University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master degree in Computer Science

Feeding events to business intelligence via an event bus

Master thesis

Supervisor
Prof. Tullio Vardanega

Candidate
Alessandro Cavaliere
Student ID 2076771

Academic year 2023-2024

"You might not think that programmers are artists, but programming is an extremely creative profession. It's logic-based creativity"

— John Romero

Acknowledgements

First and foremost, I would like to express my sincere gratitude to Prof. Tullio Vardanega, not only for his guidance and support throughout the writing of this thesis but also for the valuable knowledge and lessons he has imparted to me.

I am deeply grateful to my parents for their unwavering financial support and the freedom they gave me during my years of study, never asking anything of me but my happiness and peace of mind.

I would also like to extend my heartfelt thanks to my siblings and relatives who have consistently shown interest in my academic journey, offering their support and trust along the way.

A special thank you goes to the friends with whom I have shared this academic journey. I will always cherish the fun moments and adventures we have had together.
Likewise, I would like to thank all the other friends who, in one way or another, have been part of this experience.

Lastly, I am immensely grateful to Sabrina for her unconditional support and help throughout these years of study.

Padova, September 2024 Alessandro Cavaliere

Summary

Event-driven architectures use event buses to forward business events to consumers. Such channels witness all the stuff that's happening in the system. The way in which such events happen, as long as the history, is completely lost after the events have been delivered to consumers.
The purpose of the project is to track what goes through the event buses, by logging business events in a structured way. Such a system, commonly known as audit trail, logs all the generated events and it provides functionalities to query the data warehouse, in order to extract meaningful analytics and insights, as long as the history of what happened. The ultimate goal is to provide such knowledge and features to various business units, aiming to help them while carrying out their duties.

This document is organized as follows.
Chapter 1 introduces the concept of business intelligence and its relative process, reviewing the *state-of-the-art* technologies and solutions. It then focuses on the current situation of THRON, a Software-as-a-Service company, for what concerns business intelligence, analyzing the current issues, limitations and opportunities in the adoption of such an approach within the organization.
Chapter 2 shifts the focus to breaking down the core aspects of the business intelligence process into more manageable stages. It outlines the system requirements, explains which topics were prioritized, and provides justifications for excluding certain aspects of business intelligence.
Chapter 3 presents the system architecture, detailing each component's functional and technical aspects and explaining how they address the previously identified requirements. It also includes the testing methodologies used and an evaluation of the system's overall performance.
Chapter 4 ties the proposed solution back to the original objectives, addressing any limitations and unsatisfied requirements. It also provides examples of use cases and discusses potential ways the system can improve business intelligence in THRON, alongside suggestions for future enhancements and new functionalities.

Contents

List of Figures

List of Tables

Chapter 1

Problem statement

The analysis focused on the nature of the problem that necessitated the development of an audit trail platform system. It identified the state-of-the-art solutions available at the time and the main concerns associated with those existing solutions.

1.1 Business Intelligence

1.1.1 Definition

Business intelligence (BI) consists of a set of technology-driven processes which focuses on analyzing data to extract valuable information. Such insights can help the board of directors, managers, and employees in making reasonable business decisions. Organizations gather data from both internal and external sources, prepare them for analysis, with the final aim to extract insightful knowledge. They run queries on the elaborated data and create visual representations, such as dashboards and reports, to make analytical results accessible for daily decision-making processes and long-term strategic planning.

The main goal of BI processes is to enhance business decisions, allowing organizations to boost revenue, improve operational efficiency, and build a competitive advantage over the other competitors. To achieve this, BI combines analytics, data management, and reporting tools with various methodologies for data management and analysis.

Many different data sources can be used to feed such a process. Such sources include both internal (stored within the organization) and external data.
Internal data sources may include:

- **Operational data**: transaction records, sales data, inventory levels and production metrics. These records can be used to monitor daily operations, track performance, and manage resources;
- **Financial data**: revenue, expenses and profit margins. Such information can be useful when analyzing financial health, creating forecasts, and managing budgets;

- **Customer data**: customers' demographics, purchase history and feedback. These records can be used to understand customer behavior, eventually improve customer service, and enhance marketing strategies;
- **Employee data**: Human Resources (HR) records, performance evaluations, payroll data. Such data can be useful to manage workforce, track productivity, and develop HR strategies;
- **Logistics and supply chain data**: suppliers performance, shipment records, order fulfillment times. This information could be needed in order to optimize supply chain operations, improve efficiency, and reduce costs.

External sources may be:

- **Market data**: find market trends, competitor analysis, industry reports. These insights may be exploited in order to understand market dynamics, benchmark performance, and identify opportunities;
- **Social media data**: social media posts, likes, shares, comments. The goal of gathering such data is to monitor brand sentiment, track social trends, and engage with customers;
- **Web data**: track website traffic, user behavior analytics, and Search Engine Optimization (SEO) metrics. The data is used to optimize online presence, improve user experience, and drive online sales;
- **Economic data**: Gross Domestic Product (GDP), inflation rates, unemployment statistics. It may be helpful to assess economic conditions, forecast trends, and adjust business strategies.

Other data sources may include unstructured data or sensor and Internet of Things (IoT) data.
In the first case data can be gathered from emails, customer reviews, chat logs, images, videos, audio recordings. The goal is to provide insights which may help while analyzing customer feedback, extracting information from multimedia contents, improving customer relations, and monitoring brand presence.
The latter case may include sensor readings, equipment status, IoT devices' data, with the aim to use such information to monitor and optimize machines performance, predictive maintenance, and operational efficiency.

Each different organization may have access to a subset of such data sources, therefore it is important to tailor the BI process depending on the core business of the organizations adopting it. Different firms may benefit from different insights, thus it is relevant to recognize which of the available information is worth collecting, based on companies' needs.

To sum up, BI includes data analytics and business analytics. The first provides insights from raw data, aiding informed decision-making; the latter optimizes processes, aligns strategies, and ensures solutions meet business goals, enhancing overall efficiency and competitiveness. Predictive analytics, which consists of making predictions about future outcomes using historical data combined with statistical modeling, data mining techniques and machine learning, may be used as well to support organizations building future strategies.

1.1.2 Process

A BI process involves several steps. The starting point is the raw data gathered from different sources, which should be transformed into meaningful insights that support informed decision-making.
Figure 1.1 depicts the usual steps of such a process.

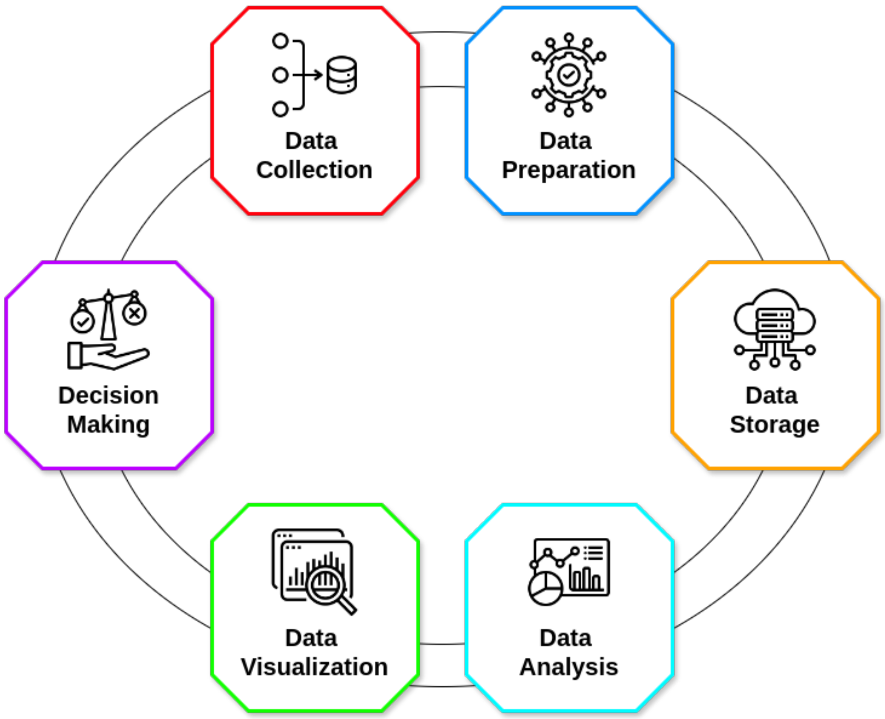

Figure 1.1: The typical steps of the BI process

Data collection

The first step involves the identification of the relevant data sources, both internal and external, that can lead to the extraction of useful information. This helps the development of a business objective, underlying which data is needed to achieve it.
Once such sources have been identified, data gathering can take place by connecting with the various data sources. The goal is to accumulate relevant and comprehensive data that serves as the foundation for the subsequent analysis.

Data preparation

Once gathered, the data must be prepared for the analysis. In this step, the collected data should be cleaned and organized with the aim of ensuring accuracy, completeness and consistency. This sub-process may involve different activities, where the following are the most important ones:

- Removing duplicate entries;

- Removing not-so-relevant records;
- Handling missing values or partial information;
- Transforming and organizing unstructured data;
- Modeling available information in a uniform manner.

Data preparation is a crucial phase, as it prepares the raw data for effective analysis and helps in avoiding misleading results due to errors in the source data.

Data storage

This stage of the BI process entails storing all the data which has been previously obtained and processed into a storage.
Depending upon whether data is processed or not, the naming convention for the destination storage changes. If data is not pre-processed (i.e. the previous step does nothing), then the storage is going to be called a data lake. A data lake is a centralized repository designed to store, process, and secure large amounts of structured, semi-structured, and unstructured data. It is merely a storage that stores mainly copies of raw data. On the other hand, if data is processed, then storage is named data warehouse. Data warehouses are systems which store cleaned and processed data, which can then be used as a source for analytics, operational reporting, and, generally speaking, to cover specific BI use cases. They are also optimized for query performance and allow for efficient data retrieval.
In both cases, the resulting storage is typically centralized, facilitating easy access and management of data, and supporting large-scale data analysis and reporting.

Data analysis

Transformed data can now be processed and analyzed in order to reveal patterns, trends, and gather useful insights with respect to the core business of the organization. To extract useful information from the data warehouse, many analytical approaches and algorithms can be used.
One of the most commonly used approaches is statistical analysis, which means investigating trends, patterns, and relationships using quantitative data. This involves the use of descriptive statistics (summarize the main features of a dataset), inferential statistics (analyze samplings to make predictions about larger populations), correlations and regression analysis (identify relationships between variables), and other techniques to extract meaningful information starting from the structured dataset.
Machine learning (ML) is also used in pattern recognition, as it allows the identification of patterns and clusters within large datasets, revealing insights that might not be immediately obvious. ML models can also help forecasting future trends based on historical data.
Lastly, data mining helps in pattern discovery, predictive modeling, anomaly detection, and classification. By using these techniques, it is possible to uncover hidden insights, forecast future events, detect anomalies, categorize data, and analyze unstructured data.
Many other techniques, and algorithms can be used depending on the nature of the dataset.

Data visualization

Once the data has been analyzed and useful insights have been extracted, they should be presented using visual tools such as charts, graphs, and dashboards. Such visualizations can help to effectively show the results of the previous step to a broader audience, including decision-makers, stakeholders, and, generally speaking, people who are not familiar with the technological tools used in the previous steps.
Overall, this stage can enable the audience to quickly catch key insights and trends. In such a way the found information can be communicated easily, facilitating data-driven decisions afterwards.

Decision making

The final step involves using the visualized data insights to inform and support business decisions. Decision-makers (executives, managers, stakeholders), use these insights to develop strategies, optimize operations, and drive business growth by leveraging on data-driven insights.

1.1.3 Usage

1.1.3.1 Advantages

There are various benefits when using BI to enhance a business organization. The main advantages have already been described previously, as they are intrinsically related with the nature of its definition and the steps of a BI process.
It is still worth it to dig deeper into them and see why firms should exploit the use of such a process to improve their business:

- **Data-driven decision making**: BI allows organizations to make their decisions based on real data and objective analysis rather than assumptions or feelings. Therefore it enhances the accuracy and reliability of decisions, leading to better outcomes;
- **Competitive advantage**: companies gain a deeper understanding of their own organization, their market, competitors, and customers. By using such knowledge, they can identify new opportunities, address weaknesses, and capitalize on their strengths, with the ultimate goal of building and gaining a competitive advantage over the main competitors;
- **Strategic planning**: when developing new strategies, BI allows companies to evaluate the effectiveness of them. Furthermore, it gives the opportunity to explore new opportunities, and align resources with the firm's goals;
- **Operational efficiency**: BI can optimize business processes by identifying bottlenecks, streamlining workflows, and reducing inefficiencies. Data-driven process improvements lead to cost savings, increased productivity, thus enhancing the overall operational efficiency;
- **Performance monitoring**: it provides real-time and historical insights into key performance indicators (KPIs) and metrics, enabling companies to monitor their progress toward their objectives and track performance over time. This facilitates quick course corrections and ensures strategic alignment.

1.1.3.2 Disadvantages

While benefiting from BI, organizations should be aware about the risks related while using such tools. Some cons may include:

- **High initial costs**: BI software can be quite pricey at first, even if the returning value to the organizations can be felt almost immediately. Small businesses might find that cost too high at first, therefore they tend to discard this opportunity. To address such a limitation, organizations can consider cloud-based solutions that can reduce the cost of implementation and maintenance, giving flexibility and scalability to expand their solutions as needed;

- **Lack of context and interpretation**: the tools are designed to provide insights based on data analysis. However, they often lack context and interpretation. This means that final users may not be able to understand the extracted insights provided without additional context. To overcome such an issue, data visualization tools should be extensively used to provide additional context to the data. They can also help users identify trends and patterns that may not be immediately apparent;

- **Risk of data overload**: BI tools can provide users with a large amount of data, which can be overwhelming and difficult to manage. This can lead to users being unable to identify important insights and making poor decisions based on incomplete information. This problem can be mitigated by investing in data management tools (like Oracle Data Management Suite, SAP, Tableau, Microsoft Power BI) that can help manage data effectively;

- **Self-service BI**: it consists of giving organizations the ability to analyze data and create visualizations on their own, without the help of a technical team. It can be more cost-saving, but it needs control and monitoring to prevent a chaotic data environment and conflicting results. Conflicting data interpretations can lead to misguided decisions and wasted resources. To establish a good BI process, data engineering and experience in properly interpreting results are needed. Furthermore, organizations may set up teams to curate data sets in data warehouses in order to avoid inconsistencies;

- **Lack of data governance**: as stated previously, BI tools can generate a large amount of data, which can be difficult to manage and secure. Without proper data governance, organizations may be at risk of data breaches and other security threats. To mitigate these risks, organizations need to establish a solid data governance policy. This should include guidelines for data management, security, and privacy, in addition to routine audits in order to guarantee compliance.

1.1.3.3 Importance and future perspectives

Overall, the role of business intelligence is to enhance an organization's operations by using relevant data. Companies that effectively use BI tools and techniques can convert their collected data into valuable insights regarding their processes and strategies. These insights enable better business decisions that boost productivity and revenue, resulting in accelerated growth and higher profits.

Without BI, organizations may miss the benefits of data-driven decision-making. Instead, executives and employees must rely on factors such as accumulated knowledge, past experiences, intuition, and raw feelings for important decisions. While these methods can yield good outcomes, they carry a higher risk of errors and missteps due to the lack of data support.

Here are some statistics that show how much BI is used in organizations and its perceived importance nowadays. In 2020 the global BI adoption rate was 26%[1], involving mainly well-established organizations (having more than 5000 employees). Such statistic was expected to grow until 33%[2] by the end of 2023, stating that BI is becoming more and more widespread as time goes on. The main goal of using BI technologies for the interviewed organizations is to develop and expand their businesses by finding new revenue streams. Furthermore, the adoption rate increases up to 52%[3] considering only software companies, which is not something unexpected, as such firms are more familiar with the technologies involving BI.
As a result, the expected market size is going to increase widely during the next few years. Figure 1.2 illustrates such a scenario, focusing on the BI market size worldwide.

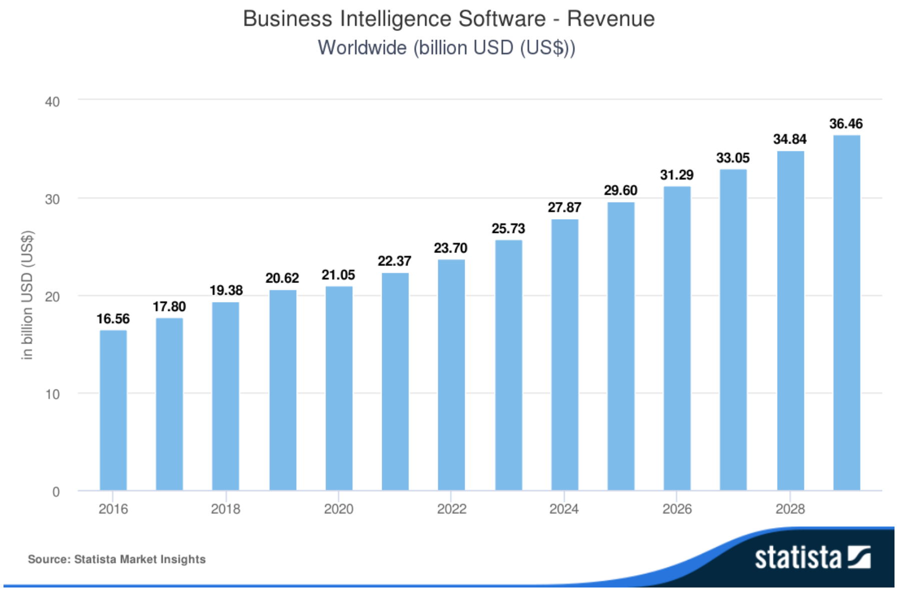

Figure 1.2: Worldwide BI market size growth, from 2016 projected until 2029
Source: Statista

Overall, most of the organizations (almost 90%[4]) consider BI an essential tool for their future development, meaning that they rely heavily on insights derived from big data.

Despite such popularity and perceived importance, most of the organizations still struggle to exploit the insights extracted from the gathered data: 97%[5] of such data is still unused. Lastly, in 2019 the 87%[6] organizations using BI are classified as having

[1] Business Intelligence trends 2020 | Wiiisdom
[2] Data and analytics trends for times of uncertainty | Gartner
[3] The business intelligence landscape | Sisense
[4] The State Of Cloud Business Intelligence, 2020 | Forbes
[5] The power of data-driven enterprise | AWS
[6] Firms' low BI and analysis maturity | Gartner

low analytics maturity, meaning that there is still a long way to go before exploiting BI at its full potential.

To conclude, the presented statistics underline the importance of Business Intelligence in nowadays organizations, not only from the point of view of the adoption rate, but also from a future perspective. Still there is a long way to go, as organizations are struggling to extract the maximum potential from this topic. The gap is set to be covered as the time goes on though.

1.1.4 Technological stack

1.1.4.1 BI process

The BI technological stack consists of several parts, each one addressing a specific topic of the data lifecycle according to the BI process described above. Since business intelligence is a very widespread field, there is no single and unique technology that is able to solve all the problems related to the process as a whole. Each step is addressing a different sub-problem, therefore it needs to be handled by specific technologies.

This section provides an overview of the main technologies that form the backbone of a typical BI stack. The *state-of-the-art* tools for each stage of the process is presented, highlighting their functionalities, advantages, and how they contribute to the overall effectiveness of a BI system.

Data sources

Even if that is not a step of the BI process, it is worth mentioning what are the sources of data from which insights can be extracted. In section 1.1.1, an exhaustive list of data sources has been provided, but only from a functional point of view.

Here is a non-exclusive list of data sources that relate to IT technologies:

- **Internal databases**: mainly relational databases (such as MySQL, PostgreSQL, Oracle) where data is stored in structured tables with predefined schemas. NoSQL databases (like MongoDB, Cassandra) are also in this field: data is stored in an unstructured or semi-structured manner, with various data types. In both cases, database connectors (like JDBC or ODBC) and custom APIs may be used to extract data from such systems;
- **Enterprise applications**: Enterprise Resource Planning (ERP) systems (like SAP, Oracle ERP) provide data related to business processes such as finance, HR, and supply chain. Customer Relationship Management (CRM) systems (such as Salesforce, HubSpot) offer customer interaction data. Customer Data Platform (CDP) systems (like Segment, Adobe Audience Manager) help create a unified and persistent customer database. Human Capital Management (HCM) systems (Workday, Oracle HCM Cloud) provide features to manage an organization's workforce, from recruitment to retirement. Each system provides data to feed BI through dedicated connectors, APIs or Extract Load Transform (ETL) tools;
- **Cloud services**: they entail cloud storage services (like Amazon S3, Google Cloud Storage) where any kind of data (files, images, videos, logs) can be stored, and cloud databases (such as Amazon DynamoDB, Snowflake) which offer data

warehouse solutions from which data can be stored. In any case, ETL tools, APIs and cloud-native data integration tools (like Fivetran, IBM DataStage, Snaplogic Data Integration and Automation) could be used to extract data from such services;

- **Web services**: RESTful APIs (or legacy SOAP APIs) are typically used to provide data from web applications and services. Integration tools (such as IBM API Connect, MuleSoft Anypoint) or middlewares support data extraction from these sources;
- **Social media**: most social networks offer APIs for accessing social media data. APIs and data connectors, along with third-party tools (like Hootsuite), enable the collection and analysis of social media interactions and trends;
- **Web**: web scraping (using tools like Beautiful Soup, Scrapy) is a way to extract data from websites, while behavioral analytic tools (such as Google Analytics) can be used to gather insights of web traffic, user behavior and user patterns;
- **Machine and IoT**: data coming from IoT devices and sensors, including industrial machinery and smart devices, is collected using protocols like MQTT and HTTP. Even cloud-based IoT solutions (Amazon Web Services (AWS) IoT, Azure IoT Hub) offer solutions for managing and extracting data from IoT devices;
- **External providers**: public datasets and market data providers could be relevant sources as they provide financial data about markets and other kinds of data (depending on the core business of the organizations) that can be used to enrich internal data. The final aim is to give more context to internal information, in order to relate internal insights with external ones;
- **Logs and events**: data coming from server logs, application logs, event or message brokers can be very helpful to manage and deeply understand companies' systems by extracting useful information about the usage and health status of them. Such data can be extracted using the ELK Stack (Elasticsearch, Logstash, Kibana) or tools like Apache Splunk.

Figure 1.3 summarizes such sources of data, relating them with real-world technologies and the macro-categories presented in section 1.1.1.

Data collection

The first step involves gathering data from different kinds of sources, as described above. Depending on the type of data source, different technologies and methodologies may be involved.
Generally speaking, ETL tools are pretty useful at this stage. ETL jobs are processes that enable data warehousing and data integration, involving three main steps (extract, transform, load) to manage and move data from source systems to a target database, typically a data warehouse. Each step of the ETL pipeline refers to a different stage of the BI process: Extract maps to data collection, Transform maps to data preparation and Load maps to data storage. Since such steps are pretty common when dealing with BI, as data is the foundation of the whole BI process, ETL tools provide an easy and compact way to extract raw data, transform it and load structured data in one shot.
Apache Airflow is one of the most widely-used ETL tools, as it allows users to define

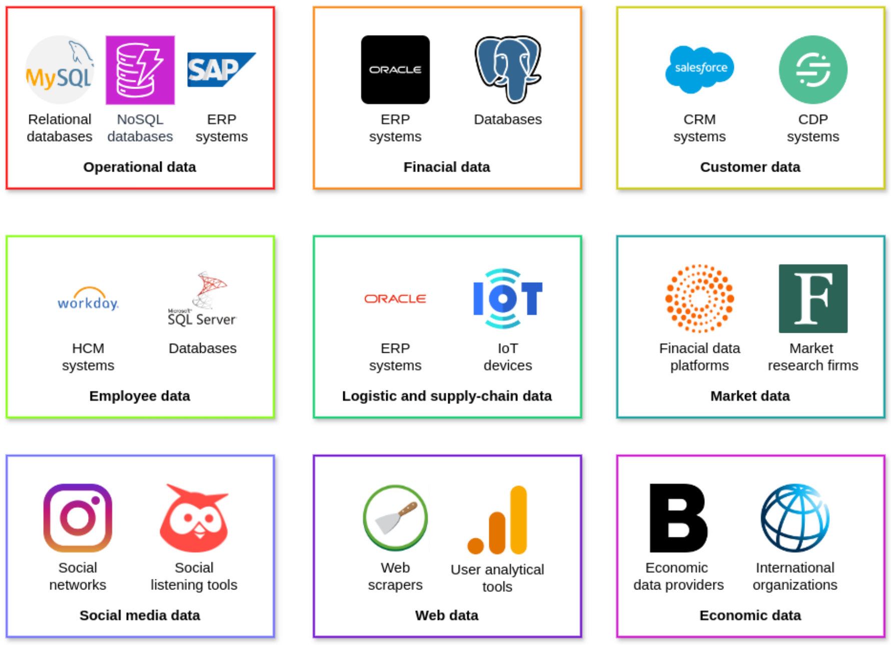

Figure 1.3: Summary of the data sources grouped by category

workflows using directed acyclic graphs of tasks, providing a flexible and scalable way to manage and orchestrate ETL processes. Furthermore, it integrates well with other tools such as Apache Spark to perform data analytics. Other on-premise solutions may be Talend and Oracle Data Integrator.
Cloud-based solutions are Amazon Glue, Amazon Data Pipeline or Azure Data Factory. They offer the same features as on-premise solutions, but they tend to prefer data sources that are within their own cloud environment.
RESTful APIs and database connectors are other ways to attach to data sources and collect data. They are not really used at this stage, as ETL (or ELT) tools allow to integrate them gracefully, without additional effort.
Figure 1.4 represents the concept of an ETL job.

Data preparation

This stage involves cleaning, transforming, and organizing data to ensure quality and consistency.
Data cleaning tools help automate the identification of wrong records and they are also able to perform error correction. The main tool for such a purpose is Alteryx Designer Cloud, a data wrangling tool that uses machine learning to clean and transform data.
For what concerns transformation, Apache Spark and programming languages like Python and R, are used to reshape data, aggregate it, and convert it into a format suitable for analysis.
As stated previously, platforms offering ETL tools usually integrate well with such technologies, allowing users to tailor data preparation to their needs.

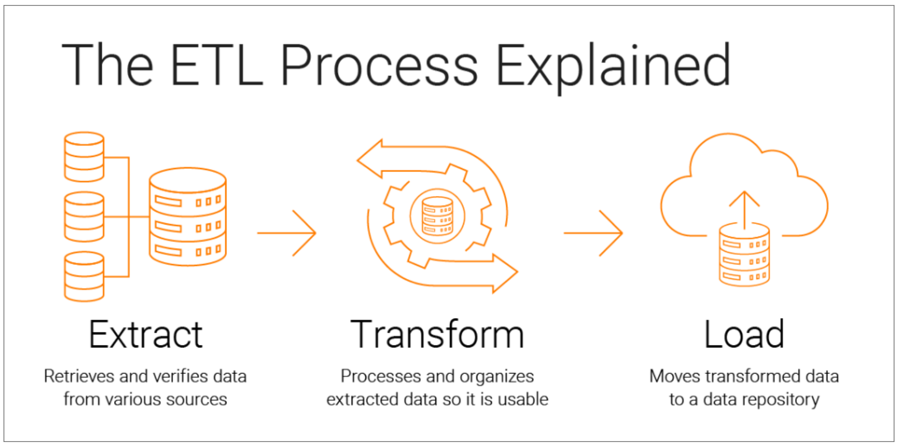
The ETL Process Explained
Extract
Retrieves and verifies data from various sources
Transform
Processes and organizes extracted data so it is usable
Load
Moves transformed data to a data repository

Figure 1.4: Explanation of the purpose of an ETL job
Source: Informatica

Data storage

After preparing the raw data, now it should be saved in a structured format that supports efficient querying and analysis.
Data warehouses like Snowflake or Amazon Redshift are optimized for analytical queries, and they are also capable of handling large volumes of data. They store data in a structured format, often using columnar storage techniques (such as Apache Parquet or Apache ORC), which enhance query performance and compress data, therefore saving storage.
Traditional databases are also used to store structured data.
Cloud storage solutions (AWS S3, Google Cloud Storage) offer scalable storage options and they can handle unstructured data as well.

Data analysis and visualization

These phases involve applying various analytical techniques to extract insights from the stored data and make them visible to the users. As for ETL tools, even for the last steps of the BI process there are tools which are able to provide more functionalities in one single platform. Such tools include the data analysis and data visualization steps.
Analytical tools let users create complex queries, perform statistical analysis, and visualize data. They offer the possibility to create interactive and intuitive visual representations of data.
Furthermore, such tools allow users to drag and drop data fields to create various types of visualizations. Dashboards can be created to view key metrics and performance indicators, enabling users to monitor business performance in real-time.
The more widespread tools are Amazon QuickSight, SAP BusinessObject, Tableau, Microsoft PowerBI, and IBM Cognos Analytics.
Figure 1.5 presents a more exhaustive list of popular tools for these steps.
Statistical software such as R and programming languages like Python provide libraries for advanced statistical analysis and modeling. With respect to analytical tools, they give more flexibility with the cost of additional specific knowledge required.
Machine learning platforms (TensorFlow or cloud-based machine learning solutions)

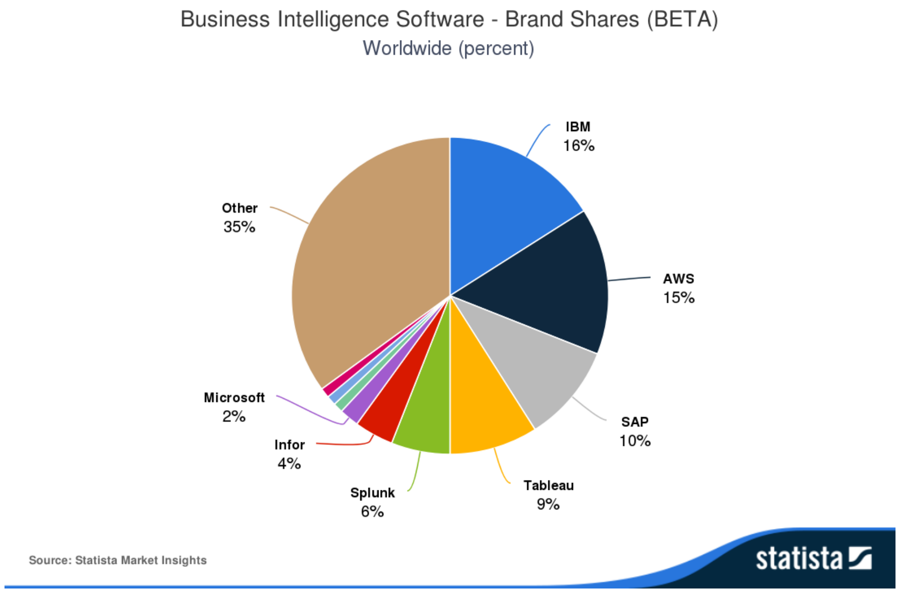

Figure 1.5: BI software share as of 2022
Source: Statista

allow the application of machine learning algorithms to identify patterns, make predictions, and extract hidden insights in the data.

Decision making

The last phase involves using the insights gained from the data to inform and support executives to take business decisions. There are no standout technologies that can support decision making at 100%, but still it is possible to work on some aspects to exploit existing platforms.
Collaboration tools can be used to facilitate communication and share insights among team members and stakeholders. Tools like Microsoft Teams and Slack integrate well with BI platforms to quickly oversee key statistics of metrics derived by the previous steps.
Eventually, decision support systems can be used as they provide frameworks for making informed decisions by integrating data analysis and reports. The aim is to help executives and managers to develop strategies and drive future business growth. For this purpose, SAP BusinessObject and IBM Cognos Analytics stand out as the main platforms that provide a tools for reporting, analytics, score carding, and monitoring of events and metrics in one place.

1.1.4.2 Cloud-based versus on-premise solutions

Cloud-based solutions (offered by providers like AWS, Google Cloud, or Microsoft Azure) offer scalability, flexibility, accessibility, and cost-effectiveness. They allow you to adjust data processing capacity based on demand and budget. Additionally, they reduce hardware, maintenance, and security costs as these are handled by the provider. As a result, the deployment is fast, enabling businesses to start their BI projects efficiently.

On the other hand, on-premise tools need to run on owned servers. Therefore they offer greater control and customization with respect to cloud solutions. They ensure data availability, enhanced security, and compliance with local regulations. However, they require more effort in installation, maintenance, and scaling, which can be costly, complex, and more time-consuming.

Each option has its challenges and there is no single answer on which solution fits best. Cloud-based tools may face integration issues, latency, and vendor lock-in; while on-premise tools involve higher complexity, scalability issues, and greater responsibility for what concerns security and backups.
Choosing the right approach depends upon the specific organization's needs, including volume of data, variety of data sources, and velocity with which data is collected.

1.2 Business Intelligence in THRON

1.2.1 THRON platform

THRON is a Software as a Service (SaaS) company (a company that hosts an application and makes it available to customers over the internet), headquartered in Piazzola sul Brenta, northern Italy. The organization specializes in the development and provision of the THRON platform (referred also as THRON from now onward), its main product. Such a platform is a comprehensive SaaS solution for Digital Asset Management (DAM) and Product Information Management (PIM). As a cloud-based software, it helps businesses centralize and optimize the management and distribution of digital assets and product information, going over the functionalities of traditional DAM and PIM systems.

A DAM system operates as a centralized repository for digital assets, offering several key benefits, as explained in figure 1.6.

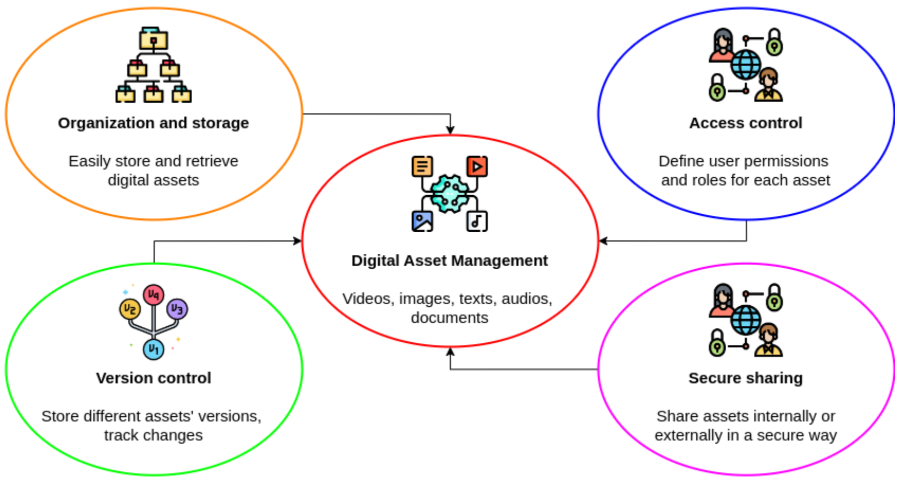
Organization and storage
Easily store and retrieve digital assets
Access control
Define user permissions and roles for each asset
Digital Asset Management
Videos, images, texts, audios, documents
Version control
Store different assets' versions, track changes
Secure sharing
Share assets internally or externally in a secure way

Figure 1.6: Key benefits of a DAM

A PIM system provides a centralized location for collecting, managing, and enriching an organization's products information. This enables the creation of comprehensive product catalogs that can be efficiently distributed to sales and e-commerce channels. Figure 1.7 shows some key advantages.

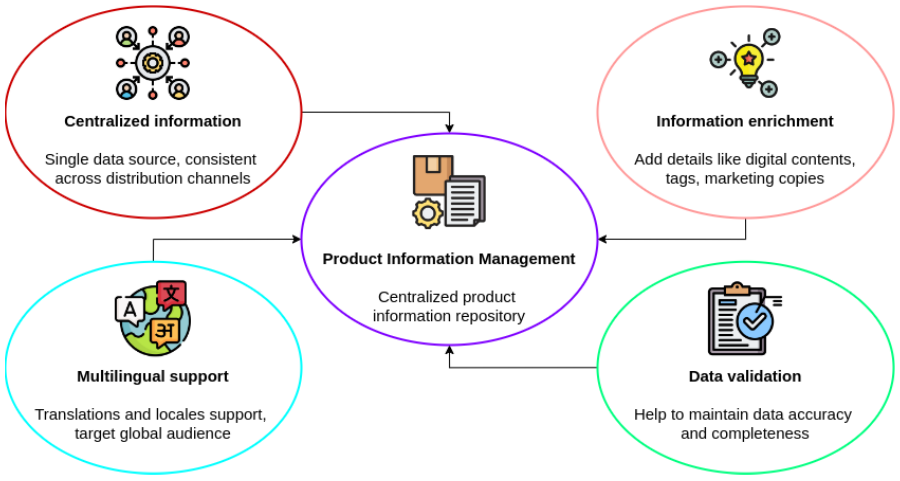
Centralized information
Single data source, consistent across distribution channels
Information enrichment
Add details like digital contents, tags, marketing copies
Product Information Management
Centralized product information repository
Multilingual support
Translations and locales support, target global audience
Data validation
Help to maintain data accuracy and completeness

Figure 1.7: Key benefits of a PIM

Many businesses use separate DAM and PIM systems, which can lead to data silos and overall inefficiency. The THRON platform integrates these functionalities in a unique product, allowing firms to benefit from the key points as figure 1.8 shows.

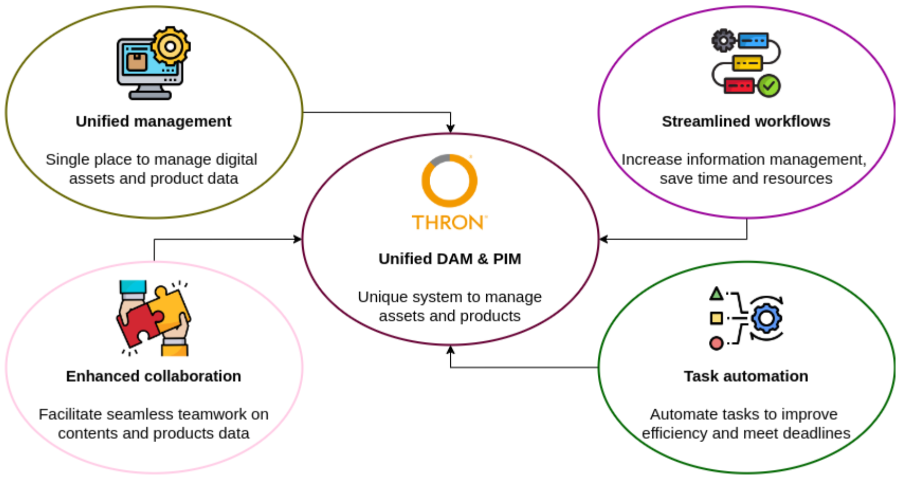
Unified management
Single place to manage digital assets and product data
Streamlined workflows
Increase information management, save time and resources
THRON
Unified DAM & PIM
Unique system to manage assets and products
Enhanced collaboration
Facilitate seamless teamwork on contents and products data
Task automation
Automate tasks to improve efficiency and meet deadlines

Figure 1.8: Main advantages of an unified DAM and PIM

By combining DAM and PIM capabilities, THRON provides a comprehensive integrated solution that addresses the challenges associated with managing digital assets and product information, therefore enhancing operational efficiency and collaboration within organizations.

1.2.2 Integrating BI into the platform

The main vision of THRON, for what concerns BI, is to generate new knowledge about platform usage by using platform data. This process involves the collection of useful data that should be processed and analyzed in order to obtain useful information. Such information then should be made accessible to a wider user base, as information that is accessible to many provides exponentially more value than the same information being accessible only to a few.

In the context of providing a unified DAM and PIM system, THRON can apply BI in several key areas to enhance the functionality and value of the platform.
BI can be applied to track and analyze content usage and engagement across various channels, providing insights into the performance of different types of content and optimizing content strategies. This may ensure the accuracy and consistency of contents and product information, enhancing customer trust by detecting discrepancies and monitoring changes. Additionally, BI can analyze customer behavior and preferences, allowing for tailored marketing strategies that improve customer experience and increase engagement. The same knowledge can be applied to drive future development based on current usage.
Furthermore, it may help identify inefficiencies in process workflows, measure resource usage, and automate routine tasks, with the goal of leading to cost savings and improved productivity. It may support sales and marketing performance analysis, while also being able to monitor usage of features.

The data retrieved through BI can be used for comprehensive reporting and dashboards, predictive analytics, and real-time monitoring. Therefore, the organization can leverage its data effectively, leading to more informed decisions, streamlined operations, and enhanced customer satisfaction thanks to the gathered knowledge that can lead future development of the product.

1.2.3 Gap analysis

1.2.3.1 Current platform BI strategy

Currently, THRON does not use business intelligence or similar processes to extract meaningful knowledge from the platform usage.
The company faces a common challenge experienced by many businesses: while data is generated from various sources, it is not readily usable. In this context, "usable" refers to the ease of access rather than availability. The data is difficult to gather due to the absence of an adequate infrastructure designed for seamless data integration and retrieval.

As of now, collecting data involves manually querying multiple information sources, including databases and other components of THRON infrastructure. This manual process creates significant overhead, reducing the potential to leverage data for generating actionable insights. Such a lack of streamlined data access prevents the company from effectively using data to drive strategic decisions and innovation.
Additionally, there is a notable gap between technical and non-technical teams. Developers, who are accustomed to work with infrastructural components, can manage the data-related tasks easily. On the contrary, data analysts (who may not have technical skills necessarily) and strategic direction, who require access to data for carrying

over their duties, are less familiar with these technical systems. This gap results in complex and time-consuming operations, creating a dependency between technical and non-technical teams. As a consequence, the inconsistent usage of data due to this lack of infrastructure, undermines its value and effectiveness, rendering the data collection process less productive.

To address these issues, it is essential to establish a robust infrastructure that facilitates easy and efficient data access for all stakeholders. By bridging the gap between technical and non-technical teams and streamlining data collection and usage processes, THRON can enhance its ability to generate and use valuable insights to reach the previously explained goals.

Regarding external tools, there are integration with Salesforce for CRM and customer analysis. However, specific details about these integration are not available. In terms of cost management, since the platform is a cloud-based solution backed by AWS infrastructure, AWS resources have been tagged for several years to provide visibility into infrastructural costs. Despite this, it was hard to analyze and forecast expenses, due to the lack of correlation between these costs and the usage of the THRON platform, particularly in tracking business operations. In a SaaS environment, where many resources are shared across various functional domains, assigning costs to specific cost centers becomes complex. This shared-resource model complicates the efforts to isolate and allocate costs accurately, resulting in inability to perform precise financial analysis and forecasting.

Table 1.1 provides a comprehensive summary of all the problems listed above, along with potential solutions.

1.2.3.2 Applying BI to the platform

Currently, there is no established process for extracting actionable insights from the THRON platform's data. As presented in section 1.2.3.1, data about the platform's usage (or coming from other sources) is not retrieved systematically. Implementing such a procedure represents a significant opportunity for improvement.

When talking about platform usage data collection, there is no reference to User Behavior Analytics (UBA). UBA focuses on tracking and analyzing both quantitative and qualitative user data to gain insights in how and why users interact with a product or website. THRON platform already incorporates tools for UBA. In contrast, the objective for THRON, as a first step to integrate BI into its platform, is to track and understand business operations conducted within the platform. Business operations are defined as actions that have tangible side effects on the THRON platform concerning a specific tenant. Examples of such business operations are presented in table 1.2.

This distinction emphasizes that the focus is on understanding actionable business activities rather than user behavior patterns.
Moreover, the components that handle business actions, may generate new business actions to be performed themselves. This is the typical scenario for background business

Problem	**Description**	**Potential solution with BI**
Data not readily usable	Data is generated from various sources, but it is not easily accessible.	Establishment of an infrastructure for seamless data integration and retrieval to support the first stages of the BI process.
Overhead due to manual querying	Collecting data involves manually querying multiple data sources, creating significant overhead.	Implement automated data integration processes to streamline data collection, possibly by defining an ETL job.
Lack of streamlined data access	There is no adequate infrastructure designed for seamless data integration and retrieval.	Develop a centralized data repository (data warehouse) to allow for easy data access and retrieval.
Gap between technical and non-technical teams	There are dependencies and inefficiencies due to differences in handling data between technical and non-technical teams.	Create user-friendly API interfaces and tools for non-technical teams to access and retrieve data in a seamless way.
Inconsistent data usage	The lack of streamlined data access prevents the effective analysis of data and its use for strategic decisions.	Implement standardized data collection and usage processes following the steps underlined by the BI process.
Difficulty in analyzing expenses	Difficulty in analyzing and forecasting expenses due to the lack of cost-usage correlation.	Collect usage data, relate it to resources and use it to feed BI. Perform analysis to extract accurate allocation costs information.

Table 1.1: Summary of problems arose from the current platform state concerning BI

Business operation	**Description**
Creation of an entity	An entity (digital content, product, product attribute, version of a content, channel of a content, locale, etc.) has been created.
Update of an entity	An entity's data, which was already present, has been modified.
Deletion of an entity	An existing entity has been permanently eliminated.
Import or export of product data	A list of products, along with specific data, has been imported (exported) into (from) the platform.
Publication of a new channel for a digital content	A new distribution channel has been created for a specific content.
Update of users permissions and role	The roles and user permissions of a digital asset have been edited.

Table 1.2: Examples of business activities that can be done using THRON platform

operations, such as the import or export of products. In this case, business events do not have a *one-to-one* relationship with user actions. In fact, a single user action can result in the generation of several business events. When talking about the import of product data, several products are being imported into the platform at once. Therefore, it may be the case that many products will be created or updated, leading to the generation of new business events, starting from the original import operation. In such a case, the whole series of events is called a flow. A "flow" is just a sequence of business activities, generated by the platform, that are emitted and handled as a result of a single action performed by the user.

Gathering and analyzing such usage data from the platform could provide valuable benefits in several areas, including:

- **Usage metrics**: collect quantitative data on how often the platform's features are used, segmented by tenants and business actions performed;
- **Usage patterns**: perform analysis to determine if features are being used as originally intended;
- **Retrospective analysis**: comparison of actual feature usage with initial expectations to assess performance and drive future development;
- **Identification of wrong platform usage**: detection of incorrect or sub-optimal use of platform functionalities;
- **Trend analysis**: historical tracking of tenants actions to identify usage trends over time.

These insights can provide THRON's executives with a deeper understanding of their product and its usage. Such knowledge can reveal peak usage times and popular

features, which is crucial for optimizing resource allocation and improving the overall service quality. Additionally, it may help while setting product development priorities, shaping marketing strategies, and planning business expansion.
From an economic perspective, analyzing usage data can uncover new revenue opportunities. For instance, it may lead to the development of targeted advertising strategies, the introduction of exclusive features, or the creation of subscription models that align with user preferences. All these actions are set to maximize revenues.
Lastly, monitoring usage helps ensure compliance with terms of service and regulatory requirements set by THRON.

In summary, establishing a system for collecting and analyzing usage data provides a foundation for data-driven decision-making. While the specific applications of this data are yet to be fully determined, starting with comprehensive data collection is the crucial first step towards uncovering hidden insights and guiding the future development.

1.2.4 Additional use cases

While the integration of BI into the THRON platform is a key objective, it offers numerous additional benefits across various departments within the company. The process of collecting usage data in the form of performed business operations can significantly enhance the functionality and efficiency of these departments.
Each use case relies on the capability to query stored information in an efficient and flexible manner. From a technical standpoint, once data is collected, prepared, and stored (through an ETL job or similar process), there are no restrictions on its potential applications. Although BI is the primary use case, the same data can be used to support other use cases without additional effort.
Table 1.3 shows some potential use cases where storing platform's usage information could be useful and powerful.

There is potential for additional informational value to be derived from this data, which may lead to the definition of new use cases based on the insights gathered. This flexibility and adaptability in data usage underscore the strategic importance of integrating BI and into the THRON platform.

Use case	**Need**	**Description**	**Output required**
Development	Gain a comprehensive understanding of THRON platform workflows.	Resolve bugs, conduct system analysis, reconstruct users actions to determine the resulting changes.	Ability to query data, eventually from a dedicated console.
Help desk	Reconstruct specific scenarios to assist users based on their reports and problems.	Identify affected entities, understand the broader impact of user actions.	Ability to execute high-level queries easily, possible through an API interface.
Customer success and product owners	Complement front-end behavioral tracking tools with information derived from business operations.	Obtain quantitative information, possibly aggregated by tenant execution.	Ability to execute high-level queries easily, possible through an API interface.
Enterprise clients	Require integration with customer-managed solutions like Splunk or Microsoft Sentinel to track changes.	Retrieve business events based on criteria (time range, user, etc.) and generate machine-readable outputs (like CSV files).	Ability to execute high-level queries through an API interface, possibly exporting the result in different formats.

Table 1.3: Additional use cases solvable by exploiting the BI process

Chapter 2

Addressed sub-problems

The analysis identified the specific portion of the original problem outlined above for focused attention. It listed the requirements and related them to the corresponding sub-problems addressed. Finally, the discussion covered the aspects that were neglected, with justifications provided for these decisions.

2.1 Primary areas of focus

2.1.1 Overview

The primary point of focus for enhancing business intelligence in THRON is to make potentially insightful data, collected from the usage of THRON platform, available for further analysis.
The effort is on establishing a robust infrastructure to support the initial phases of the BI process, namely data collection, preparation, and storage, but focusing also on providing easy data access. This foundational work is critical to ensure that THRON's data is easily accessible, well-structured, and ready for subsequent analysis. It is important to note that the focal point is just on providing access to the collected data. The goal does not encompass data analysis itself; rather, it focuses on preparing the data to be analyzed.

The first step in such a BI start-up strategy is to ensure that data from THRON's platform is accessible in an easy, straightforward manner. Currently, THRON lacks a system for logging and accessing business operations data. This makes it challenging to gain insights from platform business activities. Therefore, by systematically logging business operations and creating an infrastructure to collect, prepare, and store it, this gap is likely to be covered.

The second focus area is developing an easy, flexible method for accessing THRON's stored data. This involves enlarging the infrastructure's features such that it provides accessible interfaces such as APIs. These interfaces may enable data analysts and other stakeholders to retrieve and use the data without needing in-depth technical knowledge, therefore allowing the establishment of the further stages of the BI process.

2.1.2 Technical overview

From a technical perspective, THRON platform is designed to facilitate seamless integration with external tracking tools, thanks to the way in which it has been developed. The platform operates on a cloud-based architecture, supported by AWS services, with significant emphasis on relying on a serverless infrastructure wherever possible.

The platform relies on an event-driven architecture design pattern, where components respond to events or state changes in an asynchronous manner. Key characteristics of this approach include:

- **Event production and consumption**: producers generate events, which are then consumed by listeners designed to process these events;
- **Decoupling of components**: producers and consumers are decoupled, enhancing flexibility, scalability, and fault tolerance;
- **Event reaction**: the system is built to react to events using publish-subscribe or message-queue models;
- **Event routing**: events are transmitted through an event bus or stream that routes them from producers to consumers.

In THRON's case, events are generated as a result of user actions performed on the front-end components. Each business operation, performed by a tenant, triggers the creation of a corresponding business event. To be processed, these events must be delivered to the appropriate consumer within the platform, which handles them by applying the corresponding business logic.

An event bus serves as the intermediary between the event-producing components and the event-processing components. For THRON, this intermediary is Amazon EventBridge. THRON extensively uses it to manage the flow of business events to custom components that consume them.
Figure 2.1 summarizes such a concept by giving an example.

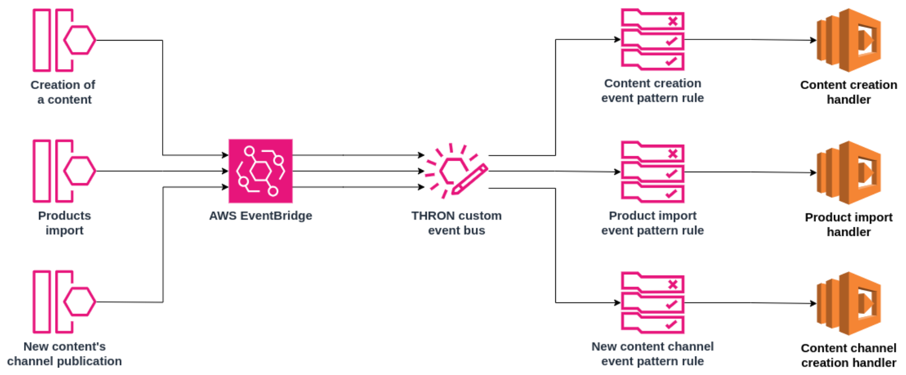
Creation of a content
Products import
New content's channel publication
AWS EventBridge
THRON custom event bus
Content creation event pattern rule
Product import event pattern rule
New content channel event pattern rule
Content creation handler
Product import handler
Content channel creation handler

Figure 2.1: Example of Amazon EventBridge usage in THRON platform

Given this setup, if the system's business operations are converted into events managed by Amazon EventBridge, then it essentially captures a comprehensive record of all activities within the platform. Thus, by attaching tracking mechanisms to this event bus, it is possible to monitor and analyze all business operations performed through the THRON platform. In this way the ephemeral events passing through that event bus can be collected and saved, making them available for a wider amount of time.

To effectively collect and manage this data, an ETL process is likely to be implemented. Such an infrastructure extracts relevant data from the event bus, transforms it into a structured and uniform format, and loads it into a centralized repository (data warehouse). This is the critical steps, which ensures that data is prepared and stored systematically for future use.

The final technical component involves setting up APIs that provide access to the collected data. Such APIs should be designed to be easy to use and flexible, allowing potential users of THRON's data to query and retrieve it in a limited manner with respect to their needs. This accessibility is vital for enabling THRON's data analysts to conduct further analysis and generate insights.

Audit Trail Platform is the name given to the system to be developed in order to address the previously described goals. Such a name originates from the combination of "Audit Trail" and "Platform".
The term "Audit Trail" typically defines a system that provides a detailed record of events or transactions within a system. It is used to track, review, and verify activities over time, and it ensures accountability, aids in troubleshooting, and user information for each tracked activity. Although the developed system does not entirely fit this definition, the name was chosen to indicate its purpose effectively.
The "Platform" component clearly refers to the THRON platform.
Thus, "Audit Trail Platform" denotes an audit trail system specific to the THRON platform.

2.1.3 Rationale

Focusing on these aspects is crucial because, as of now, THRON's data is not easily accessible, and there are no other methods to understand platform activities. Establishing a systematic approach to data collection, preparation, storage, and availability addresses this issue.
Furthermore, well-structured and easily accessible data is the core point to achieve effective BI processes and strategies. Creating a solid data infrastructure enables THRON's data analysts to perform in-depth analyses. This, in turn, allows THRON's strategic direction to be informed by accurate and data-driven insights.

The presented key points are directly aligned to THRON's needs for what concerns BI. In fact, focusing on data infrastructure directly addresses the needs of THRON's data analysts, as it provides them with the tools and access necessary to conduct meaningful analyses. This empowerment is essential for leveraging BI to gather hidden knowledge about the platform and, therefore, drive strategic decisions.
While the presented points of attention do not directly improve BI in THRON, as BI capabilities are not yet in place, they are critical in laying the groundwork. As a result of this, the stage is set for future BI activities.

2.2 Requirements

This section aims to provide a comprehensive overview of the requirements that such a system should satisfy in order to cope with the previously explained areas of interest. Requirements are grouped differently depending upon the area in which they focus on.

2.2.1 Data ingestion and storage

R1 - Persistence storage for business events

The platform's business events, currently ephemeral and available through Amazon EventBridge, require a cost-effective system for storage. It is important to determine how to save events, considering aspects such as raw data transformation, data aggregation, partitioning, and storing format. It is worth attention even the way in which the whole process takes place, as it should adopt buffering techniques and other methods that could make it efficient. To meet such goals it's worth identifying and comparing AWS ecosystem services that offer efficient and economical storage solutions.
This topic addresses the problem of inaccessible data by providing a persistent and structured storage solution. It is going to be essential for ensuring that historical elaborated data is retained for future analysis, which is crucial for long-term strategic planning. Such a statement is true for any data warehouse, as persistent storage is a fundamental requirement for any BI system.

R2 - Configurable data retention and deletion

The system must support configurable data retention and allow for the deletion of data for tenants who are no longer clients of THRON. The storage system must enable efficient and straightforward deletion of old customer data, therefore it is crucial to evaluate whether the chosen data warehouse, for storing data, supports data deletion, including associated costs and impacts on the rest of the data stored.
Such a need mitigates issues related to data management and compliance by ensuring data can be retained or deleted as required. It is also crucial for what concerns data hygiene, complying with regulations, and managing storage costs as unused data is costing money without any benefit, since it cannot be used anymore. That is a common requirement across many BI environments, as ensuring proper data management and compliance with legal standards is a pretty ordinary need.

R3 - Independence from business events structure

Since the THRON platform evolves continuously by adding new features, new business events may be defined to state the happening of new business operations. Therefore the system should remain agnostic to the form of individual business events. This means that it should not do any event-specific computation, as this creates a strong dependency between the codebase and the particular structure of an event, limiting the expansive power of such a system.
The system should ensure that new business events are treated as if they have always existed by using a uniform conversion format. In addition, the chosen ETL service must support flexible source data formats and unify them into a common structure for allowing easy querying.
This requirement addresses the need for a flexible data infrastructure that can adapt to

evolving data sources and business events without requiring significant reconfiguration. These premises are vital for maintaining a scalable and adaptable BI infrastructure that can handle future data changes seamlessly. That is another important aspect in BI, as any dynamic BI environment, where data sources and formats may change over time, needs flexibility and high adaptability.

R4 - Openness to integration with further data sources

The system should be extensible to integrate with other sources that may contain different kinds of events, such as informational ones (for example, an entity that has been seen). This means that the system should enable easy inclusion of new data sources in the ETL process. Such a service must be flexible enough to accommodate new data sources beyond the initial business events.
This facilitates the integration of additional data sources, enhancing the comprehensiveness of the BI system. It is really important for future-proofing the BI infrastructure and ensuring it can grow with the business. Even in this case, it is a universal requirement for BI systems to ensure scalability and flexibility in data integration.

R5 - Cost and performance considerations for large data volumes

The system must consider cost and performance when dealing with large data volumes over extended periods. It has to optimize data storage and query execution to manage costs while maintaining performance. For each considered service, storage models and query costs must be evaluated, also focusing on maintaining performance while managing expenses. This addresses the need for a cost-effective and high-performance data infrastructure.
Such a constraint is important for ensuring the sustainability and efficiency of the BI system over time. It is a relevant topic for any BI system dealing with significant data volumes, where ensuring manageable costs and acceptable performance are key features.

2.2.2 Data access

R6 - Effective and flexible search functionality

The system must support effective and flexible search capabilities, allowing queries based on partial information that should return business operations involving that kind of information. The system should ensure that all event data is retrieved and stored, allowing any kind of information to be searchable via APIs. Such APIs must be developed in a way that enables flexible and detailed searches of the event data.
This improves data accessibility and usability by enabling detailed and flexible search capabilities. Such a feature is critical for allowing users to find and use specific data points, enhancing the overall utility of the BI system.
That is a general requirement for BI systems, as ensuring that users can efficiently locate and use data, allow them to focus only on specific portions of the original dataset. Thus, they are able to exploit different kinds of knowledge, by analyzing the same dataset from different perspectives.

R7 - Search through a query console

The system should provide a console for users to perform queries using SQL, allowing them to retrieve that from the data warehouse without knowing the event-specific format of the original data. Such a feature could potentially leverage on an existing service which provides such a functionality.
This enhances user accessibility and empowerment, allowing technical users (such as developers and data analysts with technical skills) to perform data queries in the most flexible way possible. Such a tool is common in BI environments, as it ensures that users with varying technical skills can interact with the data seamlessly.

R8 - Search through an API Interface

The system should allow users to execute queries via a user-friendly API, abstracting complex data warehouse interactions. This enables easy integration with external components (such as front-end applications) through APIs, allowing non-technical users to use the system without the burden of knowing query languages or similar technical stuff.
The preferred way to achieve such a requirement would be by exposing APIs which embraces the REpresentational State Transfer (REST) concepts (RESTful-like APIs). Such a requirement improves accessibility and ease of use, enabling broader adoption of BI capabilities which is crucial for ensuring that all users, regardless of technical skill, can effectively use the BI system. Overall, it is another standard requirement for modern BI systems to enhance usability and accessibility.

R9 - Exportable search results

The system should support exporting search results from APIs in various formats, such as Comma Separated Values (CSV) and XLS (Microsoft Excel Spreadsheet). This enables the export of search results in multiple formats for both human and machine consumption. In the first case such a feature could be useful to perform more complex calculations, which are not possible via APIs, using external tools. In the latter case machines may consume such an input and perform custom-defined operations.
This increases data usability by allowing results to be exported and used in various contexts, making it easier to perform data sharing and further analysis, both internally and externally. It is a much-wanted feature in the BI context, as data export capabilities are necessary for broader data utilization.

R10 - Support for aggregated data queries

The system should expose APIs to support aggregated data queries against the data warehouse. Such a feature enables the retrieval of aggregated quantitative information, which is particularly useful when working with large datasets. The exposed functionality should provide results via file (like the export feature) or API response.
Such an improvement increases the efficiency of data analysis by providing pre-aggregated data, reducing the load on end users. This is an essential factor when handling large datasets efficiently and providing actionable insights. That is an usual requirement for BI systems dealing with large volumes of data, ensuring efficient data pre-processing and analysis.

R11 - Retrieval performance

The system should offer higher responsiveness for retrieving recent data compared to older historical data. In particular, the system should perform fast queries when operating with date ranges. This ensures quick access to recent data to meet user expectations. To achieve such a result, it should optimize the data storage and retrieval mechanisms to prioritize time-partitioned data access.
Such a need enhances user experience by providing faster access to the most relevant and recent data, without sacrificing too much older data access. This is a key aspect in BI systems to support real-time or near-real-time data access needs, such as making timely decision-making based on the latest available information.

R12 & R13 - Infrastructure constraints

In order to align with THRON's infrastructure and cloud development standards, the system should be developed using AWS services, preferring ones that provide serverless solutions.

2.2.3 Summing up

Table 2.1 summarizes the presented requirements. Each one is categorized depending on its nature and the functional area it belongs to.

The outlined macro-requirements primarily focus on how information should be stored and made available, considering factors like cost, efficiency, retrieval speed, and necessary transformations. These considerations directly relate to the initial stages of the BI process, specifically the steps of data collection, preparation, storage, and availability. By fulfilling these requirements, the system is capable of collecting business events, structuring them appropriately, and making them available for further consultation. This approach aligns with THRON's strategic goals in BI, laying the groundwork for future advancements in such a field.

Some requirements address the depth of searchability and the exposure of information. While still relevant for BI purposes, these elements mainly refer to secondary use cases, which are inherently satisfied by making platform information accessible. Ensuring these requirements guarantees the searchability of all available information at various levels of granularity. This accessibility enables teams to leverage the provided information, focusing on their specific areas of interest (as explained when talking about use cases). The capability to present this data in different formats (API output like JSON content, XLS, CSV) facilitates integration with external software, enhancing data analysis and supporting more in-depth research. This aligns well with the system's primary goal: not to provide comprehensive information search functionalities within the platform but to make the information readily available.

2.2.4 Deferred objectives

As stated above, while addressing the initial stages of the BI process, the proposed system focuses solely on the collection, preparation, storage, and availability of data. However, it deliberately excludes the analysis of this data.

ID	Description	Type	Functional area
R1	The system should collect, transform and store ephemeral events flowing through Amazon EventBridge in an efficient manner.	Functional	Data ingestion, data storage
R2	The system should allow flexible data retention and efficient data deletion.	Functional	Data storage
R3	The system ingestion logic should not depend upon business events' structure.	Functional	Data ingestion, data storage
R4	The system should allow easy integration of new data sources.	Functional	Data ingestion
R5	The system must rely on a cost-effective storage service, which guarantees also good performance when querying for data.	Constraint	Data storage
R6	The system should provide flexible search facilities, to allow easy retrieval of business events of interest.	Functional	Data access
R7	The system should provide a console to perform SQL queries to retrieve business events data.	Functional	Data access
R8	The system should provide API interfaces to allow easier access to business events data.	Functional	Data access
R9	The system should provide a way to export search results in common data formats.	Functional	Data access
R10	The system should expose APIs to perform data aggregations.	Functional	Data access
R11	The system should provide faster data access for recent data.	Performance	Data access
R12	The system should be developed using AWS cloud-based services.	Constraint	-
R13	The system should be developed in a serverless manner.	Constraint	-

Table 2.1: Summary of the system requirements

The system is designed to make data available in a structured and uniform manner, without delving into the extraction of insights or new knowledge from the collected data. Data analysis and data science require specialized skills and a deep understanding of the business context. Identifying which information is crucial and how to use it effectively demands a higher level of expertise in business analytics, which is beyond the scope of this project. Furthermore, THRON is in the early stages of developing its BI capabilities for what concerns its platform. The lack of previous BI initiatives means there is no clear direction or established framework for data usage. The focus, therefore, is primarily on building a solid foundation that future BI efforts can build upon.

This project aims to provide a starting point for BI at THRON. By making data available and structured, it sets the stage for future analysis and studies. This foundational work is crucial for any subsequent BI activities, offering a reliable dataset from which meaningful insights can eventually be derived. As THRON's BI capabilities mature, there are going to be opportunities to refine and expand the system. Future iterations can incorporate advanced data analysis, allowing the organization to gain deeper insights and make data-driven decisions. Ensuring that data is accessible in various formats (e.g., API output, XLS, CSV) facilitates integration with external analytical tools. This approach allows THRON's data analysts and business users to leverage external software for in-depth analysis, supporting a more flexible and adaptive BI strategy.

Chapter 3

Audit Trail Platform

The presentation covered the overall system architecture that satisfied the specified requirements. Each component was detailed in both functional and technical terms, with explanations provided on how each component related to the corresponding requirements. Additionally, the description included the testing methodologies used for the system, followed by a general evaluation of its performance.

3.1 System overview

The overall system is a new component of the THRON platform, which is in charge of:

1. Ingesting all business events generated by the platform;
2. Transforming such events' structure into a defined standard format, allowing the system to remain agnostic from it while still correctly managing and interpreting them;
3. Efficiently storing the transformed events in a data warehouse; Make such data accessible through RESTful APIs or direct queries to the data warehouse, offering different and flexible ways to collect it.

There are 3 main macro-areas that stand out from the infrastructure of Audit Trail Platform:

- **Events ingestion**: which is in charge of collecting, transforming, and storing business events;
- **Events search and aggregation**: which gives access to the stored business events in order to perform searches and aggregations;
- **Events export**: which provides capabilities to export search results and aggregations.

The ingestion component is responsible for the initial management of business events, focusing on executing an ETL process to efficiently load these events into the data warehouse. This process involves extracting raw data, transforming it into a suitable

format, and then loading it into the storage system.
Once the data is securely stored in the data warehouse, the system provides robust mechanisms for users to access this information through search functionalities and aggregation operations. These features enable users to efficiently query and retrieve the stored data for subsequent analysis phases.
Additionally, the system supports the exportation of data from the data warehouse in widely-used formats, facilitating integration with external systems or further data analysis outside the platform.

Figure 3.1 gives a summarized overview of the system and the way in which it operates.

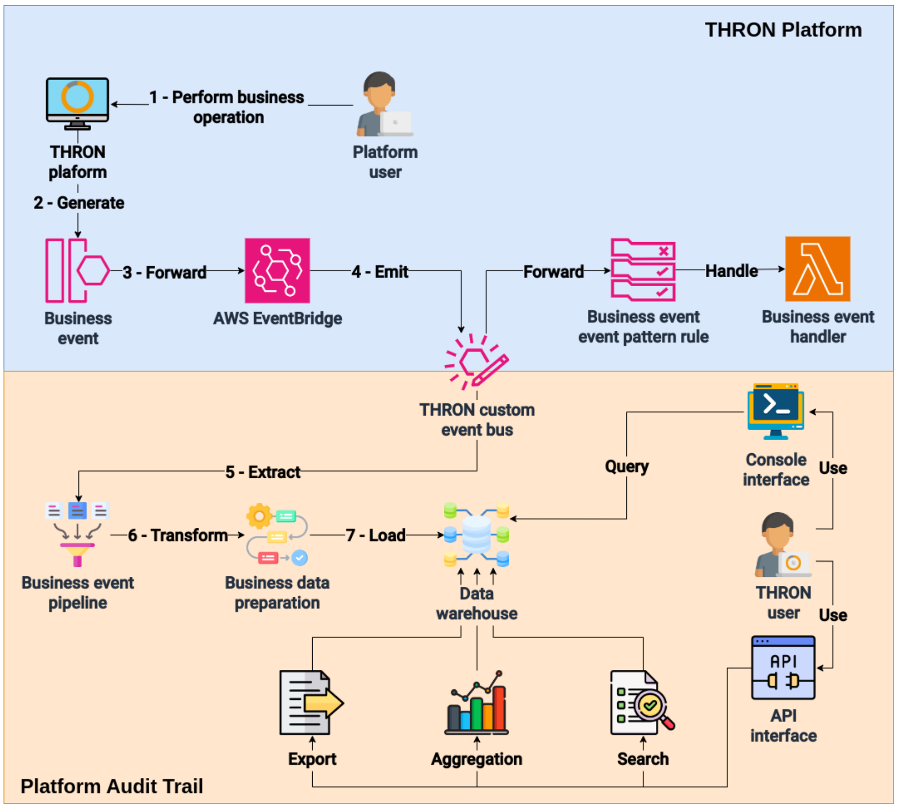

Figure 3.1: Overview of the system and its relation with the THRON platform

3.1.1 Ingestion

3.1.1.1 Functional overview

The primary functions of this component are as follows:

- **Event collection**: each business event coming through the event bus is collected, with buffering techniques applied prior to transformation;
- **Data transformation**: collected events are transformed from their original structure into a standard format defined by a table schema. This transformation

ensures the retrieval and tagging of key information while still preserving event-specific details. It is essential for enabling SQL-based queries on the stored data;

- **Data storage**: The transformed data is stored in a structured persistence system (data warehouse), with emphasis on efficiency in both storage format (space optimization) and processing time (buffering techniques to prevent system overload).

Figure 3.2 provides a detailed view of the component using a C4model representation. The numbers on the arrows represent the flow in which operations are performed.

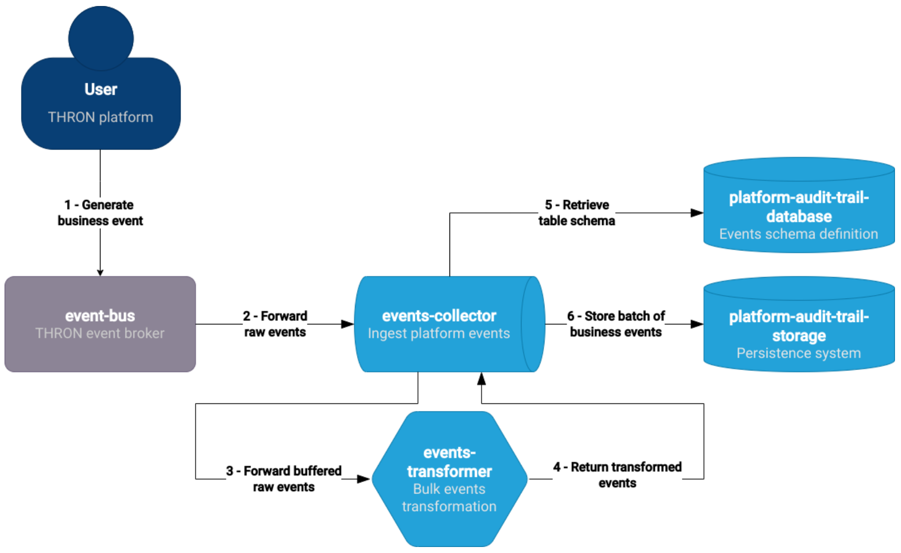
User
THRON platform
1 - Generate business event
event-bus
THRON event broker
2 - Forward raw events
events-collector
Ingest platform events
5 - Retrieve table schema
platform-audit-trail-database
Events schema definition
6 - Store batch of business events
platform-audit-trail-storage
Persistence system
3 - Forward buffered raw events
events-transformer
Bulk events transformation
4 - Return transformed events

Figure 3.2: C4model representation of the events ingestion component

This component addresses the primary sub-problems outlined in section 2.1 by implementing a standard ETL process, which is critical for data collection, preparation, and storage in the BI workflow.

Section A.1 contains a more in-depth understanding of the satisfied requirements for such a component.

3.1.1.2 Technical overview

Figure 3.3 shows the proposed solution, based on AWS services.

Extraction

To integrate the ingestion and historical storage of business events, a *catch-all* rule has been implemented on the THRON event bus. This rule ensures that every event transmitted through the THRON event bus is also relayed to Audit Trail Platform.

Amazon Data Firehose serves as the bridge between the event bus and the persistence system, meaning that the previously mentioned *catch-all* rule ensures that all events passing through the event bus are directed into the Firehose pipeline. The component

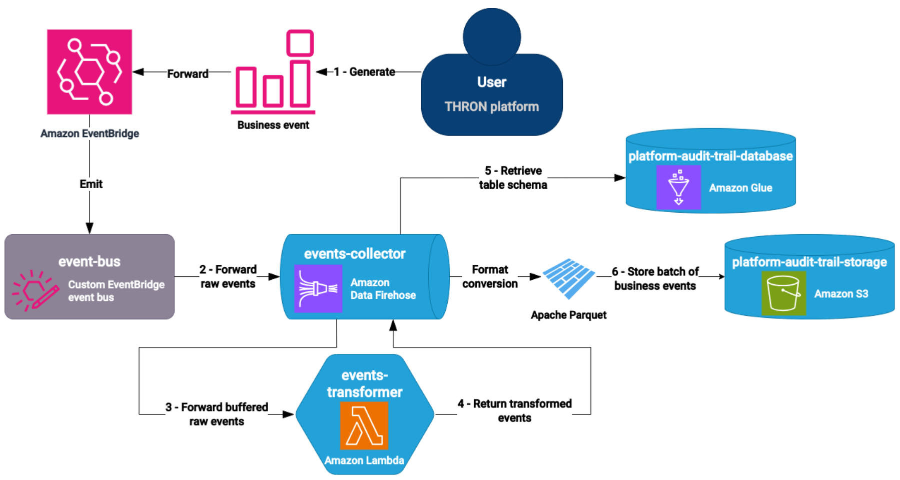

Figure 3.3: Technical overview of proposed solution for events ingestion

implements buffering policies to optimize data transformations and subsequent writes to the persistence system. This enables the system to be reactive while limiting heavy and frequent operations on the persistence layer.

Once the buffering limit is reached, the batch of business events undergoes sequential transformation. The events are standardized and re-enter the Firehose stream, where they are prepared for storage in the data warehouse.
Backup saves are performed to allow the storage of event batches in their raw format, with the aim of enabling future reprocessing, if needed.

Transformation

The transformation step is critical for extracting essential information from each event and mapping it to common, well-defined fields, ensuring that events are searchable and enabling in-depth querying across the dataset.

For example, consider the business event represented in code block 3.1, as captured by Amazon EventBridge.
All fields except the *detail* field are always present by default and contain information provided by EventBridge. The *detail* field contains custom content, specifically:

- ***metadata***: identifying information of the event;
- ***invoker***: data about the user session during which the event was triggered;
- ***data***: event-specific data essential for the component consuming the event, varying depending on the event's nature.

Handling the payload (i.e. the *data* field) of each event type in a specific manner is not feasible, as it would violate requirement R3. However, mapping the payload information is essential for guaranteeing deep searchability, needed to satisfy other use cases (beyond BI).
Therefore, the transformation process follows this logic:

```json
{
  "version": "0",
  "id": "d5907a6a-15ac-4909-e168-92e75357a1cb",
  "detail-type": "PRODUCT_CREATED",
  "source": "product-data",
  "account": "131539925590",
  "time": "2024-05-09T09:07:29Z",
  "region": "eu-west-1",
  "resources": [],
  "detail": {
    "data": {
      "client": "tenant_id",
      "createdAt": "2024-05-09T09:07:04.087Z",
      "id": "4892b55c-539a-4cc3-8434-d3c6848515b5",
      "newValue": {
        "lastModifiedAt": "2024-05-09T09:07:04.087Z"
      },
      "oldValue": null
    },
    "invoker": {
      "actor": "product-sync",
      "client": "tenant_id",
      "ip": "169.254.76.1",
      "subject": "user_of_tenant",
      "userAgent": "product-sync"
    },
    "metadata": {
      "correlationId": "1-663c91fc-16def6e97974ff4834819a93
         ",
      "createdAt": "2024-05-09T09:07:04.088020693Z",
      "emittedAt": "2024-05-09T09:07:28.96615028Z",
      "emittedBy": "product-data",
      "id": "98474349-afa7-4510-b081-e72f1376d2cb",
      "type": "PRODUCT_CREATED",
      "version": 1
    }
  }
}
```

Code block 3.1: JSON representation of an event via Amazon EventBridge.

```
{
  "id": "d5907a6a-15ac-4909-e168-92e75357a1cb",
  "correlationId": "1-663c91fc-16def6e97974ff4834819a93",
  "context": "pim",
  "type": "PRODUCT_CREATED",
  "emittedAt": "2024-05-09T09:07:28.96615028Z",
  "data": "{\"client\":\"tenant_id\",\"createdAt\":\"2024-0
     5-09T09:07:04.087Z\",
                \"id\":\"4892b55c-539a-4cc3-8434-d3c6848515
                   b5\",
                \"newValue\":{\"lastModifiedAt\":\"2024-05-
                   09T09:07:04.087Z\"},\"oldValue\":null}"
                   ,
  "dataValuesMap": {
        "client": ["tenant_id"],
        "createdAt": ["2024-05-09T09:07:04.087Z"],
        "id": ["4892b55c-539a-4cc3-8434-d3c6848515b5"],
        "newValue.lastModifiedAt": ["2024-05-09T09:07:04.08
           7Z"],

  },
}
```

Code block 3.2: JSON representation of a business event after the transformation phase.

- Fields *invoker* and *metadata* are always retrieved, relying on the standard structure of the JSON file applicable to all events emitted by THRON;
- Fields within the *data* section are mapped in a raw format as key-value pairs, preserving the original JavaScript Object Notation (JSON) nesting levels.

Code block 3.2 represents the transformation of the business event presented above. Such a standard format ensures that key information is consistently stored while also accommodating event-specific details. The process achieves this by storing common, universally present information necessary for searchability, while handling and preserving variable information in a semi-structured format without requiring custom operations for each event type.

Load

Amazon S3 is used as a persistence system. Although being a general-purpose storage service, it functions effectively as a data warehouse when storing structured data, such as business events saved in batches via Amazon Data Firehose into a designated S3 bucket.

Data coming from such a pipeline is converted in Apache Parquet columnar format to achieve efficiency in storing and querying such information.
Appendix B explains why such a decision has been made.
The generated files are partitioned by the date of ingestion, a critical implementation choice that enhances the overall querying efficiency. Partitioning minimizes the amount

of data read during queries, and combined with the storing format, optimizes data retrieval performance.

S3 lifecycle rules allow for the automatic deletion of obsolete business events (e.g., the ones older than 3 years), keeping the data warehouse clean , containing only up-to-date data according to THRON's needs. However, S3 does not support data retention when a client discontinues using the THRON platform. Since event batches are stored in Parquet files not partitioned by tenants, modifying a Parquet file to remove specific records is challenging. As a result, such customer data is not deleted, but excluded from search results using SQL WHERE clauses.

Overall evaluation

This architecture, employed for executing ETL jobs, is not novel as it aligns with standard use cases for Amazon Data Firehose. The features offered by this service are well-suited to meet the specified requirements, as it provides extensive customization options for each phase of the ETL process without requiring infrastructure management. Furthermore, the integration of such a service with complementary services effectively addresses all aspects of the ETL process, offering a robust solution for the ingestion of business events in a flexible, autonomous, and efficient manner.

Section A.1 lists the used AWS services and the way in which their features satisfy the requirements.
Section C.1 shows different approaches to the implementation of this component.

3.1.2 Search and aggregation

3.1.2.1 Functional overview

The primary function of this component is to provide efficient access to business event data stored in the data warehouse. It enables users to search for and retrieve data based on various criteria, including partial information.

The component supports two main types of searches:

- **Generic search**: retrieves business events that meet specific criteria;
- **Aggregation**: retrieves numeric and quantitative insights from historical event records based on criteria such as time ranges.

Given the potential for large data volumes, the component delivers functionalities asynchronously to prevent busy-wait scenarios.

These functionalities are provided through two modes.
RESTful APIs, by leveraging the concept of "job", which is defined as the asynchronous execution of a query against the data warehouse to perform searches or aggregations. In this context, such APIs are exposed over the HTTP protocol, using HTTP verbs and the job as a resource to define actions (create a job, retrieve its status and the results) that can be performed over such a resource. Following the RESTful principles, the actions should be performed in a stateless manner. This promotes asynchronous communication for both the systems and clients.
SQL console, allowing the execution of custom SQL queries, providing flexibility in

searching and exploring the data warehouse. Operating in asynchronous fashion is not required in this mode, as it is intended for manual use rather than system integration.

Figure 3.4 provides an overview of the functional components needed to implement such a feature. The numbers on the arrows represent the sequence in which operations are performed.

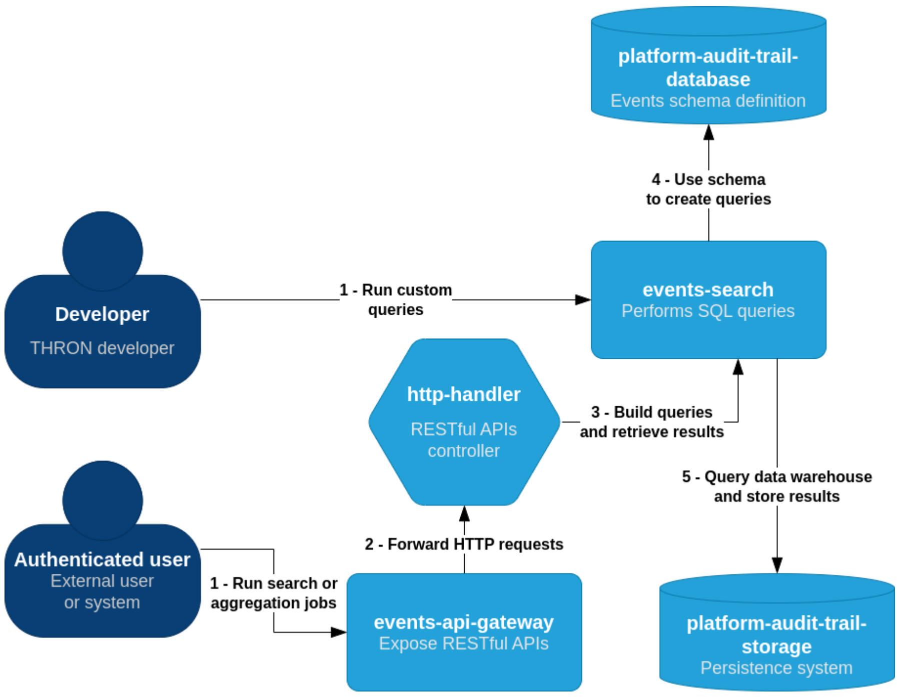

Figure 3.4: C4model representation of the events search and aggregation component

This component addresses issues identified in section 2.1. While the ingestion component manages event storage, this component facilitates access through APIs and console interfaces. It operates between the data storage and data analysis stages of the BI process, focusing on data access. It simplifies interaction with the data warehouse by providing flexible search capabilities and options for obtaining aggregated information.

By enabling targeted retrieval of specific events, this component ensures that the data analysis phase can take place using clean, relevant data, also reducing information overload and optimizing workstation performance for data analysts. It also handles data cleaning and aggregation tasks, contributing to the efficiency of subsequent BI process stages.

Section A.2 contains the list of the satisfied requirements by the features of this component.

3.1.2.2 Technical overview

Figure 3.5 illustrates the architecture for events search and aggregation using AWS services.

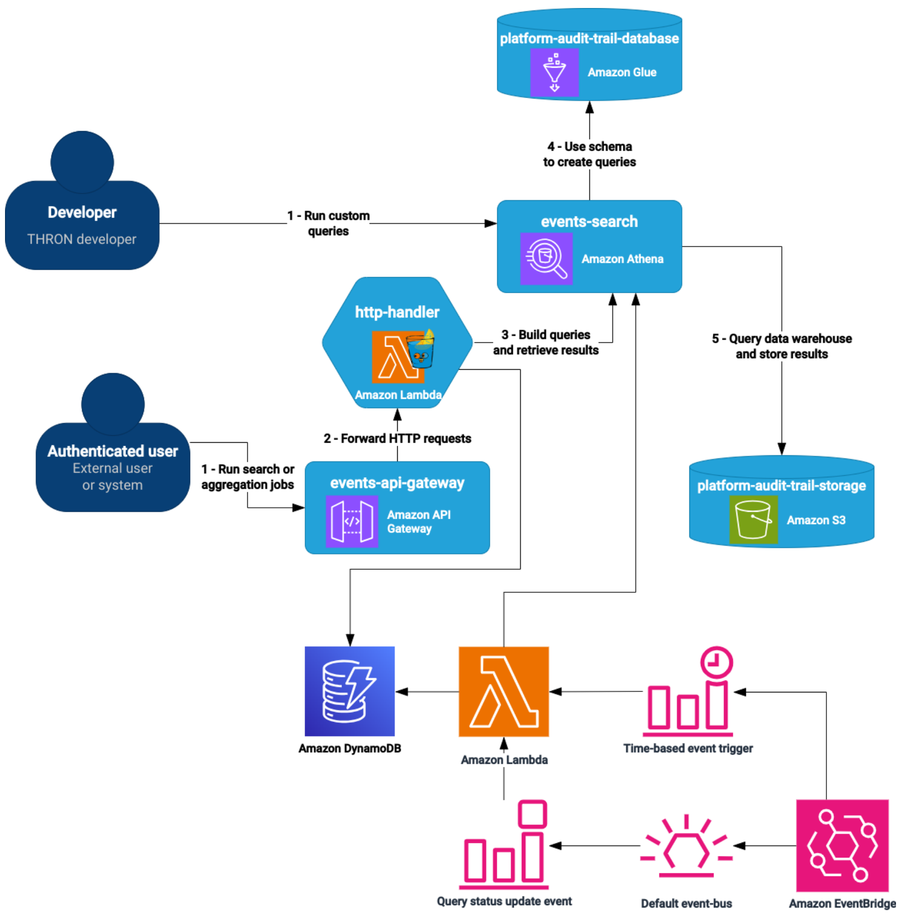

Figure 3.5: Technical overview of proposed solution for events search and aggregation

Query execution

The execution of queries is asynchronous, as running them on large datasets can take considerable time to complete. This makes a synchronous approach more challenging, especially when exposing APIs on the web. By choosing an asynchronous approach, the exposed APIs can be more manageable, scalable, and efficient.

The system executes jobs, which are tasks involving data search or aggregation. A job abstracts the concept of a query, simplifying its execution management and subsequent result exposure. A job consists of a state (running, completed, failed) and execution parameters (configuration, variable based on the task type). The following operations are defined for a job:

- **Creation**: to launch a new job (i.e. run a query);
- **Retrieve execution state**: to check when the job has finished executing (i.e. check if query has finished execution);
- **Retrieve results**: to get the results once the job is complete (i.e. retrieve query results).

The system exposes these functionalities through a RESTful-like API over the HTTP protocol. Below is an example of the RESTful endpoints provided for managing a search job.

Create a search job

- **URL**: `/actions/searches`
- **HTTP Method**: `POST`
- **Description**: initiates a new search job.
- **Header**: authorization token
- **Body**: list of filters (type, key-value pairs, date time range, flow, tenant, user, event context, etc.).
- **Response**: HTTP status 202 (Accepted), indicating that the job request is in progress.

Retrieve search job status

- **URL**: `/actions/searches/jobId`
- **HTTP Method**: `GET`
- **Description**: retrieves the execution status of a search job.
- **Header**: authorization token
- **Response**: HTTP status 200, with the job's current execution status.

Retrieve search job results

- **URL**: `/actions/searches/jobId/results`
- **HTTP Method**: `GET`
- **Description**: retrieves results of a completed search job with pagination.
- **Query parameters**: pagination token, page size.
- **Header**: authorization token
- **Response**: HTTP status 200, with a JSON array of business events matching the filters and a pagination token for subsequent pages.

Under the hood, every time a request is made towards a defined endpoint, the *http-handler* Amazon Lambda function is invoked. It interacts with Amazon Athena, the SQL query engine, to perform the requested operations towards the data warehouse.

All exposed APIs are stateless, requiring the user to manage information related to executed jobs. This allows the server to focus on abstracting the underlying complex system and performing all operations on the data warehouse based solely on the information contained in each individual request. Additionally, the job represents the resource on which operations can be performed and related information retrieved. The defined endpoints (URIs), in conjunction with the HTTP verbs used, represent the

means by which one can interact with the defined resource.
That being said, it is reasonable to assume that the exposed APIs widely adopt the majority of the concepts on RESTful APIs, but they do not 100% conform with the REST paradigm. An example of non-compliance is the lack of implementation of the Hypermedia As The Engine Of Application State (HATEOAS) constraint.

The set of exposed RESTful APIs abstracts the complexities of the data's structure, their format and the way in which they are stored in the data warehouse, enabling users to focus on filtering criteria rather than on how to query the data. This meets requirements R6 and R8.
Aggregations are handled similarly to search jobs, with variations in the filters: time range, aggregation period (daily, monthly, yearly), and event types. This directly addresses requirement R10.

3.1.2.3 Business events metadata

Certain filters lack clear indications of valid values to be used. For example, the values to be used on the search filter for retrieving specific types of business events is unclear, because there is no tracking of all event types emitted by the platform. Similarly, the key-values filter for event payloads lacks clear key definitions, as they depend on the business event's nature.

To maintain independence from business event types and their structures, a static compilation of all possible event types or payload keys is unfeasible. Both the types of business events and their payload structures can vary, making static listing impractical.

To address this, an Amazon Lambda function autonomously collects event information and stores it in a dedicated database table. Specifically, it collects:

- Event types;
- Event payload keys, grouped by event type;
- Contexts, to determine whether the event originated from a DAM or PIM component of THRON.

The Lambda function is triggered twice as follows:

1. A daily time-based trigger from Amazon EventBridge invokes the Lambda function, which queries the data warehouse, via Amazon Athena, for events registered in the past day;
2. Upon query completion, an event triggers the Lambda function again. This time it retrieves the query results, extracts the relevant information, and stores it in a Amazon DynamoDB table.

This dual invocation approach enhances efficiency by avoiding a busy-wait state, which would otherwise waste execution time and incur additional costs.

To facilitate access to this information, RESTful endpoints have been created to retrieve filter values for event types, payload keys, and contexts. This enables dynamic updates and retrieval of filter values, ensuring that users can access current information without needing to know or understand specific event structures. This approach simplifies and accelerates API usage, meeting requirement R6. Additionally, new events are taken over within a day, as the system autonomously collects and updates information about them. This mechanism not only maintains event structure independence but also ensures transparent and autonomous management of new events, therefore fulfilling requirement R3.

Overall evaluation

The proposed solution does not introduce novel technology, as the AWS services used are commonly employed for constructing a RESTful API infrastructure. However, the adoption of an asynchronous query execution model, as opposed to a synchronous one, is less conventional. This choice, while leading to more complex infrastructure management, enhances system efficiency in terms of execution time and cost for data retrieval and aggregation. Moreover, it promotes scalability and flexibility.

Cost considerations are generally manageable due to the preference for asynchronous queries to the data warehouse. The primary cost concern arises from the use of Amazon Athena, which incurs charges based on the volume of data scanned from Amazon S3 during queries. Consequently, costs rise with the number of queries and the amount of data retrieved. Measures such as data partitioning have been used to mitigate scanning costs, even though they cannot completely eliminate the issue. Users of Audit Trail Platform are responsible for these costs.

Section A.2 presents the link between AWS services and the way in which their features satisfied the requirements.
Section C.2 contains a short overview of alternative solutions for this feature.

3.1.3 Export

3.1.3.1 Functional overview

The component is designed to allow the export of the results from search or aggregation jobs, which are typically returned in JSON format. JSON is not ideal for further manipulation with external tools, and pagination complicates data conversion too. To address this, the component enables exporting results in XLS or CSV formats, which are more meant for post-processing and integration with external tools.

Given that file generation incurs additional wait time beyond query execution, the export feature is implemented asynchronously as well. A database table tracks export job statuses, with each entry containing utility information and progress status. This setup allows for status updates by various components and facilitates error tracking during the process.

Figure 3.6 provides an overview of the functional components for the export feature. The numbers on the arrows represent the operational flow of the feature.

This component enhances accessibility to event information by providing export capabilities to obtain XLS or CSV files, facilitating integration with external systems. It bridges data storage and analysis stages of the BI process, simplifying integration with

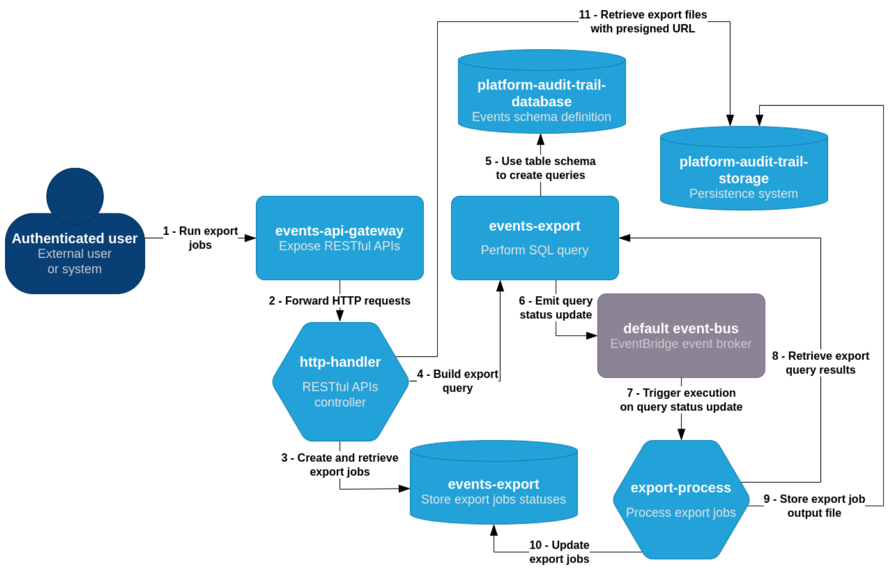
11 - Retrieve export files with presigned URL
platform-audit-trail-database
Events schema definition
platform-audit-trail-storage
Persistence system
5 - Use table schema to create queries
Authenticated user
External user or system
1 - Run export jobs
events-api-gateway
Expose RESTful APIs
events-export
Perform SQL query
2 - Forward HTTP requests
6 - Emit query status update
default event-bus
EventBridge event broker
8 - Retrieve export query results
http-handler
RESTful APIs controller
4 - Build export query
7 - Trigger execution on query status update
3 - Create and retrieve export jobs
events-export
Store export jobs statuses
export-process
Process export jobs
9 - Store export job output file
10 - Update export jobs

Figure 3.6: C4model representation of the events export component

external tools and allowing users to customize result processing, whose needs may go beyond API capabilities.

Section A.3 contains the list of the satisfied requirements by this feature.

3.1.3.2 Technical overview

The component operates similarly to the search and aggregation component previously described. Initially, it constructs and executes a query using Amazon Athena. Upon query completion, it collects the results and generates an XLS or CSV output file. The process involves potential delays from query execution and file generation, necessitating asynchronous management to avoid busy-wait scenarios for both the system and the user.

Figure 3.7 illustrates the export feature architecture, based on AWS services.

The functionality exposes the concept of export job, which is the combination of a search or aggregation job with the post-processing of the results to allow the export in different formats. Below is an example of the RESTful endpoints provided for managing the export of a search.

Create an export job

- **URL**: `/actions/searches/exports`
- **HTTP Method**: `POST`
- **Description**: initiates a new export job for search results.
- **Header**: authorization token

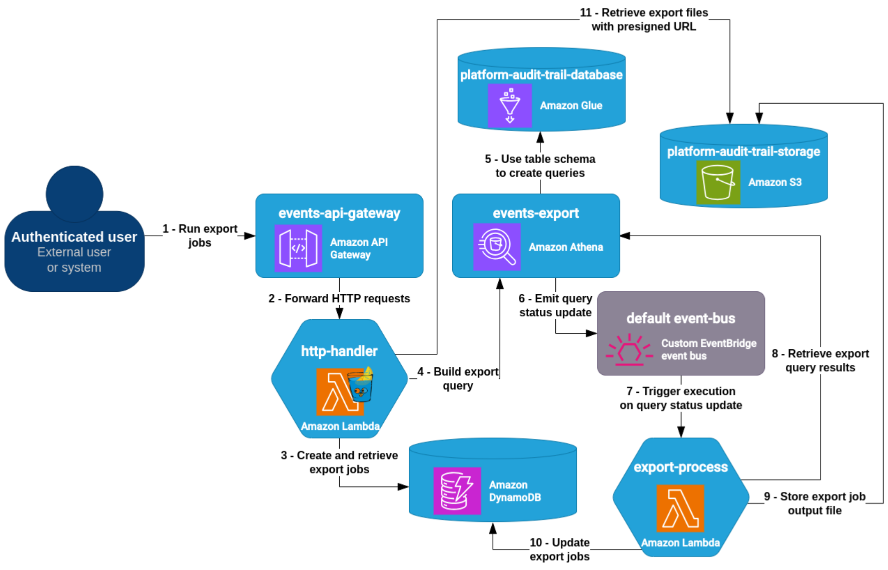
11 - Retrieve export files with presigned URL
platform-audit-trail-database
Amazon Glue
platform-audit-trail-storage
Amazon S3
5 - Use table schema to create queries
Authenticated user
External user or system
1 - Run export jobs
events-api-gateway
Amazon API Gateway
events-export
Amazon Athena
2 - Forward HTTP requests
6 - Emit query status update
default event-bus
Custom EventBridge event bus
8 - Retrieve export query results
http-handler
Amazon Lambda
4 - Build export query
7 - Trigger execution on query status update
3 - Create and retrieve export jobs
Amazon DynamoDB
export-process
Amazon Lambda
9 - Store export job output file
10 - Update export jobs

Figure 3.7: Technical overview of proposed solution for events export

- **Body**: list of search filters (identical to those used in the search job), output settings (file format, column configuration).
- **Response**: HTTP status code 202 if the request is accepted and processing is underway.

Retrieve export job's status

- **URL**: `/actions/searches/exports/jobId`
- **HTTP Method**: `GET`
- **Description**: retrieves the status of an existing export job.
- **Header**: authorization token
- **Response**: HTTP status code 200 with a body indicating the current status of the export job. If completed, a URL to access the output file is provided.

These APIs mirror the behaviors of the exposed APIs to perform searches and aggregations. The first one allows the creation of an export task, while the second one is used to track the status of such a task.

An ad-hoc Amazon DynamoDB table tracks the execution status of export jobs. Unlike Amazon Athena queries, where there is service which provides APIs for status retrieval, export operations, which are system-specific operations, do not have built-in status tracking components. The export process is asynchronous and it involves several components to achieve its goal, therefore it requires a custom solution for status management.

To establish an asynchronous process, the system relies on actions and callbacks. Once the query has been issued to Athena, upon an export request, it executes autonomously,

without requiring any component to interact with it.
Upon query completion, an event is issued and it triggers the invocation of an Amazon Lambda function which is in charge of gathering the query results and building the output file in the desired format.
The output file is stored in the persistence layer (Amazon S3). At this stage the export job is done and an URL is provided to download the generated file.

Overall evaluation

Requirement R10 is met through the integrated operation of all components involved in this functionality.

Although the approach is not novel, the architecture is notably effective in handling a complex process asynchronously. The current design also facilitates easy extension to support additional export formats. Extending functionality is simple, as it involves defining the logic for new formats, leveraging the existing data structure to generate the output file.

Section A.3 shows how the used AWS services contribute to satisfy the requirements for this feature.
Section C.3 contains a small reasoning about different approaches to implement this functionality.

3.2 Development

The implementation of Audit Trail Platform involved a total of 7286 lines of executable code, spread across different contexts. 1573 lines have been devoted to build the cloud infrastructure by defining the AWS cloud resources and their configurations, while the rest defines the business logic to expose APIs and interact with the AWS services.

The entire codebase has been entirely written in Go, a statically typed, compiled language known for its simplicity, efficiency, and strong support for concurrency. These features make it suitable for cloud computing.

11 external libraries were to develop the system. The most important ones are:

- **AWS Cloud Development Kit (CDK)**:to build the cloud infrastructure;
- **AWS Software Development Kit (SDK)**: to allow the interaction with AWS services;
- **Gin**: web framework to expose RESTful APIs;
- **Excelize**: to generate XLS and CSV export files.

3.3 Testing

3.3.1 Comprehensive code evaluation

Code review

The entire system has been developed following the Scrum agile methodology, with 2-week-long sprints to release new features incrementally. Before releasing any feature, the involved code underwent a rigorous review process. Features were developed in separate Git branches and reviewed through a pull request mechanism. Each pull request was evaluated by team members for consistency and alignment with project objectives, critical for maintaining code quality and long-term maintainability.

Prior to initiating a pull request, code was reviewed in *one-to-one* sessions with the software architect. These sessions focused on validating architectural decisions, code patterns, and adherence to THRON's internal policies, including AWS best practices and Go programming language standards.

Bi-weekly back-end competence meetings held at THRON addressed significant development topics in the field. For Audit Trail Platform, these meetings concentrated, on a couple of occasions, on critical components and system-wide issues. Discussions often focused on optimizing the structure of business events and determining the inclusion of information to enhance future utility.

Static code analysis

Static code analysis involves examining source code without execution to detect potential errors, security vulnerabilities, and compliance with coding standards. Automated tools are used to scan the codebase for syntax errors, code smells, bugs, and best practice violations, thereby improving code quality, maintainability, and security.

In this project, golangci-lint was employed as the static analysis tool, configured to THRON's coding standards. It particularly focused on keeping low overall cyclomatic complexity, good maintainability index and adhering to Go standard rules. The linter was integrated into the CI/CD pipeline, enabling automated code quality checks for every commit and pull request. This ensured that non-compliant code was flagged for review prior to merging.

Unit testing

Unit testing is a technique for verifying that individual units or components of a software application function correctly in isolation. A unit is typically the smallest testable part of an application, such as a function, method, or class. The primary goal is to ensure that each unit performs as expected under various conditions.

The following technologies were used to define and develop unit test:

- **Local mocks**: custom local mocks were created to isolate components and inject dependencies, enabling focused testing without relying on external libraries;
- **LocalStack**: used to simulate an AWS cloud environment locally, ensuring that the behavior of AWS services matched their actual behavior for accurate testing.

The verification process through unit tests achieved a code coverage of 93%, with a total of 7258 lines of code defining unit tests logic. The total number of executed unit tests is 412.

Particular emphasis was placed on testing the RESTful APIs. All scenarios, including both successful and erroneous cases, were tested to ensure alignment with the OpenAPI specification. This approach ensured high reliability and correctness of the RESTful APIs, fulfilling requirements R6 and R8.

Integration testing

Integration testing involves combining individual units or components of a system and testing them as a group to identify issues in their interactions and ensure they function correctly together according to the overall system design.

Postman was used as the integration testing tool to validate the functionality and correctness of the exposed RESTful APIs. In particular, 67 integration tests were performed with primary focus on testing edge cases and alignment with the OpenAPI specification.

3.3.2 Production rollout

The development of the system spanned 5 months, with the process being more efficient than initially anticipated. By the end of the fourth month, a semi-definitive solution was achieved. The term "semi-definitive" indicates that the ingestion phase, being the most critical and complex component, was stable and deeply tested, while the APIs exposure phase required additional refinements.

Given the independence of the API exposure phase from the ingestion phase, it was thought to be productive to start collecting business events from the production environment. The system was deployed in production with the following objectives:

- Validate the stability of the system by ensuring the correct functioning of the ETL process;
- Provide an initial batch of collected business events to data analysts to assess if Audit Trail Platform meets its intended practical objectives.

The production deployment was executed in two phases:

- **Ingestion phase deployment**: completed at the end of the fourth month of development;
- **APIs and additional functionalities deployment**: executed approximately two weeks after the ingestion phase release.

By the end of the development period, the system had been operational for about one month. Subsequent sections review some aspects of this operational period.

Information	Numeric value
Total number of tracked business events	896695
Distinct type of business events collected	25

Table 3.1: Numeric values about collected business events from production

First-month feedback

During the production verification and validation period, no critical issues were detected. As shown in table 3.1, the system ingested nearly one million business records of various types, confirming that the designed flow met expectations and handled many types of business events effectively.

AWS services were also equipped with monitoring probes to generate alerts for anomalies or critical errors. No alerts were triggered during the observed period for the ingestion phase.

Stress testing

The ingestion component showed robustness and efficiency, successfully managing real-time delivery and burst events without errors or failures. As previously detailed, its performance met expectations.

To verify API functionality, extensive search and aggregation jobs were executed. The system handled these tasks without issues, even when processing large datasets, such as full-month searches across all recorded events.

However, performance limitations were identified during testing of the export feature. While the component handled exports with smaller time windows effectively, it struggled with exporting the entire dataset of collected business events. The failure was attributable to insufficient RAM in the *export-process* sub-component, which was unable to generate the file in-place before transferring it to Amazon S3. This issue was temporarily resolved by increasing the memory allocation for the Amazon Lambda function.
A long-term solution is needed to prevent similar issues during full-load exports that span multiple years of data. Section 4.3.1 introduces a possible solution for such a problem.

Chapter 4

Retrospective

The analysis related the proposed solution's components to the addressed objectives and listed any unsatisfied requirements and limitations. Examples and methods demonstrated how use cases can be resolved and how the system could potentially enhance BI. Lastly, the overview provided potential future enhancements for the system, along with possible new functionalities.

4.1 Resolution of individual sub-problems

Audit Trail Platform addresses the aims set forth in chapter 2 quite closely. The system fully addresses the identified requirements from the initial objectives analysis. Each major component has been designed to handle a specific subset of the issues so that the overall system can work effectively and efficiently while meeting its intended goals.

4.1.1 BI process establishment

The primary focus was to establish a process capable of tracking business-level activities within the THRON platform. It involved creating an operational flow to record and store business events generated by the platform.
This challenge was effectively addressed by the ingestion component. This component successfully retrieves business events, processes them through a pipeline that applies a series of transformations and optimizations, and stores them in a standardized format. Such a process enables efficient storage and querying of historical business events.
The implementation of an ETL process proved to be a good choice to reach this objective, as it aligns seamlessly with the first three phases of the BI process, ensuring flexibility and scalability. Consequently, the ingestion component fully meets the expectations set by the first three stages of the BI process, laying the groundwork for subsequent analytical phases.

Table 4.1 shows how the features of the system help fulfilling the requirements regarding the establishment of the BI process.

System feature	**Requirement fulfillment**
Collect, transform, store business events originated by the THRON platform.	R1
Apply a standard transformation to business events to ensure flexibility and independence without any ad-hoc manipulations.	R3
Extensible infrastructure which allows for future integration of additional data sources.	R4
Use of a data warehouse and storage format that ensure space efficiency without compromising query performance.	R5
Flexible data retention with automatic deletion of old data and semi-automatic data deletion.	R2
Use of fully managed AWS services.	R12, R13

Table 4.1: Requirements tracking for BI process establishment

Regarding data retention (R2), the system cannot delete a customer's data before the retention period ends. This limitation partially affects compliance with the requirement. To mitigate this, an additional patch excludes customers who have left the platform from search, aggregation, and export results. Although not the most elegant solution, it is the most efficient way to avoid additional costs associated with manual data management.
In general, except for R12 and R13, the remaining requirements are functional. None of these imposed restrictive constraints on the system's development concerning data access. These requirements outline a standard scenario for implementing a BI process in the relevant phases, meaning that no constraints forced the solution's design, as they were aligned to standard BI process rules.
The only significant constraint, essential for the solution's implementation, is requirement R12, which technically ties the solution to the AWS ecosystem.

4.1.2 Information access

After addressing the storage of business events, the next challenge was to ensure accessible, flexible, and efficient retrieval of this information to accommodate various use cases, including non-technical users.
To meet this requirement, the system provides multiple access methods to maximize flexibility:

- **API access**: facilitates integration with external systems, enabling retrieval of relevant information. Two modes are available:
  - **Generic search**: allows retrieval of business operation dumps using basic search filters;
  - **Aggregation**: provides numerical insights over a historical period of interest.

- **Console access**: offers expert users maximum flexibility in exploring the dataset;
- **Export**: enables users to export searches and aggregations in common formats, making data accessible to external systems and non-expert users.

This range of options ensures comprehensive inclusion, allowing all user types to access and use the information, therefore maximizing the data's potential.

Table 4.2 illustrates how the features of the system help fulfilling the requirements regarding data access.

System feature	Requirement fulfillment
Abstract complexities such as data structure, storage method, and query execution to allow users to focus solely on the information they need.	R8
Perform customized data queries to meet the needs of more advanced users	R7
Provide data access ensuring consistent performance and speed, regardless of the time period queried.	R11
Perform both simple and advanced searches.	R8
Aggregate data at various levels.	R10
Export search and aggregation results, facilitating the use of external tools for further manipulation.	R9
Use of fully managed AWS services.	R12, R13

Table 4.2: Requirements tracking for business events data access

No requirement imposes a strong constraint on the system. The access methods provided are standard search functionalities, including filtering, aggregation, and exportation, commonly found in other data access tools. Therefore, although the requirements were derived from internal needs, they did not impose restrictive conditions on system development, as they align with standard search tool functionalities.
As previously mentioned, the only strict constraint is R12, which technically binds the solution to the AWS ecosystem.

4.2 Coverage of use cases

4.2.1 Business Intelligence

In the context of business intelligence, although analyzing and determining the utility of the collected data is outside the project's scope, let us analyze interesting insights that can be extracted from the collected data.

From an aggregation, quantitative information can be obtained about which functionalities were most used during the specified period. Additionally, by comparing different time frames, usage trends of the platform's features can be identified, providing insights into historical usage patterns.
By observing the aggregation values, one could highlight anomalous behavior from a client, such as an unusually high number of certain business events, which may indicate non-compliant use of specific functionalities.
The numerical data on which features are most used by clients is particularly valuable. It enables the strategic team to understand which functionalities are most popular and how they are used, potentially leading to tailored pricing policies or similar strategies based on how the platform is used.

A generic search provides qualitative information about the operations performed by the tenants. By examining how events are logged for a particular tenant, usage patterns of the platform's features can be identified. This is critical for development purposes, as it helps to guide future platform enhancements by improving workflows based on the identified usage patterns.
Additionally, general information can be gathered about the lifecycle of entities (particularly contents and products). For example, the history of operations on a specific entity can be reviewed to identify issues with platform usage or to study how different clients manage various entities, in order to drive the future development efforts based on these insights.

Overall, while aggregations are more suited to BI purposes, generic searches are more applicable to other use cases. However, searches can still provide valuable insights, as they offer access to a range of information that is not available when data is presented in an aggregated form.

4.2.2 Other use cases

Development

The development team needs to monitor business events within the platform. This is crucial for identifying potential bugs, such as event flows that do not trigger correctly or anomalies in event handling.
It is particularly valuable when dealing with external bug reports that are difficult to reproduce. In such cases, developers can access the historical event logs to determine what caused the error, making the system a powerful tool for debugging.
With the web console provided by the system, developers can freely construct and execute SQL queries to explore the database in detail, tailored to their specific needs.

Help desk

The help desk team typically assists customers when they need support or report issues with the platform. Since customers often lack technical knowledge, they are usually unable to provide precise information about the problem or how it occurred. This can make it challenging for the team to diagnose the issue based on the limited information provided.
With the system that logs all business events without omitting any details, this process becomes more straightforward. A team member can use the API (or a dedicated

front-end interface) to retrieve events even starting from minimal information.
Once the target event is identified, further searches can retrieve the entire event history of interest. This allows the team member to clearly understand what happened, making it easier to interpret the user's actions, which are often unclear. As a result, providing guidance on how to resolve an issue or perform a specific operation becomes much simpler.

Customer success and product owner

For customer success team and the product owner, whose purpose is to guide customers in using the platform effectively and drive future development, aggregated quantitative data on how each customer uses the platform's features may be necessary.
By leveraging the system's features (search, aggregation, and export), they can easily obtain an overview of how and to what extent the platform is being used. This insight can then inform development directions or actions aimed at enhancing the platform's quality and features.

Enterprise clients

External clients may require tracking the operations they perform within the platform. Regardless of the type of tracking or the client's objectives, it is necessary to provide them with the ability to export recorded business events in a machine-readable format. To meet this requirement, the export functionality is the most suitable solution. It allows external integration systems to retrieve a dump of business operations (filtered by specific criteria, such as a time range) and obtain the result in a machine-readable file format, such as CSV. This enables the integration of the audit system with external tools, therefore extending the platform's utility and functionality.

4.3 Outlook

In conclusion, Audit Trail Platform effectively archives the business events generated by the THRON platform and makes them accessible through various methods to a wide range of users, each one able to use this data to address specific use cases. As highlighted in section 4.2, these capabilities are fully aligned with the challenges and corresponding proposed solutions discussed in chapter 2.

4.3.1 Current architecture and limitations

The system, in all its functionalities and components, has been developed using *state-of-the-art* technologies, design patterns and development practices, particularly within the AWS ecosystem and its offered services.

From a functional perspective, the system is mature, stable, and built on a well-established architecture that does not rely on temporary solutions to address the issues it targets. However, from a technical and implementation point of view, there are some limitations that, while not problematic, should be considered for future development. These primarily concern performance and overall efficiency in delivering functionalities. These are minor implementation details, mainly related to the configuration of the AWS services used, which could be optimized to enhance performance and reduce costs.

A notable example is the limitation related to the export functionality. As highlighted in section 3.3.2, exporting large amounts of data requires allocating a significant amount of memory since the file is generated locally. While this does not cause inefficiency, it constitutes a challenge to cost reduction, as it necessitates higher memory allocation for the Amazon Lambda function, leading to increased costs. A more efficient approach would be to use Amazon S3's streaming upload capabilities to generate and upload the file in chunks. This method would reduce the memory required for file creation, therefore containing related costs. Additionally, combining this feature with the use of Go channels and Goroutines can significantly improve the efficiency of both the generation and upload phases, which is the only aspect of the system where efficiency has been slightly compromised.

4.3.2 System evolution

Deferred objectives

Regarding the deferred objectives outlined in section 2.2.4, it is challenging to define a clear direction for the future development of the system. This is primarily due to the fact that the business intelligence approach within THRON is still in its early stages. Therefore, the initial focus should be on establishing the BI process and running it for a sufficiently long period to assess the utility of the system and the BI process itself.

Given these premises, before going on with the development of new features and components, it is essential for data analysts to work on an initial sample of the collected data to understand its value and informative potential. This step is crucial for the establishment and implementation of the subsequent stages of the BI process that were deferred during the development of this project. Once the true potential of the collected data is understood, it will be possible to determine the future development direction for both the THRON platform and this system.

Features extension

In the long term, the system is designed to easily integrate new use cases, corresponding to new methods of accessing information. Data access is centered around SQL queries, so if there is a need to provide data in a specific format that facilitates analysis, new RESTful APIs can be exposed, leveraging the existing infrastructure to generate the necessary queries.

An example of this is the introduction of aggregation capabilities. Initially, the system only supported generic searches. However, recognizing the need for quantitative information, the system was extended to support in-place aggregations. This extension significantly aided the early stages of analysis, as manual aggregation would have been challenging due to limited computational resources to be used while analyzing large amounts of data.

The primary role of Audit Trail Platform, beyond logging activities within the platform, is to provide users with easy access to data in various ways. Therefore, any improvement in data access should be considered and, if reasonably feasible, implemented. This tool must be user-friendly and efficient, ensuring that anyone needing information on business events can access it without difficulty, therefore simplifying the process for all users.

4.3.3 Personal considerations

As a student, this project turned out to be interesting, challenging, and far from trivial. The concept and basic idea are quite simple, as it can be reduced to a mere log-tracking system. However, the development of the solution required complex reasoning on various fronts, such as how to create the infrastructure, which AWS services to use, and how to make each component of the system efficient.
Starting from a simple solution capable of addressing all the initially posed problems, I mainly worked on details, sometimes even very marginal ones, to improve it step by step. This involved an in-depth study of the operating and cost logic of the AWS services used, trying to make the most of each individual functionality while minimizing costs. It is not always true that the best solution is the most elegant and efficient; sometimes the best solution is the most economical, though this depends on the application and business context.
Another very interesting aspect concerns the system architecture, where considerable time was spent looking for a way to promote asynchronous processing, scalability, and flexibility in every component of the system. It was not easy to define a system designed to scale and be asynchronous, extendable at the same.
In conclusion, this project provided the opportunity to mature both from an engineering and development perspective. Every small problem encountered became a challenge that allowed me to connect academic knowledge with real-world needs. This was very important because it enabled me to experience well-known issues and address them using technologies that represent the state of the art for distributed systems developed on the cloud. It is a significant sign of how many theoretical and abstract notions eventually find practical application and must be addressed daily within the IT world.

As a developer in THRON, this journey provided my first real experience working on a reasonably complex project. Being integrated into a team context greatly aided the design and development phases. Firstly, it allowed me to have a well-defined work plan, which, combined with the use of the Scrum development framework, significantly contributed to the success of the project. Frequent interactions with the software architect, the product owner, and stakeholders allowed the system to be developed incrementally and to solve issues as they arose. The ongoing dialogue ensured that I always had a clear understanding of the situation, keeping development times and deadlines in check.
Additionally, having the opportunity to discuss project-related topics and issues with people outside the project helped me view them from different perspectives. This was especially beneficial during the design phase, as it allowed me to learn from design mistakes and how to address them.
The contribution of every team member to the project, even in minimal ways, proved to be important. Sometimes it's not necessary to reinvent the wheel; simply discussing the problem with someone more experienced can provide new insights into both the issue and the solution, just by looking at them from different perspectives.

From THRON's perspective as a company, this project originated from the need to lay the groundwork for developing a business intelligence process within the organization. The added value that Audit Trail Platform can provide is not limited to the BI context, but can also contribute to the improvement and simplification of the duties carried out by various company departments. This makes the system very promising, especially if expectations will be met once the system will be widely adopted within the organization.

It is interesting to see how, starting from a very simple idea, one can end up creating a system that, while simple in nature, has great potential for positive impact within a whole company.

Appendix A

Features and requirements

A.1 Ingestion

Functionality	**BI process stage**	**Requirements fulfillment**
Attach to the event bus and collect business events, applying buffering policies.	Data collection	R1, R4, R12, R13
Transform raw events into structured ones.	Data preparation	R1, R3, R12, R13
Store transformed events into the data warehouse in an efficient manner.	Data preparation	R1 R2, R5, R11, R12, R13

Table A.1: Features and satisfied requirements for events ingestion

Service	**Feature**	**Requirements fulfillment**
EventBridge	Fully-managed AWS service	R12, R13
EventBridge	Emitting events to any attached component via defined rules	R1
Data Firehose	Fully-managed AWS service	R12, R13
Data Firehose	Collecting events from an event bus defined on Amazon EventBridge	R1
Data Firehose	Allowing many different AWS services as data sources	R1
Data Firehose	Buffering optimizations in space and time	R1
Data Firehose	Storing batches of data in columnar format (i.e. Apache Parquet)	R5 R11
Lambda	Fully-managed AWS service	R12, R13
Lambda	Custom code to define event-independent transformation logic	R1, R3
Glue	Fully-managed AWS service	R12, R13
Glue	Storing table schema of transformed events	R1, R3
S3	Fully-managed AWS service	R12, R13
S3	Low-cost object storage	R1, R5
S3	Custom lifecycle rules defined on objects	R2

Table A.2: AWS services, features and satisfied requirements for events ingestion

A.2 Search and aggregation

Functionality	**BI process stage**	**Requirements fulfillment**
Run search jobs to retrieve events based on specific use cases to satisfy.	Data access (between data storage and data analysis)	R6, R12, R13
Run search jobs to perform aggregations with the aim of retrieving numerical information.	Data access (between data storage and data analysis)	R10, R12, R13
Expose REST APIs to allow external users to query, in an asynchronous manner, the data warehouse seamlessly.	Data access (between data storage and data analysis)	R8, R12, R13
Expose a console interface to allow advanced users to perform any type of query against the data warehouse.	Data access (between data storage and data analysis)	R7

Table A.3: Features and satisfied components for events search and aggregation

Service	**Feature**	**Requirements fulfillment**
API Gateway	Fully-managed AWS service	R12, R13
API Gateway	Expose RESTful APIs for search and aggregation	R6, R8, R10
Athena	Fully-managed AWS service	R12, R13
Athena	Web-based SQL console	R7
Athena	Interaction with Amazon S3 objects stored in Apache Parquet format	R5
Athena	Define custom SQL queries against an S3 data warehouse	R6, R8, R10
DynamoDB	Fully-managed AWS service	R12, R13
DynamoDB	Table to store business events metadata	R6, R8, R9, R10

Table A.4: AWS services, features and satisfied requirements for events search and aggregation

A.3 Export

Functionality	**BI process stage**	**Requirements fulfillment**
Run export jobs to search or aggregate events based on specific use cases to satisfy.	Data access (between data storage and data analysis)	R9, R12, R13
Expose RESTful APIs to allow external users to export, in an asynchronous manner, business events from the data warehouse.	Data access (between data storage and data analysis)	R9, R12, R13

Table A.5: Features and satisfied requirements for events export

Service	**Feature**	**Requirements fulfillment**
API Gateway	Expose RESTful APIs for export	R9
Athena	Define custom SQL queries against an Amazon S3 data warehouse	R9
DynamoDB	Table to store export jobs statuses	R9

Table A.6: AWS services, features and satisfied requirements for events export

Appendix B

Data storing format experiment

An experiment was conducted with two identical ingestion pipelines, one using Apache Parquet format to store elaborated data and the other storing data in JSON format. The results, shown in table B.1, indicate that the Parquet data warehouse required almost half the storage space compared to the JSON format.

Information	**JSON storage**	**Parquet storage**
Number of events stored	997	997
Total size	770 KB	456 KB

Table B.1: Storage insights about data warehouses used for the experiment

To assess the impact on query performance, different queries were executed, and the results of such executions are presented in table B.2. The 5 executed queries are as follow:

1. Given date and time range, retrieve the id and the type of the events registered;
2. Given a time range, find events by type;
3. Given a correlationId, find all the events belonging to that same flow;
4. Find all the events that originated (flow sources) other events;
5. Given a correlationId, find out the source of that particular event, if any.

Query	**JSON**		**Parquet**	
	Data scanned	Execution time	Data scanned	Execution time
1	770 KB (all)	609 ms	47 KB	574 ms
2	770 KB (all)	561 ms	16 KB	600 ms
3	770 KB (all)	1961 ms	10 KB	1907 ms
4	770 KB (all)	2705 ms	19 KB	3101 ms
5	770 KB (all)	1768 ms	9 KB	3371 ms

Table B.2: Queries performance between JSON and Parquet data warehouses

The results demonstrate that querying the Parquet data warehouse is more efficient than querying the raw JSON data. Parquet does not only maintains, but also enhances query performance, making it a critical tool, especially for the query phases.

Appendix C

Alternative solutions

C.1 Ingestion

Two alternative solutions were considered in addition to the implemented approach. The first alternative also used AWS services, while the third option explored external services.

AWS alternative solution

An alternative approach using only AWS services is shown in figure C.1.

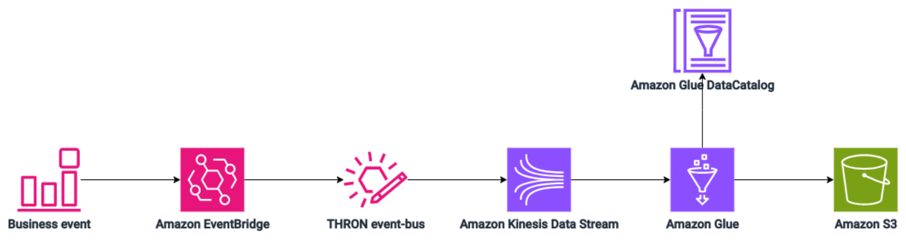

Figure C.1: Alternative events ingestion solution using AWS services

This approach centers on Amazon Glue. It requires a rigid schema defined in the AWS Glue Data Catalog to specify the structure of raw business events. Based on this schema, transformations can be defined for individual fields, including type changes, field renaming, and unnesting nested fields, resulting in a destination schema representing the transformed business events.

For the transformation phase, Glue requires a source to extract business events. This is facilitated by Amazon Kinesis Data Stream, which offers more source and destination options compared to Amazon Data Firehose, which does not support Glue as a destination. The method of attaching the stream to the event bus is analogous to the previously presented solution.
In this setup, Glue processes the stream, applies transformation rules, and writes the data to Amazon S3, similar to the primary solution. However, the implementation complexities are significantly higher in this approach. The main challenge lies in the transformation phase, as Glue lacks the flexibility of Amazon Lambda functions, which

allow for a broader range of operations on input data. This limitation complicates the mapping of event payloads, potentially resulting in a destination schema that is dependent on the original event structure.

While Glue is more self-managed and reduces user workload, its reduced flexibility compared to a Firehose-based pipeline diminishes its appeal. Additionally, the cost of setting up a working environment with Kinesis Data Stream and Glue is considerably higher than the chosen solution. The combination of reduced flexibility and higher costs led to the decision to discard this alternative.

On-premise alternative solution

This approach involves using infrastructure and services external to the AWS ecosystem. However, this option was not explored in detail, as it does not meet the requirements outlined in R12 and R13.

Even if considered, this solution would have introduced significant challenges, including complex mechanisms for performing custom transformations. Additionally, the economic impact would have been worse than the chosen solution, as the cost of managing external infrastructure and an ETL service would likely exceed the cost of the Amazon Data Firehose pipeline. Furthermore, additional expenses related to data transfer outside the AWS ecosystem would have arisen, particularly due to vendor lock-in.

C.2 Search and aggregation

Alternative solutions were not explored, as Amazon Athena is currently the only method available for querying the S3-based data warehouse. The combination of AWS services used to develop the RESTful API interface represents a standard practice. Although alternative solutions could be evaluated, they are unlikely to offer significant advantages over the chosen approach. Moving outside the AWS ecosystem for RESTful interface development would introduce similar issues as described in section C.1.

C.3 Export

An alternative simpler method could have been a single Amazon Lambda function executing the query, waiting for completion, and generating the export file. This approach would have resulted in inefficiencies, higher resource consumption, and increased costs. The selected architecture optimizes execution efficiency and cost-effectiveness.

No alternative solutions were explored due to the constraints associated with Amazon Athena, as discussed in section C.2.

Glossary

Agile project management methodology focused on iterative development, flexibility, and collaboration, where teams deliver small, incremental improvements to a product or project, allowing for rapid adjustments based on feedback and changing requirements. 45, 67

Amazon Athena serverless query service for analyzing data stored in Amazon S3 using SQL. It supports querying of structured, semi-structured, and unstructured data without requiring infrastructure management. Athena integrates with the AWS ecosystem and efficiently handles complex queries. 39–43, 64

Amazon Data Firehose AWS managed service for real-time data streaming to various destinations, such as data lakes and data warehouses. It allows an easy and flexible implementation of ETL jobs, while also supporting automatic scaling, data batching, compression, and encryption. 32, 35, 36, 63, 64

Amazon DynamoDB fully managed NoSQL database service provided by AWS, designed for fast, predictable performance and seamless scalability. It supports key-value and document data models, enabling flexible schema designs. DynamoDB is optimized for handling large-scale workloads with consistent low latency, making it suitable for high-throughput applications such as real-time data processing, gaming, and IoT. 40, 43

Amazon EventBridge fully managed, serverless, and scalable event bus that facilitates integration between AWS services, SaaS applications, and custom applications. It allows events to be ingested, filtered, transformed, and delivered with custom data attached. It supports defining multiple event buses based on the source of the events, whether they come from AWS services or custom applications. It uses rules to match incoming events and direct them to the appropriate targets. Each rule can route events to multiple targets simultaneously, either based on event patterns (events matching a certain structure) or schedules (time-based triggers). 22, 24, 28, 33, 40, 57

Amazon Glue fully managed ETL (Extract, Transform, Load) service provided by AWS that automates data discovery, preparation, and integration. It is used to catalog, clean, and transform data from various sources, making it ready for analytics or machine learning applications. 63

Amazon Kinesis Data Stream real-time data streaming service provided by AWS that allows users to collect, process, and analyze large streams of data in real-time. It is used for applications such as log and event data collection, real-time analytics, and machine learning, enabling scalable data ingestion and processing. 63

Amazon Lambda serverless computing service that executes code in response to various events within the AWS ecosystem. It scales automatically, charges based on compute time consumed, and is ideal for building scalable, cost-effective applications. 39, 40, 44, 47, 53, 63, 64

Amazon S3 (Simple Storage Service) scalable object storage service provided by AWS, designed for secure, durable, and highly available data storage. It is suitable for use cases such as data backup, archival, and large-scale data lakes, with a pay-as-you-go pricing model. S3 integrates with AWS services and supports features such as versioning, lifecycle policies, and fine-grained access controls, ensuring efficient data management and security. 35, 41, 44, 47, 53, 59, 60, 63, 65

Apache Parquet columnar storage file format optimized for large-scale data processing. It provides efficient data compression and encoding, reducing data storage size and improving query performance by minimizing the amount of data scanned. 11, 35, 57, 59, 61

C4model framework for visualizing and describing the architecture of software systems. It consists of four levels of abstraction: Context (high-level overview), Container (major components and their interactions), Component (detailed design within containers), and Code (detailed class-level design), aiming to provide clear and structured documentation of system architecture. 32

CSV file format used for storing tabular data in plain text, where each line represents a row and values are separated by commas. It is commonly used for data import and export between applications. 26, 27, 29, 41, 42, 44, 52

Customer success team dedicated to helping customers achieve their goals with a company's products or services, focusing on building long-term relationships, providing proactive support, and driving customer satisfaction, retention, and growth. 20, 52

Data silos isolated collections of data accessible only to specific departments or systems, hindering information sharing and collaboration across an organization, often leading to inefficiencies and fragmented decision-making. 14

Enterprise large-scale business or organization, typically with complex structures, operations, and needs, often requiring comprehensive solutions and technologies to manage processes, resources, and growth effectively. 20

Go channel concurrency mechanism in Go that allows goroutines to communicate and synchronize by passing values between them. Channels enable safe data sharing without explicit locking, supporting both unbuffered (synchronous) and buffered (asynchronous) communication. 53

Goroutine lightweight, concurrent function execution in Go. It runs independently of each other and is managed by Go's runtime scheduler, enabling efficient multitasking without the overhead of traditional threads. 53, 66

HATEOAS constraint of RESTful architecture that allows clients to interact with a REST API entirely through hypermedia links provided dynamically by the server, without requiring prior knowledge of the API structure. 40

Help desk team responsible for providing technical support and assistance to users, typically addressing IT-related issues, troubleshooting, and resolving problems to ensure smooth operations within an organization. 20, 51

JSON lightweight data interchange format used for transmitting structured data between a server and a client. It is easy for humans to read and write, and simple for machines to parse, with data represented as key-value pairs. Commonly used in web APIs and applications. 35, 39, 41, 61

OpenAPI standard specification for defining RESTful APIs, allowing developers to describe the structure, endpoints, and operations of an API in a machine-readable format. 46

Product owner key role in agile methodologies responsible for defining the vision of a product, prioritizing the product backlog, and ensuring the development team delivers features that align with customer needs and business goals. 20, 52, 54, 67

Scrum agile framework for managing complex projects, where work is divided into short, iterative cycles called sprints. A Scrum team consists of a product owner, Scrum Master, and development team, focusing on delivering incremental improvements through continuous feedback and collaboration. 45, 54

Tenant in cloud computing, a tenant refers to a single client or customer that shares resources in a multi-tenant architecture, where multiple users or organizations securely share the same infrastructure while keeping their data and operations isolated. 18, 22, 24, 36, 51

Vendor lock-in a situation where a customer becomes dependent on a single vendor for products or services, making it difficult or costly to switch to another provider due to proprietary technologies, high switching costs, or contractual restrictions. 13, 64

XLS file format used by Microsoft Excel for storing spreadsheet data. It supports various features such as formulas, charts, and multiple sheets, and is used for organizing and analyzing data in a structured format. 26, 27, 29, 41, 42, 44

Bibliography

Consulted websites

Amazon API Gateway. URL: https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html.

Amazon Athena. URL: https://docs.aws.amazon.com/athena/latest/ug/what-is.html.

Amazon Data Firehose. URL: https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html.

Amazon DynamoDB. URL: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html.

Amazon EventBridge. URL: https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-what-is.html.

Amazon Glue. URL: https://docs.aws.amazon.com/glue/latest/dg/what-is-glue.html.

Amazon Kinesis Data Stream. URL: https://docs.aws.amazon.com/streams/latest/dev/introduction.html.

Amazon Lambda. URL: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html.

Amazon S3. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html.

Apache Parquet. URL: https://parquet.apache.org/docs/overview/.

C4model software representation. URL: https://c4model.com/.

Definition of business intelligence. URL: https://www.techtarget.com/searchbusinessanalytics/definition/business-intelligence-BI.

Disadvantages of business intelligence. URL: https://www.canvasintelligence.com/5-disadvantages-of-business-intelligence-and-how-to-avoid-them/.

ETL tools: on-premise vs cloud-based. URL: https://www.linkedin.com/advice/3/what-some-advantages-disadvantages-using-cloud-based.

Exploiting business intelligence advantages. URL: https://blog.bismart.com/en/how-do-companies-use-business-intelligence.

Key business intelligence statistics. URL: https://dataprot.net/statistics/business-intelligence-statistics/.

Limitations of business intelligence. URL: https://www.analyticssteps.com/blogs/limitations-business-intelligence-bi.

The business intelligence process. URL: https://www.zoho.com/creator/decode/the-6-essential-stages-of-business-intelligence-bi.

Useful data for business intelligence and analytics. URL: https://www.domo.com/learn/article/what-kind-of-data-is-useful-for-business-intelligence-and-analytics.