UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DEPARTMENT OF INFORMATION ENGENEERING

MASTER THESIS IN COMPUTER ENGINEERING

Design and Development of a Cloud-Based Data Lake and Business Intelligence Solution on AWS

MASTER CANDIDATE

Christian Marchiori

Student ID 2078343

SUPERVISOR

Prof. Gianmaria Silvello

University of Padova

ACADEMIC YEAR
2023/2024
DATE : X DECEMBER 2024

To my parents
and friends

Abstract

This thesis focuses on the design and implementation of a Business Intelligence (BI) system and a Data Lake for the company UNOX S.p.A., leveraging the technologies provided by the Amazon Web Services (AWS) platform. The project was developed in a corporate context where rapid and accurate information analysis is crucial to improving operational performance and supporting data-driven strategic decisions.

The BI system developed allows the collection and analysis of data from various sources, helping to reduce inefficiencies, flag potential issues, identify new revenue streams, and pinpoint areas for future growth. Through the creation of a centralized Data Lake and the automation of data integration and analysis processes, it became possible to optimize the management of corporate information, providing a comprehensive and detailed view of business performance.

The system was designed to be easily accessible to various company teams, including Research and Development, IT, and other technical departments, thanks to intuitive tools such as AWS QuickSight. These tools enable a simple and visual interaction with the data without requiring specific knowledge of DBMS or query languages. This approach has made data more accessible at all company levels, promoting greater autonomy in their analysis.

The implementation made use of AWS components such as Amazon S3, Glue, and QuickSight, ensuring scalable, secure, and fully automated data management. The final result is an integrated Business Intelligence system that significantly reduces the time required for analysis and reporting, enhancing decision-making capabilities and promoting a data-driven corporate culture.

Sommario

Il presente lavoro di tesi riguarda la progettazione e implementazione di un sistema di Business Intelligence (BI) e di un Data Lake per l'azienda UNOX S.p.A, sfruttando le tecnologie messe a disposizione dalla piattaforma Amazon Web Services (AWS). Il progetto si colloca in un contesto aziendale in cui l'analisi rapida e precisa delle informazioni è cruciale per migliorare le performance operative e supportare decisioni strategiche basate sui dati.

Il sistema di BI realizzato consente di raccogliere e analizzare dati provenienti da fonti diverse, contribuendo a ridurre le inefficienze, segnalare eventuali criticità, individuare nuovi flussi di ricavi e identificare aree di crescita futura. Attraverso la creazione di un Data Lake centralizzato e lautomazione dei processi di integrazione e analisi dei dati, si è reso possibile ottimizzare la gestione delle informazioni aziendali, garantendo una visione complessiva e dettagliata delle prestazioni.

Il sistema è stato progettato per essere facilmente accessibile anche da team aziendali, tra cui Ricerca e Sviluppo, IT e altri reparti tecnici, grazie all'uso di strumenti intuitivi come AWS QuickSight, che consentono un'interazione semplice e visiva con i dati, senza necessità di conoscenze specifiche in DBMS e linguaggi di query. Questo approccio ha permesso di rendere i dati fruibili a tutti i livelli aziendali, favorendo una maggiore autonomia nella loro analisi.

L'implementazione ha utilizzato componenti AWS quali Amazon S3, Glue e QuickSight, garantendo una gestione scalabile, sicura e completamente automatizzata dei dati. Il risultato finale è un sistema integrato di Business Intelligence che riduce significativamente i tempi di analisi e reporting, migliorando la capacità decisionale e promuovendo una cultura aziendale data-driven.

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

BI Business Intelligence

AWS Amazon Web Services

B2B Business-to-Business

ETL Extract-Transform-Load

KPI Key Performance Indicator

RDBMS Relational Database Management System

JSON JavaScript Object Notation

DDC Data Driven Cooking

EC2 Elastic Compute Cloud

S3 Simple Storage Service

VPC Virtual Private Cloud

IAM Identity and Access Management

RDS Relational Database Service

ORC Optimized Row Columnar

SQL Structured Query Language

CSV Comma Separated Values

ASL Amazon States Language

HTTP Hypertext Transfer Protocol

ACID Atomicity, Consistency, Isolation, Durability

1 Introduction

1.1 The company

This thesis was conducted in collaboration with Unox S.p.A., a leading company in the professional cooking ovens market. Unox manufactures various oven series and distributes its products to over 130 countries worldwide. In addition to its product offerings, the company provides after-sales services, including cooking training, customer support, and technical assistance.

1.1.1 Company Profile

Unox S.p.A. has been active since 1990, specializing in the production of professional appliances for the catering and baking industries. As a product-focused company, its primary emphasis is on manufacturing and customer support for its ovens, rather than software development. Unox operates in the Business-to-Business (B2B) sector, serving a diverse range of clients, from small bakeries to large restaurant chains and catering centers.

Initially, the company capitalized on a market gap by producing ovens primarily for suppliers of frozen croissants in Southern Europe. These products were mainly provided to small retailers through lease agreements with large suppliers. Over time, Unox shifted its focus to producing higher-quality products for direct sale to end-users. This strategic change allowed the company to expand into Northern Europe and laid the groundwork for further growth in other continents.

Today, Unox offers a wide range of products, reflecting its evolution over the years. Recently, the company has increasingly focused on the digital advancement of its offerings, aiming to provide customers with a more comprehensive and satisfying experience. The company's flagship models now feature touch-screen panels, voice control, remote operation, data management for large companies, and other advanced features.

To meet the demands of this digital shift, Unox began developing software in-house to support these services, expanding its workforce to include software developers. The company currently has two dedicated software development teams and a Research and Development (R&D) team.

Despite its growth into a multinational corporation, Unox has remained family-owned. Production was always mainly based in Italy until a new production site was opened in the US in 2022. Since the beginning, international expansion has been focused on commercial operations, with the company establishing sales offices worldwide.

Unox employs a high level of vertical integration, producing most of its components in-house or through subsidiaries.

1.1.2 Software Development at Unox

Despite being a company primarily focused on manufacturing high-quality professional ovens, software development plays a critical role in Unox's operations. The digital transformation of its products, combined with the need for integrated connectivity and advanced functionalities, has made software a cornerstone of the company's offerings. Today, Unox ovens are fully digitized and connected, featuring capabilities such as remote control via mobile applications, data export for performance analysis, and integration with external systems through APIs. This shift has led to the formation of specialized software development teams that ensure Unox ovens remain at the cutting edge of technological innovation.

Unox currently operates with four distinct software development teams, each with a specialized focus:

- **IT Team**: This team is responsible for the internal infrastructure that supports Unox's daily operations. Their work includes managing the company's network, ensuring data security, and maintaining the systems that enable smooth communication and operational efficiency across all departments.

- **R&D Team**: Unox's Research and Development team focuses on innovation, developing new technologies for ovens to improve performance, energy efficiency, and user experience. This team collaborates with various departments to drive the technical evolution of Unox's products.
- **Software Developer Team**: This team is at the heart of the technical development process, responsible for creating, maintaining, testing, and documenting the algorithms that optimize oven performance. Their tasks include defining technical specifications in collaboration with other teams, implementing reliable and efficient solutions, and supporting field technicians to resolve software-related issues. From initial concept to final release, the Software Developer team ensures that each line of code contributes to the operation of Unox products.
- **Digital Experience Team**: As part of the company's push towards a fully connected ecosystem, the Digital Experience team focuses on developing cloud-based applications, both for web and mobile platforms, and managing REST APIs. They are responsible for creating the digital interfaces that allow users to remotely control and monitor ovens, manage data streams from connected devices globally, and integrate Unox products with other systems. Additionally, this team designs and maintains the cloud infrastructure, ensuring the reliability and scalability of Unox's digital services. They collaborate with data scientists to extract valuable insights from the vast amounts of telemetry data produced by the ovens. They also coordinate closely with the UI/UX team to deliver intuitive user experiences. This is the team I have been a part of during my internship, where I contributed to the development and enhancement of Unox's digital services, helping to bridge the gap between product performance and user interaction.

Each of these teams plays a critical role in ensuring that Unox continues to lead the market, not only with its physical products but also through its advanced digital offerings.

1.2 Initial problem

In the modern business environment, the ability to access information quickly and accurately is a key factor for competitiveness. Strategic decisions rely on precise analysis, which requires not only access to data but also appropriate tools for managing and interpreting it. Without the implementation of a structured BI system, such as the one developed in this project, Unox faces several challenges that limit the effectiveness and efficiency of its data extraction and analysis processes, impacting the entire company structure.

The main issue concerns the way data is requested, processed, and distributed within the company. Specifically, when an employee from a department like Research and Development, IT, or a technical division needs data for specific analyses, they turn to the Digital Experience team, which is designated to manage access to the company's databases and has greater expertise in running complex queries. While other teams may have some knowledge in this area, the Digital Experience team is responsible for overseeing and managing these processes. However, this approach slows down the flow of information and affects overall company efficiency, creating bottlenecks in decision-making processes.

The data extraction process involves several labor-intensive stages: the Digital Experience team must first understand the nature of the problem, identify the relevant tables and data, envision the final output of the analysis, and then develop a TypeScript script to access the databases, execute queries, and transform the data into a usable format for analysis. This "ad hoc" approach, while effective for specific requests, requires a significant amount of time and resources. Each request for analysis or reporting typically involves hours of work from an expert, limiting the company's ability to respond quickly to new market demands or opportunities.

In some cases, automated scripts are developed to reduce the repetitiveness of this process, executing periodic extractions and sending the data via email to relevant stakeholders. However, while this approach is useful, it remains limited. First, each automated script must be specifically developed for each case, leading to development and maintenance costs, along with cloud resource execution costs (such as AWS Lambda used to automate the periodic execution of these scripts). Additionally, these automations only cover a small portion of the company's overall needs and lack the flexibility to quickly respond to more complex or unexpected analysis requests.

The current working model also strongly limits the autonomy of non-technical teams. Many employees, despite needing data to improve their analyses and make informed decisions, are unable to directly access the information, as using advanced query languages or interacting with complex databases is beyond their expertise. This not only increases the workload for the Digital Experience team but also slows response times and decision-making, negatively impacting overall operational efficiency.

Other motivations behind this project include the need to improve data governance and integration between different information silos. In an environment where data is fragmented across various systems and databases, it becomes difficult to obtain a unified and coherent view of business performance, identify inefficiencies, or explore new growth opportunities. Implementing a BI system, supported by an AWS-based Data Lake, overcomes these barriers, improving the management of company data and simplifying real-time access to information.

In summary, the main reasons for implementing a new Business Intelligence system are:

- **Simplifying data access for non-technical teams**: Creating an interface and tools that enable employees without advanced technical skills to perform analysis and reporting independently.
- **Data integration**: Overcoming data fragmentation and ensuring a coherent integration of information from various systems, facilitating collaboration and data-driven decision-making.
- **Improving operational efficiency**: Optimizing the use of the Digital Experience team's resources, reducing the workload related to ad hoc requests and allowing them to focus on higher-value tasks.
- **Reducing data extraction and analysis times**: Eliminating bottlenecks and automating repetitive processes to allow different teams to independently access relevant information.
- **Scalability and flexibility**: Adopting a scalable and flexible platform, such as AWS, to efficiently manage large volumes of data and quickly adapt to the company's evolving needs.

With these premises, the project aims to revolutionize the way the company manages, accesses, and analyzes data, significantly improving the effectiveness of decision-making processes.

1.3 Goals

The primary objective of this project is to develop a unified and flexible system for managing the data generated by industrial ovens, without the need to create separate workflows for different stages of the process. The solution must ensure a consistent approach both during the bulk load phase, which involves large-scale data imports, and in the subsequent operational phase, where smaller but more frequent data updates are managed. This requires the design of a data ingestion infrastructure that can dynamically adapt to different data volumes, ensuring efficiency and ease of maintenance.

Another key goal is to ensure the efficiency of the system in terms of resource usage, with a strong focus on cost optimization. Given that the architecture relies heavily on several managed services from AWS, such as Glue, Lambda, and Step Functions, it is crucial to minimize resource consumption, reducing the execution time and memory usage of various tasks. This helps to keep operational costs in check, as AWS pricing is often directly tied to the resources utilized.

An additional objective of the project is to provide a system that integrates a query engine for data analysis and a dashboard for Key Performance Indicator (KPI) visualization. Queries should be executed using tools like AWS Athena, which allows for SQL queries to be run directly on data stored in Amazon S3, leveraging a flexible and scalable system without the need for complex database setups. The interactive dashboards, created using AWS QuickSight, will allow users to visualize data intuitively, monitor key metrics, and generate customized reports.

Furthermore, the system must ensure that the data is kept synchronized with the production databases, offering up-to-date access to information for reporting and monitoring purposes. A secondary, yet innovative, optional objective is the exploration of generative AI techniques to automate the creation of dashboards in AWS QuickSight, further simplifying the user experience and enhancing the overall efficiency of the data visualization process.

In summary, the project aims to build a solution that is flexible, efficient, and capable of supporting data analysis and visualization effectively, with a focus on automation, resource optimization, and ease of use.

1.4 Proposed Solution

The proposed system is built on a scalable, automated architecture using AWS cloud technologies, enabling near real-time access to up-to-date information for reporting and analysis. The overarching goal is to create a data pipeline that reliably extracts, transforms, stores, and makes data available for analysis with minimal human intervention.

The system is designed to automate the entire data lifecycle, from the extraction of raw data to its transformation into structured formats, storage in a data lake, and final usage for analytics. Automation was a critical requirement, as the high frequency and volume of data generated by the ovens demanded a process that could run continuously without manual oversight. Additionally, ensuring that the data is always current and accessible for users was a priority, which required careful orchestration and scheduling of data extraction and processing tasks.

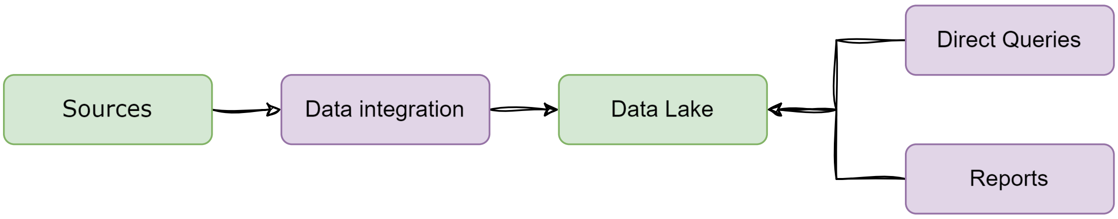

Figure 1.1: System general flow.

The primary sources of data include two distinct databases: **PostgreSQL**, which stores general operational data about the ovens or users' management data, and **MongoDB**, which holds IoT data from sensors, alarms, and other event-driven information. Since these databases have different structures and serve different purposes, the solution had to accommodate specific workflows for each.

For the PostgreSQL database, AWS Glue was selected as the main Extract-Transform-Load (ETL) service. Glue is a fully managed, serverless ETL tool that simplifies data preparation by running Python or Scala scripts without the need for managing servers. In this context, Glue is responsible for connecting to the PostgreSQL database, extracting the necessary tables, transforming the data into a columnar file format, and loading it into Amazon S3 for long-term storage. One of the key challenges was ensuring that the system did not reprocess previously extracted data during subsequent extractions. This issue was addressed using Glue's Bookmarks feature, which tracks the progress of each job by recording the last data row processed. When the ETL job runs again, it starts from where the last job left off, ensuring only new data is ingested.

For the MongoDB database, which stores unstructured IoT data, the extraction process required a custom approach, as Glue's Bookmarks feature does not support MongoDB. To address this, the system leverages AWS Lambda, a serverless computing service that runs code in response to specific events. A master Lambda function orchestrates multiple worker Lambdas, each of which processes data from a specific set of devices (ovens). This system distributes the workload efficiently, allowing for a scalable and flexible data extraction pipeline. Each worker Lambda extracts, filters, and transforms the data, then formats it in Parquet and stores it in Amazon S3. By employing this distributed architecture, the system ensures that even large volumes of IoT data are processed efficiently and in parallel.

The extracted data is stored in a data lake on Amazon S3, organized into three distinct layers:

- raw,
- curated,
- and analytics.

These layers reflect the level of transformation and aggregation applied to the data. In the raw layer, data is stored in its original form, directly after extraction, without any significant transformations. The curated layer includes data that has undergone partitioning, formatting, and compression to optimize performance for querying. Finally, in the analytics layer, data is pre-aggregated to facilitate specific use cases, such as recurring reports or complex queries, improving the efficiency of downstream analytics.

To streamline the management of this multi-stage process, AWS Step Functions are used to orchestrate the entire workflow. Step Functions allow the system to define and automate the execution of each task in the pipeline, ensuring that each job runs in the correct sequence and avoiding conflicts between components. Additionally, the frequency of execution can be easily configured, allowing for flexible scheduling of data extraction and processing based on business requirements.

Once the data is processed and stored in S3, it is cataloged using the AWS Data Catalog. The Data Catalog consolidates metadata for all the tables and files stored in S3, making it easier for other AWS services to reference and query the data. This unified metadata management system allows users to interact with the data seamlessly, without needing to define the underlying storage paths or configurations manually.

For querying and analyzing the data, the solution integrates two key tools: AWS Athena and AWS Quicksight. AWS Athena is an interactive query service that enables users to run SQL queries directly on the data stored in Amazon S3, leveraging the metadata defined in the Data Catalog. This provides a powerful tool for on-demand analysis without requiring the setup of additional databases or data warehouses. AWS Quicksight, on the other hand, is a BI tool that allows users to create interactive dashboards, visualizations, and reports. By integrating Quicksight, the system enables non-technical users to explore the data, generate insights, and produce business reports with an intuitive interface.

In conclusion, the proposed solution offers a fully automated, scalable, and flexible architecture for managing and analyzing the data generated by industrial ovens. It leverages advanced cloud services to ensure that data is continually updated, efficiently processed, and readily available for users, while minimizing the need for manual intervention. This approach not only improves the overall efficiency of data management but also enhances the ability to derive meaningful insights from large volumes of industrial data.

1.5 Outcomes

The results of the project were highly positive, with the implemented system proving to function effectively and reliably. During the testing phase, a snapshot of the production database was used, allowing the entire data ingestion pipeline to be validated and the system's performance to be tested in a realistic environment. Specifically, the initial ingestion of IoT data was completed using a snapshot of the production database up to September 2, 2024, ensuring that all historical data was successfully loaded into the data lake. Once ready for use with the live production databases, the transition can be easily achieved by running a pre-configured script that adapts the custom bookmarks, updates the database credentials and connection settings, and activates the scheduler that automates the regular execution of the system.

One of the key advantages of the implemented solution is the significant reduction in on-demand query times, with an estimated improvement of X% compared to the previous approach. Thanks to the architecture based on Amazon S3, Athena, and AWS Quicksight, queries are now much faster and more efficient, making the data readily available for analysis without substantial delays.

Another significant benefit is the simplification of access to the query platforms. Before the implementation, anyone outside the Digital Experience team had to either obtain access to a specific database or install a GUI client and have the necessary credentials for each database. With the new system, data access is managed directly through the AWS console, provided the user has an account with the necessary permissions to use query or reporting tools. If an employee does not have an AWS account, it can be quickly created by one of the company's software teams.

From a cost perspective, the monthly operational cost of the system has been estimated at around 400$, with fluctuations based on actual usage. The initial ingestion of all historical data, going back to 2015, incurred a one-time cost of approximately 3000$, primarily due to AWS Lambda and AWS Glue services, which were essential for populating the data lake.

While the overall results are highly satisfactory, there is still room for improvement, particularly in optimizing the efficiency of AWS Lambda functions. Reducing the execution times and memory usage of the Lambda functions could significantly lower both the computation costs and the query response times. This would enhance the overall system efficiency and further reduce the data ingestion costs.

1.6 Outline

The subsequent chapters are structured to explore the necessary backgrounds, describe the entire system from a technical point of view, prove the application functionalities through a main use case, perform other experiments and evaluate the system's performance.

[...TO DO...]

2

Background

2.1 DBMS and client GUIs

2.1.1 PostgreSQL

PostgreSQL, often called Postgres, is a robust open-source Relational Database Management System (RDBMS) known for its flexibility, stability, and full compliance with SQL standards. It supports a wide range of advanced features, such as complex queries, foreign keys, views, triggers, and stored procedures. One of its distinguishing characteristics is its support for both structured and semi-structured data, including JavaScript Object Notation (JSON), making it suitable for modern applications that need to manage different data formats. PostgreSQL also offers powerful indexing techniques (e.g., B-tree, GIN, GiST) to optimize query performance, as well as Multi-Version Concurrency Control (MVCC), which enables high transaction throughput without locking issues, allowing multiple users to interact with the database simultaneously.

In the project developed at UNOX S.p.A., PostgreSQL is used to manage key operational data related to industrial ovens. This includes information about companies, devices (ovens), recipes, and device groupings. These datasets are critical for the Data Driven Cooking (DDC) platform, which leverages the stored information to enhance oven performance, improve operational processes, and provide customers with detailed insights.

PostgreSQL is considered a popular choice for many organizations since is open-source, has extensive community support, and is involved in continuous development. It offers a highly customizable solution for both small-scale applications and large enterprise systems.

2.1.2 MongoDB

MongoDB is an open-source NoSQL database designed to handle large volumes of unstructured and semi-structured data. Unlike traditional relational databases like PostgreSQL, described in the previous section 2.1.1, which store data in structured tables with fixed schemas, MongoDB uses a flexible, document-based model. Data is stored in JSON-like BSON (Binary JSON) format, allowing for dynamic, schema-less storage where each document can have a different structure. This makes MongoDB ideal for use cases where data is heterogeneous or rapidly changing, as it does not require predefined schemas or rigid structures like a relational database.

In contrast to PostgreSQL, which excels at managing structured, relational data with well-defined relationships, MongoDB is optimized for handling data that doesn't fit into regular table structures, such as hierarchical or nested data. Additionally, MongoDB offers easy horizontal scaling, distributing data across multiple servers to handle high write loads, making it well-suited for applications that generate large amounts of real-time data.

At UNOX, MongoDB is used to store IoT data generated by the industrial ovens. This includes sensor readings like temperature, humidity, alarms, and detailed records of cooking processes. The flexible schema in MongoDB allows the system to efficiently capture and store a wide variety of data points, which may differ from oven to oven, or even from one cooking session to another. This dynamic approach enables real-time monitoring and analysis of oven performance, helping to ensure that the ovens operate efficiently and providing actionable insights based on the data collected.

2.1.3 TablePlus

TablePlus is a versatile database management tool designed to simplify working with relational databases. It supports a wide range of databases like PostgreSQL, MySQL, and SQLite, making it a go-to solution for developers who need to manage different database systems through a single interface. The most appreciated feature of TablePlus is its intuitive, streamlined interface, which enables users to easily run queries, edit data, and manage tables without needing to rely heavily on complex command-line tools.

The tool also emphasizes security, providing secure connections via SSH and SSL, which is crucial when dealing with sensitive data. It is lightweight and fast, ideal for those who require quick access to data and the ability to make changes efficiently. Other features that distinguish Table Plus among database professionals are the ability to preview and revert changes, multi-step undo, and export options enhance productivity, and reduce the risk of errors, making TablePlus a popular choice among database professionals.

2.1.4 Studio 3T

Studio 3T is a dedicated tool for managing MongoDB databases, offering a range of features designed to make interacting with NoSQL data easier. Unlike general database tools, Studio 3T is optimized for MongoDB's document-based structure. It provides a visual interface for building and executing queries, which is especially useful for those who prefer not to write complex MongoDB queries by hand.

Key features include the ability to migrate data, visualize aggregation pipelines, and easily manage indexes and collections. Studio 3T also allows users to translate SQL queries into MongoDB's query language, making the transition for those familiar with SQL-based databases smoother. It is a powerful tool for handling MongoDB-specific tasks, enabling developers and administrators to efficiently manage large datasets in a more user-friendly environment.

2.1.5 Prisma

2.2 Amazon Web Services

Amazon Web Services (AWS) is a cloud computing platform that provides a broad set of services, including computing power, storage, and networking, through the internet. Businesses and developers use AWS to build and run a wide variety of applications, from simple websites to complex enterprise systems. One of its key benefits is the ability to scale resources up or down based on demand, which eliminates the need for managing physical hardware and allows for greater flexibility. AWS operates on a global network of data centers, offering services that ensure high availability and reliability for users across different regions. AWS enables companies to access the resources they need with minimal upfront investment, through its modular services, such as EC2 for virtual servers and Simple Storage Service (S3) for data storage. AWS also promotes a pay-as-you-go pricing model, where users only pay for the resources they use, making it affordable for both small startups and large enterprises. Its extensive suite of tools allows users to implement everything from basic hosting solutions to advanced analytics and AI models.

Below, some of the AWS tools used to build the system's architecture will be presented.

2.2.1 IAM

AWS Identity and Access Management (IAM) allows you to securely manage access and permissions to AWS resources. With IAM, you can create and control users, groups and roles, assigning detailed permissions to specify who can access what resources and with what privileges. For example, you can grant a user access to certain AWS services or specific actions on a database, while restricting other operations.

IAM uses the concept of "least privilege", allowing you to configure very precise accesses and monitor activities via logs. It is a fundamental tool for ensuring security and control in the AWS infrastructure.

2.2.2 Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (EC2) is a main AWS service that allows to launch and manage virtual servers in the cloud, called instances. EC2 offers a range of instance types to meet different computational needs, allowing users to select the optimal balance of CPU, memory, and storage for their workloads. Therefore, when you create an instance it is possibile to choose the specific hardware configurations that best suit their specific requirements, offering flexibility in performance and cost management. EC2 enables on-demand access to computing power, with the ability to deploy and manage instances without the need for physical servers.

EC2 also integrates well with other AWS services, allowing to build reliable and scalable cloud-based systems. It supports multiple operating systems, such as Linux and Windows, and offers both temporary and persistent storage options, depending on the type of application.

For instance, in this project, an EC2 instance was used to host a snapshot of the production database, allowing safe testing and development with real data without affecting the live environment. This demonstrate how EC2 can help isolate and manage testing or development environments effectively.

2.2.3 Relational Database Service (RDS)

Amazon RDS is an easy-to-manage relational database service optimized for total cost of ownership. It is easy to configure, use and scale according to demand. Amazon RDS automates several database management tasks such as provisioning, configuration, backups and patching. Amazon RDS allows customers to create a new database in some minutes and offers the flexibility to customize databases to their needs by choosing from 8 engines and 2 deployment options. Customers can optimize performance with features such as Multi-AZ with two readable standbys, optimized writes and reads, and AWS Graviton3-based instances, with a choice of pricing options to manage costs effectively.

You have the option to enable automated backups or create manual backup snapshots as needed. These backups can be used to restore your database efficiently and reliably using Amazon RDS's restoration process.

Beyond the security features included in your database package, you can manage access by utilizing AWS IAM to assign user roles and permissions. You can also enhance database protection by placing them within a Virtual Private Cloud (VPC). Moreover, security groups can be used to control what IP addresses or Amazon EC2 instances can connect to your databases.

The table 2.1 compares the database management models between Amazon EC2 and Amazon RDS, highlighting customer and AWS responsibilities for various features.

2.2.4 Lambda

AWS Lambda is a serverless computing service that allows code to be executed without having to manage servers or infrastructure directly. Launched in 2014, Lambda allows developers to execute functions in response to specific events, such as changes to a database, HTTP requests, or file updates in an Amazon S3 bucket.

Feature	**EC2 management**	**RDS management**
Application optimization	Customer	Customer
Scaling	Customer	AWS
High availability	Customer	AWS
Database backups	Customer	AWS
Database software patching	Customer	AWS
Database software install	Customer	AWS
Operating system (OS) patching	Customer	AWS
OS installation	Customer	AWS
Server maintenance	AWS	AWS
Hardware lifecycle	AWS	AWS
Power, network, and cooling	AWS	AWS

Table 2.1: Comparison of Amazon EC2 and Amazon RDS database management models

The serverless model eliminates the need for manual provisioning, management, or scaling of resources, as Lambda takes care of these tasks automatically. Users only pay for the code execution time, measured in milliseconds, and the number of requests, making it a highly cost-efficient option for many applications. The service supports several programming languages, including Python, Node.js, Java, Go, and C#, making it flexible for a wide range of use cases, such as real-time data processing, application monitoring, and automation of repetitive tasks.

To generate a lambda, first, you create your function by uploading your code and choosing the memory, timeout period, and AWS IAM role. Then, you specify the AWS resource to trigger the function, which can be a particular Amazon S3 bucket, Amazon DynamoDB table, or Amazon Kinesis stream. When the resource changes, Lambda will run your function, launching and managing the compute resources as needed to keep up with incoming requests.

2.2.5 Glue

AWS Glue is a fully managed extraction, transformation and loading service, designed to facilitate the preparation and integration of data for analysis. AWS Glue automates the processes of data discovery, cataloguing, cleaning, transformation and movement between different sources such as data lakes, relational databases and other storage resources. The service is designed to simplify the work of preparing data for analysis and modelling by eliminating the need to configure and manage servers. The three main features offered by Glue are:

Jobs ETL

ETL Jobs in AWS Glue are the main operating units that perform the extraction, transformation and loading processes. An ETL job reads data from a source, transforms it as required (such as cleaning, merging or format conversion) and loads it into a destination, such as a data warehouse or data lake. AWS Glue automatically generates Scala or Python code to perform these operations, but also offers the possibility of customising scripts. These jobs can be executed on demand or scheduled at regular intervals, integrating with other AWS resources.

Data Catalog

The AWS Glue Data Catalogue is a centralised metadata repository that organises and manages information on datasets from different sources. It stores data schemas, formats and partitions, facilitating access and queries via tools such as Amazon Athena and Amazon Redshift, without requiring manual configuration.

Crawler

AWS Glue Crawlers automate data discovery and cataloguing by analysing sources to automatically identify schemas and partitions. They update or create tables in the data catalogue, reducing manual work and simplifying metadata management.

In summary, AWS Glue, through the use of Jobs ETL, Data Catalogue and Crawler, provides a powerful and scalable platform for large-scale data management, optimising workflows and ensuring easy integration with other AWS services.

2.2.6 Athena

Amazon Athena is an interactive query service that allows data analysis directly on files stored in Amazon S3 using standard SQL. Athena is a serverless solution, which means that it does not require the management of infrastructure or servers: users only pay for the queries executed, based on the volume of data processed.

The service is optimised to work with large datasets and common data formats such as CSV, JSON, Parquet and Apache ORC[1], Apache Iceberg[2] allowing structured and semi-structured information to be analysed efficiently. Thanks to the AWS Glue Data Catalogue described in the section 2.2.5, Athena can easily access previously defined metadata and schemas, reducing the time needed for data preparation.

Athena lends itself well to ad-hoc data lake analysis, reporting and data monitoring scenarios, without having to load data into a traditional database. Its features make it ideal for big data analytics environments, where speed of execution and ease of use are crucial, and it integrates well with other AWS services, such as Amazon QuickSight for data visualisation.

2.2.7 QuickSight

Amazon QuickSight is a cloud-based BI service from AWS, designed to create dashboards, interactive visualisations and reports from large volumes of data. QuickSight empowers users to analyse data in a simple and intuitive way, providing tools to create charts, tables and advanced visualisations that help make data-driven decisions. The tool supports two modes of data access:

- SPICE (Super-fast, Parallel, In-memory Calculation Engine) mode, which uses an in-memory engine for fast analysis performance and is available for a fee,
- and Direct Query mode, which is free but generally slower, as it queries directly on data sources.

QuickSight allows dashboards to be shared with other users and offers support for access from mobile devices, making it a flexible tool for real-time data analysis and visualisation.

Finally, one of QuickSight's most innovative features is its integration with generative artificial intelligence, which allows analyses to be built quickly via natural language prompts. Users can enter simple questions or queries in natural

[1] `https://orc.apache.org/`
[2] `https://iceberg.apache.org/`

language and get automatically generated charts, tables and insights. This makes the analysis of datasets more accessible even for those without technical expertise, speeding up the decision-making process.

2.2.8 Step Functions

AWS Step Functions is a fully managed workflow orchestration service that allows different AWS services to be coordinated in sequential or parallel workflows. Using AWS Step Functions, complex processes can be defined as states, with each step in the workflow representing an activity, such as executing a job, waiting for an event or calling the API of any AWS service, such as AWS Lambda, Amazon S3 or AWS Glue.

The service uses a visual representation based on a state machine model, allowing users to build and monitor workflows intuitively. Alternatively, workflows can be defined via Amazon States Language (ASL), a JSON language that allows the user to specify the behaviour of each state, conditions for passing between states, and error and exception handling criteria.

AWS Step Functions is ideal for automating distributed processes such as data pipelines, application management and microservice orchestration. It also enables workflows with error handling, retries and conditional operations, ensuring process reliability and resilience. Its extensive support for long-running tasks make it a powerful tool for coordinating the execution of complex tasks in a secure manner.

2.2.9 Event Bridge

AWS EventBridge is a fully managed event routing service, designed to connect applications using real-time event streams. It enables the creation of event-based architectures, where various services or applications automatically react to events generated by other AWS applications or services. Each event represents an action or change of state, such as changes in a database or updates in an S3 bucket, and is routed to appropriate destinations according to predefined rules.

Specifically, the system built uses EventBridge Scheduler, a feature that allows events to be scheduled at regular intervals or at specific times. This service is useful for performing recurring tasks or planned actions, such as starting jobs on AWS Lambda or other resources. In practice, the EventBridge Scheduler functions as a serverless 'cron', allowing time-based workflows to be automated without having to manage a dedicated infrastructure.

2.3 File formats

2.3.1 Apache Parquet

Apache Parquet[3] is an open-source, column-oriented data file format designed for efficient data storage and retrieval. It provides high-performance compression and encoding schemes to manage complex datasets in bulk and is supported by many programming languages and analytics tools. Initially used exclusively in the Hadoop ecosystem, Parquet is now employed by platforms such as Apache Spark[4] and various cloud services to meet the demands of data warehousing.

Parquet's main features include:

- **Columnar storage**: Unlike row-based formats like CSV or Avro, Parquet stores the values of each column next to one another, allowing for better compression and faster querying, especially when accessing only a subset of columns.
- **Self-describing**: Each Parquet file contains metadata, such as the schema and structure, facilitating interoperability between services that write, store, and read Parquet files.
- **Efficient compression**: By leveraging the fact that columnar data tends to be of the same type, Parquet achieves more effective compression than row-based formats. This reduces storage needs and accelerates data transfers.
- **Flexible encoding schemes**: Parquet supports various encoding schemes, such as Run-Length Encoding (RLE), Dictionary Encoding, and Delta Encoding, which further optimize compression and performance.
- **Schema evolution**: Parquet allows schemas to evolve over time by adding or removing fields without affecting existing data, making it ideal for dynamic environments.

[3]`https://parquet.apache.org/`
[4]`https://spark.apache.org/`

Row Format vs. Columnar Format

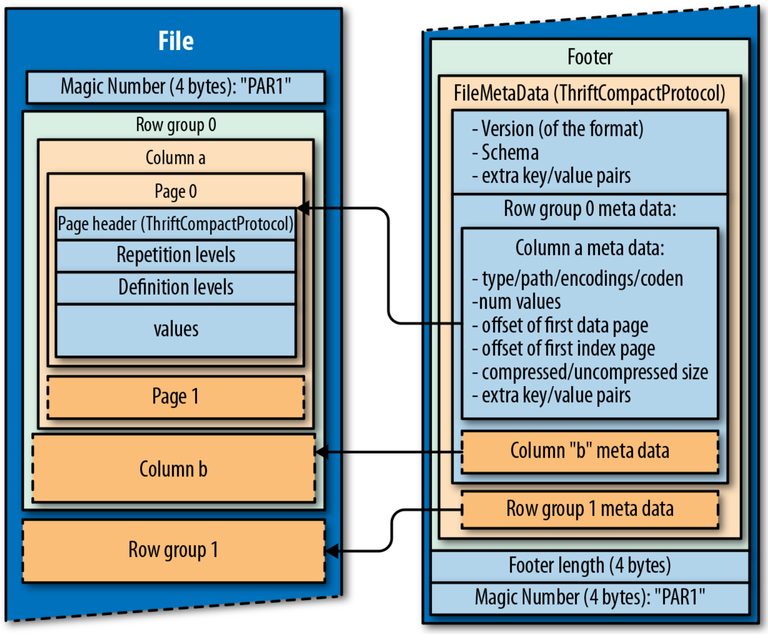

Figure 2.1: Parquet File Layout

As you can see in figure 2.1, each Parquet file contains a header, one or more data blocks, and a footer. The data itself is stored in the data blocks, while the footer holds metadata about row groups, columns, the Parquet format version, and a 4-byte magic number. The data is organized into:

1. **Row groups**: These logically partition the dataset into rows. Each row group contains a column chunk for every column in the dataset. Row group size can be pre-configured: larger groups improve sequential I/O but need more buffer memory. A recommended size is between 512 MB and 1 GB.

2. **Column chunks**: Each column chunk represents contiguous data for a specific column within a row group.

3. **Pages**: A page is the smallest indivisible unit in Parquet, used for compression and encoding. The division into pages allows for more efficient compression and parallelized reads. Page size can be pre-configured: smaller data pages allow more precise reads, like single-row lookups, while larger pages reduce space and parsing overhead. The recommended page size is 8KB.

This structure is optimized for analytical queries that only require a subset of columns, reducing I/O by reading less data. Moreover, Parquet files contain statistics like the minimum and maximum values for each column, enabling engines to skip irrelevant data blocks during queries.

Compression Techniques

In Parquet, compression is performed at the column level, supporting various encoding methods, including:

- **Plain encoding**: The default encoding, used when no more efficient method is applicable.
- **Dictionary encoding**: Frequently occurring values are stored in a dictionary, and the data is replaced with the corresponding keys, reducing storage size. This method is applied dynamically when advantageous.
- **Run-Length Encoding (RLE)**: When consecutive values are the same, they are stored as a single value along with their count. Parquet combines bit-packing with RLE to achieve better compression.
- **Delta Encoding**: Instead of storing raw values, the difference between consecutive values is stored, which is useful for sequential data.

Parquet also supports compression algorithms such as Gzip, Snappy, Brotli, and LZO.

Performance Benchmarking

A benchmarking study conducted by Cloudera compares Parquet, Avro[5], and CSV across various tasks [3]. The results are summarized in Table 2.2.

From the table, it is evident that Parquet consistently outperforms Avro and CSV in terms of storage space and query speed. For instance, Parquet files required only 750 MB of disk space for the narrow dataset, compared to Avro's 1 GB and CSV's 4 GB. Similarly, Parquet demonstrated a 97.5% compression ratio on the wide dataset, allowing it to process 3.5 times less data in the same operations, compared to Avro. While Parquet performed well across the board, its efficiency is particularly notable for read-heavy operations, like analytical queries involving group-by and column scans.

Another study on the choice of the most efficient format in the Apache Hadoop system between parquet, orc, avro, CSV, JSON [2] shows that Parquet

[5] `https://avro.apache.org/`

Metric	Parquet	Avro	CSV
Narrow Dataset (3 columns, 83.8M rows)			
Dataset to file format (s)	74	72	45.33
Row count (s)	5.33	5.33	45.33
Group by (s)	9	24	N/A
All data pass (s)	8.33	15.66	N/A
Disk space (MB)	750	1,000	4,000
Wide Dataset (103 columns, 694M rows)			
Dataset to file format (s)	138	180	68
Row count (s)	2.66	45.33	68
Group by (s)	31	54	N/A
All data pass (s)	33.33	140	N/A
Disk space (GB)	5	18	200

Table 2.2: Performance Comparison of Parquet, Avro, and CSV

is better than its competitors from the point of view of storage, all data search, unique row search and sorting. However, it has a similar performance to ORC in the grouping and filtering tasks.

However, Parquet can be slower to write than row-based formats like CSV or Avro, due to the overhead of generating metadata.

Use Cases and Limitations

Parquet is particularly well-suited for scenarios that require efficient compression and fast query performance on large datasets.It is often used for analytical queries that need to access a subset of columns, in data pipelines where schema evolution is crucial, and in cases where efficient data storage and retrieval are key considerations.However, one limitation of Parquet is the potential performance drop when querying full rows in datasets with a very wide structure, typically with more than 100 columns. In such cases, row-based formats like Avro or ORC may offer better performance, as they are optimized for retrieving entire rows.

2.3.2 Apache Iceberg

Apache Iceberg[6] is an open-source format for managing large tables in Data Lakes. Initially developed by Netflix and later open-sourced, Iceberg was designed to overcome the limitations of more traditional table formats such as Apache Hive, which do not scale well in complex and distributed Data Lakes. Iceberg natively supports table management on both Hadoop Distributed File System (HDFS) and cloud storage, such as Amazon S3, Google Cloud Storage, or Microsoft Azure.

The primary goal of Iceberg is to provide a table format that efficiently supports the management of distributed data, allowing for reliable large-scale read and write operations, ensuring data consistency, and facilitating operations such as dataset scanning and version control.

Like Apache Parquet described in the previous section 2.3.1, Iceberg is based on a layered architecture that separates metadata management from the data itself. This separate approach allows for more effective management of read and write operations and enables scalable modification operations. Metadata in Iceberg is crucial for table management. Iceberg uses a snapshot system to keep track of table versions, enabling rollback operations and ensuring full control of changes. The metadata includes information on schema, partitioning, data files, and delete files. Each data modification generates a new snapshot that can be consulted to view the table's evolution over time.

Since snapshots store details about the snapshot's timestamp, partition, and relevant data files, Iceberg supports versioning. Therefore, a snapshot provides a view of the entire dataset at a specific point in time.

Iceberg Features

Iceberg tables offer several benefits compared to other formats traditionally used in the data lake, including:

- **Schema evolution**: Supports commands to add, drop, update, or rename columns without causing side effects or inconsistency.
- **Partitioning**: Iceberg introduces dynamic and flexible partitioning. Traditionally, Data Lakes use file path-based partitioning, which can be inef-

[6] `https://iceberg.apache.org/`

ficient and difficult to manage. Instead, Iceberg implements a metadata-based partitioning strategy that reduces overhead and improves query performance. Two key improvements in partitioning are:

  - *Partition evolution*: Facilitates the modification of partition layouts in a table, such as data volume or query pattern changes, without needing to rewrite the entire table.
  - *Hidden partitioning*: Iceberg automatically handles partitioning by transforming column values (e.g., converting `event_time` into `event_date`) without requiring user-maintained partition columns. This allows queries to benefit from partitioning transparently, hiding the physical layout from producers and consumers. The separation of physical and logical partitioning enables the flexible evolution of partition schemes over time, improving performance without costly migrations.

- **Time travel**: Allows users to query any previous version of the table to examine and compare data or reproduce results using past queries.
- **Version rollback**: Quickly fixes discovered issues by resetting tables to a known good state.
- **Increased performance**: Ensures data files are intelligently filtered for faster processing through advanced partition pruning and column-level statistics.
- **Transactional consistency**: Helps users avoid partial or uncommitted changes by tracking atomic transactions with ACID (Atomicity, Consistency, Isolation, Durability) properties.
- **Table optimization**: Optimizes query performance to maximize the speed and efficiency with which data is retrieved. The main optimization is file compaction. This is particularly useful in Data Lakes, where data is often written in small files, causing fragmentation that can negatively impact query performance. Periodic compaction reduces the number of fragmented files and improves efficiency.

ACID Operations

One of the key contributions of Apache Iceberg to the Data Lake world is the implementation of support for ACID operations, a crucial innovation to ensure the consistency and correctness of data operations.

- **Atomicity**: Changes to the data are treated as atomic transactions, meaning multiple operations are applied as a block, ensuring they are either fully completed or not applied at all. This is particularly useful in scenarios requiring concurrent writes.
- **Consistency**: Thanks to advanced metadata management, Iceberg ensures that all operations on a table remain consistent, even in distributed environments.
- **Isolation**: Iceberg provides transaction isolation, ensuring that concurrent operations do not interfere with each other. Using a snapshot system, each read or write operation accesses a consistent version of the table.

These features make Iceberg ideal for managing highly reliable and resilient data pipelines.

Table Layout

The Iceberg table format offers similar capabilities to SQL tables in traditional databases. However, unlike such datasets, Iceberg operates in an open and accessible manner, allowing multiple engines (such as Spark, Dremio[7], etc.) to work on the same dataset.

[7] `https://www.dremio.com/`

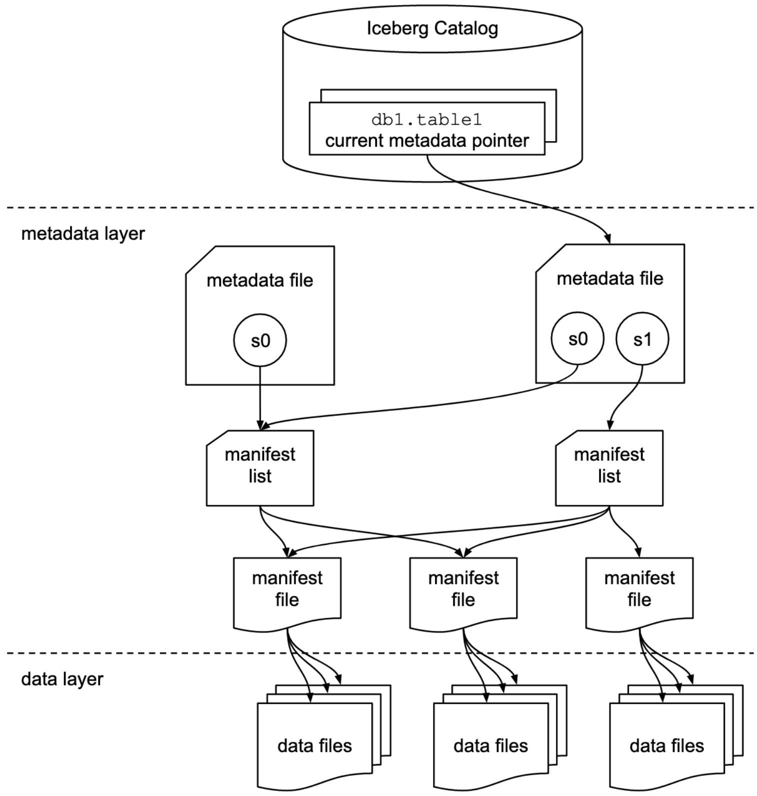

Figure 2.2: Iceberg Table Layout

Through metadata files, Iceberg tracks point-in-time snapshots by maintaining all deltas as a table. Each snapshot provides a full description of the table's schema, partition, and file information. Additionally, Iceberg intelligently organizes snapshot metadata hierarchically, enabling data processing engines to efficiently apply changes without redefining all dataset files, ensuring optimal performance at data lake scale.

The Iceberg table architecture consists of three layers:

- **The Iceberg catalog**: This is where services find the location of the current metadata pointer, which helps identify where to read or write data for a given table. Each table's references or pointers exist here, identifying the current metadata file.

- **The metadata layer**: This layer consists of three components: the metadata file, the manifest list, and the manifest file. The metadata file contains information about a table's schema, partition details, snapshots, and the current snapshot. The manifest list includes a list of manifest files along with information about the manifests that form a snapshot. Manifest files track data files and include other details and statistics about each file.

- **The data layer**: Each manifest file tracks a subset of data files, which contain information about partition membership, record count, and column upper and lower bounds.

3 System Development

3.1 System Architecture Design

The design of the system was a crucial part of my internship, consuming approximately 20% of the total hours dedicated to the project. It was crafted to meet the specific requirements of the business while also taking into account the current context and future needs of the company. UNOX, having collaborated with AWS since 2019, leverages innovative AWS technologies to maintain a competitive edge. For this project, we were guided by an AWS Solution Architect and an Enterprise Account Manager, who played a pivotal role in helping us build a cutting-edge system that adheres to AWS's Well-Architected Framework principles, ensuring Operational Excellence, Reliability, Performance Efficiency, Security, Cost Optimization, and Sustainability.

The AWS Solution Architect, in particular, provided detailed comparisons of the various AWS technologies available, enabling key decision-makers such as myself, the team leader, and the company's CTO to gain a comprehensive understanding of the potential architectures we could build. During the early stages of my internship, we held recurring meetings to explore potential solutions that best aligned with our specific requirements and the nature of UNOX's business operations. These discussions helped us identify the ideal path forward, though the initial solutions inevitably evolved as we encountered and addressed practical challenges during the system's development.

A core aspect of the system design was to ensure a clear separation between the storage layer and the business analytics layer, effectively decoupling data producers (such as operational systems) from data consumers (like reporting and predictive analytics systems). This separation was essential to facilitate data science activities, where a data lake provides a convenient storage layer for experimental data, supporting both the input and output of data analysis and machine learning tasks. The architecture also needed to support autonomous creation and use of data, without the need for coordination between programs or analysts, while at the same time enabling the sharing and re-use of massive datasets through a distributed computational framework.

3.1.1 Data Lake vs. Data Warehouse vs. Data Lakehouse

The initial idea for this project was to implement a data lake, a more innovative and flexible approach compared to traditional data warehouses. While both are used for storing large volumes of data, they serve different purposes and have distinct architectural characteristics.

Traditionally, a **data warehouse** is optimized for structured data, meaning data is cleaned, organized, and stored in a predefined schema, making it ideal for business reporting and analytics. However, data warehouses typically involve high setup and maintenance costs, and they require significant preprocessing to ensure data consistency before it can be used for analysis. Hence, they suffer from limited flexibility for advanced analytics, including machine learning tasks.

A **data lake**, on the other hand, offers a more flexible storage solution. It is capable of storing vast amounts of both structured and unstructured data in its raw form, allowing for greater adaptability. This means that data lakes are not bound by rigid schemas and can accommodate data from diverse sources without the need for heavy preprocessing. Data lakes are particularly suitable for data science, machine learning, and exploratory analysis, as they allow analysts and data scientists to directly interact with raw data, creating an environment where experimentation can thrive. A detailed analysis of the differences between data warehouses and data lakes is given in Table 3.1 [4].

For this project, the data lake was selected because it enables autonomous data handling, and data producers do not need to coordinate directly with consumers. It also provides a shared storage framework, which facilitates collaboration between teams and allows for the re-use of large datasets without duplication or complex integration.

Parameters	Data Warehouse	Data Lake
Data	Focuses only on business processes	Stores everything
Processing	Highly processed data	Mainly unprocessed data
Type of Data	Mostly in tabular form and structured	Can be unstructured, semi-structured, or structured
Task	Optimized for data retrieval	Share data stewardship
Agility	Less agile, has a fixed configuration	Highly agile, can be configured and reconfigured as needed
Users	Widely used by business professionals and business analysts	Used by data scientists, data developers, and business analysts
Storage	Expensive storage for fast response times	Designed for low-cost storage
Security	Allows better control of the data	Offers less control
Schema	Schema on writing (predefined schemas)	Schema on reading (no predefined schemas)
Data Processing	Time-consuming to introduce new content	Helps with fast ingestion of new data
Data Granularity	Data at the summary or aggregated level of detail	Data at a low level of detail or granularity
Tools	Mostly commercial tools	Can use open-source tools such as Hadoop or MapReduce

Table 3.1: Comparison between Data Warehouse and Data Lake

However, while data lakes excel in flexibility, they can sometimes suffer from challenges related to data governance, data quality, performance, and metadata management. As a result, organizations have adopted a two-tier architecture: storing data in lakes and then moving curated data to warehouses for structured analytics. The two-tier model (data lakes + data warehouses) introduces new complexities, including reliability and cost issues. Indeed, maintaining consistency between the lake and warehouse is complex and storing data in two places and running ETL processes increase costs.

This is where a more modern architecture, the **Data Lakehouse** [1], promoted by Databricks[1], a cloud platform built on Apache Spark that enables unified data analytics, machine learning, and big data processing, comes into play.

The lakehouse architecture combines the best features of both data lakes and data warehouses. It retains the ability of a data lake to store raw and

[1] `https://www.databricks.com/`

semi-structured data while incorporating some of the data management and performance optimization features of a data warehouse. This hybrid approach allows for real-time analytics and ACID transactions on large datasets by adding structured layers of metadata to the raw data. In fact, the system developed in this project can be interpreted as a data lakehouse. Although it functions primarily as a data lake, we have integrated several layers that provide pre-aggregated tables in Parquet or Iceberg formats, which are directly usable for advanced analysis. These formats not only offer significant performance benefits through better compression and faster query times but also enable ACID operations. This structured approach allows us to maximize the system's potential for advanced business intelligence while maintaining the flexibility and scalability inherent in a data lake.

3.1.2 The whole system

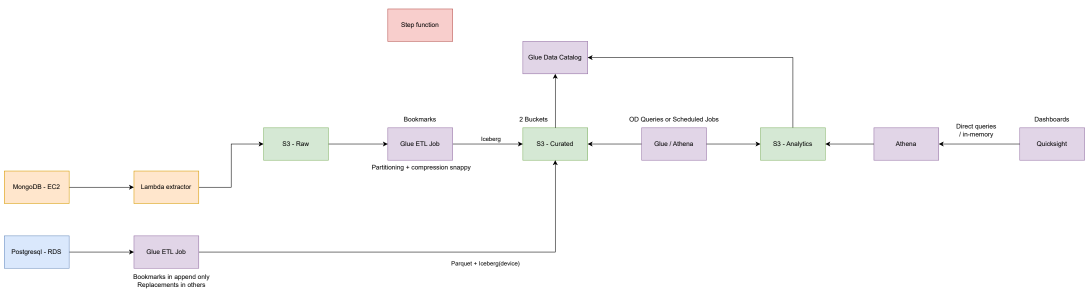

Figure 3.1: The Whole System

3.2 Data Sources

3.3 Data Ingestion

3.4 Data Integration

3.5 Data Cataloguing

3.6 Direct Queries

3.7 Report creation

3.8 Orchestration and Scheduling

4

Experiments and Analysis

4.1 Alarms Report use case

5

Conclusions and Future Works

References

[1] Michael Armbrust et al. "Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics". In: *Proceedings of CIDR*. Vol. 8. 2021, p. 28.

[2] Vladimir Belov, Andrey Tatarintsev, and Evgeny Nikulchev. "Choosing a Data Storage Format in the Apache Hadoop System Based on Experimental Evaluation Using Apache Spark". In: *Symmetry* 13.2 (2021). ISSN: 2073-8994. DOI: 10.3390/sym13020195. URL: https://www.mdpi.com/2073-8994/13/2/195.

[3] Don Drake. *Benchmarking Apache Parquet: The Allstate Experience*. 2016. URL: https://blog.cloudera.com/benchmarking-apache-parquet-the-allstate-experience/.

[4] Athira Nambiar and Divyansh Mundra. "An overview of data warehouse and data lake in modern enterprise data management". In: *Big data and cognitive computing* 6.4 (2022), p. 132.

Acknowledgments