and launched, it is possible to connect to many different types of data sources, such as online services (Salesforce, Azure Blob Storage, Etc.), databases (SQL Server, Access, Amazon Redshift, Etc.), and files (Excel, JSON, Etc.). In this specific project, the selected data source was the SQL Server, which is compatible with Import and DirectQuery connections [49].

When building a visual within Power BI Desktop, with the DirectQuery connection, queries are sent to the underlying data source to retrieve the necessary data. The time taken to refresh the visual depends on the performance of the underlying data source.

On the other hand, with the Import connection, queries will be imported into the Power BI cache, and any changes to the underlying data are only displayed in the visuals after reimporting


createDimFactTables_DW_MIMIC.sql - not connected* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
PARAMS.PARAMS_ResourcesTra
frameworkDb
Execute
Object Explorer
Connect
DW.DW_DIM_ADMISSIONS
DW.DW_DIM_CAREGIVERS
DW.DW_DIM_D_CARE_UNITS
DW.DW_DIM_D_ITEMS_CATEGORY_CHART
DW.DW_DIM_D_ITEMS_CATEGORY_LABITEMS
DW.DW_DIM_D_ITEMS_CATEGORY_PROCEDURE
DW.DW_DIM_D_ITEMS_CHART
DW.DW_DIM_D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM
DW.DW_DIM_D_ITEMS_MICROBIOLOGY_ORGANISM
DW.DW_DIM_D_ITEMS_MICROBIOLOGY_SPECIMEN
DW.DW_DIM_D_ITEMS_PROCEDURE
DW.DW_DIM_D_LABITEMS
DW.DW_DIM_D_SERVICES
DW.DW_DIM_DATE
DW.DW_DIM_DATE_ADMISSION
DW.DW_DIM_DATE_DISCHARGE
DW.DW_DIM_DATE_EVENTS
DW.DW_DIM_ICUSTAYS
DW.DW_DIM_PATIENTS
DW.DW_Dim_Time
DW.DW_FACT_CHARTEVENTS
createDimFactTable...sql - not connected*

```sql
-- Insert the SA values that don't exist in the DW
  SELECT SA.AdmissionId, SA.PatientId, SA.PatientDoadmit,
  SA.PatientToadmit, SA.PatientDodisch, SA.PatientTodisch,
  SA.PatientDod, SA.AdmissionType, SA.AdmissionDiagnosis,
  SA.AdmissionExpireFlag, DW2.PatientKey
  FROM [SA].[SA_DIM_ADMISSIONS] as SA
  LEFT JOIN [DW].[DW_DIM_ADMISSIONS] as DW
  ON DW.AdmissionId=SA.AdmissionId
  LEFT JOIN [DW].[DW_DIM_PATIENTS] as DW2
  ON DW2.PatientId=SA.PatientId
  LEFT JOIN DW.DW_DIM_DATE_ADMISSION as DW3
  on DW3.DayId=SA.PatientDoadmit
  LEFT JOIN DW.DW_DIM_DATE_DISCHARGE as DW4
  on DW4.DayId=SA.PatientDodisch
WHERE ISNULL(DW.PatientDoadmit,0) != ISNULL(SA.PatientDoadmit,0) or
ISNULL(DW.PatientToadmit,0) != ISNULL(SA.PatientToadmit,0) or
ISNULL(DW.PatientDodisch,0) != ISNULL(SA.PatientDodisch,0) or
ISNULL(DW.PatientTodisch,0) != ISNULL(SA.PatientTodisch,0) or
   ISNULL(DW.PatientDod,0) != ISNULL(SA.PatientDod,0)
   or ISNULL(DW.AdmissionType,0) != ISNULL(SA.AdmissionType,0) or
   ISNULL(DW.AdmissionDiagnosis,0) != ISNULL(SA.AdmissionDiagnosis,0)
   or ISNULL(DW.AdmissionExpireFlag,0) != ISNULL(SA.AdmissionExpireFlag,0)
 ORDER BY SA.AdmissionId,  SA.PatientId
```

Figure 4.6: Loading new records from the Staging Area to the Data Warehouse Admissions Table

the data through the refresh button. Models with imported data can be refreshed at most once per hour while using DirectQuery always shows the latest data in the source (data automatically refreshes every 15 minutes) [50].

There are also limitations in the data transformations that can be applied within Query Editor. With imported data, a sophisticated set of transformations can easily be applied to clean and reshape the data before using it to create visuals, such as pivoting data from a column to a row. Those transformations are more limited in the DirectQuery connection [50]. In this dissertation, all the data transformations were applied during the ETL phase, so this limited feature is not an obstacle.

If the data is extensive, using the Import connection would not be feasible, but DirectQuery, by contrast, requires no massive data transfer due to it is queried in place.

When importing data, every date/datetime column will also have a built-in date hierarchy available by default so that the user can choose the appropriate level (year, month, day) of information. The built-in date hierarchy is not available when using DirectQuery; however, the loading of hierarchy date dimensions overcomes this restriction.

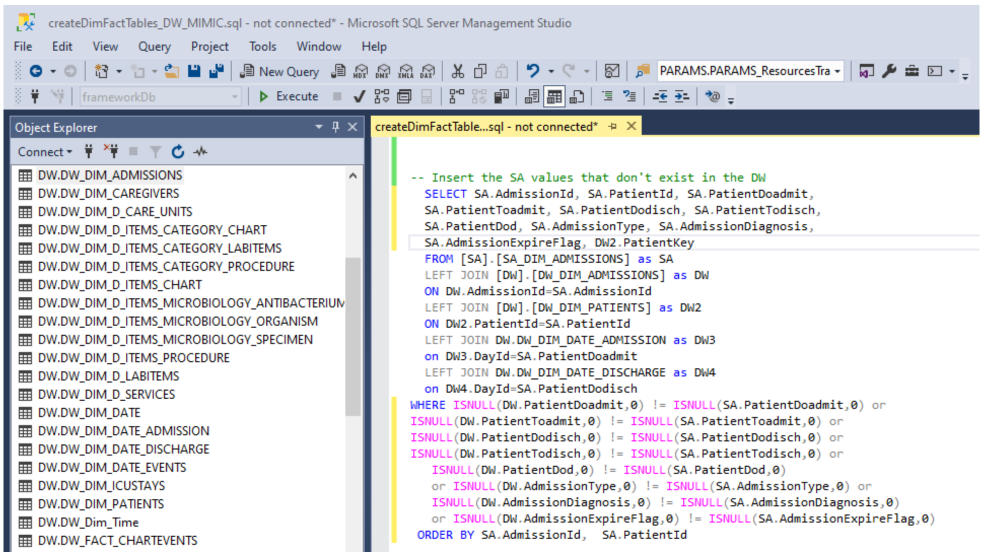
Considering the clinical scenario where the data is massive, frequently changing, and near real-time reporting is needed, the DirectQuery connection was selected to create the medical dashboards.

The Power BI Desktop has the following three main tabs:

Relationships – visualize and edit the select database tables and relational model;

Report – provide a dynamic and user-friendly interface for data visualization without using any coding language or have specific technical knowledge;

Data – scrutinize the database tables' data.


createDimFactTables_DW_MIMIC.sql - not connected* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
PARAMS.PARAMS_ResourcesTra
frameworkDb
Execute
Object Explorer
Connect
DW.DW_DIM_ADMISSIONS
DW.DW_DIM_CAREGIVERS
DW.DW_DIM_D_CARE_UNITS
DW.DW_DIM_D_ITEMS_CATEGORY_CHART
DW.DW_DIM_D_ITEMS_CATEGORY_LABITEMS
DW.DW_DIM_D_ITEMS_CATEGORY_PROCEDURE
DW.DW_DIM_D_ITEMS_CHART
DW.DW_DIM_D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM
DW.DW_DIM_D_ITEMS_MICROBIOLOGY_ORGANISM
DW.DW_DIM_D_ITEMS_MICROBIOLOGY_SPECIMEN
DW.DW_DIM_D_ITEMS_PROCEDURE
DW.DW_DIM_D_LABITEMS
DW.DW_DIM_D_SERVICES
DW.DW_DIM_DATE
DW.DW_DIM_DATE_ADMISSION
DW.DW_DIM_DATE_DISCHARGE
DW.DW_DIM_DATE_EVENTS
DW.DW_DIM_ICUSTAYS
DW.DW_DIM_PATIENTS
DW.DW_Dim_Time
DW.DW_FACT_CHARTEVENTS
createDimFactTable...sql - not connected*

```sql
-- Insert the SA values that don't exist in the DW
  SELECT SA.AdmissionId, SA.PatientId, SA.PatientDoadmit,
  SA.PatientToadmit, SA.PatientDodisch, SA.PatientTodisch,
  SA.PatientDod, SA.AdmissionType, SA.AdmissionDiagnosis,
  SA.AdmissionExpireFlag, DW2.PatientKey
  FROM [SA].[SA_DIM_ADMISSIONS] as SA
  LEFT JOIN [DW].[DW_DIM_ADMISSIONS] as DW
  ON DW.AdmissionId=SA.AdmissionId
  LEFT JOIN [DW].[DW_DIM_PATIENTS] as DW2
  ON DW2.PatientId=SA.PatientId
  LEFT JOIN DW.DW_DIM_DATE_ADMISSION as DW3
  on DW3.DayId=SA.PatientDoadmit
  LEFT JOIN DW.DW_DIM_DATE_DISCHARGE as DW4
  on DW4.DayId=SA.PatientDodisch
WHERE ISNULL(DW.PatientDoadmit,0) != ISNULL(SA.PatientDoadmit,0) or
ISNULL(DW.PatientToadmit,0) != ISNULL(SA.PatientToadmit,0) or
ISNULL(DW.PatientDodisch,0) != ISNULL(SA.PatientDodisch,0) or
ISNULL(DW.PatientTodisch,0) != ISNULL(SA.PatientTodisch,0) or
   ISNULL(DW.PatientDod,0) != ISNULL(SA.PatientDod,0)
   or ISNULL(DW.AdmissionType,0) != ISNULL(SA.AdmissionType,0) or
   ISNULL(DW.AdmissionDiagnosis,0) != ISNULL(SA.AdmissionDiagnosis,0)
   or ISNULL(DW.AdmissionExpireFlag,0) != ISNULL(SA.AdmissionExpireFlag,0)
 ORDER BY SA.AdmissionId,  SA.PatientId
```

Figure 4.7: Updating the Admissions Table's attributes using a combination of type 1 and 2 SCD

4.4.1 Power BI Desktop Report Features

In the report tab, there are four panes: fields, visualizations, bookmarks, and filters. The fields pane displays all the database tables and respective attributes and calculated metrics. The user selects the chart type (e.g., bar, column, line, gauge, doughnut, funnel, scatter, pie, and maps) in the visualizations pane and drags the fields pane's desired columns.

The bookmarks allow the user to save the currently configured view of a report page, including filtering and the visuals' state, to later facilitate the reports' navigation.

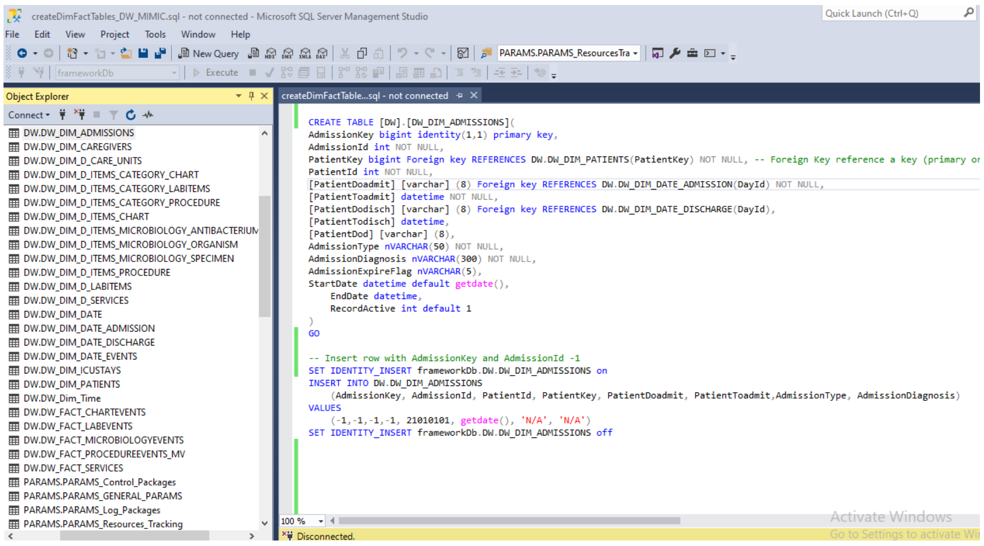
There are three levels of filters in Power BI:

Report-level filters – affect all of the data in the report and act as universal filters.

Page-level filters – only filter the data on a given page, making them useful for creating pages that focus on particular data subsets. For example, create page-level filters to make a page focus solely on a specific medical area. Page-level filters operate within the context of the report-level filters, which means that a page-level filter cannot override a report-level filter and can not be programmed to filter the data on other pages.

Visual-level filters – only filter the data on a given visual, whether it is a table, chart, or card. These are the most granular filters, and they operate within the context of both the page-level and report-level filters, so visual-level filters cannot override them or be programmed to filter data on other visuals.

Power BI reports use features called "slicers" which are objects (e.g., checkbox lists, dropdown lists, buttons, and sliders) embedded in the report body that interactively "slice and dice" the data. When applying a filter, it is impossible to select what variables of the report will be affected: a report-level filter will invariably filter all the data in the report, and a page-level filter will do the


createDimFactTables_DW_MIMIC.sql - not connected - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Project Tools Window Help
New Query
PARAMS.PARAMS_ResourcesTra
frameworkDb
Execute
Object Explorer
Connect
DW.DW_DIM_ADMISSIONS
DW.DW_DIM_CAREGIVERS
DW.DW_DIM_D_CARE_UNITS
DW.DW_DIM_D_ITEMS_CATEGORY_CHART
DW.DW_DIM_D_ITEMS_CATEGORY_LABITEMS
DW.DW_DIM_D_ITEMS_CATEGORY_PROCEDURE
DW.DW_DIM_D_ITEMS_CHART
DW.DW_DIM_D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM
DW.DW_DIM_D_ITEMS_MICROBIOLOGY_ORGANISM
DW.DW_DIM_D_ITEMS_MICROBIOLOGY_SPECIMEN
DW.DW_DIM_D_ITEMS_PROCEDURE
DW.DW_DIM_D_LABITEMS
DW.DW_DIM_D_SERVICES
DW.DW_DIM_DATE
DW.DW_DIM_DATE_ADMISSION
DW.DW_DIM_DATE_DISCHARGE
DW.DW_DIM_DATE_EVENTS
DW.DW_DIM_ICUSTAYS
DW.DW_DIM_PATIENTS
DW.DW_Dim_Time
DW.DW_FACT_CHARTEVENTS
DW.DW_FACT_LABEVENTS
DW.DW_FACT_MICROBIOLOGYEVENTS
DW.DW_FACT_PROCEDUREEVENTS_MV
DW.DW_FACT_SERVICES
PARAMS.PARAMS_Control_Packages
PARAMS.PARAMS_GENERAL_PARAMS
PARAMS.PARAMS_Log_Packages
PARAMS.PARAMS_Resources_Tracking
createDimFactTable...sql - not connected
100 %
Disconnected.
Activate Windows
Go to Settings to activate Wi

```sql
CREATE TABLE [DW].[DW_DIM_ADMISSIONS](
AdmissionKey bigint identity(1,1) primary key,
AdmissionId int NOT NULL,
PatientKey bigint Foreign key REFERENCES DW.DW_DIM_PATIENTS(PatientKey) NOT NULL, -- Foreign Key reference a key (primary or
PatientId int NOT NULL,
[PatientDoadmit] [varchar] (8) Foreign key REFERENCES DW.DW_DIM_DATE_ADMISSION(DayId) NOT NULL,
[PatientToadmit] datetime NOT NULL,
[PatientDodisch] [varchar] (8) Foreign key REFERENCES DW.DW_DIM_DATE_DISCHARGE(DayId),
[PatientTodisch] datetime,
[PatientDod] [varchar] (8),
AdmissionType nVARCHAR(50) NOT NULL,
AdmissionDiagnosis nVARCHAR(300) NOT NULL,
AdmissionExpireFlag nVARCHAR(5),
StartDate datetime default getdate(),
    EndDate datetime,
    RecordActive int default 1
)
GO

-- Insert row with AdmissionKey and AdmissionId -1
SET IDENTITY_INSERT frameworkDb.DW.DW_DIM_ADMISSIONS on
INSERT INTO DW.DW_DIM_ADMISSIONS
    (AdmissionKey, AdmissionId, PatientId, PatientKey, PatientDoadmit, PatientToadmit,AdmissionType, AdmissionDiagnosis)
VALUES
    (-1,-1,-1,-1, 21010101, getdate(), 'N/A', 'N/A')
SET IDENTITY_INSERT frameworkDb.DW.DW_DIM_ADMISSIONS off
```

Figure 4.8: Creation of the Admissions Table in the Data Warehouse

same for the page. However, slicers are a bit more flexible because they can be programmed to affect particular objects on the page.

The next Chapter presents the developed Clinical Report and the respective Business Intelligence Indicators and dashboards.

4.5 Summary

In the present Chapter, the implementation of solutions based on the BI concept, at Beth Israel Deaconess Medical Center, promotes the creation of intensive care units' indicators and improves the clinical service provided through the representation of useful and dynamic information.

The first step of the Business Intelligence Architecture is the understand and select the BI Tools. In this dissertation it was used, on-premises, the Visual Studio, and, in the Microsoft Azure Environment, the Data Factory to build the Clinical ETL.

The Business Intelligence Architecture process begins with the collection of clinical de-identified data, in CSV files format, from the MIMIC-III demo v.1.4 and subsequent extraction, transformation, and loading (ETL) of the data to the dimensions and facts of the Clinical Data Warehouse (DW).

On-premises, it was implemented a Clinical ETL with a total of 58 packages. The extraction sources were 10 CSV files from the Beth Center Hospital, the Fisher Scientific Excel (contains the antibiotic disk prices) and an external parameters Excel File that describes medical categories, Critical Hospital Units and Services. The transforming actions consisted of changing the data type, semantic re-categorization of descriptive codes, adding attributes, and restructuring the dates to the ISO 8601 format "YYYYMMD". In the Azure Data Factory, it was only implemented a short extraction pipeline which consisted into obtaining the listed antibacterial disks prices.

In both environments it was necessary to manage the Clinical Data Warehouse, using the SQL Server Management Studio. In the front-end, DW tables act as a source for the report and respective dashboards, providing the end user's optimized information, presented in the Power BI Desktop (PBID).

Chapter 5

Clinical Report

This chapter presents the business indicators and the developed Clinical Solution that allows medical professionals to quickly and confidently make data-driven business decisions while saving valuable time and resources. To conclude the chapter, it is presented an analysis and discussion of the Clinical Report's four pages: general hospital admissions, medical procedures, electronic charted measurements, and microbiology and laboratory.

5.1 Introduction

Healthcare analytics solutions enable clinics, hospitals, caregivers, and managers to quickly and confidently make data-driven business decisions while saving valuable time and resources. These solutions commonly include hospital and department statistics, and information to analyze the bed availability, discharge rate, average waiting time, health professionals' efficiency rate, and revenue summary.

5.2 Business Indicators

The business indicators support the analyzes presented in the report when evaluating the hospital admission process' different components performance and the respective clinical procedures (e.g., X-Ray, electroencephalogram, or dialysis), electronic charted nurse-verified and laboratory measurements. The selected indicators are specific to the clinical database available data and needs.

In this project, it is crucial that the hospitals managers have useful healthcare analytics solutions to make informed clinical and business decisions such as evaluating which caregivers have a heavier work load or what microbiology tests have a higher costs.

Table 5.1 presents the 23 indicators developed for this project as well as their respective description.

Table A.2 presents DAX (Data Analysis eXpressions) codes elaborated.

Table 5.1: Business Indicators

Report Page	**Indicator**	**Description**
Admissions	Waiting Medical Assistance Time (AVG)	Average time (minutes) between the hospital admission and the first medical exam performed
	Waiting Medical Assistance Time (MIN)	Minimum time (minutes) recorded between the hospital admission and the first medical exam performed
	Waiting Medical Assistance Time (MAX)	Maximum time (minutes) recorded between the hospital admission and the first medical exam performed
	Hospitalization Time (AVG)	Average hospitalization time (days)
	Hospitalization Time (MIN)	Minimum hospitalization time (days)
	Hospitalization Time (MAX)	Maximum hospitalization time (days)
	Hospital Admissions by Service	Total of admissions per hospital service
	% Hospital Admission Deaths	Percentage of hospital deaths that occur during hospitalization
	% Hospital Admission Type	Percentage of hospital admission types
Procedures	% CategoryProcedures	Percentage of performed procedures by medical category (e.g., Imaging)
	% CategoryItemsProcedures	Percentage of performed procedures by a medical category's item (e.g., X-Ray)
	CareuintProcedures	Total of performed procedures by hospital intensive care unit
	TypeCaregiverProcedures	Total of performed procedures by hospital caregivers type (e.g., Nurse)
	CaregiverProcedures	Total of performed procedures by hospital caregiver
Electronic Chart	% CategoryChart	Percentage of charted observations by medical category (e.g., Routine Vital Signs)
	% CategoryItemsProcedures	Percentage of charted observations by a medical category's item (e.g., Heart Rate)
	CareuintChart	Total of charted observations by hospital intensive care unit
	TypeCaregiverChart	Total of charted observations by hospital caregivers type
	CaregiverChart	Total of charted observations by hospital caregiver
Microbiology	% CategorySpecimen	Percentage of microbiology tests by the specimen (e.g., Blood Culture)
	AntibioticPrice	Total antibacterial disks spending, in euros, by antibiotic (e.g., Penicillin)
Laboratory	% CategoryLaboratory	Percentage of executed laboratory tests by medical category (e.g., Hematology)
	% CategoryItemsLaboratory	Percentage of executed laboratory tests by a medical category's item (e.g., Glucose)

5.3 Clinical Dashboard

The visual solution built-in PBID has four pages: general hospital admissions page, procedures, chart, and microbiology & laboratory.

Note that the available database does not have enough data (only characterizes 100 patients and 129 hospital admissions) to conduct in-depth analyzes and retrieve reliable conclusions.

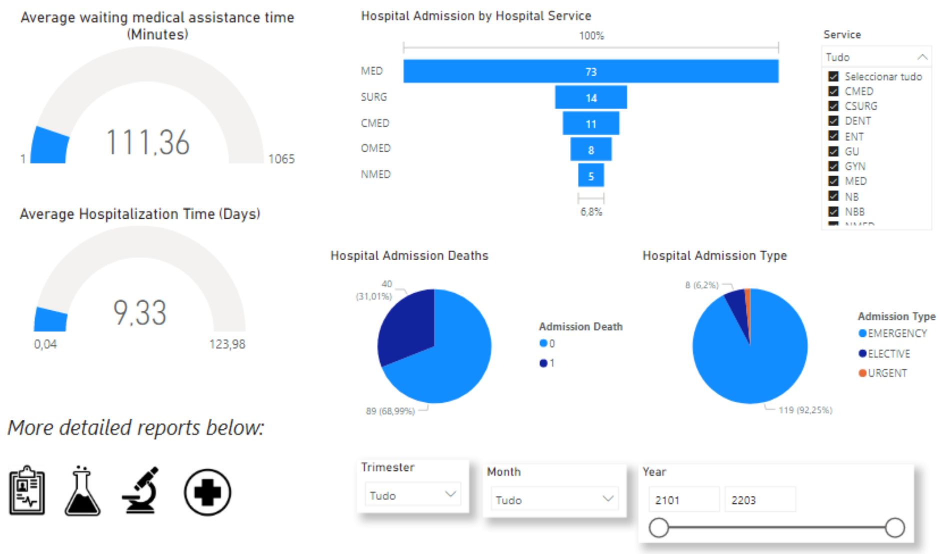


Figure 5.1: General Admissions Page Overview

Admissions Page

The admissions page (see Figure 5.1) includes four slicers (year, month, trimester, and service) and the following five graphs:

1. Average, minimum, and maximum waiting time (minutes) between the hospital admission and the first patient's medical exam;
2. Average, minimum, and maximum hospitalization time (days);
3. Total of admissions per hospital service;
4. Total of hospital deaths that occur during hospitalization (%);
5. Total of hospital admissions stratified by admission type (%).

In the "hospital admission type" pie chart (see Figure A.2 in Appendix), the 8 (6.2%) previously planned hospital admission – Selective Admissions – were assigned to the CSURG (Cardiac Surgery), TSURG (Thoracic Surgical), GU (Genitourinary), ORTHO (Orthopaedic Surgical), SURG (Surgical), and VSURG (Vascular Surgical) services and all the patients survived the surgeries.

All the patients represented in this report are deceased. Thus, the death flag in the hospital admission pie chart only signals the deaths that occurred within the hospital admission.

When analyzing the "hospital admission death" pie chart (see Figure A.1 in Appendix), it is possible to conclude that the 40 admissions that resulted in death within the hospital (30%) belonged to the MED (Medical General), SURG, CMED (Cardiac Medical), NMED (Neurologic Medic), OMED (Orthopaedic Medicine) and TRAUM (Trauma) services and were all in the context of Emergency or Urgent admissions.

Figure 5.1 shows the five services with the highest number of hospital admissions: MED (73), SURG (14), CMED (11), OMED (8), and NMED (5). The top five admission services represent 90% of the hospitalization admissions total (129), and the MED service describes more than 50% of this total. Note that each admission can be associated with more than one service since the patient can be transferred. In this visual element, it is only considered the most recent transfer service (ActiveFlag equals 1) and the valid current services, meaning the CurrServiceKey has to be different from -1.

In Figure 5.1, the hospitalization time gauge chart displays the average, minimum, and maximum of 9.33, 0.04, and 123.98 days, respectively.

Similarly, the waiting medical assistance time gauge chart shows that the average, minimum, and maximum time between the hospital admission and the first patient's medical exam are, respectively, 111.36, 1, and 1065 minutes.

Table 5.2 contains only the services that have more than one hospital admission.

The average waiting medical assistance time (WMAVG) of the CSURG (28.60 min), NSURG (63.60 min), and TRAUM (76.63 min) services are significantly lower than the global average (111.36 min). The medical assistance priority is higher in these services since they are associated with trauma and/or life-threatening surgeries. However, the low WMAVGs can also be related to the reduced number of hospital admissions (3–4), leading to bias conclusions.

On the other hand, the WMAVG of the OMED (304.33 min) and NMED (225.17 min) services are double the global average (111.36 min), which corroborates the expectation that non-surgical services have a lower medical assistance priority. These values can be inflated due to the low number of hospital admissions in the OMED and NMED services.

Regarding the filters, the page only accounts for the data associated with a valid AdmissionKey, meaning it has to be different from -1.

The hospital admission main page has four icons on the bottom left corner that redirects to the other three medical pages (procedures, chart, and microbiology & laboratory).

Procedures Page

The procedures page (see Figure 5.2) includes six slicers (year, month, trimester, care units, no of procedures completed by a caregiver, and the items and respective procedure categories) that affect the three visual elements. This page filter only the valid AdmissionKey, meaning it has to be different from -1.

The funnel chart displays the top 5 care units with the highest number of performed procedures:

MICU – Medical intensive care unit (378)

SICU – Surgical intensive care unit (172)

CCU – Coronary care unit (141)

TSICO – Trauma/surgical intensive care unit (36)

Table 5.2: Total of hospital admissions and average, minimum and maximum medical assistance waiting time per hospital service

Service	**Service Description**	**No of admissions**	**Average medical assistance waiting time (min)**	**Minimum medical assistance waiting time (min)**	**Maximum medical assistance waiting time (min)**
MED	Medical - general service for internal medicine	73	97.57	1	985
SURG	Surgical - general surgical service not classified elsewhere	14	105.71	1	782
CMED	Cardiac Medical - for non-surgical cardiac related admissions	11	104.54	1	853
OMED	Orthopaedic medicine - non-surgical, relating to musculoskeletal system	8	304.33	2	782
NMED	Neurologic Medical - non-surgical, relating to the brain	5	225.17	1	1065
TSURG	Thoracic Surgical - surgery on the thorax	4	196.67	1	853
TRAUM	Trauma - injury or damage caused by physical harm	4	76.63	26	107
CSURG	Cardiac Surgery - for surgical cardiac admissions	4	28.60	1	98
NSURG	Neurologic Surgical - surgical, relating to the brain	3	63.60	1	218
Total		**129**	**111.36**	**1**	**1065**

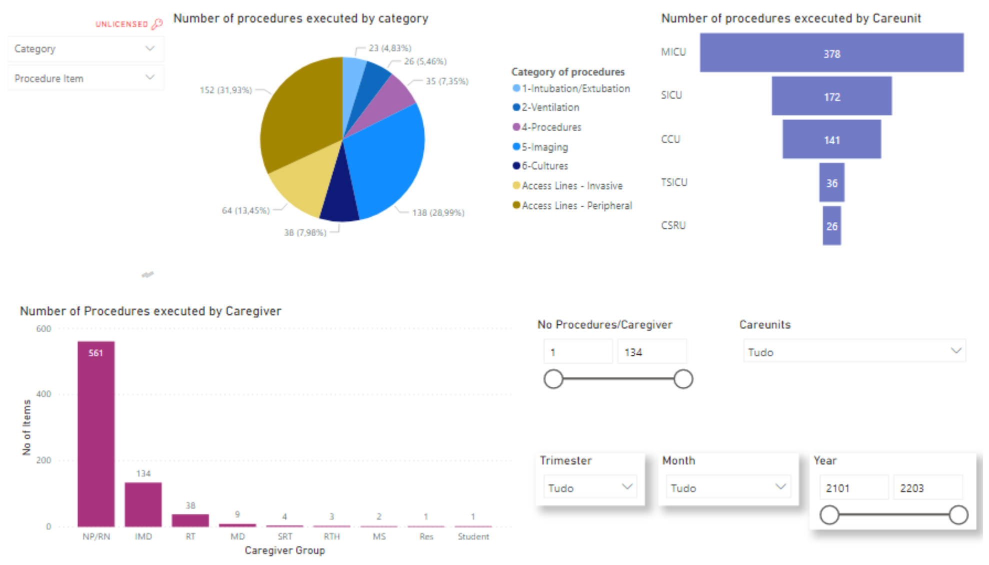


Figure 5.2: Procedures Page Overview

CSRU – Cardiac surgery recovery unit (26)

Note that this visual element was filtered to assure that only valid care units are displayed (CareunitKey different than -1).

The Care Units slicer allows the user to conduct specific analyzes on one or multiple care units.

The pie chart reveals the top 10 most common procedures performed in patients, organized by their respective categories. As seen in Figure 5.2, the most common executed procedures are associated with the Imaging (138) and the Access Lines – Peripheral (152) categories.

When a visual has a hierarchy, the drill mode allows the user to drill down or up to explore data in-depth details. In the procedures pie chart, the user can drill up the desired category and explore what procedures were executed. For example, the user has to drill up the Imaging category to investigate which procedures, from the top 10, belong to this category. In this particular case, the drill up shows that 27% of the top 10 procedures that belong to the Imaging category are CT Scans (37), and 73% are Chest X-Rays (101).

However, if the user wants to search all the performed procedures that belong to the Imaging category, it has to select the respective slices' desired category. When the category slicer filters only the Imaging category, the total of procedures is 184, which is divided by the procedures Abdominal X-Ray (4), Chest X-Ray (101), CT Scan (37), Interventional Radiology (2), Magnet Resonance Imaging (6), Pelvis (1), Transthoracic Echo (12), Ultrasound (14), and X-Ray (7) (see Figure A.3. As expected, the Chest X-ray (55%) and the CT Scan (20%) are the majority of the total number of executed imaging procedures.

This visual element was filtered to guarantee that only valid procedure categories and respective items are displayed (ItemCategoryKey and ItemKey different from -1).

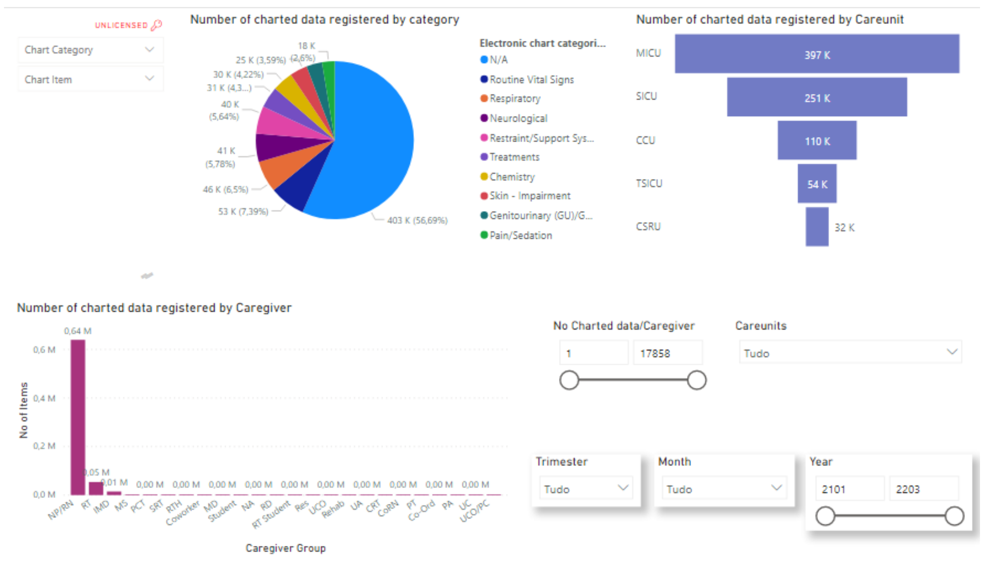


Figure 5.3: Chart Page Overview

The bar chart in Figure 5.2 represents the number of performed procedures by the caregiver group (e.g., NP/RN, MD). Generally, Registered Nurse (RN) and Nurse Practioner (NP) perform almost 75% (561) of all the medical procedures (753). FlagActive (binary flag), in this visual element, has to be 1 is a binary to filter only the caregivers who are currently employed at the Beth Israel Deaconess Medical Center.

The caregivers' chart also has a drill mode that allows the user to explore, inside a caregivers' group, the professionals' identifier codes that executed medical interventions and analyze which categories or procedures or care units are associated with them. The user can also evaluate which caregivers have higher or lower records of performed medical interventions accordingly to the available slicers (date, category, procedures, or care units).

Furthermore, it is also possible to restrict all visual elements' view by using the slicer range of the total procedures (regardless of the category and care unit) performed by caregivers.

For example, Figure A.4 shows that inside the NP/NR group, there are two caregivers (with the identifiers code 1134 and 5879), that since the day they started working at the hospital, have executed at least eight medical interventions (6 and 2, respectively) and part or all of them were Chest X-ray or Ultrasound administered at the medical intensive care unit, in the 4th trimester.

Chart Page

The chart page (see Figure 5.3) structure is very similar to the procedures page. It includes the same six slicers (year, month, trimester, care units, number of charted measurements noted by a caregiver, and the items and respective chart categories) that affect the three visual elements. The page filters the AdmissionKeys different from -1.

The funnel chart displays the top 5 valid care units (CareunitKey different than -1) with the highest number of noted electronic charted measurements:

MICU – Medical intensive care unit (397k)

SICU – Surgical intensive care unit (251k)

CCU – Coronary care unit (110k)

TSICO – Trauma/surgical intensive care unit (54k)

CSRU – Cardiac surgery recovery unit (32k)

The charted data graphic reveals the top 10 most common charted measurements registered, organized by their respective categories. As seen in Figure 5.3, the majority of registered electronic data is not associated with any specific category – “N/A” (57%) –, which includes typical ICU patients “check-in” measurements such as heart rate and arterial blood pressure. From a total of 68 chart categories, the highlighted ones are Routine Vital Signs (52k), Neurologic (41k), Restraint/Support Systems (40k), Treatments (31k), Skin – Impairment (25k), Genitourinary (23k), Respiratory (20k), Pain/Sedation (18k), and Skin – Assessment (14k).

Contrary to the procedures page, the drill mode is not available, and the user can not search the desired electronic charted measurements of a specific category since each category has at least 20 items. However, if the user wants to evaluate a specific chart item’s statistics, it can use the respective slicer.

Nevertheless, if the user wants to search all the registered charted measurements that belong to a specific category, it has to select the respective slices’ desired category. This visual element was filtered to ensure that only valid electronic charted items are represented (ItemKey differs from -1).

The bar chart in Figure 5.3 represents the number of registered electronic chart measurements by the caregiver group. Generally, Registered Nurse (RN) and Nurse Practioner (NP) register over 75% (639,863) of all the charted measurements (844,733). In this visual element, FlagActive has to be 1 is a binary to filter only the currently employed caregivers at the hospital.

The caregivers’ chart has a drill mode identical to the one on the procedure page.

For example, Figure A.5 shows two caregivers, when selected the Respiratory Therapist group, with the identifiers code 2569 and 3828 that have registered 35 and 1, respectively, charted data from the Respiratory, Alarms, and Pulmonary categories, in February of 2130. All the 36 medical records were noted at the Medical Intensive Care Unit.

Figure A.6 results from the selection of the year 2130, the Coronary Care Unit, and the Skins categories. It was registered 466 electronic data from the Skin – Impairment (277), Skin – Assessment (172), and Skin – Incisions (17) categories. All these electronic charted notes (466) were retrieved solely by nurses with identifier codes 2050 (138), 3671 (87), 3255 (72), 3397 (61), 1733 (60), and 7352 (48).

Microbiology & Laboratory Page

The Microbiology & Laboratory page (see Figure 5.4) structure consists of three slicers (year, month, trimester) that affect the four visual elements.

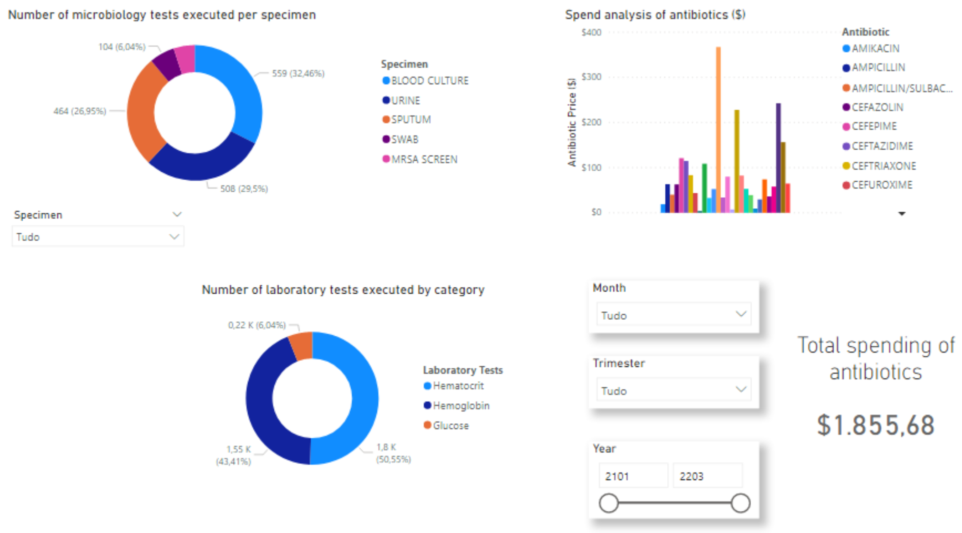


Figure 5.4: Microbiology & Laboratory Page Overview

Regarding the microbiology field, it is possible to analyze the number of microbiology tests by specimens, and the antibiotic disks costs. In the laboratory field, the user can evaluate the number of laboratory tests executed by category and lab items.

The laboratory pie chart exhibits (see Figure 5.4) the top 10 most common tests executed, organized by three categories: Chemistry (13,821), Hematology (3561), and Blood Gas (395). The visual element has a drill mode that allows the user to drill down or up to explore data in-depth details. For example, inside the Hematology category, the top tests are Hematocrit (1800), Hemoglobin (1546), and Glucose (215).

However, if the user wants to search all the executed laboratory tests that belong to the Hematology category, it has to select the respective slices' desired category. Hematology tests total 15,319 divided by Hematocrit, Platelet Count, Hemoglobin, White Blood Count, MCH, MCHC, MCV, RDW, Red Blood Cells, and PTT (see Figure A.7).

This visual element was filtered to guarantee that only valid admissions, laboratory categories, and respective items are displayed (AdmissionKey, ItemCategoryKey, and ItemKey different from -1).

The microbiology pie chart displays (see Figure 5.4) the top 5 most common specimen used in microbiology tests: Blood Culture (559), Urine (508), Sputum (464), Swab (104), and MRSA Screen (87). To investigate other specimen statistics, the user has to use the slicer placed on the graphic's left.

The antibiotics bar char shows the estimate of the antibiotic disk costs associated to the microbiology tests, assuming that five antibiotic disks are enough replicas to ensure the veracity of the test for each bacteria culture. The highest costs are associated with the Gentocimin (367.20$), Tobracymin (242.55$), Meropenem (228$), and Thrimethropin/Sulfa (156.42$) disks. Furthermore, a card displays the updated total of antibiotic disks hospital costs (1855.68$).

Unfortunately, the medical database did not have information on the procedures, electronic charted nurse-verified measurements, and laboratory tests costs to the hospital, medical insurances, and patients, so it is impossible to evaluate each department's revenue balance. However, there is a small estimate of the antibiotic disk costs, in the Microbiology & Laboratory page, by using the current market values of the BD BBL^{TM} Sensi-Disc^{TM} brand available in the Fisher Scientific platform [47]. This estimate does not account for the reagents, and the microbiology equipment (e.g., Petri dishes, growth mediums, inoculation loops) costs.

5.4 Summary

This Chapter summarizes the calculated business indicators to create healthcare analytics solutions that allow clinics, hospitals, medical professionals, and managers to quickly and confidently make data-driven business decisions while saving valuable time and resources. The developed Clinical Report includes four pages: general hospital admissions, medical procedures, electronic charted measurements, and microbiology and laboratory.

Unfortunately, the Clinical database only characterizes 100 patients and 129 hospital admissions so it is very difficult to conduct depth analyzes.

In the General Admission Page it was possible to conclude that all the selective admissions (previously planned hospital admissions) were assigned to surgical services and all the patients survived the surgical procedures. The MED service describes more than 50% of the hospitalization admissions. The average waiting medical assistance time (time between the hospital admission and the first patient's medical test) and the average hospitalization time is 111.36 minutes and 9.33 days, respectively.

In the Procedures Page, it is possible to conclude that the most common medical interventions are associated with the Imaging (138) and the Access Lines – Peripheral (152) categories.

In the Chart Page, the majority of registered electronic data is not associated with any specific category – "N/A" (57%) –, which includes common ICU patients "check-in" measurements such as heart rate and arterial blood pressure.

Generally, Registered Nurse (RN) and Nurse Practioner (NP) perform the majority (approximately 75%) of all the medical procedures and electronic charted measurements.

In the Laboratory section, there are three categories: Chemistry, Hematology and Blood Gas. The Chemistry category represents over 75% of the analyzed exams.

On the other hand, in the Microbiology section, it was calculated an estimate of the antibiotic disk costs (1855.68$). However this estimate does not account for the reagents, and the microbiology equipment (e.g., Petri dishes, growth mediums, inoculation loops) costs.

Chapter 6

Monitoring Framework

This chapter states the problem in question and introduces the solution proposal, showing the monitoring framework architecture, the necessary resources and development methodology to implement it. To conclude the chapter, it is presented monitoring framework dashboard.

6.1 Introduction

Fault monitoring is a complex process carried out by assessing the behavior of the system, namely when it is exposed to factors that may cause a fault. That is, regardless of the area, the fault monitoring process consists in obtaining useful information to have an early proactive, instead of a reactive response to prevent future failures [51].

Failure prevention is often related to the concept of risk, so its importance is proportional to the adverse effects that may arise from a possible failure. This characteristic makes the prevention of hospital database failures a central issue, one that is at risk to the quality of services provided to the patient.

To build an efficient failure prevention system, it is necessary to assess the performance of the resources that may cause these same failures in the database.

The analysis of the performance of a database depends both on the environment that surrounds it and on its vendor and aims to promote fault diagnosis [51]. Some types of database failures are described below [52].

System Crashes – The systems fail due to hardware malfunction or a bug in the database software or the operating system itself. The system needs to be rebooted.

User Error – When a user, inadvertently, performs a wrong action such as deleting a row or deleting a table by accident.

Application Software Errors – While running a user program, a transaction might have multiple SQL statements to access the database, and one of the statements might fail due to various reasons. The Database Engine usually detects these errors.

Network failures – Can occur while using a client-server configuration or a distributed database system where communication networks connect multiple database servers.

Natural and physical disasters – Damage caused to data, hardware, and software due to natural disasters (e.g., fires, floods, earthquakes, power failures).

Disk failure – When the disk cannot be accessed to write or read data. It is necessary to activate the disk recovery mechanisms and recover the lost files.

Memory Failure – When there is not enough memory, the database may have to resort to disk more times, slowing down the system causing it to become unavailable.

Processor Failure – When the processor is overloaded, it cannot respond to all requests and may cause the entire system to block.

In this dissertation, the construction of the monitoring process was divided into three phases:

Define the Action Area – Establish which components/functions to monitor to minimize the cost in terms of resource usage, resulting from this process. This selection should be made with the help of the database's end users, as they investigate abnormalities in the database's behavior.

Despite the existence of a heterogeneous set of failures, this dissertation focuses only on failures resulted from the ETL process (Application Software Errors) and on the monitoring of server resources (CPU, RAM, and Memory) and their allocation, since these are very common and serious in databases that contain a lot of information and high utilization target.

Thus, decreasing the area of action reduces the amount of data to collect, thus promoting more concise and correct monitoring.

Statistics Collection – Statistics should be collected at the level of the computer and cloud system relevant to the specified action area.

The information from the monitoring process (error messages and the server resources statistics) are stored in the server database to be analyzed whenever necessary.

Analyze Collected Statistics – The visualization reporting tools (in this particular case Microsoft Power BI) facilitate the analyses of the collected statistics to ascertain the existence of performance problems, or symptoms of the same, in the database system.

6.2 Monitoring Framework Overview

In the healthcare sector, DW integrates data from a wide variety of internal and external sources, providing an optimized and effective information platform for health decision makers [15]. However, a clinical DW development is complicated and time-consuming, primarily due to the need to integrate multiple heterogeneous data sources, whose data are generally not structured. Thus, it emerges the necessity to build a central framework to monitor all the ETL processes, implemented in the cloud or on-premises, since evaluating each data source individually is a very demanding and time-consuming task.

Since the multiple clinical data sources can be in different servers, it was necessary to ensure secure and reliable communication between these servers and the monitoring framework server (Azure SQL Server).

Thus, it was decided to send the monitoring information (ETL errors and the resources' performance metrics) via e-mail to prevent firewall restrictions. Then, the Azure Logic Application interprets the e-mail body and subject to retrieve the monitoring information and store it into the respective Azure SQL database tables. Finally, the end-user can analyze the monitoring dashboard and be alerted if one or more projects fail or are close to failure.

SendGrid is a cloud-based SMTP (Simple Mail Transfer Protocol) provider that allows sending e-mails without having to maintain e-mail servers and provides two ways to send an e-mail: through SMTP relay or Web APIs [53].

A server-side Web Application Programming Interface (API) consists of one or more publicly exposed endpoints to a defined request-response message system, typically expressed in JSON or XML, which is exposed via the HTTP (HyperText Transfer Protocol)-based web server. The HTTP requests and responses are used to access a website that is specialized for access by arbitrary computer programs [54].

The SendGrid Email API leverages SMTP to allow customers to send large amounts of transactional and triggered e-mails because it is faster since SMTP involves back-and-forth communication between the client and the server, having a slower performance when sending bulk e-mails. Furthermore, some environments may block SMTP ports (usually port 25) due to built-in or firewall restrictions, which is not an issue with Web APIs since the world-wide-web itself runs on HTTP, and most firewalls allow HTTP connections [53].

After evaluating both protocols, the SendGrid Web API was chosen to communicate the clinical databases' monitoring information, implemented on-premises or in Microsoft Azure Cloud, and the monitoring framework server. The SendGrid free plan allows the user to send 100 e-mails per day.

The following sections will discuss the steps to monitor the ETL's errors and the server/-database performance measures.

6.2.1 ETL Monitoring

Although it is common for an ETL project to implement a Log Register (e.g., see Log table in Section 4.3) to monitor the running ETL state, it becomes tough, from the end-user perspective, to analyze all the projects' Log information.

Thus, developing a central framework to monitor all the ETL's error messages allows the user to analyze multiple solutions implemented in different environments.

6.2.1.1 On-premises

In the Visual Studio Clinical Solution, when there are errors in one or more tasks in the package, its event handler OnError is called.

This solution has 4 types of ETL packages: E – Extration, T – Transformation, L – Loading, A – All ("E_ALL", "T_ALL" and "L_ALL"), and T ("ETL_ALL").

For the "E", "T", and "L" package types, the event handler OnError contains a simple SQL task to update the Log table, setting the binary flag to zero and storing the error message provided by the system local package variable (ErrorDescription).

On the other hand, the event handler OnError of the "A" and "T" package types run multiple packages, so it was necessary to develop a simple C# script to calculate the number of errors that occur in the run packages and to concatenate the respective error messages (provided by ErrorDescription of each local failed package) into a package variable. If at least one error occurred in the running package, the Log table is updated.

After the package stops running, the local and the global error message are sent using the SendGrid Web API service. Note that if one of the extraction packages runs with an error, the user will receive three e-mails because besides the local package's failure, the "E_ALL" and "ETL_ ALL" run with mistakes.

After creating a SendGrid account and generating the API Key, a simple C# script was developed to send an HTTP request, from the host web address, via POST. When this script is invoked, its input variables are the recipient and sender e-mail address, the subject, the e-mail body, and the generated API Key to configure the request's authorization header.

Similarly, it was also developed a short Microsoft Visual Basic code to send the ETL error messages using a SMTP connection, using the same input variables when invoked.

6.2.1.2 Microsoft Azure

The Microsoft Azure Functions resource is a serverless service that allows the user to run a script or piece of code. Azure Functions are integrated with Azure Data Factory (ADF), allowing the user to run these functions in a pipeline.

One of the Azure Functions created was able to extract the error information required for each failed Activity and stored it into a JSON object.

Azure Functions also support an output binding for SendGrid, which facilitates the SendGrid integration with the Azure Data Factory. The second Azure function configures the SendGrid output, including the recipient and sender e-mail address, the subject, the e-mail body, and the generated API Key. Then, the error message variable is incorporated into the body message before sending it.

6.2.2 Performance Monitoring

Performance monitoring ensures that the information about the application operating underperformance test is the closest to real-time than possible. It is essential to define a baseline, a range of measurements that represent acceptable performance under typical operating conditions. This baseline provides a reference point, which makes it easier to identify problems when they occur [55].

When troubleshooting system problems, performance data provides information about system resources' behavior when the problem occurs, which helps identify the cause(s).

Finally, monitoring application performance allows the user to project future growth and to plan for how changing the system configurations might affect future operations [55].

Performance monitoring helps verify whether the application meets its performance objectives by collecting metrics that characterize the application's behavior under different workload conditions (performance test, stress, or single-user operation). These metrics can be: response time, throughput, and resource utilization (e.g., CPU, memory, disk I/O, network bandwidth).

6.2.2.1 On-premises

The hardware components of a server are involved in servicing user requests, so it is crucial to monitor Windows system resources. The timely performance of these components is directly related to overall perceived application performance. Therefore, a problem with one or more of these four areas is likely to result in user complaints. SQL Server relies heavily on CPU performance, available memory, and disk throughput, whereas the client performance depends heavily on network performance.

Furthermore, SQL Server can be very demanding on memory, and performance can suffer if physical memory becomes exhausted. The disk is almost certainly the slowest component because of its mechanical nature. SQL Server's need to retrieve disk data often means any delays at the disk I/O will impact overall performance.

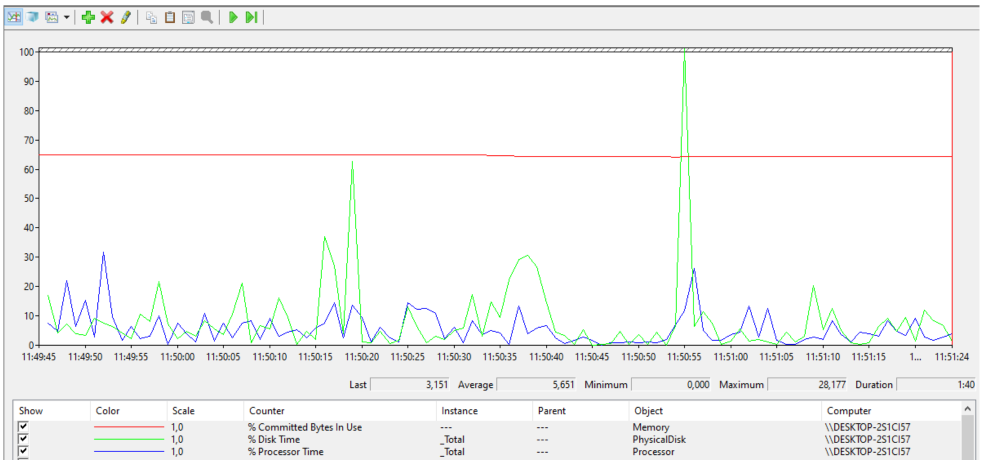
The Windows Performance Monitor is a system monitoring program, available in Windows 7 and 10, that determines the cause of problems on a local or remote computer by measuring the performance of hardware, software services, and applications by analyzing data, such as CPU, physical disk, memory, and network usage.

Windows Performance Monitor has over 350 performance measurement criteria, called "counters", available that can be displayed, in real-time, as a graph, a bar chart, or numeric values.

In this project, when monitoring system resources of a machine with SQL Server installed, the most relevant counters to be tracked are:

Processor Time (percentage) – The percentage of elapsed time that the processor spends executing a non-Idle thread (the purpose of Idle threads is to keep the CPU occupied until the next computation or process is fed). This counter is the primary indicator of processor activity and displays the average percentage of busy time observed during the sample interval. It is calculated by monitoring the percentage of time that the service was inactive, and then subtracting that value from 100% [56].

If this percentage stays over 80% constantly, it is difficult for the processor to handle all the processes.



Last 3,151 Average 5,651 Minimum 0,000 Maximum 28,177 Duration 1:40

Show	Color	Scale	Counter	Instance	Parent	Object	Computer
✓	🟥	1,0	% Committed Bytes In Use	---	---	Memory	\\DESKTOP-2S1CI57
✓	🟩	1,0	% Disk Time	_Total	---	PhysicalDisk	\\DESKTOP-2S1CI57
✓	🟦	1,0	% Processor Time	_Total	---	Processor	\\DESKTOP-2S1CI57

Figure 6.1: Performance monitor graph with the counters Total Processor Time (percentage), Memory committed bytes in use (percentage), and Total Physical Disk Time (percentage).

In this particular case, this counter is calculated for each processor since the local machine system has four. Each processor has an Idle thread that consumes unproductive processor cycles not used by any other threads.

Total Processor Time (percentage) – The fraction of time that all the processors on the system are busy executing non-idle threads [56].

In this case, if all four processors are always busy, this value is 100%, and if one-quarter of the processors are 100% busy, then the value is 25%.

Memory committed bytes in use (percentage) – This counter shows what percentage of RAM is currently in use or is committed. This counter fluctuates its values as different programs are opened and closed, but if it keeps on increasing, there might be a memory leak [56].

Total Physical Disk Time (percentage) – This counter monitors the percentage of elapsed time taken when a drive is busy processing read and write requests [56].

In Figure 6.1, it is possible to observe the performance monitor chart with the previously selected counters (Total Processor Time (percentage), Memory committed bytes in use (percentage), and Total Physical Disk Time (percentage)) when running the ETL process described in Section 4.3. The Physical disk time highest peak signalizes the end of the ETL process.

PowerShell is a task automation and configuration management framework from Microsoft, consisting of a command-line shell and the associated scripting language [56]. PowerShell allows the user to search the performance counters available in the Windows Performance Monitor and select the relevant ones. Therefore, a short PowerShell script was developed to calculate the Processor Time for the total and each of the four processors, Physical disk time, and the memory committed bytes in use.

The aim is to monitor the performance resources when the ETL process runs so that the developed PowerShell script runs simultaneously to the ETL. Therefore, the ETL process (meaning the package "ETL_ALL") is scheduled, using the SQL Server Agent that allows the user to set the time and the frequency to run a specific package. Then, the Windows Task Manager, which provides the ability to schedule the launch of programs or scripts at pre-defined times, is used to launch the developed PowerShell Script simultaneously as the SQL Server Agent job. Note that the Task Manager can launch the script as often as the user desires within the time frame of the ETL process running.

In this particular project, knowing that the ETL of the clinical database takes approximately 20 minutes to finish running, the Task Manager will launch the script at the same time as the SQL Server Agent job (7 PM), and repeat the process every 5 minutes, reading the pre-selected metrics at a total of three times.

6.2.2.2 Microsoft Azure

Azure resources generate a significant amount of monitoring data. Azure Monitor collects and aggregates data from various sources into a Metrics or Logs platform where it can be used for analysis, visualization, and alerting.

Logs are events that occurred within the system (e.g., ETL failure or successful) and are usually stored in a table with structured data with a timestamp.

Metrics are numerical values that describe an aspect of a system at a particular point in time. They are collected at regular intervals and are identified with, at least, a timestamp, a name, and a value.

Metrics in Azure Monitor are stored in a time-series database, which is optimized for analyzing timestamped data, making these metrics very useful for alerting and fast detection of issues.

Azure Monitor allows the user to monitor several resources (e.g., Data Factory or Azure SQL Database). The performance measurement criteria (metrics) can be displayed, near real-time, as a graph or numeric values. It is also possible to create alerts for each selected metric to warn the user when they reach a pre-defined threshold.

This dissertation, since both implemented ETL (on-premises and Microsoft Azure) have the same source database (Azure SQL Database), makes sense to monitor the Azure SQL Database. In this particular case, the chosen metrics are the CPU Percentage and the Data Space Use Percentage.

In Figure 6.2, it is possible to observe the Azure monitor chart with the previously selected metrics (CPU Percentage and the Data Space Use Percentage) when running the ETL process described in Section 4.3.

Additionally, it is also possible to monitor the pipeline developed in the Data Factory (Integration Server) using the Azure Monitor.

Similarly to the Windows PowerShell, there is an Azure PowerShell where it is possible to investigate the performance metrics (see Appendix A.8) available in the Azure Monitor and select the relevant ones. Therefore, it was possible to develop a short Azure PowerShell script, using the Az module, to calculate the Percentage of CPU and the Data Space Use Percentage.

The Azure PowerShell scripts are run Azure Automation runbook. Azure automation includes runbook jobs and watchers. Billing for jobs (0.002€/minute) is based on the number of job run

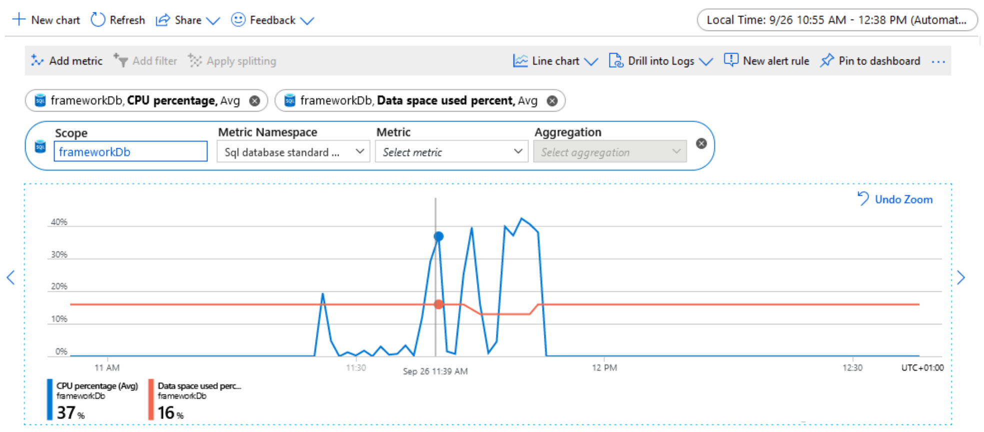


Figure 6.2: Azure monitor graph with the metrics CPU Percentage and the Data Space Use Percentage.

time minutes used in the month, and for watchers (0.002€/hour) is based on the number of hours used in a month [57].

Considering the low complexity of the developed pipeline in the Data Factory and the Azure Automation runbook pricing, the performance measures Azure PowerShell script was not implemented in the runbook. However, the developed Azure PowerShell transition to the Azure Automation runbook would be a simple task.

The aim is to monitor the performance resources when the ETL process runs, so the developed PowerShell script has runs simultaneously to the ETL. The Azure PowerShell script can be scheduled using the Azure Automation scheduling features.

The Azure Automation notebook needs to synchronize to the Azure resource that the user wants to monitor, which can be the Data Factory or the Azure SQL Database. Note that the Data Factory Scheduler allows the user to set the time and the frequency to run a specific pipeline.

In summary, during the Clinical ETL running process (on-premises), it is possible to monitor the local server using the Windows Performance Monitor metrics and the Azure SQL database using Azure Monitor metrics of this resource. The developed pipeline in the Azure Data Factory (Extraction of the antibacterial disk prices) with a database integrated into the Azure SQL Database, the performance metrics for both these resources (Azure SQL Database and Azure DF) can be obtained using the Azure Monitor functions.

6.2.2.3 Notifications

The next step after retrieving the performance counter, whether it is on-premises or from the cloud, is to send a notification with these metrics to later be interpreted by the Logic Application (see Subsection 6.2.3).

These notifications, similarly to the ETL error notifications, are sent using the SendGrid Web API.

After creating a SendGrid account and generating the API Key, it was developed a simple PowerShell function to send an HTTP request, from the host web address, via POST using the *Invoke–RestMethod* command (see Code A.1). When this function is invoked, the user has to provide the recipient and sender e-mail address, the subject, the e-mail body, and the generated API Key to configure the request's authorization header.

On-premises, the PowerShell script to send an e-mail was incorporated into the performance counters calculating PowerShell script. Then, each time the Task Manager launched both scripts, an e-mail would be sent with the timestamp, metrics names, and respective values calculated during the ETL run.

However, on-premises, sometimes the firewalls are blocked, for security reasons, that do not accept HTTP requests, and it is not possible to use the SendGrid PowerShell script. One possible solution is to add to the performance counters PowerShell script a function to save, in a CSV file, these metrics names and the respective values along with the timestamp, in a pre-defined local computer path. Thus, there will be appended three rows with the timestamp, metrics names, and values in the CSV file for each run ETL process.

Each CSV file contains the performance counters' information for each time a specific ETL project run. In this case, if the project MIMC runs 50 times, there will be 150 rows in the CSV file. The CSV File (Clinical.csv) and the Visual Studio solution (Clinical.snl) where the ETL was implemented have the same name to easily associate each file to its project if the local machine has multiple VS solutions, and consequently, multiple ETL processes implemented.

Finally, in the Visual Studio Solution, it was created a package, to run at the end of the ETL process, to read the CSV File and to insert the new rows into the ETL database (in this case, Azure SQL database) monitoring parameters table (PARAMS.PARAMS_Resources_Tracking). Then, the new rows of this table are read using SQL statements and sent, in the body e-mail, using the SendGrid Web API script implemented in the Visual Studio Solution, similar to the process in Subsection 6.2.3.

In the Microsoft Azure environment, it is also possible to implement the script A.1 to send an e-mail with the SendGrid Web API. However, it has two crucial steps with associated costs:

1. Create an Azure KeyVault and store the SendGrid API key in it. The subscription to this service costs 4.217€/key per month and 0.026€/10,000 transactions.

2. Create a runbook, with Azure Automation resource, that retrieves the API key and implements both scripts (calculate performance metrics and sends an e-mail) in the Azure PowerShell. It also has the functionality to schedule the notebook to run at a specific time and frequency.

When evaluating the associated costs in sending a SendGrid e-mail with the Azure Automation, and knowing that the pipeline created in the Data Factory was very simple and only a prototype, it was decided not to invest in these Azure Resources. However, in both environments on-premise and cloud, the SendGrid Web API method was proven to work to send e-mails on failure and success, as seen in the Subsection 6.2.1. Furthermore, since both environments are compatible with PowerShell scripts, the SendGrid one would be easily implemented in this project.

6.2.3 Logic Application

Logic Apps is the Azure Integration software as a service (SaaS) solution from Microsoft that can connect data or devices anywhere on-premises or in the cloud using APIs calls (see more details in Section 2.7.4).

The goals Logic Application workflow is to store both project's ETL error messages and performance measures into the respective Azure SQL database tables. The monitoring schema is composed of three tables: Projects, ETL_MONITORING, and RESOURCES_MONITORING. The first table contains the listed names and respective primary keys of all the implemented projects on-premises and Microsoft Azure Cloud. The ETL_MONITORING stores the ETL primary key, projectKey (FK references the Project table), the isSuccess binary flag (0 if the ETL run with failure; 1 if it was a success), the ETL process timestamp, and the respective error message if applied.

On the other hand, the RESOURCES_MONITORING stores the row primary key (rowKey), projectKey (FK references the Project table), the monitoringKey, the resource monitoring timestamp, the resource name, and its value. Each row describes one performance metric retrieved from one ETL project, which is identified by the ProjectKey. The monitoringKey allows to aggregate the metrics obtained and sent, in the same e-mail, through a PowerShell script. For example, for each mail sent containing the performance metrics obtained during the project Clinical's ETL process, it is inserted one row for each performance metric with the same projectKey and the same monitoringKey but with different sequential rowKey.

The built workflow has two activated triggers every time a new e-mail arrives and two actions that execute stored procedures using the Microsoft e-mail account's connection and the Azure SQL Database connection, respectively.

The first pair of trigger and action aims to read the subject and body of every e-mail sent to the "ETL Monitoring" Microsoft e-mail folder and to process this information in the stored procedure. The message's subject has the following structure "ETL from project: <project_name>", and in the stored procedure, the project name is compared to the listed project names of the Project table present in the Azure SQL Database. The message's body first line is the following "ETL from project <project_name> run with <state>". If the <state> is "success", it is inserted a row into the ETL_MONITORING table the corresponding projectKey and the isSuccess binary flag is set to 1. If the <state> is "failure", it is inserted a row with corresponding projectKey, isSuccess is set to 0, and the message body (contains the description of all the errors that occur) is added to the errorMessage column.

The second pair of trigger and action interprets the subject and body of all the e-mails sent to the "Resources Monitoring" Microsoft e-mail folder. The analysis of the message subject is similar to the one described before. Besides matching the message subject name to the existing project names table, the stored procedure reads every row of the e-mail and inserts the timestamp, metric name, and respective value into RESOURCES_MONITORING table, along with the generated monitoringKey.

6.2.4 Monitoring Dashboard

Clinical institutes often have multiple ETL processes, which can be implemented on-premises or in the cloud, so it is important to have a central monitoring dashboard to analyze the performance metrics and ETL states (success or failure) of each ETL project. This dashboard was built in the Power BI Desktop application.

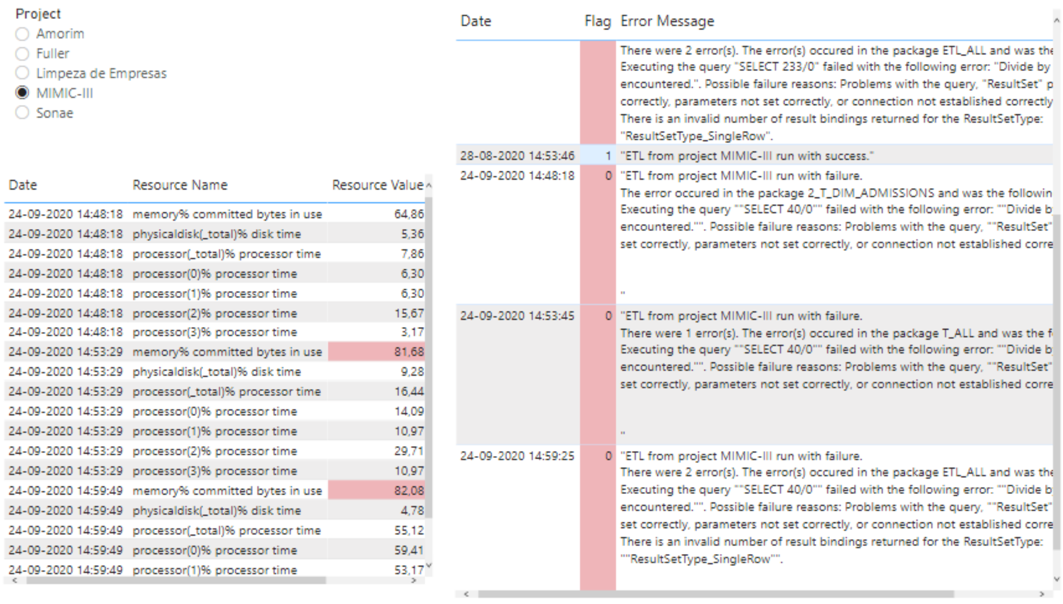
The dashboard in Figure 6.3 has a slicer to navigate the different projects and two tables to visualize, for each project, the latest performance metrics values, and the error ETL messages (if applied).

In the ETL state table, if the binary flag isSuccess is zero, the table row color turns red so the user can spot more easily the project that needs closer attention. The same applies to the Performance Metrics table if any of these measurements is higher than 80%.

6.3 Summary

The clinical DW integrates multiple heterogeneous data sources in the healthcare sector, providing an optimized and effective information platform for health decision-makers. Since evaluating, individually, each clinical data source is a very demanding and time-consuming task, it is necessary to build a central framework to monitor all the Clinical ETL processes, implemented in the cloud or on-premises. The central framework was implemented in the Microsoft Azure Environment to monitor the different clinical ETL's errors and the resources' performance metrics. On-premises, the selected server counters to monitor are: Total Processor Time (percentage), Memory committed bytes in use (percentage), and Total Physical Disk Time (percentage)). The CPU Percentage and the Data Space Use Percentage are the chosen metrics to monitor the Azure SQL Database.

The monitoring information (ETL errors and the resources' performance metrics) were sent via e-mail to prevent firewall restrictions, using the SendGrid Web API service. Then, the Azure Logic Application interprets the e-mail body and subject to retrieve the monitoring information and store it into the respective Azure SQL database tables. Finally, the end-user can analyze the monitoring dashboard and be alerted if one or more projects fail or are close to failure.



Project
- ○ Amorim
- ○ Fuller
- ○ Limpeza de Empresas
- [✓ MIMIC-III]
- ○ Sonae

Date	Resource Name	Resource Value
24-09-2020 14:48:18	memory% committed bytes in use	64,86
24-09-2020 14:48:18	physicaldisk(_total)% disk time	5,36
24-09-2020 14:48:18	processor(_total)% processor time	7,86
24-09-2020 14:48:18	processor(0)% processor time	6,30
24-09-2020 14:48:18	processor(1)% processor time	6,30
24-09-2020 14:48:18	processor(2)% processor time	15,67
24-09-2020 14:48:18	processor(3)% processor time	3,17
24-09-2020 14:53:29	memory% committed bytes in use	81,68
24-09-2020 14:53:29	physicaldisk(_total)% disk time	9,28
24-09-2020 14:53:29	processor(_total)% processor time	16,44
24-09-2020 14:53:29	processor(0)% processor time	14,09
24-09-2020 14:53:29	processor(1)% processor time	10,97
24-09-2020 14:53:29	processor(2)% processor time	29,71
24-09-2020 14:53:29	processor(3)% processor time	10,97
24-09-2020 14:59:49	memory% committed bytes in use	82,08
24-09-2020 14:59:49	physicaldisk(_total)% disk time	4,78
24-09-2020 14:59:49	processor(_total)% processor time	55,12
24-09-2020 14:59:49	processor(0)% processor time	59,41
24-09-2020 14:59:49	processor(1)% processor time	53,17

Date	Flag	Error Message
		There were 2 error(s). The error(s) occured in the package ETL_ALL and was the Executing the query "SELECT 233/0" failed with the following error: "Divide by encountered.". Possible failure reasons: Problems with the query, "ResultSet" p correctly, parameters not set correctly, or connection not established correctly There is an invalid number of result bindings returned for the ResultSetType: "ResultSetType_SingleRow".
28-08-2020 14:53:46	1	"ETL from project MIMIC-III run with success."
24-09-2020 14:48:18	0	"ETL from project MIMIC-III run with failure. The error occured in the package 2_T_DIM_ADMISSIONS and was the followin Executing the query ""SELECT 40/0"" failed with the following error: ""Divide b encountered."". Possible failure reasons: Problems with the query, ""ResultSet"" set correctly, parameters not set correctly, or connection not established corre "
24-09-2020 14:53:45	0	"ETL from project MIMIC-III run with failure. There were 1 error(s). The error(s) occured in the package T_ALL and was the f Executing the query ""SELECT 40/0"" failed with the following error: ""Divide b encountered."". Possible failure reasons: Problems with the query, ""ResultSet"" set correctly, parameters not set correctly, or connection not established corre "
24-09-2020 14:59:25	0	"ETL from project MIMIC-III run with failure. There were 2 error(s). The error(s) occured in the package ETL_ALL and was the Executing the query ""SELECT 40/0"" failed with the following error: ""Divide b encountered."". Possible failure reasons: Problems with the query, ""ResultSet"" set correctly, parameters not set correctly, or connection not established corre There is an invalid number of result bindings returned for the ResultSetType: ""ResultSetType_SingleRow"".

Figure 6.3: Monitoring Dashboard

Chapter 7

Conclusions and Future Work

In this chapter, the conclusions about the work performed are presented, from the perspective of the results and contributions obtained. In addition, the future work to be carried out is also presented.

7.1 Conclusions

The Health industry generates and collects a vast amount of data every day (images, diagnoses, medical records, among others) from several data sources, making it very difficult to aggregate, structure, and organize this data efficiently into one information system.

SClínico/SONHO V2/LIGHT systems appeared as a strategy defined by the Portuguese Ministry of Health to achieve uniformity of clinical records, increasing the efficiency of health professionals and, consequently, saving resources and improving quality of care.

However, since 2017, only Centro Hospitalar de Leiria and Centro Hospitalar Distrital de Santarém have gradually implemented the SClínico/SONHO V2/LIGHT systems. Unfortunately, even with this platform, there are still many other heterogeneous information systems (for example, ALERT is used to request a medical specialty).

Thus, in the health units, Business Intelligence (BI) systems are crucial in the quality of services provided since the data needs to be permanently available and the information flow working correctly. BI systems take even a more significant role in the Health Industry because improving medical professionals, and hospital managers' decision-making can save lives.

In this dissertation, it was developed a Clinical DW to replicate the existing healthcare solutions, evaluate medical professionals' workload, analyze the volume of medical interventions and laboratory tests, and calculate the hospital revenue balance.

Unfortunately, the Clinical database only characterizes 100 patients and 129 hospital admissions, so it is challenging to conduct depth analyzes.

The Clinical Report includes four pages: general hospital admissions, medical procedures, electronic charted measurements, and microbiology and laboratory. In the General Admission Page, it was possible to conclude that all the selective admissions (previously planned hospital admissions) were assigned to surgical services. All the patients survived the surgical procedures. The MED service describes more than 50% of the hospitalization admissions. The average waiting medical assistance time (time between the hospital admission and the first patient's medical test) and the average hospitalization time is 111.36 minutes and 9.33 days, respectively.

In the Procedures Page, it is possible to conclude that the most common medical interventions are associated with the Imaging (138) and the Access Lines – Peripheral (152) categories.

In the Chart Page, the majority of registered electronic data is not associated with any specific category – "N/A" (57%) –, which includes typical ICU patients "check-in" measurements such as heart rate and arterial blood pressure.

Generally, Registered Nurse (RN) and Nurse Practioner (NP) perform the majority (approximately 75%) of all the medical procedures and electronic charted measurements.

In the Laboratory section, there are three categories: Chemistry, Hematology, and Blood Gas. The Chemistry category represents over 75% of the analyzed exams.

On the other hand, in the Microbiology section, it was calculated an estimate of the antibiotic disk costs (1855.68$). However, this estimate does not account for the reagents and the microbiology equipment (e.g., Petri dishes, growth mediums, inoculation loops) costs.

Due to the Hospital's data source heterogeneity, one healthcare institute can have multiple Clinical Extract, Transform, and Load processes implemented in different environments (on-premises or cloud).

Therefore, it emerges the necessity to monitor multiple Clinical Data Warehouses to prevent process failures that can put the patient's health and safety at risk.

In this project, it was proposed to build a central monitoring framework prototype based in the Microsoft Azure Environment. The communication between the framework server and the clinical data source servers is established via the SendGrid Web API service.

The SendGrid proved to be an efficient method to transfer the monitoring information (ETL error messages and server performance measurements) from the clinical DW to the Azure SQL database framework.

The monitoring dashboard shows the end-user the ETL error messages and server performance measurements of the multiple Clinical ETL processes supervised. Instead of checking each ETL's Log table, the user can easily navigate in the report to observe if any project run with failure or be alerted if any of the performance metrics (e.g., CPU, RAM) have dangerously high values.

7.2 Future Work

The MIMIC-III demo v.1.4 was not the most suitable database to develop a clinical ETL since there were only 100 registered deceased patients, making it very difficult to assess the medical interventions' impact, hospitalization period, and admission waiting time in the hospital survival rate. Furthermore, the invoice costs associated with the billing of diagnoses, medical procedures, and drugs administrated were not available, making it impossible to evaluate the Hospital revenue.

Regarding the Azure environment's performance metrics, two Azure PowerShell scripts were developed: one to calculate the Percentage of CPU and the Data Space Use Percentage of the Azure SQL database and the other to send this monitoring information using the SendGrid Web API service. However, after evaluating the associated costs in implementing the Azure PowerShell scripts in the Azure Automation Runbook, and knowing that the pipeline created in the Data Factory was only a prototype, it was decided not to invest in this Azure Resource.

In the future, a more complex Clinical ETL should be implemented in the Azure Data Factory to test developed PowerShell scripts, and consequently, store the Azure SQL database performance metrics into the monitoring framework database.

Although the SendGrid Web API service is compatible with the Microsoft Azure Environment, it would help implementing this method in other cloud vendors.

Currently, the user can analyze the performance metrics in the monitoring dashboard's table since it is only retrieved a few points per ETL execution. However, it could be useful for the user to visualize this information under a graphic form. Additionally, it would also help the user see ETL's past months or year statistics.

Appendix A

Supplementary Material

Listing A.1: Function Send-EmailWithSendGrid

```
Function Send-EmailWithSendGrid {
     Param
    (
        [Parameter(Mandatory=$true)]
        [string] $From,

        [Parameter(Mandatory=$true)]
        [String] $To,

        [Parameter(Mandatory=$true)]
        [string] $ApiKey,

        [Parameter(Mandatory=$true)]
        [string] $Subject,

        [Parameter(Mandatory=$true)]
        [string] $Body
    )

    $headers = @{}
    $headers.Add("Authorization","Bearer␣$ApiKey")
    $headers.Add("Content-Type", "application/json")

    $jsonRequest = [ordered]@{
        personalizations= @(@{to = @(@{email =  "$To"})
            subject = "$SubJect" })
            from = @{email = "$From"}
            content = @( @{ type = "text/plain" value = "$Body"})
            } | ConvertTo-Json -Depth 10

    Invoke-RestMethod  -Uri "https://api.sendgrid.com/v3/mail/send"
    -Method Post -Headers $headers -Body $jsonRequest
}
```

Table A.1: Clinical ETL's Control Packages Table

ID	PackageName	TypeEtl	OrderPackage	CreationDate	AlterDate	CreateUser	AlterUser	Record_Active
12	1_E_CSV_PATIENTS.dtsx	E	1	2020-07-09 11:16:51.943	NULL	B2F_Margarida	NULL	1
1	1_E_CSV_ADMISSIONS.dtsx	E	2	2020-07-09 11:16:51.857	NULL	B2F_Margarida	NULL	1
2	1_E_CSV_CAREGIVERS.dtsx	E	3	2020-07-09 11:16:51.863	NULL	B2F_Margarida	NULL	1
4	1_E_CSV_D_CARE_UNITS.dtsx	E	4	2020-07-09 11:16:51.877	NULL	B2F_Margarida	NULL	1
6	1_E_CSV_D_ITEMS_CATEGORY.dtsx	E	5	2020-07-09 11:16:51.890	NULL	B2F_Margarida	NULL	1
5	1_E_CSV_D_ITEMS.dtsx	E	6	2020-07-09 11:16:51.883	NULL	B2F_Margarida	NULL	1
7	1_E_CSV_D_LABITEMS.dtsx	E	7	2020-07-09 11:16:51.897	NULL	B2F_Margarida	NULL	1
8	1_E_CSV_D_SERVICES.dtsx	E	8	2020-07-09 11:16:51.910	NULL	B2F_Margarida	NULL	1
9	1_E_CSV_ICUSTAYS.dtsx	E	9	2020-07-09 11:16:51.913	NULL	B2F_Margarida	NULL	1
10	1_E_CSV_LABEVENTS.dtsx	E	10	2020-07-09 11:16:51.923	NULL	B2F_Margarida	NULL	1
11	1_E_CSV_MICROBIOLOGYEVENTS.dtsx	E	11	2020-07-09 11:16:51.937	NULL	B2F_Margarida	NULL	1
13	1_E_CSV_PROCEDUREEVENTS_MV.dtsx	E	12	2020-07-09 11:16:51.950	NULL	B2F_Margarida	NULL	1
3	1_E_CSV_CHARTEVENTS.dtsx	E	13	2020-07-09 11:16:51.870	NULL	B2F_Margarida	NULL	1
59	1_E_CSV_D_ITEMS_ANTIBACTERIUM_PRICES.dtsx	E	14	2020-08-16 23:46:15.933	NULL	B2F_Margarida	NULL	1
15	2_T_DIM_PATIENTS.dtsx	T	14	2020-07-14 16:31:14.447	NULL	B2F_Margarida	NULL	1
16	2_T_DIM_ADMISSIONS.dtsx	T	15	2020-07-14 16:31:14.487	NULL	B2F_Margarida	NULL	1
17	2_T_DIM_CAREGIVERS.dtsx	T	16	2020-07-14 16:31:14.497	NULL	B2F_Margarida	NULL	1
18	2_T_DIM_D_CARE_UNITS.dtsx	T	17	2020-07-14 16:31:14.523	NULL	B2F_Margarida	NULL	1
19	2_T_DIM_D_SERVICES.dtsx	T	18	2020-07-14 16:31:14.533	NULL	B2F_Margarida	NULL	1
20	2_T_DIM_D_ITEMS_CATEGORY_CHART.dtsx	T	19	2020-07-14 16:31:14.543	NULL	B2F_Margarida	NULL	1
21	2_T_DIM_D_ITEMS_CATEGORY_LABITEMS.dtsx	T	20	2020-07-14 16:31:14.550	NULL	B2F_Margarida	NULL	1
22	2_T_DIM_D_ITEMS_CATEGORY_PROCEDURE.dtsx	T	21	2020-07-14 16:31:14.560	NULL	B2F_Margarida	NULL	1
23	2_T_DIM_D_ITEMS_ANTIBACTERIUM.dtsx	T	22	2020-07-14 16:31:14.570	NULL	B2F_Margarida	NULL	1
24	2_T_DIM_D_ITEMS_CHART.dtsx	T	23	2020-07-14 16:31:14.580	NULL	B2F_Margarida	NULL	1
25	2_T_DIM_D_ITEMS_ORGANISM.dtsx	T	24	2020-07-14 16:31:14.590	NULL	B2F_Margarida	NULL	1
26	2_T_DIM_D_ITEMS_PROCEDURE.dtsx	T	25	2020-07-14 16:31:14.600	NULL	B2F_Margarida	NULL	1
27	2_T_DIM_D_ITEMS_SPECIMEN.dtsx	T	26	2020-07-14 16:31:14.610	NULL	B2F_Margarida	NULL	1
28	2_T_DIM_D_LABITEMS.dtsx	T	27	2020-07-14 16:31:14.617	NULL	B2F_Margarida	NULL	1
29	2_T_DIM_ICUSTAYS.dtsx	T	28	2020-07-14 16:31:14.623	NULL	B2F_Margarida	NULL	1
30	2_T_FACT_LABEVENTS.dtsx	T	29	2020-07-14 16:31:14.633	NULL	B2F_Margarida	NULL	1
31	2_T_FACT_MICROBIOLOGYEVENTS.dtsx	T	30	2020-07-14 16:31:14.640	NULL	B2F_Margarida	NULL	1
32	2_T_FACT_PROCEDUREEVENTS_MV.dtsx	T	31	2020-07-14 16:31:14.647	NULL	B2F_Margarida	NULL	1
33	2_T_FACT_CHARTEVENTS.dtsx	T	32	2020-07-14 16:31:14.653	NULL	B2F_Margarida	NULL	1
34	2_T_FACT_SERVICES.dtsx	T	33	2020-07-14 16:31:14.663	NULL	B2F_Margarida	NULL	1
35	3_L_DIM_PATIENTS.dtsx	L	34	2020-07-20 22:09:31.683	NULL	B2F_Margarida	NULL	1
36	3_L_DIM_ADMISSIONS.dtsx	L	35	2020-07-20 22:09:31.697	NULL	B2F_Margarida	NULL	1
37	3_L_DIM_CAREGIVERS.dtsx	L	36	2020-07-20 22:09:31.703	NULL	B2F_Margarida	NULL	1
38	3_L_DIM_D_CARE_UNITS.dtsx	L	37	2020-07-20 22:09:31.710	NULL	B2F_Margarida	NULL	1
39	3_L_DIM_D_SERVICES.dtsx	L	38	2020-07-20 22:09:31.720	NULL	B2F_Margarida	NULL	1
40	3_L_DIM_D_ITEMS_CATEGORY_CHART.dtsx	L	39	2020-07-20 22:09:31.727	NULL	B2F_Margarida	NULL	1
41	3_L_DIM_D_ITEMS_CATEGORY_LABITEMS.dtsx	L	40	2020-07-20 22:09:31.733	NULL	B2F_Margarida	NULL	1
42	3_L_DIM_D_ITEMS_CATEGORY_PROCEDURE.dtsx	L	41	2020-07-20 22:09:31.740	NULL	B2F_Margarida	NULL	1
43	3_L_DIM_D_ITEMS_ANTIBACTERIUM.dtsx	L	42	2020-07-20 22:09:31.747	NULL	B2F_Margarida	NULL	1
44	3_L_DIM_D_ITEMS_CHART.dtsx	L	43	2020-07-20 22:09:31.757	NULL	B2F_Margarida	NULL	1
45	3_L_DIM_D_ITEMS_ORGANISM.dtsx	L	44	2020-07-20 22:09:31.763	NULL	B2F_Margarida	NULL	1
46	3_L_DIM_D_ITEMS_PROCEDURE.dtsx	L	45	2020-07-20 22:09:31.770	NULL	B2F_Margarida	NULL	1
47	3_L_DIM_D_ITEMS_SPECIMEN.dtsx	L	46	2020-07-20 22:09:31.780	NULL	B2F_Margarida	NULL	1
48	3_L_DIM_D_LABITEMS.dtsx	L	47	2020-07-20 22:09:31.787	NULL	B2F_Margarida	NULL	1
49	3_L_DIM_ICUSTAYS.dtsx	L	48	2020-07-20 22:09:31.797	NULL	B2F_Margarida	NULL	1
50	3_L_FACT_LABEVENTS.dtsx	L	49	2020-07-20 22:09:31.803	NULL	B2F_Margarida	NULL	1
51	3_L_FACT_MICROBIOLOGYEVENTS.dtsx	L	50	2020-07-20 22:09:31.810	NULL	B2F_Margarida	NULL	1
52	3_L_FACT_PROCEDUREEVENTS_MV.dtsx	L	51	2020-07-20 22:09:31.817	NULL	B2F_Margarida	NULL	1
53	3_L_FACT_CHARTEVENTS.dtsx	L	52	2020-07-20 22:09:31.827	NULL	B2F_Margarida	NULL	1
54	3_L_FACT_SERVICES.dtsx	L	53	2020-07-20 22:09:31.833	NULL	B2F_Margarida	NULL	1
55	E_ALL.dtsx	A	54	2020-07-20 22:09:31.840	NULL	B2F_Margarida	NULL	1
56	T_ALL.dtsx	A	55	2020-07-20 22:09:31.850	NULL	B2F_Margarida	NULL	1
57	L_ALL.dtsx	A	56	2020-07-20 22:09:31.857	NULL	B2F_Margarida	NULL	1
58	ETL_ALL.dtsx	F	57	2020-07-20 22:09:31.863	NULL	B2F_Margarida	NULL	1

Table A.2: Business Indicators' DAX Expressions

Report	Indicator	DAX
Admissions	Waiting Medical Assistance Time (AVG)	AVERAGEX('Admission Events', 'Admission Events'[Time 1st Test]*)
	Waiting Medical Assistance Time (MIN)	MINX('Admission Events', 'Admission Events'[Time 1st Test]*)
	Waiting Medical Assistance Time (MAX)	MAXX('Admission Events', 'Admission Events'[Time 1st Test]*)
	Hospitalization Time (AVG)	AVERAGEX('DW DW_DIM_ADMISSIONS', 'DW DW_DIM_ADMISSIONS'[Date Difference]**)
	Hospitalization Time (MIN)	MINX('DW DW_DIM_ADMISSIONS', 'DW DW_DIM_ADMISSIONS'[Date Difference]**)
	Hospitalization Time (MAX)	MAXX('DW DW_DIM_ADMISSIONS', 'DW DW_DIM_ADMISSIONS'[Date Difference]**)
	Hospital Admissions by Service	Calculate(COUNT('DW DW_FACT_SERVICES'[AdmissionKey]) , FILTER('DW DW_FACT_SERVICES', 'DW DW_FACT_SERVICES'[ActiveServiceFlag] = 1), FILTER('DW DW_Fact_SERVICES', 'DW DW_FACT_SERVICES'[CurrServiceKey]<>-1), USERELATIONSHIP('DW DW_DIM_ADMISSIONS'[AdmissionKey], 'DW DW_FACT_SERVICES'[AdmissionKey]))
	% Hospital Admission Deaths	COUNT('DW DW_DIM_ADMISSIONS'[AdmissionKey])
	% Hospital Admission Type	COUNT('DW DW_DIM_ADMISSIONS'[AdmissionKey])
Procedures	% CategoryProcedures	Calculate(COUNT('DW DW_FACT_PROCEDUREEVENTS_MV'[ItemKey]), USERELATIONSHIP('DW DW_FACT_PROCEDUREEVENTS_MV'[ItemKey], 'DW DW_DIM_D_ITEMS_PROCEDURE'[ItemKey]), USERELATIONSHIP('DW DW_DIM_D_ITEMS_PROCEDURE'[ItemCategoryKey], 'DW DW_DIM_D_ITEMS_CATEGORY_PROCEDURE'[ItemCategoryKey]))
	% CategoryItemsProcedures	The visual element has a hierarchy with the % CategoryProcedures and % CategoryItemsProcedures, the drill down feature reveals the subcategories
	CareuintProcedures	Calculate(COUNT('DW DW_FACT_PROCEDUREEVENTS_MV'[ItemKey]), USERELATIONSHIP('DW DW_FACT_PROCEDUREEVENTS_MV'[IcustayKey], 'DW DW_DIM_ICUSTAYS'[IcustayKey]), USERELATIONSHIP('DW DW_DIM_ICUSTAYS'[LastCareUnitKey], 'DW DW_DIM_D_CARE_UNITS'[CareunitKey]), FILTER('DW DW_DIM_D_CARE_UNITS', 'DW DW_DIM_D_CARE_UNITS'[CareunitKey]<>-1))
	TypeCaregiverProcedures	Calculate(COUNT('DW DW_FACT_PROCEDUREEVENTS_MV'[ItemKey]), USERELATIONSHIP('DW DW_FACT_PROCEDUREEVENTS_MV'[IcustayKey], 'DW DW_DIM_CAREGIVERS'[CaregiverKey]), FILTER('DW DW_DIM_CAREGIVERS', 'DW DW_DIM_CAREGIVERS[CaregiverKey]<>-1)
	CaregiverProcedures	The visual element has a hierarchy with the % TypeCaregiverProcedures and % CaregiverProcedures, the drill down feature reveals the subcategories
Electronic Chart	% CategoryChart	Calculate(COUNT('DW DW_FACT_CHARTEVENTS'[ItemKey]), USERELATIONSHIP('DW DW_FACT_CHARTEVENTS_MV'[ItemKey], 'DW DW_DIM_D_ITEMS_CHART'[ItemKey]), USERELATIONSHIP('DW DW_DIM_D_ITEMS_CHART'[ItemCategoryKey], 'DW DW_DIM_D_ITEMS_CATEGORY_CHART'[ItemCategoryKey]))
	% CategoryItemsChart	The visual element has a hierarchy with the % CategoryChart and % CategoryItemsChart, the drill down feature reveals the subcategories
	CareuintChart	Calculate(COUNT('DW DW_FACT_CHARTEVENTS'[ItemKey]), USERELATIONSHIP('DW DW_FACT_CHARTEVENTS'[IcustayKey], 'DW DW_DIM_ICUSTAYS'[IcustayKey]), USERELATIONSHIP('DW DW_DIM_ICUSTAYS'[LastCareUnitKey], 'DW DW_DIM_D_CARE_UNITS'[CareunitKey]), FILTER('DW DW_DIM_D_CARE_UNITS', 'DW DW_DIM_D_CARE_UNITS'[CareunitKey]<>-1))
	TypeCaregiverChart	Calculate(COUNT('DW DW_FACT_CHARTEVENTS'[ItemKey]), USERELATIONSHIP('DW DW_FACT_CHARTEVENTS'[IcustayKey], 'DW DW_DIM_CAREGIVERS'[CaregiverKey]), FILTER('DW DW_DIM_CAREGIVERS', 'DW DW_DIM_CAREGIVERS[CaregiverKey]<>-1)
	CaregiverChart	The visual element has a hierarchy with the % TypeCaregiverChart and % CaregiverChart, the drill down feature reveals the subcategories
Microbiology	% CategorySpecimen	Calculate(COUNT('DW DW_FACT_MICROBIOLOGYEVENTS'[SpecimenKey]), USERELATIONSHIP('DW DW_FACT_MICROBIOLOGYEVENTS_MV'[SpecimenKey], 'DW DW_DIM_D_ITEMS_MICROBIOLOGY_SPECIMEN'[ItemKey]), FILTER('DW DW_DIM_D_ITEMS_MICROBIOLOGY_SPECIMEN', 'DW DW_DIM_D_ITEMS_MICROBIOLOGY_SPECIMEN'[ItemKey]<>-1))
	AntibioticPrice	Calculate(COUNT('DW DW_FACT_MICROBIOLOGYEVENTS'[AntibioticKey])* SUM('DW DW_DIM_D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM'[ItemPrice])*5 /SUM('DW DW_DIM_D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM'[ItemQuantity]), USERELATIONSHIP('DW DW_DIM_D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM'[ItemKey], 'DW DW_FACT_MICROBIOLOGYEVENTS'[AntibioticKey]), FILTER('DW DW_DIM_D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM', 'DW DW_DIM_D_ITEMS_MICROBIOLOGY_ANTIBACTERIUM'[ItemKey]<>-1))
Laboratory	% CategoryLaboratory	Calculate(COUNT('DW DW_FACT_CHARTEVENTS'[ItemKey]), USERELATIONSHIP('DW DW_FACT_CHARTEVENTS_MV'[ItemKey], 'DW DW_DIM_D_ITEMS_CHART'[ItemKey]), USERELATIONSHIP('DW DW_DIM_D_ITEMS_CHART'[ItemCategoryKey], 'DW DW_DIM_D_ITEMS_CATEGORY_CHART'[ItemCategoryKey]))
	% CategoryItemsLaboratory	The visual element has a hierarchy with the % CategoryChart and % CategoryItemsChart, the drill down feature reveals the subcategories

*Time 1st Test = MIN(MIN(MIN('Admission Events'[Time Admission Chart], 'Admission Events'[Time Admission Lab]), 'Admission Events'[Time Admission Micro]), 'Admission Events'[Time Admission Proced])

**Date Difference = CALCULATE(1.0*(MIN('DW DW_DIM_ADMISSIONS'[PatientTodisch])-MIN('DW DW_DIM_ADMISSIONS'[PatientToadmit])), USERELATIONSHIP('DW DW_DIM_ADMISSIONS'[AdmissionKey], 'DW DW_FACT_SERVICES'[AdmissionKey]))

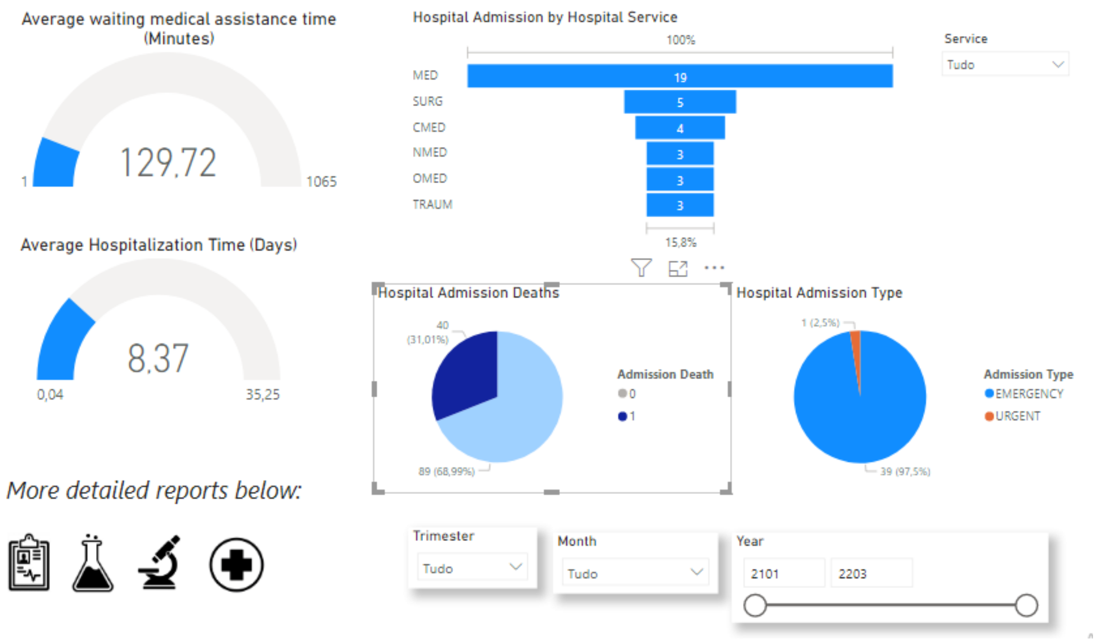


Figure A.1: General Admissions Page. Admissions that resulted in hospital death.

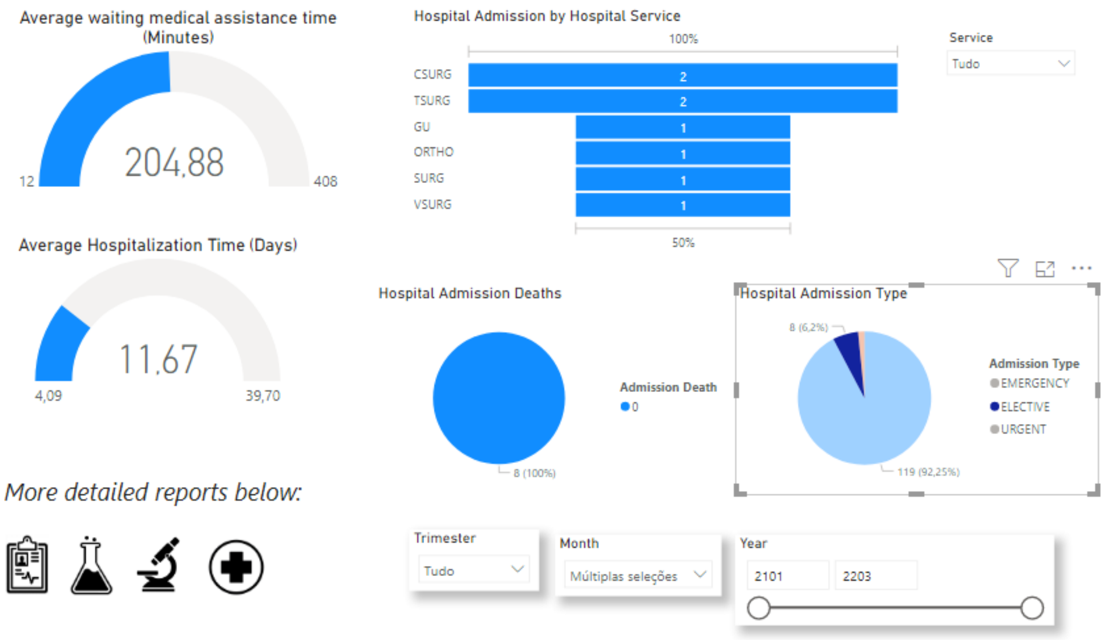
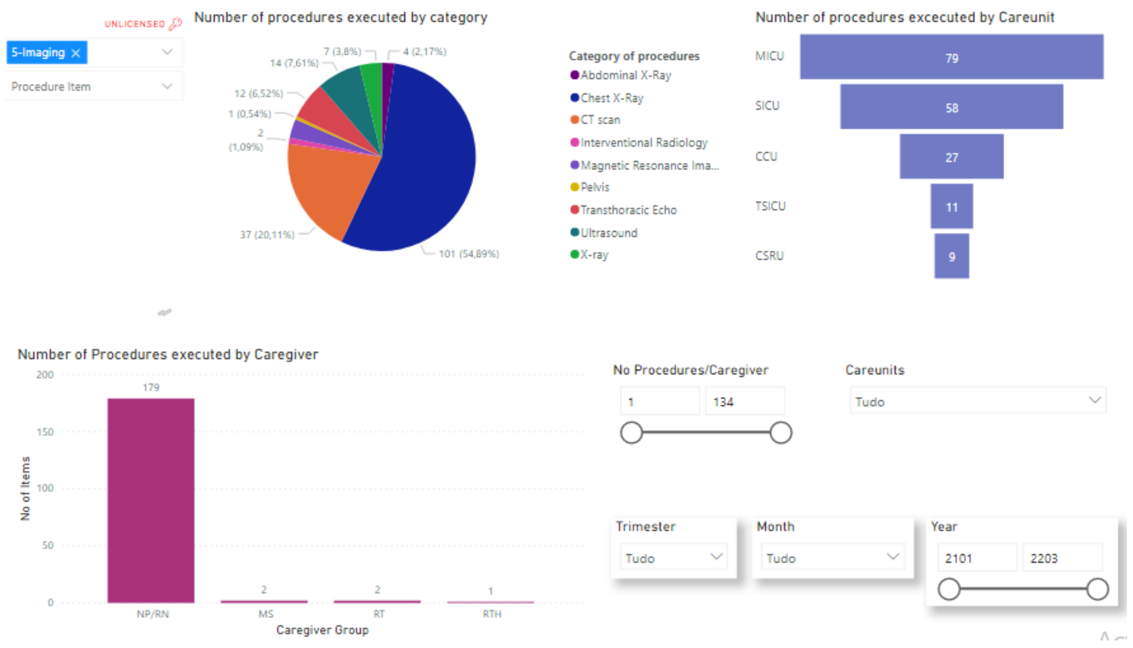
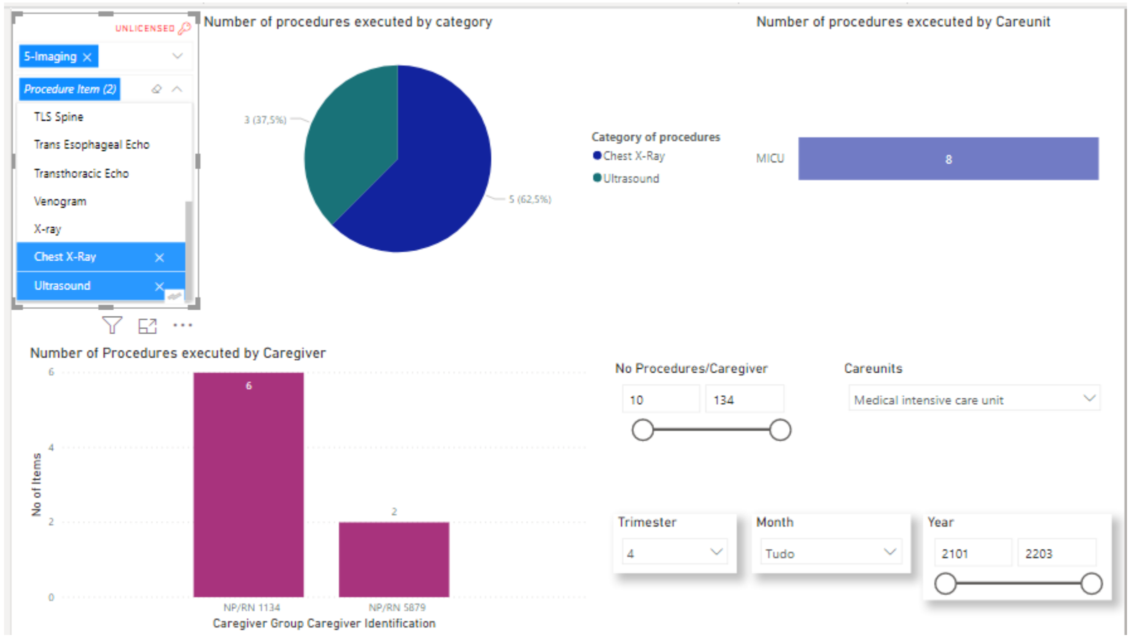


Figure A.2: General Admissions Page filtered by selective admissions



5-Imaging
Procedure Item
No Procedures/Caregiver: 1 – 134
Careunits: Tudo
Trimester: Tudo
Month: Tudo
Year: 2101 – 2203

Figure A.3: Procedures Page Overview filtered with the medical intervention category Imaging.


UNLICENSED
Number of procedures executed by category
3 (37,5%)
5 (62,5%)
Category of procedures
● Chest X-Ray
● Ultrasound
Number of procedures excecuted by Careunit
MICU 8
Number of Procedures executed by Caregiver
No of Items
6
4
2
0
6
2
NP/RN 1134
NP/RN 5879
Caregiver Group Caregiver Identification

5-Imaging
Procedure Item (2)
TLS Spine
Trans Esophageal Echo
Transthoracic Echo
Venogram
X-ray
Chest X-Ray
Ultrasound
No Procedures/Caregiver: 10 – 134
Careunits: Medical intensive care unit
Trimester: 4
Month: Tudo
Year: 2101 – 2203

Figure A.4: Procedures Page Overview. Medical interventions: X-Ray, Ultrasound from the Imaging category. Care unit: MICU. Date: 4th trimester between 2101 and 2203

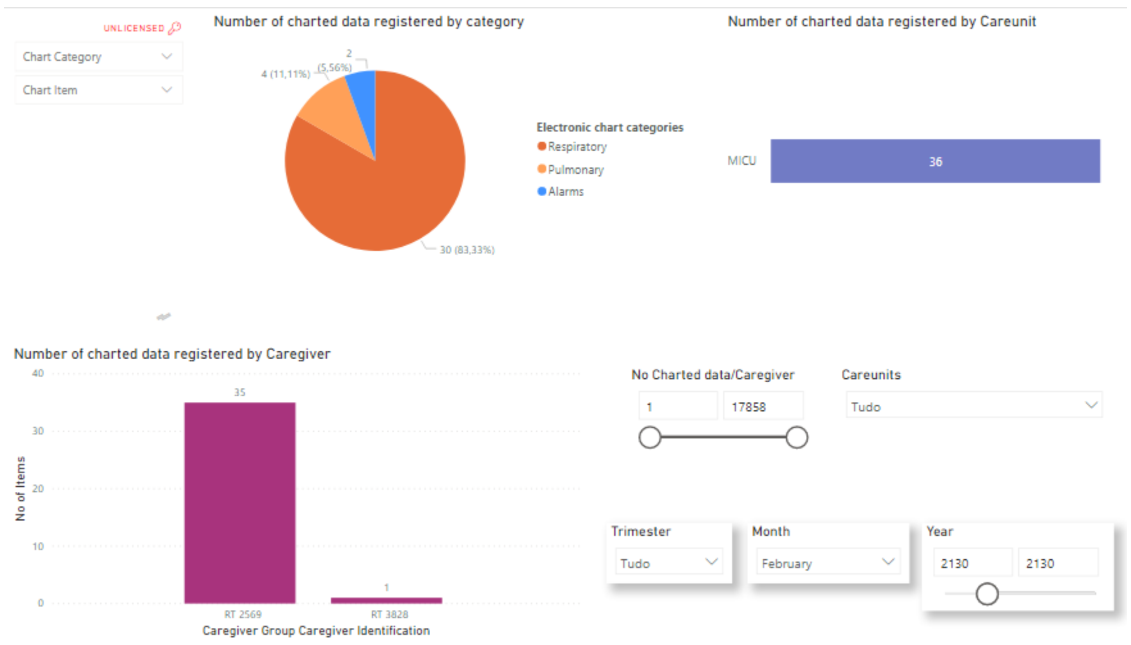

UNLICENSED
Chart Category
Chart Item
Number of charted data registered by category
2 (5,56%)
4 (11,11%)
30 (83,33%)
Electronic chart categories
● Respiratory
● Pulmonary
● Alarms
Number of charted data registered by Careunit
MICU
36
Number of charted data registered by Caregiver
40
30
20
10
0
No of Items
35
1
RT 2569
RT 3828
Caregiver Group Caregiver Identification
No Charted data/Caregiver
1
17858
Careunits
Tudo
Trimester
Tudo
Month
February
Year
2130
2130

Figure A.5: Chart Page Overview of the Respiratory Therapist, on February 2130.

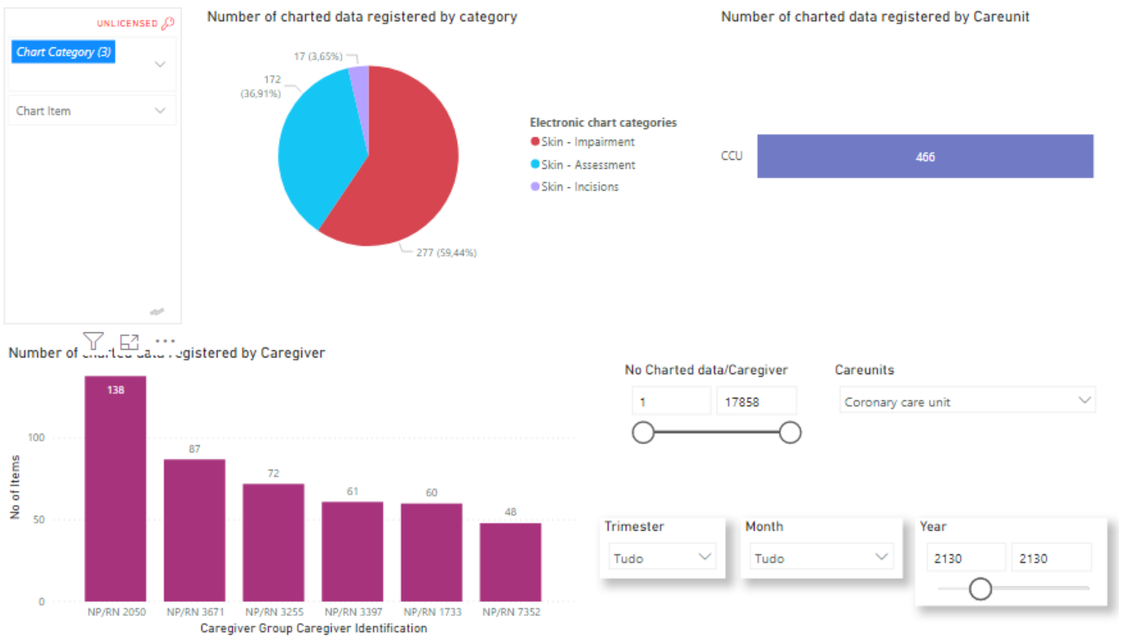


Figure A.6: Chart Page Overview of the Nurses working at the Coronary Care Unit, in 2130, within the Skin categories.

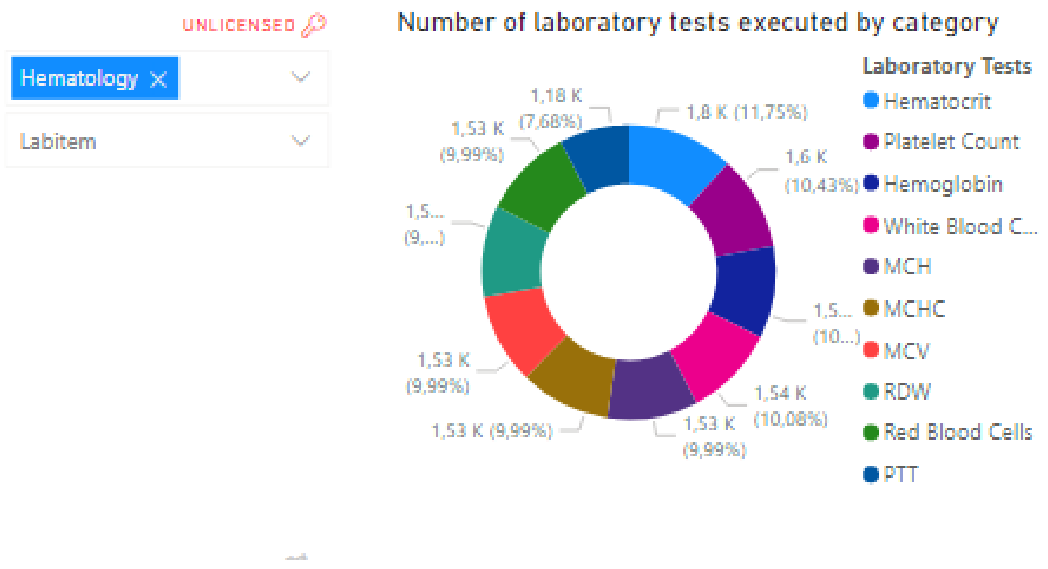
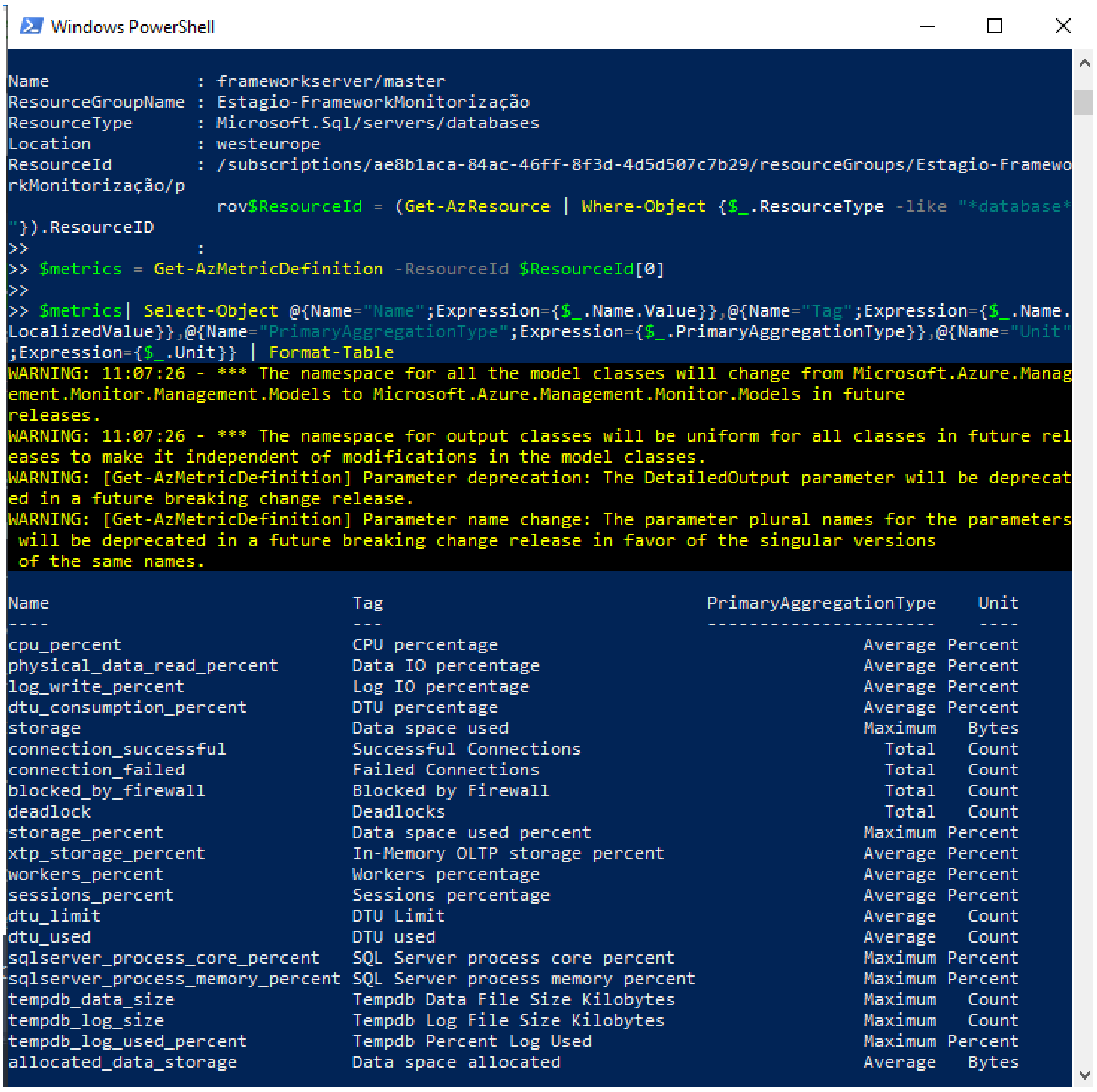


Figure A.7: Laboratory Hematology Tests



```
Name              : frameworkserver/master
ResourceGroupName : Estagio-FrameworkMonitorização
ResourceType      : Microsoft.Sql/servers/databases
Location          : westeurope
ResourceId        : /subscriptions/ae8b1aca-84ac-46ff-8f3d-4d5d507c7b29/resourceGroups/Estagio-Framewo
rkMonitorização/p
                    rov$ResourceId = (Get-AzResource | Where-Object {$_.ResourceType -like "*database*
"}).ResourceID
>>                :
>> $metrics = Get-AzMetricDefinition -ResourceId $ResourceId[0]
>>
>> $metrics| Select-Object @{Name="Name";Expression={$_.Name.Value}},@{Name="Tag";Expression={$_.Name.
LocalizedValue}},@{Name="PrimaryAggregationType";Expression={$_.PrimaryAggregationType}},@{Name="Unit"
;Expression={$_.Unit}} | Format-Table
WARNING: 11:07:26 - *** The namespace for all the model classes will change from Microsoft.Azure.Manag
ement.Monitor.Management.Models to Microsoft.Azure.Management.Monitor.Models in future
releases.
WARNING: 11:07:26 - *** The namespace for output classes will be uniform for all classes in future rel
eases to make it independent of modifications in the model classes.
WARNING: [Get-AzMetricDefinition] Parameter deprecation: The DetailedOutput parameter will be deprecat
ed in a future breaking change release.
WARNING: [Get-AzMetricDefinition] Parameter name change: The parameter plural names for the parameters
 will be deprecated in a future breaking change release in favor of the singular versions
 of the same names.

Name                             Tag                                  PrimaryAggregationType    Unit
----                             ---                                  ----------------------    ----
cpu_percent                      CPU percentage                                      Average Percent
physical_data_read_percent       Data IO percentage                                  Average Percent
log_write_percent                Log IO percentage                                   Average Percent
dtu_consumption_percent          DTU percentage                                      Average Percent
storage                          Data space used                                     Maximum   Bytes
connection_successful            Successful Connections                                Total   Count
connection_failed                Failed Connections                                    Total   Count
blocked_by_firewall              Blocked by Firewall                                   Total   Count
deadlock                         Deadlocks                                             Total   Count
storage_percent                  Data space used percent                             Maximum Percent
xtp_storage_percent              In-Memory OLTP storage percent                      Average Percent
workers_percent                  Workers percentage                                  Average Percent
sessions_percent                 Sessions percentage                                 Average Percent
dtu_limit                        DTU Limit                                           Average   Count
dtu_used                         DTU used                                            Average   Count
sqlserver_process_core_percent   SQL Server process core percent                     Maximum Percent
sqlserver_process_memory_percent SQL Server process memory percent                   Maximum Percent
tempdb_data_size                 Tempdb Data File Size Kilobytes                     Maximum   Count
tempdb_log_size                  Tempdb Log File Size Kilobytes                      Maximum   Count
tempdb_log_used_percent          Tempdb Percent Log Used                             Maximum Percent
allocated_data_storage           Data space allocated                                Average   Bytes
```

Figure A.8: Azure Database Resource Metrics

References

[1] Shahidul Islam Khan and Abu Sayed Latiful Hoque. Privacy and security problems of national health data warehouse: a convenient solution for developing countries. *2016 International Conference on Networking Systems and Security (NSysS)*, pages 151–152, Jan 2016. doi:10.1109/NSysS.2016.7400708.

[2] Maria Antonina Mach and M. Salem Abdel-Badeeh. Intelligent techniques for business intelligence in healthcare. *2010 10th International Conference on Intelligent Systems Design and Applications*, pages 545–550, Nov 2010. doi:10.1109/ISDA.2010.5687209.

[3] Mu-Hsing Kuo, Dillon Chrimes, Belaid Moa, and Wei Hu. Design and Construction of a Big Data Analytics Framework for Health Applications. *ResearchGate*, pages 631–636, Dec 2015. doi:10.1109/SmartCity.2015.140.

[4] SONHO - aprendis, May 2018. [Online; accessed 1. Sep. 2020]. URL: http://aprendis.gim.med.up.pt/index.php/SONHO#SONHO_v2.

[5] Interoperabilidade Técnica: LIGHt; PNB; NCP – SPMS, Sep 2020. [Online; accessed 1. Sep. 2020]. URL: http://www.spms.min-saude.pt/2017/06/interoperabilidade-tecnica-light-pnb-ncp.

[6] Ana Paula Pinheiro. Os sistemas de informação na prática do médico de família: onde está a interoperabilidade? *Revista Portuguesa de Medicina Geral e Familiar*, 34(4):250–4, Jul 2018. doi:10.32385/rpmgf.v34i4.12486.

[7] Paulo Silva, César Quintas, Júlio Duarte, Manuel Santos, José Neves, António Abelha, and José Manuel Machado. Hospital database workload and fault forecasting. *IEEE*, 2012. doi:10.1109/IECBES.2012.6498150.

[8] What is business intelligence? Your guide to BI and why it matters, Feb 2020. [Online; accessed 05. Feb. 2020]. URL: https://www.tableau.com/learn/articles/business-intelligence.

[9] BNP - Business intelligence, Feb 2020. [Online; accessed 11. Feb. 2020]. URL: http://bibliografia.bnportugal.gov.pt/bnp/bnp.exe/registo?1694528.

[10] H. P. Luhn. A Business Intelligence System. *IBM J. Res. Dev.*, 2(4):314–319, Oct 1958. doi:10.1147/rd.24.0314.

[11] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. Wiley, Apr 2002. URL: https://www.amazon.com/Data-Warehouse-Toolkit-Complete-Dimensional/dp/0471200247.

[12] Stanisław Kozielski and Robert Wrembel. *New Trends in Data Warehousing and Data Analysis | Stanisław Kozielski | Springer*. Springer US, 2009. doi:10.1007/978-0-387-87431-9.

[13] William H. Inmon. *Building the Data Warehouse Fourth Edition*. Wiley, Oct 2005. URL: http://fit.hcmute.edu.vn/Resources/Docs/SubDomain/fit/ThayTuan/DataWH/Bulding%20the%20Data%20Warehouse%204%20Edition.pdf.

[14] Fact Tables and Dimension Tables, Jan 2003. [Online; accessed 11. Feb. 2020]. URL: https://www.kimballgroup.com/2003/01/fact-tables-and-dimension-tables.

[15] Razi O. Mohammed and Samani A. Talab. Clinical Data Warehouse Issues and Challenges. *International Journal of u- and e-Service, Science and Technology*, 7(5):251–262, Oct 2014. doi:10.14257/ijunesst.2014.7.5.22.

[16] João Ferreira, Miguel Miranda, António Abelha, and José Machado. O Processo ETL em Sistemas Data Warehouse, Feb 2020. [Online; accessed 13. Feb. 2020]. URL: https://docplayer.com.br/497244-O-processo-etl-em-sistemas-data-warehouse.html.

[17] Krish Krishnan. *Data Warehousing in the Age of Big Data (The Morgan Kaufmann Series on Business Intelligence)*. Morgan Kaufmann, May 2013.

[18] Vasco Santos and Orlando Belo. No Need to Type Slowly Changing Dimensions. *ResearchGate*, pages 129–136, Mar 2011. URL: https://www.researchgate.net/publication/259293002_No_Need_to_Type_Slowly_Changing_Dimensions.

[19] Victor Chang, Yen-Hung Kuo, and Muthu Ramachandran. Cloud computing adoption framework: A security framework for business clouds. *Future Gener. Comput. Syst.*, 57:24–41, Apr 2016. doi:10.1016/j.future.2015.09.031.

[20] Cloud Computing and Business Intelligence Market Study, Feb 2020. [Online; accessed 11. Feb. 2020]. URL: https://www.martechcube.com/whitepapers/cloud-computing-and-business-intelligence-market-study.

[21] John Brandon. What is Infrastructure-as-a-Service? Everything you need to know about IaaS. *TechRadar*, Dec 2019. URL: https://www.techradar.com/news/what-is-infrastructure-as-a-service.

[22] Gurudatt Kulkarni, Prasad Khatawkar, and Jayant Gambhir. Cloud Computing-Platform as Service. *ResearchGate*, -1(-2):115–120, Dec 2011. URL: https://www.researchgate.net/publication/234166315_Cloud_Computing-Platform_as_Service.

[23] Mihaela Muntean. Considerations Regarding Business Intelligence in Cloud Context. *ResearchGate*, 19(4/2015):55–67, Dec 2015. doi:10.12948/issn14531305/19.4.2015.05.

[24] Cameron Fisher. Cloud versus On-Premise Computing. *American Journal of Industrial and Business Management*, 08(09):1991–2006, Jan 2018. doi:10.4236/ajibm.2018.89133.

[25] What is AWS, Feb 2020. [Online; accessed 14. Feb. 2020]. URL: https://aws.amazon.com/what-is-aws.

[26] Cloud Computing Services | Microsoft Azure, Feb 2020. [Online; accessed 14. FeB. 2020]. URL: https://azure.microsoft.com/en-us.

[27] Google Cloud, Feb 2020. [Online; accessed 14. Feb. 2020]. URL: https://cloud.google.com.

[28] Cloud Services Terminology Guide: Comparing AWS vs Azure vs Google | CloudHealth by VMware, Feb 2020. [Online; accessed 14. Feb. 2020]. URL: https://www.cloudhealthtech.com/blog/cloud-comparison-guide-glossary-aws-azure-gcp.

[29] Comparing Cloud Instance Pricing: AWS vs Azure vs Google vs IBM | Flexera Blog, Nov 2017. [Online; accessed 16. Feb. 2020]. URL: https://www.flexera.com/blog/cloud/2017/11/comparing-cloud-instance-pricing-aws-vs-azure-vs-google-vs-ibm.

[30] Tableau Server Pricing, Demo, Reviews, Features, Feb 2020. [Online; accessed 15. Feb. 2020]. URL: https://www.selecthub.com/business-intelligence-tools/tableau-server/?from_category=69.

[31] Qlikview Pricing - How Much Does Qlikview Cost, Feb 2020. [Online; accessed 15. Feb. 2020]. URL: https://www.selecthub.com/big-data-analytics-tools/qlikview.

[32] Power BI vs Tableau vs Qlikview | Top 6 Differences (with Infographics), Nov 2019. [Online; accessed 15. Feb. 2020]. URL: https://www.wallstreetmojo.com/power-bi-vs-tableau-vs-qlikview.

[33] Power BI Pricing - Microsoft Power BI Price Info, Feb 2020. [Online; accessed 15. Feb. 2020]. URL: https://www.selecthub.com/business-intelligence-tools/microsoft-bi.

[34] Himani Bansal. Tableau vs Qlik Sense vs Power BI—Choose best BI Tool for Big Data Visualization. *Medium*, Jun 2019. URL: https://medium.com/javarevisited/tableau-vs-qlik-sense-vs-power-bi-choose-best-bi-tool-for-big-data-visualiz

[35] Azure SQL Database, Feb 2020. [Online; accessed 12. Feb. 2020]. URL: https://azure.microsoft.com/en-us/services/sql-database/#product-overview.

[36] tamram. Introduction to Azure Storage - Cloud storage on Azure, Sep 2020. [Online; accessed 26. Sep. 2020]. URL: https://docs.microsoft.com/en-us/azure/storage/common/storage-introduction.

[37] Data Factory - Data Integration Service | Microsoft Azure, Feb 2020. [Online; accessed 12. Feb. 2020]. URL: https://azure.microsoft.com/en-us/services/data-factory/#overview.

[38] Logic App Service | Microsoft Azure, Feb 2020. [Online; accessed 12. Feb. 2020]. URL: https://azure.microsoft.com/en-us/services/logic-apps/#features.

[39] Pricing - Functions | Microsoft Azure, Sep 2020. [Online; accessed 26. Sep. 2020]. URL: https://azure.microsoft.com/en-us/pricing/details/functions.

[40] MIMIC-III Clinical Database v1.4, Aug 2020. [Online; accessed 22. Aug. 2020]. URL: https://physionet.org/content/mimiciii/1.4.

[41] Alistair E. W. Johnson, Tom J. Pollard, Lu Shen, Li-Wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. MIMIC-III, a freely accessible critical care database. *Sci. Data*, 3:160035., May 2016. arXiv:27219127, doi:10.1038/sdata.2016.35.

[42] Tom J. Pollard, Alistair E. W. Johnson, Jesse D. Raffa, Leo A. Celi, Roger G. Mark, and Omar Badawi. The eICU Collaborative Research Database, a freely available multi-center database for critical care research. *Sci. Data*, 5(180178):1–13, Sep 2018. doi:10.1038/sdata.2018.178.

[43] Somnath Bose, Alistair E. W. Johnson, Ari Moskowitz, Leo Anthony Celi, and Jesse D. Raffa. Impact of Intensive Care Unit Discharge Delays on Patient Outcomes: A Retrospective Cohort Study. *J. Intensive Care Med.*, 34(11-12):924–929, Oct 2018. doi:10.1177/0885066618800276.

[44] MIMIC-III Clinical Database Demo v1.4, Apr 2019. [Online; accessed 23. Aug. 2020]. URL: https://physionet.org/content/mimiciii-demo/1.4.

[45] Jan Hudzicki. Kirby-Bauer Disk Diffusion Susceptibility Test Protocol. *American Society of Microbiology*, Dec 2009. URL: https://www.asmscience.org/content/education/protocol/protocol.3189.

[46] Contributors to Wikimedia projects. Thermo Fisher Scientific - Wikipedia, Aug 2020. [Online; accessed 1. Sep. 2020]. URL: https://en.wikipedia.org/w/index.php?title=Thermo_Fisher_Scientific&oldid=975761270.

[47] Susceptibility Testing Discs, Sep 2020. [Online; accessed 1. Sep. 2020]. URL: https://www.fishersci.com/us/en/browse/90222069/susceptibility-testing-discs.

[48] Health System Analytics: The Missing Key to Unlock Value-based Care, Sep 2020. [Online; accessed 27. Sep. 2020]. URL: https://www2.deloitte.com/us/en/pages/life-sciences-and-health-care/articles/health-system-analytics.html.

[49] davidiseminger. Using DirectQuery in Power BI - Power BI, Sep 2020. [Online; accessed 16. Sep. 2020]. URL: https://docs.microsoft.com/en-us/power-bi/connect-data/desktop-directquery-about.

[50] peter myers. DirectQuery model guidance in Power BI Desktop - Power BI, Sep 2020. [Online; accessed 16. Sep. 2020]. URL: https://docs.microsoft.com/en-us/power-bi/guidance/directquery-model-guidance.

[51] *Effective Monitoring and Alerting*. O'Reilly Media, Inc., 2016. URL: https://www.oreilly.com/library/view/effective-monitoring-and/9781449333515/ch01.html.

[52] Johnny Long, Bill Gardner, and Justin Brown. Chapter 4 - Document Grinding and Database Digging. In *Google Hacking for Penetration Testers (Third Edition)*, pages 61–78. Syngress, Jan 2016. doi:10.1016/B978-0-12-802964-0.00004-0.

[53] SendGrid Documentation, Sep 2020. [Online; accessed 23. Sep. 2020]. URL: https://sendgrid.com/docs.

[54] Web API Design The Missing Link Best Practices for Crafting Interfaces that Developers Love | Course Hero, Sep 2018. [Online; accessed 23. Sep. 2020]. URL: https://pages.apigee.com/rs/351-WXY-166/images/Web-design-the-missing-link-ebook-2016-11.pdf.

[55] HP Performance Engineering Best Practices Series - PDF Free Download, Jun 2015. [Online; accessed 22. Sep. 2020]. URL: https://docplayer.net/8196271-Hp-performance-engineering-best-practices-series.html.

[56] Jeffery Hicks, Aleksandar Nikolic, Richard Siddaway, and Oisin Grehan. *PowerShell Deep Dives*. Manning Publications, Jul 2013. URL: https://livebook.manning.com/book/powershell-deep-dives/about-this-book/34.

[57] Pricing - Automation | Microsoft Azure, Sep 2020. [Online; accessed 24. Sep. 2020]. URL: https://azure.microsoft.com/en-us/pricing/details/automation.