UNIVERSIDADE D
COIMBRA

Sergii-Zinovii Mykolyshyn

Data Models for Edge Computing

VOLUME 1

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software Engineering, advised by Professor Vasco Pereira and Professor Bruno Cabral and presented to Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2021

Faculty of Sciences and Technology

Department of Informatics Engineering

Data Models for Edge Computing

Sergii-Zinovii Mykolyshyn

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software Engineering advised by Prof. Vasco Pereira and Prof. Bruno Cabral and presented to the Faculty of Sciences and Technology / Department of Informatics Engineering

September 2021

UNIVERSIDADE D
COIMBRA

This page is intentionally left blank.

Acknowledgements

This report describes the performed work during 2020/2021 internship in the Department of Informatics Engineering (DEI), from University of Coimbra. This is the final step of the Master's course, which englobes all the academic experience acquired in this institution.

I would like to thank my parents for giving me the chance to study in the best place I could imagine, for believing in me, for teaching me to never give up on my dreams and to work hard to accomplish my goals.

I would also like to thank Professor Vasco Pereira, for his availability, clarifying my doubts, for every feedback, for guiding me during this phase, and for helping elaborating the scientific paper and this report. Many thanks to Professor Bruno Cabral for clarifying my doubts and for giving objective and clear feedback of what can be changed and improved in the application, always with a lot of ideas. Also to Professor Jorge Bernardino for sharing his knowledge, helping with the report elaboration and on the writing of the Data Modeling tools paper, it was a big pleasure.

To Gonçalo Carvalho for being a very good colleague who became a friend, for sharing the experiences, helping every time he could, giving his best explaining my doubts, giving a lot of feedback on this report elaboration, and by helping a lot in the scientific paper.

To my friends, from the school, and who continue by my side until today, by supporting me and being with me in good and worse times. To the friends I made in the University and became very close during this course.

To my girlfriend Lígia, who is one of the best people I met until today. I can not express in words my gratitude to you, for being very kind, understandable, pushing me to give my best, and being there for me when I most need you.

Finally, to everyone who I met during this University journey and who somehow changed my life and lead me to where and who I am today.

To all of you, an enormous thank you.

This page is intentionally left blank.

Abstract

Edge Computing (EC) is an architecture paradigm that brings computation closer to end users, with the aim of reducing latencies, bandwidth consumption, and also achieving greater reliability, compared to a Cloud architecture.

The objective of this work is to collaborate in the extension of a database platform o enable the databases modeling at Edge and Cloud levels simultaneously, maintaining total transparency for the end-user.

By placing the database on the Edge, performing aggregation and summarization functions on the collected data, before sending it to the Cloud, it is expected to achieve the benefits of EC. The platform should be able to perform this data transformation on the database located on the Edge and convert it to a single entity data warehouse located on the Cloud, which aggregates the whole data, aiming to increase the database performance.

To reach this goal, it was acquired in-depth knowledge about EC and how Data Models can be used with these systems, analyzed and evaluated existing Data Modeling tools, with the aim of identifying differences between each and choose one tool to extend.

The produced outcome is a tool that helps to transform a classical single-layer database design from an Entity Relationship (ER) diagram, into a multi-layer system with the original database located on the Edge and a data warehouse on the Cloud. In addition, the script is automatically generated, with all the data summarized. This approach will reduce error's probability, time spent in creating scripts manually, and make them modular, by adapting it to any use case.

Keywords

Data Modeling, Edge Computing, Databases, Internet of Things, Cloud Computing, Entity-Relationship

This page is intentionally left blank.

Resumo

EC é um paradigma de arquitetura que aproxima a computação dos utilizadores finais, com o objetivo de reduzir latências, consumo de largura de banda e também alcançar maior confiabilidade, comparativamente a uma arquitetura na nuvem.

O objetivo deste trabalho é colaborar na extensão de uma plataforma de base de dados para possibilitar a modelação de bases de dados nos níveis da Edge e da Cloud simultaneamente, mantendo a total transparência para o utilizador final.

Ao colocar a base de dados na Edge, realizando funções de agregação e sumarização nos dados recolhidos, antes de enviá-los para a nuvem, espera-se obter os benefícios da EC. A plataforma deve ser capaz de realizar essa transformação de dados localizados na Edge e convertê-los em uma data warehouse de entidade única localizada na nuvem, que agrega todos os dados, visando aumentar o desempenho da base de dados.

Para alcançar este objetivo, realizaram-se pesquisas aprofundadas sobre a EC e como também sobre os Modelos de Dados que podem ser utilizados com estes sistemas, analisaram-se e foram avaliadas as ferramentas de Modelação de Dados existentes, com o objetivo de identificar as diferenças entre cada uma e escolher uma ferramenta para extender.

O resultado produzido é uma plataforma que ajuda a transformar um design clássico de base de dados de camada única de um diagrama ER, num sistema multicamadas com a base de dados original localizada na Edge e uma data warehouse na nuvem. Além disso, o script é gerado automaticamente, com todos os dados sumarizados e agregados. Essa abordagem reduzirá a probabilidade de erro, o tempo gasto na criação manual de scripts e torná-los-á modulares, adaptando-os a qualquer caso de uso.

Palavras-Chave

Modelação de dados, Computação na Edge, Bases de dados, Internet das Coisas, Computação na nuvem, Entidade-Relacionamento

This page is intentionally left blank.

Contents

This page is intentionally left blank.

Acronyms

AI Auto Incremet. 9, 14–20

AP Access Points. 5, 6

BRR Business Readiness Rating. 20

CC Cloud Computing. 1, 4, 5, 21

CDM Conceptual Data Modeling. 12

CH Check. 14, 16, 17

CM Conceptual Modeling. 11–20

DEI Department of Informatics Engineering. iii, 1, 7, 14, 38, 43, 58, 59

EC Edge Computing. v, vii, 1, 2, 4, 5, 21, 24

ER Entity Relationship. v, vii, 1, 7, 8, 11, 12, 14, 21, 25, 32

FK Foreign Key. 12, 14–20, 38

IoT Internet of Things. 5, 7, 58

MEC Multi-access Edge Computing. 5, 6

NN Not Null. 9, 13–20

OO Object-Oriented. 12

PK Primary Key. 9, 12, 14–20

SQL Structured Query Language. 9, 13–15, 18, 32, 38, 40, 49, 56, 58

UML Unified Model Language. 12, 16, 18

UQ Unique. 14–20

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

This dissertation under the Master's program in Informatics Engineering, named Data Models for Edge Computing, describes the developed work during the first and second semesters of the school year 2020/2021. The thesis was advised by Professor Vasco Pereira and Professor Bruno Cabral from the Department of Informatics Engineering (DEI) of the University of Coimbra.

1.1 Motivation

Cloud Computing (CC) services prevailed in the last decade by bringing cost-effectiveness, speed, and low response time. The number of devices connected to the Cloud is growing, and the challenge to deliver data in a reasonable period and without losses is getting harder to accomplish. Edge Computing (EC) paradigm has been emerging in the last years, helping to solve problems that CC cannot anymore. It brings the computation closer to the final user, achieving greater reliability, reducing latency, and reducing bandwidth ingress to store or retrieve the data from the Cloud. In an EC system, data and computing resources are placed closer to the mobile devices and sensors, which have the name of Edge of the Internet [1].

The motivation of this work is to extend an existing platform which uses the Entity Relationship (ER) model and adapt it to create an important tool for the future development of EC systems. The idea is to extend a tool where users can illustrate separate databases for the Edge and the Cloud. The tool must auto-generate a script to retrieve the data and create databases for different engines, like MySQL or PostgreSQL.

Before this work, a platform to represent both the Edge and Cloud layers was nonexistent. Because of this, many different tools were analyzed as potential candidates to be extended, as it will be possible to read in Chapter 2. ONDA[2] was selected because this tool is a property of DEI, works with ER model, is easy to understand the source code and it is possible to keep in contact with its authors easily.

1.2 Objectives

The purpose of this thesis is, first, to become aware of the Edge and Cloud Computing paradigms and the ER model, which will represent databases in these two locations.

Second, is to evaluate the available data modeling tools and the features they possess, to create an overall knowledge of the differences between them.

The third goal is to extend ONDA, by introducing the concept of layers (Edge and Cloud). On the Edge, users must be capable to create entities that store all the information regarding to the database. On the Cloud, the users can create an entity ("Data Summary") to summarize the necessary data, choose an aggregation function to apply on the selected fields, in a defined time window. These data selection and time window options will be implemented in the user interface, and later the tool will translate this information into an auto-generated MySQL or PostgreSQL script. This task is crucial because all the database creation and data aggregation were previously made by hand. This is error prone and makes it difficult to adapt project changes. Current approach tends to minimize the amount of human errors and produce quicker results, by automating these tasks. Also, the existence of such tool that aggregates the data can make a dramatic increase in the performance of the data warehouse by reducing the number of rows to be accessed when computing a query.

Finally, this implementation will be tested in the context of the research, to validate the results and understand what can be improved in the next versions of this application.

1.3 Document Outline

This document is organized as follows: in Chapter 2 are provided concepts searched during the first semester of the thesis, such as EC, data models, and tools to modulate a database. In Chapter 3 are provided the project planning for the internship, how the work was managed, what risks were identified, and how to avoid them. In Chapter 4 are provided the requirements to implement in the new version of the application and it is shown and explained the initial Mock-ups. In Chapter 5 are provided the architecture of the application and it is shown the general flow of the old and new implementations. In Chapter 6 are provided the tool implementation process. In Chapter 7 are provided the tests made to the new implementation and the results are analyzed. Last, Chapter 8 presents the acquired experience, future work, and the final considerations.

1.4 Project Team Constitution

This team is composed by Professor Vasco Pereira, Professor Bruno Cabral, Professor Jorge Bernardino, PhD student Gonçalo Carvalho and Master's student Sergii Mykolyshyn. Together, Professors and Gonçalo helped to write a significant part of the scientific paper named Comparative Analysis of Data Modeling Design Tools. Also, everybody participated in the work planning, requirement gathering, definition of the first project mock-ups, brainstorming with ideas, correction of this thesis, and making sure that everything would go as expected in the work plan.

This page is intentionally left blank.

Chapter 2

State of the art

In this chapter is provided the state-of-the-art related to the Edge Computing (EC) Paradigm and Data Models. In Section 2.1 is explained the background knowledge related to EC. In Section 2.1 are explained the key architectures that enable EC. In Section 2.3 is explained what is a Data Warehouse and how it is related with this thesis. In Section 2.4 is explained what a data model is, its evolution in the past decades, the analyzed Data Modeling tools, and the results of their evaluation. In Section 2.5 is explained the methodology behind tools evaluation and how the evaluation was performed.

This literature review started with an exploration of different academic search engines, such as DBLP, Google Scholar, and IEEE Xplore Digital Library. The search keywords used were: "Edge Computing" with a raise in publications since 2016; "Cloud Computing" which emerged more and more since 2009; "Data Modeling" and "Data Model" - these two search keywords are continuously growing in publications since the middle of 1980.

2.1 Edge Computing Paradigm

EC is a recent paradigm where computing and storage resources are placed at the Internet's edge, very close to mobile devices, sensors and users. Professor Satyanarayanan from the University Carnegie Mellon associates terms as Cloudlets, Microdata centers and Fog nodes to this new paradigm [1]. The EC has its origins near 1999, when Akamai Technologies launched a system that originally delivered Web objects (images and documents) on servers at the network edge, creating Content Delivery Networks and improving web sites scalability, reliability, and performance. It has evolved to distribute pages and applications to the network's edge, providing customers with greater computing capacity [3]. Before the development of EC, traditional Cloud Computing (CC) brought scalability, simplicity, security, fault tolerance, and solved the computing and storage problems in a centralized way [4, 5, 6].

With the growth of CC, practically 90 percent of global Internet users rely on cloud-based services [4]. This approach started to produce problems, such as significant latency, data transmission overhead, loss of privacy, energy consumption, and location limitations [6, 7].

For example, a self-driving car generates 1 Gigabyte of data every second and involves real-time processing to perform correct decisions, and for this, data needs to be processed at the Edge for short response time and efficient processing [8].

The EC paradigm emerged to tackle the CC problems, acting as its complement instead of

its replacement. These two exist together since CC continues important in the development of Internet of Things (IoT) devices, that are getting gradually more intelligent [6].

The next subsection will present the main architectures that permit the EC paradigm to work.

2.2 Key architectures that enable Edge Computing

The Edge Computing paradigm can be classified into three main types, Fog Computing, Multi-access Edge Computing (MEC) and Cloudlet Computing, explained in the next subsections.

2.2.1 Fog Computing

Fog Computing is a virtualized platform that provides computing, storage and networking services between devices and CC Data Centers and brings high quality of service by reducing latency, amount of data traffic, and improving efficiency [9]. This architecture extends CC paradigm to fully support IoT. By doing the processing and storage of data closer to the Edge of the network, it reduces service latency, conserves network bandwidth and speeds up real-time processing [10, 11]. In Fog computing, large numbers of heterogeneous devices communicate and cooperate between themselves, to perform storage and treatment of data without involving third parties [12].

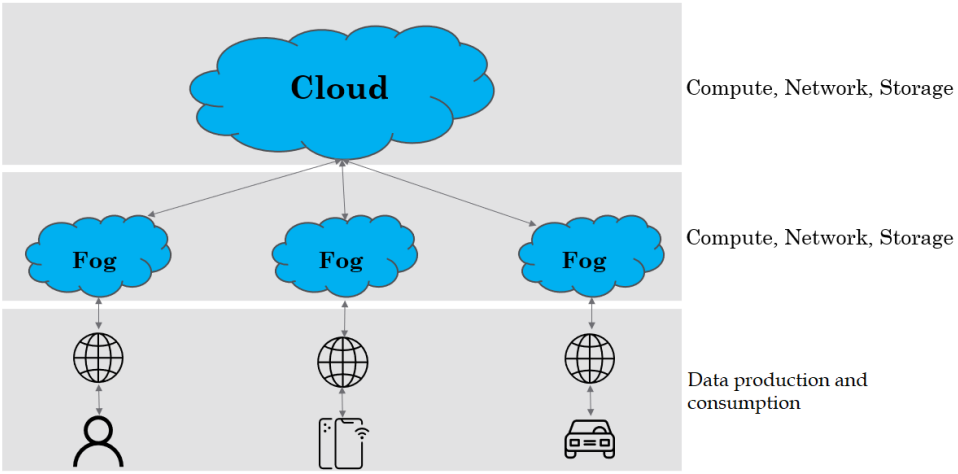


Figure 2.1: Fog Computing Paradigm

2.2.2 Multi-access Edge Computing

Our mobile devices are expected to be small to fit in our pockets. Due to this physical size constraint, the computing power and battery capacity of mobile devices are reduced and the user experience can not be as satisfactory as awaited. The technology evolution lead to new solutions and the computations started to be offloaded to cloud infrastructures, via Access Points (AP) associated with users. Locating computing servers at the AP, close to the users, avoids delays between the cloud and AP.

With the appearance of EC, the network edge is the radio access network and the AP started to be equipped with one more computing server, the mobile edge computing server.

The advantages of such technologies are the proximity to users' device, due to the AP proximity and the improvement in computing and radio resource efficiency, since an AP controls both of them [13].

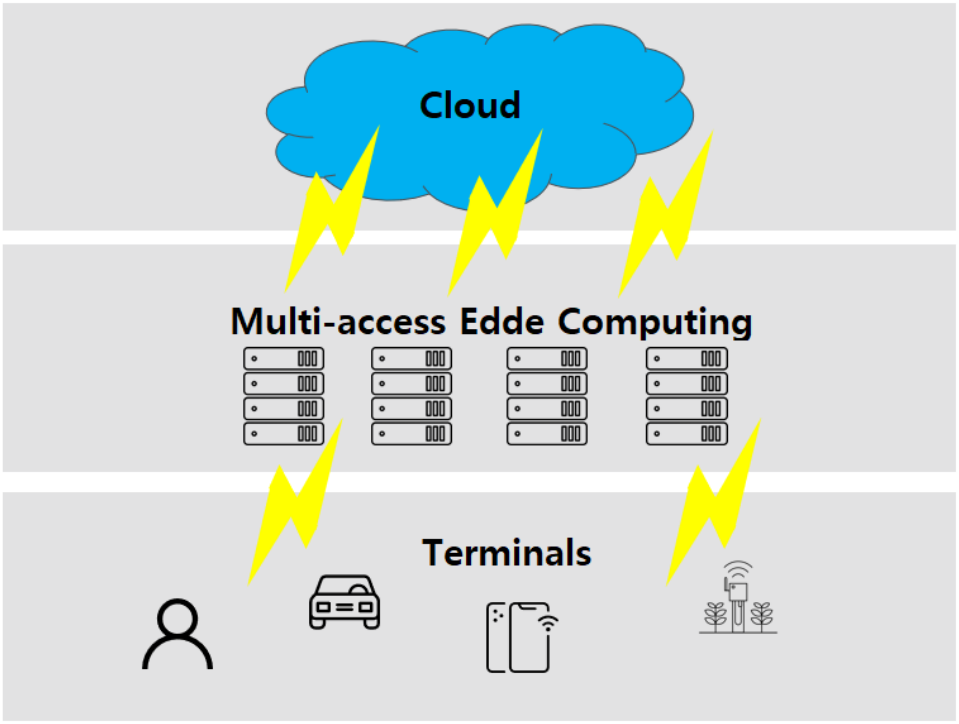


Figure 2.2: Multi-access Edge Computing

2.2.3 Cloudlet Computing

Comparing to MEC and Fog Computing, Cloudlet Computing is a newer paradigm [14] and is "a trusted cluster of computers, well connected to the Internet, with resources available to use for nearby mobile devices" [15]. With this paradigm, it becomes possible to offload mobile applications as mobile transactions and payments to a capable cloudlet such as desktop or laptop in the proximity of the device. This type of technology boosts the application's performance, saves mobile data, and extends mobile device battery life. The key challenge of the Cloudlets is the continuous need to be connected to it, so when a user cannot be around his Cloudlet, he cannot perform such computations [16].

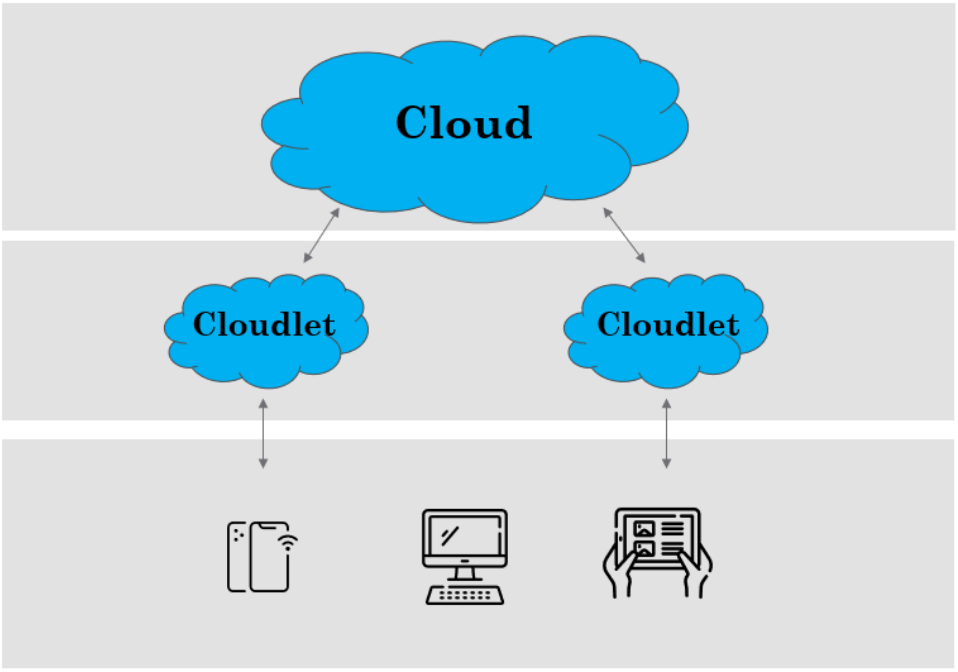


Figure 2.3: Cloudlet Computing

2.3 Data Warehouses

One of the main objectives of this work is to define a construction of a single table data warehouse from the visual representation of an Entity Relationship (ER) diagram. A data warehouse is an integrated and time varying collection of data derived from operational data used in strategic decision making. It is essentially a database that stores integrated, often historical, and aggregated information extracted from multiple, heterogeneous, autonomous, and distributed information sources [17].

In the context of this work, a system of meteorologic data collection related to the city of Coimbra is being developed. IoT sensors located in the Department of Informatics Engineering (DEI) and the data from OpenWeatherMap API are being collected on the network's Edge, near the user. An Edge server gathers this meteorologic data, such as air quality, humidity, temperature. Only parts of the this information will be selected and transformed into a data warehouse. This data warehouse is represented by a single entity named "Data Summary". The main goal is to store this data grouped and organized on the Cloud, for further processing by Artificial Intelligence algorithms. The algorithms will make decisions based on the results, such as open/close blinds or turn on/off air conditioning.

Since the data are collected on the Edge layer and transferred to the Cloud, it is important to aggregate them, making the data occupy less space, and reducing bandwidth consumption and time spent transferring it. This means that the collected data are all integrated in one place, the "Data Summary" entity. This new entity will have three features: dimensions, facts, and functions. Dimensions represent the data granularity of aggregation. These will have a default time dimension, which sets the aggregation time window (hour, day, week, or month). The aggregations decrease the amount of rows to be accessed, increasing the database performance. Facts portray the data values of the dimensions'. Functions depict the mathematical operations (aggregation functions) to apply to the data, such as average, count, maximum, minimum, and standard deviation.

Figure 2.4 represents the explained idea with example of the data aggregation about humidity.

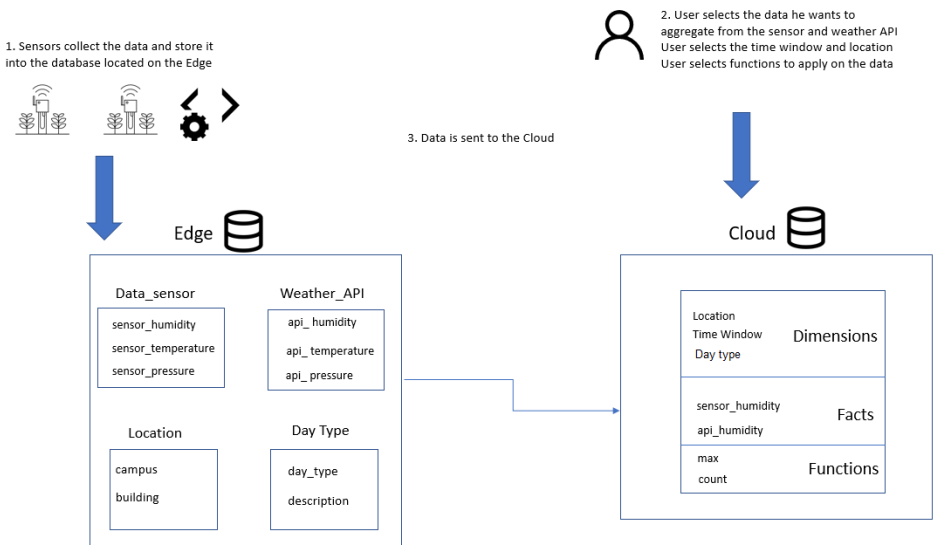


Figure 2.4: Visual representation of the idea

2.4 Data Modeling

This section explains what a data model is and how data models evolved in the past decades. It also outlines the research on data modeling tools and presents tool evaluation.

2.4.1 Data Models and Data Modeling and its importance

A *data model* is a set of concepts that can describe the data structure and operations on a database [18]. These data structures include objects, relations between these objects, and rules that define how data is organized. Determining the business needs will lead to the data model. The business stakeholders' feedback is crucial to define rules and requirements to be incorporated into the design of a new system or adapted in an iteration of an existing one [19]. Data models are adopted to manage and analyze data representing any information system. The data model is an essential element of the system development or database design processes. Although the data modeling phase embodies only a smaller dimension of the development effort, its influence on the eventual result is reasonably broader than any other phase.

Data modeling creates a visual representation of either a whole information system or parts of it to reveal connections between data points and structures. This is the first step in database design, and Simsion and Witt [20] defined it as "*a design activity which classifies information in an organized way and defines their relations.*" Therefore, the process of data modeling involves professional data modelers working closely with business stakeholders, as well as potential users of the information system [21].

The ER model is one of the fundamental conceptual data models, which is usually associated with relational databases. This model is the focus of this work because it is the model most often adopted at this stage of conceptual design. An Entity Relational Diagram is a drawing that communicate the relationships between tables, also known as entities [22].

An entity is a "thing" or "object" in the real world that is distinguishable from other objects. Relationships have cardinalities, attributes, and constraints. The cardinality of a relationship indicates the number of occurrences between two entities [23].

Any database architect needs to work with a tool that allows an easy data model design. Such a choice will have a direct impact on the project quality. The design tool must be suitable to represent a database, also be easy to use, and support a different number of database engines. In addition, the tool should allow defining constraints such as Primary Key (PK), Not Null (NN), Auto Incremet (AI), and produce a Structured Query Language (SQL) script from the representation of data objects created by the user.

2.4.2 Data Models Evolution

In this section, are analyzed the different database models, their evolution, and characteristics.

In the past decades, the evolution of data models, which average ten years between them, has come a long way (Fig. 2.5). Before 1950, the File System model was the only way to store data. From 1950 emerged the Hierarchical model, and in the 1960s, appeared the Network model. Later, around the 1970s, the Relational and Semantic models gained popularity, and one decade later, appeared the Object model. Around 1990 arose the Object-Relational, and at the beginning of the 2000s arrived the NoSQL database models.

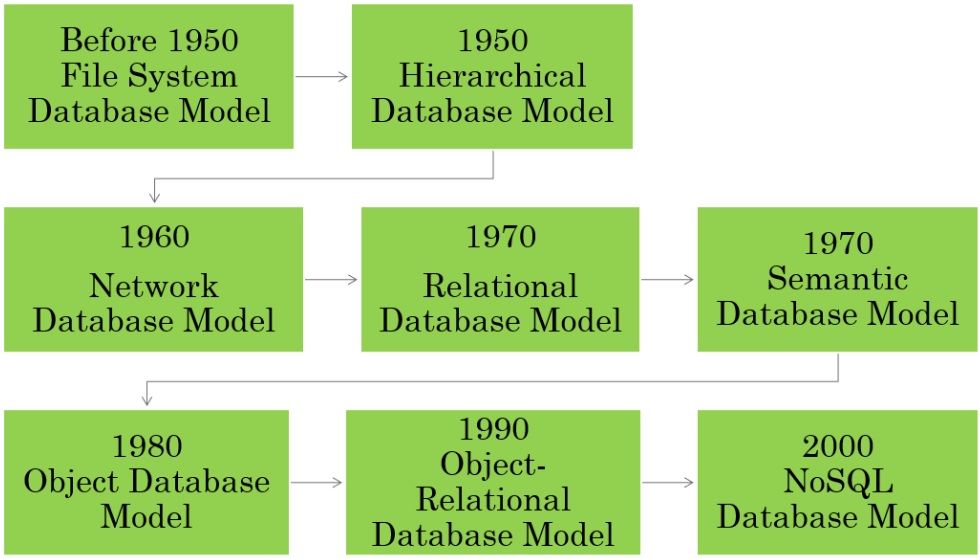


Figure 2.5: Data Models Evolution

File System Database Model

In the **File System database model** there are no modeling techniques, and that data storage is in individual or multiple flat files, with no structure. This aspect adds expense, complexity and has a limited search capability [24]. Niazi et al. [25] developed their research on new databases and used a database to manage file system metadata for Distributed and Hierarchical File Systems.

Hierarchical Database Model

The **Hierarchical database model** is a collection of connected records through links with a parent-child relationship. This model is used to store information as related data objects [26]. It starts with a root node, the parent of all parents, which has one or more children (one-to-many relationship), while each child possesses only one parent [24]. Kroenke [27] described the hierarchical model as "*a data structure in which the elements of the structure have only one-to-many relationships with one another.*"

One example of this model can be formed by a system that relates Customers with Orders, and the administrator wants to find information about a specific order. First, it is necessary to find the Customer and then loop through the linked list of Orders. In the worst-case scenario, the intended order is the last stored in the system, and this example shows the limitation of this implementation. Black et al. [28] provided another example of this model with a Hierarchical database for surveillance, achieving outstanding success.

Network Database Model

Kroenke [27] described the **Network Database Model** as a data structure "in which at least one relationship is many-to-many". This model improved the Hierarchical model by associating each file with n number of other files, allowing child tables to have over one parent (many-to-many relationships) [24].

Relational Database Model

The **Relational Database Model** is now the most widely used data model. It has been the basis for many investigations and studies since E. F. Codd introduced it in 1970 [29]. Each table, also known as a relation, is a subset of the Cartesian product, characterized by a name distinct from all other tables in the database. In a table, each row represents an individual group of related data values, also known as tuples. Also, operations combining different data types are not allowed [30].

This model removed the search limitation of the Hierarchical model, with prior knowledge of the data structure. And grants relationship possibilities between any data, ensuring many-to-many relationships. Also, the data can be retrieved from one or multiple related tables in the database [24].

Semantic Database Model

The **Semantic database model** is a high-level semantics-based database description and structuring formalism. It represents a database by types of entities existing in the application environment, the classifications and groupings of those entities, and the structural interconnections. The semantic database model also enhances the effectiveness and usability of database systems [31]. This model emerged to address some weaknesses of the Relational data model. For example, the poor representation of "real-world" entities, only using one construct for both entities and relationships, and the difficulty of managing complex objects. The relational constraint of atomic values means that accommodating hierarchical or nested structures is challenging.

This data model attempts to provide more expressive means of representing information

than the Hierarchical, Network, and Relational models, by providing a richer set of constructs to build schemas. Such constructs allow the database developer to incorporate more of the meaning of some application domain [32]. The prime advantages over Data-oriented models are the separation of conceptual and physical components, the diminished semantic of relationship types, and the abstraction resources [33].

Object Database Model

The **Object database model** combines the Object-Oriented concepts with the ER Model. It expresses data and metadata in terms of objects. This model represents each object by at least one object type that specifies the attributes and methods that its instances will have [34]. It solves the Relational model's many-to-many relationships' complexity by separating complex elements into minor parts [24].

Object-Relational Database Model

The **Object-Relational Database Model** merges the best features of the Relational and Object models by adding object-oriented concepts such as objects, methods, constructors, arrays of objects, and nested tables.

These data models decrease redundant data, are compatible with object-oriented interfaces, and have a large storage capacity, access speed, and manipulation power [35].

NoSQL Database Model

The **NoSQL database model** emerged with the continuous development of the Internet and Cloud computing. The NoSQL databases are not built primarily on tables and rarely use SQL for data manipulation. These systems are efficient when working with vast amounts of data and when the data nature does not require a relational model.

The fundamental characteristics of these databases are i) strong consistency - everybody sees the same version of data; ii) high availability - the clients can always find at least one copy of the requested data; iii) partition-tolerance - the system keeps its characteristics, even when deployed on different servers [36].

Because Relational databases are one of the most used and NoSQL databases are rising demand, some authors, such as [37], [38], and [39], addressed the integration of Relational and NoSQL databases. These authors had the goal to overcome the drawbacks of the Relational model in distributed systems, in big data environments, and with high data transfer associated with the number of users. Reniers et al. [40] address similar migration techniques from Object to NoSQL databases by aggregating the data model and using a flexible schema.

2.4.3 Background in conceptual modeling

In this subsection, it is addressed the Conceptual Modeling (CM) topic, identifying key features and languages.

CM describes the physical or social aspects of the world abstractly. The result of a proper and rigorous CM design is a functionally richer, less error-prone, adequately attuned, able

to adapt to varying user requirements, and less expensive system [41]. Thus, designing the CM at the beginning of the development cycle should be mandatory. It is easier to follow and adapt to user requirements and explore existing relationships between the concepts.

Moody et al. [42] mentioned the vast amount of alternative designs to address Conceptual Data Modeling (CDM). Several alternative models could provide accurate solutions, but may have quite distinct implications for database and system design. The process of data modeling is not simple, meaning it usually demands multiple iterations [43].

Thalheim [44] pointed different CM notations used to describe requirement specifications, such as the ER diagram [45], the Unified Model Language (UML) [46], which are the most regularly employed, Business Process Modeling and Notation, and Model-driven Engineering. Object-Oriented (OO) models are essentially expressive and more fitted to describe static and dynamic features of complex applications. The OO modeling field relates objects and attributes, whereas the real-world realm deals with things and properties [47].

The ER data model has existed for over 35 years. An ER diagram is appropriate for data modeling because it is abstract and is easy to discuss and explain. It is easy to translate ER models to relations. The base of this type of modeling are entities, which hold information and relationships, defined as the associations between entities [30].

The primary advantages of CM for general systems and specifically for Database Management Systems are:

- Provide a high-level perception of how the system will work;
- Join different mental models into a single CM design;
- Ensure that the data representation is accurate - missing fields in the database cause unreliable results;
- Get a clear understanding of the data that developers can manage when building the actual database;
- Identify any redundant or missing data;
- Make maintenance and upgrades faster and more affordable.

Next, it is performed a qualitative analysis of data modeling tools.

2.4.4 Data Modeling Tools

One of the main goals of this thesis is to analyze data modeling tools that convert the conceptual model into a physical model. The list of analyzed tools appears in the Table 2.1. An ER diagram is appropriate for data modeling because it is abstract and is easy to discuss and explain. The base of this type of modeling are entities, which hold information and relationships, defined as the associations between entities [30].

To create an ER model, it is necessary to specify entities containing fields (attributes) that will be the columns and relations between tables. In this model, the relationships amongst tables have a cardinality setting that illustrates the following options: one-to-one, one-to-many, zero-to-one, zero-to-many, and many-to-many. Despite this graphical representation, the physical model produces a better comprehension of the relationships, because it defines the relationship cardinality through the PK and Foreign Key (FK)

constraints, as well as Unique or NN. Converting the design of the CM into a physical model offers a straightforward interpretation of the model.

After these steps, and with a proper definition of the business logic, the next step is *Forward Engineering*, which is the auto-generation of a SQL script from the created representation. This last step is crucial for any database engineer to minimize errors and time spent creating a database.

From the list of *79 Data Modeling Tools Compared* [48] and *20 Best Data Modeling Tools* [49], were selected those that allow the user to perform *Forward Engineering*, and only considered tools that continued receiving updates after 2018. They were sorted into four major product types: Online free tools, Online commercial tools, Desktop free tools, and Desktop commercial tools. The four product types are able to broadly represent all the products available today, is well-aligned with the different users' and enterprises' needs.

Table 2.1: Modeling Tools

Modeling Tools	
Product Type	**Name**
Online free tools (I)	Dbdesigner.id `https://dbdesigner.id`
	Onda `http://onda.dei.uc.pt/v3`
	WWW SQL Designer `https://github.com/ondras/wwwsqldesigner`
Online commercial tools (II)	Dbdesigner.net `https://app.dbdesigner.net`
	dbDiffo `https://dbdiffo.com`
	GenMyModel `https://www.genmymodel.com`
	Lucidchart `https://lucid.app`
	sqlDBM `https://sqldbm.com`
Desktop free tools (III)	MySQL Workbench - Community Version `https://github.com/mysql/mysql-workbench`
	pgModeler `https://github.com/pgmodeler/pgmodeler`
	Umbrello UML `https://github.com/KDE/umbrello`
Desktop commercial tools (IV)	dbSchema `https://dbschema.com`
	dbWrench `http://www.dbwrench.com`
	Erwin Data Modeler `https://erwin.com/products/erwin-data-modeler`
	Navicat `https://www.navicat.com`
	Oracle SQL Developer Data Modeler `https://www.oracle.com/database/technologies/appdev/sqldeveloper-landing.html`
	PowerDesigner `https://www.sap.com/products/powerdesigner-data-modeling-tools.html`

Another relevant characteristic is that some tools have a visual representation of the conceptual data model, but others only show the logical data model. However, others have both types of visualization. On the one hand, the logical data model only describes the data and its relations. On the other, the physical data model displays' table structures, including column name, data type, and the constraints, such as PK, FK, and Unique (UQ), represent the relationships between tables.

In the following subsections, is performed a qualitative analysis of the selected tools. It is provided a brief description of the key features, such as the release year; real-time collaboration options; generation of the physical model; the presence of *Reverse Engineering* (auto-generation of ER from SQL) and *Forward Engineering*; supported database engines and data types; different constraints; the presence of CM; finally, the pros and cons are analyzed.

Online free tools

Online free tools work on every platform, receive constant updates from the community, and do not require installing the software. Also, every developer can contribute to any of these projects.

Dbdesigner.id

Dbdesigner.id is a database design tool for web developers and beginners, which started in 2019 under MIT License as a hobby project and is continuously under construction. The authors' goal was to make database management accessible to everyone. They developed it in Javascript, HTML, and CSS. To use Dbdesigner, they require creating a new user account. With this tool, it is possible to share a link with a project contributor to work simultaneously. It does not allow for a CM design. The tool does not provide reverse engineering, and only works with MySQL. For each column, the user can choose among different types (tinyint, smallint, bigint, int, bigint, float, double, datetime, date, timestamp, char, varchar, binary, blob, text, json). It enables an option to specify constraints such as PK, FK, NN, UQ, and AI. It is also possible to set a default value for each entry in a table. The user can select the referencing table and column name to create a relationship.

Pros: Link sharing for collaboration.

Cons: Needs registration and only uses MySQL.

Onda

Onda is a database modeling tool developed by the DEI community, with the first version released in 2014, developed with Javascript, HTML, and CSS. Also, it has fast loading times. The tool does not require any configuration, but there is no real-time collaboration possibility. With Onda, it is possible to draw conceptual databases and visualize the physical model. There is no reverse engineering option, but it is possible to perform forward engineering for the most famous database engines like PostgreSQL, MySQL, Oracle, MariaDB, and SQLite. Each column can have different types, such as boolean, integer, float, date, character, varchar, text, or BLOB. It is possible to add constraints for each column, such as PK, NN, AI, Check (CH), and UQ. The FK are automatically added in the physical model through the designed relationships.

Pros: Zoom in/out, possibility to export the CM into different database engines.

Cons: Some bugs on the physical model and script generation.

WWW SQL Designer

Released in 2005, WWW SQL Designer allows users to create and export data models to SQL scripts. The interface has a mini-map for fast navigation. It does not require configuration, does not have real-time collaboration or representation of the CM. With this tool, it is possible to perform forward engineering for the MySQL database engine, but it is impossible to perform reverse engineering. It supports many database constraints, such as PK, FK, UQ, NN, and AI. There are different data types such as int, decimal, char, binary, BLOB, date, and time.

Pros: Many data types, and drag-and-drop features.

Cons: No real-time collaboration, no representation of the CM, and only exports MySQL scripts.

Online commercial tools

This subsection will introduce the online commercial tools. Because they work online, it is possible to use them on every platform. It is necessary to get a subscription or activation key. Otherwise, it is limited, where the features are only available for a short period or with narrow options.

Dbdesigner.net

Since 2006, Dbdesigner.net is a database schema designer for data modeling. The table representation is clean and has different colors for each table entry. Dbdesigner does not require any configuration to use it. It is possible to share the database design with other users to work simultaneously on it. This tool does not provide the visualization of the databases' CM model. Dbdesigner also offers a reverse engineering option. Besides it, forward engineering enables code export for engines like MySQL, Microsoft SQL Server, PostgreSQL, Oracle, and SQLite. This tool has distinct features that make it unique, such as a mini-map for fast navigation, keyboard shortcuts, instant save with history, copy and paste, undo and redo, and notes and comments. It supports data types such as binary, boolean, date, decimal, float, integer, and varchar. The constraints are: PK, FK, NN, AI, and UQ.

Pros: Developers can work simultaneously, has several database design templates, features as instant save, undo and redo, easy-to-use interface. Also, it enables reverse engineering.

Cons: No representation of the CM.

dbDiffo

Released in 2014, Dbdiffo is a database modeling tool similar to Onda. Before using the tool, it is necessary to specify the model name and choose the database engine for script generation. With Dbdiffo, it is impossible to do a real-time collaboration, and it does not provide the CM design. This tool does not allow reverse engineering, but regarding forward engineering, it is possible to export scripts for database engines like IBM DB2, Microsoft SQL Server, MySQL, Oracle, and PostgreSQL. The tables' columns may be bigint, binary, bit, blob, char, date, datetime, decimal, double, float, integer, longblob,

longtext, mediumint, mediumtext, numeric, smallint, text, time, timestamp, varchar, or year. Regarding the constraints, it is possible to define PK, FK, NN, and AI.

Pros: History toolbar, and unlimited undo.

Cons: No real-time collaboration and no reverse engineering.

GenMyModel

Released in 2012, GenMyModel speeds up the design of software architecture and business processes. It is easy to add new entities and create relations between them because of its interface and the provided documentation. It is necessary to log into the application via GitHub or Google account, create a new diagram, and select a Relational Database. It is possible to create a database from scratch or select an existing project from the cloud. GenMyModel has real-time collaboration with a chat and also allows the creation of the CM, since it has its base in the UML. The supported database's engines are Apache Hive, Oracle, MySQL, and PostgreSQL. This tool supports different data types, such as boolean, binary, character, date, float, integer, time, or varchar. The tool can auto-generate PDF and MS Word documents based on custom templates and export diagrams to GitHub. The developers can use the open API and integration functions to build integrations for testing or code proofing. Regarding the constraints, it is possible to define PK, FK, NN, CH, and UQ.

Pros: Real-time team collaboration features, it auto-generates the documentation of the data models.

Cons: If using the free version, it only provides basic features and limits the design to 20 objects, including not only the tables, but also each column and relationship.

LucidChart

Released in 2008, Lucidchart is a powerful tool, it has a free version, and also offers a trial to explore its full potential. It is unnecessary to do any configuration before using the tool. Lucidchart has team collaboration, is based in the UML, and allows CM design. It is not possible to perform reverse engineering. Forward engineering is possible to several database engines such as MySQL, PostgreSQL, Microsoft SQL Server, and Oracle. Each column may have any data type introduced by the user, later converted into the specific database engine data types. As in other tools, it is possible to choose from the list of existing types. The constraints are PK, FK, NN, and AI.

Pros: Real-time collaboration feature, and different templates.

Cons: The user must know the specificities of the database engine data types because it is possible to insert any string in the data type field.

sqlDBM

Released in 2017, sqlDBM has a free version that comes with limited features, and it is possible to try the full version for 14 days. This design tool only requires configuring the database type, and it does not offer CM design. There is also a team collaboration tool. It is impossible to perform reverse engineering, but forward engineering allows to export the script to Microsof SQL Server, MySQL, Snowflake, Amazon Redshift, PostgreSQL, and Azure Synapse Analytics. Each column attribute can be of the type bigint, bigint unsigned, binary, bit, blob, char, date, datetime, decimal, double, double unsigned, float,

integer, numeric, text, time, timestamp, varchar, or year. Also, it is possible to specify the constraints PK, FK, NN, AI, and UQ.

Pros: This tool has forward engineering and team collaboration possibilities. Little tutorial, in the beginning, explaining functionalities, no need to sign-up. "Undo" and "redo" options also present.

Cons: No CM design.

Desktop free tools

In this subsection, are analyzed four desktop free tools. These require installation, and as make part of the open-source community are free to use, and every developer can contribute to their development.

MySQL Workbench - Community Version

Created in 2002, MySQL Workbench is an application used to manage and design a database schema. The open-source version has a GPL license with a GitHub repository. The significant differences between Community and Enterprise versions are the non-presence of Schema & Model Validation, automated documentation of databases, and non-existence of firewall specification rules. Before working with the tool, it is necessary to set up the connection to the existing database. Otherwise, it is impossible to export the MySQL script resulting from the establishment of tables and relations. MySQL Workbench does not have a real-time collaboration feature, but has the capacity to design the CM. There is a possibility to reverse and forward engineer for MySQL Server databases. Different categories organize the entity types like numeric, characters, time, geometry, and others, such as bits or boolean. The number of constraints is also considerable, it enables an option to insert PK, FK, NN, UQ, AI, Binary, and .

Pros: Unlimited "Undo" and "Redo" options.

Cons: Only available for Windows machines and no real-time collaboration. Also, necessity to set up the connection to the existing database.

pgModeler

Created in 2006, pgModeler is a database modeling tool designed for PostgreSQL databases. Despite the need to pay for the compiled version, it is possible to get the open-source version and compile it manually. The tool has different colors to help visualization. If there are missing functionalities, it is possible to create new extensions and contribute to open-source code development. It is unnecessary to make any configuration before using the application. The tool does not have an option for real-time collaboration, and it has the feature to design the CM. It also provides reverse and forward engineering for PostgreSQL databases. pgModeler has a database management module where it is possible to run SQL commands, explore the objects, and handle data. It has distinct entity types like bigint, bit, bool, char, date, decimal, float, int, json, money, text, time, and varchar. Also, different constraints such as PK, FK, UQ, Exclude, CH, AI and NN are present.

Pros: It is possible to collaborate on the tool.

Cons: Only supports PostgreSQL database engine.

Umbrello UML

Released in 2006, Umbrello UML is a UML diagram program developed by an international free software community. This tool does not require any configuration. It is not possible to perform real-time collaboration. It has a feature to design conceptual data models. It is impossible to reverse engineering, but the forward engineering option allows generating SQL scripts for MySQL and PostgreSQL. Each table column can have different types, such as bool, char, double, float, int, or string. And there are different constraints: PK, FK, AI, UQ, and NN.

Pros: Feature for CM.

Cons: Limited number of databases to export scripts, and no real-time collaboration.

Desktop commercial tools

In this subsection, we analyze desktop commercial tools. We chose these according to Google Trends, since the tools number in this product type is high. For us to consider a tool, it had to be googled at least ten times per week, from 2004 until 2021, worldwide, and we found seven desktop proprietary tools matching the criteria. To use either tool, first, it is necessary to install it on the machine and then either try a free trial or buy the full version.

dbSchema

Released in 2016, dbSchema does not require configurations. dbSchema cannot provide a conceptual data model, neither exists real-time collaboration. It has both reverse and forward engineering that works with all relational databases, including SqlServer, SAP Adaptive Server, Oracle, MySql, Ingres, Informix, Db2, Derby, Firebird, Frontbase, Cache, Pervasive, PostgreSQL, and Sqlite. Like all the previous tools, it allows adding different data types to each column, like blob, boolean, char, date, double, float, int, json, real, text, varchar. Farther, the constraints are PK, FK, NN, and AI.

Pros: The tool has both reverse and forward engineering for many database motors.

Cons: No real-time collaboration.

dbWrench

dbWrench is a multi-platform database design and synchronization software, released in 2004, and does not require any configuration. dbWrench does not support the design of the CM and has no real-time collaboration. The reverse engineering does not exist in this tool, and the forward engineering tool generates SQL scripts for Microsoft SQL Server, Oracle, PostgreSQL, and MySQL. It is possible to connect to a database and run the code for table creation. It offers some default column templates that save time creating tables. When adding a column, there is a possibility to specify data types like binary, blob, bit, boolean, char, date, decimal, double, float, json, number, real, time, varchar. The constraints are PK, FK, NN, and AI.

Pros: Reverse and forward engineering for several databases engines.

Cons: No real-time collaboration.

Erwin Data Modeler

Founded in 1988, the platform allows creation and maintenance of data warehouses and databases. This tool provides several tutorials to help understand how to do data modeling. Financial services, healthcare, critical infrastructure, and technology companies use Erwin Data Modeler. The tool does not require any configuration, and does not have an option of real-time collaboration. Erwin provides a possibility to design the conceptual data model. It only has forward engineering that supports database engines such as Oracle, MySQL, IBM DB2, SAP IQ, and Teradata. The different data types for columns in this tool are char, integer, date, boolean, real, and float. The tool has constraints such as PK, FK, NN, AI, and UQ.

Pros: Supports several database engines for forward engineering.

Cons: It is necessary to fill in a form to try the trial version, but there is no guarantee that the application will be accepted. The local version of the tool doesn't work on Mac OS, and it is necessary to use the cloud version. No real-time collaboration, and reverse engineering.

Table 2.2: Comparison of different tools

Product Type	Tool name	Characteristics							
		Open source	Online	Supported databases	Supported OS	Constraints	CM design	Free	Licence
I	Dbdesigner.id	✓	✓	MySQL	Windows, Linux, macOs	PK, FK, UQ, AI, NN	✗	✓	MIT
	Onda	✓	✓	MySQL, postgreSQL, MariaDB, Oracle, SQLite	Windows, Linux, macOs	PK,FK,NN,CH,AI,UQ	✓	✓	CCANCSAIL*
	WWW SQL Designer	✓	✓	MySQL, sqlLite, Oracle, PostgreSQL, mssql, web2py	Windows, Linux, macOs	PK, FK, UQ, AI, NN	✗	✓	BSD-3-Clause
II	Dbdesigner.net	✗	✓	MySQL, Microsoft SQL, PostgreSQL, Oracle, SQLite	Windows, Linux, macOs	PK, FK, UQ, AI, NN	✗	✗	Proprietary
	dbDiffo	✗	✓	IBM DB2, MS SQL Server, MySQL, Oracle, PostgreSQL	Windows, Linux, macOs	PK, FK, NN, AI	✗	✗	Proprietary
	GenMyModel	✗	✓	Apache Hive, Oracle, MySQL, PostgreSQL	Windows, Linux, macOs	PK, FK	✓	✗	Proprietary
	Lucidchart	✗	✓	MS SQL Server, MySQL, Oracle	Windows, Linux, macOs	PK, FK, NN, AI	✓	✗	Proprietary
	sqlDBM	✗	✓	MS SQL Server, MySQL, Snowflake, Amazon Redshift, PostgreSQL, Azure Synapse Analytics	Windows, Linux, macOs	PK, FK, NN, AI	✗	✗	Proprietary
III	MySQLWorkbench Community Version	✓	✗	MySQL	Windows	PK, FK, NN, UQ, AI, Binary, Unsigned	✓	✓	GPL
	pgModeler	✓	✗	PostgreSQL	Windows, Linux, macOs	PK, FK, UQ, E, CH	✓	✓	GPL
	Umbrello UML	✓	✗	mySQL, PostgreSQL	Windows, macOs	PK, FK, AI, UQ	✓	✓	LGPL
IV	dbSchema	✗	✗	SqlServer, SAP Adaptive Server, Oracle, MySql, Ingres, Informix, Db2, Derby, Firebird, Frontbase, Cache, Pervasive, PostgreSQL, SQlite	Windows, Linux, macOs	PK, FK, NN, AI	✗	✗	Proprietary
	dbWrench	✗	✗	Microsoft SQL Server, Oracle, PostgreSQL, MySQL	Windows, Linux, macOs	PK, FK, NN, AI	✗	✗	Proprietary
	Erwin Data Modeler	✗	✗	Oracle, MySQL, IBM, DB2, SAP IQ, Teradata	Windows, Linux, macOs	PK, FK, AI, NN	✓	✗	Proprietary
	Navicat	✗	✗	MySQL, Microsoft SQL Server, Oracle, PostgreSQL, SQLite, MariaDB	Windows, Linux, macOs	PK, FK, UQ, AI, NN	✓	✗	Proprietary/ Shareware
	Oracle SQL Developer Data Modeler	✗	✗	Oracle, DB2, MySQL, and Microsoft SQL Server MySQL	Windows, Linux, macOs	PK, FK, UQ, AI, NN	✓	✗	Proprietary
	PowerDesigner	✗	✗	Oracle, PostreSQL, IBM DB2, SQP IQ, Microsoft SQL, Teradata	Windows	PK, FK, UQ, AI, NN	✓	✗	Proprietary

Constraints meaning: PK - Primary Key; FK - Foreign Key; UQ - Unique; AI - Auto Increment; NN - Not Null; E - Exclude; CH - Check; U - Unsigned
*Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License

Navicat

Navicat is proprietary software created in 2002 and provides a mini-map for fast navigation. It allows adding colors to tables, thus making them more visually appealing. This tool does not require any configuration and does not have real-time collaboration. It is a powerful and cost-effective database design tool that allows designing CM. It allows performing reverse and forward engineering processes. This tool allows creating data models for MySQL,

Microsoft SQL Server, Oracle, PostgreSQL, SQLite, and MariaDB databases. It has several data types, such as blob, boolean, integer, varchar, date, and timestamp. For the data constraints, there are PK, FK, NN, AI, and UQ.

Pros: Performs reverse and forward engineering.

Cons: No real-time collaboration.

Oracle SQL Developer Data Modeler

They released Oracle SQL Developer in 2006, and it is an integrated development environment that simplifies the development and management of Oracle Databases. It is unnecessary to do any configuration. This application permits conceptual data modeling. Although, it does not have real-time collaboration. The application allows performing forward engineering for the Oracle Database engine. It supports managing the Oracle Database performance, security, storage, and settings. The tool has different data types such as blob, char, decimal, float, date, and timestamp. There is the possibility to define PK, FK, AI, and UQ constraints.

Pros: Quick loading time, and the existence of a tutorial that explains how the tool works.

Cons: Only supports Oracle databases.

PowerDesigner

PowerDesigner is a collaborative enterprise modeling tool, released in 1989 with the name of "AMC*Designer" and is currently owned by SAP. It is a modeling tool for easy visualization, understanding, and management of the data in a project. It is unnecessary to make any configurations. PowerDesigner has real-time collaboration, and it is possible to implement a CM. Also, it has multiple database connections to model the data. The tool also offers reverse engineering. However, forward engineering allows working with the most popular data management systems, such as Oracle, PostgreSQL, IBM DB2, SQP IQ, Microsoft SQL Server, and Teradata. This tool only works on Windows OS. PowerDesigner allows creating multiple entities at once and saves a lot of time. The present data types are integer, decimal, money, boolean, characters, text, date, and timestamp. For the data constraints, there are PK, FK, NN, AI, and UQ.

Pros: Has real-time collaboration, and several database engines for forward engineering.

Cons: Only available for Windows, and no reverse engineering.

In this section, was performed a qualitative evaluation of each tool by presenting concise descriptions and pros and cons. Table 2.2 displays a summary of the fundamental characteristics of each. In the following section, it is explained the evaluation method. And the scoring assessment. Hence, producing an objective analysis.

2.5 Tools Evaluation

Tools Evaluation is performed using a combination between Business Readiness Rating (BRR) and OSSpal methodologies, as explained in the scientific paper, already submitted for revision, in the end of this document. To avoid repeated text, it was decided that the new methodology and explanation of the punctuation would not be in this part of the text.

2.6 State of the art conclusions

This Chapter is an introduction to the EC and CC paradigms, for those who have little or none notions about these topics. One of the most important conclusions to be drawn from here is that the Edge and Cloud should exist together, as a compliment to one another, as it will be in this extended application. The data modeling plays an essential role in any system development or database design processes. Within the scope of this work will be used the ER data model, because it is the most widely used model. The analysis of data modeling tools was performed with help of PhD student Gonçalo Carvalho, who adapted the evaluation methodology and made the calculations, Professor Vasco Pereira, Professor Bruno Cabral and Professor Jorge Bernardino. This analysis will help database architects to choose the best tool to perform their projects, according their needs.

This page is intentionally left blank.

Chapter 3

Planning

In order for any project to be successful, it must be previously planned and analyzed, to have a clear view of the final product. Planning the software development process is a crucial step. It includes project requirements gathering, work planning, product design, and risks identification. This chapter explains the planning of this internship. In Section 3.1 is explained what methodology was adopted in the development of the thesis, also how the work was managed. In Section 3.2 is explained how the work was divided and the duration of each task. In Section 3.3 is explained how the risk management works, the project success criteria and are presented risks associated to this work.

3.1 Process Management

For the development of this thesis was adopted an Agile methodology. This methodology argues that the development process should seek client satisfaction through continuous deliveries of the software, by constantly keeping in touch with the client [50]. In this specific case, the client is referred as the supervisors. The agile methodology is unique due to the 12 principles it has [51]:

1. The highest priority is to satisfy the customer through early and continuous software delivery;
2. The changes in requirements are welcome, even late in development. Agile processes harness change for the customer's advantage;
3. Deliver working software frequently, from a couple of weeks to a couple of months;
4. Business people and developers must work together daily throughout the project;
5. Build projects around motivated individuals;
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation;
7. Working software is the primary measure of progress;
8. Agile processes promote sustainable development;
9. Continuous attention to technical excellence and good design enhances agility;
10. Simplicity – the art of maximizing the amount of work not done is essential;

11. The best architectures, requirements, and designs emerge from self-organizing teams;

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

At the beginning of the second semester, were defined project requirements, and also were designed low-level mock-ups. Since this work follows an incremental methodology, the defined requirements could be changed. This means that new requirements could be added. Also, the requirements that made little sense at some stage of the project could be removed.

It was defined since the beginning that there would exist meetings every week, via Zoom, between the supervisors and the intern. Apart from that, each month it was filled a document with the project progress. This document included information such as developed work and planned work for the next month. Each month can be called a **sprint**, since it is treated "as a project with a month's horizon" [52].

3.2 Work Planning

The present thesis is divided in two semesters. In the first semester, the focus is mostly on bibliographic research concerning Edge Computing (EC), data models, data modeling tools, initial study of the tools, and decision and specification of the new functionalities to implement, explained in Chapter 4. In the second semester, the gained knowledge is applied on the requirement specification, implementation of functionalities, results tests and analysis.

Table 3.1 shows the estimated and real times for the project parts during the two work semesters.

Table 3.1: Masters Thesis Estimated and real work times

Task	Name	Estimated time (weeks)	Real time (weeks)
1	Bibliographic Research	4	4
2	Analysis of data modeling tools	12	20
3	Familiarization with the modeling tool	3	2
4	Specification of the requirements and extension of the data modeling tool	10	12
5	Project tests and results analysis	4	4
6	Writing of the Masters Thesis	23	23

Tasks

1. **Bibliographic Research**

   This task encompasses an analysis of the state-of-the-art on EC and different Data Models suitable for EC Paradigm, starting in September 2020 and with a duration of the first semester.

2. **Analysis of data modeling tools**

   This task started with the research on many Data Modeling Tools and writing a comparative scientific paper on the existing tools and understanding which one is the most suitable for a data modeler to use. It started in October and had a duration of 9 months.

3. **Familiarization with the modeling tool**

   We can divide this task into two different stages. The first stage is code familiarization, by correcting an identified Onda [2] problem. The second stage was the specification of requirements and their implementation to present on the intermediate evaluation. A drop-down menu was added to the navigation bar, to select between different table types and also implemented two new types of tables (facts table and dimension table). In Chapter 6, are provided the details about problem-solving and familiarization with the code. This task occurred between December and mid-January.

4. **Specification of the requirements and extension of the data modeling tool**

   In this stage, the project requirements were collected and later implemented following an agile methodology. Also in this step were designed low level mock-ups, which suffered changes during the project implementation. At the end of the second semester, it is expected to be possible to draw an extended Entity Relationship (ER) database model on the Edge and the Cloud. These models must have data aggregation functions to gather data from the Edge and transfer them to a Data Warehouse placed on the Cloud. This work duration is of about five months.

5. **Project tests and results analysis**

   Tool testing in a defined use case and results analysis. The duration of this step was of approximately one month.

6. **Writing of the Masters Thesis**

   The writing of the Masters' Thesis started in October 2020 and is a continuous work.

In the Figure 3.1 appears the Gantt chart to illustrate the work progress during development of this thesis. To each row corresponds a task from the previous list and its duration, in weeks.

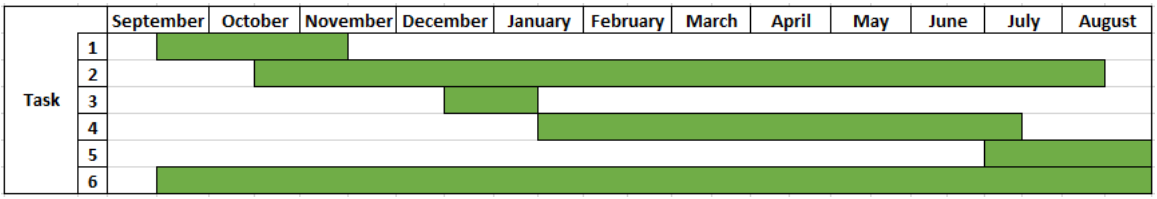


Figure 3.1: Work progress, with each month divided in sections of 2 weeks

3.3 Risks Management

In this section is explained the project success criteria and how it can be achieved. There are also identified, evaluated and it is defined a risk mitigation process, that will prevent the negative impact during this thesis execution.

A risk is an event that may have a negative impact on the project execution. Risk Management is the process of identifying, evaluating and responding to mentioned events, in order to handle first the risks with greater probability of occurrence, or with highest impact (highest loss) [53]. The risks are identified by their ID, description, impact on the project, scale of the impact, and their probability.

There does not exist a project without risks, and this work is no exception. In this section, the main risks are identified before they become problems and affect the work. After risk identification, it is important to follow the mitigation plan in order to minimize the occurrence of the risks.

3.3.1 Project Threshold of Success

Any project is considered successful when it is completed at time, cost (which is the same as time in this work) and according to the intended scope [54]. The intended scope is a working prototype with successful test results and all the "Must" requirements implemented.

3.3.2 Risk Analysis

When a risk is identified, it must be categorized, accordingly to a probability to occur and the impact it may cause to the project. This is important in any project management environment, in order to save time and resources in risks that have low impact or low probability to occur, and focus on the most important things.

A risk is defined by the following entities:

- **ID** - Identifies the risk, to differentiate one risk from another;
- **Description** - Describes the problem that can occur;
- **Project impact** - What will happen, in case the risk occurs;
- **Impact scale** - In case of the occurrence, the impact it will have in the project execution;
- **Probability** - Probability of risk occurrence.

Impact Scale

- **High** - Extreme event, means large delays;
- **Medium** - Large impact, but can be managed with effort, using mitigation plan;
- **Low** - Minor impact on the project schedule.

Probability Scale

- **High** - High probability of occurrence;
- **Medium** -Medium probability of occurrence;
- **Low** - Low probability of occurrence.

3.3.3 Identified risks and mitigation strategies

Tables 3.2, 3.3, 3.4 and 3.5 display the identified risks, accordingly to the previous specifications. The four main risks of this work are changes in project requirements, poor documentation, failure to deliver the work on defined time, and technology risks. Below each table it is explained the mitigation strategy used to overcome the issue. It is also explained the impact and probability scale of the identified risks.

Table 3.2: Risk 1 - Changes in requirements

ID	Risk 1
Description	Changing requirements
Project impact	Changes to the schedule Overloaded sprints Abandoned tasks
Impact Scale	Medium
Probability	High

As the project follows an Agile methodology, the risk of adding new requirements became a constant problem during the second semester of the project development, this is why the probability of this to happen is high. Every time this happened, it was analyzed the impact it would have on the project execution and if the changes were worth the effort. This mitigation strategy guaranteed that the sprints would not get overloaded and that all the tasks were executed properly and in time. The impact scale is defined as medium, but it also varies from one requirement to another.

Table 3.3: Risk 2 - Poor documentation

ID	Risk 2
Description	Poor documentation, leading to problems to understand the code, the project structure and the functionalities
Project impact	Deliver wrong results Delays on project delivery
Impact Scale	High
Probability	Low

Because an existing tool without documentation was extended, it was important to not repeat the same mistake, which means that this risk did not became real. Every peace of code was commented, in order to avoid that further versions would suffer from the same problem. This is the reason why the probability of occurrence of this risk is low. The poor documentation has a high impact on the project, because it is normal that after some time even the developers can forget how some functionalities are implemented.

Table 3.4: Risk 3 - Failure to deliver on time

ID	Risk 3
Description	Failure to deliver the defined tasks for a sprint in time
Project impact	Delays in project implementation Uncompleted tasks Client dissatisfaction Bad working atmosphere
Impact Scale	High
Probability	Medium

This risk became real in different points of the second semester of this thesis, during the extensions of the data modeling tool. This is the reason why the probability to occur is defined as medium. The impact scale of this risk is high, this is why it had to be mitigated properly. In the weekly meetings the undone tasks were discussed to be done differently, simplified or even discarded if not important.

Table 3.5: Risk 4 - Technology risks

ID	Risk 4
Description	Technology risks - as the goal of the project is to extend an existing tool, there is a need to first understand the project structure, and evolving technologies, and continue to develop using the existing project libraries
Project impact	Deliver wrong results Overloaded sprints Delays in project implementation
Impact Scale	High
Probability	Medium

The technology risks occurred in different points of the development process, which leads to medium probability scale. This could also have a high project impact. To mitigate this risk, the amount of research and study was higher and constant, to avoid delayed deliveries and overloaded sprints.

3.3.4 Risks Matrix

In this subsection is referred a Risks Matrix. This technique helps visualizing their incidence, in order to pay more attention to the most critical risks, before they become a real problem during the project execution.

In the previous subsection were identified four risks:

- **Risk 1** - Changes in requirements and priorities;
- **Risk 2** - Poor documentation;
- **Risk 3** - Failure to deliver on time;
- **Risk 4** - Technology risks.

Table 3.6: Risk Matrix

		Probability		
		Low	Medium	High
Impact	High	**R2**	**R3** **R4**	
	Medium			**R1**
	Low			

As it is possible to see in Table 3.6, the risks that deserve more attention in this work are **R1**, **R2**, and **R3**, located in the orange part of the matrix. Although the risk **R2** is located in the green zone, it should not be discarded, but the amount of paid attention is also less than the previous three.

After this analysis, the risks were properly mitigated and always minimized, which lead to a successful project execution.

3.4 Planning conclusions

The software development process planning is a crucial step in any project, and should not be underestimated. This work was developed using Agile methodology, explained in Section 3.1. The initial work plan suffered changes and the project was not concluded in the end of June as expected, but in the start of September. This required to change the work plan and adjust it to the new deadline. The risks management part helped avoiding the risks to become the true, and was successfully followed. This thesis is an example of a good work planning and its importance.

This page is intentionally left blank.

Chapter 4

Requirements Specification

This chapter discusses the Functional and Non-Functional Requirements of the new version of ONDA.

The definition of this type of requirements is essential in the initial phase of any project. Requirement specification makes it clear and easy to understand the problem that needs to be solved and find the best approach to do it. Each requirement has a number, a description, and a priority.

4.1 Functional Requirements

A functional requirement is any requirement which specifies what the system should do. They appear better explained in the Table 4.1.

For the requirement prioritization it was used the MoSCoW scale [55], in order to keep the deadline. MoSCoW initials stand for:

- Must have - essential requirements, where there is no point in delivering the project without them;
- Should have - important requirements but if not implemented, the system can be still used;
- Could have - desirable requirements that could be implemented if don't affect the deadline of the project;
- Won't have - not implemented requirements during the project execution, due to lack of resources/time but can be implemented in the future.

Table 4.1: Functional Requirements with an identifier, description and defined priority

Functional Requirement No.	Function Requirement Description	Priority
FR 1	User should be able to create two layers, first representing Edge and second representing Cloud.	Must
FR 2	When one of the layers is selected, the new entities must be added to it. The entities can not transit between layers.	Must
FR 3	User should be able to specify the layer name, database engine, and database name of each of the layers' databases	Must
FR 4	User should be able to specify each database engine language (mySQL or postgreSQL)	Should
FR 5	User should be able to select the data warehouse type (single or star)	Should
FR 6	User should be able to create a new data entity ("Data Summary"), which aggregates the selected information from the Edge	Must
FR 7	Allow users to select the data he wants to add to the new entity	Must
FR 8	User should select a mathematical function he wants to apply on the selected data	Must
FR 9	Allow user to select a time to update the database, and the time window between the data collections	Must
FR 10	User should be able to visualize the new table properties	Must
FR 11	Allow user to delete a field of the "Data Summary" table	Must
FR 12	Allow user to change the function to apply on the selected data of the "Data Summary" table	Should
FR 13	Allow users to connect the tables from different layers visually, with links	Must
FR 14	When typing the field name to add to the "Data Summary" table, there should appear an autocomplete dropdown with the available options, to minimize the errors	Should
FR 15	Generate physical representation of the database	Must
FR 16	The physical view of "Data Summary" entity should have the function of the selected field, field name, selected dimensions and its fields	Must
FR 17	The application should generate the final Structured Query Language (SQL) script	Must

When all the Functional Requirements with "Must" priority are implemented, the application should follow the flow from the Fig. 4.1.

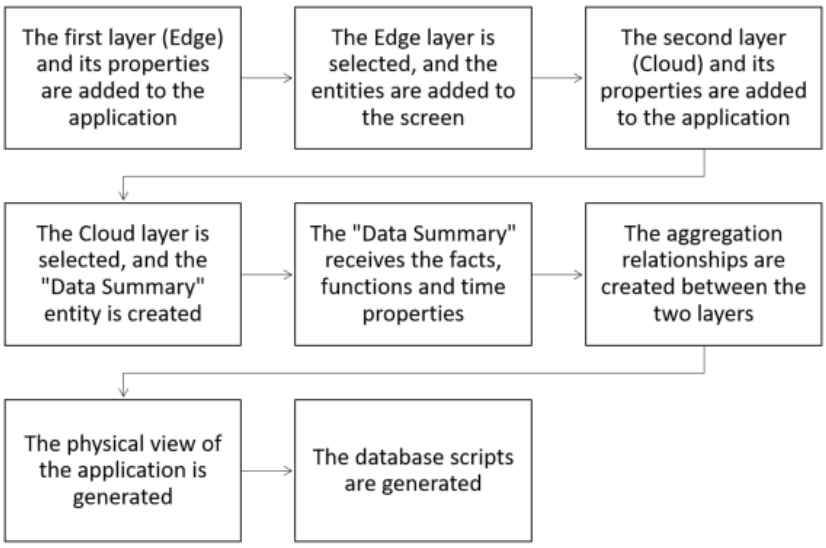


Figure 4.1: ONDA Application Flow after implementation of "Must" Requirements

The first step is to create the Edge layer and add this layer properties such as its name and the database name. The second step is to select this layer and create new entities and to add the respective fields to the tables. It is also possible to upload a previously saved Entity Relationship (ER) model to this layer. In the third place, it is necessary to create

the second layer and also add the properties, as mentioned previously. The fourth step is to add a new entity, named "Data Summary". In the fifth place is to add the facts (entries from the Edge layer tables) to this new entity, aggregation functions (maximum, minimum, standard deviation, among others) that user wants to apply on the data and choose the starting point to collect the data (time window). After this step, it is necessary to click on the Physical View button, to generate the physical representation of the database. The last step is to click on the Script button, where the application generates the final script with two separate databases, regarding to the Edge and the Cloud, according to the selected fields, functions, time, and database properties.

4.2 Non-Functional Requirements

Non-functional requirements are requirements that do not have a specific functionality in the system, they are requirements that help to obtain the desired solution, specifying the qualities that the system can have. Some examples of these requirements are *scalability*, *compatibility*, *security*, and *usability*.

The goal of this project is to extend an existing tool, so in this new implementation only the *usability* was considered.

Usability can be divided into five dimensions [56]:

1. **Learnability** - how easy is it for users to accomplish basic tasks the first time they interact with the design;
2. **Efficiency** - how quickly the user reaches his goals;
3. **Memorability** - if the user can start using the application again after some time without using it;
4. **Errors** - how often users make mistakes;
5. **Satisfaction** - the user evaluation of the user interface.

The Non-Functional Requirement of *usability* will be tested and validated in Section 7.2. The other requirements will not be tested because this project was not designed to have capacity to support significant number of users simultaneously, nor to be compatible with mobile devices. There is no sensitive information involved in the platform, so the *security* is not a requirement as well.

4.3 Mock-ups

The goal of creating Mock-ups is to align the mentioned requirements with the design of the application, to save time and effort during the project implementation.

In the beginning of the second semester, after specifying the requirements, the first low-level Mock-ups were drawn, to better understand the possible final look of the application. Also, it gave a better notion of the interaction between the user and ONDA application. These Mock-ups had a general idea of the different layers and the user interface properties to manipulate the data, as it is possible to see in Fig 4.2. The low-level Mock-ups do not have to represent every detail, and the changes are applied quickly, unlike the high-level Mock-ups, where the design is similar to the final product.

On the left side of the figure, it is represented the first layer ("Edge") and its tables. Some of these entities are connected to the Data Summary entity named "Summary Entity", with

aggregation relationship, represented with a black square, and a line connecting the tables. In the middle there is the new "Cloud" layer, with a data entity named "Summary_Entity", which has facts, functions, time window options, and the aggregation connections from the first layer.

On the right side, it is possible to see a "Properties" panel, regarding the "Summary Entity". These options include the table name, data warehouse type, which will be only single in this new implementation, facts properties, which include an input field to add facts names and associate a function to the fact. Below are the dimensions, the information in this area is updated after a new "aggregation" relationship, from Layer 1 to Layer 2. In the bottom are displayed the time options, which include the time field, starting date from which the user wants to collect the data, time granularity, which is the time window between data collections and the database refresh rate.

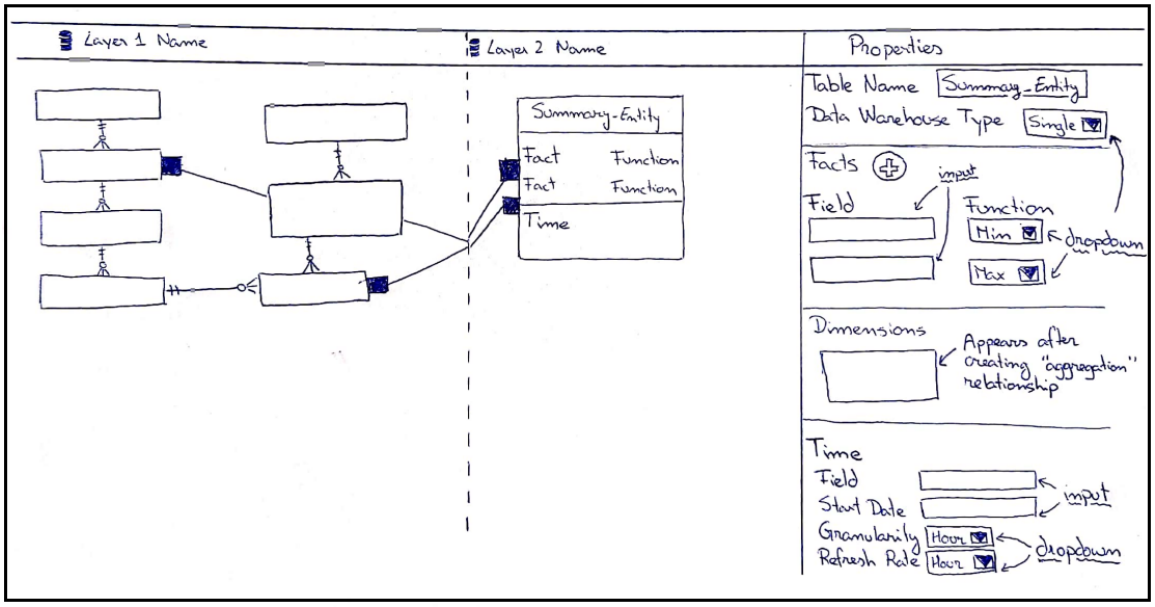


Figure 4.2: ONDA Conceptual View and Data Summary Entity properties Low level Mock-up

Figure 4.3 displays the High Level Mock-up, which gives a clear idea of how the application should look in its final version. It is normal that this representation is not final and can suffer changes, depending on the Clients feedback.

Next will be shown and explained different buttons used in this ONDA implementation. On the left side it is displayed the button mock-up and on the right side there is the actual button.

The first button is the one that adds layer properties, as shown in Fig. 4.4. When user clicks it, the layer properties are displayed on the right panel and it is possible to set the layer and database names.

Figure 4.5 represents the dropdown option buttons. It is possible to select between multiple choices when user clicks on them. For example, when user clicks on "Single", appears an option to select "Single" or "Star" regarding the database type; when user clicks "Min", it is possible to select the mathematical function regarding the fact, such as "Max", "Average", "Count" or "StDev". Last, when user clicks on "Hour", appear options like "Day", "Week", "Month" or "Year", regarding the time window options.

Figure 4.6 refers to the buttons that is necessary to click if the user wants to add a new

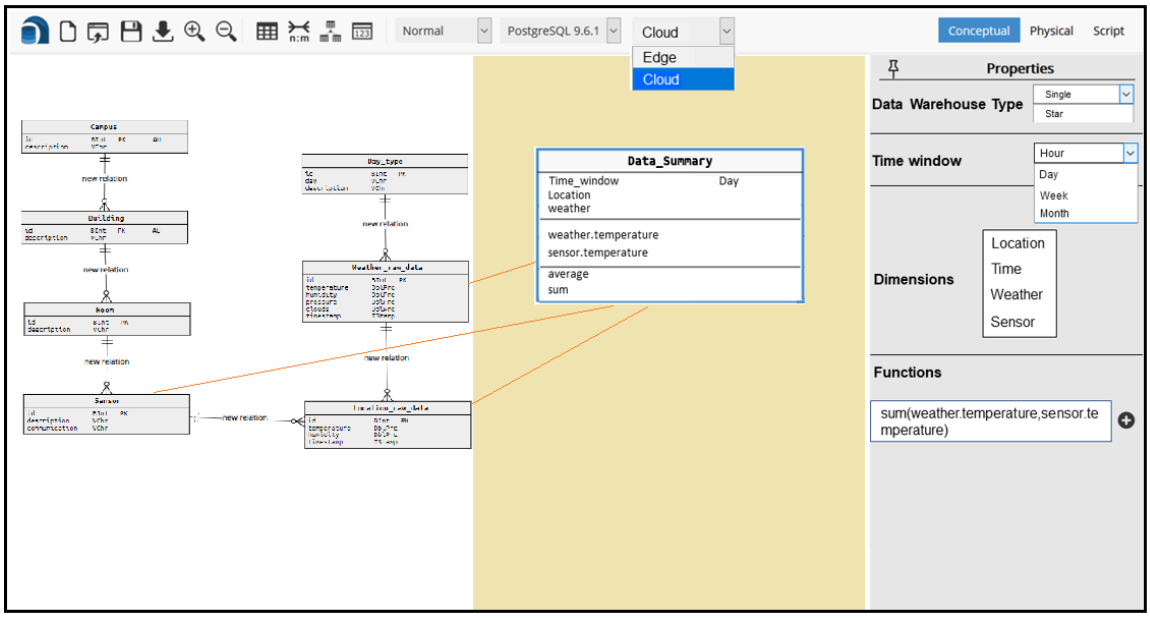
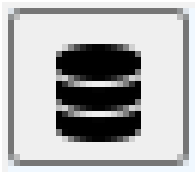


Figure 4.3: ONDA Conceptual View and Data Summary Entity properties High level Mock-up

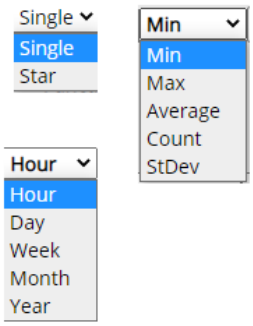
(a) Mock-up of the button to add layer properties



(b) Actual button to add layer properties

Figure 4.4: Buttons to add layer properties

(a) Mock-up of the dropdown buttons



(b) Actual dropdown buttons

Figure 4.5: Dropdown buttons

fact to the "Summary Entity".

(a) Mock-up of the button to add a new fact

(b) Actual button to add a new fact

Figure 4.6: Dropdown buttons

4.4 Requirements Specification conclusions

Requirements Specification is another crucial activity in any software development project. For this thesis, first were defined 17 Functional Requirements, which followed the MoSCoW scale of priorities. These requirements will be testes in Chapter 7. Later, were identified the Non-Functional Requirements and one of them was considered, the usability. Last, were defined the application mock-ups, to align the requirements with the design. These mock-ups suffered little changes and the final product is slightly different from the first idea.

This page is intentionally left blank.

Chapter 5

Architecture of the Application

Current chapter discusses the architecture of the application, the technologies used to implement the Front-End and the Back-End.

ONDA [2] is a database modeling tool created by the Department of Informatics Engineering (DEI) community, developed in *HTML*, *CSS* and *Javascript*. The existence of large number of *Javascript* libraries and a big online community, made the implementation easier than it would be without them. The implementation includes new functionalities, mentioned in Chapter 4.

When a user opens ONDA interface, it is possible to create a new Entity Diagram or open an already saved one, create tables (entities), add relations between two tables and select the type of database engine to generate the outcome script.

By adding an entity, the user creates a conceptual representation of the database. The first field of each added entity is automatically defined as a Primary Key, and can be changed. The relations between tables can be defined as "zero-to-one", "one-to-one", "zero-to-many" or "one-to-many", on each side of the relation. The physical representation is a translation of the conceptual model. The physical model adds the Foreign Key (FK) associated with the relationships, and may add new tables depending on the type of relationships defined in the conceptual model. The Structured Query Language (SQL) script is the final result of the design process. It provides the code, to be imported into a database engine, containing the instructions represented through the physical model.

Currently, ONDA does not allow to create two distinct databases in the same diagram, neither it is possible to generate a script that stores and aggregates all the data in the same place. Because of these limitations, the final product is aimed to generate a SQL script, to help database architects to create two distinct databases in different physical places. Here it is introduced the concept of layers, which in this thesis have the name of "Edge" and "Cloud". The possibility to use both Edge and Cloud services aims to enhance the system performance.

This Chapter is divided in two Sections. Section 5.1 explains the general application architecture of both current and new implementations and technologies that will be used to implement the new solution. Section 5.2 has an explanation of the current Application Flow that leads to the script and also has the steps to follow in order to produce the new expected result.

5.1 Architecture and Technologies

For the implementation purpose, it was produced a System Architecture Diagram, which is a good starting point for diagramming the software system, and allows to see the big picture of the application (Fig. 5.1).

This diagram applies to the existing ONDA implementation, and to the new version.

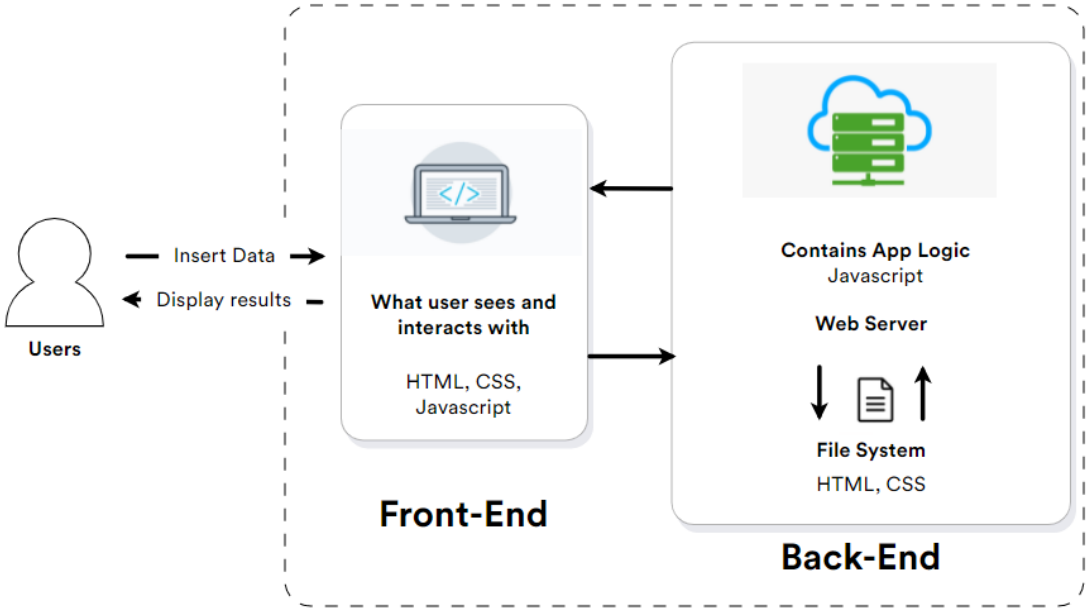


Figure 5.1: ONDA System Application Diagram

5.1.1 Application Scheme

Front-End

Application Front-End refers to the part that user sees and interacts with. Since the Front-End of the previous versions of ONDA was all developed using *HTML*, *CSS* and *Javascript*, the idea was to continue and extend the existing work, only with a goal to add new functionalities. For the drawing purposes and drag and drop functionalities, it was used *JQuery* version 1.12.1 (*Javascript* library), which has a free version of a library named *joint.js* [57].

Back-End

The Back-End refers to all the logic part implemented in the application, from the button click handlers, to insertion of tables, its properties, and final script generation. For the implementation of the Back-End logical structure, it was used *Node.js*, a Javascript runtime engine. Node allows to run the code on the web browser and receive and send information between Front-End and Back-End in JSON format, to produce the content on the browser. The development of Back-End functionalities, as in the original version, was through the Javascript language.

For this project, there is no need to have a database attached, since all the information

is stored locally on the web server, and as soon as the browser is closed, the information is lost. This can be prevented by saving the conceptual diagram into a JSON file, which later can be retrieved and loaded into the application.

5.1.2 C4 Architecture Model

To better understand the application architecture, it was used an existing model with a graphical notation technique, named C4 Architecture Model [58].

It is composed by four hierarchical levels:

- **Context diagrams** are the level one and they show the system and its relationship with users and other systems;
- **Container diagrams** are the level two and they decompose the system into inter-related containers. A container represents level one applications or data stores;
- **Component diagrams** are the level three and they decompose containers into inter-related components, and relate the components to other containers or other systems from level two;
- **Code diagrams** are the level four and they provide additional details about the design of the architectural elements from level three that can be mapped to code.

Accordingly to the official website of this model [58], the level three and four diagrams should only be done if the developer feels it adds value. In case of this work the application already exists, there is no need to identify which components will be used, how they will be structured, and how they will be interacting with other structural blocks.

Context diagram

The first level of this architecture provides a starting point of the application, showing the software system and the interactions it has with the world. At this level the detail is not important, since it is explained in the next levels.

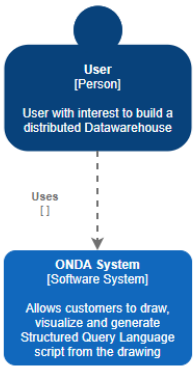


Figure 5.2: ONDA Context Diagram

Figure 5.2 has a User, which is a person who uses the Software System. The Software System is the highest level of abstraction, it is something that delivers value to the users. This specific application allows users to draw, visualize and generate SQL script from the drawing.

Container diagram

In the second level of abstraction, the system is "zoomed in" from previous level and divided in containers. A container is something that needs to be running in order for the software system to work.

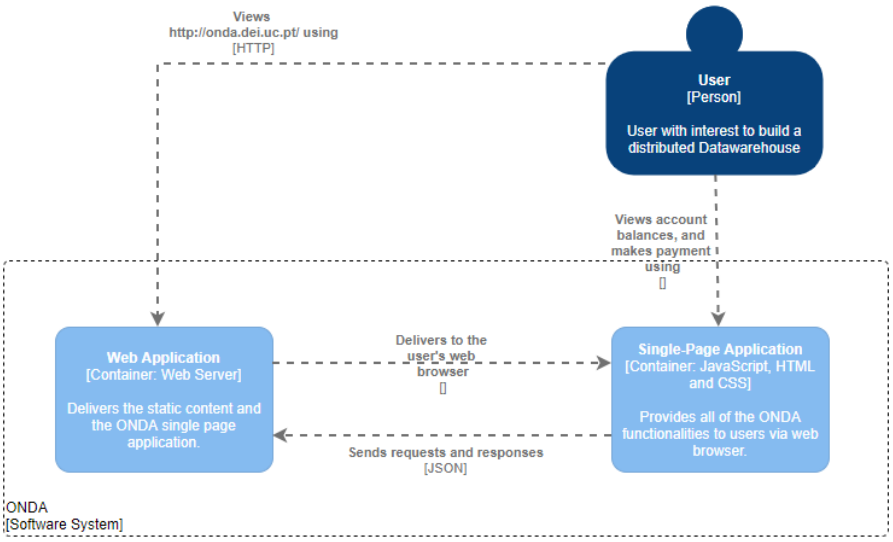


Figure 5.3: ONDA Container Diagram

In the Figure 5.3, the containers are represented as blue rectangles. The Web Server delivers the static content and the ONDA single page application. The Single Page Application provides all of the ONDA functionalities to users via web browser, it is the Front-End of the application.

5.2 New ONDA Version

The Application Flow of the new tool version is the same as the existing in the current ONDA version, as represented in the Fig. 5.4.

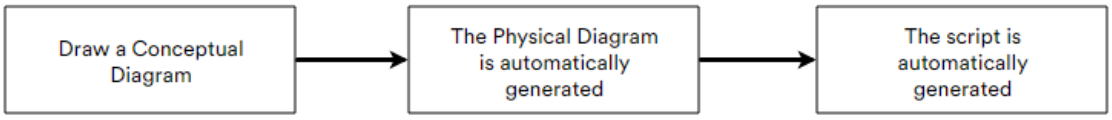


Figure 5.4: ONDA Application Flow

First, the user creates tables and relations on the Conceptual View, then it is automatically generated a Physical View considering this information, and in the last step, it is generated a Script with the data entities, their properties, and relations.

In the new ONDA implementation, the idea is the same, but the steps to generate the final script are different from the old version, as Fig 5.5 provides an example.

The application automatically generates a Physical representation of the database from the Conceptual view. Then, the generated script is different from the normal ONDA version, because it includes two distinct database configurations, the "Edge Layer" and the "Cloud

Layer" databases, which has a Single "Data Summary" entity, with all the data aggregation options.

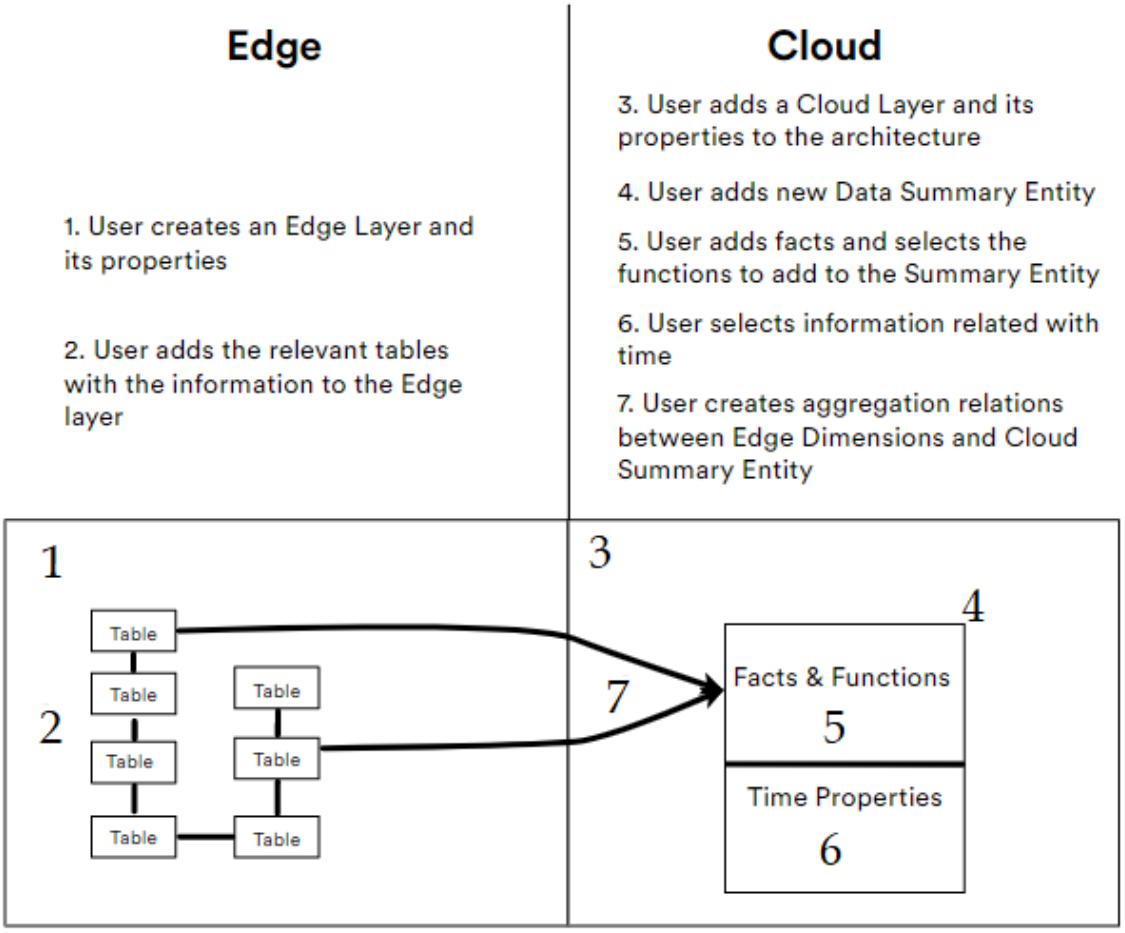


Figure 5.5: Sequence of steps to automatically generate the script

In this new platform version were included different modifications, such as:

- Included a new button to create layers;
- Layer properties menu, where it is possible to change the database and layer names;
- Possibility to add tables to the new layers;
- Created a new "Data Summary" entity;
- Added options menu to the "Data Summary". In this menu it is possible to add/remove facts and functions, add/remove time window properties;
- Added new type of relationship - aggregation;
- Created table relationships between different layers, represented by lines and squares;
- Changed the physical view of the "Data Summary" entity;
- Added support to the new data warehouse script.

5.2.1 Architecture of the Application conclusions

ONDA is a database modeling tool created by the DEI community. The application can be divided into two parts, the Front-End (the view), and the Back-End (the logical part). To represent it better was developed a C4 Architecture Model, composed by four levels, but only represented by the first two, the other two are optional and big level of detail doesn't add much value to this project. The new ONDA version with the new modifications were also explained in this Chapter. The application architecture is important to identify the technologies that will be used and how they will interact within the platform and with the outside world.

This page is intentionally left blank.

Chapter 6

Implementation

This chapter shows the most relevant details of the final project implementation part, with an explanation of major problems that appeared during the development.

The ONDA user interface was build after defining the Functional Requirements and low-level mock-ups, defined in Sections 4.1 and 4.3. These suffered continuous improvements, which aimed the most intuitive usability of the application. For this reason, the final result is not exactly the same as it was defined in the beginning.

In the Figure 6.1 is shown the application version as it was in the beginning of extension.

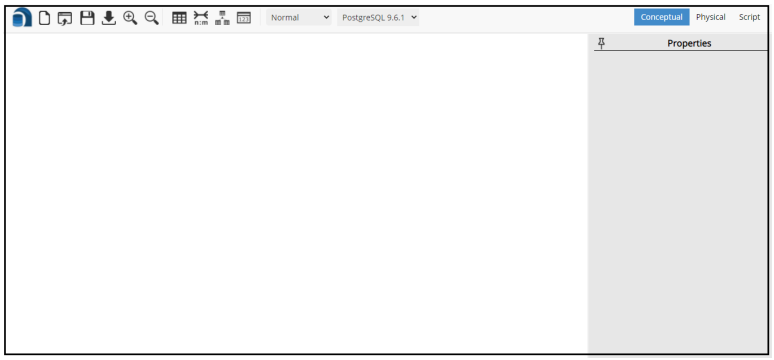


Figure 6.1: ONDA Interface before extension

6.1 Project Development

This section clarifies the different phases of the project development. This development can be divided in five parts: Layers, "Summary Entity", Aggregation connections, Physical Diagram, and Scripts.

The application main screen can be divided in three main parts: the properties bar on the top of the page, the manipulation canvas, where the user adds new tables and creates relationships between them, and the table properties tab, where the user adds, removes and changes the table properties.

6.1.1 Layers

In the first phase, the aim was to introduce a concept of creating a new layer and being able to add tables to each one of them, where it would not be possible to drag tables from one side to another. One of the mentioned layers will represent the database placed on the Edge of the network and the other will represent the data warehouse located on the Cloud. Figure 6.2 shows the new properties bar, where exists a button to add a new layer to the application. This button is highlighted in red.

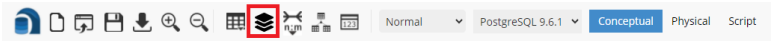


Figure 6.2: Application properties bar

After clicking this button, the application looks like in the Figure 6.3. It divides the program in two separate layers. To add a table to the respective layer, it is necessary to select a side and then the entity will be added to it. This concept was implemented with help of external library named *Joint.js*, which helps with canvas manipulation and attaches entities to the pretended side.

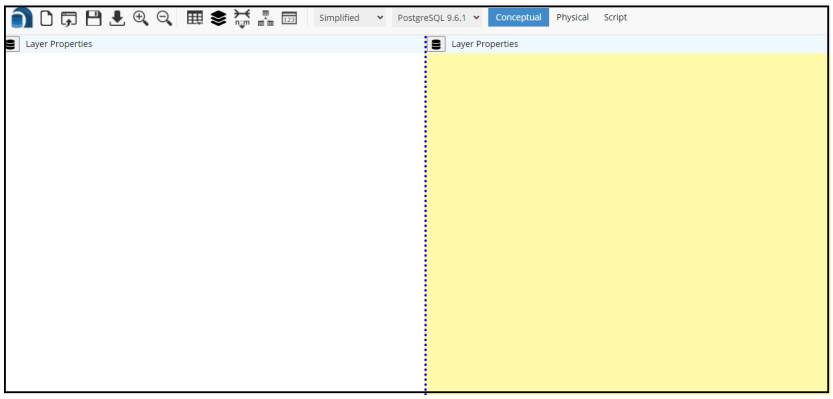


Figure 6.3: Two distinct ONDA layers

In the first implementation phase, arose the Risk 4 - Technology risk, which was mitigated by discarding Requirements with "Should" priority, in order to deliver the application in time. As this work is an extension of a tool, the problem was to first understand the previous implementation, regarding the used library and the existing project files and functions. The rest of the application was developed in *Javascript*, *HTML* and *CSS*, so it was necessary to continue the implementation using these.

6.1.2 "Summary Entity"

The second phase was the implementation of the "Summary Entity". This entity summarizes all the relevant information that can be selected in the table properties. This information regards to the field names, aggregation functions ("Min", "Max", "Count", "Average", "StDev") that apply to each field, dimensions, which are the table names where the aggregation start, and time properties, as explained in 2.3.

To add this new table, it is necessary to click on the button highlighted in Figure 6.4 and select "Summary Entity" from the displayed options.

Figure 6.5 displays and example of the "Summary Entity" table and its properties.

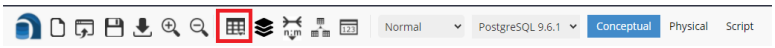


Figure 6.4: Add Summary Entity table

The example table has 12 fields, each of them with a function that applies to it. Each field starts with its source table name, to indicate where the data is coming from, because there could exist more than one table with the same field name. For this reason, it is not possible to see the full length table field.

On the right side of the table are displayed the table properties.

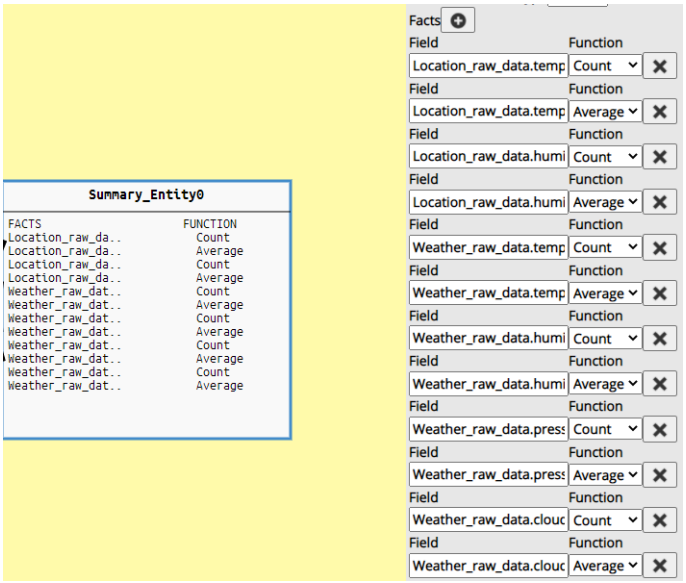


Figure 6.5: Summary Entity table and its properties

It is possible to add more fields, by clicking the "plus" button near "Facts". This function displays an input field, with autocomplete function, to minimize the amount of errors on the input, as it displayed in Figure 6.6. When the user selects a pretended field, it is highlighted with a green color, then it is necessary to press the "check" button.

To change a field function, with a click on the dropdown which has "Min" as default, from there it is possible to select a pretended function.

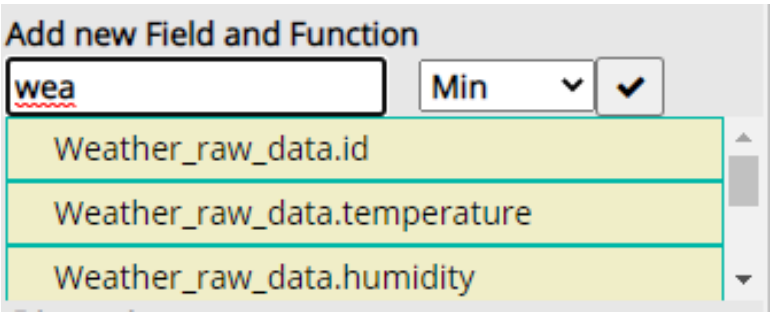


Figure 6.6: Autocomplete to add a new table field

The table properties also allow to change the function, without having to delete the field. Each fact has a function attached to it, then it is necessary to click on the dropdown option and simply change the function, which updates the system. It is also possible to delete any field, simply by pressing the "x" button.

6.1.3 Aggregation connections

The third phase was the implementation of the visual connections to represent the aggregation of tables with the "Summary Entity". Once more arose the R4 - Technologies risk, because it was necessary to use the *Joint.js* library. This library also helps to create visual relationships between tables. The R4 risk was properly mitigated, as previously explained in Chapter 3.

To create this visual connection, it is necessary to click on the button highlighted in Figure 6.7 and select "Aggregation" from the displayed options.

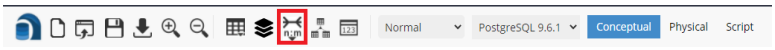


Figure 6.7: Create aggregation relationship

The next step is to select the first table, where the relationship starts. After it, simply click on the "Summary Entity" table, and the connection is created, as displayed on the Figure 6.8. This table can aggregate more than one relationships, that appear on table properties, and later is generated in the script.

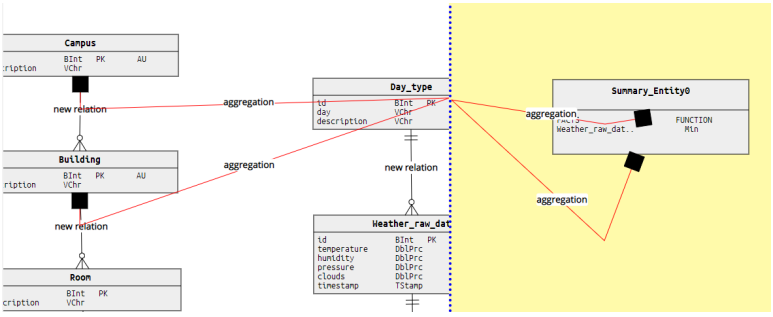


Figure 6.8: Aggregation relationships example

As it is possible to see, the connections design are not perfect and will be perfected in the next versions of this prototype implementation. The line that starts on the tables on the left goes until the middle of the screen, and after starts in the same point and goes until the "Summary Entity". This part gave a lot of problems because of the existence of two layers, and this connection was not easy to achieve.

6.1.4 Physical Diagram

The fourth phase was the implementation of generation of the physical diagram. Without this step, it is impossible to achieve the last step, which is the automatic generation of the script.

To generate the physical view of the application, it is necessary to click on the button named "Physical", highlighted in Figure 6.9.

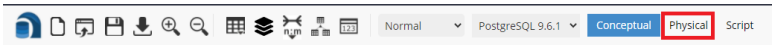


Figure 6.9: Generate physical diagram

The Physical view is different from Conceptual, because there don't appear aggregation relationships, and the information is displayed differently. First appears the summarization

function, then the field and then the field data type, as it is shown in Figure 6.10. In the end appear the dimensions and respective fields, which come from aggregations.

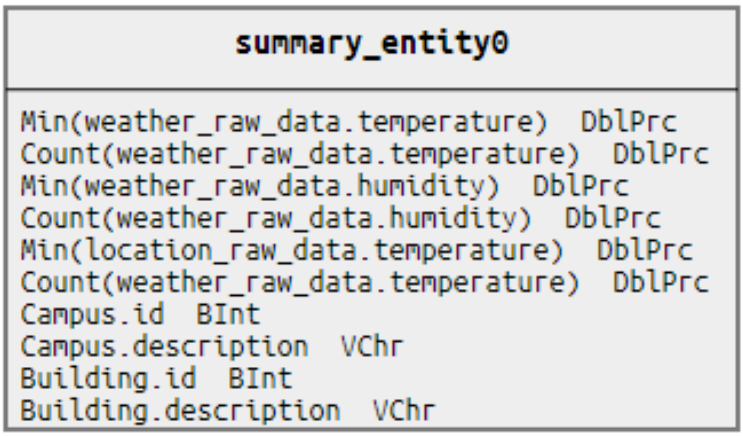


Figure 6.10: "Data Summary" physical view example

6.1.5 Scripts

In the fifth and final phase, the script generation was defined. This is the main objective of the application, and it is very important that the generated script outputs a valid Structured Query Language (SQL) code. To generate the result, it is necessary to click on the highlighted button in Figure 6.11.

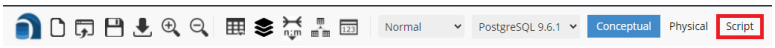


Figure 6.11: Generate scripts

It is also necessary to specify the layer properties, as shown in Figure 6.12. These properties include the layer and database names, and database language engine. The user can choose this last option, but it was not implemented and it was established that the application will only generate MySQL scripts.

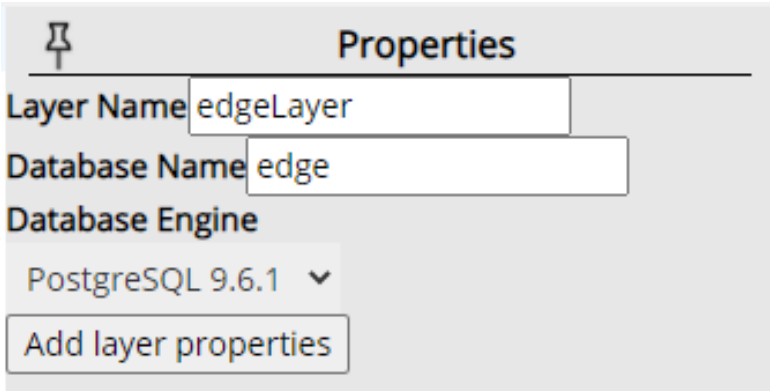


Figure 6.12: Layer Properties example

The final output creates the first layer database, its tables, with respective fields and relationships. Then, it creates the second layer database, "Summary Entity" table, with the fields that belong to it. Finally, it generates the events. The events include information with

the start time from when the data is being collected, what data needs to be summarized, aggregated and inserted in the database, and the time interval between these insertions.

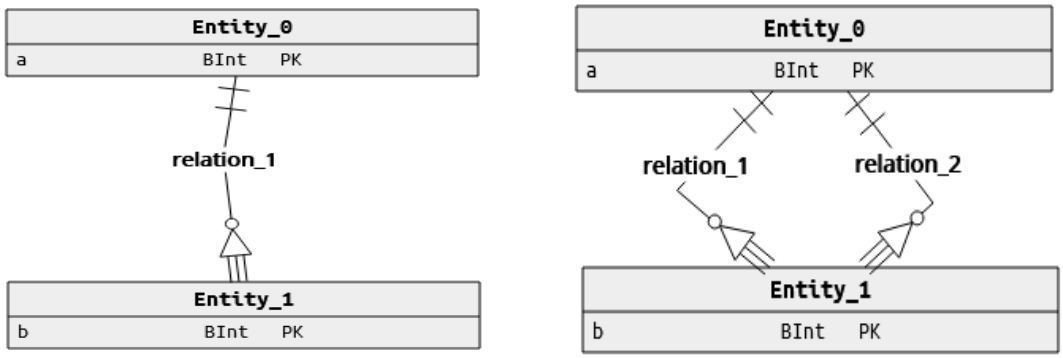
6.2 Additional improvements

Before the actual implementation and extension, the starting point was to resolve a problem that ONDA had. This would be a good starting point to a better understanding of how the tool works. The most critical problem was in the generation of the Physical diagram from Conceptual view, when existed over one weak relation between the tables.

Conceptual View

To explain the existing problem in the tool, Fig. 6.13 will show the difference between the Conceptual View and the Physical View.

As it is possible see in the Figure.6.13a and Figure.6.13b, the conceptual view of the two entities presents the information inside the tables and their relationships. This information is the same, excepting that the Fig. 6.13b has two weak relations, while Fig.6.13a only has one.



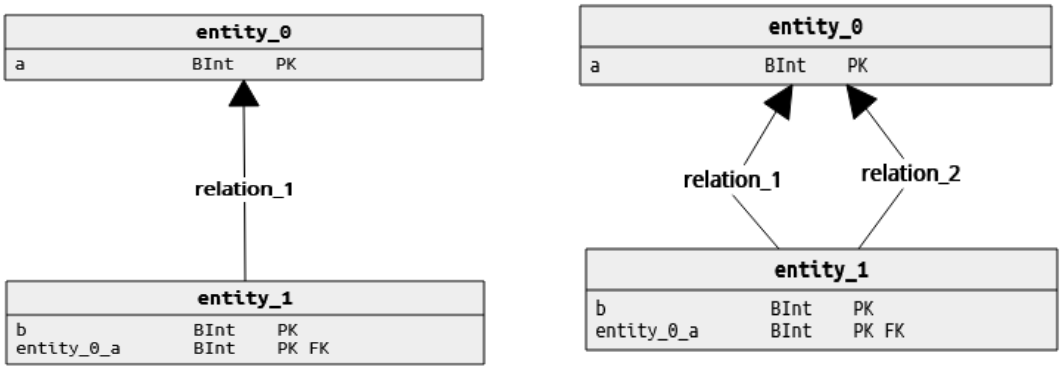
(a) Conceptual diagram with one relation (b) Conceptual diagram with two relations

Figure 6.13: Conceptual View of ONDA tables and its relations

Physical View

After the user creates a conceptual schema, the next step is to generate the physical diagram. As it is possible to see in Fig.6.14 the "entity_1" of the Fig.6.14b only has two entries and should have one more. For each existing relation to the same table it should create a new entry and this did not happen.

In the Fig.6.14b is shown the physical diagram with a problem. The physical table named "entity_1" is exactly the same as in the Fig. 6.13a, where only exists one relation.

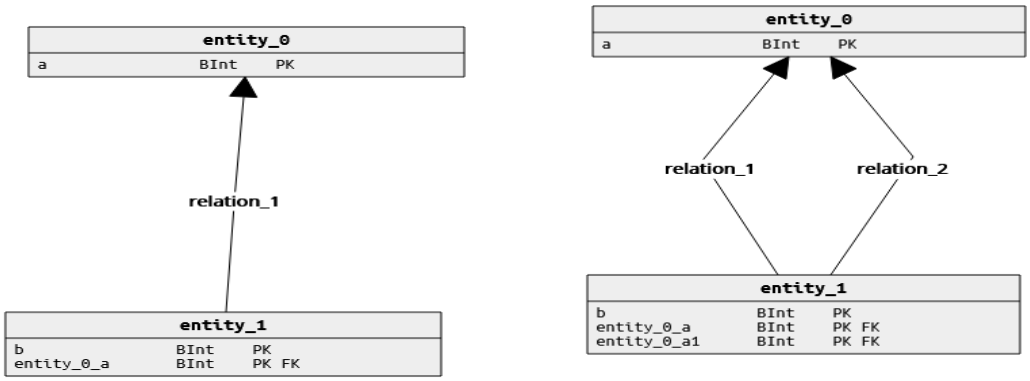


(a) Physical diagram with one relation

(b) Physical diagram with two relations - wrong version

Figure 6.14: Wrong version of physical diagram

This problem was corrected and the source code was sent back to the Professor. After this step started the implementation mentioned before in this Chapter. Fig.6.15 shows how things are in the correct version of ONDA. When a new weak relation is created, a new entity is added to the related table, as we can see in Fig.6.15b. The table named "entity_1" has "entity_0_a" and "entity_0_a1", which didn't happen in the Fig.6.14b.



(a) Physical diagram with one relation

(b) Physical diagram with two relations - correct version

Figure 6.15: Correct version of physical diagram

6.3 Implementation conclusions

The implementation part of this work, apart from the writing this thesis, was one of the hardest challenges. Once the requirements were outlined and the additional improvements finished, started the actual implementation, which lasted almost six month. The aggregation connections required knowledge of an external library, which had to be studied and tested, the "Summary Entity" with its properties was also difficult to implement. The implementation will be finally tested in the next Chapter, to identify what can be improved or changed.

This page is intentionally left blank.

Chapter 7

Tests

This chapter describes the Functional and Non-Functional tests performed on the developed software. It is also performed an application validation, with comparison of the generated and expected scripts, in order to understand if the final product works as it should.

7.1 Functional Tests

Functional Testing is a quality assurance technique for testing software solutions in various levels to test its functionalities in order to avoid failures. This type of testing is of black box type, which means that all test cases are created based on the project specifications and with no knowledge of the implemented code. This way, these tests focus on what the the solution does, caring with the inputs it takes and the outputs it gives. The tests were performed on the most important requirements, defined in Table 4.1, with priority *Must Have*. In the next Table is displayed an example of a test case in this project.

Table 7.1: Functional Test Template

ID	Test ID
Requirement ID	Requirement ID to which the test belongs
Title	Brief test description
Result	**Pass** in case the test produces an expected result **Fail** in case the test produces an unexpected result

A test case is defined by the test ID, the tested requirement ID, a test title, which is the brief test description and the test result, with its outcome. In the end of each table there is produced the sequence of the steps to perform the test.

7.1.1 FR1 - Functional Requirement 1

This Functional Test with ID FT1 tests the first Functional Requirement - FR1.

Table 7.2: Test of Functional Requirement 1

ID	FT1
Requirement ID	FR1
Scenario	User should be able to create two layers, first representing Edge and second representing Cloud
Result	**Pass**, the system has two different layers created

The application has a new button with label "Add Layer", where user clicks to create it. ONDA already has the first layer created by default, and it is possible to change its properties. When the button is clicked, appears a message saying "Layer Created" and the new layer is created with a different color and separation, in order to distinguish the layers.

7.1.2 FR2 - Functional Requirement 2

This Functional Test with ID FT2 tests the second Functional Requirement - FR2.

Table 7.3: Test of Functional Requirement 2

ID	FT2
Requirement ID	FR2
Scenario	When one of the layers is selected, the new entities must be added to it. The entities can not transition between layers
Result	**Pass**, different tables were created in respective layer and could not be dragged between layers

After creating a new layer, the user clicked on the first that was already created and added a new entity, by clicking on "Add Entity" button, and selected "Entity". Then, the user added a "Summary Entity". The same procedure was done by clicking on the new layer, and the tables were added successfully. In the end, when the user tried to drag the tables from one layer to another, it was not possible.

7.1.3 FR3 - Functional Requirement 3

This Functional Test with ID FT3 tests the third Functional Requirement - FR3.

Table 7.4: Test of Functional Requirement 3

ID	FT3
Requirement ID	FR3
Scenario	User should be able to specify the layer name, IP address, database engine,database name, and database credentials of the layer 1 and 2 databases
Result	**Pass**, layer properties were added successfully

On the upper left side of the screen, exists a button with a text saying "Layer Properties". The user clicked on the button, filled in the information with layer and database names,

and clicked the button "Add layer properties". In the bottom of the screen appeared a message "Layer Properties Added Successfully".

7.1.4 FR6 - Functional Requirement 6

This Functional Test with ID FT4 tests the sixth Functional Requirement - FR6.

Table 7.5: Test of Functional Requirement 6

ID	FT4
Requirement ID	FR6
Scenario	User should be able to create a new data entity ("Data Summary"), which aggregates the selected information from the Edge
Result	**Pass**, the new entity was created successfully

The user created a new layer, named "Edge", clicked on it and added a new entity named "Data Summary", through the "Add Entity" button. The table appeared on the screen and it was possible to see its properties.

7.1.5 FR7 - Functional Requirement 7

This Functional Test with ID FT5 tests the seventh Functional Requirement - FR7.

Table 7.6: Test of Functional Requirement 7

ID	FT5
Requirement ID	FR7
Scenario	Allow users to select the data he wants to add to the new entity
Result	**Pass**, user was able to add all the pretended data to the new entity, and even delete the unwanted data, without problems

After creating a new data entity, it was possible to add a new field, through the input present on the table properties. When the user typed the fields he wanted, the application made suggestions, in order to minimize the errors and help the user. Afterwards, it was necessary to confirm the field by clicking on the button with a "check" icon.

7.1.6 FR8 - Functional Requirement 8

This Functional Test with ID FT6 tests the eighth Functional Requirement - FR8.

Table 7.7: Test of Functional Requirement 8

ID	FT6
Requirement ID	FR8
Scenario	User should select an aggregation function he wants to apply on the selected data
Result	**Pass**, it was possible to add a function and change it

When the user selects a field from the input, the default function attached to it is "Min". It is possible to change to four other functions, such as: "Max", "Average", "Count" and

"StDev". Afterwards, it is necessary to confirm the operation, by clicking on the button with a "check" icon.

7.1.7 FR9 - Functional Requirement 9

This Functional Test with ID FT7 tests the ninth Functional Requirement - FR9.

Table 7.8: Test of Functional Requirement 9

ID	FT7
Requirement ID	FR9
Scenario	Allow user to select a time field to update the database, and the time window between the data collections
Result	**Pass**, the time window was selected and appears on the final Structured Query Language (SQL) script

After adding all the relevant table fields, the user needed to pick which is the time window field to group the data on, the data granularity, and the time frequency to update the database. After this, the user clicked on the button with a "check" icon and the time properties were added successfully.

7.1.8 FR10 - Functional Requirement 10

This Functional Test with ID FT8 tests the tenth Functional Requirement - FR10.

Table 7.9: Test of Functional Requirement 10

ID	FT8
Requirement ID	FR10
Scenario	User should be able to visualize the new table properties
Result	**Pass**, the table properties appear successfully

When the user clicked on the "Data Summary" table, the selected data and time window appeared successfully. The problem was when the user clicked on more than once on the table, the dimensions were being duplicated. It was a visual problem, that did not affect the final script, but made an illusion that the dimensions were being added continuously.

7.1.9 FR11 - Functional Requirement 11

This Functional Test with ID FT9 tests the eleventh Functional Requirement - FR11.

Table 7.10: Test of Functional Requirement 11

ID	FT9
Requirement ID	FR11
Scenario	Allow user to delete a field of the "Data Summary" table
Result	**Pass**, the intended field was deleted

After adding a new field to the "Data Summary" entity, it was also possible to delete the selected data field and add different information, by clicking on the button with a "x" icon.

There exists more than one delete button, one for each table field, this guarantees that only the selected field was deleted. The user selected the table by clicking on it, and then pressing the delete button.

7.1.10 FR13 - Functional Requirement 13

This Functional Test with ID FT10 tests the thirteenth Functional Requirement - FR13.

Table 7.11: Test of Functional Requirement 13

ID	FT10
Requirement ID	FR13
Scenario	Allow users to connect the tables from different layers visually, with links
Result	**Pass**, the links are added successfully

The user clicked on the first layer, pressed the "Add Relation" button, and selected "Aggregation". Then, selected the table to begin the relation. After, he clicked in the "Data Summary" entity and created a new "aggregation" relationship. Every time the user wanted to create a relation between tables, he needs to click on the layer where the first table is. The test result is a pass, but sometimes the connections are not clear enough.

7.1.11 FR15 - Functional Requirement 15

This Functional Test with ID FT11 tests the fifteenth Functional Requirement - FR15.

Table 7.12: Test of Functional Requirement 15

ID	FT11
Requirement ID	FR15
Scenario	Generate physical representation of the database
Result	**Pass**, the physical view was generated successfully

The user introduced the tables and its relations and clicked on the button "Physical". The aggregation relationship connections do not appear on the physical diagram, but only the table fields from the aggregated tables (dimensions), which appear in the "Summary Entity" table, as it is supposed to be.

7.1.12 FR16 - Functional Requirement 16

This Functional Test with ID FT12 tests the sixteenth Functional Requirement - FR16.

Table 7.13: Test of Functional Requirement 16

ID	FT12
Requirement ID	FR16
Scenario	The physical view of "Data Summary" entity should show the the selected field function, field name, selected dimensions and its fields
Result	**Pass**, all the information appears as expected

The "Summary Entity" has the selected field names and respective functions and also, as mentioned in Table 7.13, it does not have the aggregation connections, but possesses the dimension fields.

7.1.13 FR17 - Functional Requirement 17

This Functional Test with ID FT13 tests the seventeenth Functional Requirement - FR17.

Table 7.14: Test of Functional Requirement 17

ID	FT13
Requirement ID	FR17
Scenario	The application should generate the final SQL script
Result	**Pass**, the application generated a valid script

After adding the layer properties and creating the conceptual and physical schemas, the user clicked on "Script" button, located on the upper right size of the screen, which generated the SQL script. This script was validated and will be explained in the next Section.

7.2 Application Validation

In order to validate the application, were produced two proofs of concept. In both of them, the scripts were also created manually, to validate and compare them.

7.2.1 First Use Case

The first use case refers to a meteorologic data collection related to the city of Coimbra, from Internet of Things (IoT) sensors installed in Department of Informatics Engineering (DEI) and from the OpenWeatherMap API. An Edge server gathers this meteorologic data and then the user aggregates and summarizes these data into a Cloud Data Warehouse.

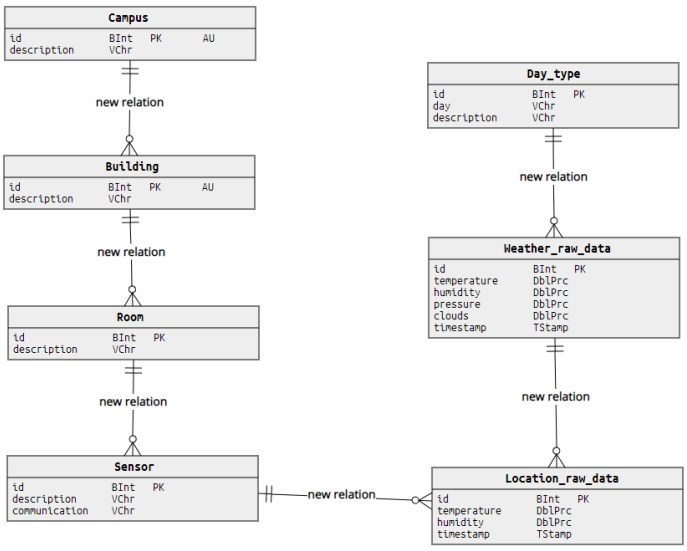


Figure 7.1: Tables from the Edge

Figure 7.1 displays the tables which collect such data, with fields and their relationships, while Figure 7.2 displays what information will be gathered on the cloud. The data are collected from tables "Location_raw_data", which is the actual sensors from DEI and from "Weather_raw_data", which are the information from OpenWeatherMap. All the collected data in this use case, are the count of the times that the data appears in a certain time window, and its average. The data started to be collected on 12/07/21 and are updated every hour. The data are aggregated between tables "Campus", "Building", "Room" and "Sensor" from the Edge, and the table "Summary_Entity_0" from the Cloud.

The application produced a script, which was compared with the script produced manually and was approved by the project supervisors.

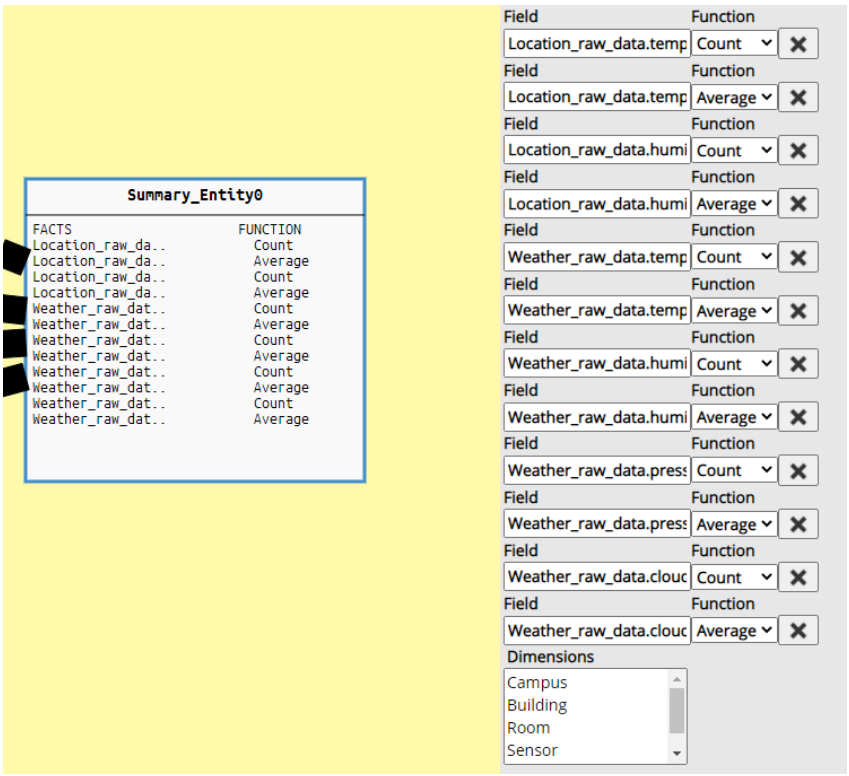


Figure 7.2: Data Summary table from the Cloud

The final script that the application produces is located in the Appendix 8.3. The first part of the script, until the "Cloud data warehouse" comment regards the database located at the Edge of the network. From this part until the end, it regards the Cloud data warehouse and the data selection.

It is possible to see that the data is collected from the 12th of july of 2021, with an interval of one hour between the data.

7.2.2 Second Use Case

The second use case was about a sales shop, represented in Figure 7.3.

In this scenario it is pretended to discover the average amount of money spent by a costumer in a store in a defined product, as can be seen in Figure 7.4.

This use case produced practically everything right, but the operation "Group by" is done by the shop_id, because it is the first table of the aggregations. The correct version would

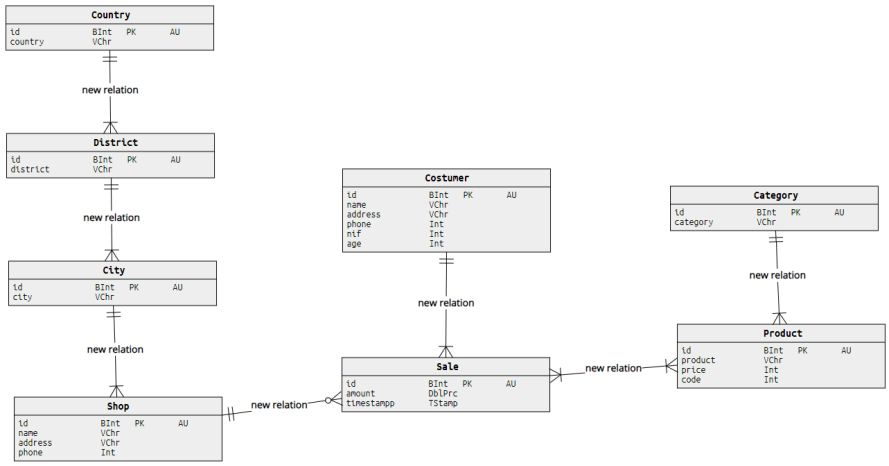


Figure 7.3: Sales Shop Tables from the Edge

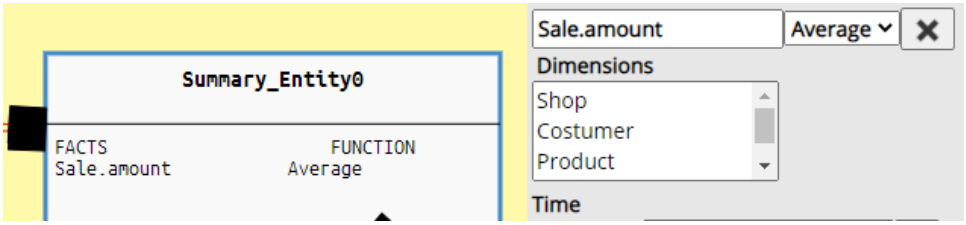


Figure 7.4: Data Summary table from the Cloud

have the information grouped by the costumer_id, because its granularity is the lowest. The initial requirements do not include this issue and it was only discovered in the late part of implementation, with the only option to leave it for the future work.

The final script that the application produces is located in the Appendix 8.3. The first part of the script, until the "Cloud data warehouse" comment regards the database located at the Edge of the network. From this part until the end, it regards the Cloud data warehouse and the data selection, like in the example above.

It is possible to see that the data is collected from the 12th of august of 2012, with an interval of one hour between the data.

7.3 Non-Functional Tests

As mentioned in section 4.2, the only Non-Functional Requirement that will be tested is *usability*. For this purpose, the application was tested by a database architect, who pretended to build a data warehouse.

The tests were conducted by one responsible, the facilitator, who asked the database architect to perform simple actions, without telling explicitly where to click and how exactly do the operations. In the end, the participant was asked some questions related to the performance of the application:

- Question: How would you describe your overall experience with the application?
  Answer: "The overall experience is good, the application is easy to use and the design is intuitive. As a database engineer, I would use it whenever I needed, it's a good tool."

- Question: What did you like the most about using ONDA?
  Answer: "The auto-complete feature, makes it very easy to add new facts and avoid errors when typing."
- Question: What did you like the least?
  Answer: "It was somewhat difficult to create the aggregation connections, there could exist a manual explaining it."

Tasks list

1. Create a new layer and add layer properties;
2. Add a table to each layer;
3. Add facts to the "Data Summary" entity;
4. Remove facts from the "Data Summary";
5. Create "aggregation" between tables from layer one to the "Data Summary";
6. Generate Physical View;
7. Add time window properties;
8. Generate MySQL script;

In eight tasks, the participant had trouble in the task number five - Create "aggregation" between tables from layer one to the "Data Summary". This means that the user efficiency was 87.5%. After the questions and answers, the user was satisfied with the application and rate it with an eight in ten grade.

7.4 Tests Conclusions

As it is possible to verify from the above tests, all the requirements with priority "Must Have" were implemented and the project prototype passed successfully on the tests.

During the tests, appeared a problem in FR10 because after creating the "Summary Entity", which could induce the user in mistake, when every time the table was clicked on and the dimensions kept appearing duplicated, as shown in Figure 7.5.

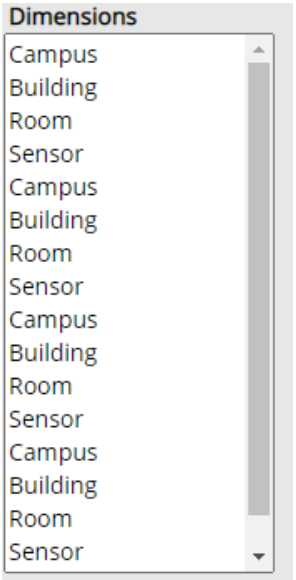


Figure 7.5: "Summary Entity" properties error

The other problem exists in drawing "aggregation" relationships. The connections can get confusing sometimes, making it difficult to understand what is happening.

The Functional and Non-Functional tests served to validate the application and to identify what can be improved in the next ONDA versions. The platform was validated because the generated scripts produced the expected outcomes, and all the indispensable Functional Requirements passed tests successfully. Two of four Functional Requirements with the "Should" priority were not implemented, the specification of the database engine language and the selection of the data warehouse type. On the other side, the changing of the summarization function and the autocomplete feature were implemented successfully and work as it is expected.

This page is intentionally left blank.

Chapter 8

Conclusions

In this chapter are done the conclusions of the thesis, approaching the acquired experience, what more could be done in the future and the final considerations.

8.1 Experience acquired

During the first semester of the internship, the work was focused on the research of the theoretical concepts, writing of the thesis, and the Comparative Analysis of Data Modeling Design Tools scientific paper. In this semester, the research and writing skills were improved.

In the second semester the writing of the thesis and the scientific paper continued, simultaneously with the development of the application. The definition of a good work plan with risk management techniques and risk mitigation strategies were crucial for the project success. Then, the requirements gathering and mock-ups drawing helped visualising the application final look. In this semester the programming skills were tested, from the logical (back-end) part to the visual representation (front-end). The biggest issue was related to the fact that it is an application extention, with a goal to change less code as possible, what sometimes was a big challenge. The external libraries were difficult to understand, as well as the existing code, but it was overpast with the mitigation techniques from Chapter 3. By going through the all phases of the software development, from the project management, to the final work testing, the acquired experience will help entering into the real world projects.

8.2 Future Work

As future work of this application, it will be interesting to develop or correct the next topics:

- Correct the "Summary Entity" properties bug that duplicates the displayed data;
- Calculate the lowest granularity of the tables or set it manually in table options, in order to group the information correctly;
- Make aggregation lines look more attractive to the user;
- Add support to PostgreSQL database engine, as defined in the Functional Requirement 4;

- Add support to the "Star" Data Warehouse, as defined in the Functional Requirement 5;

These bullet options were not implemented in order to deliver a working prototype by the time that project ended. In the weekly meetings, it was decided to discard the Functional Requirement 4 and 5 ("Should" have) priority, and implement them if the deadline would permit.

8.3 Final Considerations

As in any project there are difficulties, this was not an exception. From the writing part to the implementation, this was a very challenging work, and required a lot of dedication. With this internship, it is aimed to help the database architect to choose the right data modeling tools, among the analyzed alternatives in the scientific paper, which are also present in the Chapter 2. The developed product attends the initial expectations, and results in a data modeling tool which produces a multi-layer system with a one database on the Edge and a data warehouse on the Cloud layers, simultaneously. It also automatically generates a script, with all the data summarized and aggregated. This approach reduces error's probability and time spent, by avoiding the manual creation of the scripts. Despite this project is delivered after the initially set deadline, it meets the defined requirements and produces working scripts, and is ready to help data modelers in creating their own multi-layer databases.

This page is intentionally left blank.

Appendix A

Listing 8.1: MySQL Script of the first proof of concept

```
--
-- Edge database
--
--

CREATE DATABASE edge;
CREATE TABLE edge.campus (
        id              BIGINT,
        description VARCHAR(512),
        PRIMARY KEY(id)
);

CREATE TABLE edge.building (
        id              BIGINT,
        description VARCHAR(512),
        campus_id       BIGINT NOT NULL,
        PRIMARY KEY(id)
);

CREATE TABLE edge.room (
        id              BIGINT,
        description VARCHAR(512),
        building_id BIGINT NOT NULL,
        PRIMARY KEY(id)
);

CREATE TABLE edge.sensor (
        id              BIGINT,
        description     VARCHAR(512),
        communication VARCHAR(512),
        room_id  BIGINT NOT NULL,
        PRIMARY KEY(id)
);

CREATE TABLE edge.location_raw_data (
        id                      BIGINT,
        temperature     DOUBLE PRECISION,
        humidity                DOUBLE PRECISION,
        timestamp               TIMESTAMP,
        weather_raw_data_id BIGINT NOT NULL,
        sensor_id               BIGINT NOT NULL,
        PRIMARY KEY(id)
);

CREATE TABLE edge.weather_raw_data (
        id              BIGINT,
        temperature DOUBLE PRECISION,
        humidity        DOUBLE PRECISION,
        pressure        DOUBLE PRECISION,
        clouds   DOUBLE PRECISION,
        timestamp       TIMESTAMP,
        day_type_id BIGINT NOT NULL,
        PRIMARY KEY(id)
);

CREATE TABLE edge.day_type (
        id              BIGINT,
```

```
        day      VARCHAR(512),
        description VARCHAR(512),
        PRIMARY KEY(id)
);

ALTER TABLE building ADD CONSTRAINT building_fk1 FOREIGN KEY (campus_id) REFERENCES campus(id);
ALTER TABLE room ADD CONSTRAINT room_fk1 FOREIGN KEY (building_id) REFERENCES building(id);
ALTER TABLE sensor ADD CONSTRAINT sensor_fk1 FOREIGN KEY (room_id) REFERENCES room(id);
ALTER TABLE location_raw_data ADD CONSTRAINT location_raw_data_fk1 FOREIGN KEY (weather_raw_data_id) REFERENCES weather_raw_data(id);
ALTER TABLE location_raw_data ADD CONSTRAINT location_raw_data_fk2 FOREIGN KEY (sensor_id) REFERENCES sensor(id);
ALTER TABLE weather_raw_data ADD CONSTRAINT weather_raw_data_fk1 FOREIGN KEY (day_type_id) REFERENCES day_type(id);

--
-- Cloud data warehouse
--
--
CREATE DATABASE cloud;

CREATE TABLE cloud.summary_entity0 (
        id BIGINT(20) UNSIGNED NOT NULL AUTO_INCREMENT,
        location_raw_data_temperature_COUNT double      DEFAULT NULL,
        location_raw_data_temperature_AVERAGE double    DEFAULT NULL,
        location_raw_data_humidity_COUNT         double DEFAULT NULL,
        location_raw_data_humidity_AVERAGE       double DEFAULT NULL,
        weather_raw_data_temperature_COUNT       double DEFAULT NULL,
        weather_raw_data_temperature_AVERAGE     double DEFAULT NULL,
        weather_raw_data_humidity_COUNT  double DEFAULT NULL,
        weather_raw_data_humidity_AVERAGE        double DEFAULT NULL,
        weather_raw_data_pressure_COUNT  double DEFAULT NULL,
        weather_raw_data_pressure_AVERAGE        double DEFAULT NULL,
        weather_raw_data_clouds_COUNT    double DEFAULT NULL,
        weather_raw_data_clouds_AVERAGE  double DEFAULT NULL,
        campus_id        bigint UNSIGNED NOT NULL,
        campus_description       varchar(50),
        building_id      bigint UNSIGNED NOT NULL,
        building_description     varchar(50),
        room_id  bigint UNSIGNED NOT NULL,
        room_description         varchar(50),
        sensor_id        bigint UNSIGNED NOT NULL,
        sensor_description       varchar(50),
        sensor_communication     varchar(50),
        date    DATE,
        year    INT,
        month   TINYINT,
        day     TINYINT,
        hour    TINYINT,
        PRIMARY KEY (id)
);

--
--
-- EVENTS
--
SET GLOBAL event_scheduler =  "ON";
CREATE EVENT cloud.hour_sum_data
ON SCHEDULE EVERY 1 Hour
STARTS CONCAT(DATE(12/07/2021 INTERVAL 1 Hour), '␣00:00:01') -- now()
ON COMPLETION PRESERVE ENABLE
DO
        INSERT INTO cloud.summary_entity0(
```

```
        location_raw_data_temperature_COUNT,
        location_raw_data_temperature_AVG,
        location_raw_data_humidity_COUNT,
        location_raw_data_humidity_AVG,
        weather_raw_data_temperature_COUNT,
        weather_raw_data_temperature_AVG,
        weather_raw_data_humidity_COUNT,
        weather_raw_data_humidity_AVG,
        weather_raw_data_pressure_COUNT,
        weather_raw_data_pressure_AVG,
        weather_raw_data_clouds_COUNT,
        weather_raw_data_clouds_AVG,
        campus_id,
        campus_description,
        building_id,
        building_description,
        room_id,
        room_description,
        sensor_id,
        sensor_description,
        sensor_communication,
        date,
        year,
        month,
        day,
        hour)
SELECT
        COUNT(edge.location_raw_data.temperature),
        ROUND(AVG(edge.location_raw_data.temperature),2),
        COUNT(edge.location_raw_data.humidity),
        ROUND(AVG(edge.location_raw_data.humidity),2),
        (
                SELECT COUNT(edge.weather_raw_data.temperature)
                FROM edge.weather_raw_data
                WHERE edge.weather_raw_data_timestamp BETWEEN DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:00:00')
                AND DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:59:59')
        ),
        (
                SELECT ROUND(AVG(edge.weather_raw_data.temperature),2)
                FROM edge.weather_raw_data
                WHERE edge.weather_raw_data_timestamp BETWEEN DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:00:00')
                AND DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:59:59')
        ),
        (
                SELECT COUNT(edge.weather_raw_data.humidity)
                FROM edge.weather_raw_data
                WHERE edge.weather_raw_data_timestamp BETWEEN DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:00:00')
                AND DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:59:59')
        ),
        (
                SELECT ROUND(AVG(edge.weather_raw_data.humidity),2)
                FROM edge.weather_raw_data
                WHERE edge.weather_raw_data_timestamp BETWEEN DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:00:00')
                AND DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:59:59')
        ),
        (
                SELECT COUNT(edge.weather_raw_data.pressure)
                FROM edge.weather_raw_data
                WHERE edge.weather_raw_data_timestamp BETWEEN DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:00:00')
                AND DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:59:59')
```

```
),
(
        SELECT ROUND(AVG(edge.weather_raw_data.pressure),2)
        FROM edge.weather_raw_data
        WHERE edge.weather_raw_data_timestamp BETWEEN DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:00:00')
        AND DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:59:59')
),
(
        SELECT COUNT(edge.weather_raw_data.clouds)
        FROM edge.weather_raw_data
        WHERE edge.weather_raw_data_timestamp BETWEEN DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:00:00')
        AND DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:59:59')
),
(
        SELECT ROUND(AVG(edge.weather_raw_data.clouds),2)
        FROM edge.weather_raw_data
        WHERE edge.weather_raw_data_timestamp BETWEEN DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:00:00')
        AND DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:59:59')
),
edge.campus_id,
edge.campus_description,
edge.building_id,
edge.building_description,
edge.room_id,
edge.room_description,
edge.sensor_id,
edge.sensor_description,
edge.sensor_communication,
DATE(edge.location_raw_data_timestamp),
YEAR(edge.location_raw_data_timestamp),
MONTH(edge.location_raw_data_timestamp),
DAY(edge.location_raw_data_timestamp),
HOUR(edge.location_raw_data_timestamp)
FROM edge.location_raw_data
JOIN edge.sensor ON edge.sensor_id = edge.location_raw_data.sensor_id
JOIN edge.room ON edge.room_id = edge.sensor.room_id
JOIN edge.building ON edge.building_id = edge.room.building_id
JOIN edge.campus ON edge.campus_id = edge.building.campus_id
WHERE edge.location_raw_data_timestamp BETWEEN DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:00:00')
AND DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:59:59')
GROUP BY edge.location_raw_data_sensor_id;
```

This page is intentionally left blank.

Appendix B

Listing 8.2: MySQL Script of the second proof of concept

```
--
-- Edge database
--
--

CREATE DATABASE edge;
CREATE TABLE edge.country (
        id       bigint AUTO_INCREMENT,
        country varchar(50),
        PRIMARY KEY(id)
);

CREATE TABLE edge.district (
        id       bigint AUTO_INCREMENT,
        district         varchar(512),
        country_id bigint NOT NULL,
        PRIMARY KEY(id)
);

CREATE TABLE edge.city (
        id              bigint AUTO_INCREMENT,
        city     varchar(50),
        district_id bigint NOT NULL,
        PRIMARY KEY(id)
);

CREATE TABLE edge.shop (
        id       bigint AUTO_INCREMENT,
        name     varchar(100),
        address varchar(512),
        phone    int,
        city_id bigint NOT NULL,
        PRIMARY KEY(id)
);

CREATE TABLE edge.sale (
        id              bigint AUTO_INCREMENT,
        amount   double precision,
        timestampp       timestamp,
        shop_id  bigint NOT NULL,
        costumer_id bigint NOT NULL,
        PRIMARY KEY(id)
);

CREATE TABLE edge.costumer (
        id       bigint AUTO_INCREMENT,
        name     varchar(50),
        address varchar(512),
        phone    int,
        nif      int,
        age      int,
        PRIMARY KEY(id)
);

CREATE TABLE edge.product (
        id              bigint AUTO_INCREMENT,
        product  varchar(100),
        price    int,
```

```
        code     int,
        category_id bigint NOT NULL,
        PRIMARY KEY(id)
);

CREATE TABLE edge.category (
        id       bigint AUTO_INCREMENT,
        category varchar(100),
        PRIMARY KEY(id)
);

CREATE TABLE edge.product_sale (
        product_id bigint,
        sale_id  bigint,
        PRIMARY KEY(product_id,sale_id)
);

ALTER TABLE district ADD CONSTRAINT district_fk1 FOREIGN KEY (country_id) REFERENCES country(id);
ALTER TABLE city ADD CONSTRAINT city_fk1 FOREIGN KEY (district_id) REFERENCES district(id);
ALTER TABLE shop ADD CONSTRAINT shop_fk1 FOREIGN KEY (city_id) REFERENCES city(id);
ALTER TABLE sale ADD CONSTRAINT sale_fk1 FOREIGN KEY (shop_id) REFERENCES shop(id);
ALTER TABLE sale ADD CONSTRAINT sale_fk2 FOREIGN KEY (costumer_id) REFERENCES costumer(id);
ALTER TABLE product ADD CONSTRAINT product_fk1 FOREIGN KEY (category_id) REFERENCES category(id);
ALTER TABLE product_sale ADD CONSTRAINT product_sale_fk1 FOREIGN KEY (product_id) REFERENCES product(id);
ALTER TABLE product_sale ADD CONSTRAINT product_sale_fk2 FOREIGN KEY (sale_id) REFERENCES sale(id);

--
-- Cloud data warehouse
--
--
CREATE DATABASE cloud;

CREATE TABLE cloud.summary_entity0 (
        id BIGINT(20) UNSIGNED NOT NULL AUTO_INCREMENT,
        sale_amount_AVERAGE      double precision       DEFAULT NULL,
        shop_id  bigint UNSIGNED NOT NULL,
        shop_name        varchar(50),
        shop_address     varchar(50),
        shop_phone       int    UNSIGNED NOT NULL,
        costumer_id      bigint UNSIGNED NOT NULL,
        costumer_name    varchar(50),
        costumer_address         varchar(50),
        costumer_phone   int    UNSIGNED NOT NULL,
        costumer_nif     int    UNSIGNED NOT NULL,
        costumer_age     int    UNSIGNED NOT NULL,
        product_id       bigint UNSIGNED NOT NULL,
        product_product  varchar(50),
        product_price    int    UNSIGNED NOT NULL,
        product_code     int    UNSIGNED NOT NULL,
        date    DATE,
        year    INT,
        month   TINYINT,
        day     TINYINT,
        hour    TINYINT,
        PRIMARY KEY (id)
);

--
--
-- EVENTS
```

```
--
SET GLOBAL event_scheduler =  "ON";
CREATE EVENT cloud.hour_sum_data
ON SCHEDULE EVERY 1 Hour
STARTS CONCAT(DATE(12/08/12 INTERVAL 1 Hour), '␣00:00:01') -- now()
ON COMPLETION PRESERVE ENABLE
DO
        INSERT INTO cloud.summary_entity0(
                sale_amount_AVG,
                shop_id,
                shop_name,
                shop_address,
                shop_phone,
                costumer_id,
                costumer_name,
                costumer_address,
                costumer_phone,
                costumer_nif,
                costumer_age,
                product_id,
                product_product,
                product_price,
                product_code,
                date,
                year,
                month,
                day,
                hour)
SELECT
                ROUND(AVG(edge.sale.amount),2),
                edge.shop.id,
                edge.shop.name,
                edge.shop.address,
                edge.shop.phone,
                edge.costumer.id,
                edge.costumer.name,
                edge.costumer.address,
                edge.costumer.phone,
                edge.costumer.nif,
                edge.costumer.age,
                edge.product.id,
                edge.product.product,
                edge.product.price,
                edge.product.code,
                DATE(edge.sale.timestamp),
                YEAR(edge.sale.timestamp),
                MONTH(edge.sale.timestamp),
                DAY(edge.sale.timestamp),
                HOUR(edge.sale.timestamp)
                FROM edge.sale
                JOIN edge.shop ON edge.shop.id = edge.sale.shop_id
                JOIN edge.city ON edge.city.id = edge.shop.city_id
                JOIN edge.district ON edge.district.id = edge.city.district_id
                JOIN edge.country ON edge.country.id = edge.district.country_id
                WHERE edge.sale.timestamp BETWEEN DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:00:00')
                AND DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 1 HOUR), '%Y-%m-%d␣%H:59:59')
                GROUP BY edge.sale.shop_id;
```

References

[1] M. Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[2] Onda. `http://onda.dei.uc.pt/v3`. Accessed: 17-10-2020.

[3] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. *IEEE Internet Computing*, 6(5):50–58, 2002.

[4] M. Aazam, S. Zeadally, and K. A. Harras. Fog computing architecture, evaluation, and future research directions. *IEEE Communications Magazine*, 56(5):46–52, 2018.

[5] Robert Elsenpeter Toby Velte, Anthony Velte. *Cloud Computing, A Practical Approach*. McGraw-Hill Osborne Media, 1 edition, 2009.

[6] K. Cao, Y. Liu, G. Meng, and Q. Sun. An overview on edge computing research. *IEEE Access*, 8:85714–85728, 2020.

[7] Chao Li, Yushu Xue, Jing Wang, Weigong Zhang, and Tao Li. Edge-oriented computing paradigms: A survey on architecture design and system management. *ACM Comput. Surv.*, 51(2), April 2018.

[8] Weisong Shi Jie Cao, Quan Zhang. *Edge Computing: A Primer*. SpringerBriefs in Computer Science. Springer International Publishing, 1st ed. edition, 2018.

[9] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. New York, NY, USA, 2012. Association for Computing Machinery.

[10] Long Mai, Nhu-Ngoc Dao, and Minho Park. Real-time task assignment approach leveraging reinforcement learning with evolution strategies for long-term latency minimization in fog computing. *Sensors*, 18:2830, 08 2018.

[11] Zaigham Mahmood. *Fog Computing*. Springer International Publishing, 1st ed. edition, 2018.

[12] Sudeep Tanwar. *Fog Computing for Healthcare 4.0 Environments: Technical, Societal, and Future Implications*. Signals and Communication Technology. Springer International Publishing;Springer, 1st ed. edition, 2021.

[13] L. Li, T. Q. S. Quek, J. Ren, H. H. Yang, Z. Chen, and Y. Zhang. An incentive-aware job offloading control framework for multi-access edge computing. *IEEE Transactions on Mobile Computing*, 20(1):63–75, 2021.

[14] K. Dolui and S. K. Datta. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In *2017 Global Internet of Things Summit (GIoTS)*, pages 1–6, 2017.

[15] H. Li, G. Shou, Y. Hu, and Z. Guo. Mobile edge computing: Progress and challenges. In *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 83–84, 2016.

[16] Y. Li and W. Wang. The unheralded power of cloudlet computing in the vicinity of mobile devices. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 4994–4999, 2013.

[17] Bodo Hüsemann, Jens Lechtenbörger, and Gottfried Vossen. *Conceptual data warehouse design*. Citeseer, 2000.

[18] Shamkant B Navathe. Evolution of data modeling for databases. *Communications of the ACM*, 35(9):112–123, 1992.

[19] IBM definition of Data Modeling. `https://www.ibm.com/cloud/learn/data-modeling`. Accessed: 27-04-2021.

[20] Graeme Simsion and Graham Witt. *Data modeling essentials*. Elsevier, 2004.

[21] Wikipedia definition of Data Modeling. `https://www.ibm.com/cloud/learn/data-modeling`. Accessed: 27-04-2021.

[22] Ralph Kimball and Margy Ross. *The DataWarehouse Toolkit*. John Wiley & Sons Inc, 2004.

[23] Abraham Silberschatz, Henry F Korth, and S Sudarshan. *Database system concepts*. McGraw-Hill Education, New York, New York, USA, 7th edition, 1991.

[24] N.P. Singh and C.S. Gupta. *Relational Database Management Systems*. Abhishek Publications, 2014.

[25] Salman Niazi, Mahmoud Ismail, Seif Haridi, and Jim Dowling. Hopsfs: Scaling hierarchical file system metadata using newsql databases. In Sherif Sakr and Albert Y. Zomaya, editors, *Encyclopedia of Big Data Technologies*. Springer, 2019.

[26] Steve C Wotring and John R Ripley. System and method for transforming a relational database to a hierarchical database, December 16 2003. US Patent 6,665,677.

[27] D Kroenke. Database processing: fundamentals, and implementation, 2000.

[28] James Black, Dimitrios Makris, and Tim Ellis. Hierarchical database for a multi-camera surveillance system. *Pattern Anal. Appl.*, 7(4):430–446, 2004.

[29] EF Codd. A relational model of data for large shared data banks. communicattions of the acm, 13 (6), 377–387, 1970.

[30] Nelson Eng Adrienne Watt. *Database Design, 2nd Edition*. The BCcampus Open Textbook Project, 2014.

[31] Michael Hammer and Dennis Mc Leod. Database description with sdm: A semantic database model. 6(3):351–386, September 1981.

[32] Paul Beynon-Davies. *Database systems*. Springer, 2004.

[33] Richard Hull and Roger King. Semantic database modeling: Survey, applications, and research issues. *ACM Comput. Surv.*, 19(3):201–260, September 1987.

[34] Michael Grossniklaus. *An object-oriented version model for context-aware data management*. Zürich : ETH Zürich, 2007. Zugl.: Diss., Univ., Zürich, 2007.

[35] J. Bhogal and P. Moore. Towards object-oriented context modeling: Object-oriented relational database data storage. In *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pages 542–547, 2014.

[36] A B M Moniruzzaman and Syed Akhter Hossain. Nosql database: New era of databases for big data analytics - classification, characteristics and comparison, 2013.

[37] S. Bouamama. Migration from a Relational Database to NoSQL. *International Journal of Knowledge-Based Organizations*, 8(3):63–80, 2018.

[38] J. Pokorný. Integration of Relational and NoSQL Databases. In *Asian Conference on Intelligent Information and Database Systems - Intelligent Information and Database Systems*, volume 10752, pages 35–45, 2018.

[39] Denio; Mello Ronaldo S. Schreiner, Geomar A.; Duarte. When Relational-Based Applications Go to NoSQL Databases: A Survey. *Information*, 10(7):22, 2019.

[40] Vincent Reniers, Dimitri Van Landuyt, Ansar Rafique, and Wouter Joosen. Object to nosql database mappers (ONDM): A systematic survey and comparison of frameworks. *Inf. Syst.*, 85:1–20, 2019.

[41] D Batra and G M Marakas. Conceptual data modelling in theory and practice. *European Journal of Information Systems*, 4(3):185–193, 1995.

[42] Daniel L. Moody and Graeme G. Shanks. What makes a good data model? Evaluating the quality of entity relationship models. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 881 LNCS:94–111, 1994.

[43] Drazen Brdjanin, Goran Banjac, Danijela Banjac, and Slavko Maric. An experiment in model-driven conceptual database design. *Software and Systems Modeling*, 18(3):1859–1883, 2019.

[44] Bernhard Thalheim. *Entity-Relationship Modelling: Foundations of Database Technology*. Springer-Verlag, Inc, New York, 2000.

[45] Peter Chen. The Entity-Relationship Model—toward a Unified View of Data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36, 1976.

[46] Welcome To UML, https://www.uml.org/, 2019. Accessed: 2021-06-28.

[47] Sabah Al-Fedaghi and Haya Alahmad. Orientation in Conceptual Modeling Frameworks. pages 1298–1303, 2018.

[48] 79 Data Modeling Tools Compared. `https://www.databasestar.com/data-modeling-tools/`. Accessed: 16-10-2020.

[49] 20 Best Data Modeling Tools. `https://www.guru99.com/data-modeling-tools-design-database.html`. Accessed: 16-10-2020.

[50] Agile Methodologies. `https://www.xpand-it.com/blog/top-5-agile-methodologies/`. Accessed: 09-06-2021.

[51] 12 principles of Agile methodology. `https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/`. Accessed: 03-06-2021.

[52] SCRUM Sprint Duration. https://www.scrum.org/resources/blog/weekly-scrum-interview-question-what-duration-sprint. Accessed: 09-06-2021.

[53] Rist Management. https://en.wikipedia.org/wiki/Risk_management. Accessed: 04-05-2021.

[54] Ofer Zwikael and Jack Meredith. Evaluating the success of a project and the performance of its leaders. *IEEE Transactions on Engineering Management*, PP:1–13, 07 2019.

[55] MOSCOW PRIORITISATION. https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation. Accessed: 14-05-2021.

[56] Non-Functional Requirements. Accessed: 2021-07-28.

[57] JointJS Library. https://resources.jointjs.com/docs/jointjs/v3.3/joint.html. Accessed: 08-06-2021.

[58] C4 Architecture Model. https://c4model.com/. Accessed: 08-06-2021.

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Comparative Analysis of Data Modeling Design Tools

GONÇALO CARVALHO[1] [A], SERGII MYKOLYSHYN[1] [B], BRUNO CABRAL[1] [C], JORGE BERNARDINO[2,1] [D] (IEEE MEMBER), VASCO PEREIRA[1] [E],

[1]Univ Coimbra, CISUC, Department of Informatics Engineering, Portugal (e-mail: gcarvalho@dei.uc.pt, sergii.myk@gmail.com, bcabral@dei.uc.pt, vasco@dei.uc.pt)

[2]Polytechnic of Coimbra, ISEC, Coimbra, Portugal (e-mail: jorge@isec.pt)

Corresponding authors: Gonçalo Carvalho (e-mail: gcarvalho@dei.uc.pt)

A:0000-0001-7095-5003, B:0000-0003-0851-8165, C:0000-0001-9699-1133, D:0000-0001-9660-2011, E:0000-0002-4225-9075

ABSTRACT Conceptual modeling describes the physical or social aspects of the world abstractly, encompassing the interpretation of data production, gathering, visualization, and analysis. The quality of the data analysis system will condition the excellence of any decision-making process. Thus, accurately specifying the database model is essential. The primary goal of our work is to compare tools that can create this physical model. We recognize several types of data models, but this work will only encompass the relational data model. We will evaluate free and commercial data modeling tools. But it is challenging to decide how to compare them and which elements are crucial. We propose a new approach for software tools' evaluation based on the Business Readiness Rating (BRR) model and the OSSpal evaluation methodology. In this work, we show that this new methodology can be tailored to the needs of each individual developer or team, thus providing fitter and meaningful results. Also, by applying this hybrid approach to the evaluation of database modelling tools, we show it can robustly handle the bias from lesser relevant evaluation categories.

INDEX TERMS Data Modeling, Design Tools, Databases, Database Modeling Tools

I. INTRODUCTION

A *data model* is a set of concepts that can describe the data structure and operations on a database [1]. These data structures include objects, relations between these objects, and rules that define how data is organized. Determining the business needs will lead to the data model. The business stakeholders' feedback is crucial to define rules and requirements to be incorporated into the design of a new system or adapted in an iteration of an existing one [2].

Data modeling creates a visual representation of either a whole information system or parts of it to reveal connections between data points and structures. This is the first step in database design, and Simsion and Witt [3] defined it as "*a design activity which classifies information in an organized way and defines their relations.*" Therefore, the process of data modeling involves professional data modelers working closely with business stakeholders, as well as potential users of the information system [4].

The Entity Relationship (ER) model is one of the fundamental conceptual data models, which is usually associated with relational databases. This model is the focus of this work because it is the model most often adopted at this stage of conceptual design. An Entity Relationship Diagram (ERD) is a drawing that communicate the relationships between tables, also known as entities [5]. An entity is a "thing" or "object" in the real world that is distinguishable from other objects. Relationships have cardinalities, attributes, and constraints. The cardinality of a relationship indicates the number of occurrences between two entities [6].

Any database architect needs to work with a tool that allows an easy data model design. Such a choice will have a direct impact on the project quality. The design tool must be suitable to represent a database, also be easy to use, and support a different number of database engines. In addition, the tool should allow defining constraints such as Primary Key (PK), Not Null (NN), Auto Increment (AI), and produce a Structured Query Language (SQL) script from the representation of data objects created by the user.

To the best of our knowledge, this is the first work that evaluates database modeling tools using a formal evaluation

methodology. The major contributions of this work are the following:

- Provide a background on conceptual modeling and data modeling tools;
- Propose a new methodology for evaluating software modeling tools based on the Business Readiness Rating (BRR) model and the OSSpal methodology;
- Demonstrate an evaluation of free and commercial data modeling tools. This evaluation reproduces an accurate result reflecting the usefulness of the tools and their level of productivity for users.

We organized the rest of this paper as follows. In Section II, we review similar works. In Section III, we provide the background on conceptual models. In Section IV, we describe and analyze each tool. In Section V, we explain the evaluation methodology used in the tools' assessment. In Section VI, we evaluate the selected tools. Finally, in Section VII, we describe the main conclusions.

II. RELATED WORK

In this section, we will review other approaches that evaluated software quality through different methodologies.

In 2005, SpikeSource, the Center for Open Source Investigation at Carnegie Mellon West, and Intel Corporation created the BRR model [7], which lets IT managers promptly deliver informed and educated decisions about open-source software. They developed BRR to be a complete, simple, adaptable, and consistent model to help choose the right software. The authors evaluated open-source software according to 12 categories: functionality, usability, quality, security, performance, scalability, architecture, support, documentation, adoption, community, and professionalism. They divided the BRR into four phases: i) *A quick assessment*, to identify a list of components, measure each component, and remove any component that does not fit the user requirements. ii) *Target usage assessment*, define the 12 category weights according to importance from 1 to 12, choose the top seven (or less) and assign a percentage of importance totaling 100%. Set the metric weights within each category according to their importance, also totaling 100%. iii) *Data collection and processing*, collect data for each metric in each category, and calculate the applied weighting for each metric. iv) *Data translation*, calculate the final BRR score.

In 2017, Wasserman et al. [8] proposed an extension to the BRR, which originated the OSSpal open-source software assessment methodology. Motivated by solving the shortcomings of the original approach, such as i) some bias in the BRR score, according to the evaluator knowledge of the project, besides existing documentation and commercial support; ii) the lack of details provided by a single numeric score; iii) the reduced amount of adequate software to endure this evaluation; and iv) the prime consideration of opinions of others, including both peers and experts. Thus, the authors introduced some changes, among other minor improvements, i) because the BRR only used the top seven ranked categories, which may leave out of the analysis important categories to other evaluators, Wasserman et al. condensed the Categories from 12 into seven, which the authors state to be the most important in open-source software: Functionality, Operational Software Characteristics, Support and Service, Documentation, Software Technology Attributes, Community and Adoption, and Development Process.; ii) removed the final score calculation formula; iii) created a list of adequate software for evaluation and grouped them into categories based on the software taxonomy produced annually by the International Data Corporation (IDC) [9]; and iv) developed a website for users to use and rank the software tools to surpass the impossibility to assess which tool is better amongst two or three with the same feature score.

We resorted to several databases, DBLP - computer science bibliography, Google Scholar, and IEEE Xplore, to find other works regarding the use of these evaluation schemas.

From 2017 to 2019 several papers used the OSSpal methodology in different research areas, such as Business Intelligence Tools [10], [11], Data Mining Tools [12], E-commerce Tools [13], Project Management Tools [14]–[17], and NoSQL databases [18]. These works used the same implementation and analyzed three or four tools. The results when evaluating the same tool were different, for example, OpenProject 4.5 and 3.45 in [15] and [14] respectively, and ProjectLibre 3.82 and 3.6 in [15] and [16] respectively. This highlight's the subjective approach of the evaluation, because the Categories' weights and the number of analyzed characteristics encompassing the Functionality category were different while evaluating the same tools. Also, all these works had a final score, so they were an assessment through the BRR model assessment rather than an evaluation by the OSSpal methodology.

Nevertheless, the penalty for high scores in less critical measures is one of the OSSpal methodology shortcomings. Ultimately, it will depend on the person evaluating the software products, which can lead to some bias scoring.

In our approach, we will use the BRR model, with the OSSpal Categories, which are more adapted to the tools under evaluation, and we introduce some changes by providing a broader range of values for the evaluation of the features. We also evaluate a significantly higher amount of tools (17), both free and commercial, in a different research area, and through our Department survey, we aimed to remove the subjectiveness of our evaluation.

III. BACKGROUND IN CONCEPTUAL MODELING

In this section, we address the Conceptual Model (CM) topic, identifying key features and languages.

CM describes the physical or social aspects of the world abstractly. The result of a proper and rigorous CM design is a functionally richer, less error-prone, adequately attuned, able to adapt to varying user requirements, and less expensive system [19]. Thus, designing the CM at the beginning of the development cycle should be mandatory. It will be easier to follow and adapt to user requirements and explore existing relationships between the concepts.

We adopt data models to manage and analyze data representing any information system. The data model is an essential element of the system development or database design processes. Although the data modeling phase embodies only a smaller dimension of the development effort, its influence on the eventual result is reasonably broader than any other phase. Moody et al. [20] mentioned the vast amount of alternative designs to address Conceptual Data Modeling (CDM). Several alternative models could provide accurate solutions, but may have quite distinct implications for database and system design. The process of data modeling is not simple, meaning it usually demands multiple iterations [21].

Thalheim [22] point different CM notations used to describe requirement specifications, such as the ER diagram [23], the Unified Modeling Language (UML) [24], which are the most regularly employed, Business Process Modeling and Notation, and Model-driven Engineering. Object-Oriented (OO) models are essentially expressive and more fitted to describe static and dynamic features of complex applications. The OO modeling field relates objects and attributes, whereas the real-world realm deals with things and properties [25].

The ER data model has existed for over 35 years. An ER diagram is appropriate for data modeling because it is abstract and is easy to discuss and explain. It is easy to translate ER models to relations. The base of this type of modeling are entities, which hold information and relationships, defined as the associations between entities [26].

The primary advantages of CM for general systems and specifically for Database Management Systems (DBMS) are:

- Provide a high-level perception of how the system will work;
- Join different mental models into a single CM design;
- Ensure that the data representation is accurate - missing fields in the database cause unreliable results;
- Get a clear understanding of the data that developers can manage when building the actual database;
- Identify any redundant or missing data;
- Make maintenance and upgrades faster and more affordable.

Next, we perform a qualitative analysis of data modeling tools.

IV. MODELING TOOLS

In this section, we will analyze different data modeling tools. We will include the physical model and the script generation (forward engineering) in the analysis's scope. However, we will not cover the physical deployment, access, and configuration of the database.

To create an ER model, it is necessary to specify tables (entities) containing fields (attributes) that will be the columns and relations between tables. In this model, the relationships amongst tables have a cardinality setting that illustrates the following options: one-to-one, one-to-many, zero-to-one, zero-to-many, and many-to-many. Despite this graphical representation, the physical model produces a better comprehension of the relationships. Because it translates these into tables, and it defines the cardinality through the PK and Foreign Key (FK) constraints, as well as or NN. Converting the design of the CM into a physical model offers a straightforward interpretation of the model.

After these steps, and with a proper definition of the business logic, the next step is *Forward Engineering*, which is the auto-generation of a SQL script from the created representation. This last step is crucial for any database engineer to minimize errors and time spent creating a database.

From the list of *79 Data Modeling Tools Compared* [27] and *20 Best Data Modeling Tools* [28], we selected those that allow the user to perform *Forward Engineering*, and only considered tools that continued receiving updates after 2018. We sorted them into four major product types: Online free tools, Online commercial tools, Desktop free tools, and Desktop commercial tools. The four product type are able to broadly represent all the products available today, is well-aligned with the different users' and enterprises' needs.

TABLE 1: Modeling Tools

Modeling Tools	
Product Type	**Name**
Online free tools (I)	Dbdesigner.id https://dbdesigner.id
	Onda http://onda.dei.uc.pt/v3
	WWW SQL Designer https://github.com/ondras/wwwsqldesigner
Online commercial tools (II)	Dbdesigner.net https://app.dbdesigner.net
	dbDiffo https://dbdiffo.com
	GenMyModel https://www.genmymodel.com
	Lucidchart https://lucid.app
	sqlDBM https://sqldbm.com
Desktop free tools (III)	MySQL Workbench - Community Version https://github.com/mysql/mysql-workbench
	pgModeler https://github.com/pgmodeler/pgmodeler
	Umbrello UML https://github.com/KDE/umbrello
Desktop commercial tools (IV)	dbSchema https://dbschema.com
	dbWrench http://www.dbwrench.com
	Erwin Data Modeler https://erwin.com/products/erwin-data-modeler
	Navicat https://www.navicat.com
	Oracle SQL Developer Data Modeler https://www.oracle.com/database/technologies/appdev/sqldeveloper-landing.html
	PowerDesigner https://www.sap.com/products/powerdesigner-data-modeling-tools.html

Another relevant characteristic is that some tools have a visual representation of the conceptual data model, but others only show the logical data model. However, others have both types of visualization. On the one hand, the logical data model only describes the data and its relations. On the other, the physical data model displays' table structures, including column name, data type, and the constraints, such as PK, FK, and Unique (UQ), represent the relationships between tables.

In the following subsections, we perform a qualitative analysis of the selected tools. We provide a brief description of the key features, such as the release year; real-time collaboration options; generation of the physical model; the presence of *Reverse Engineering* (auto-generation of ER from SQL) and *Forward Engineering*; supported database engines and data types; different constraints; the presence of CM; finally, the pros and cons are analyzed.

A. ONLINE FREE TOOLS

Online free tools work on every platform, receive constant updates from the community, and do not require installing the software. Also, every developer can contribute to any of these projects.

Dbdesigner.id

Dbdesigner.id is a database design tool for web developers and beginners, which started in 2019 under MIT License as a hobby project and is continuously under construction. The authors' goal was to make database management accessible to everyone. They developed it in Javascript, HTML, and CSS. To use Dbdesigner, they require creating a new user account. With this tool, it is possible to share a link with a project contributor to work simultaneously. It does not allow for a CM design. The tool does not provide reverse engineering, and only works with MySQL. For each column, the user can choose among different types (tinyint, smallint, bigint, int, bigint, float, double, datetime, date, timestamp, char, varchar, binary, blob, text, json). It enables an option to specify constraints such as PK, FK, NN, UQ, and AI. It is also possible to set a default value for each entry in a table. The user can select the referencing table and column name to create a relationship.

Pros: Link sharing for collaboration.

Cons: Needs registration and only uses MySQL.

Onda

Onda is a database modeling tool developed by the Department of Informatics Engineering (DEI) community, with the first version released in 2014, developed with Javascript, HTML, and CSS. Also, it has fast loading times. The tool does not require any configuration, but there is no real-time collaboration possibility. With Onda, it is possible to draw conceptual databases and visualize the physical model. There is no reverse engineering option, but it is possible to perform forward engineering for the most famous database engines like PostgreSQL, MySQL, Oracle, MariaDB, and SQLite. Each column can have different types, such as boolean, integer, float, date, character, varchar, text, or BLOB. It is possible to add constraints for each column, such as PK, NN, AI, Check Constraint (CH), and UQ. The FK are automatically added in the physical model through the designed relationships.

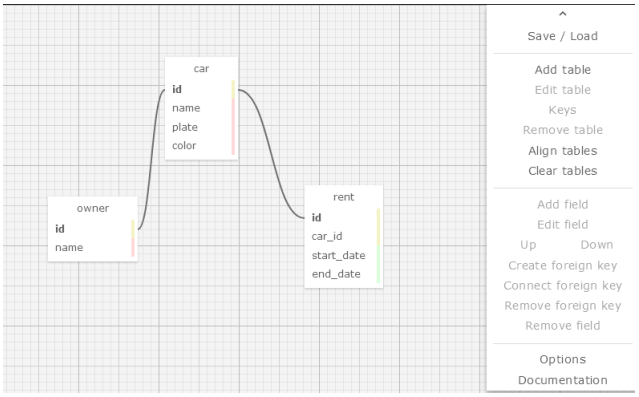


FIGURE 1: Example of a database model at WWW SQL Designer

Pros: Zoom in/out, possibility to export the CM into different database engines.

Cons: Some bugs on the physical model and script generation.

WWW SQL Designer

Released in 2005, WWW SQL Designer allows users to create and export data models to SQL scripts. The interface has a mini-map for fast navigation (Fig. 1). It does not require configuration, does not have real-time collaboration or representation of the CM. With this tool, it is possible to perform forward engineering for the MySQL database engine, but it is impossible to perform reverse engineering. It supports many database constraints, such as PK, FK, UQ, NN, and AI. There are different data types such as int, decimal, char, binary, BLOB, date, and time.

Pros: Many data types, and drag-and-drop features.

Cons: No real-time collaboration, no representation of the CM, and only exports MySQL scripts.

B. ONLINE COMMERCIAL TOOLS

This subsection will introduce the online commercial tools. Because they work online, it is possible to use them on every platform. It is necessary to get a subscription or activation key. Otherwise, it is limited, where the features are only available for a short period or with narrow options.

Dbdesigner.net

Since 2006, Dbdesigner.net is a database schema designer for data modeling (Fig. 2). The table representation is clean and has different colors for each table entry. Dbdesigner does not require any configuration to use it. It is possible to share the database design with other users to work simultaneously on it. This tool does not provide the visualization of the

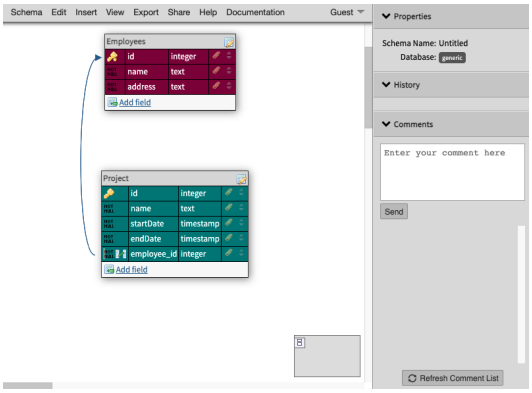


FIGURE 2: Example of a database model at Dbdesigner.net

databases' CM model. Dbdesigner also offers a reverse engineering option. Besides it, forward engineering enables code export for engines like MySQL, Microsoft SQL Server, PostgreSQL, Oracle, and SQLite. This tool has distinct features that make it unique, such as a mini-map for fast navigation, keyboard shortcuts, instant save with history, copy and paste, undo and redo, and notes and comments. It supports data types such as binary, boolean, date, decimal, float, integer, and varchar. The constraints are: PK, FK, NN, AI, and UQ.

Pros: Developers can work simultaneously, has several database design templates, features as instant save, undo and redo, easy-to-use interface. Also, it enables reverse engineering.

Cons: No representation of the CM.

dbDiffo

Released in 2014, Dbdiffo is a database modeling tool similar to Onda. Before using the tool, it is necessary to specify the model name and choose the database engine for script generation. With Dbdiffo, it is impossible to do a real-time collaboration, and it does not provide the CM design. This tool does not allow reverse engineering, but regarding forward engineering, it is possible to export scripts for database engines like IBM DB2, Microsoft SQL Server, MySQL, Oracle, and PostgreSQL. The tables' columns may be bigint, binary, bit, blob, char, date, datetime, decimal, double, float, integer, longblob, longtext, mediumint, mediumtext, numeric, smallint, text, time, timestamp, varchar, or year. Regarding the constraints, it is possible to define PK, FK, NN, and AI.

Pros: History toolbar, and unlimited undo.

Cons: No real-time collaboration and no reverse engineering.

GenMyModel

Released in 2012, GenMyModel speeds up the design of software architecture and business processes. It is easy to add new entities and create relations between them because of its interface and the provided documentation. It is necessary to log into the application via GitHub or Google account, create a new diagram, and select a Relational Database. It is possible to create a database from scratch or select an existing project from the cloud. GenMyModel has real-time collaboration with a chat and also allows the creation of the CM, since it has its base in the UML. The supported database's engines are Apache Hive, Oracle, MySQL, and PostgreSQL. This tool supports different data types, such as boolean, binary, character, date, float, integer, time, or varchar. The tool can auto-generate PDF and MS Word documents based on custom templates and export diagrams to GitHub. The developers can use the open API and integration functions to build integrations for testing or code proofing. Regarding the constraints, it is possible to define PK, FK, NN, CH, and UQ.

Pros: Real-time team collaboration features, it auto-generates the documentation of the data models.

Cons: If using the free version, it only provides basic features and limits the design to 20 objects, including not only the tables, but also each column and relationship.

LucidChart

Released in 2008, Lucidchart is a powerful tool, it has a free version, and also offers a trial to explore its full potential. It is unnecessary to do any configuration before using the tool. Lucidchart has team collaboration, is based in the UML, and allows CM design. It is not possible to perform reverse engineering. Forward engineering is possible to several database engines such as MySQL, PostgreSQL, Microsoft SQL Server, and Oracle. Each column may have any data type introduced by the user, later converted into the specific database engine data types. As in other tools, it is possible to choose from the list of existing types. The constraints are PK, FK, NN, and AI.

Pros: Real-time collaboration feature, and different templates.

Cons: The user must know the specificities of the database engine data types because it is possible to insert any string in the data type field.

sqlDBM

Released in 2017, sqlDBM has a free version that comes with limited features, and it is possible to try the full version for 14 days. This design tool only requires configuring the database type, and it does not offer CM design. There is also a team collaboration tool. It is impossible to perform reverse engineering, but forward engineering allows to export the script to Microsof SQL Server, MySQL, Snowflake, Amazon Redshift, PostgreSQL, and Azure Synapse Analytics. Each column attribute can be of the type bigint, bigint unsigned, binary, bit, blob, char, date, datetime, decimal, double, double unsigned, float, integer, numeric, text, time, timestamp, varchar, or year. Also, it is possible to specify the constraints PK, FK, NN, AI, and UQ.

Pros: This tool has forward engineering and team collaboration possibilities. Little tutorial, in the beginning, explain-

ing functionalities, no need to sign-up. "Undo" and "redo" options also present.

Cons: No CM design.

C. DESKTOP FREE TOOLS

In this subsection, we analyze four desktop free tools. These require installation, and as make part of the open-source community are free to use, and every developer can contribute to their development.

MySQL Workbench - Community Version

Created in 2002, MySQL Workbench is an application used to manage and design a database schema. The open-source version has a GPL license with a GitHub repository. The significant differences between Community and Enterprise versions are the non-presence of Schema & Model Validation, automated documentation of databases, and non-existence of firewall specification rules. Before working with the tool, it is necessary to set up the connection to the existing database. Otherwise, it is impossible to export the MySQL script resulting from the establishment of tables and relations. MySQL Workbench does not have a real-time collaboration feature, but has the capacity to design the CM. There is a possibility to reverse and forward engineer for MySQL Server databases. Different categories organize the entity types like numeric, characters, time, geometry, and others, such as bits or boolean. The number of constraints is also considerable, it enables an option to insert PK, FK, NN, UQ, AI, Binary, and Unsigned (U).

Pros: Unlimited "Undo" and "Redo" options.

Cons: Only available for Windows machines and no real-time collaboration. Also, necessity to set up the connection to the existing database.

pgModeler

Created in 2006, pgModeler is a database modeling tool designed for PostgreSQL databases. Despite the need to pay for the compiled version, it is possible to get the open-source version and compile it manually. The tool has different colors to help visualization (Fig. 3). If there are missing functionalities, it is possible to create new extensions and contribute to open-source code development. It is unnecessary to make any configuration before using the application. The tool does not have an option for real-time collaboration, and it has the feature to design the CM. It also provides reverse and forward engineering for PostgreSQL databases. pgModeler has a database management module where it is possible to run SQL commands, explore the objects, and handle data. It has distinct entity types like bigint, bit, bool, char, date, decimal, float, int, json, money, text, time, and varchar. Also, different constraints such as PK, FK, UQ, Exclude (E), CH, AI and NN are present.

Pros: It is possible to collaborate on the tool.

Cons: Only supports PostgreSQL database engine.

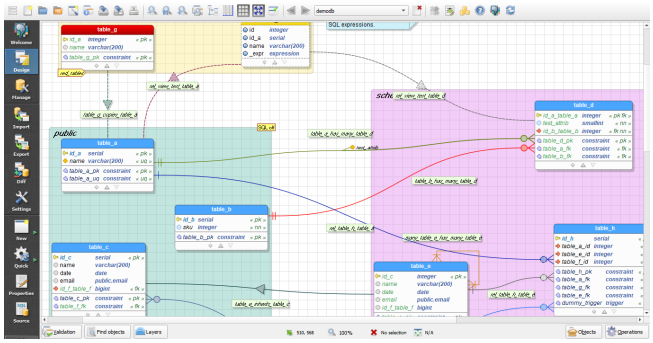


FIGURE 3: Example of a database model at pgModeler

Umbrello UML

Released in 2006, Umbrello UML is a UML diagram program developed by an international free software community. This tool does not require any configuration. It is not possible to perform real-time collaboration. It has a feature to design conceptual data models. It is impossible to reverse engineering, but the forward engineering option allows generating SQL scripts for MySQL and PostgreSQL. Each table column can have different types, such as bool, char, double, float, int, or string. And there are different constraints: PK, FK, AI, UQ, and NN.

Pros: Feature for CM.

Cons: Limited number of databases to export scripts, and no real-time collaboration.

D. DESKTOP COMMERCIAL TOOLS

In this subsection, we analyze desktop commercial tools. We chose these according to Google Trends, since the tools number in this product type is high. For us to consider a tool, it had to be googled at least ten times per week, from 2004 until 2021, worldwide, and we found seven desktop proprietary tools matching the criteria. To use either tool, first, it is necessary to install it on the machine and then either try a free trial or buy the full version.

dbSchema

Released in 2016, dbSchema does not require configurations. dbSchema cannot provide a conceptual data model, neither exists real-time collaboration. It has both reverse and forward engineering that works with all relational databases, including SqlServer, SAP Adaptive Server, Oracle, MySql, Ingres, Informix, Db2, Derby, Firebird, Frontbase, Cache, Pervasive, PostgreSQL, and Sqlite. Like all the previous tools, it allows adding different data types to each column, like blob, boolean, char, date, double, float, int, json, real, text, varchar. Farther, the constraints are PK, FK, NN, and AI.

Pros: The tool has both reverse and forward engineering for many database motors.

Cons: No real-time collaboration.

dbWrench

dbWrench is a multi-platform database design and synchronization software, released in 2004, and does not require any configuration. dbWrench does not support the design of the CM and has no real-time collaboration. The reverse engineering does not exist in this tool, and the forward engineering tool generates SQL scripts for Microsoft SQL Server, Oracle, PostgreSQL, and MySQL. It is possible to connect to a database and run the code for table creation. It offers some default column templates that save time creating tables. When adding a column, there is a possibility to specify data types like binary, blob, bit, boolean, char, date, decimal, double, float, json, number, real, time, varchar. The constraints are PK, , NN, and AI.

Pros: Reverse and forward engineering for several databases engines.

Cons: No real-time collaboration.

Erwin Data Modeler

Founded in 1988, the platform allows creation and maintenance of data warehouses and databases. This tool provides several tutorials to help understand how to do data modeling. Financial services, healthcare, critical infrastructure, and technology companies use Erwin Data Modeler. The tool does not require any configuration, and does not have an option of real-time collaboration. Erwin provides a possibility to design the conceptual data model. It only has forward engineering that supports database engines such as Oracle, MySQL, IBM DB2, SAP IQ, and Teradata. The different data types for columns in this tool are char, integer, date, boolean, real, and float. The tool has constraints such as PK, FK, NN, AI, and UQ.

Pros: Supports several database engines for forward engineering.

Cons: It is necessary to fill in a form to try the trial version, but there is no guarantee that the application will be accepted. The local version of the tool doesn't work on Mac OS, and it is necessary to use the cloud version. No real-time collaboration, and reverse engineering.

Navicat

Navicat is proprietary software created in 2002 and provides a mini-map for fast navigation. It allows adding colors to tables, thus making them more visually appealing. This tool does not require any configuration and does not have real-time collaboration. It is a powerful and cost-effective database design tool that allows designing CM. It allows performing reverse and forward engineering processes. This tool allows creating data models for MySQL, Microsoft SQL Server, Oracle, PostgreSQL, SQLite, and MariaDB databases. It has several data types, such as blob, boolean, integer, varchar, date, and timestamp. For the data constraints, there are PK, FK, NN, AI, and UQ.

Pros: Performs reverse and forward engineering.

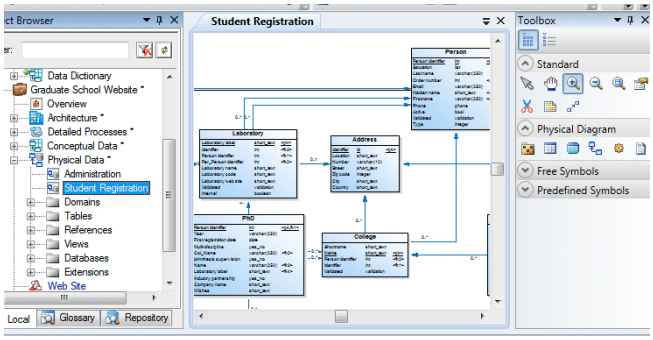


FIGURE 4: Example of a database model in PowerDesigner

Cons: No real-time collaboration.

Oracle SQL Developer Data Modeler

They released Oracle SQL Developer in 2006, and it is an integrated development environment that simplifies the development and management of Oracle Databases. It is unnecessary to do any configuration. This application permits conceptual data modeling. Although, it does not have real-time collaboration. The application allows performing forward engineering for the Oracle Database engine. It supports managing the Oracle Database performance, security, storage, and settings. The tool has different data types such as blob, char, decimal, float, date, and timestamp. There is the possibility to define PK, FK, AI, and UQ constraints.

Pros: Quick loading time, and the existence of a tutorial that explains how the tool works.

Cons: Only supports Oracle databases.

PowerDesigner

PowerDesigner is a collaborative enterprise modeling tool, released in 1989 with the name of "AMC*Designer" and is currently owned by SAP. It is a modeling tool for easy visualization, understanding, and management of the data in a project (Fig. 4). It is unnecessary to make any configurations. PowerDesigner has real-time collaboration, and it is possible to implement a CM. Also, it has multiple database connections to model the data. The tool also offers reverse engineering. However, forward engineering allows working with the most popular data management systems, such as Oracle, PostgreSQL, IBM DB2, SQP IQ, Microsoft SQL Server, and Teradata. This tool only works on Windows OS. PowerDesigner allows creating multiple entities at once and saves a lot of time. The present data types are integer, decimal, money, boolean, characters, text, date, and timestamp. For the data constraints, there are PK, FK, NN, AI, and UQ.

Pros: Has real-time collaboration, and several database engines for forward engineering.

Cons: Only available for Windows, and no reverse engineering.

In this section, we did a qualitative evaluation of each tool by presenting concise descriptions and pros and cons. Table

TABLE 2: Comparison of different tools

Product Type	Tool name	Characteristics							
		Open source	Online	Supported databases	Supported OS	Constraints	CM design	Free	Licence
I	Dbdesigner.id	✓	✓	MySQL	Windows, Linux, macOs	PK, FK, UQ, AI, NN	✗	✓	MIT
	Onda	✓	✓	MySQL, postgreSQL, MariaDB, Oracle, SQLite	Windows, Linux, macOs	PK,FK,NN,CH,AI,UQ	✓	✓	CCANCSAIL*
	WWW SQL Designer	✓	✓	MySQL, sqlLite, Oracle, PostgreSQL, mssql, web2py	Windows, Linux, macOs	PK, FK, UQ, AI, NN	✗	✓	BSD-3-Clause
II	Dbdesigner.net	✗	✓	MySQL, Microsoft SQL, PostgreSQL, Oracle, SQLite	Windows, Linux, macOs	PK, FK, UQ, AI, NN	✗	✗	Proprietary
	dbDiffo	✗	✓	IBM DB2, MS SQL Server, MySQL, Oracle, PostgreSQL	Windows, Linux, macOs	PK, FK, NN, AI	✗	✗	Proprietary
	GenMyModel	✗	✓	Apache Hive, Oracle, MySQL, PostgreSQL	Windows, Linux, macOs	PK, FK	✓	✗	Proprietary
	Lucidchart	✗	✓	MS SQL Server, MySQL, Oracle	Windows, Linux, macOs	PK, FK, NN, AI	✓	✗	Proprietary
	sqlDBM	✗	✓	MS SQL Server, MySQL, Snowflake, Amazon Redshift, PostgreSQL, Azure Synapse Analytics	Windows, Linux, macOs	PK, FK, NN, AI	✗	✗	Proprietary
III	MySQLWorkbench Community Version	✓	✗	MySQL	Windows	PK, FK, NN, UQ, AI, Binary, Unsigned	✓	✓	GPL
	pgModeler	✓	✗	PostgreSQL	Windows, Linux, macOs	PK, FK, UQ, E, CH	✓	✓	GPL
	Umbrello UML	✓	✗	mySQL, PostgreSQL	Windows, macOs	PK, FK, AI, UQ	✓	✓	LGPL
IV	dbSchema	✗	✗	SqlServer, SAP Adaptive Server, Oracle, MySql, Ingres, Informix, Db2, Derby, Firebird, Frontbase, Cache, Pervasive, PostgreSQL, SQlite	Windows, Linux, macOs	PK, FK, NN, AI	✗	✗	Proprietary
	dbWrench	✗	✗	Microsoft SQL Server, Oracle, PostgreSQL, MySQL	Windows, Linux, macOs	PK, FK, NN, AI	✗	✗	Proprietary
	Erwin Data Modeler	✗	✗	Oracle, MySQL, IBM, DB2, SAP IQ, Teradata	Windows, Linux, macOs	PK, FK, AI, NN	✓	✗	Proprietary
	Navicat	✗	✗	MySQL, Microsoft SQL Server, Oracle, PostgreSQL, SQLite, MariaDB	Windows, Linux, macOs	PK, FK, UQ, AI, NN	✓	✗	Proprietary/ Shareware
	Oracle SQL Developer Data Modeler	✗	✗	Oracle, DB2, MySQL, and Microsoft SQL Server MySQL	Windows, Linux, macOs	PK, FK, UQ, AI, NN	✓	✗	Proprietary
	PowerDesigner	✗	✗	Oracle, PostreSQL, IBM DB2, SQP IQ, Microsoft SQL, Teradata	Windows	PK, FK, UQ, AI, NN	✓	✗	Proprietary

Constraints meaning: PK - Primary Key; FK - Foreign Key; UQ - Unique; AI - Auto Increment; NN - Not Null; E - Exclude; CH - Check; U - Unsigned

*Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License

2 displays a summary of the fundamental characteristics of each. In the following section, we will explain the evaluation method. And the scoring assessment. Hence, producing an objective analysis.

V. THE EVALUATION METHODOLOGY

Since we focus on evaluating data modeling tools for relational databases, this section will explain how to perform this assessment.

We are proposing a hybrid evaluation methodology that will suppress the previously mentioned shortcomings. And for that, we developed an approach based on the BRR calculations, but instead of the 12 categories, we will use the seven defined in the OSSpal methodology, these encompass all significant categories by combining some of the original 12 into a single category. Also, we introduced some changes in the evaluation of the Functionality features of BRR. First, we removed the binary answers by providing a broader range of values. Second, we discarded adding extra features, no bonus score for tools, and only scored a tool with -1 when a feature is absent from the proposed set.

The OSSpal methodology combines quantitative and qualitative measures to evaluate and compare software tools in several categories. The examiner assigns a quantitative score to the tools instead of only analyzing the pros and cons. OSSpal proposes the following seven categories:

- **Functionality**: analyses how well the software meets the user's requirements.
- **Operational Software Characteristics**: evaluates how secure the software is, how well does it perform, how good is the User Interface (UI), and how easy is the software to install, configure, deploy, and maintain.
- **Support and Service**: examines how well is the software component supported and if there is commercial or community support or both.
- **Documentation**: assesses if there is a suitable tutorial and reference documentation for the software.
- **Software Technology Attributes**: analyses how good the software architecture is and how portable, extensible, open, and easy to integrate it is.
- **Community and Adoption**: examines the adoption of the component by community, market, and industry. Also, how active is the community for the software.
- **Development Process**: evaluates the level of profes-

sionalism of the development process and the project organization.

After assessing the previous OSSpal categories, four steps were followed according to the BRR model:

1) Identify all the software components to be analyzed and measure them considering the evaluation criteria.
2) Select an appropriate weighting factor for each category/metric (sub-categories):
   a) Assign a percentage of importance for each category that will be used as a weighting factor. The sum of all the weigh factors should add up to 100%.
   b) Similarly, if a specific category is evaluated using multiple metrics, assign a percentage of importance for each metric, totaling 100% within the category.
3) Score each category/metric and attribute it a value from 1 to 5 (1 - Unacceptable, 2 - Poor, 3 - Acceptable, 4 - Very Good, 5 - Excellent).
4) The category evaluation and the weighting factors (Eq. 1) should be used to calculate the final score (Eq. 2).

Category Score =

$$\frac{\sum Metric\ score \times Metric\ weight}{\sum Metric\ weights} \quad (1)$$

Final Score =

$$\frac{\sum Category\ score \times Category\ weight}{\sum Category\ weight} \quad (2)$$

As defined in the BRR model [7], the Functionality category will have a different approach because *"each type of software application has a unique set of features that needs to be fulfilled by the software package"*. The original model evaluates the presence of features, not being a typical qualitative or quantitative measurement. However, in our hybrid approach we introduce a quantitative measure of these features to highlight the differences between the tools. The method to assess this category ends with Equation 3, and is as follows:

- Specify the features to analyze, weighting them from 1 to 3 (less important to very important);
- Compare the feature list of the component being evaluated with the standard feature list. For each feature:
  - If met, classify the implementation of the feature using a scale from 1 to 3 (poor implementation to full implementation) and multiply by the feature weight;
  - If not met, deduct importance weight from the sum (classify as -1)
- Standardize the result to a scale from 1 to 5 (Tab. 3):
  - The result is the cumulative sum of all the feature results, which punishes the missing features.

Functionality Score =

$$\frac{\sum Feature\ score \times Feature\ weight}{\sum Feature\ weights} \quad (3)$$

TABLE 3: Quality ranking

Values	Score	Evaluation
>96%	5	Excellent
[90% - 96% [	4	Good
[80% - 90% [	3	Acceptable
[65% - 80% [	2	Poor
<65%	1	Unacceptable

Equation 3 is nothing more than an example of Equation 1, specifically for the Functionality category.

VI. MODELING TOOLS EVALUATION

In this section, to evaluate the data modeling tools, we will show the application of the methodology proposed in the previous section.

To define the weights to be given to the OSSPal categories, we carried out a survey. In this survey, we consulted with 18 professionals in the area of databases and software engineering. First, we asked them to give a score (in percentage) of the seven OSSpal categories totaling 100% according to their experience in database modeling and general software engineering. Then, we asked them to ponder, with values ranging from one to three, the chosen features of the Functionality category. Tables 4 and 5 display the statistics of our survey.

Regarding the *Category* (Table 4), we asked to fill each Category with a percentage totaling 100%. We analyzed the average, standard deviation (STDDEV), and median. The maximum and minimum values of each Category were disparate. The lower value was a 15% difference between the values in the *Support and Service* and *Development Process* categories, up to 42% in the *Functionality* category. Thus, we evaluated the average without the maximum and minimum values and presented the difference between these values. The only positive difference is in the *Operational Software Characteristics* category. We considered the average values because of three reasons: i) the sum of the average values total 100%; ii) the sum of average without maximum and minimum values was only 98.09%, and the differences are not significant between the two approaches; iii) the sum of median values was only 92.50%, and the differences are not significant between the two approaches except for the *Functionality* with a difference of -5.17%.

The *Functionality* Category had the highest weight, and we further decomposed it into features or sub-categories. We asked the same community to rank and weight each measure with a value between 1 (less important) and 3 (very important). We also analyzed the average, standard deviation (STDDEV), and median. The maximum and minimum values of each Functionality were equivalent amongst the functionalities. Because the variation of the values is minor, the differences between averages (with and without maximum and minimum values) are not significant (0.01), we used the average values.

TABLE 4: OSSpal Category weights (%)

Category	AVG	STDDEV	MEDIAN	MAX	MIN	AVG (w/o MAX-MIN)	AVG (w/o MAX-MIN)-AVG
Functionality	**32.67**	13.37	27.50	60	18	31.88	-0.79
Operational Software Characteristics	**19.28**	5.87	20.00	30	5	19.50	0.22
Documentation	**10.83**	5.17	10.00	20	3	10.75	-0.08
Support and Service	**10.58**	4.94	10.00	20	5	10.34	-0.24
Software Technology Attributes	**10.22**	5.09	10.00	20	2	10.13	-0.10
Community and Adoption	**10.22**	6.36	10.00	30	2.5	9.47	-0.75
Development Process	**6.19**	4.40	5.00	15	0	6.03	-0.16
Sum	**100**	-	92.50	-	-	98.09	-

TABLE 5: Functionality Features Weights

Measures	AVG	STDDEV	MEDIAN	MAX	MIN
Supported databases	**2.37**	0.58	2.00	3.00	1.00
Supported constraints	**2.21**	0.61	2.00	3.00	1.00
Supported OS	**1.95**	0.69	2.00	3.00	1.00
Reverse engineering	**1.84**	0.81	2.00	3.00	1.00
Real-time collaboration	**1.79**	0.77	2.00	3.00	1.00
CM design	**1.79**	0.69	1.84	3.00	1.00
Sum	**11.95**				

TABLE 6: Measures of the Functionality Category for each tool

Product Type	Tool name	Measures and weights						Weighted total	W.Total / Sum Weights (Eq. 3)	Percentage (%)	Normalization Tab 3
		Supp. DBs	Supp. constraints	Supp. OS	Reverse engineering	Real-time collab.	CM design				
		(2.37)	(2.21)	(1.95)	(1.84)	(1.79)	(1.79)				
I	Dbdesigner.id	1	3	3	-1	3	-1	16.58	1.39	46.26	**1**
	Onda	3	3	3	-1	-1	3	21.32	1.78	59.47	**1**
	WWW SQL Designer	1	3	3	-1	-1	-1	9.42	0.79	26.28	**1**
II	Dbdesigner.net	3	3	3	3	3	-1	28.68	2.40	80.03	**3**
	dbDiffo	3	2	3	-1	-1	3	19.11	1.60	53.30	**1**
	GenMyModel	3	3	3	-1	3	3	28.47	2.38	79.44	**2**
	Lucidchart	3	2	3	-1	3	3	26.26	2.20	73.27	**2**
	sqlDBM	3	3	3	3	-1	-1	21.53	1.80	60.06	**1**
III	MySQL Workbench	1	3	1	3	-1	3	20.05	1.68	55.95	**1**
	pgModeler	1	3	3	3	-1	3	23.95	2.00	66.81	**2**
	Umbrello UML	2	3	2	-1	-1	-1	9.84	0.82	27.46	**1**
IV	dbSchema	3	2	3	3	-1	-1	19.32	1.62	53.89	**1**
	dbWrench	3	2	3	-1	-1	-1	11.95	1.00	33.33	**1**
	Erwin Data Modeler	3	3	3	-1	-1	-1	14.16	1.19	39.50	**1**
	Navicat	3	3	3	3	-1	-1	21.53	1.80	60.06	**1**
	Oracle SQL Developer	3	3	3	-1	-1	3	21.32	1.78	59.47	**1**
	PowerDesigner	3	3	1	3	3	3	31.95	2.67	89.13	**3**

The primary category is *Functionality* as it encompasses, among others, the number of supported databases engines, the restrictions it has, and constraints. Therefore, this category got a weight of **32.67%**. In the second place, *Operational Software Characteristics* has **19.28%** as well and includes areas such as security, performance, usability, reliability, and scalability, which are crucial to evaluate each tool. The *Documentation* category comes in third with **10.83%** once good information helps with installation, configuration, and extension of the software easily. *Support and Service*, *Community and Adoption*, and *Software Technology Attributes* had **10.58%**, **10.22%**, and **10.22%** respectively because the software needs to be supported, modular, easy to extend, and integrate. The latter category also measures if the project is extensible and how fast problem resolution is. With less importance, *Development Process* categories received a weight of and **6.19%** respectively. Table 4 represents these weights, ordered from most to less important, based on the average (in bold).

The next step was to decompose the Functionality category into the most relevant characteristics (features or subcategories). Table 5 displays the results of our survey to the research community survey, previously explained, the average (in bold) ordered these, leading to the following layout: Supported databases (2.37), Supported constraints (2.21), Supported OS (1.95), Reverse engineering (1.84), Real-time collaboration (1.79), and CM design (1.79). The sum of the weights was **11.95**.

TABLE 7: Assessment of the OSSpal Categories' score for each tool

Product Type	Tool name	Categories						
		Functionality (32.67)	Operational Software Characteristics (19.28)	Documentation (10.83)	Support and Service (10.58)	Software Technology Attributes (10.22)	Community and Adoption (10.22)	Development Process (6.19)
I	Dbdesigner.id	1	5	2	2	4	3	3
	Onda	1	4	1	3	4	2	4
	WWW SQL Designer	1	3	4	5	4	3	5
II	Dbdesigner.net	3	5	5	2	3	3	3
	dbDiffo	1	3	4	3	3	2	3
	GenMyModel	2	3	3	2	3	3	3
	Lucidchart	2	4	3	4	3	4	5
	sqlDBM	1	5	4	2	3	3	5
III	MySQL Workbench	1	4	2	2	4	3	3
	pgModeler	2	4	5	5	4	4	4
	Umbrello UML	1	3	5	4	4	5	5
IV	dbSchema	1	4	5	3	3	3	5
	dbWrench	1	3	3	3	3	3	3
	Erwin Data Modeler	1	4	4	2	3	3	3
	Navicat	1	3	4	2	3	3	3
	Oracle SQL Developer	1	5	3	2	3	3	3
	PowerDesigner	3	3	4	2	3	3	5

Table 6 displays the result of the assessment of each Feature of the Functionality Category, according to the methodology criteria. Between brackets are the average weight values that resulted from our community survey. As previously mentioned, any software that does not incorporate a measure, its weight (importance) will be deducted. Also, to better differentiate the tools, we added a wider range for the values depending on each measure:

1) Supported databases, this category accounts for the number of database engines that the tool can generate SQL scripts from the conceptual model. This characteristic has the highest weight value because it is important for a tool to have a wide range of choices, not limit the user. The tools that support one database got the score of one, the tools that support two, a score of two, and three or more databases receive a score of three;
2) The number of different constraints that the tool has goes in the feature: Supported constraints. If it has at least two of the following group PK, AI, and NN, the tool receives a score of two, and those that have these or more, such as UQ or E have a score of three;
3) In the feature Supported OS, we considered three OS: Windows, Linux, and macOS. Each tool will score one point for each supported OS. Thus, a tool like Powerdesigner that only supports Windows will score one, Umbrello UML supports Windows and macOS will score two, and the rest three.
4) Only some tools have reverse engineering, so those that have it receive a score of three, otherwise the set importance will be deducted from the cumulative sum, so the score will be set as -1;
5) Regarding the feature Real-time collaboration, the tools that have it received three, otherwise the set importance will also be deducted from the cumulative sum, so the score will be set as -1;
6) The evaluation of the feature CM design is according to the tools' capacity to provide a CM design. If provided, the tool has a score of three, otherwise the set importance will be deducted from the cumulative sum, so the score will be set as -1.

Following the proposed methodology, we calculated the Weighted total, representing the cumulative multiplication of the score by the feature's weight (importance). For example, Dbdesigner. id Functionality score (Equation 3) is calculated as: $[(1 \times 2.37) + (3 \times 2.21) + (3 \times 1.95) + (-1 \times 1.84) + (3 \times 1.79) + (-1 \times 1.79)]/11.95 = 1.39$.

Using this value, we assessed the percentage of each tool score. Finally, we converted the values according to Table 3, represented in the last column. None of the tools achieved the highest score (5). Dbdesigner.net and PowerDesigner, both with a score of three (3), were the best-rated tools in this Category.

The last step is to calculate each tool's score, using Equation 2, taking into account the values of different categories from the Table 7, multiplying it by its weights, where we converted the percentage into unit values, and adding up these scores. The result of these calculations is shown in Table 8.

After applying the proposed methodology to the tools (Eq. 2), in the product type of Online free tools, **WWW SQL Designer** received the highest score, **2.89**. This tool has some shortcomings, such as supporting only one database, not allowing real-time collaboration, and not having relationship cardinalities. These contributed to a low value in the Functionality category, the one that has the highest weight. However, in the remaining categories, the tool performs well, giving it a prominent place.

In the product type of Online commercial tools, **Dbdesigner.net** ended the assessment with **3.50**, the overall

TABLE 8: Final score

Product Type	Tool name	Score
I	WWW SQL Designer	**2.89**
	Dbdesigner.id	2.62
	Onda	2.38
II	Dbdesigner.net	**3.50**
	Lucidchart	3.20
	sqlDBM	2.86
	GenMyModel	2.57
	dbDiffo	2.35
III	pgModeler	**3.56**
	Umbrello UML	3.10
	MySQL Workbench	2.43
IV	PowerDesigner	**3.13**
	dbSchema	2.88
	Oracle SQL Developer	2.63
	Erwin Data Modeler	2.54
	Navicat	2.35
	dbWrench	2.35

highest value. This tool has a high number of supported databases, restrictions, and operating systems. It supports reverse engineering and real-time collaboration. Although it is not extensible, the tool is easy to work with, with good documentation, qualified support, and broadly adopted by the community.

In the product type of Desktop free software, **pgModeler** received the highest score, **3.56** out of 5. This tool got better results, despite its Functionality score, because it only supports PostgreSQL and does not have real-time collaboration. PgModeler is extensible, has clear documentation and widely adopted by the community.

In the product type of Desktop commercial software, **PowerDesigner** got the highest score, **3.13**. This tool supports several database engines, cardinality options, real-time collaboration, allows performing reverse engineering, and it has good documentation. However, it only supports one Operating System (OS).

All the previously mentioned and analyzed tools offer unique resources to its users, and the differences in the final score display their heterogeneity. This classification results mainly from the fundamental differences in the Functionality category. From the six features in our approach, all tools have three (with the highest importance according to our survey). However, the BRR model imposes a penalization when a tool has a missing feature.

In addition, it allows us to generate results that accurately reflect the usefulness of the tools and their level of productivity for users. Also, we aimed to identify all crucial Functionality characteristics. Ultimately, the choice must also encompass the purpose of the project and the developer's skills. Nevertheless, we only considered six features. Despite our survey to classify their importance, we acknowledge the threat that, for other evaluators, these specific features may not be the best, the most adequate, or the most representative for data modeling tools. We tried to remove the subjectivity by gathering input from several database and software engineering professionals, but based on the data presented. Anyone can change the weights as they see fit to suit their specific context, thus achieving more adequate results for their needs.

VII. CONCLUSIONS

This paper covers the evaluation of six open-source and eleven proprietary database modeling tools using a new and tailored approach. We gathered information for this analysis from the documentation available on each tool's website, by installing, testing, and using the tools. Also, we selected the proprietary desktop software according to Google Trends, then applied filter criteria because the number of modeling tools is high.

We used a hybrid methodology based on BRR and OSSpal and made a survey within our department research community to assess category and functionalities weigths. WWW SQL Designer (2.89), Dbdesigner.net (3.50), pgModeler (3.56), and PowerDesigner (3.13) are the most valuable tools in each product type. Overall, pgModeler, an open-source desktop tool, achieved the highest value amongst the 17 evaluated tools.

To provide a meaningful evaluation of database modelling tools, it was necessary to develop a hybrid methodology, since neither BRR nor OSSPal were able to fully satisfy the requirements of this analysis per se. BRR is prone to ignore important information, since it considers subjectively the seven highest ranked categories out of the 12 possible. And the OSSpal methodology does not provide a referential final score. Furthermore, the combination of both methodologies and the further improvement to the functional evaluation, have shown that our method provides more meaningful and tailored results for the evaluators, that can be also adapted to support decision in specific contexts, just by changing the weights used.

ACKNOWLEDGMENTS

This work is also supported by the European Regional Development Fund (FEDER), through the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the PORTUGAL 2020 framework [Project InfraCrit with Nr. 039555 (POCI-01-0247-FEDER-039555)].

REFERENCES

[1] S. B. Navathe, "Evolution of data modeling for databases," Communications of the ACM, vol. 35, no. 9, pp. 112–123, 1992.
[2] "IBM definition of Data Modeling." https://www.ibm.com/cloud/learn/data-modeling. Accessed: 27-04-2021.
[3] G. Simsion and G. Witt, Data modeling essentials. Elsevier, 2004.
[4] "Wikipedia definition of Data Modeling." https://www.ibm.com/cloud/learn/data-modeling. Accessed: 27-04-2021.
[5] R. Kimball and M. Ross, The DataWarehouse Toolkit. John Wiley & Sons Inc, 2004.

[6] A. Silberschatz, H. F. Korth, and S. Sudarshan, Database system concepts. New York, New York, USA: McGraw-Hill Education, 7th ed., 1991.
[7] OpenBRR, "Business Readiness Rating for Open Source," tech. rep., 2005.
[8] A. I. Wasserman, X. Guo, B. McMillian, K. Qian, M.-Y. Wei, and Q. Xu, "Osspal: Finding and evaluating open source software," pp. 193–203, 2017.
[9] International Data Corporation, "International Data Corporation Software Taxonomy," 2016.
[10] T. Ferreira, I. Pedrosa, and et al., "Evaluating open source business intelligence tools using osspal methodology," in Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (A. L. N. Fred and J. Filipe, eds.), pp. 283–288, SciTePress, 2017.
[11] N. Leite, I. Pedrosa, and et al., "Open source business intelligence platforms' assessment using osspal methodology," in Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, ICETE 2018 - Volume 1: DCNET, ICE-B, OPTICS, SIGMAP and WINSYS, Porto, Portugal, July 26-28, 2018 (C. Callegari, M. van Sinderen, P. Novais, P. G. Sarigiannidis, S. Battiato, Á. S. S. de León, P. Lorenz, and M. S. Obaidat, eds.), pp. 356–362, SciTePress, 2018.
[12] A. K. Pereira, A. P. Sousa, J. R. Santos, and et al., "Open source data mining tools evaluation using osspal methodology," in Proceedings of the 13th International Conference on Software Technologies, ICSOFT 2018, Porto, Portugal, July 26-28, 2018 (L. A. Maciaszek and M. van Sinderen, eds.), pp. 706–712, SciTePress, 2018.
[13] T. Ferreira, I. Pedrosa, and et al., "Evaluating open source e-commerce tools using osspal methodology," in Proceedings of the 20th International Conference on Enterprise Information Systems, ICEIS 2018, Funchal, Madeira, Portugal, March 21-24, 2018, Volume 1 (S. Hammoudi, M. Smialek, O. Camp, and J. Filipe, eds.), pp. 213–220, SciTePress, 2018.
[14] H. C. de Paula and et al., "An application of osspal for the assessment of open source project management tools," in Proceedings of the 15th International Conference on Web Information Systems and Technologies, WEBIST 2019, Vienna, Austria, September 18-20, 2019 (A. Bozzon, F. J. D. Mayo, and J. Filipe, eds.), pp. 411–417, ScitePress, 2019.
[15] A. Oliveira and et al., "Evaluating open source project management tools using osspal methodology," in Proceedings of the 15th International Conference on Web Information Systems and Technologies, WEBIST 2019, Vienna, Austria, September 18-20, 2019 (A. Bozzon, F. J. D. Mayo, and J. Filipe, eds.), pp. 343–350, ScitePress, 2019.
[16] J. F. Marques and et al., "Evaluation of asana, odoo, and projectlibre project management tools using the osspal methodology," in Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2019, Volume 2: KEOD, Vienna, Austria, September 17-19, 2019 (J. L. G. Dietz, D. Aveiro, and J. Filipe, eds.), pp. 397–403, ScitePress, 2019.
[17] S. Cruz and et al., "Project management tools assessment with osspal," in Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2019, Volume 2: KEOD, Vienna, Austria, September 17-19, 2019 (J. L. G. Dietz, D. Aveiro, and J. Filipe, eds.), pp. 390–396, ScitePress, 2019.
[18] A. Calçada and et al., "Evaluation of couchbase, couchdb and mongodb using osspal," in Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2019, Volume 1: KDIR, Vienna, Austria, September 17-19, 2019 (A. L. N. Fred and J. Filipe, eds.), pp. 427–433, ScitePress, 2019.
[19] D. Batra and G. M. Marakas, "Conceptual data modelling in theory and practice," European Journal of Information Systems, vol. 4, no. 3, pp. 185–193, 1995.
[20] D. L. Moody and G. G. Shanks, "What makes a good data model? Evaluating the quality of entity relationship models," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 881 LNCS, pp. 94–111, 1994.
[21] D. Brdjanin, G. Banjac, D. Banjac, and S. Maric, "An experiment in model-driven conceptual database design," Software and Systems Modeling, vol. 18, no. 3, pp. 1859–1883, 2019.
[22] B. Thalheim, Entity-Relationship Modelling: Foundations of Database Technology. Springer-Verlag, Inc, New York, 2000.
[23] P. Chen, "The Entity-Relationship Model—toward a Unified View of Data," ACM Transactions on Database Systems (TODS), vol. 1, no. 1, pp. 9–36, 1976.
[24] "Welcome To UML, https://www.uml.org/," 2019.
[25] S. Al-Fedaghi and H. Alahmad, "Orientation in Conceptual Modeling Frameworks," in IEEE 15th International Conference on Dependable, Autonomic and Secure Computing, IEEE 15th International Conference on Pervasive Intelligence and Computing, IEEE 3rd International Conference on Big Data Intelligence and Computing, pp. 1298–1303, 2018.
[26] N. E. Adrienne Watt, Database Design, 2nd Edition. The BCcampus Open Textbook Project, 2014.
[27] "79 Data Modeling Tools Compared." https://www.databasestar.com/data-modeling-tools/. Accessed: 16-10-2020.
[28] "20 Best Data Modeling Tools." https://www.guru99.com/data-modeling-tools-design-database.html. Accessed: 16-10-2020.

GONÇALO CARVALHO received the B.Sc. in Geography from University of Coimbra (UC), the M.Sc. degree in Geographical Information Systems from University of Trás-os-Montes e Alto Douro (UTAD) and B.Sc. in Informatics Engeneering from Polytechnic of Coimbra (IPC), institutions located in Portugal, in 2005, 2009 and 2016, respectively. After the experience as a web and software developer between 2015 and 2018, he currently has a studentship for his Ph.D. research from the University of Coimbra. His main research interests are in the areas of databases, distributed systems, edge computing, cyber-physical systems, and machine learning.

SERGII MYKOLYSHYN received his Bachelor's degree from the University of Coimbra (UC) in 2019. He is currently pursuing his Masters's degree in Informatics Engineering, specializing in Software Engineering, also in the UC. His main research interest areas are databases and Cloud/Edge Computing.

BRUNO CABRAL concluded his Ph.D. with honors in 2009 at the University of Coimbra (UC) in the area of Informatics Engineering. Bruno holds a Tenured Professor position with the Informatics Engineering Department of the University of Coimbra (UC). Bruno has been an Adjunct Associate Teaching Professor at the Carnegie Mellon University (CMU), USA, and a faculty of the Dual-degree Masters in Software Engineering (MSE). He is the coordinator of the Master Program in Software Engineering at the UC and a senior researcher of the Systems and Software Engineering group of Centre of Informatics and Systems of the University of Coimbra (CISUC). His main research interests are in the areas of distributed systems, parallel programming languages, machine learning, and dependable computing. Bruno was either the PI or staff member on multiple EU, Government and privately funded research projects, and frequently works as a consultant to the software industry.

JORGE BERNARDINO (Member, IEEE) received the Ph.D. degree from the University of Coimbra, in 2002. From 2005 to 2010, he was the President of ISEC (Coimbra Engineering Institute). From 2017 to 2019, he was also the President of ISEC Scientific Council. In 2014, he was a Visiting Professor at CMU. He was the Director of the Applied Research Institute (i2A) of Polytechnic of Coimbra from 2019 to 2021. He is currently a Coordinator Professor with the Polytechnic of Coimbra—ISEC, Portugal. He has authored more than 200 publications in refereed conferences and journals and participated in several national and international projects. His main research interests include big data, NoSQL, data warehousing, dependability, and software engineering.

VASCO PEREIRA received his Ph.D in Informatics Engineering in 2016, from the Faculty of Sciences and Technology of the University of Coimbra (Portugal). He is currently an Assistant Professor at the Department of Informatics Engineering at the same university and the vice-coordinator of the bachelor's degree in Informatics Engineering. He is also a researcher at the Laboratory of Communications and Telematics of the Centre of Informatics and Systems of the University of Coimbra (CISUC) where he has been involved in national and European research projects. He is currently involved in projects such as 5G - Components and Services for 5G Networks (POCI-01-0247-FEDER-024539), MobiWise - from mobile sensing to mobility advising (P2020 SAICTPAC/0011/2015) and SOCIALITE - Social-Oriented Internet of Things Architecture, Solutions and Environment (POCI-01-0145-FEDER-016655). His main research interests include QoS, performance in Wireless Sensor Networks and IoT. His homepage can be reached at http://faculty.uc.pt/uc26416.