UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

A novel cloud and fog-based architecture to support spatial analytics in smart cities

João Paulo Clarindo dos Santos

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

João Paulo Clarindo dos Santos

A novel cloud and fog-based architecture to support spatial analytics in smart cities

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Cristina Dutra de Aguiar

USP – São Carlos
January 2022

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

S237n

Santos, João Paulo Clarindo dos
A novel cloud and fog-based architecture to support spatial analytics in smart cities / João Paulo Clarindo dos Santos; orientadora Cristina Dutra de Aguiar. -- São Carlos, 2022.
107 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2022.

1. IoT. 2. spatial data warehouses. 3. fog computing. 4. smart cities. 5. parallel and distributed processing. I. Aguiar, Cristina Dutra de, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

João Paulo Clarindo dos Santos

Uma nova arquitetura baseada em computação em nuvem e computação em névoa para a análise de dados espaciais em cidades inteligentes

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Cristina Dutra de Aguiar

USP – São Carlos
Janeiro de 2022

This work is dedicated to all data lovers.

ACKNOWLEDGEMENTS

I thank my parents and my family for all the support and encouragement for me to move to a city 1,895 km away from my hometown, Maceió, Brazil, in order to fulfil my biggest dream. Without you, this dissertation would not be possible.

I thank all my friends. All were essential. I can't name them all here, so feel represented.

I thank my graduate advisor, Prof. Fabio J. Coutinho, from the Federal University of Alagoas (Ufal), for his great support during the dissertation, whether in the academic context or in other activities. I also extend my thanks to all friends and colleagues from *Instituto de Computação* – Ufal.

I thank the ICMC/USP for offering a high-quality education, with qualified professors, administrative technicians, and collaborators. Proud to be a student of one of the best postgraduate programs in computing in Brazil!

I thank the Brazilian National Council for Scientific and Technological Development (CNPq) for the financial support (grant 133951/2019-7).

And finally, I thank Professor Cristina Dutra de Aguiar, for her high-quality guidance. In addition, she helped me a lot in the difficulties I went through during this project, and I am honoured that I will continue my doctoral journey together. Thanks for everything! I also extend my thanks to my research group friends, in particular, João Pedro Castro and Juliana Freitas, for their great contributions.

“Information is the oil of the 21st century,
and analytics is the combustion engine.”
(Peter Søndergaard)

ABSTRACT

SANTOS, J. P. C. **A novel cloud and fog-based architecture to support spatial analytics in smart cities**. 2022. 107 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2022.

Providing an infrastructure to accommodate a large number of people in cities is a major challenge for public authorities and private companies. Thereby, the concept of smart cities emerged, which use technologies like sensors and Internet of Things (IoT) devices to aid in urban growth. These devices generate spatial data that can be used for spatial analytics by smart city managers to improve the population's quality of life. However, these IoT devices quickly generate a large volume of spatial data, causing big data problems. A smart city manager can benefit from using concepts such as fog computing, spatial data warehouses, data lakes, and parallel and distributed storage and processing environments to handle this massive amount of data. Based on a systematic review, there are no studies in the literature that consider all of these concepts in the context of smart cities. Therefore, we propose a novel architecture that aims smart city managers in spatial analytics. This architecture is composed of four layers: (i) terminal, which consists of a network of IoT devices; (ii) fog computing, which contains data lakes for real-time data processing; (iii) cloud computing, in which spatial data warehouses are used to support SOLAP (Spatial Online Analytical Processing) queries carried out in batch; and (iv) analytical tools, which incorporate data visualisation and analysis tools. Furthermore, we introduce a set of guidelines to aid smart cities managers to implement the proposed architecture, by describing and discussing important issues and examples of tools and technologies. The proposed architecture and guidelines were validated through two case studies that use real data generated by IoT devices disposed in smart cities. We investigated the execution of three categories of spatial queries, as well as the execution of queries in the fog, in the cloud, and in both environments. These case studies demonstrated the architecture's efficiency and effectiveness to support spatial analytics in the context of smart cities.

Keywords: IoT, spatial data warehouses, fog computing, smart cities, parallel and distributed processing.

RESUMO

SANTOS, J. P. C. **Uma nova arquitetura baseada em computação em nuvem e computação em névoa para a análise de dados espaciais em cidades inteligentes**. 2022. 107 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2022.

Prover uma infraestrutura para acomodar uma grande quantidade de pessoas em cidades tem se mostrado um grande desafio para o poder público e empresas privadas. Logo, surgiu o conceito de cidades inteligentes, que utilizam tecnologias para auxílio no crescimento urbano. Essas tecnologias, que consistem em sensores e dispositivos de internet das coisas (ou *Internet of Things*, IoT), geram dados espaciais, que podem ser utilizados no auxílio à tomada de decisão por gestores de cidades inteligentes para a melhoria da qualidade de vida da população. Entretanto, esses dispositivos IoT geram um grande volume de dados espaciais, de forma veloz, ocasionando problemas de *big data*. Um gestor de cidades inteligentes pode se beneficiar com o uso de conceitos como computação em névoa, *data warehouses* espaciais, *data lakes* e ambientes de processamento e armazenamento paralelo e distribuído para lidar com esse grande volume de dados e a necessidade de análise voltada à tomada de decisão. Entretanto, com base em uma revisão sistemática, não foram identificados estudos que aplicam todos esses conceitos no contexto de cidades inteligentes. Essa limitação motiva o desenvolvimento desta dissertação de mestrado, na qual são introduzidas as seguintes contribuições. Primeiramente, é proposta uma arquitetura que visa auxiliar gestores de cidades inteligentes no processo analítico de dados espaciais. Essa arquitetura envolve quatro camadas: (i) terminal, que consiste em uma rede de dispositivos IoT; (ii) computação em névoa, que contém *data lakes* para o processamento de dados em tempo real; (iii) computação em nuvem, na qual são utilizados *data warehouses* espaciais para prover suporte para consultas *Spatial On-line Analytical Processing (SOLAP)* em lote; e (iv) ferramentas analíticas, que incorpora ferramentas de visualização e análise dos dados. Além disso, são propostas diretrizes para auxílio na implementação da arquitetura proposta, com discussão de desafios que devem ser investigados e com exemplos de tecnologias e ferramentas que podem ser empregadas. A arquitetura e as diretrizes propostas foram validadas por meio de dois estudos de caso que utilizam dados reais gerados em cidades inteligentes. Nesses estudos de caso, foram investigados a execução de diferentes categorias de consultas espaciais e o processamento de consultas espaciais na névoa, na nuvem ou em ambos os ambientes. Os resultados demonstraram a eficiência e a eficácia da arquitetura e das diretrizes no suporte ao processo analítico de dados espaciais.

Palavras-chave: IoT, *data warehouses* espaciais, computação em névoa, cidades inteligentes, processamento paralelo e distribuído.

LIST OF FIGURES

LIST OF SOURCE CODES

LIST OF QUERIES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface
BI	Business Intelligence
BRT	Bus Rapid Transit
CIC	*Cidade Industrial de Curitiba*
CSV	Comma-Separated Values
DoS	Denial of Service
DW	Data Warehouse
ELT	Extract, Load, and Transform
ETL	Extract, Transform, and Load
GIS	Geographic Information Systems
GLONASS	*Globalnaya Navigatsionnaya Sputnikovaya Sistema*
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GSM	Global System for Mobile
IaaS	Infrastructure-as-a-Service
ICT	Information and Communication Technology
IoST	Internet of Spatial Things
IoT	Internet of Things
IPPUC	*Instituto de Pesquisa e Planejamento Urbano de Curitiba*
JSON	JavaScript Object Notation
NoSQL	Not Only SQL
OLAP	On-line Analytical Processing
PaaS	Platform-as-a-Service
POIs	Points of Interest
RDDs	Resilient Distributed Datasets
RFID	Radio Frequency IDentification
RIT	*Rede Integrada de Transporte*
SaaS	Software-as-a-Service
SASs	Spatial Analytics Systems
SDW	Spatial Data Warehouse
SOLAP	Spatial On-Line Analytical Processing

SOLAPaaS	SOLAP as a Service
SQL	Structured Query Language
URBS	*Urbanização de Curitiba*
Wi-Fi	Wireless Fidelity

CONTENTS

CHAPTER

1

INTRODUCTION

In the last few years, the world population has been growing rapidly. From projections made by the United Nations, the population will reach 8 billion people in 2025 (FRAGA; QUEIROLO, 2018). Hence, providing the necessary infrastructure to accommodate a significant amount of people in cities can be a challenge for public authorities and companies. According to Ramaswami *et al.* (2016), the meta-principles for developing a sustainable and healthy city are "improvements in transportation, basic sanitation and energy supply", "sustainability", and "technology integration". Thus, the concept of smart cities emerged.

There are several definitions of smart cities in the literature, as discussed in (ISMAGILOVA *et al.*, 2019). In this study, the authors state that there is no agreement regarding the most accepted definition. However, there are elements and terms that are frequently highlighted, such as Information and Communication Technology (ICT), Internet of Things (IoT), interaction, sustainability, citizens, and quality of life. Amongst these definitions, we limit our scope to two that encompass the concepts of ICT and IoT. These definitions are described as follows. Peng, Nunes and Zheng (2017) describe smart cities as being "essentially built by utilising a set of advanced ICT, including smart hardware devices (e.g., wireless sensors, smart meters, smart vehicles, and smartphones), mobile networks (e.g., Wi-Fi and 3G/4G/5G network), data storage technologies (e.g., data warehouse and cloud platforms), and software applications (e.g., back-office control systems, mobile apps, and big data analytical tools)". Additionally, Yeh (2017) defines that a smart city "involves the implementation and deployment of ICT infrastructures to support social and urban growth through improving the economy, citizens involvement, and government efficiency".

A network of IoT devices can be used to provide information in a smart city. According to Patel and Patel (2016), IoT can be classified as "interconnected objects that have data regularly collected, analysed, and used to initiate action, providing a wealth of intelligence for planning, management, and decision-making". The different layers of an IoT architecture include: (i) the *smart device/sensor layer*, which is responsible for collecting data from the environment through

the employment of connection standards such as Wi-Fi, Global System for Mobile (GSM), and Bluetooth; (ii) the *network layer*, which is composed of gateways and gateway networks that support different communication protocols for sending data to the *service layer*; and (iii) the *service layer*, in which data is processed and prepared to obtain the information required by a desired application (PATEL; PATEL, 2016; ATZORI; IERA; MORABITO, 2017).

IoT technology is very important in a smart city environment. For instance, it is possible to apply this paradigm in the context of urban mobility, where sensors placed on streets and highways collect data on the number of vehicles and average vehicle speed to assist in decision-making for the improvement of urban traffic. Another IoT application scenario includes monitoring a public transport system, whose fleet contains sensors that collect data related to the number of passengers, vehicle type (e.g., buses, trams, taxis), route taken, and maximum speed, aiming to improve the existing lines. Other examples include pollution control, water consumption analyses, electric energy measurements, and tourist attraction measurements (ATZORI; IERA; MORABITO, 2017).

Data generated on these scenarios usually include spatial data, which can be represented by geometries (such as points, lines, and polygons) or combinations of them (ZEE; SCHOLTEN, 2014). For example, sensors inserted in highways, vehicles, smartphones, and wearables can generate spatial data related to the region where it is located, based on Global Navigation Satellite Systems (GNSS) technologies. These positioning systems are generally composed of ground receivers and orbital platforms, including the North American Global Positioning System (GPS) and the Russian *Globalnaya Navigatsionnaya Sputnikovaya Sistema* (GLONASS) (English: Global Navigation Satellite System) (BANSAL; CHANA; CLARKE, 2021; ELDRANDALY; ABDEL-BASSET; SHAWKY, 2019). Sensors with these technologies can generate data about objects' position and dimension, usually a point, in longitude-latitude format (RAMNATH *et al.*, 2017).

Performing analytical queries on data generated by an IoT network in a smart city can assist managers in the spatial analytics. For instance, a smart cities manager can be interested in determining "how many vehicles travelled through a given district, per day, per month". The query results can be displayed on a map according to the district, helping the manager to intuitively obtain the necessary knowledge. In order to enable the execution of this type of query, IoT data needs to be extracted, transformed, and loaded in a Spatial Data Warehouse (SDW). A SDW is a subject-oriented, integrated, time-variant, and non-volatile collection of conventional and spatial data. It provides support for the costly Spatial On-Line Analytical Processing (SOLAP) queries, which are analytical queries extended with spatial predicates (HAN; STEFANOVIC; KOPERSKI, 1998; RIVEST; BÉDARD; MARCHAND, 2001).

1.1 Motivation

The number of IoT devices is growing rapidly. According to Horwitz (2019), in 2025, more than 75 billion connected IoT devices are expected. As the number of devices increases, the volume of spatial data increases considerably, causing problems related to big data. In a smart city context, sensors all over the city can collect and transmit masses of data, such that data scale becomes increasingly big (CHEN; MAO; LIU, 2014). Nonetheless, for IoT applications that require low latency, using only environments such as clusters and cloud computing may be insufficient to deal with the delay caused by data transferring between these environments and the devices (JAVADZADEH; RAHMANI, 2020).

The fog computing paradigm emerged to solve these problems. It employs computational resources close to the edge of the network, providing data processing, storage, and distribution services (BONOMI *et al.*, 2012; TANG *et al.*, 2015; SHI; DUSTDAR, 2016; JAVADZADEH; RAHMANI, 2020). The hierarchical architecture of fog computing includes a set of fog nodes, which are low-processing devices located at the edge of the network, where data can be pre-processed and stored. Some advantages of using fog computing include the wide geographic distribution of services, sensor networks distributed on a large scale, real-time interactions, and predominance of wireless access. Another characteristic is the heterogeneity, since sensors of different natures can exist on a single network.

Data generated by IoT devices tends to be heterogeneous (LAN *et al.*, 2019), so it is necessary to implement preprocessing and data cleaning mechanisms. In the fog nodes, data lakes can be used to support this heterogeneity. According to Fang (2015), data lake is a repository that improves the capture, refinement, archival, and exploration of raw data. These repositories enable the Extract, Load, and Transform (ELT) operations, where data is transformed according to the demand for analyses. Furthermore, data lakes also support the continuous flow of data (i.e., data streaming) generated by IoT devices. Thus, using data lakes on fog nodes can benefit real-time IoT data analyses, as well as creating prediction models for machine learning (KAUR; SINGH; NAYYAR, 2020).

After the data preprocessing in the fog nodes, it is sent to a cloud computing environment, which can contain an SDW. To deal with big data, the management of the SDW can benefit from the employment of parallel and distributed data processing frameworks, such as Hadoop (SHVACHKO *et al.*, 2010) and Spark (ZAHARIA *et al.*, 2016), to reduce the complexity of the cloud. Furthermore, the SOLAP query processing can also benefit from the use of Spatial Analytics Systems (SASs), which are developed on top of those frameworks to provide extended functionalities to deal with spatial data and spatial predicates (CASTRO; CARNIEL; CIFERRI, 2020).

The challenge is to propose an IoT architecture for smart cities that encompasses cloud computing, fog computing, and frameworks for parallel and distributed data processing, and

also provides efficient support for storing SDWs and providing spatial analytics. Based on the systematic review described in Chapter 3, we did not identify any study in the literature that consider all these technologies in the same setting. That is, existing approaches investigate separately the use of SDWs (YUAN; ZHAO, 2012), data lakes (THEODOROU; DIAMANTOPOULOS, 2019), and parallel and distributed processing environments (ELDRANDALY; ABDEL-BASSET; SHAWKY, 2019; JO; JOO; LEE, 2019; WANG; ZHONG; WANG, 2019). This gap in the literature motivates the development of our work.

1.2 Objectives

Motivated by the difficulty in managing data generated by IoT devices in the context of smart cities, the primary objective of this dissertation is to propose a user-centric architecture aimed to assist smart city managers in the spatial analytics process. The architecture encompasses cloud computing, fog computing, and frameworks for parallel and distributed data processing. It also stores conventional and spatial data using data lakes and SDWs. As a result, the architecture provides support for real-time and batch processing, as well as enables the SOLAP query processing.

Based on this objective, we define the following hypothesis:

Hypothesis: It is possible to assist the smart city manager in the spatial analytics process based on data from IoT devices, using fog and cloud computing environments, data lakes and SDWs stored in a parallel and distributed processing environment.

Through the investigation of this hypothesis, we introduce the following contributions:

- Identification of particularities related to spatial data generated by IoT devices in smart cities and technologies that can be used to assist spatial analytics in this context. These technologies refer to distributed data processing frameworks, fog and cloud computing solutions, and systems for storing and querying conventional and spatial data in data lakes and SDWs

- Proposal of an architecture aimed to help smart cities managers and residents to perform spatial analytics. The architecture is composed of four layers: (i) terminal layer, which consists of a network of IoT devices; (ii) fog layer, which contains fog nodes and stores data lakes for real-time processing; (iii) cloud layer, which relies on a distributed parallel processing environment that contains SDWs for batch processing; and (iv) analytics tools layer, which is composed of tools to assist in data analyses and visualization.

- Definition of a set of guidelines to assist in the implementation of the proposed architecture. These user-centric guidelines describe important characteristics and functionalities of each

layer that should be considered to employ the architecture. They also highlight related technologies and tools.

- Validation of the efficacy and effectiveness of the proposed architecture through two case studies. The first one refers to a real dataset that contains vehicle traffic data collected from sensors distributed in the municipality of Aarhus, Denmark (ALI; GAO; MILEO, 2015). The second case study uses real data related to the public transportation of the city of Curitiba, Brazil, where sensors placed on buses collect data related to the spatial position[1]. Depending on the demand of these case studies, we carried out real-time and batch SOLAP queries against data stored in a data lake and in an SDW.

Preliminary results related to these contributions generated the papers described as follows:

- SANTOS, J. P. C.; CIFERRI, C. D. d. A. Processamento de consultas analíticas espaciais sobre dados de cidades inteligentes. In: **35th Brazilian Symposium on Databases — Companion Proceedings.** Rio de Janeiro, 2020. p. 37–43.
- SANTOS, J. P. C.; CASTRO, J. P. d. C.; CIFERRI, C. D. d. A. SOLAP Query Processing over IoT Networks in Smart Cities: A Novel Architecture. In: **Proceedings of XXI GeoInfo – Brazilian Symposium in Geoinformatics**. São José dos Campos, Brazil: INPE, 2020. p. 118–129.
- SANTOS, J. P. C.; CASTRO, J. P. C.; AGUIAR, C. D. Combining Fog and Cloud Computing to Support Spatial Analytics in Smart Cities. In: **Journal of Information and Data Management.** 2021.

The proposed architecture and guidelines, although focusing on the smart cities context, can be applied to any context where spatial data is generated by a network of IoT devices, such as spatial data generated by smartphones, which are made available through social media. Therefore, the architecture and the guidelines can be used as a basis for the emergence of new solutions in the academic and market.

[1] <http://dadosabertos.c3sl.ufpr.br/curitibaurbs/>

1.3 Dissertation Organisation

The remaining chapters of this dissertation are organised as follows:

- Chapter 2 contextualises the technical background needed to understand this dissertation. We describe concepts related to IoT, fog computing, and data preprocessing using data lakes. We also detail representations of spatial data, spatial relationships, and spatial queries. Other concepts described in this chapter are related to SDWs and parallel and distributed processing environments.
- Chapter 3 describes the systematic review, in which the planning and conduction phases are discussed.
- Chapter 4 presents the proposed architecture and the guidelines for implementing the architecture. We also describe some pipelines based on the architecture, which are implementations of the architecture using technologies available on the market.
- Chapter 5 presents two case studies based on the proposed architecture and guidelines. The first case study focus on the execution of three different categories of spatial queries, e.g., spatial queries with topological predicates, spatial queries with metric relationships, and spatial queries with type-dependent operations; The second case study focus on three different environments that spatial queries can be executed: fog computing environments, cloud computing environments or both.
- Chapter 6 describes the concluding remarks of this dissertation, highlighting the main contributions and future work.

CHAPTER

2

TECHNICAL BACKGROUND

In this chapter, we describe the technical background used to develop our work. Section 2.1 introduces concepts related to the Internet of Things, as well as discusses the application of IoT devices in the smart cities context. Section 2.2 contextualises fog computing and its architectural components. Section 2.3 details data preprocessing and storage techniques, such as ELT and data lake. Section 2.4 details the basic definitions of spatial data, spatial relationships, and spatial queries. Section 2.5 describes the use of repositories to aid spatial analytics in smart cities, such as data lakes and spatial data warehouses. Section 2.6 details parallel and distributed processing and storage environments, including cloud computing. Finally, Section 2.7 describes the final considerations about the chapter.

2.1 Internet of things

Atzori, Iera and Morabito (2017) define Internet of Things (IoT) as "a conceptual framework that leverages on the availability of heterogeneous devices and interconnection solutions, as well as augmented physical objects providing a shared information base on global scale, to support the design of applications involving at the same virtual level both people and representations of objects". According to Patel and Patel (2016), an IoT architecture consists of the following different layers of technologies that support IoT: (i) sensor layer, which is made up of smart objects integrated with sensors; (ii) network layer, where the objects are interconnected using connections protocols; (iii) management service layer, where information is obtained through data processing and analytics; and (iv) application layer, which covers IoT applications for smart cities and smart health, among others.

Several devices were developed to meet IoT specifications. These devices collect data from sensors, which can be applied in various scenarios, such as national security (AFZAL *et al.*, 2019), multimedia environments (ALVI *et al.*, 2015), robotics (SIMOENS; DRAGONE; SAFFIOTTI, 2018), medicine (MAHDAVINEJAD *et al.*, 2018), and wearable devices (HIREMATH;

YANG; MANKODIYA, 2015). In smart cities, IoT devices can generate spatial data related to vehicle trajectory, energy, and public transport (ELDRANDALY; ABDEL-BASSET; SHAWKY, 2019), for instance.

There are several IoT devices with different characteristics, such as Radio Frequency IDentification (RFID) tags (JING *et al.*, 2018), Arduino (JO; BALOCH, 2017), humidity and pressure sensors (KOTSEV *et al.*, 2016), vehicle sensors (Wu He; Gongjun Yan; Li Da Xu, 2014), smart traffic lights (MIZ; HAHANOV, 2014), and pollution sensors (KAMILARIS; PITSILLIDES, 2014). These devices are arranged in a network using communication protocols such as IEEE 802.11 (known as Wireless Fidelity (Wi-Fi)), 3G/4G/5G, and Bluetooth (GUBBI *et al.*, 2013; ATZORI; IERA; MORABITO, 2017). In an IoT network, spatial data, which is obtained through the position, speed, and estimated transmission time of Global Navigation Satellite System (GNSS) satellites, can be processed close to the network edge. The following section details how this can be done.

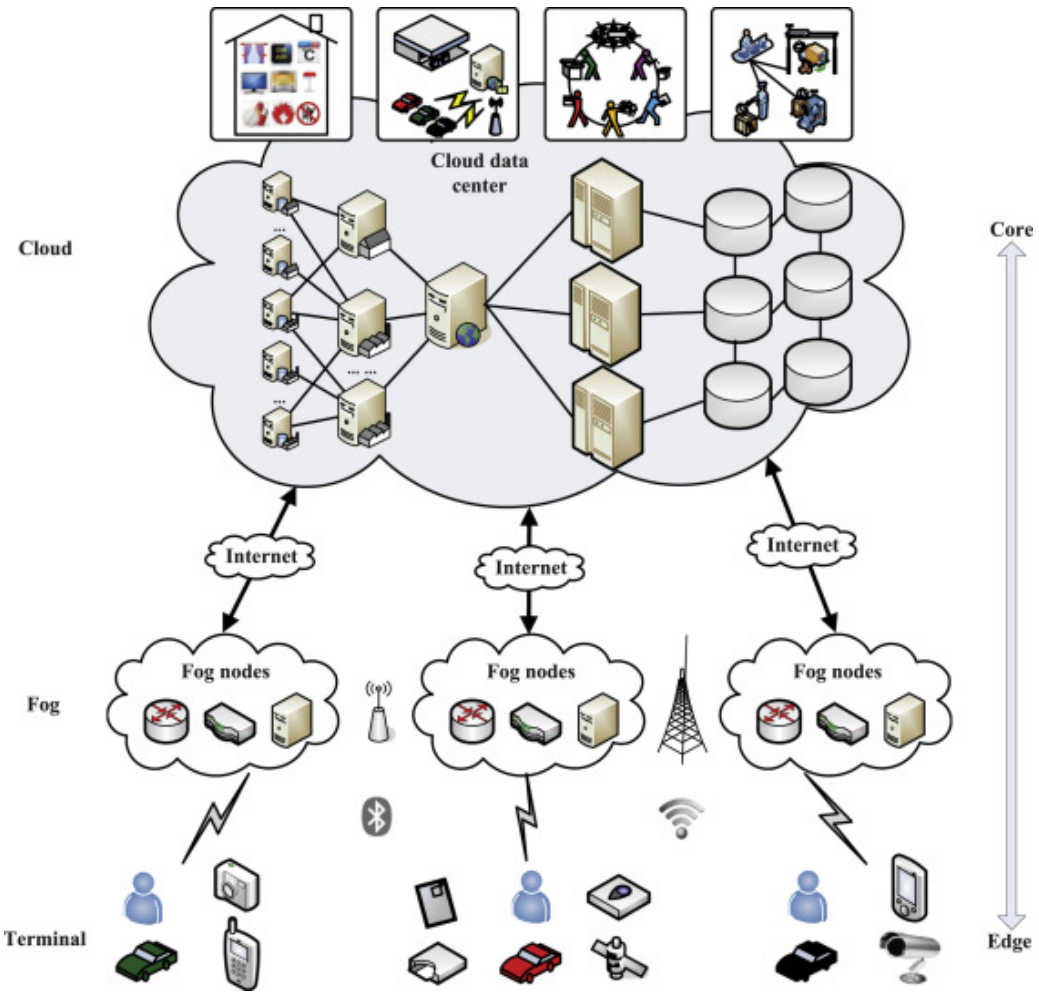
2.2 Fog computing

After collecting the data, it is necessary to define where this data will be processed. For IoT applications that require low latency, centralised processing can be inefficient due to the delay caused by transferring data from the IoT devices to the cloud (DASTJERDI; BUYYA, 2016). Thus, a paradigm called edge computing emerged. This paradigm uses computational resources close to the edge of the network for local storage and preliminary data processing (SHI; DUSTDAR, 2016; DASTJERDI; BUYYA, 2016).

However, computing resources on devices at the edge of the network may have low processing power or memory capacity. Thus, Bonomi *et al.* (2012) proposed the concept of fog computing, which "is a highly virtualised platform that provides compute, storage, and networking services between end devices and traditional cloud computing data centres, typically, but not exclusively located at the edge of the network". Some authors (SHI; DUSTDAR, 2016; HU *et al.*, 2017; DASTJERDI; BUYYA, 2016) adopt this concept as "edge computing". In this dissertation, we adopt the term "fog computing" and consider "edge computing" as synonymous.

Figure 1 shows the hierarchical architecture of fog computing. It is composed of three layers: (i) terminal layer, which consists of several IoT devices that generate and send data to the upper layers; (ii) fog layer, which is located on the edge of the network and is composed of many fog nodes that have capabilities to compute, transmit, and temporarily store the received sensed data; and (iii) cloud layer, which consists of multiple high-performance servers and storage devices. The cloud layer also provides various application services, such as analytics, data visualisation, and Business Intelligence (BI) (HU *et al.*, 2017). Some advantages of using fog computing include the wide geographic distribution of services, large-scale distributed sensor networks, real-time interactions, and the use of wireless networks (BONOMI *et al.*, 2012; DASTJERDI; BUYYA, 2016).

Figure 1 – The hierarchical architecture of fog computing.



Source: Hu *et al.* (2017).

Data generated by IoT devices tends to be heterogeneous (LAN *et al.*, 2019; HU *et al.*, 2017). Therefore, it is necessary to define methods for preprocessing and storing the data in the fog node (and by extension, in cloud and/or local environments). In the following section, we describe these methods, including ELT and data lake.

2.3 Data Preprocessing

Data generated by IoT devices is heterogeneous (LAN *et al.*, 2019; HU *et al.*, 2017). For example, sensors with different technologies can generate similar data but with different semantics. In addition, various standardization issues such as incorrect and incomplete data can occur. Thus, providing data quality is important to decide how data can be used in data analytics (SHEHAB; BADAWY; ALI, 2021). According to Han, Kamber and Pei (2012), the main steps for data preprocessing include:

- **Data cleaning.** "Cleaning" the data by filling in missing values, identifying or removing outliers, and resolving inconsistencies;
- **Data integration.** Merging the same data present in multiple sources into a coherent and integrated version of the data;
- **Data reduction.** Providing a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data;
- **Data transformation.** Transforming or consolidating data for mining, by smoothing noisy data, aggregating, normalizing, and discretising.

These steps are part of the Extract, Transform, and Load (ETL) process, where data is (i) extracted from the sources; (ii) transported to a data staging area, where the processing takes place; (iii) cleaned, integrated, reduced, transformed, and loaded in a Data Warehouse (DW) (VASSILIADIS, 2009). However, for handling large amount of data, the ETL process is inefficient, as the transformation step takes up a large amount of time and computational resources (FANG, 2015). Thereby, the concept of Extract, Load, and Transform (ELT) emerged, where data is transformed on demand after being loaded (WAAS *et al.*, 2013).

In the ELT process, data is usually loaded into a data lake, which is, according to Fang (2015), "a massive data repository based on low-cost technologies that improves the capture, refinement, archival, and exploration of raw data within an enterprise". Processing data on a data lake allows the manipulation of heterogeneous data from a programming language on demand. Thus, libraries offered by languages used in data science, such as Python and R, can be used to query this data (WAAS *et al.*, 2013; FANG, 2015). Table 1 indicates the main differences between the traditional ETL process with data warehouse and the ELT process with data lake.

Table 1 – Comparing ETL with data warehouse and ELT with data lake.

	ETL with data warehouse	**ELT with data lake**
Schema	Requires work at the beginning of the process, but offers performance security and integration.	Works well for data types where data value is not known.
Scale	Large data volumes at moderate cost.	Extreme data volumes at low cost.
Access	Data accessed through standard SQL.	Data accessed through programs created by developers or other query engines.
Query	SQL.	Various.
Data	Cleaned and transformed.	Raw.

Source: Adapted from Fang (2015).

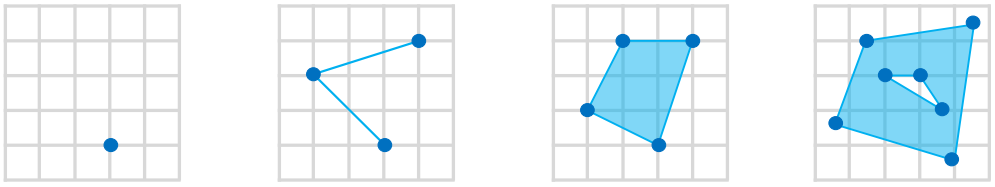
IoT devices usually generate spatial data in addition to conventional data. The following section describes concepts related to spatial data, including how it is represented and manipulated to allow spatial queries.

2.4 Spatial data

Spatial data (or geographic data), according to Güting (1994), is a component that represent the geometry of spatial objects. Considering the vector format, this data can be categorised as having simple or complex geometry. Figure 2 illustrates simple geometry spatial data types: point (Figure 2a), which represents an exact location in the space; line (Figure 2b), which represents a set of connected points; and polygon (Figure 2c), which represents a set of connected lines to form a closed polygonal chain. A polygon can contain holes (Figure 2d).

Figure 2 – Simple geometry spatial data representations

(a) Point (b) Line (c) Basic polygon (d) Polygon

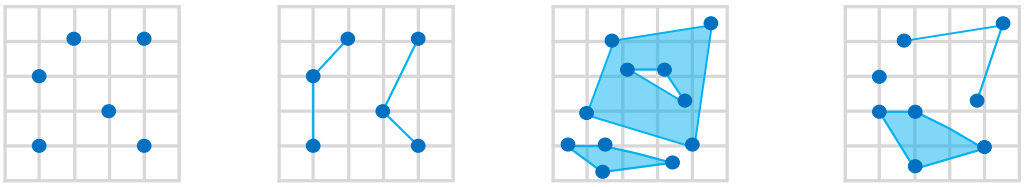


Source: Elaborated by the author.

A complex geometry spatial data represents a collection of simple geometry spatial data. Figure 3 illustrates collections of points (Figure 3a), lines (Figure 3b), polygons (Figure 3c), and a heterogeneous collection containing points, lines, and polygons (Figure 3d).

Figure 3 – Complex geometry spatial data representations

(a) Multipoints (b) Multilines (c) Multipolygons (d) Geometry Collection



Source: Elaborated by the author.

According to Egenhofer (1989), vector spatial data can be related through spatial relationships, which can be classified as metric, topological, and directional. Metric relationships exploit the existence of measurements, such as distances, where mathematical operations such as the Euclidean distance can be used. Directional relationships are qualitative spatial relations that describe how an object or a region is placed with regard to other objects or regions (e.g.,“left”, “above”, “beside”, and “south”) (EGENHOFER, 1989; PEUQUET; CI-XIANG, 1987). Finally, topological relationships describe qualitative properties that characterise the relative position of spatial objects (e.g., “contains” and “intersects”) (EGENHOFER, 1989).

Spatial data can also be represented in the raster format (BONHAM-CARTER, 1994), which is a bitmap image, such as satellite images and scanned maps, that represents a grid of

pixels. These pixels can be analysed for proper conversion to vectors using image processing and pattern recognition (CONGALTON, 1997; GÜTING, 1994). Raster formats are used, for example, in meteorological applications, where there is constant change in images, such as cloud formation and wind direction. However, we base our work on the vector format to represent spatial data.

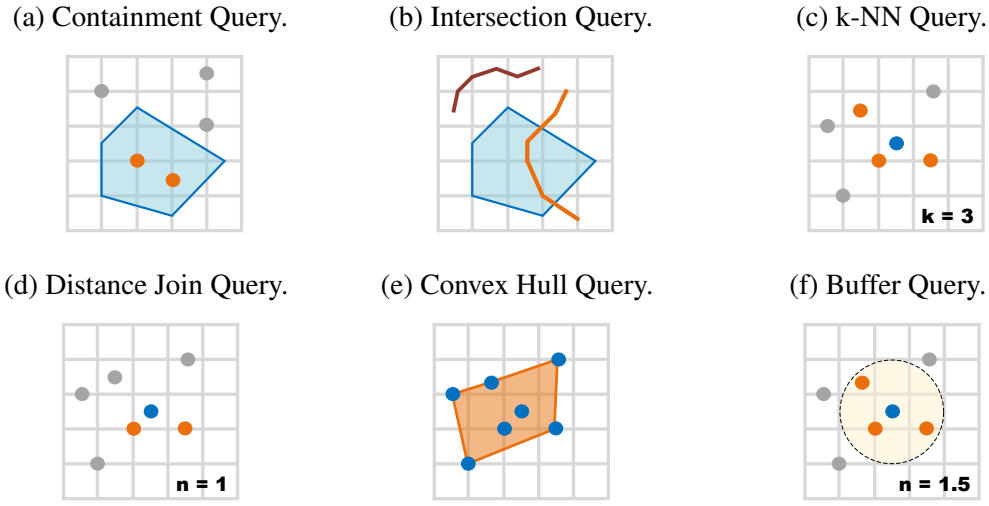
2.4.1 Spatial queries

Spatial queries are queries whose at least one of the predicates involves a spatial relationship. There are different types of spatial queries that relate objects that are embedded in d-dimensional Euclidean space (E^d), according to the definitions of Gaede and Günther (1998). The types of queries of interest to our work are described as follows.

- **Intersection Query** or **Region Query** or **Overlap Query**. Given an object o' with spatial extent, find all objects o having at least one point in common with o'.
- **Containment Query.** Given an object o' with spatial extent, find all objects o enclosed by o'.
- **k-Nearest-Neighbor Query.** Given an object o' with spatial extent, find the k-nearest objects o from o'. Common distance functions for points include the Euclidean and the Manhattan distances.
- **Convex hull Query.** Given a collection R of spatial objects, find the smallest convex object o' that encloses all objects o in R.
- **Buffer Query.** Given an object o' with spatial extent, find all objects o enclosed or intersected in a zone that is drawn around o' within a specified distance n of o'.
- **Spatial Join.** Given two collections R and S of spatial objects and a spatial predicate θ, find all pairs of objects $(o,o') \in R \times S$ where $\theta(o.G, o'G)$ evaluates to true, being $o.G$ and $o'.G$ spatial extents. As for the spatial predicate θ, many spatial relationships can be used, including intersects, contains, and distance.

Figure 4 illustrates these queries. In Figures 4a and 4b, o' is a polygon, while in the remaining figures, o' is one or a set of points. In Figure 4f, the buffer area is represented in yellow. In particular, Figure 4d depicts a spatial join with distance as a spatial predicate. Therefore, the spatial join distance is defined as: given two collections R e S of spatial objects, finds all pairs of objects $(o,o') \in R \times S$ where the distance between o and o' is less than or equal to a n value.

Figure 4 – Examples of spatial queries with topological predicates. Object o' is represented in blue and objects o are represented in orange.



Source: Elaborated by the author.

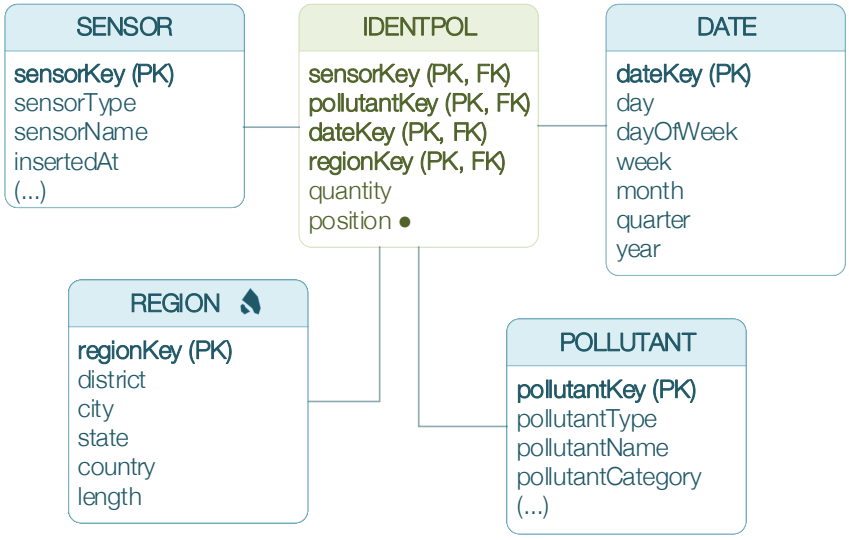
2.5 Spatial data warehousing

According to Malinowski and Zimnyi (2008), Bimonte, Tchounikine and Miquel (2005), Han, Stefanovic and Koperski (1998), a spatial data warehouse (SDW) "is a subject-oriented, integrated, time-variant, and non-volatile collection of spatial and non-spatial data in support of management's decision-making process". To enable complex analyses and visualisation, the data in a warehouse is typically modelled multidimensionally through a star schema, which contains a large central table (the *fact* table) and a set of smaller satellite tables (the *dimension* tables) displayed in a radial pattern around the fact table (HAN; STEFANOVIC; KOPERSKI, 1998; KRIPPENDORF; Il-Yeol Song, 1997).

A fact table contains measures of interest and foreign keys related to the dimension tables. The combination of the foreign keys is the primary key of the fact table. The measures can be classified as: (i) numeric measure, which is usually a conventional numeric data; and (ii) spatial measure, which contains one or a collection of pointers to spatial objects (HAN; STEFANOVIC; KOPERSKI, 1998). A dimension table consists of a primary key and several attributes. These attributes may relate to each other through hierarchies of attributes. These hierarchies are based on the granularity of data, and specify that an attribute of a higher level of granularity can be generated from an attribute of a lower level of granularity (HARINARAYAN; RAJARAMAN; ULLMAN, 1996). The hierarchy of attributes is represented by the operator $\preceq$.

Figure 5 illustrates a star schema based on a network of sensors aimed to investigate aspects related to pollution considering each sensor, data, region, and pollutant. The table IdentPol is a fact table that contains the conventional measure quantity of pollutants and the spatial measure position, which is represented by a point (•). The tables Date, Sensor, Region, and Pollutant are dimension tables.

Figure 5 – Example of a star scheme based on a network of sensors in a smart city that measure the amount of pollutants and their geographic positions.



Source: Elaborated by the author.

According to Han, Stefanovic and Koperski (1998), there are three cases for modelling dimensions in a spatial data cube:

1. **Non-spatial dimension.** A dimension containing only non-spatial data. In Figure 5, the dimensional table Pollutant contains non-spatial attributes whose hierarchies are non-spatial, such as *pollutantType* and *pollutantCategory*.

2. **Spatial-to-non-spatial dimension.** A dimension where the lowest attribute hierarchy is spatial, but the highest hierarchies are not spatial. For instance, a *state* is spatially represented as a polygon. However, each state can be generalised to a non-spatial value, such as an alphanumeric value, like a telephone area code, and its further generalization is non-spatial, and thus playing a similar role as a non-spatial dimension.

3. **Spatial-to-spatial dimension.** A dimension where all levels of the attribute hierarchy is spatial. In Figure 5, Region is a dimension in which all attributes are spatial (and it is identified by an icon ( ), which represents a MultiPolygon (VAISMAN; ZIMÁNYI, 2014)). The granularity of these data is determined from the analysis of the topological relationships between these objects (MATEUS *et al.*, 2016).

The spatial attributes in an SDW can be manipulated through Spatial On-line Analytical Processing (SOLAP) queries (RIVEST; BÉDARD; MARCHAND, 2001). The spatial queries described in Section 2.4 can be used in the analytical process just like traditional On-line Analytical Processing (OLAP) operations. An example of a query in a SOLAP environment, using the star schema illustrated in Figure 5, is "to identify the average quantity of pollutants

in districts that are located in the state of São Paulo (state is represented by a window query, i.e., a rectangle), grouped by day, by month". A *roll-up* (or *drill-down*) operation can be done to increase (or decrease) the time granularity, and a containment query can be used to get the average quantity of pollutants collected by the sensors that are contained in the districts of the state of São Paulo.

Other possible analytical operations that may be present in a SOLAP query include *slice and dice*, *pivot*, and *drill-across*. The *slice and dice* operation restricts data being analysed to one of its subsets, such as when a given value or set of values is used to select data related to an attribute. The *pivot* operation reorients the data multidimensional view, offering different perspectives of the same data. Finally, the *drill-across* operation compares numeric measurements from different fact tables that share at least one common dimension table (VAISMAN; ZIMÁNYI, 2014; CIFERRI *et al.*, 2013).

2.6 Parallel and Distributed Processing Systems

An SDW can hold a large amount of data, so it can be placed in an environment that allows for efficient processing and storage. Thus, parallel and distributed processing frameworks available in computer clusters or cloud computing environments can be used. These solutions are aimed to simplify the interaction between the infrastructure and the user. In this section, we describe two Apache solutions that deal with parallel and distributed processing (Section 2.6.1). We also detail platforms as a service available in cloud environments (Section 2.6.2).

2.6.1 Hadoop and Spark

Hadoop is a parallel and distributed processing framework that implements the MapReduce programming model, developed by Dean and Ghemawat (2008). This model incorporates two functions. The map function processes input data in the form of key-value pairs and transform this data in intermediate outputs in the form of key-value pairs. The reduce function processes the generated intermediate output, groups values associated with a key, and generates a set of key-value pairs as output. It is possible to execute several map and reduce functions in sequence. The map and reduce functions can require a large amount of read and write data to disk, causing iterative processing overhead. (SINGH; KHAMPARIA; LUHACH, 2019).

To overcome this limitation, Zaharia *et al.* (2010) introduced Spark, which uses fault-tolerant collections of objects that can be partitioned into a cluster, causing data to be manipulated in parallel (ZAHARIA *et al.*, 2016). These collections, called Resilient Distributed Datasets (RDDs), allow intermediate outputs in the main memory, reducing the amount of disk accesses when compared to Hadoop (SINGH; KHAMPARIA; LUHACH, 2019). RDDs support several types of functions, including map and reduce. Another difference between Spark and Hadoop is that Spark uses directed acyclic graphs, where vertices represent the RDDs and the edges

represent the operation to be applied on the RDDs. RDDs. The use of these graphs guarantees to Spark a better global optimization than Hadoop. (ZAHARIA *et al.*, 2016).

Hadoop and Spark usually rely on distributed file systems, such as the Hadoop Distributed File System (HDFS). According to Shvachko *et al.* (2010), HDFS "is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications". Any file is divided in blocks, which are distributed and replicated across the nodes. By default, each block of the file is replicated across three nodes. The HDFS architecture has two types of nodes: `NameNode`, or master node, which has the metadata about file blocks location; and `DataNode`, or worker node, where data is stored. If the `NameNode` fails, there is the possibility of using a *backup*, which is a secondary `NameNode`. During the execution of the application, communication is performed as follows. First, the application accesses the `nameNode` and uses the stored metadata to identify the location of the blocks that contain data from the file. Second, the application accesses the `dataNodes` to perform read and write operations on the blocks.

According to García-García *et al.* (2017), Hadoop and Spark have limitations related to spatial objects manipulation, as these frameworks do not have indexing mechanisms that allow selective access to specific regions of spatial data. Therefore, several solutions have emerged that extend these frameworks, aiming at the optimised execution of spatial queries.

These solutions, called Spatial Analytics Systems (SASs), incorporate spatial data manipulation in Hadoop and Spark frameworks, providing optimised techniques for processing spatial queries. Castro, Carniel and Ciferri (2019) introduces a user-centric view of several existing Hadoop and Spark-based SAS in the literature, such as Hadoop-GIS (AJI *et al.*, 2013), SpatialHadoop (ELDAWY; MOKBEL, 2015), SpatialSpark (YOU; ZHANG; GRUENWALD, 2015) and GeoSpark (YU; WU; SARWAT, 2015). This user-centric view provides a detailed description of several SASs considering aspects related to general characteristics, support for spatial data handling aspects, and support for aspects inherent to distributed systems.

2.6.2 Cloud computing

Cloud computing, according to Wang *et al.* (2010), "is a set of network enabled services, providing scalable, quality of services guaranteed, normally personalised, inexpensive computing infrastructures on demand, which could be accessed in a simple and pervasive way". There are several players on the market that offer cloud computing services, such as Microsoft Azure[1], Google Cloud[2], and Amazon Web Services[3]. There are several services offered by these players, such as Infrastructure-as-a-Service (IaaS), which provides hardware virtually, Software-as-a-Service (SaaS), which offers software that can be used via the Internet, and Platform-as-a-Service (PaaS), which offers a complete set of technologies required to develop and to operate SaaS

[1] <http://azure.microsoft.com>
[2] <http://cloud.google.com>
[3] <http://aws.amazon.com>

(BEIMBORN; MILETZKI; WENZEL, 2011).

Parallel and distributed frameworks can be disposed in an IaaS, as well as PaaS with complete data processing and storage solutions. Therefore, SDWs can benefit from these services. (DEHNE *et al.*, 2015). Mateus *et al.* (2016) describe the concept of SOLAP as a Service (SOLAPaaS), which defines a SOLAP environment as a cloud service. This service allows a SOLAP server to run on virtual machines in the cloud. In the IoT context, a SOLAPaaS service can be used for spatial analytics on data generated by IoT devices located at the edge of the network.

2.7 Final Remarks

In this chapter, we describe the technical background required for this dissertation. Thereby, we evaluated the following topics: (i) IoT devices and its applications; (ii) fog computing, where we describe the hierarchical architecture; (iii) data preprocessing, with definitions of ELT with data lake and ETL with data warehouse; (iv) spatial data manipulation, where we describe concepts about the definition of spatial data types, spatial relationships and spatial queries; (v) spatial data warehousing, where we describe the concept of multidimensional modelling of conventional and spatial data; and (vi) parallel and distributed processing systems, including frameworks available in computer clusters or cloud computing environment.

In the next chapter (Chapter 3), we describe the systematic review process we did in the context of this dissertation, which encompasses the concepts described in this chapter to identify approaches in the literature that are related to the objective of our work.

CHAPTER

3

SYSTEMATIC REVIEW

In this chapter, we describe a systematic review conducted for the development of this project. According to Nakagawa *et al.* (2017), the systematic review process is divided into three phases: planning, conduction, and publishing. Section 3.1 describes the planning phase, which defines the objectives, research questions, keywords, and search strings. Section 3.2 shows the selection phase, in which searches are performed and the studies that are most related to the project are selected based on search questions. Section 3.3 details the conduction phase, in which the selected in the selection phase are synthesised. Finally, Section 3.4 describes the final considerations about the chapter.

3.1 Planning Phase

Planning is the first phase of a systematic review. It consists of defining the research questions (Section 3.1.1), the search engines (Section 3.1.2), the keywords and search strings (Section 3.1.3), and the selection criteria (Section 3.1.4).

3.1.1 Research questions

The research questions that we defined for this systematic review are:

RQ1: How large amounts of spatial data (big data) are generated by IoT devices, and how can this data be manipulated?

RQ2: How can spatial data generated by IoT devices be managed in an environment that contains a spatial data warehouse?

RQ3: How can spatial queries help spatial analytics in the smart cities context?

RQ4: How can a data lake be used in fog computing environments?

RQ5: Are there studies that relate SDW, data lake, and fog computing in the smart cities context to support spatial analytics?

3.1.2 Search engines

The search engines we used for the systematic review were defined based on criteria such as the number of studies available and the broad scope. Based on these criteria, we choose IEEE Xplore Digital Library[1], ACM Digital Library[2], Elsevier ScienceDirect[3]. At Elsevier ScienceDirect, searches were limited only to open access publications. DBLP[4] was not considered because publications indexed in this source, in general, are already indexed in the chosen search engines (BATISTA *et al.*, 2018). To manage the results, the Elsevier Mendeley[5] tool was used, which has a plug-in compatible with all major web browsers, improving the indexing of these results for this systematic review.

3.1.3 Keywords and search strings

For the definition of the search strings, it is necessary to define the keywords based on the search questions. Therefore, we defined the following keywords: *internet of things, iot, spatial data, big data; smart cities, smart city; fog computing, edge computing; data lake; spatial data warehouse, sdw.* We elaborate the search strings based on these keywords, using logical operations, such as `AND` and `OR`. The search strings used in this systematic review are listed as follows:

- `"spatial data" AND ("internet of things" OR iot) AND "big data";`
- `(SDW OR "spatial data warehouse") AND ("internet of things" OR iot OR "smart cities");`
- `"data lake" AND ("fog computing" OR "edge computing");`

Regarding Elsevier ScienceDirect, we do not consider the acronym "SDW", as most of the results returned referred to *Spin Density Wave*, a physics term related to the conductivity of materials. This term is not related to the subject of this study. This ambiguity did not often occur in the other chosen search engines. We chose not to include "cloud computing" as a search string, as cloud computing can be considered an architectural component of a fog computing environment (BONOMI *et al.*, 2012; BONOMI *et al.*, 2014; HU *et al.*, 2017; DASTJERDI; BUYYA, 2016). As cloud computing allows the implementation of parallel and distributed processing frameworks as-a service, we also do not include these frameworks (and by extension, SASs), as search strings.

[1] <https://ieeexplore.ieee.org>
[2] <https://dl.acm.org>
[3] <https://www.sciencedirect.com/>
[4] <https://dblp.org/>
[5] <https://www.mendeley.com>

3.1.4 Selection criteria

The selection criteria consist of defining which studies will be included or excluded, the selection procedures for choosing studies, and the quality criteria of the journals and conferences. We chose not to include articles in Portuguese, due to limited results in search engines.

3.1.4.1 Inclusion criteria

- Studies that answer at least one of the research questions;
- Studies in English;
- Studies with open access;
- Studies recommended for inclusion by a specialist;
- Studies between 2016 and 2021 (more recent studies).

3.1.4.2 Exclusion criteria

- Studies that do not meet any of the research questions;
- Studies not available in English;
- Studies that are not available for free or complete;
- Studies where the author of this dissertation is an author.

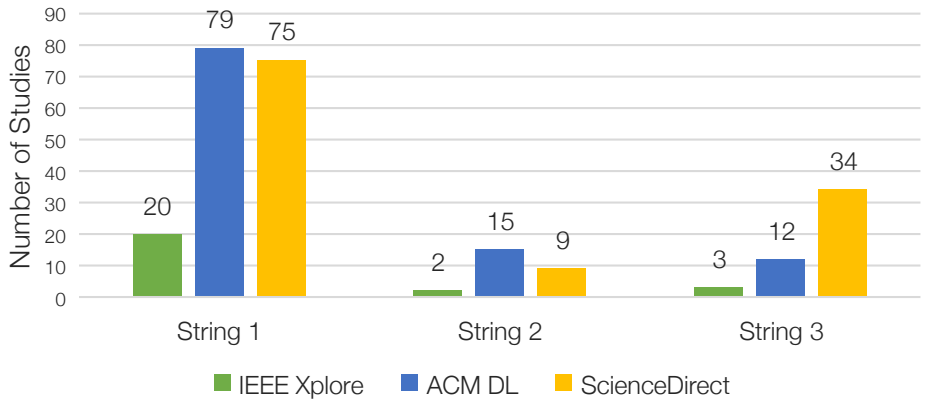
3.1.4.3 Selection procedures

1. **Initial selection**: From the keywords, we identified the search strings used in search engines. From reading the title and abstract, we identified whether the study meets the criteria. If so, we used it in the final selection.
2. **Final selection**: At this stage, we performed a complete reading of the studies that meet the previous criteria, as well as checked whether the study still meets the inclusion criteria.
3. **Obtaining information**: Using Mendeley for study selection, we extracted information in order to synthesise the content of results.
4. **Quality criteria**: The journals and conferences in which the papers were published were checked using these criteria, like the reputation of the journal or conference where the study was published, number of citations, author impact factor and writing quality.

3.2 Conduction Phase

Based on the search strings, our search retrieved 335 studies. Figure 6 illustrates the number of studies found by the search engines that we defined. There was a large return of studies that were identified from the string that relates spatial data to the internet of things and big data (String 1). However, most of these studies refer to trajectory prediction using GPS data obtained from IoT devices, and do not list in detail how the data is processed or analysed after extraction. These studies also do not introduce architectures for IoT data management, presenting the spatial context only as an example of data that can be generated by IoT devices.

Figure 6 – Number of studies found by search engines, using the search strings listed.



String 1: "spatial data" **AND** ("internet of things" **OR** iot) **AND** "big data"
String 2: (SDW **OR** "spatial data warehouse") **AND** ("internet of things" **OR** iot **OR** "smart cities")
String 3: "data lake" **AND** ("fog computing" **OR** "edge computing")

Source: Elaborated by the author.

Using the selection criteria that we defined in Section 3.1.4, we selected five studies, four from IEEE Xplore (Eldrandaly, Abdel-Basset and Shawky (2019), Jo, Joo and Lee (2019), Liu *et al.* (2016) and Theodorou and Diamantopoulos (2019)) and one from Elsevier ScienceDirect (Wang, Zhong and Wang (2019)). No studies were found in ACM DL that meet the selection criteria.

Additionally, a study was included based on a specialist indication: Yuan and Zhao (2012). This study does not meet the research criterion related to the period from 2016 to 2021, but meets the research question RQ2.

3.3 Reporting phase

In this phase, we classify the studies in groups in order to answer the research questions defined in Section 3.1.1. We detail these groups in the following sections: Section 3.3.1, which group studies related to IoT, spatial data and big data; Section 3.3.2, which contains studies concerning SDW, IoT, and smart cities; Section 3.3.3, which encompasses studies related to data lake and fog computing; and Section 3.3.4, which includes studies that encompass all topics related to this systematic review.

3.3.1 IoT, spatial data and big data

The objective of this group is to identify studies that relate large amounts of spatial data generation (big data) with IoT devices, thus addressing the research question RQ1. Therefore, we identified the following studies: Eldrandaly, Abdel-Basset and Shawky (2019), Jo, Joo and Lee (2019) and Wang, Zhong and Wang (2019).

Eldrandaly, Abdel-Basset and Shawky (2019) define Internet of Spatial Things (IoST), a model that expands the traditional IoT architecture to handle spatial data using fog computing concepts. The authors also present a literature review, where they describe IoT classifications in other areas (such as medical, multimedia, robotics and nanotechnology) and the relationships of IoT with spatial data.

Jo, Joo and Lee (2019) detail the development of a platform that aims to "efficiently storing, extracting, processing, and analysing geospatial big data". The architecture of this platform is supported by several layers, including ETL and data analysis layers. The authors conducted tests on a Hadoop environment to identify areas in Seoul, South Korea, that need more hospitals to treat patients with thermal injuries.

Wang, Zhong and Wang (2019) describe a cloud integrated Geographic Information Systems (GIS) platform architecture designed to process and analyse spatio-temporal big data. This architecture provides support for data streaming, enabling real-time data analysis using Spark. After that, the processed data is sent to a GIS for analysis. The authors used flight-related data for the case study, where this data was processed in a Spark cluster, with real-time monitoring. According to the authors, the study makes a contribution to smart cities systems, by offering solutions for spatial resources' management, city operation, and city maintenance.

3.3.2 Spatial data warehouses, IoT, and smart cities

The objective of this group is to identify studies that relate IoT and smart cities with an SDWing environment, thus addressing the research questions RQ2 and RQ3. We identified the following studies: Liu *et al.* (2016) and Yuan and Zhao (2012).

Liu *et al.* (2016) describe some methods for collecting heterogeneous data obtained from numerous sources in a smart city, enabling the integration of these data in analytical environments (like spatial data warehouses). The spatial data sources described by the authors are generic, making only the question RQ3 be covered by this study.

Yuan and Zhao (2012) present SDWIT, a framework that expands a spatial data warehousing environment in IoT context. The ETL process is performed between the device layer and the storage layer. Unlike a traditional SDW, SDWIT updates the data in real-time (or close to it). The framework is not related to smart cities in its conception, making only the question RQ2 be covered by this study.

3.3.3 Fog computing and data lakes

The objective of this group is to identify studies that relate fog computing with data lakes, thus addressing the research question RQ4. We identified one study, Theodorou and Diamantopoulos (2019). This study introduces a novel architecture for ELT on edge gateways, where the data is stored in a lightweight repository, acting as a data lake. This repository, called by the authors as *data lagoon*, contains raw data from IoT devices, which is retrieved using streaming protocols. After that, the data is transformed and integrated with other data. The processed data can be sent to the cloud or to other services at the edge, such as the *local actuation* component, which deals with the IoT devices management.

3.3.4 Spatial data warehouses, IoT, fog computing, data lake, and smart cities

The objective of this group is to identify studies with similar objectives to this dissertation, thus addressing the research question RQ5. From the search criteria defined in Section 3.1.4, no studies were found that relate fog computing, SDW, and data lake in an architecture that aims to assist managers in the spatial analytics in the smart cities context.

3.4 Final remarks

In this chapter, we contextualise the systematic review conducted for this dissertation. Table 2 lists the topics covered in this systematic review, by study. Among the studies we selected, Eldrandaly, Abdel-Basset and Shawky (2019) and Wang, Zhong and Wang (2019) cover several topics related to this dissertation. However, these studies do not foresee the use of spatial data warehousing environments. On the other hand, the study of Yuan and Zhao (2012) aims to create an environment based on the use of a SDW in the IoT context. Nonetheless, this study does not include fog computing and smart cities concepts in the proposed architecture. Theodorou and Diamantopoulos (2019) proposes to use a data lake in a fog node, but they have not detailed how the data can be queried over the data lake. The architecture proposed in this study also does not foresee spatial data usage.

According to the systematic review, it was not possible to find a study that meets the research question RQ5, that relate fog computing, SDW and data lake in an architecture that aims to assist managers in decision-making in smart cities' context. We fill this gap in the literature, as shown in the last line of Table 2.

Table 2 – Comparison between selected studies in the systematic review by topics.

Study	Spatial Data	SDW	IoT	Smart Cities	Fog Computing	Cloud Computing
Eldrandaly, Abdel-Basset and Shawky (2019)	✓	✗	✓	✓	✓	✓
Jo, Joo and Lee (2019)	✓	✗	✓	✓	✗	✓
Wang, Zhong and Wang (2019)	✓	✗	✗	✓	✗	✓
Liu *et al.* (2016)	✓	✓	✗	✓	✗	✗
Yuan and Zhao (2012)	✗	✓	✓	✗	✗	✗
Theodorou and Diamantopoulos (2019)	✗	✗	✓	✗	✓	✓
Our study	✓	✓	✓	✓	✓	✓

Source: Elaborated by the author.

In the next chapter (Chapter 4), we describe the novel architecture that we propose, which aims to help smart city managers for decision-making based on data generated by IoT devices, using fog computing and a spatial data warehousing environment in the cloud.

CHAPTER

4

PROPOSED ARCHITECTURE

In this chapter, we propose an architecture aimed to process and store data generated by IoT devices in smart cities. The proposed architecture is divided into four layers: terminal, fog, cloud, and analytics tools. We also introduce a set of guidelines to aid smart cities manages to make decisions about the implementation of each layer according to the requirements of the smart city application. Based on these guidelines, we show examples of pipelines, which instantiate the components of the architecture with technologies. This chapter is organised as follows. Section 4.1 describes the architecture and its layers. Section 4.2 introduces the guidelines. Section 4.3 shows examples of pipelines. Finally, Section 4.4 describes the final considerations about the chapter.

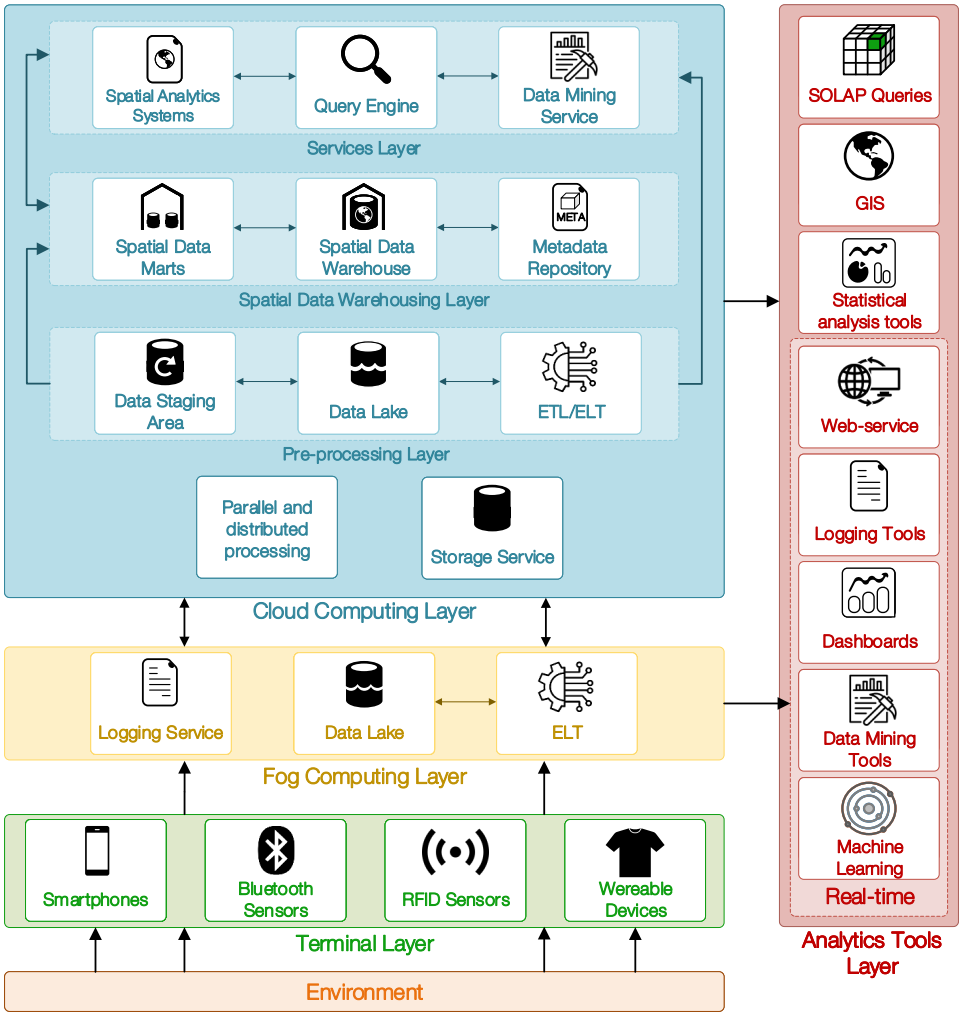
4.1 The Proposed Architecture

In this section, we propose a novel architecture for collecting and analysing, data from IoT devices in smart cities. The proposed architecture is depicted in Figure 7. It achieves its goals through the employment of four different layers: (i) the terminal layer; (ii) the fog computing layer; (iii) the cloud computing layer, which is divided into three sub-layers: preprocessing, spatial data warehousing, and services; and (iv) the analytics tools layer. In the following sections, we discuss each layer.

4.1.1 Terminal layer

The terminal layer consists of a network of IoT devices, which are interconnected by using technologies such as RFID, GPS, and network communication standards, such as Ethernet, Bluetooth, and Wi-Fi. These devices are available in many parts of a smart city, such as weather stations, traffic lights, and public transportation. The devices are aimed to collect spatial and conventional data.

Figure 7 – Proposed architecture overview, which encompasses cloud computing, fog computing, and frameworks for parallel and distributed data processing.



Source: Elaborated by the author.

4.1.2 Fog computing layer

Data collected by terminal layer devices are sent to receivers in the fog computing layer. These receivers, called fog nodes, can be limited with regard to data processing and storage. However, by being located close to the network edge, they are in an optimal position to allow the execution of real-time data analytics and ELT operations. This is due to the low latency in communication between the terminal layer devices and these nodes.

The collected data from the sensors is temporarily stored in a data lake, which can be used for real-time queries. These queries, which can be executed via Application Programming Interface (API), can be done entirely in the data lake, using only the data from the IoT sensors, or in a hybrid way, with additional queries in the cloud to obtain the required attributes for the query in the fog node. Finally, a logging service is made available for monitoring the sensor's health and to the fog node itself.

4.1.3 Cloud computing layer

After the data goes through the fog computing layer, it is sent to the cloud computing layer. This layer contains parallel and distributed processing and storage. Due to the scalable nature inherent to cloud computing environments, the infrastructure can be improved in terms of storage and processing according to the queries demand from clients. This layer is divided into three sub-layers: preprocessing, spatial data warehousing, and services. These sub-layers are described as follows.

- **Preprocessing layer.** In this layer, data is prepared so that it can be stored in the SDW. The ETL/ELT component is responsible for extracting, transforming and loading data into the SDW. The ETL process uses the data staging area for intermediate data storage before it is loaded. In a big data context, the ELT process can be used, where data is extracted and loaded into the data lake, to be processed and stored in the SDW when the information needs to be obtained.

- **Spatial data warehousing layer.** This layer contains an environment that contains an SDW. The SDW, which is contained in the storage service, is organised multidimensionally to enable SOLAP queries. In addition to the SDW, there can be many spatial data marts, which can represent a small SDW with a limited scope when compared to the main SDW. This layer also encompasses a metadata repository, which includes information about the SDW, its location and structure, and the semantic data description disposed in the SDW.

- **Services layer.** This layer offers services to aid in the spatial analytics. The query engine run queries (including SOLAP queries) using a query language such as Structured Query Language (SQL). If the search engine does not support spatial queries, a SAS can be used to execute these queries. A data mining service can be used to create prediction models based on data available in the storage service. These models can help analytics based on the hidden patterns' discovery in the data.

4.1.4 Analytics Tools layer

After the data is made available in a SDW, the smart city manager can use tools that enable strategic decision-making. These tools, which can be available both in the fog and in the cloud, as well as external clients, are responsible for enabling visualisation and querying based on the data collected and processed. Some of these tools include:

- **SOLAP queries tools.** Allow the user to analyse data using complex multidimensional views, enabling the data analysis under different aggregation levels and data types (conventional and/or spatial).

- **GIS tools.** Allow the user to analyse spatial data and visualise the results on maps.

- **Statistical analysis tools.** Allow the user to analyse the data using statistical methods.
- **Web-service.** Allow the user to extract raw data using APIs.
- **Logging Tools.** Allow the user to monitor system from logs.
- **Dashboards.** Allow the user to arrange the data through graphs and tables.
- **Data Mining Tools.** Allow the user to create and manage prediction models.
- **Reporting Tools.** Allow the user to produce reports based on data.

4.2 Guidelines

In this section, we propose a set of guidelines to aid smart cities managers in the process of implementing the proposed architecture to support spatial analytics in smart cities. Due to the different characteristics of each smart city, managers should choose the appropriated hints provided by the guidelines according to these characteristics. Therefore, it is not mandatory to follow every guideline in its completeness. Thus, a concise yet general description of each guideline is provided, allowing further specialisation based on the requirements imposed by each smart city application. The guidelines are named according to the layer that they should be applied. For instance, Guidelines T1 and T2 are guidelines from the terminal layer.

4.2.1 Terminal layer

Guideline T1. Investigating the IoT devices heterogeneity. A smart city manager should consider the IoT network heterogeneity, considering the communication protocol between the devices and the fog and/or cloud, the capacity for processing and storing data and the data format collected in the terminal layer. To this end, the manager can use the findings introduced in Lan *et al.* (2019), which describe "a general ontology-based resource description model of IoT devices to provide a consistent view of heterogeneous sensing devices for IoT applications in the cloud" (LAN *et al.*, 2019, p. 44199).

Guideline T2. Deploying IoT devices. IoT devices must be deployed in the terminal layer considering the communication protocols supported by each sensor and the coverage of each device. A smart city manager must also consider the communication compatibility between these devices and the fog nodes. Some investigations found in the literature can be used to assist in the deployment of these devices. We indicate the work of Alablani and Alenazi (2020), which introduces an algorithm for sensor distribution and sink placement called EDTD-SC. This algorithm uses triangulation and clustering techniques to find optimal locations to improve sensor coverage over a smart city.

4.2.2 Fog computing layer

Guideline F1. Distributing fog nodes across the fog computing layer. After the IoT devices disposition over the terminal layer, a smart city manager must define which devices should be used as fog nodes. For instance, some approaches in the literature use Raspberry Pi computers[1], which are small single-boarded computers, as fog nodes, using containerization over these resource-limited devices (BELLAVISTA; ZANNI, 2017; XU; ZHANG, 2019). Because Raspberry Pi computers are low cost and support many communication protocols, they are a viable choice to an IoT network heterogeneous nature. Communication between the fog nodes and the cloud computing layer can be carried out using 4G/5G or Wi-Fi protocols. Each fog node uses the Docker container technology[2] for creating containers for each application available in a fog node (i.e., ETL and real-time data analytics). We indicate the work of Prado *et al.* (2020), which presents the challenges and opportunities on the schedule of smart containers in the Cloud-Fog-IoT interfaces.

Guideline F2. Securing the connection between IoT devices and fog nodes. A smart city manager must be concerned with the data flow security between the IoT devices and the fog nodes, as sensitive information may be transmitted. For instance, at the terminal layer, some security threats include signal jamming between IoT devices and Denial of Service (DoS) attacks (PUTHAL *et al.*, 2019). At the fog computing layer, security threats include phishing attacks, infected code injection, session hijacking, and distributed DoS (PUTHAL *et al.*, 2019; RAUF; SHAIKH; SHAH, 2018). To deal with these issues, smart cities managers can take decisions using as a basis the work of Ni *et al.* (2018). In this work, the authors discuss security-related challenges in fog-assisted IoT applications. The authors also review state-of-the-art solutions to address security and privacy issues in a fog environment deployed in an IoT network.

Guideline F3. Enabling the ELT process. To enable ELT in the fog computing layer, a smart city manager must model the ELT process, defining which data will be extracted and loaded from the IoT devices, which transformations must be executed after loading and the data flow schema design. A solution for modelling and workflow monitoring for ELT is Apache Airflow[3], which is an open-source platform that uses directed acyclic graphs for authoring, scheduling, and monitoring workflows. We indicate that the tasks be written using the Python programming language, since it is natively supported by Airflow, which also provides integration with Hadoop, Spark, and several cloud platforms. For data streaming, an example is Apache Kafka[4], which is a distributed data store optimised for ingesting and processing streaming data in real-time. Kafka supports APIs, which can be used to send data to cloud for storage and parallel and distributed processing.

[1] <https://www.raspberrypi.org/>
[2] <https://www.docker.com/>
[3] <http://airflow.apache.org/>
[4] <http://kafka.apache.org>

Guideline F4. Modelling the data lake. After enabling the ELT routines, the smart city manager must define the data lake modelling, with focus on structure and metadata management in the data lake. As the data lake is in an infrastructure with limited processing power and storage, the manager must be concerned with the data format, data querying, data loading and the availability time of data in the data lake. Sawadogo and Darmont (2021) describe different approaches to the data lake design. The authors discuss data ingestion (e.g., tools like Kafka and protocols like HTTP), data storage (e.g., Not Only SQL (NoSQL) and HDFS), data processing and data access. The paper does not discuss the deploying of the data lake in the fog computing layer. In this sense, the work of Rani *et al.* (2021), which is a systematic review, can be used to define a storage-as-a-service in the fog computing layer.

Guideline F5. Deploying the data lake environment. After modelling the data lake, a smart city manager should define the environment to implement it. Multiple data management systems, like NoSQL databases or parallel and distributed storage services can be used to deploy the data lake. An example is Couchbase Server Lite[5], which is a document-oriented database developed for embedded devices. For the data lake deployment that stores key-value structured data, RedisEdge[6] may be used. Both Couchbase Lite and RedisEdge can synchronise with NoSQL servers located at the cloud through an API. For integration with parallel and distributed storage services, Wang *et al.* (2020) presents Apache IoTDB[7], which is "an IoT native database with high performance for data management and analysis". IoTDB can be integrated with HDFS, Spark, and Flink[8]. IoTDB uses TsFiles, a data file format for time-series data storage. TsFiles can be directly synchronised with an active IoTDB instance on the cloud and can be persisted on HDFS. As stated in Guideline F1, storage services can be operated by containers inserted in the fog node. Taking into account the data heterogeneity from IoT devices (Guideline T1), polyglot persistence (MEDVEDEV *et al.*, 2016; SADALAGE; FOWLER, 2012) can be used in a fog node network since it takes into consideration the node processing and physical storage, as well as the queries that should execute in each node.

Guideline F6. Enabling analytical data processing. In a smart city, IoT devices generate data constantly, enabling real-time data analysis. Some approaches can be used, such as data querying over the data lake on demand or prediction models generation for machine learning operations. The following guidelines detail these approaches.

Guideline F6.1. **Enabling analytical queries on data lake.** Based on the storage services chosen according to Guideline F5 and their related services, different query mechanisms can be used. For example, Couchbase Lite allows queries with selection, projection, grouping, and joining using an interface compatible with numerous programming languages. Apache IoTDB

[5] <https://www.couchbase.com/products/lite>
[6] <https://redislabs.com/redis-enterprise/more/redis-edge/>
[7] <https://iotdb.apache.org/>
[8] <https://flink.apache.org/>

uses an SQL-like interface for queries. These query mechanisms use APIs to integrate with the query visualisation services. The temporal scope of the queries that can be issued against the data lake are restricted to the data lake model defined in Guideline F4.

Guideline F6.2. **Enabling data mining.** In a fog node, data are loaded constantly, enabling real-time data analytics through the use of techniques like stream data analytics, machine learning, and deep learning (PRABHU *et al.*, 2020). To this end, a smart cities manager can use the findings presented in the work of Kaur, Singh and Nayyar (2020), where the authors describe machine learning applications that can be used in fog computing scenarios, using supervised learning, distributed decision trees, and clustering methods for big data. Another work to be used as a basis is described in (SAVAGLIO; FORTINO, 2021), which presents a methodology that enables IoT data mining in fog/cloud deployments. The knowledge discovery process can be performed on the fog nodes, enabling high efficiency, responsiveness, and scalability.

4.2.3 Cloud computing layer

Guideline C1. Choosing a parallel and distributed processing service. A smart city manager should select the most appropriate parallel and distributed data processing framework and distributed file system. The frameworks offered by Apache, such as Hadoop and Spark, are widespread in the market. The main PaaS in the market offer these frameworks natively, including Microsoft Azure, Google Cloud, and Amazon Web Services. These frameworks can also run in a private cloud, using tools like OpenStack. Furthermore, these frameworks can also be employed on a local cluster.

Guideline C2. Enabling spatial queries. The SDW application should process SOLAP queries efficiently. Therefore, a smart city manager must select a query engine that is able to completely fulfil the SDW application requirements and that is compatible with the data processing and storage services chosen in Guideline C1. While PaaS in the market support spatial queries[9,10], Hadoop and Spark must use a SAS to support these queries. Because there are several SASs available in the literature with different characteristics and capabilities, choosing the most appropriate SAS can become considerably challenging. Thus, smart cities managers should use as a basis of choice the state-of-the-art user-centric comparison of existing SASs described by Castro, Carniel and Ciferri (2020). For a system-centric view of SASs, the work of Pandey *et al.* (2018) should be referred, as it compares several SASs in the literature based on their query processing performance.

Guideline C3. Enabling the ELT process. A smart city manager should model the ELT process and the tools necessary for the execution. As well as in Guideline F3, workflow monitoring tools, like Apache Airflow, can be used in a cloud environment. Other tools, like Pentaho Data

[9] <https://docs.aws.amazon.com/redshift/latest/dg/geospatial-functions.html>
[10] <https://cloud.google.com/bigquery/docs/gis-data>

Integration[11], Azure Data Factory[12], Google Cloud Dataflow[13] and AWS Glue[14], can be used with the parallel and distributed processing frameworks indicated in Guideline C1.

Guideline C4. Modelling and deploying the data lake environment. As described in Guideline F4, the work of Sawadogo and Darmont (2021) describes different approaches to data lake design and metadata modelling. The authors also discuss two main approaches to combine a data lake and a data warehouse in a global data management system: (i) data lake sourcing a data warehouse; and (ii) data lake within a data warehouse. The storage service must be compatible with the parallel and distributed service chosen in Guideline C1.

Guideline C5. Multidimensional data modelling contained at data lake. A smart city manager should perform multidimensional modelling at the logical and the physical level. At the logical level, the manager should use of schemas such as the star schema, the snowflake schema, and the Geographic Hybrid Star Schema (SIQUEIRA *et al.*, 2012). At the physical level, the manager should use structures such as star join bitmap indexes and materialised views. For a conventional star schema modelling, the work of (KIMBALL *et al.*, 2011) can be used. In a spatial data context, Vaisman and Zimányi (2014) present design discussions to model an SDW and also introduce the concept of trajectory data warehouses, which can be appropriate for IoT devices that collect data from sensors installed in moving objects, for instance, in vehicles. The work of Mateus *et al.* (2016) details novel schemas design for an SDW deployed in a cloud environment. The authors also evaluate the SOLAP query processing performance with the employment of a cloud-based spatial index.

Guideline C6. Configuring the SDW to process SOLAP queries. After choosing the appropriate SAS, a smart city manager must configure the SDW environment in the cloud computing layer to process SOLAP queries. The smart city manager must consider the parallel and distributed storage chosen in Guideline C1 and the periodicity in which data should be extracted from the fog computing layer, as well as carefully specify data distribution over the SDW. These SOLAP services must support APIs and GIS applications in order to visualise the SOLAP queries results.

Guideline C7. Ensuring secure spatial queries processing. Since cloud computing environments can be virtually accessed from anywhere, smart cities managers should be concerned with security issues related to cloud applications. In the same way that fog nodes are subject to security threats like phishing and DDoS attacks (Guideline F2), a cloud environment also has security concerns (ALMORSY; GRUNDY; MÜLLER, 2010; BALANI; VAROL, 2020). Solutions for issues related to security in a cloud computing environment are presented in Singh, Jeong and Park (2016). Another way to ensure confidentiality is the encryption of data stored in the SDW.

[11] <https://pentaho.com>
[12] <https://azure.microsoft.com/services/data-factory/>
[13] <https://cloud.google.com/dataflow>
[14] <https://aws.amazon.com/glue/>

Data encryption should be done carefully so that it does not compromise the SDW application performance. To this end, we indicate the work of Mateus *et al.* (2016), which proposes an encryption methodology for a cloud DW stored according to the star schema, considering the capability of processing analytical queries over the encrypted DW.

Guideline C8. Enabling data mining, machine learning, and deep learning operations. The smart city manager can use machine learning and deep learning for analytics. To this end, the manager can use the suggestions from: (i) Soomro *et al.* (2019), which describes, from a systematic review, several works that present data mining solutions for thematic domains like climate science, energy management and transport; and (ii) Adi *et al.* (2020), which reviews how IoT-generated data are processed for machine learning. The authors classify many analytics techniques for IoT, like descriptive, predictive, prescriptive, and adaptive, and the applications of these techniques.

4.2.4 Analytics Tools layer

Guideline A1. Selecting appropriate analytics tools. A smart city manager may select analytical tools to aid in analytics. There are several tools that are open source, free, and are consolidated in the market and in the literature. For instance, for a GIS tool, QGIS[15] can be chosen, and for dashboards and statistical analysis, environments like R[16] and Conda[17] can be employed. If the manager chooses a PaaS as suggested in Guideline C1, additional analysis tools can be offered. For example, in an AWS environment, Amazon QuickSight[18] can be used for statistical analysis and dashboards.

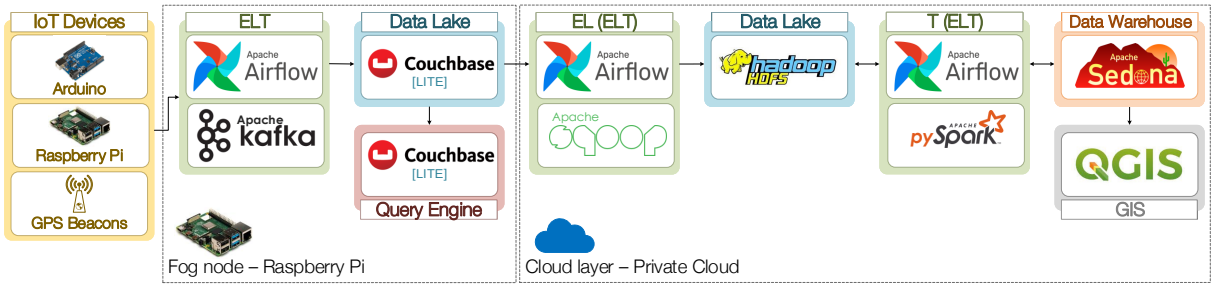
[15] <https://www.qgis.org>
[16] <https://www.r-project.org/>
[17] <https://docs.conda.io/en/latest/>
[18] <https://aws.amazon.com/quicksight/>

4.3 Pipelines

In this section, we exemplify some pipelines of the architecture proposed in Section 4.1 based on the guidelines presented in Section 4.2. We use both open source and paid technologies.

In Figure 8, we show a pipeline for a traditional data warehousing application, associating each step with its corresponding guidelines. IoT devices, like Arduino, Raspberry Pi, and GPS beacons are distributed at the Terminal layer, which are collecting data from the environment (Guideline T2). These devices send data to the fog computing layer, where the nodes are Raspberry Pi devices (Guidelines F1 and F2). The data format generated by the IoT devices includes JavaScript Object Notation (JSON) files, Comma-Separated Values (CSV) files, and binary files. Data from these files are extracted, loaded, and transformed into a data lake using Apache Airflow and Apache Kafka (Guideline F3). After defining the data lake modelling (Guideline F4), the data is loaded into Couchbase Lite, a document-oriented, embedded device compatible NoSQL (Guideline F5). Queries using data persisted in the data lake can be made using the Couchbase Lite query service (Guideline F6.1).

Figure 8 – Pipeline for a traditional data warehousing application using open source technologies.



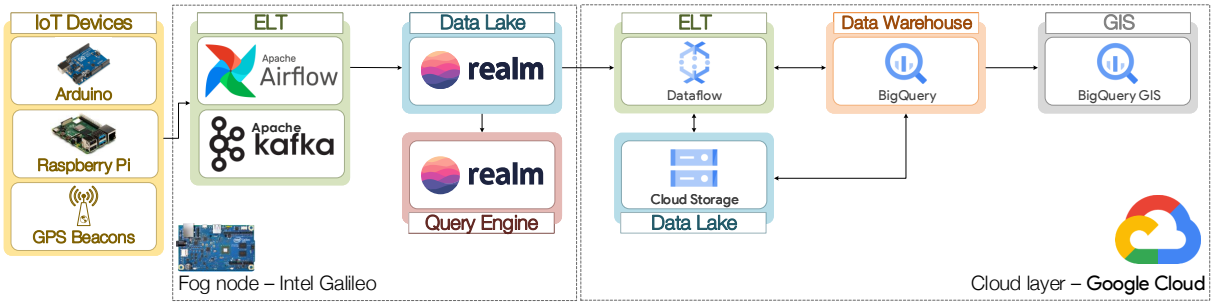
Source: Elaborated by the author.

In the cloud computing layer, after choosing Apache Spark with Apache Sedona[19] and HDFS as the frameworks for parallel and distributed processing and storage, arranged in a private cloud (Guidelines C1 and C2), the data from the fog nodes are extracted and loaded into the data lake contained in the cloud using Apache Airflow and Apache Sqoop[20] (Guideline C3). The data lake is stored in the HDFS, and the transformation of the data to be used in the data warehousing environment is performed using Apache Airflow with PySpark[21] (Guideline C4). After defining the multidimensional model, (Guideline C5), SOLAP queries are performed using SedonaSQL (Guideline C6). SOLAP queries visualisation is carried out using QGIS (Guideline A1).

[19] <https://sedona.apache.org/>
[20] <https://sqoop.apache.org/>
[21] <https://spark.apache.org/docs/latest/api/python/>

The pipeline illustrated in Figure 8 can be adapted to use cloud technologies available as PaaS, as depicted in the pipeline of Figure 9. The fog computing layer is implemented similarly to the previous pipeline, with Realm[22] as the data lake and query engine (Guidelines from F1 to F6.1). The cloud computing layer uses technologies from Google Cloud (Guidelines C1 and C2). The ELT process is done using Dataflow, and the data from the Data Lake is stored using Cloud Storage (Guidelines C3 and C4) and data from the Data Warehouse is stored in BigQuery. SOLAP queries are performed using BigQuery (Guideline C6) and SOLAP queries visualisation can be done using BigQuery GIS (Guideline A1).

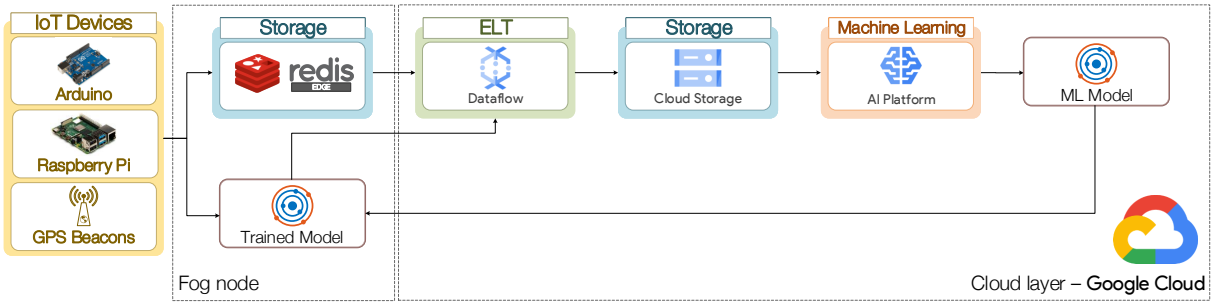
Figure 9 – Pipeline for a traditional data warehousing application using Google Cloud Services.



Source: Elaborated by the author.

Our proposed architecture allows machine learning applications over the collected data, based on Guidelines F6.2 and C8. Figure 10 illustrates an example of a pipeline using a Google Cloud Service environment. In the fog computing layer, data is stored in RedisEdge. The ELT process is done using Dataflow and the data is stored in Cloud Storage. The machine learning model is created using AI Platform. Once the model is created, it is sent to the fog computing layer to enable predictions directly in the fog.

Figure 10 – Pipeline for a machine learning application using Google Cloud Services.



Source: Elaborated by the author.

22 <https://realm.io/>

4.4 Final Remarks

In this chapter, we propose a novel architecture for collecting and analysing data generated by IoT devices in smart cities. The architecture is divided into four layers, which include terminal, fog, cloud, and analytics tools layers. To assist smart city managers in spatial analytics and implementing the proposed architecture, we propose a series of guidelines for modelling, processing, storing and analysing data. Finally, we describe examples of pipelines that employ technologies that can be used based on the proposed architecture. In the next chapter, we will present two case studies that use the architecture described in this chapter.

CHAPTER

5

CASE STUDIES

In this chapter, we describe two case studies that illustrate the use of proposed architecture. We define the requirements of two spatial applications, whose objective is to process data collected from multiple IoT devices in the context of smart cities. These case studies are aimed to validate the efficacy and effectiveness of the proposed architecture. Focusing on details regarding the performance and reliability of the architecture is out of the scope of this dissertation. Section 5.1 describes the use of a dataset related to vehicle traffic and parking, which was collected using IoT sensors arranged in a smart city. Section 5.2 describes a real-time pipeline based on data collected by IoT sensors contained in a smart city bus fleet. Finally, Section 5.3 describes the final considerations about the chapter.

5.1 Vehicle Traffic Analyses

In this case study, we use data related to vehicle traffic collected by sensors placed on streets and highways. The collected data encompasses the number of vehicles and the average vehicle speed, which can provide valuable insights for the improvement of urban traffic. We use a dataset provided by Ali, Gao and Mileo (2015), which contains vehicle traffic data observed between two street sensors and vehicle count data observed in sensors placed in parking garages. These sensors are distributed in the municipality of Aarhus, Denmark. The dataset, which is publicly available in the authors' website[1], contains both conventional (e.g., distance in metres between the sensors, type of road, etc.) and spatial data (e.g., the sensors locations, represented by points) referring to the period from February to June 2014. These sensors generate data every five minutes.

As the dataset only provides data regarding the sensor location (i.e., points), we extended it with new information to enrich the analyses performed in our spatial application. To this end, we use road (i.e., lines) and city (i.e., polygons) data obtained by Geofabrik[2] from OpenStreetMap,

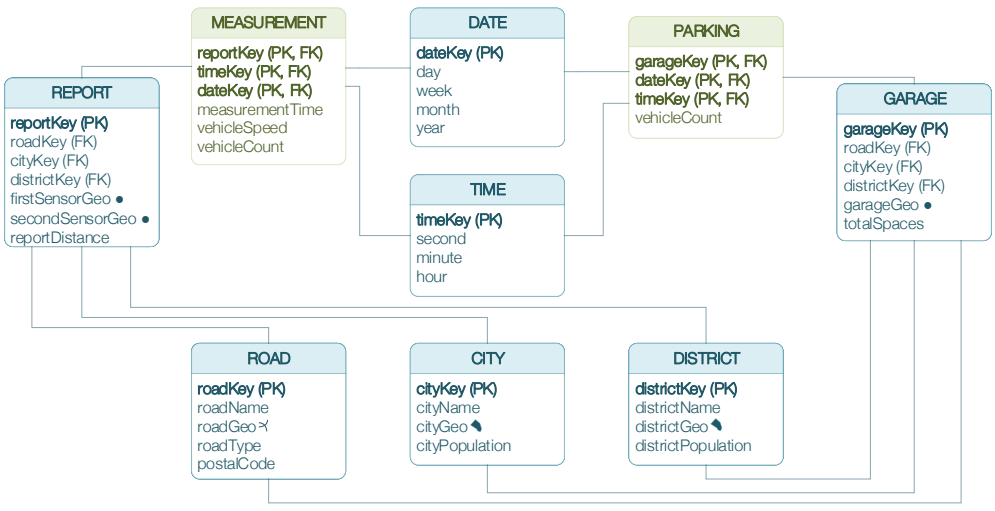
[1] <http://iot.ee.surrey.ac.uk:8080/>
[2] <https://www.geofabrik.de/>

and statistical district data obtained from OpenDataDK[3]. We guarantee the spatial relationship between the data in the sense that a road intersects with sensors, a district contains multiple roads, and a city contains several districts.

The requirements imposed by the SDW application are described as follows. The application should be deployed in the cloud and should communicate with a SAS to process its queries. Furthermore, data handled by the application should be stored in an SDW designed according to the logical schema depicted in Figure 11, which should also be located in the cloud. Another application requirement is that it should support different spatial queries types based on the definitions of Gaede and Günther (1998), such as spatial join, containment, and k-nearest neighbour queries. The application should also provide good performance results. Finally, it is important to highlight that the developers who are going to implement the application have some previous SQL programming language knowledge.

The schema illustrated in Figure 11 represents data related to sensors and parking lots scattered in a smart city that collect data on vehicle quantity and average speed. The spatial attributes in the snowflake schema are represented as described by Vaisman and Zimányi (2014). There are seven dimension tables in the SDW: (i) Date and Time, storing the moment in which a measurement occurred; (ii) Report, storing the distance between the two street sensors that performed the measurement and their geographic locations; (iii) Garage, storing the geographic location of the sensors placed in parking garages and the total spaces available for vehicles to park there; and (iv) Road, District, and City, storing the geographic locations associated with the report and with the garage. The dimension tables are linked through two fact tables: (i) Measurement, which stores, for each measurement, the vehicle count, the measurement time, and the vehicle speed; and (ii) Parking, which stores the vehicle count.

Figure 11 – Logical schema of the SDW proposed to support the case study.



Source: Elaborated by the author.

[3] <https://www.opendata.dk/city-of-aarhus/statistikdistrikter>

According to the proposed architecture (Section 4.1) and guidelines (Section 4.2), the case study application should be implemented as follows: (i) use of Apache Airflow to perform the ETL process; (ii) storage of the SDW data in HDFS as the application requires data storage in the cloud; and (iii) selection of Sedona as the SAS to process spatial queries aimed to support spatial analytics, as it complies with the application's requirements regarding performance and spatial queries, as well as supports the SQL programming language through the use of SedonaSQL (PANDEY *et al.*, 2018; CASTRO; CARNIEL; CIFERRI, 2020).

In Section 5.1.1, we describe aspects related to data loading, including the preprocessing of the data. Once the IoT sensors data loading process into the SDW is complete, smart cities managers are able to execute different analyses using SedonaSQL. We define some query examples that address key points of the application's requirements, which are divided into three different categories: (i) spatial queries with a topological predicate (Section 5.1.2); (ii) spatial queries with metric relationships (Section 5.1.3); and (iii) spatial queries with type-dependent operations (Section 5.1.4). These queries may be employed by a smart cities manager to support spatial analytics. We employ QGIS[4] to visualise the results of the queries.

5.1.1 Data Loading into the Cloud Layer

The dataset used in this case study consists of 449 reports and 8 garages. Data from these reports and garages are stored in CSV files. To be loaded into the SDW, the data must go through an ELT process in the fog layer (Guideline F5). Thus, Apache Airflow should be employed to: (i) extract the data from the CSV files; (ii) perform transformations to arrange the data according to the logical schema depicted in Figure 11; and (iii) load the data into HDFS for later use by Sedona. To accurately simulate the fog layer, Airflow was executed in a Docker container.

Furthermore, in order to employ SedonaSQL for processing spatial queries over the SDW stored in HDFS, it is necessary to load its tables into structures called DataFrames. These structures, which resemble relational tables, do not transform the textual representations of the spatial data into spatial objects by default. Therefore, it is necessary to carry out the transformations. SedonaSQL provides a function to convert Well-Known Text (WKT) representations into spatial objects. An example of using this function during the loading process of the Report table is detailed in Query 1.

Query 1 – Using a function to convert WKT representations in SedonaSQL.

```
1: SELECT reportID, roadID, districtID, cityID, reportDistance,
2:        ST_GeomFromWKT(firstSensorGeo) AS firstSensorGeo,
3:        ST_GeomFromWKT(secondSensorGeo) AS secondSensorGeo
4: FROM sensor
```

[4] <https://qgis.org/>

5.1.2 Spatial Queries with Topological Predicates

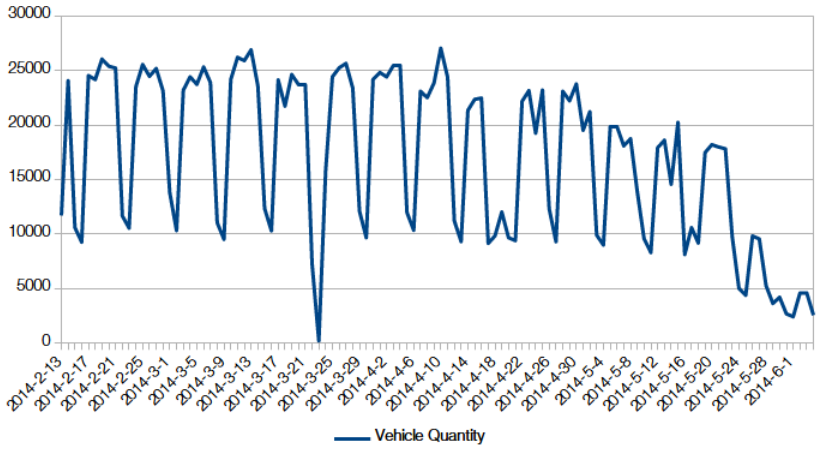
In this section, we describe queries that use topological predicates. The two topological predicates used in our case study analyses are the containment and the intersection spatial join. The containment query, listed in Query 2, returns the quantity of vehicles that travelled in Aarhus University/Community Hospital district grouped by day and month. This query can also be classified as a slice analytical operation, since it returns a subset of the data considering a fixed value.

Query 2 – Return the quantity of vehicles that travelled in Aarhus University/Community Hospital district grouped by day and month.

```
 1: SELECT day, month, SUM(vehicleCount)
 2: FROM measurement, date, report, district, road
 3: WHERE ST_Contains(districtGeo, roadGeo)
 4:     AND ST_Intersects(roadGeo, ST_MakeLine(firstSensorGeo,
 5:         secondSensorGeo))
 6:     AND measurement.reportID = report.reportID
 7:     AND measurement.dateID = date.dateID
 8:     AND report.districtID = district.districtID
 9:     AND report.roadID = road.roadID
10:     AND district.name = 'Universitetet/Kommunehospitalet'
11: GROUP BY day, month ORDER BY day, month
```

By interpreting the query results, as displayed in Figure 12, a smart cities manager can obtain different types of knowledge. For instance, the measurement of zero vehicles on March 25, 2014 (2014-3-25) could indicate that this was a day in which the sensors in the designated district were entirely disabled. Another example of knowledge that can be obtained is when the traffic in this district seems more intense in weekdays when compared with weekends.

Figure 12 – Containment query results.



Source: Research data.

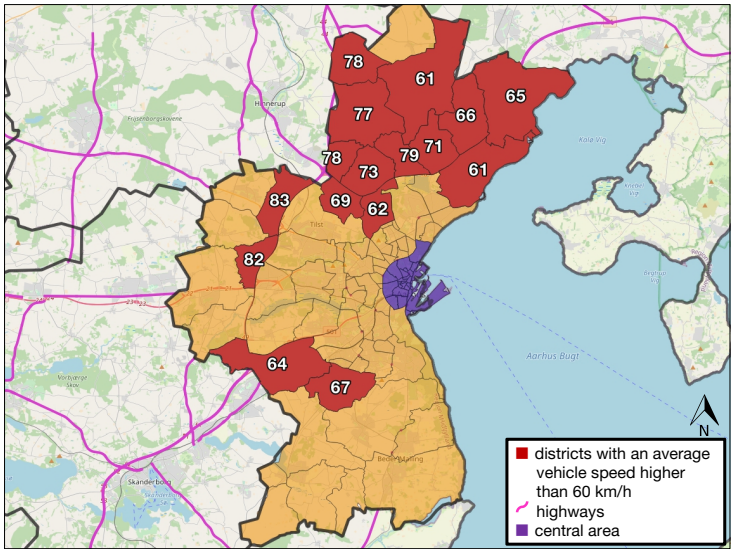
The intersection spatial join query is represented in Query 3. It returns districts in which the average vehicle speed reported from the sensors sets which intercept these sensors is greater than 60 km/h (37.28 mph). This query can be used to check if there are districts where the maximum permitted speed is not being respected by the drivers.

Query 3 – Return the districts whose average vehicle speed reported from the sensors sets which intercept it is greater than 60 km/h.

```
1: SELECT districtGeo, AVG(vehicleSpeed) AS a
2: FROM measurement, report, district
3: WHERE ST_Intersects(ST_MakeLine(firstSensorGeo,
4:          secondSensorGeo), districtGeo)
5:     AND measurement.reportID = report.reportID
6:     AND report.districtID = district.districtID
7: GROUP BY districtGeo
8: HAVING a >= 60
```

The query results are depicted in Figure 13, with each selected district being highlighted in red and the average vehicle speed (in km/h) displayed in its centre. These results indicate that the average vehicle speed is higher in the northern districts of the municipality of Aarhus. A smart cities manager can extract different types of knowledge from this information. An example is the fact that drivers can be less inclined to drive over the speed limit in municipality central areas (highlighted in purple in Figure 13), probably due to the increased number of pedestrians in these areas. Another example resides in the assumption that the average speed in the northern districts is higher due to the fact that some of them connect with external highways (displayed as pink lines in Figure 13).

Figure 13 – Intersection spatial join query results.



Source: Research data.

5.1.3 Spatial Queries with Metric Relationships

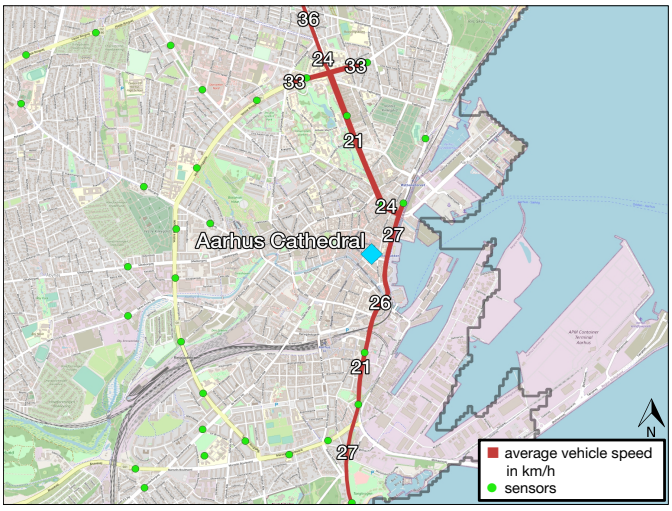
In this section, we describe queries that use metric relationships. The two metric relationships used in our case study analyses are the k-nearest neighbours and the distance spatial join. The k-nearest neighbours query, represented in Query 4, returns the average vehicle speed identified by the 10 nearest reports from the Aarhus Cathedral, which is represented by a point. This type of analyses is necessary to verify if drivers are respecting the speed limit in the surrounding area of a highly frequented point of interest, as shown in Figure 14.

Query 4 – Return the average vehicle speed identified by the 10 nearest reports from the Aarhus Cathedral.

```
1: SELECT AVG(vehicleSpeed), ST_MakeLine(firstSensorGeo,
2:     secondSensorGeo) AS reportGeo
3: FROM measurement, report
4: WHERE measurement.reportID = report.reportID
5: GROUP BY reportGeo
6: ORDER BY ST_Distance(reportGeo,
7:     ST_GeomFromWKT('POINT(10.210556 56.156944)'))
8: LIMIT 10
```

The results displayed in Figure 14 enable smart cities managers to perform a wide variety of analyses. In particular, one can identify that the highest average speeds around Aarhus Cathedral can often be observed in the main streets of its district, which are highlighted in red. Smart cities managers can also observe that these speeds do not go over 33 km/h. This can indicate that drivers do not tend to speed up in this region, a fact that might indicate the occurrence of heavy traffic.

Figure 14 – K-nearest neighbours query returning the average vehicle speed identified by the 10 nearest reports from the Aarhus Cathedral.



Source: Research data.

The distance spatial join query, listed in Query 5, compares the amount of occupied parking spots and the amount of vehicles transiting in the streets in the last week of May, considering the sensors in the streets that are located at a distance of at most 100 m from sensors placed in parking garages. Only parking garages that had at least one vehicle parked in the period were considered. This query also can be classified as a drill-across operation, where the queried fact tables are Parking and Report.

Query 5 – Distance join query returning the average number of vehicles in traffic and the average number of parked vehicles, considering the sensors in the streets of the municipality of Aarhus that are located at a distance of at most 100 m from sensors placed in parking garages.

```
SELECT garage.garageID AS "Garage Name",
    road.roadName AS "Street Name",
    ROUND(AVG(measurement.vehiclecount),0)
        AS "Average number of vehicles in traffic",
    ROUND(AVG(parking.vehiclecount),0)
        AS "Average number of parked vehicles"
FROM (SELECT reportID, SUM(vehiclecount) AS vehiclecount
      FROM measurement, date
      WHERE measurement.dateID = date.dateID
        AND month = 5 AND week = 4
      GROUP BY reportID) AS measurement,
     (SELECT parking.garageID,
        AVG(vehiclecount) AS vehiclecount
      FROM parking, date
      WHERE parking.parkingID = date.dateID
        AND month = 5 AND week = 4
      GROUP BY garageID) AS parking, report, garage, road
WHERE report.reportID = measurement.reportID
      AND parking.garageID = garage.garageID
      AND garage.roadID = road.roadID
      AND ST_Distance(ST_MakeLine(report.firstsensorgeo,
            report.secondsensorgeo), garage.geo) ≤ 0.01
GROUP BY garage.garageID
```

The query results are displayed in Table 3. By analysing these results, a smart cities manager can realise that there is no direct relationship between the average number of vehicles in traffic and the average number of parked vehicles in the period. This is clear due to the fact that there are streets with: (i) a high number of vehicles in traffic and few vehicles parked (such as Kalkværksvej); (ii) a high number of vehicles in traffic and a high amount of vehicles parked (such as Værkmestergade); and (iii) a few number of vehicles in traffic and a high amount of vehicles parked (such as Østergade). However, it is important for a smart cities manager to run this analysis on different weeks and months in order to investigate if this lack of relationship is dependent on the time period.

Table 3 – Distance join query results.

Garage Name	Street Name	Average number of vehicles in traffic	Average number of parked vehicles
BRUUNS	Værkmestergade	5,605	166
BUSGADEHUSET	Frederiksgade	4,223	84
KALKVAERKSVEJ	Kalkværksvej	4,458	49
MAGASIN	Åboulevarden	4,223	108
SALLING	Østergade	3,618	191

Source: Research data.

5.1.4 Spatial Queries with Type-Dependent Operations

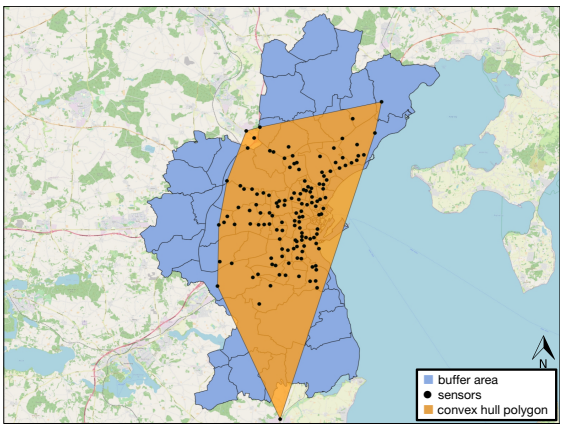
In this section, we describe queries that use type-dependent operations, i.e., operations which can be executed over only one specific type of spatial object. In our case study analyses, we are interested in spatial queries that involve generating a new geometry from a distance around a specific geometry. For instance, Query 6 is a convex hull query, which returns the minimal convex polygon (i.e., the *convex hull*) that contains all the sensors in the municipality of Aarhus, in order to get the sensor coverage area.

Query 6 – Return the minimal convex polygon which contains all sensors in Aarhus municipality.

```
1: SELECT ST_ConvexHull(ST_Collect(sensors.sensorGeo))
2: FROM (SELECT report.firstSensorGeo AS sensorGeo FROM report
3:       UNION
4:       SELECT report.secondSensorGeo AS sensorGeo FROM report)
5:       AS sensors
```

The results displayed in Figure 15 reveal that several districts in the extremities of the municipality are beyond the sensor coverage area. Further, it is possible to visualise that there are sensors positioned outside the municipality limits. A smart cities manager should analyse these results to better determine the sensor distribution across the municipality districts.

Figure 15 – Convex Hull query returning the minimal convex polygon that contains all the sensors in the municipality of Aarhus.



Source: Research data.

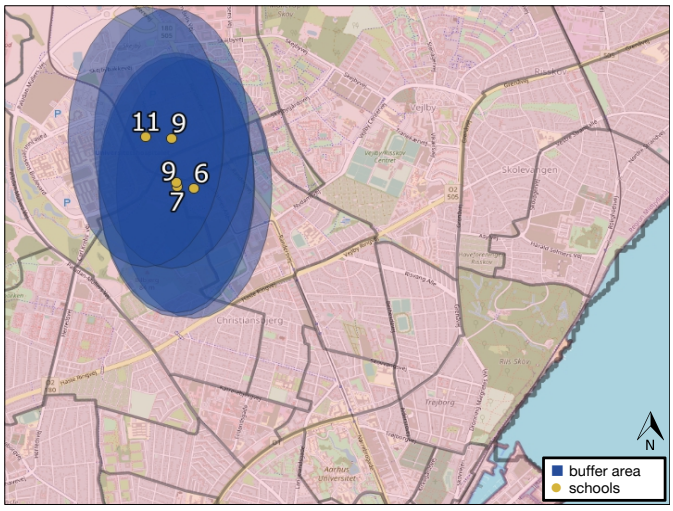
Another analysis, which represents a buffer query, returns the average vehicle count of the sensors located within a 100 m buffer of each school in the municipality of Aarhus, considering the five highest measurements. Data relating to schools in the municipality of Aarhus was extracted from OpenStreetMap and contains as attributes the identifier, school name, and its representation as a point. This type of analyses can be useful to determine the movement around school areas in different municipality regions. The following command, described in Query 7 expresses this query.

Query 7 – Return the average vehicle count of the sensors located within a 100 m buffer of each school in the municipality of Aarhus, considering the five highest measurements.

```
SELECT ROUND(AVG(measurement.vehicleCount),0)
    as AVGvehicleCount, schoolID
FROM schools, report, measurement,
    ST_Buffer(schools.geom, 0.01) AS schoolbuffer,
    ST_MakeLine(firstSensorGeo, secondSensorGeo) AS sensorset
WHERE ST_Intersects(sensorset,schoolbuffer)
    AND measurement.reportID = report.reportID
GROUP BY schoolID
ORDER BY AVGvehicleCount DESC
LIMIT 5
```

Figure 16 shows the query results. It is important to highlight that the buffers are not depicted as regular circular shapes due to the map projection adopted. Through an analysis of the query results, a smart cities manager can verify if the movement is higher around a certain school in comparison with the others.

Figure 16 – Buffer query returning the average vehicle count of the sensors located within a 100 m buffer of each school in the municipality of Aarhus, considering the five highest measurements.



Source: Research data.

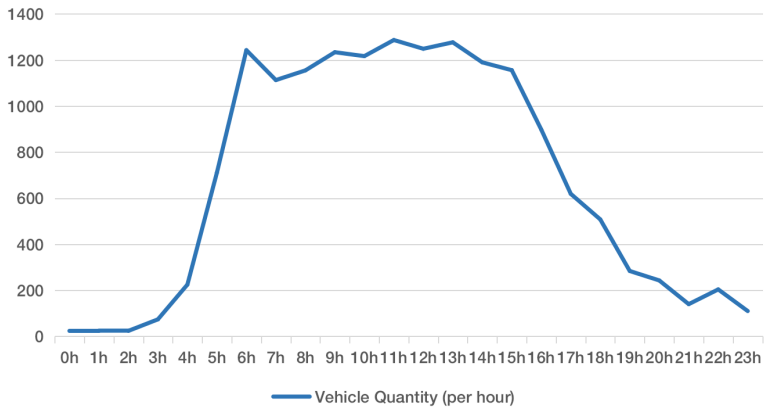
It is also possible for the manager to seek more detailed analyses, investigating the average vehicle count per hour for a certain school, as shown in Query 8.

Query 8 – Return the average vehicle count of the sensors located within a 100 m buffer of a school in the municipality of Aarhus.

```
SELECT SUM(measurement.vehiclecount), hour, day
FROM schools, report, measurement, date, time
    ST_Buffer(schools.geom, 0.01) AS schoolbuffer,
    ST_MakeLine(firstSensorGeo, secondSensorGeo) AS sensorset
WHERE ST_Intersects(sensorset,schoolbuffer)
    AND measurement.reportID = report.reportID
    AND measurement.dateID = date.dateID
    AND measurement.timeID = time.timeID
    AND schools.schoolID = 4448940550
    AND month = 2
GROUP BY hour, day
ORDER BY day, hour
```

After interpreting the results depicted in Figure 17, a smart cities manager can obtain some knowledge regarding the movement of vehicles around the analysed school. For instance, it is possible to realise that the number of vehicles in the vicinity of the school is greater during business hours. A smart cities manager can also observe that there is a small decline in the vehicle count at 7h, a fact that can be further investigated.

Figure 17 – Buffer query returning the average vehicle count per hour and day in February 2014, considering the period from 0h to 23h.



Source: Research data.

5.2 Public Transportation

In this case study, we use data related to public transportation, where sensors placed on buses collect data related to the spatial position to assist in the spatial analytics process for the improvement of urban traffic. We use data related to the Integrated Transportation Network (in Portuguese, *Rede Integrada de Transporte* (RIT)), a Bus Rapid Transit (BRT) system in Curitiba, Brazil, implemented in 1974. This system contains express bus lanes, in which bi-articulated buses that use tube-shaped stations are operated, as illustrated in Figure 18. These stations enable fare prepayment and platform level boarding, encompassing accessibility for people with disabilities (URBS, 2020).

Figure 18 – Tube station belonging to RIT, Curitiba's BRT system. These stations allow accessibility for people with disabilities and fare prepayment.

Source: URBS (2020).

We use a web-service provided by *Urbanização de Curitiba* (URBS), a mixed economy company that controls the public urban transportation of Curitiba. The web-service provides real-time generated data related to the bus position by time, date, and line. This data is publicly available in the city's administration website in a dataset containing historical data[5] and in a web-service containing real-time data.[6]. There is conventional (e.g., bus line name, bus model, etc.) and spatial data (e.g., bus locations, represented by points) referring to the period from January 2017 to today. The sensors generate data every two minutes. The API offers more complete data in relation to the dataset containing historical data, such as accessibility, status (delayed, on time or early) and vehicle category. This allows for real-time analysis of the transport system.

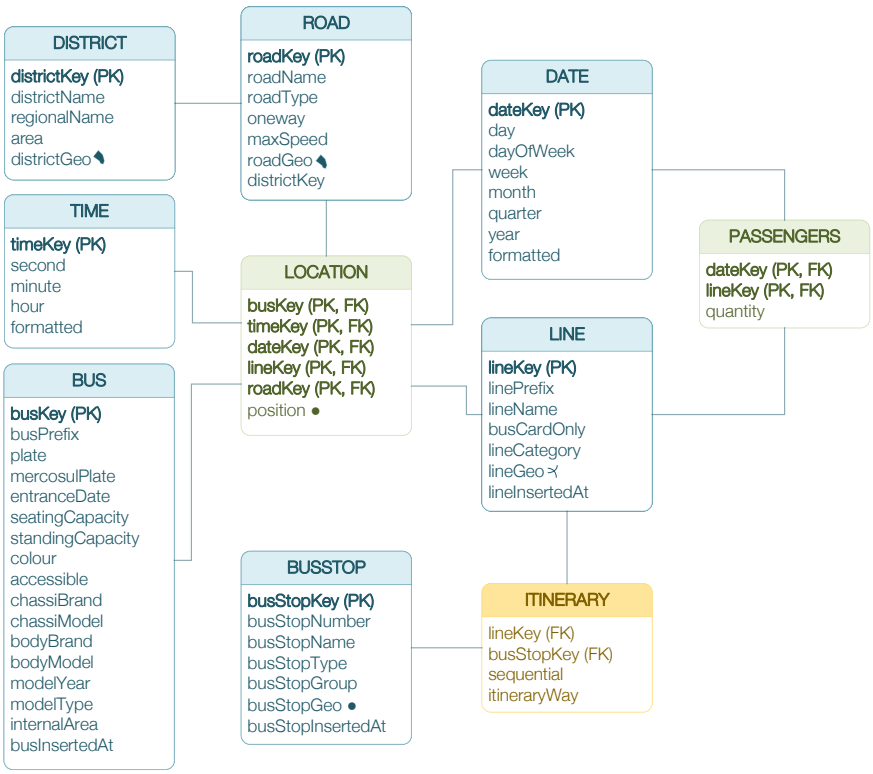
As the dataset only provides data regarding the sensor location (i.e., points), we extended it with new information to enrich the analyses performed in our spatial application. To this end, we use road data obtained by BBBike[7] from OpenStreetMap and statistical district data obtained from the *Instituto de Pesquisa e Planejamento Urbano de Curitiba* (IPPUC)[8]. We guarantee the spatial relationship between the data in the sense that a road contains buses and a district contains

5 <http://dadosabertos.c3sl.ufpr.br/curitibaurbs/>
6 <http://transporteservico.urbs.curitiba.pr.gov.br/>
7 <https://extract.bbbike.org/>
8 <http://ippuc.org.br/geodownloads/geo.htm>

multiple roads. Furthermore, we use data relating to the number of passengers transported per line, per month. We obtained this data by using the Brazilian's access to information law (BRASIL, 2011).

Data handled by the application is stored in an SDW designed according to the logical schema depicted in Figure 19, which should also be located in the cloud. This schema represents data related to bus locations in the city and number of passengers. The spatial attributes in the star-schema are represented as described by Vaisman and Zimányi (2014). There are seven dimension tables in the SDW: (i) Date and Time, storing the moment in which a measurement occurred; (ii) Bus, storing data related to a bus (e.g., model, plate, colour); (iii) Line, storing data related to a line (e.g., name, category); (iv) BusStop, storing data related to bus stops; and (iv) Road and District, storing the geographic locations associated with the report and the garage, respectively. The dimension tables are linked through two fact tables: (i) Location, which stores the bus position; and (ii) Passengers, which stores the quantity of passengers. Finally, there is a bridge table, which, according to Kimball *et al.* (2011), is a solution that is used for multivalued dimensions. In this case, the bridge table Itinerary is linked to Line and BusStop, where *a line has several bus stops* and *a bus stop has several lines*. A line itinerary have a sequence of bus stops defined in sequential. A line has the outbound or inbound itinerary, where it is indicated under itineraryWay.

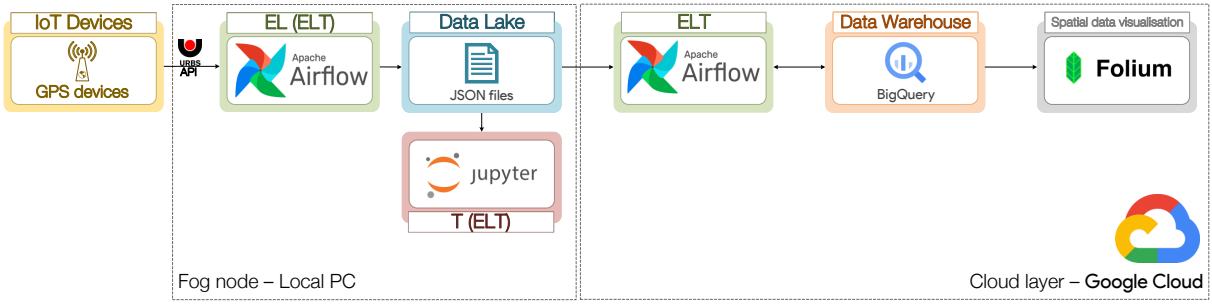
Figure 19 – Logical schema of the SDW proposed to support the case study, which models data related to sensors contained in buses and other public transportation data.



Source: Elaborated by the author.

According to the proposed architecture (Section 4.1) and guidelines (Section 4.2), the case study application should be implemented as illustrated in Figure 20. The fog computing layer consists of Apache Airflow for performing the data extraction and loading from URBS API and Jupyter notebooks[9] for transformation and visualisation of data contained in the data lake, which is a JSON file. The cloud computing layers consists of Apache Airflow for performing the extraction, transformation and loading from the fog node data. Since the historical data provided by URBS is very large, with approximately 405 GB just for the data related to the bus position, Google Cloud BigQuery can be used, as an SDW, to process the spatial queries aimed to analytics, as it complies with the application's requirements regarding performance and spatial queries, as well as supports the SQL programming language. We consider that smart city managers have some previous knowledge of the services of this platform and SQL.

Figure 20 – Pipeline related to the case study, which models data related to sensors contained in buses and other public transportation data.



Source: Elaborated by the author.

In Section 5.2.1, we describe aspects related to data preprocessing. Once the process of loading data from the IoT sensors into the SDW is complete, smart cities managers are able to execute different analyses using BigQuery. We define some query examples that address key points of the application's requirements, which are divided into three different categories: (i) fog-only queries (Section 5.2.2); (ii) fog and cloud queries (Section 5.2.3); and (iii) SOLAP cloud queries (Section 5.2.4). These queries may be employed by a smart cities manager to support spatial analytics. To visualise the results of the queries, we employ Folium[10], a Python library that builds interactive maps on a Leaflet[11] map. Folium supports choropleth maps and heatmaps.

[9] <https://jupyter.org/>
[10] <http://python-visualization.github.io/folium/>
[11] <https://leafletjs.com/>

5.2.1 Data preprocessing

The data we use in this case study is extracted through an API provided by URBS. We developed several functions to obtain the required data, such as `getVeiculos`, which returns data on the position of buses, and `getLinhas`, which returns all the bus lines. Historical data is available through CSV files. After being extracted, the data must go through an ELT process in the fog layer (Guideline F5). Thus, Apache Airflow should be employed to: (i) extract the data from the API and CSV files; (ii) perform transformations to arrange the data according to the logical schema depicted in Figure 19; and (iii) load the data into a data lake, which is implemented in this case study as JSON files. In the fog data lake, data can be transformed and visualised using Jupyter notebooks.

After a 24-hour period disposed in the data lake in the fog node, the data is sent to the Google Cloud, where it is transformed and loaded into the Location fact table, which is contained in Google BigQuery. Thus, it is possible to perform spatial queries. To this end, it is necessary to carry out the spatial data transforming. BigQuery provides a function, listed in Query 9, that converts WKT representations into spatial objects. For simplicity, we assume that all WKT representations have been converted to spatial objects.

Query 9 – Using a function to convert WKT representations in Google BigQuery.

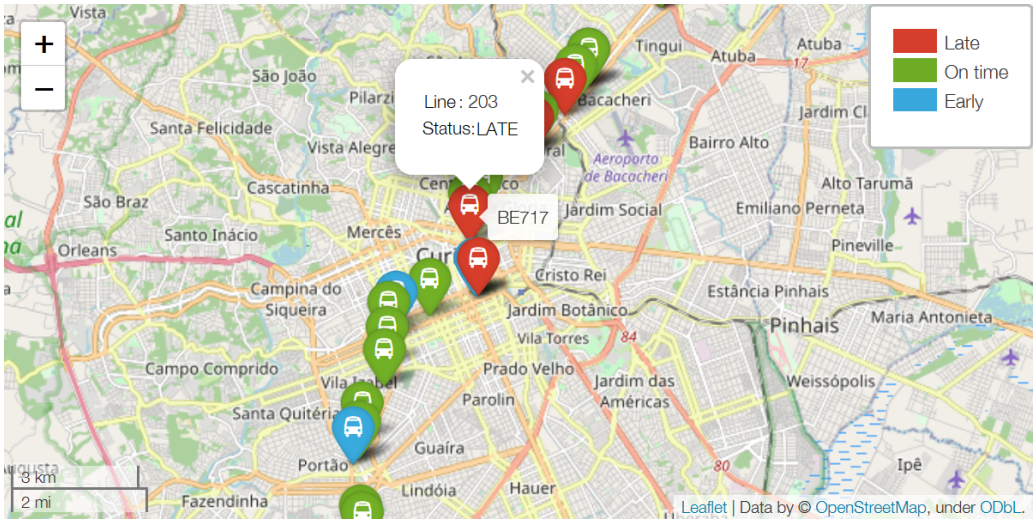
```
1: SELECT busKey, timeKey, dateKey, lineKey, roadKey,
2:        ST_GeogFromText(geom) AS position
3: FROM location
```

The data uploaded in the fog and in the cloud allows numerous queries according to the demand foreseen by the smart city manager. Using the data lake in the fog allows real-time queries, but does not allow querying supplementary data that can be integrated. Nonetheless, cloud queries allow analytics considering multiple dimensions, including spatial dimensions, providing a wide view of the business, but, in this case, there is no support for real-time queries. Therefore, mixed queries using the real-time capability of fog and cloud dimension tables can be performed. In the following sections, we show examples of queries that can be performed only in the fog, only in the cloud, and in both environments. The source codes of these queries are listed in Appendix A

5.2.2 Fog-only queries

Real-time queries can be performed on fog nodes using Python libraries. In this section, we present two examples of queries that can be executed only with data extracted from sensors, without using external databases. The first query, listed in Source Code 2 at Section A.1, returns the vehicle's location and status, similar to bus monitoring apps like CittaMobi[12] and Moovit[13]. Figure 21 illustrates the result of a query based on the express line 203 (*Santa Cândida – Capão Raso*). Different colours of the markings are used, as follows. Green, red, and blue markings indicate that the vehicle is on time, late, or early, respectively. In Figure 21, a bus, whose number is BE717, is delayed in relation to the time defined in its scale.

Figure 21 – Buses belonging to line 203 of the Curitiba's public transport system, tagged with bus status.



Source: Research data.

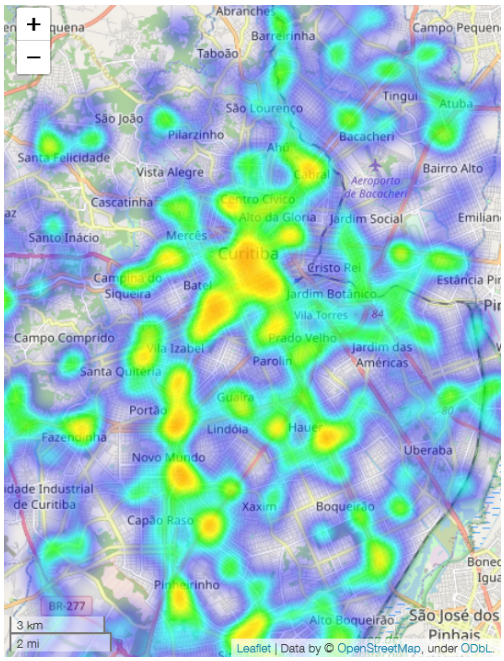
From this query, a smart city manager can have a real-time view of the status of vehicles, by line, by minute. If a line has an excess of late buses, the manager can create guidelines for bus companies and drivers to reduce these occurrences, such as the use of appropriate vehicles on the lines and the training of drivers for efficient driving. However, external factors such as traffic jams can lead to delays. Thus, a new query to detect traffic jams is necessary.

The second query, listed in Source Code 3 at Section A.1, is aimed to generate heatmaps based on the spatial position of all buses that are travelling in the city. From these heatmaps, it is possible to detect traffic jams. Figure 22 illustrates heat maps generated by data obtained at 2021-08-24 (August 24, 2021), with an interval of three hours between data collections. There is a large agglomeration of buses in downtown and near bus terminals during peak hours (Figures 22a and 22d).

[12] <https://www.cittamobi.com.br>
[13] <https://moovit.com>

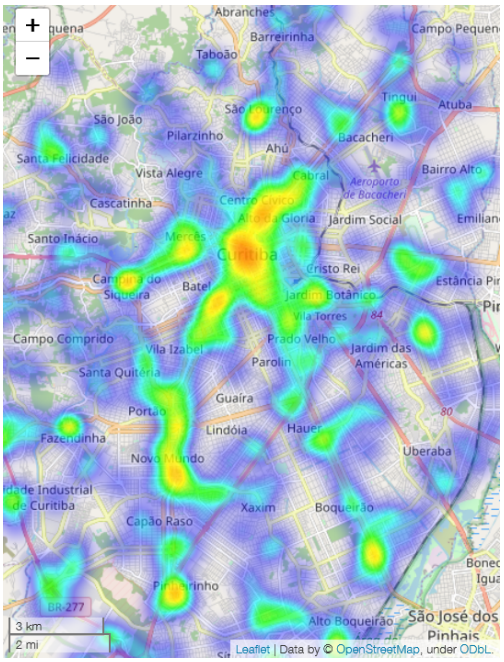
Figure 22 – Heatmaps that represent bus positions in the city of Curitiba. Data were collected at three-hour intervals on 2021-08-24.

(a) 07:55.

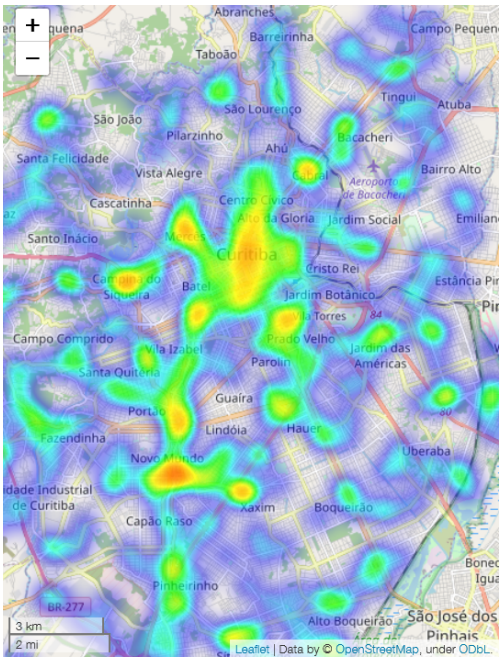


(b) 10:55.

(c) 13:55.



(d) 16:55.



Source: Research data.

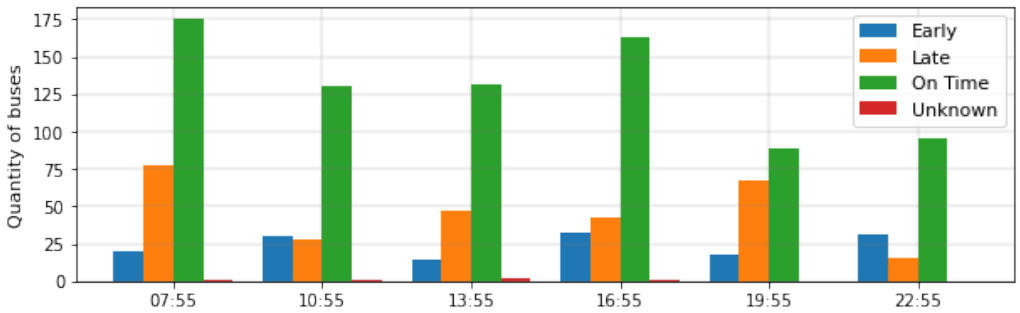
There are several vehicles that are not operating on the lines. The high amount of these vehicles in the bus garages generates anomalies in the heatmap, therefore vehicles in this situation are disregarded in the analyses. The smart city manager can use these results to check the circulation of buses near the terminals and busy avenues and highways in order to detect possible interruptions in traffic and improve the vehicle flow.

5.2.3 Mixed queries

Although it is possible to perform queries only with the data available in the fog data lake, eventually additional data is needed to obtain a more complete query. In this section, we present two examples of queries that can be executed with data located in both the fog and the cloud, and how that data can be integrated. The first query evaluates the status of the buses that belong to companies that are part of a consortium by time and status. The objective of the query is to verify which period of time in a day have more buses that are late in relation to their respective schedules. Curitiba has three consortia, Pontual, Transbus, and Pioneiro. In this query we consider the Pontual consortium, which is formed from the companies Glória, Mercês, and Santo Antônio (URBS, 2021a). The data obtained in real-time does not have information about the bus companies, thus a cloud query can be used to get the company associated with that vehicle, using the vehicle number as a parameter.

This query, listed in Source Code 4, is detailed in Section A.2. The results of this query are displayed in the bar chart depicted in Figure 23. At 19:55, there is a smaller difference in the amount of buses that are late or on time. With this information, the smart city manager can verify the reasons that make buses late in the region where the consortium operates, like traffic jams, based on the feedback from drivers who were allocated to the line at that time. This query complements the queries listed in Section 5.2.3, which can help the smart city manager to improve the lines present in these consortia, offering a smaller amount of late buses.

Figure 23 – Bar graph that lists the number of Pontual Consortium buses that circulated on 2021-08-24, grouped by time and status.



Source: Research data.

The second query checks the estimated arrival time of a bus to the next bus stop and to the bus terminal. This information can be used by passengers to check if there is a bus near the bus stop of interest. As the data contained in the data lake in the fog does not have information regarding the bus stops, such as the type and spatial position, it is necessary to perform a cloud query which returns these values, based on the bus current position.

The source code for this query, listed in Source Code 5, is detailed in Section A.2, and use Query 10 to return the bus stop closest to a given bus, using the vehicle's line, direction, and location data. First, let us analyse the estimated arrival time of a bus to the next bus stop.

As an example, consider that the bus has the number BE717, belongs to line 203, and was travelling on August 24, 2021 at 16:55, heading to Capão Raso. The spatial location of this bus is `(-49.292586 -25.481235)` From the equation $t = (S/1000)/v$, we calculate the estimated time of arrival in the next bus stop, where S is the distance and $v = 17km/h$ represents the velocity in km/h. According to URBS (2021b), 17 km/h is the average speed of the express bus lines. Similarly, it is possible to calculate the distance from the bus to the bus terminal, which is the last bus stop in the line's itinerary sequence.

Query 10 – Returns the distance from a position related to the bus number BE717, which was on line 203 on 2021-08-24, at 16:55, to the nearest bus stop, as well as the estimated time.

```
SELECT ROUND(((d/1000)/17)*60) nextBusPointArrivalTime,
    d distanceNextBusPoint, busStopName
FROM (SELECT bs.busStopName, bs.busStopGeo, finalBusStopGeo,
      ST_Distance(bs.busStopGeo,
          ST_GeogFromText('POINT (-49.292586 -25.481235)')) d,
      FROM `urbs.linenew` line,
          `urbs.itinerary` itinerary,
          `urbs.busstopnew` busstop
      WHERE line.lineKey = itinerary.lineKey
          AND bs.busStopKey = itinerary.busStopKey
          AND line.linePrefix = '203'
          AND itinerary.itineraryWay = 'VOLTA'
      ORDER BY d ASC
      LIMIT 1)
```

Therefore, it is possible to define the following indicatives related to the BE717 bus:

- **24 m** away from the nearest bus stop, with an expected **immediate arrival**.
- **1173 m** from the bus terminal, with an estimated arrival time of **four minutes**.

These queries can be useful for the passenger, who can calculate the time needed to wait for the next bus. The smart city manager may use the data generated by this query to define the bus stops with a longer vehicle arrival time, in order to identify problems in a specific line or bus.

5.2.4 Cloud queries

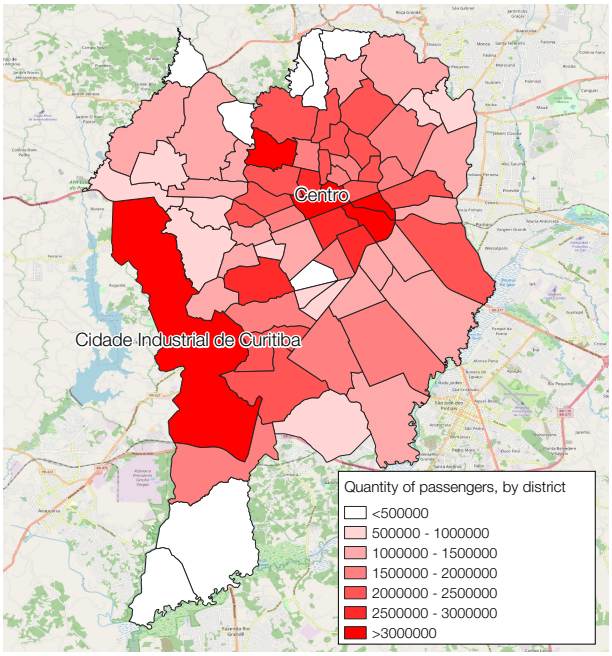
Since the SDW is located entirely in the cloud, in this section we present examples of queries that can be done using the data according to in the star-schema illustrated in Figure 19. A first query is to check the number of passengers per district, per month, in order to check the demand for lines in the districts. This is a drill-across analytical query that requires measures from the fact tables Passengers and Location. Query 11 lists two subqueries: the first returns the number of passengers per line in the month of December 2020, while the second returns the lines that operate in the districts based on the buses position.

Query 11 – Returns the number of passengers who passed through the districts of the city of Curitiba in December 2020.

```
SELECT  district.name,  district.districtGeo,
    SUM(totalPassengers.quantity) totalQuantity
FROM
    -- First Query
    (SELECT line.linePrefix,
        SUM(passengers.qtde_total) quantity
    FROM `urbs.passengers` passengers, `urbs.line` line,
        `urbs.date` date
      WHERE passengers.dateKey = date.dateKey
        AND passengers.linekey = line.linekey
        AND date.year=2020
        AND date.month = 12
      GROUP BY linePrefix
      ORDER BY quantity DESC) totalPassengers,
    -- Second Query
   (SELECT DISTINCT line.linePrefix, district.districtKey
   FROM `urbs.line` line, `urbs.location` location,
     `urbs.road` road, `urbs.district` district
   WHERE location.roadKey = road.roadkey
       AND road.districtkey = district.districtkey
       AND location.linekey = line.linekey
   ORDER BY prefix, districtKey) lineDistrict,
   `urbs.district` district
WHERE district.districtKey = lineDistrict.districtKey
    AND lineDistrict.linePrefix = totalPassengers.linePrefix
GROUP BY district.name, district.districtGeo
ORDER BY district.name;
```

The result of this query is illustrated in Figure 24. It is possible to see many passengers who travelled in the central area of the city, as well as in *Cidade Industrial de Curitiba* (CIC), located in the west of the city. In this district, there is a large amount of industries, however, due to the high population density and accessibility to public transport and other facilities, such as supermarkets and schools, there is a large increase in the population of CIC, surpassing in population cities such as Araucária and Pinhais (CURITIBA, 2017). Based on the result illustrated in Figure 24, a smart city manager can provide improvements for the lines that run through CIC, performing new queries identifying the average number of passengers. Lines with greater demand can be changed in order to have more appropriate vehicles or a larger number of buses, or new routes can be created.

Figure 24 – Returns the number of passengers who passed through the districts of the city of Curitiba in December 2020.



Source: Research data.

The second query integrates data contained in the cloud with external data. This external data, also provided by the web service of URBS, Points of Interest (POIs) located in Curitiba. They contain a name, type of POI (e.g., hospitals, hotels, schools, etc.), and geographic position. Hospitals tend to have a higher demand for elderly passengers and people with disabilities, so it is necessary to have bus stops that offer comfort and accessibility. While a tube station contains elevators and ramps, *Chapéu Chinês*-type bus stops have only a canopy and no seats (CRUZ, 2015). Thus, this query consists of checking the bus stops types contained in a 20-metre buffer over hospitals. BigQuery does not support the `ST_Buffer` function (GOOGLE, 2021), therefore, the buffer operation was calculated using Geopandas. Query 12 returns the bus stops, which are represented by points, using the `ST_Contains` function. The points must be distinct, as there may be intersecting buffers.

Query 12 – Returns the bus stops located within a 20 metres buffer of each hospital in Curitiba.

```
1: SELECT DISTINCT busStop.busStopType, busStop.busStopGeo
2: FROM 'urbs.pois' pois, 'urbs.busStop' busStop
3: WHERE ST_Contains(pois.geom, busStop.busStopGeo)
```

Figure 25 illustrates the result of Query 12. The vast majority of bus stops contained in hospital buffers are of the new furniture type, which contain a canopy and seats (CRUZ, 2015), ideal for elderly people and people with disabilities. However, there are a significant number of bus stops without seats, such as *Chapéu Chinês* and signs on a pole. This fact, more seen in hospitals on the outskirts of the city, should be seen with concern, since it is essential that there is accessibility at these stops so that elderly people or people with disabilities have the necessary comfort to use hospitals. Based on this data, the smart city manager can arrange for the exchange of these bus stops to more modern and accessible stops.

Figure 25 – Bus stops located within a 20 metres buffer of each hospital in Curitiba.

Source: Research data.

5.3 Final Remarks

In this chapter, we present two case studies based on the proposed architecture and that ware defined according to the proposed guidelines. The first case study involves data related to the number of vehicles that travel through sensors contained in the municipality of Aarhus, Denmark. The queries carried out focused on topological, metric, and type dependent spatial relationships. The second case study involves data from sensors contained in buses in Curitiba, Brazil, with query examples that can be performed on the fog layer, the cloud layer, and both on the fog and cloud layers. We also introduced some insights that can be obtained by the smart city manager to support the spatial analytics.

The next chapter, Chapter 6, presents the conclusion of the dissertation.

CHAPTER

6

CONCLUSIONS

In this dissertation, we propose an architecture aimed to help smart cities managers to enable analyses over IoT data in the context of smart cities. The architecture is composed of four layers. The terminal layer consists of a network of interconnected IoT devices aimed to collect spatial and conventional data. The fog computing layer is responsible for data extracting, loading, and transforming for real-time analyses. The cloud computing layer is a cloud computing environment based on parallel and distributed data processing frameworks and contains a spatial data warehouse. The analytics tools layer is composed of tools that enable data visualisation and querying based on the data collected and processed.

Based on our architecture, we introduce a set of guidelines to aid smart cities managers in the process of its implementation. We provide a concise yet general description of each guideline, allowing further specialisation based on the requirements imposed by each smart city application. Therefore, managers should employ the guidelines according to the specific characteristics of the smart city application in which the architecture is being employed. The proposed guidelines focus on the following issues: investigating the IoT devices heterogeneity, deploying these devices in the terminal layer, distributing fog nodes, securing the connection between IoT devices and fog nodes, enabling the ELT process in fog nodes, modelling and deploying the data lake, enabling analytical data in the fog computing layer, enabling spatial queries and the ELT process and modelling and deploying the cloud data lake, multidimensional data modelling in order to run SOLAP queries, enabling data mining and machine learning, and selecting appropriate analytics tools.

We validate the proposed architecture and guidelines by employing them to implement two case studies. The first case study consists of a spatial data warehousing application that analyses data collected from real IoT devices located in the municipality of Aarhus, Denmark. Three different categories of spatial queries were executed, i.e., spatial queries with a topological predicate, spatial queries with metric relationships, and spatial queries with type-dependent operations. We also highlight important findings that can be obtained from these queries and

how these findings can assist smart cities managers in the spatial analytics process. The second case study consists of a spatial data warehousing application that analyses data from sensors contained in buses belonging to the public transport of the city of Curitiba, Brazil. Three different categories of spatial queries were executed, i.e. fog queries, cloud queries, and both cloud and fog queries. Queries performed in the fog are aimed to support real-time data analyses, while queries carried out in the cloud are aimed to support batch data analyses. Although our case studies encompass specific cities, our architecture can be applied to any smart city that employs spatial data generated by IoT devices to improve government intelligence.

This chapter is organised as follows. Section 6.1 lists the papers that were produced during the period of execution of this dissertation. Section 6.2 describes the difficulties found in the development of the work. Section 6.3 presents future work.

6.1 Publications

During the development of this dissertation, the following papers were published.

- SANTOS, J. P. C.; CIFERRI, C. D. A. Processamento de consultas analíticas espaciais sobre dados de cidades inteligentes. In: **35th Brazilian Symposium on Databases: Database Theses and Dissertations Workshop, Companion Proceedings.** Rio de Janeiro, 2020. p. 37–43.

- SANTOS, J. P. C.; CASTRO, J. P. C.; CIFERRI, C. D. A. SOLAP Query Processing over IoT Networks in Smart Cities: A Novel Architecture. In: **Proceedings of XXI GeoInfo - Brazilian Symposium in Geoinformatics**. São José dos Campos, Brazil: INPE, 2020. p. 118–129

- CLARINDO, J. P., C; CASTRO, J. P.; AGUIAR, C. D. Combining Fog and Cloud Computing to Support Spatial Analytics in Smart Cities. **Journal of Information and Data Management**, 12(4), 2021. p. 342-360 <https://doi.org/10.5753/jidm.2021.1798>

Compared to the aforementioned papers, we also present in this master's thesis some additional contributions. We are preparing a new paper to submit to a journal containing these contributions, which are described as follows.

- We introduce new components in the architecture, thus providing new possibilities for data manipulation. For instance, the use of a data lake in the fog node, the possibility of applying machine learning techniques, and the specification of the ETL and the ELT processes;

- We define a new set of guidelines, based on these new architectural components;

- We describe the second case study (Section 5.2), which analyses real-time data from public transport in the city of Curitiba.

During the development of the master's activities, the following papers were also prepared and published. These papers are not related to the theme of this dissertation. They were written with researchers from the Federal University of Alagoas, as a consequence of the work carried out during graduation.

- CLARINDO, J. P.; FONTES, W. S.; COUTINHO, F. QualiSUS: um dataset sobre dados da Saúde Pública no Brasil. In: **XXXVI Simpósio Brasileiro de Banco de Dados: Dataset Show-case Workshop, SBBD 2019 Companion.** Fortaleza, CE: SBC, 2019. p. 418–428
- VASCONCELOS, F.; TAVARES, J. V.; RIBEIRO, M. U.; COUTINHO, F. J.; CLARINDO, J. P. CandiDATA: um dataset para análise das eleições no Brasil. In: **XXXIV Simpósio Brasileiro de Banco de Dados: Dataset Showcase Workshop, SBBD 2021 Companion.** Online: SBC, 2021.

6.2 Difficulties in the Development of the Work

In this dissertation, we faced several difficulties, as described as follows.

- **Lack of related studies for the development of the guidelines.** Some topics covered in this dissertation are relatively recent, implying on the lack of related studies. However, several papers were published in 2021, supporting the definition of the proposed guidelines.
- **Setting up of a new Hadoop/Spark cluster.** The cluster that was available to run Spark operations was outdated, with computers manufactured in 2013. To overcome this issue, we set up a new cluster, using more modern computers manufactured in 2020. This set up, carried out in 2021, proved to be complex, due to the need to install the new version of Hadoop and Spark. The current versions of Hadoop and Spark are much newer than those that were present in the old cluster, increasing the complexity of learning new features.
- **Limited documentation of the Apache Sedona.** Sedona is an Apache incubating project, thus it is currently under development. Features and documentation are limited.
- **Using Apache Spark in cluster mode with Python.** Apache Spark can be used with Python, Java, and Scala programming languages. However, the Spark standalone mode does not support the cluster mode for Python applications (APACHE, 2021). In the case study described in Section 5.1, it was possible to use Spark together with Sedona due to the low-volume dataset. However, in the case study described in Section 5.2, it was not possible to perform queries using the client mode, where operations are performed only

on one node. Due to the difficulty in developing solutions in other languages, we chose to migrate the data from this case study to Google Cloud BigQuery, which also supports SQL. The migration of the case study to the Google Cloud was intended to support the fact that the proposed architecture is flexible to be applied, considering different requirements. Therefore, the first case study was carried out in a local cluster, while the second case study was executed in a cloud computing platform.

- **New coronavirus pandemic.** The new coronavirus pandemic, which started in 2020, caused several delays in the schedules foreseen in the qualification, due to problems in accessing the tools necessary for the development of the work.

Some additional difficulties faced in the development of the case study described in Section 5.2 are detailed as follows.

- **Data access.** URBS provides a large part of the public transport data of Curitiba through APIs. Access to the APIs is only granted upon a request through Brazilian's access to information law (BRASIL, 2011). Although the law determines that there is a period in which the response should be provided, It took a longer time than expected to obtain the required data.

- **Data in PDF.** The data on the monthly number of passengers, which was also obtained using the access to information law, was made available in the PDF format. Therefore, it was necessary to convert the format to CSV in order to be able to load the data into BigQuery.

- **ETL/ELT processing over the data.** The data provided by URBS is very large, with approximately 405 GB just for the veiculos table. Because data from the streets that the buses travelled were not available, we carried out an additional spatial query to generate the Location fact table. Executing this query was costly, taking several days to build the fact table.

- **Limitations in analysing historical vehicle data.** The API provides several additional data about the status of the vehicles: status in relation to the line schedule, accessibility, and position in the line schedule. However, historical data provided in JSON files only gives the bus number, line number, collection date, and geographic position of the vehicle. This ended up limiting the Location fact table, which does not contain this additional data.

- **Lack of standardization.** Spatial coordinates are not standardised. For instance, the decimal is represented as "`.`" in older data, while in newer data it is represented as "`,`". Also, several fields are missing, limiting the scope of data analyses.

6.3 Future Work

Future work include:

- **Investigating data mining aspects.** In this dissertation, we do not cover in depth data mining operations that can be done using data from smart cities. For instance, machine learning models can be used to predict traffic jams, quantity of pollution, and people flow in a street or district, in order to enable the smart city manager to improve people's quality of life. Supervised, unsupervised, and reinforcement learning algorithms can be applied to improve spatial analytics.

- **Applying process mining techniques.** According to Aalst (2012), process mining refers to techniques that enable the extraction of knowledge from event logs, using, for example, data mining to find valuable patterns in these logs. In a smart city context, sensors can generate a varied amount of data related to measurements and time. Supporting detection of potential issues in the data flow and processes that can be automated may introduce interesting findings to the spatial analytics.

- **Describing new spatial queries.** In this dissertation, we describe spatial queries involving topological predicates and metric relationships. We intend to investigate new query categories, such as spatial queries with numerical operations, geometric set operations, and directional relationships. We also plan to investigate where these queries should be executed: fog, cloud or both environments.

- **Improving the performance of the spatial queries.** Star-joins operations tend to be very expensive in terms of processing. Although parallel and processing frameworks can be used to improve these operations, many techniques can be applied to execute spatial queries that involves star-joins (SANGAT; TANIAR; MESSOM, 2020; BRITO *et al.*, 2016). New case studies related to big spatial data should investigate the application of the different star-join techniques to verify which technique is more adequate to the characteristics of the application. New algorithms should be developed whenever necessary.

- **Integrating mixed queries automatically.** Queries performed both in the fog node and the cloud environment and then integrated are hard-coded only, with all the parameters needed for the query being passed as function parameters. We intend to create mechanisms so that queries are automatically executed and integrated according to the attributes defined by the smart city manager, avoiding the need to create new functions for specific queries.

- **Creating case studies related to other contexts.** The proposed architecture and guidelines, although focused on smart cities, is generic and can be used in other applications, such as mining, agriculture, industry, and medical areas. New case studies can be developed using the data generated in these applications.

BIBLIOGRAPHY

AALST, W. V. D. Process mining. **Communications of the ACM**, v. 55, n. 8, p. 76–83, 8 2012. Citation on page 91.

ADI, E.; ANWAR, A.; BAIG, Z.; ZEADALLY, S. Machine learning and data analytics for the IoT. **Neural Computing and Applications**, Springer Science and Business Media Deutschland GmbH, v. 32, n. 20, p. 16205–16233, 10 2020. ISSN 14333058. Citation on page 61.

AFZAL, B.; UMAIR, M.; SHAH, G. A.; AHMED, E. Enabling IoT platforms for social IoT applications: Vision, feature mapping, and challenges. **Future Generation Computer Systems**, v. 92, p. 718–731, 2019. ISSN 0167739X. Citation on page 33.

AJI, A.; WANG, F.; VO, H.; LEE, H.; LIU, Q.; ZHANG, X.; SALTZ, J. Hadoop-GIS: A high performance spatial data warehousing system over MapReduce. **Proceedings of the VLDB Endowment**, Association for Computing Machinery, v. 6, n. 11, p. 1009–1020, 2013. Citation on page 42.

ALABLANI, I.; ALENAZI, M. EDTD-SC: An IoT Sensor Deployment Strategy for Smart Cities. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 20, n. 24, p. 7191, 12 2020. ISSN 1424-8220. Available: <https://www.mdpi.com/1424-8220/20/24/7191>. Citation on page 56.

ALI, M. I.; GAO, F.; MILEO, A. CityBench: A configurable benchmark to evaluate RSP engines using smart city datasets. In: **Arenas M. et al. (eds) The Semantic Web - ISWC 2015. ISWC 2015. Lecture Notes in Computer Science**. Bethlehem, PA, USA: Springer, 2015. v. 9367, p. 374–389. ISBN 9783319250090. Available: <http://www.ict-citypulse.eu>. Citations on pages 31 and 65.

ALMORSY, M.; GRUNDY, J.; MÜLLER, I. An Analysis of the Cloud Computing Security Problem. In: **Proceedings of the APSEC 2010 Cloud Workshop**. Sydney: APSEC, 2010. Available: <http://arxiv.org/abs/1609.01107>. Citation on page 60.

ALVI, S. A.; AFZAL, B.; SHAH, G. A.; ATZORI, L.; MAHMOOD, W. Internet of multimedia things: Vision and challenges. **Ad Hoc Networks**, Elsevier, v. 33, p. 87–111, 10 2015. ISSN 1570-8705. Available: <https://www.sciencedirect.com/science/article/pii/S1570870515000876>. Citation on page 33.

APACHE. **Submitting Applications**. 2021. Available: <https://spark.apache.org/docs/3.1.2/submitting-applications.html>. Citation on page 89.

ATZORI, L.; IERA, A.; MORABITO, G. Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm. **Ad Hoc Networks**, Elsevier, v. 56, p. 122–140, 3 2017. ISSN 1570-8705. Available: <https://www.sciencedirect.com/science/article/pii/S1570870516303316>. Citations on pages 28, 33, and 34.

BALANI, Z.; VAROL, H. Cloud Computing Security Challenges and Threats. In: **8th International Symposium on Digital Forensics and Security, ISDFS 2020**. Beirut, Lebanon: IEEE, 2020. ISBN 9781728169392. Citation on page 60.

BANSAL, M.; CHANA, I.; CLARKE, S. A Survey on IoT Big Data. **ACM Computing Surveys**, Association for Computing Machinery (ACM), v. 53, n. 6, p. 1–59, 2 2021. ISSN 0360-0300. Available: <https://dl.acm.org/doi/10.1145/3419634>. Citation on page 28.

BATISTA, N. A.; SOUSA, G. A.; BRANDÃO, M. A.; SILVA, A. P. C.; MORO, M. M. Tie Strength Metrics to Rank Pairs of Developers from GitHub. **Journal of Information and Data Management**, v. 9, n. 1, 2018. Citation on page 46.

BEIMBORN, D.; MILETZKI, T.; WENZEL, S. Platform as a Service (PaaS). **Business & Information Systems Engineering**, v. 3, n. 6, p. 381–384, 2011. ISSN 1867-0202. Available: <https://doi.org/10.1007/s12599-011-0183-3>. Citation on page 43.

BELLAVISTA, P.; ZANNI, A. Feasibility of fog computing deployment based on docker containerization over RaspberryPi. In: **ICDCN '17: 18th International Conference on Distributed Computing and Networking**. New York, NY, USA: ACM, 2017. p. 1–10. ISBN 9781450348393. Available: <http://dl.acm.org/citation.cfm?doid=3007748.3007777>. Citation on page 57.

BIMONTE, S.; TCHOUNIKINE, A.; MIQUEL, M. Towards a Spatial Multidimensional Model. **Proceedings of the 8th ACM international workshop on Data warehousing and OLAP - DOLAP**, ACM Press, New York, New York, USA, 2005. Citation on page 39.

BONHAM-CARTER, G. **Geographic information systems for geoscientists : modelling with GIS**. [S.l.]: Pergamon, 1994. 417 p. ISBN 9781483144948. Citation on page 37.

BONOMI, F.; MILITO, R.; NATARAJAN, P.; ZHU, J. Fog computing: A platform for internet of things and analytics. **Studies in Computational Intelligence**, Springer Verlag, v. 546, p. 169–186, 2014. ISSN 1860949X. Citation on page 46.

BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. Fog computing and its role in the internet of things. In: **MCC '12: Proceedings of the first edition of the MCC workshop on Mobile cloud computing**. Helsinki Finland: ACM, 2012. p. 13. ISBN 9781450315197. Available: <http://dl.acm.org/citation.cfm?doid=2342509.2342513>. Citations on pages 29, 34, and 46.

BRASIL. **LEI Nº 12.527, DE 18 DE NOVEMBRO DE 2011.** 2011. Available: <http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2011/lei/l12527.htm>. Citations on pages 76 and 90.

BRITO, J. J.; MOSQUEIRO, T.; CIFERRI, R. R.; CIFERRI, C. D. A. Faster cloud Star Joins with reduced disk spill and network communication. In: **Procedia Computer Science**. [S.l.]: Elsevier B.V., 2016. v. 80, p. 74–85. Citation on page 91.

CASTRO, J. P.; CARNIEL, A.; CIFERRI, C. Analyzing spatial analytics systems based on Hadoop and Spark: A user perspective. **Software: Practice and Experience**, John Wiley and Sons Ltd, v. 50, n. 12, p. 2121–2144, 12 2020. ISSN 0038-0644. Available: <https://onlinelibrary.wiley.com/doi/10.1002/spe.2882>. Citations on pages 29, 59, and 67.

CASTRO, J. P. C.; CARNIEL, A. C.; CIFERRI, C. D. A. A User-centric View of Distributed Spatial Data Management Systems. In: **GEOINFO**. [S.l.: s.n.], 2019. p. 80–91. Citation on page 42.

CHEN, M.; MAO, S.; LIU, Y. Big data: A survey. **Mobile Networks and Applications**, Kluwer Academic Publishers, v. 19, n. 2, p. 171–209, 1 2014. ISSN 1383469X. Citation on page 29.

CIFERRI, C.; CIFERRI, R.; GÓMEZ, L.; SCHNEIDER, M.; VAISMAN, A.; ZIMÁNYI, E. Cube algebra: A generic user-centric model and query language for OLAP cubes. **International Journal of Data Warehousing and Mining**, v. 9, n. 2, p. 39–65, 4 2013. ISSN 15483924. Available: <http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jdwm.2013040103>. Citation on page 41.

CLARINDO, J. P.; FONTES, W. d. S.; COUTINHO, F. QualiSUS: um dataset sobre dados da Saúde Pública no Brasil. In: **XXXIV Simpósio Brasileiro de Banco de Dados: Dataset Showcase Workshop, SBBD 2019 Companion**. Fortaleza, CE: SBC, 2019. p. 418–428. No citation.

CONGALTON, R. G. Exploring and evaluating the consequences of vector-to-raster and raster-to-vector conversion. **Photogrammetric Engineering and Remote Sensing**, v. 63, n. 4, p. 425–434, 1997. Citation on page 38.

CRUZ, E. **A evolução dos pontos de ônibus de Curitiba** . Curitiba: [s.n.], 2015. Available: <https://www.gazetadopovo.com.br/haus/estilo-cultura/para-nao-perder-o-onibus-da-historia/>. Citations on pages 84 and 85.

CURITIBA. **CIC tem mais moradores que cidades como Guarapuava e Paranaguá - Prefeitura de Curitiba**. 2017. Available: <https://www.curitiba.pr.gov.br/noticias/cic-tem-mais-moradores-que-cidades-como-guarapuava-e-paranagua/42472>. Citation on page 84.

DASTJERDI, A. V.; BUYYA, R. Fog Computing: Helping the Internet of Things Realize Its Potential. **Computer**, v. 49, n. 8, p. 112–116, 8 2016. ISSN 0018-9162. Available: <http://ieeexplore.ieee.org/document/7543455/>. Citations on pages 34 and 46.

DEAN, J.; GHEMAWAT, S. MapReduce. **Communications of the ACM**, ACM, v. 51, n. 1, p. 107, 1 2008. ISSN 00010782. Available: <http://portal.acm.org/citation.cfm?doid=1327452.1327492>. Citation on page 41.

DEHNE, F.; KONG, Q.; RAU-CHAPLIN, A.; ZABOLI, H.; ZHOU, R. Scalable real-time OLAP on cloud architectures. **Journal of Parallel and Distributed Computing**, Academic Press Inc., v. 79-80, p. 31–41, 6 2015. ISSN 07437315. Citation on page 43.

EGENHOFER, M. J. A formal definition of binary topological relationships. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. Berlin, Heidelberg, Germany: Springer Verlag, 1989. v. 367 LNCS, p. 457–472. ISBN 9783540512950. ISSN 16113349. Citation on page 37.

ELDAWY, A.; MOKBEL, M. F. SpatialHadoop: A MapReduce framework for spatial data. In: **2015 IEEE 31st International Conference on Data Engineering**. IEEE, 2015. v. 1, p. 1352–1363. ISBN 978-1-4799-7964-6. Available: <http://ieeexplore.ieee.org/document/7113382/>. Citation on page 42.

ELDRANDALY, K. A.; ABDEL-BASSET, M.; SHAWKY, L. A. Internet of Spatial Things: A New Reference Model With Insight Analysis. **IEEE Access**, v. 7, p. 19653–19669, 2019. ISSN 2169-3536. Available: <https://ieeexplore.ieee.org/document/8634002/>. Citations on pages 28, 30, 34, 48, 49, and 51.

FANG, H. Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. In: **2015 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, IEEE-CYBER 2015**. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2015. p. 820–824. ISBN 9781479987290. Citations on pages 29 and 36.

FRAGA, E.; QUEIROLO, G. **Crescimento populacional fará mundo mudar de cara até 2100**. São Paulo: [s.n.], 2018. Available: <https://www1.folha.uol.com.br/mundo/2018/07/crescimento-populacional-fara-mundo-mudar-de-cara-ate-2100.shtml>. Citation on page 27.

GAEDE, V.; GÜNTHER, O. Multidimensional access methods. **ACM Computing Surveys**, ACM, v. 30, n. 2, p. 170–231, 6 1998. ISSN 03600300. Available: <http://portal.acm.org/citation.cfm?doid=280277.280279>. Citations on pages 38 and 66.

GARCÍA-GARCÍA, F.; CORRAL, A.; IRIBARNE, L.; MAVROMMATIS, G.; VASSILAKOPOULOS, M. A comparison of distributed spatial data management systems for processing distance join queries. In: **Lecture Notes in Computer Science**. [S.l.]: Springer Verlag, 2017. v. 10509 LNCS, p. 214–228. ISBN 9783319669168. Citation on page 42.

GOOGLE. **Geography functions in Standard SQL** . 2021. Available: <https://cloud.google.com/bigquery/docs/reference/standard-sql/geography_functions>. Citation on page 84.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. Internet of Things (IoT): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, North-Holland, v. 29, n. 7, p. 1645–1660, 9 2013. ISSN 0167-739X. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X13000241>. Citation on page 34.

GÜTING, R. H. An introduction to spatial database systems. **The VLDB Journal**, Springer-Verlag, v. 3, n. 4, p. 357–399, 10 1994. Available: <http://link.springer.com/10.1007/BF01231602>. Citations on pages 37 and 38.

HAN, J.; KAMBER, M.; PEI, J. Data Mining: Concepts and Techniques. **Data Mining: Concepts and Techniques**, Elsevier Inc., 2012. Citation on page 35.

HAN, J.; STEFANOVIC, N.; KOPERSKI, K. Selective materialization: An efficient method for spatial data cube construction. In: **LNCS**. Berlin, Heidelberg, Germany: Springer, 1998. v. 1394, p. 144–158. ISBN 3540643834. Citations on pages 28, 39, and 40.

HARINARAYAN, V.; RAJARAMAN, A.; ULLMAN, J. D. Implementing Data Cubes Efficiently. **SIGMOD Record (ACM Special Interest Group on Management of Data)**, Association for Computing Machinery (ACM), v. 25, n. 2, p. 205–216, 6 1996. ISSN 01635808. Available: <http://portal.acm.org/citation.cfm?doid=235968.233333>. Citation on page 39.

HIREMATH, S.; YANG, G.; MANKODIYA, K. Wearable Internet of Things: Concept, architectural components and promises for person-centered healthcare. **Proceedings of the 2014 4th International Conference on Wireless Mobile Communication and Healthcare - "Transforming Healthcare Through Innovations in Mobile and Wireless Technologies", MOBIHEALTH 2014**, Institute of Electrical and Electronics Engineers Inc., p. 304–307, 1 2015. Citation on page 34.

HORWITZ, L. **Internet of Things (IoT) - The future of IoT miniguide: The burgeoning IoT market continues**. 2019. Available: <https://www.cisco.com/c/en/us/solutions/internet-of-things/future-of-iot.html>. Citation on page 29.

HU, P.; DHELIM, S.; NING, H.; QIU, T. Survey on fog computing: architecture, key technologies, applications and open issues. **Journal of Network and Computer Applications**, v. 98, p. 27–42, 2017. Citations on pages 34, 35, and 46.

ISMAGILOVA, E.; HUGHES, L.; DWIVEDI, Y. K.; RAMAN, K. R. Smart cities: Advances in research—An information systems perspective. **International Journal of Information Management**, v. 47, p. 88–100, 2019. Citation on page 27.

JAVADZADEH, G.; RAHMANI, A. M. Fog Computing Applications in Smart Cities: A Systematic Survey. **Wireless Networks**, Springer, v. 26, n. 2, p. 1433–1457, 2 2020. ISSN 15728196. Available: <https://doi.org/10.1007/s11276-019-02208-y>. Citation on page 29.

JING, C.; WANG, S.; WANG, M.; DU, M.; ZHOU, L.; SUN, T.; WANG, J. A Low-Cost Collaborative Location Scheme with GNSS and RFID for the Internet of Things. **ISPRS International Journal of Geo-Information**, Multidisciplinary Digital Publishing Institute, v. 7, n. 5, p. 180, 5 2018. ISSN 2220-9964. Available: <http://www.mdpi.com/2220-9964/7/5/180>. Citation on page 34.

JO, B.; BALOCH, Z. Internet of Things-Based Arduino Intelligent Monitoring and Cluster Analysis of Seasonal Variation in Physicochemical Parameters of Jungnangcheon, an Urban Stream. **Water**, Multidisciplinary Digital Publishing Institute, v. 9, n. 3, p. 220, 3 2017. ISSN 2073-4441. Available: <http://www.mdpi.com/2073-4441/9/3/220>. Citation on page 34.

JO, J.; JOO, I. H.; LEE, K. W. Constructing national geospatial big data platform: Current status and future direction. In: **IEEE WF-IoT**. [S.l.: s.n.], 2019. p. 979–982. ISBN 9781538649800. Citations on pages 30, 48, 49, and 51.

KAMILARIS, A.; PITSILLIDES, A. The impact of remote sensing on the everyday lives of mobile users in urban areas. In: **2014 Seventh International Conference on Mobile Computing and Ubiquitous Networking (ICMU)**. IEEE, 2014. p. 153–158. ISBN 978-1-4799-2231-4. Available: <http://ieeexplore.ieee.org/document/6799087/>. Citation on page 34.

KAUR, A.; SINGH, P.; NAYYAR, A. Fog Computing: Building a Road to IoT with Fog Analytics. In: **Studies in Big Data**. Singapore: Springer, 2020. v. 76, p. 59–78. Available: <https://link.springer.com/chapter/10.1007/978-981-15-6044-6_4>. Citations on pages 29 and 59.

KIMBALL, R.; ROSS, M.; THORNTHWAITE, W.; MUNDY, J.; BECKER, B. **The Data Warehouse Lifecycle Toolkit**. Hoboken, NJ: John Wiley & Sons Inc, 2011. ISBN 9780470149775. Citations on pages 60 and 76.

KOTSEV, A.; SCHADE, S.; CRAGLIA, M.; GERBOLES, M.; SPINELLE, L.; SIGNORINI, M. Next Generation Air Quality Platform: Openness and Interoperability for the Internet of Things. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 16, n. 3, p. 403, 3 2016. ISSN 1424-8220. Available: <http://www.mdpi.com/1424-8220/16/3/403>. Citation on page 34.

KRIPPENDORF, M.; Il-Yeol Song. The translation of star schema into entity-relationship diagrams. In: **Database and Expert Systems Applications. 8th International Conference, DEXA '97. Proceedings**. [S.l.: s.n.], 1997. Citation on page 39.

LAN, L.; SHI, R.; WANG, B.; ZHANG, L. An IoT Unified Access Platform for Heterogeneity Sensing Devices Based on Edge Computing. **IEEE Access**, Institute of Electrical and Electronics Engineers Inc., v. 7, p. 44199–44211, 2019. ISSN 21693536. Citations on pages 29, 35, and 56.

LIU, S.; PENG, L.; CHI, T.; WANG, X. Research on multi-source heterogeneous data collection for the Smart City public information platform. In: **International Geoscience and Remote Sensing Symposium (IGARSS)**. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2016. v. 2016-November, p. 623–626. ISBN 9781509033324. Citations on pages 48, 49, 50, and 51.

MAHDAVINEJAD, M. S.; REZVAN, M.; BAREKATAIN, M.; ADIBI, P.; BARNAGHI, P.; SHETH, A. P. Machine learning for internet of things data analysis: a survey. **Digital Communications and Networks**, Elsevier, v. 4, n. 3, p. 161–175, 8 2018. ISSN 2352-8648. Available: <https://www.sciencedirect.com/science/article/pii/S235286481730247X>. Citation on page 33.

MALINOWSKI, E.; ZIMNYI, E. **Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications (Data-Centric Systems and Applications)**. 1. ed. [S.l.]: Springer Publishing Company, Incorporated, 2008. 1–436 p. Citation on page 39.

MATEUS, R. C.; SIQUEIRA, T. L. L.; TIMES, V. C.; CIFERRI, R. R.; CIFERRI, C. D. A. Spatial data warehouses and spatial OLAP come towards the cloud: design and performance. **Distributed and Parallel Databases**, Springer New York LLC, v. 34, n. 3, p. 425–461, 9 2016. Available: <http://link.springer.com/10.1007/s10619-015-7176-z>. Citations on pages 40, 43, 60, and 61.

MEDVEDEV, A.; ZASLAVSKY, A.; SANTIAGO, M. I.; HAGHIGHI, P. D.; HASSANI, A. Storing and indexing IoT context for smart city applications. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. St. Petersburg, Russia: Springer Verlag, 2016. v. 9870 LNCS, p. 115–128. ISBN 9783319463001. ISSN 16113349. Available: <https://link.springer.com/chapter/10.1007/978-3-319-46301-8_10>. Citation on page 58.

MIZ, V.; HAHANOV, V. Smart traffic light in terms of the cognitive road traffic management system (CTMS) based on the Internet of Things. In: **Proceedings of IEEE East-West Design & Test Symposium (EWDTS 2014)**. IEEE, 2014. p. 1–5. ISBN 978-1-4799-7630-0. Available: <http://ieeexplore.ieee.org/document/7027102/>. Citation on page 34.

NAKAGAWA, E. Y.; SCANNAVINO, K. R. F.; FABBRI, S.; FERRARI, F. C. **Revisão Sistemática da Literatura em Engenharia de Software: Teoria e Prática**. Elsevier Brasil, 2017. ISBN 9788535285970. Available: <https://books.google.com.br/books?id=kCspDwAAQBAJ>. Citation on page 45.

NI, J.; ZHANG, K.; LIN, X.; SHEN, X. S. Securing Fog Computing for Internet of Things Applications: Challenges and Solutions. **IEEE Communications Surveys and Tutorials**, Institute of Electrical and Electronics Engineers Inc., v. 20, n. 1, p. 601–628, 1 2018. ISSN 1553877X. Citation on page 57.

PANDEY, V.; KIPF, A.; NEUMANN, T.; KEMPER, A. How good are modern spatial analytics systems? In: **Proceedings of the VLDB Endowment**. Association for Computing Machinery, 2018. v. 11, n. 11, p. 1661–1673. ISSN 21508097. Available: <https://dl.acm.org/doi/10.14778/3236187.3236213>. Citations on pages 59 and 67.

PATEL, K. K.; PATEL, S. M. Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. **IJSR**, v. 6, n. 5, p. 6122–6132, 2016. Citations on pages 27, 28, and 33.

PENG, G. C. A.; NUNES, M. B.; ZHENG, L. Impacts of low citizen awareness and usage in smart city services: the case of London's smart parking system. **Information Systems and e-Business Management**, Springer Verlag, v. 15, n. 4, p. 845–876, 11 2017. ISSN 16179854. Available: <https://link.springer.com/article/10.1007/s10257-016-0333-8>. Citation on page 27.

PEUQUET, D. J.; CI-XIANG, Z. An algorithm to determine the directional relationship between arbitrarily-shaped polygons in the plane. **Pattern Recognition**, v. 20, n. 1, p. 65–74, 1987. ISSN 00313203. Citation on page 37.

PRABHU, C.; JAN, T.; PRASAD, M.; VARADARAJAN, V. Fog Analytics - a Survey. **Malaysian Journal of Computer Science**, v. 1, n. SI1, p. 141–151, 11 2020. Citation on page 59.

PRADO, R. Pérez de; GARCÍA-GALÁN, S.; MUÑOZ-EXPÓSITO, J. E.; MARCHEWKA, A.; RUIZ-REYES, N. Smart Containers Schedulers for Microservices Provision in Cloud-Fog-IoT Networks. Challenges and Opportunities. **Sensors**, MDPI AG, v. 20, n. 6, p. 1714, 3 2020. ISSN 1424-8220. Available: <https://www.mdpi.com/1424-8220/20/6/1714>. Citation on page 57.

PUTHAL, D.; MOHANTY, S. P.; BHAVAKE, S. A.; MORGAN, G.; RANJAN, R. Fog Computing Security Challenges and Future Directions [Energy and Security]. **IEEE Consumer Electronics Magazine**, Institute of Electrical and Electronics Engineers Inc., v. 8, n. 3, p. 92–96, 5 2019. ISSN 21622256. Citation on page 57.

RAMASWAMI, A.; RUSSELL, A. G.; CULLIGAN, P. J.; SHARMA, K. R.; KUMAR, E. Meta-principles for developing smart, sustainable, and healthy cities. **Science**, American Association for the Advancement of Science, v. 352, n. 6288, p. 940–3, 5 2016. ISSN 1095-9203. Available: <http://www.ncbi.nlm.nih.gov/pubmed/27199418>. Citation on page 27.

RAMNATH, S.; JAVALI, A.; NARANG, B.; MISHRA, P.; ROUTRAY, S. K. IoT based localization and tracking. In: **2017 International Conference on IoT and Application (ICIOT)**. IEEE, 2017. p. 1–4. ISBN 978-1-5386-1698-7. Available: <http://ieeexplore.ieee.org/document/8073629/>. Citation on page 28.

RANI, R.; KUMAR, N.; KHURANA, M.; KUMAR, A.; BARNAWI, A. Storage as a service in Fog computing: A systematic review. **Journal of Systems Architecture**, Elsevier B.V., v. 116, p. 1383–7621, 6 2021. ISSN 13837621. Available: <https://doi.org/10.1016/j.sysarc.2021.102033>. Citation on page 58.

RAUF, A.; SHAIKH, R. A.; SHAH, A. Security and privacy for IoT and fog computing paradigm. In: **2018 15th Learning and Technology Conference, L and T 2018**. Jeddah, KSA: IEEE, 2018. p. 96–101. ISBN 9781538648179. Citation on page 57.

RIVEST, S.; BÉDARD, Y.; MARCHAND, P. Toward better support for spatial decision making: defining the characteristics of Spatial On-Line Analytical Processing (SOLAP). **Geomatica**, v. 55, n. 4, p. 539–555, 2001. Citations on pages 28 and 40.

SADALAGE, P. J.; FOWLER, M. **NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence**. 1st. ed. Chicago, IL, USA: Addison-Wesley Professional, 2012. ISBN 0321826620. Citation on page 58.

SANGAT, P.; TANIAR, D.; MESSOM, C. Distributed ATrie Group Join: Towards Zero Network Cost. **IEEE Access**, Institute of Electrical and Electronics Engineers Inc., v. 8, p. 111598–111613, 2020. ISSN 21693536. Citation on page 91.

SAVAGLIO, C.; FORTINO, G. A Simulation-driven Methodology for IoT Data Mining Based on Edge Computing. **ACM Transactions on Internet Technology**, Association for Computing Machinery (ACM), v. 21, n. 2, p. 1–22, 3 2021. ISSN 1533-5399. Available: <https://doi.org/10.1145/3402444>. Citation on page 59.

SAWADOGO, P.; DARMONT, J. On data lake architectures and metadata management. **Journal of Intelligent Information Systems**, Springer, v. 56, n. 1, p. 97–120, 2 2021. ISSN 15737675. Available: <https://doi.org/10.1007/s10844-020-00608-7>. Citations on pages 58 and 60.

SHEHAB, N.; BADAWY, M.; ALI, H. A. Toward feature selection in big data preprocessing based on hybrid cloud-based model. **The Journal of Supercomputing 2021**, Springer, p. 1–40, 7 2021. ISSN 1573-0484. Available: <https://link.springer.com/article/10.1007/s11227-021-03970-7>. Citation on page 35.

SHI, W.; DUSTDAR, S. The Promise of Edge Computing. **Computer**, v. 49, n. 5, p. 78–81, 5 2016. ISSN 0018-9162. Available: <http://ieeexplore.ieee.org/document/7469991/>. Citations on pages 29 and 34.

SHVACHKO, K.; KUANG, H.; RADIA, S.; CHANSLER, R. The Hadoop Distributed File System. In: **2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)**. Incline Village, NV, USA: IEEE, 2010. p. 1–10. ISBN 978-1-4244-7152-2. Available: <http://ieeexplore.ieee.org/document/5496972/>. Citations on pages 29 and 42.

SIMOENS, P.; DRAGONE, M.; SAFFIOTTI, A. The Internet of Robotic Things. **International Journal of Advanced Robotic Systems**, SAGE PublicationsSage UK: London, England, v. 15, n. 1, p. 172988141875942, 1 2018. ISSN 1729-8814. Available: <http://journals.sagepub.com/doi/10.1177/1729881418759424>. Citation on page 33.

SINGH, A.; KHAMPARIA, A.; LUHACH, A. K. Performance comparison of Apache Hadoop and Apache Spark. In: **Proceedings of the Third International Conference on Advanced Informatics for Computing Research - ICAICR '19**. New York, New York, USA: ACM Press, 2019. p. 1–5. ISBN 9781450366526. Available: <http://dl.acm.org/citation.cfm?doid=3339311.3339329>. Citation on page 41.

SINGH, S.; JEONG, Y. S.; PARK, J. H. A survey on cloud computing security: Issues, threats, and solutions. **Journal of Network and Computer Applications**, Academic Press, v. 75, p. 200–222, 11 2016. ISSN 10958592. Citation on page 60.

SIQUEIRA, T. L. L.; CIFERRI, C. D. de A.; TIMES, V. C.; CIFERRI, R. R. The SB-index and the HSB-index: Efficient indices for spatial data warehouses. **GeoInformatica**, Springer US, v. 16, n. 1, p. 165–205, 1 2012. ISSN 13846175. Citation on page 60.

SOOMRO, K.; BHUTTA, M. N. M.; KHAN, Z.; TAHIR, M. A. **Smart city big data analytics: An advanced review**. Wiley-Blackwell, 2019. e1319 p. Available: <https://doi.org/10.1002/widm.1319>. Citation on page 61.

TANG, B.; CHEN, Z.; HEFFERMAN, G.; WEI, T.; HE, H.; YANG, Q. A hierarchical distributed fog computing architecture for big data analysis in smart cities. In: **ACM International**

Conference Proceeding Series. [S.l.]: Association for Computing Machinery, 2015. v. 07-09-Ocobert-2015. ISBN 9781450337359. Citation on page 29.

THEODOROU, V.; DIAMANTOPOULOS, N. GLT: Edge gateway ELT for data-driven intelligence placement. **Proceedings - 2019 IEEE/ACM Joint 4th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation and Evolution, RCoSE/DDrEE 2019**, Institute of Electrical and Electronics Engineers Inc., p. 24–27, 5 2019. Citations on pages 30, 48, 50, and 51.

URBS. **História do Transporte de Curitiba**. 2020. Available: <https://www.urbs.curitiba.pr.gov.br/transporte/historia-transporte>. Citation on page 75.

____. **Empresas Operadoras**. 2021. Available: <https://www.urbs.curitiba.pr.gov.br/transporte/rede-integrada-de-transporte/37>. Citation on page 81.

____. **Obras do Ligeirão Norte-Sul passam por vistoria**. 2021. Available: <https://www.urbs.curitiba.pr.gov.br/noticia/obras-do-ligeirao-norte-sul-passam-por-vistoria>. Citation on page 82.

VAISMAN, A.; ZIMÁNYI, E. **Data Warehouse Systems: Design and Implementation**. Berlin, Heidelberg, Germany: Springer Publishing Company, Incorporated, 2014. 625 p. ISBN 978-3-642-54654-9. Citations on pages 40, 41, 60, 66, and 76.

VASCONCELOS, F.; TAVARES, J. V.; RIBEIRO, M. U.; COUTINHO, F. J.; CLARINDO, J. P. CandiDATA: um dataset para análise das eleições no Brasil. In: **XXXIV Simpósio Brasileiro de Banco de Dados: Dataset Showcase Workshop, SBBD 2021 Companion**. Online: SBC, 2021. No citation.

VASSILIADIS, P. A Survey of Extract–Transform–Load Technology. **International Journal of Data Warehousing and Mining (IJDWM)**, IGI Global, v. 5, n. 3, p. 1–27, 1 2009. Available: <https://www.igi-global.com/article/survey-extract-transform-load-technology/3894www.igi-global.com/article/survey-extract-transform-load-technology/3894>. Citation on page 36.

WAAS, F.; WREMBEL, R.; FREUDENREICH, T.; THIELE, M.; KONCILIA, C.; FURTADO, P. On-Demand ELT Architecture for Right-Time BI. **International Journal of Data Warehousing and Mining**, 4 2013. Citation on page 36.

WANG, C.; HUANG, X.; QIAO, J.; JIANG, T.; RUI, L.; ZHANG, J.; KANG, R.; FEINAUER, J.; MCGRAIL, K. A.; WANG, P.; LUO, D.; YUAN, J.; WANG, J.; SUN, J. Apache IoTDB. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 13, n. 12, p. 2901–2904, 8 2020. ISSN 2150-8097. Available: <https://dl.acm.org/doi/10.14778/3415478.3415504>. Citation on page 58.

WANG, L.; LASZEWSKI, G. von; YOUNGE, A.; HE, X.; KUNZE, M.; TAO, J.; FU, C. Cloud Computing: a Perspective Study. **New Generation Computing**, v. 28, n. 2, p. 137–146, 2010. ISSN 1882-7055. Available: <https://doi.org/10.1007/s00354-008-0081-5>. Citation on page 42.

WANG, S.; ZHONG, Y.; WANG, E. An integrated GIS platform architecture for spatiotemporal big data. **Future Generation Computer Systems**, Elsevier B.V., v. 94, p. 160–172, 5 2019. ISSN 0167739X. Citations on pages 30, 48, 49, and 51.

Wu He; Gongjun Yan; Li Da Xu. Developing Vehicular Data Cloud Services in the IoT Environment. **IEEE Transactions on Industrial Informatics**, v. 10, n. 2, p. 1587–1595, 5 2014. ISSN 1551-3203. Available: <http://ieeexplore.ieee.org/document/6709775/>. Citation on page 34.

XU, Q.; ZHANG, J. PiFogBed: A Fog Computing Testbed Based on Raspberry Pi. In: **2019 IEEE IPCCC**. London, United Kingdom: IEEE, 2019. ISBN 9781728110257. Citation on page 57.

YEH, H. The effects of successful ICT-based smart city services: From citizens' perspectives. **Government Information Quarterly**, JAI, v. 34, n. 3, p. 556–565, 9 2017. ISSN 0740-624X. Available: <https://www.sciencedirect.com/science/article/pii/S0740624X16300521>. Citation on page 27.

YOU, S.; ZHANG, J.; GRUENWALD, L. Large-scale spatial join query processing in Cloud. In: **Proceedings - International Conference on Data Engineering**. [S.l.]: IEEE Computer Society, 2015. v. 2015-June, p. 34–41. ISBN 9781479984411. ISSN 10844627. Citation on page 42.

YU, J.; WU, J.; SARWAT, M. GeoSpark: A cluster computing framework for processing large-scale spatial data. In: **ACM GIS**. New York: [s.n.], 2015. p. 1–4. ISBN 9781450339674. Available: <http://dl.acm.org/citation.cfm?doid=2820783.2820860>. Citation on page 42.

YUAN, L.; ZHAO, J. Construction of the system framework of Spatial Data Warehouse in Internet of Things environments. In: **2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)**. Nanjing, China: IEEE, 2012. p. 54–58. ISBN 978-1-4673-1744-3. Available: <http://ieeexplore.ieee.org/document/6463121/>. Citations on pages 30, 48, 49, 50, and 51.

ZAHARIA, M.; CHOWDHURY, M.; FRANKLIN, M. J.; SHENKER, S.; STOICA, I. Spark: Cluster Computing with Working Sets. In: **HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing**. Boston, MA, USA: USENIX Association, 2010. p. 10. Available: <https://dl.acm.org/citation.cfm?id=1863113>. Citation on page 41.

ZAHARIA, M.; FRANKLIN, M. J.; GHODSI, A.; GONZALEZ, J.; SHENKER, S.; STOICA, I.; XIN, R. S.; WENDELL, P.; DAS, T.; ARMBRUST, M.; DAVE, A.; MENG, X.; ROSEN, J.; VENKATARAMAN, S. Apache Spark. **Communications of the ACM**, ACM, v. 59, n. 11, p. 56–65, 10 2016. ISSN 00010782. Available: <http://dl.acm.org/citation.cfm?doid=3013530.2934664>. Citations on pages 29, 41, and 42.

ZEE, E. van der; SCHOLTEN, H. Spatial dimensions of big data: Application of geographical concepts and spatial technology to the internet of things. **SCI**, Springer Verlag, v. 546, p. 137–168, 2014. ISSN 1860949X. Citation on page 28.

APPENDIX

A

FOG AND MIXED QUERIES SOURCE CODES

In this appendix, we present the source codes used in the fog and mixed queries in the case study described in Section 5.2. Source Code 1 lists functions that can be used in both the fog and mixed contexts. `load_to_bigquery()` function permits to insert a row (or a set of rows) in a BigQuery table.

Source Code 1 – Basic functions that allow queries from fog or mixed contexts.

```
 1: import requests
 2: import folium
 3: import pandas as pd
 4: import bigquery
 5:
 6: # Construct a BigQuery client object.
 7: client = bigquery.Client()
 8:
 9: # Define URBS webservice URL and auth token
10: ws_url = 'http://transporteservico.urbs.curitiba.pr.gov.br/'
11: auth = "TOKEN"
12:
13: # Load a row to a BigQuery table
14: def load_to_bigquery(table_id, rows):
15:     client.insert_rows_json(table_id, rows)
16:
17: # Funcion to initialise a folium map centred in Curitiba, Brazil.
18: def initalise_map():
19:     return folium.Map(width=700,height=350,location=[-25.45, -49.3
      ], zoom_start=12, control_scale=True)
```

A.1 Fog queries

Source Code 2 lists how the first query described in Section 5.2.2 can be executed, from the generation of a map indicating the buses belonging to bus line 203 of the public transport system in Curitiba, Brazil. The result of this query is illustrated in Figure 21.

Source Code 2 – Plotting a map the points related to buses from the line 203 on Curitiba, Brazil, with tags indicating the status of these vehicles.

```
1: # Function to get all vehicles in a line by line number.
2: def get_vehicle(line):
3:     return requests.get(f'{ws_url}getVeiculos.php?linha={line}&{
       auth}'.json())
4:
5: # Initialise a folium map and get all vehicles in 203 line.
6: m = initialise_map()
7: d = get_vehicle('203')
8:
9: for k, v in d.items():
10:     # Generating folium parameters from vehicle dictionary.
11:     bus, line, status, position = k,v['CODIGOLINHA'],v['SITUACAO'
       ],[float(v['LAT']),float(v['LON'])]
12:
13:     # If the bus is not in/going to garage.
14:     if line != 'REC':
15:         # Colour the marker according to the bus status.
16:         if status == 'ATRASADO':
17:             colour = 'red'
18:         elif status == 'NO HORÁRIO'
19:             colour = 'green'
20:         else:
21:             colour = 'blue'
22:
23:         # Inserts the marker into the folium object.
24:         folium.Marker(
25:             location=position,
26:             tooltip=bus,
27:             popup='Line: {}\nStatus: {}'.format(line,status),
28:             icon=folium.Icon(color=colour, icon='bus',prefix='fa')
29:         ).add_to(m)
```

Source Code 3 lists how heatmaps illustrated in Figure 22 were generated using folium library.

Source Code 3 – Generating a heatmap using folium.

```
# Get all buses in city through a request from URBS API.
def get_bus():
    return r.get(ws_url + 'getVeiculos.php?c=' + auth).json()

# Initialise a folium map and get all buses positions.
m = initialise_map()
d = get_bus()

# Generate the heatmap excluding out of line buses.
heat_data = [[float(v['LAT']), float(v['LON'])]
             for k, v in d.items() if (v['CODIGOLINHA']!='REC')]
HeatMap(heat_data, radius=14).add_to(m)
```

A.2 Mixed queries

Source Code 4 lists how the query illustrated in Figure 23 was performed using Pandas. The `get_companies` function performs a query to BigQuery, which returns all vehicles with their respective companies. For optimization reasons, the query result is arranged in a `dict`, not needing to perform other queries. After that, the company name is associated with the data from the IoT devices by bus number, and, by using a Pandas dataframe, the data is grouped by time and status.

Source Code 4 – Mixed query that returns a bar chart that contains the number of buses belonging to Pontual Consortium, grouped by time, by status.

```
# Function that returns a dictionary whose key is the bus number
    and the value is the company associated with the bus.
def get_company():
    query_job = client.query('select company, busPrefix from `urbs
    .bus``')
    comp_dict = {}

    for comp in query_job.result():
        comp_dict = {**comp_dict,**{comp.busPrefix:comp.company}}
    return  comp_dict


# Get all vehicles with their companies
companies = get_company()

# Select only vehicles belonging to the Pontual consortium.
cons = ['GLÓRIA','MERCÊS','SANTO ANTÔNIO']
d = get_bus()

for k, v in d.items():
    if companies.get(v['COD']) in cons and v['SITUACAO'] != '':
        actual = {k:{'STATUS':v['SITUACAO'], 'TIME':v['REFRESH']}}
        nd = {**nd,**actual}

# Group by time, by status and plotting using Pandas
df = pd.DataFrame(d).T.groupby('STATUS').TIME.value_counts().
    unstack(0).plot.bar()
```

Source Code 5 lists how Query 10 can be performed using BigQuery API to execute SQL queries. Using the `get_vehicle()` function, we return all vehicles in line 203, and if the BE717 bus is on the line, then execute Query 10, marked in the source code as a "$", in order to return the nearest bus stop and the estimated time of arrival.

Source Code 5 – Mixed query that returns the nearest bus stop and the estimated time of arrival. The "$" indicates a SQL query, in this case, Query 10.

```
1: # Function to get all vehicles in a line by line number.
2: def get_vehicle(line):
3:     return requests.get(f'{ws_url}getVeiculos.php?linha={line}&{
       auth}'.json())
4:
5: d = get_vehicle('203')
6:
7: for k, v in d.items():
8:     if k = 'BE717':
9:     client.query('$')
10:    res_dict = {}
11:
12:    for result in query_job.result():
13:        print(result)
```


ICMC
USP
SÃO CARLOS