POLITECNICO DI TORINO

MASTER's Degree in COMPUTER ENGINEERING

MASTER's Degree Thesis

ETL Connectors

Supervisors

Prof. ANDREA BOTTINO

Mr. DAVIDE DAL FARRA

Candidate

SEYEDALI MOUSAVI

December 2023

Summary

The data ingestion and transformation is an essential element of the data processing pipeline. It mainly involves the Client which has to provide a way to communicate the information back to Evo, usually using some data exchange format and, eventually, uploading those resources to Evo premises. The inefficiency of this method sees both the Client and Evo in dealing with a number of operations: 1. data exchange protocol agreement 2.Client data extraction and transformation (according to the above DEP) 3. Evo data transformation and ingestion
These steps highlights on a first sight the presence of two ETL processes, one on Client premises and the other at Evo, leading to a huge waste of resources (development man/hours) and, most important, opening a number of issues and critical elements like the need to maintain multiple data ingestion pipelines, usually one per each client. The project aims to reduce the effort required at client onboarding, leveraging the Client from developing a custom ETL solution only to speak with our systems and Evo from implementing and maintaining a custom data pipeline for every client. In order to achieve such goal the idea is to retrieve data from client premises and ship the information to Evo either in real-time or on a batch. The tool used to retrieve the data is called connector and could be either a developed standalone application, deployed on client premises or a remote service interfacing with the client ERP through, for example, ODBC protocol and acting as a middle-layer.
The SAP R/3 system (now S/4HANA) operates differently compared to SQL-based systems like Oracle E-Business Suite and Oracle's PeopleSoft systems. The major differences include: The native data manipulation language is ABAP, which is a proprietary SAP language. Table names are cryptic compared to those in SQL-based ERP systems. In addition to database tables, SAP contains logical tables called pool tables and cluster tables. These tables contain multiple physical tables and must be managed differently from SQL-based tables. The SAP connector assists you in managing all these issues. Furthermore, the SAP connector enables you to comply with the administrative and security processes of the SAP environment.

Acknowledgements

From the start, I feel compelled to articulate my heartfelt appreciation for Professor Andrea Bottino. His guidance, like a torch in a dense forest, illuminated the path of my research. Similarly, Davide dal Fara, not just my boss but a mentor in the truest sense, has stood by my side, relentlessly shaping the contours of this thesis. Their combined wisdom, insights, and tireless support form the very bedrock of this work. It's not just about the knowledge they imparted but the challenges they threw my way, urging me to think deeper, reach higher, and tap into reservoirs of potential I didn't even know I had. To them, I owe an immeasurable debt of gratitude.

Diving into personal realms, Fariba, my anchor and now my wife, deserves more than just a thank you. Through thick and thin, her unwavering love and support have been constants. In moments of doubt, her encouraging whispers; in triumphs, her shared joy—both fueled my spirit. This thesis, while a culmination of academic pursuits, mirrors the love we share and the dreams we've woven together. Her reminders that there's passion, heart, and soul behind the words I etch have been invaluable. Fariba, thank you for being my beacon, my reminder of what truly matters in the grand tapestry of life.

Lastly, a nod to the friends I've made at the Polytechnic of Turin University. Their camaraderie, shared laughter, late-night debates, and unwavering support have been integral to this journey. They've left indelible marks on my academic voyage, and for that, I am eternally grateful.

For that reason, in the beginning, let me say thank you straight from my heart. Professor Andrea Bottino. his direction served like a light to guide in a dark forest. the path of my research. Likewise, Davide dal Fara, not only my boss but also a friend of mine. the best in the sense of a mentor who has been with me all along, fashioning them out in me. of this thesis. It is made up of their wisdoms, thoughts, and never-ending aids. bedrock of this work. Knowledge is not the only

thing we learned from them. The struggles they forced upon me; forcing me to dig deeper, reach farther up, and touch. into stores of possible energies that I wasn't aware of! To them, I owe an immeasurable debt of gratitude. My anchor and wife, who dives deep into her own private realm, deserves even more. than just a thank you. Her steadfast love and devotion in good times and bad. have been constants. Her inspiring words in times of doubt and her congratulatory voices in successes. she kept on giving me the feeling she was happy about it, fueling my spirit even more. This being a summary cummination thesis. our shared love and intertwined dreams are reflected in academic endeavors. Her mentions that passion, heart, and soul, not just words, go in what I etch. been invaluable. Fariba, thanks for reminding me about what. In reality, life's grand tapestry would be incomplete without it. Finally, acknowledgement to my friends at the Polytechnic of Turin University. Shared laughs, sleepless nights, intense conversations, solidarity. this has made them an important part of this journey. I have marked indefectibly into my academics. journey, and thank you forever.

Table of Contents

List of Tables VIII

List of Figures IX

Acronyms XII

1 Introduction 1
- 1.1 Pre-process ETL before the main ETL process . . . 2
- 1.2 Thesis objective . . . 2
- 1.3 Utilized datasources . . . 3
  - 1.3.1 AdventureWorks . . . 3
  - 1.3.2 SalesForce RESTful API . . . 3
- 1.4 Thesis structure . . . 4

2 Literature Review 6

3 Technological architecture and design overview 10
- 3.1 Aims and objectives . . . 11
  - 3.1.1 Reliability and stability . . . 11
  - 3.1.2 Scheduling data ingestion . . . 12
  - 3.1.3 Security . . . 12
  - 3.1.4 Concurrency control . . . 12
  - 3.1.5 Compatibility . . . 12
  - 3.1.6 Supporting various data sources . . . 13
  - 3.1.7 Noise detection . . . 13
- 3.2 Tools and Methodologies . . . 13
- 3.3 Expected outcomes . . . 14
- 3.4 Functional Requirements . . . 15
- 3.5 Non-Functional Requirements . . . 15
  - 3.5.1 Functional Requirements Use Cases . . . 16
- 3.6 Diagrams . . . 16

4 Technological design and using and configuring the application 21
4.1 Application Environment 21
4.1.1 Connector configuration 25
4.2 Methods of data exporting 28
4.2.1 Export from the results table 28
4.2.2 Export the data from the Activity tab 30
4.2.3 Tasks 30
4.2.4 A usage of Salesforce RESTful API data source 32
4.3 An intro to the current ETL process and the Connector application usage 34
4.3.1 Data extraction 34
4.3.2 Data transformation 35
4.3.3 Data load 38

5 Conclusions and potential future development 40

Bibliography 43

List of Tables

3.1 Functional Requirements 15
3.2 Non-Functional Requirements 16
3.3 Use case 1, UC1 - User logins to the EVO portal 16
3.4 Use case 2, UC2 - User adds a connector 16
3.5 Use case 3, UC3 - User selects the connector and the data source . . 16
3.6 Use case 4, UC4 - User writes the query and waits for the data . . . 17
3.7 Use case 5, UC5 - User retrieves the data and exports to CSV file . 17
3.8 Use case 6, UC6 - User creates tasks for data extraction 17
3.9 Use case 7, UC7 - User monitors the ran query or the scheduled task 17
3.10 Use case 8, UC8 - User provides the connection string of the database 17
3.11 Use case 9, UC9 - User makes the connector available 17

List of Figures

3.1 High level design 14
3.2 Context Diagram 18
3.3 Use case diagram 18
3.4 Deployment Diagram 19
3.5 Class Diagram 19
3.6 Sequence Diagram 20

4.1 Connector configuration 1 22
4.2 Connector configuration 2 23
4.3 Application startup page 25
4.4 List of connectors 25
4.5 Add a connector 26
4.6 Add a connector - Generated key 26
4.7 Prepare the connector 27
4.8 List of tables 28
4.9 Query execution 28
4.10 Query timer 29
4.11 Small query result 29
4.12 Big query result 30
4.13 Big query result - CSV 31
4.14 Big query elapsed time 31
4.15 Activity queries 32
4.16 New task definition 33
4.17 Tasks list 33
4.18 Salesforce configuration 34
4.19 Salesforce API query 34
4.20 Raw data 35
4.21 Normal select example 36
4.22 Transformed data extraction 36
4.23 Transform pipeline 1 37
4.24 Transform pipeline 2 38

4.25 Main ETL pipeline 39
5.1 Time (hour) used for EVO pre-process 41

Acronyms

EVO
EVO Development s.r.l

CustomerAI
One of EVO's products

ETL
Extract-Transform-Load

SAP
Systems, Applications Products in Data Processing

CSV
Comma Separated Values

FTP
File Transfer Protocol

YML
YAML Ain't Markup Language

FR
Functional Requirements

NFR
Non-Functional Requirements

BPMN
Business Process Modeling Notation

MDD

Model-driven development

RESTful

REpresentational State Transfer

API

Application Programming Interface

DW

Data Warehouse

OASIS

Organization for the Advancement of Structured Information Standards

ISO

International Organization for Standardization

IEC

International Electrotechnical Commission

SQL

SQL query language

FaaS

Function as a Service

ORM

Object Relational Mapping

CRM

Customer Relationship Management

XLSX

Microsoft Excel Spreadsheet

DTSX

Data Transformation Services Package XML File Format

SSIS

SQL Server Integration Services

Chapter 1

Introduction

The ETL process includes three steps, Extract, Transform, and Load. The extraction of data from external data sources usually changes over time because of their schema needs. This causes a lot of requests to change the extraction method and the scripts demanding a lot of back and forth between the data consumer and the data provider. Specially, in an already deployed workflow, a minor change can cause the whole process to fail. These changes will affect the extraction flow because the consumers of the data have their own data model which cannot be changed on a daily routine. Thus, they need to reconfigure the process in order to pass their validation rules and transform the data correctly. Arture Wojciechowski has worked on a framework [1] to manage this process and make it automatic in a way that an erroneous execution of an ETL workflow leads to the reparation of the activities and starts interacting with the external data source. The reparation of the ETL activities is handled by some algorithms that are accessed from the API. Although the reparation process is semi-automatic this has been an approach close to what we are going to do in this thesis. As mentioned earlier, the raw data is usually provided by the data providers, here referring to EVO's clients, and those data should then be used in ETL of the client and then the extracted data will be sent to EVO premises for the next ETL process. The flow then goes through the pre-processing step in EVO premises the starts data ingestion. The data ingestion then is done with SSIS scripts and at the end the data is loaded into the data warehouses of EVO. Due to the frequent changes in the data before the pre-processing step, a need for a tool that can improve this process arises. Whether to make it fully automatic with a framework specially designed for that flow or a semi-automatic framework, can reduce a lot of effort in the whole process. Zineb El Akkaoui, Esteban Zimànyi in a paper about a Model-Driven Framework for ETL Process Development [2] have introduced a framework for model-driven development of Extract-Transform-Load (ETL) processes. The main focus is on overcoming the challenge of ETL process development being time-consuming and

complex, often due to the reliance on specific technologies from the outset. This dependence limits the sharing and reuse of methodologies and best practices across projects using different technologies. The paper proposes using vendor-independent models for a unified design of ETL processes. The Business Process Modeling Notation (BPMN) is utilized for this purpose, allowing for expressive and well-known standard modeling. The proposed framework employs an MDD approach, covering the entire development process, including the automatic generation of vendor-specific code for various platforms.

1.1 Pre-process ETL before the main ETL process

The problem related to continuous changes in the data flow can be partially handled in a way that the flow is pre-processed and then passed to the main ETL process. This phase is preconditioning the data for future ETL stages. Different types of data are collected and go through a series of operations like cleaning and normalizing so that they are consistent and correct before being processed. Removing duplicates, data validation, and fixing non-aligned data are the operations that go under this step. In addition filtering and separations using given criteria are performed at the pre-processing step to make certain that appropriate data is only delivered to the ETL main procedure. This step is important since it helps to drastically lower the complications involved in the core ETL jobs by identifying the data issues as well as format matters at the beginning. Through such pre-processing, the overall efficiency and trustworthiness of the data integration process are improved such that data put in the target system are of high quality and ready for analysis or reporting.

1.2 Thesis objective

This thesis is meant to considerably reduce the effort associated with data exchange between EVO and its customers. This thesis suggests that an ETL connector should be put to the client's premises to improve information effectiveness and reliability. This approach addresses the existing issues involving intricate coordination and lengthy tries at the time of changing data's form. ETL connector plays a critical role in automating most of the tasks related to manual processing and the subsequent communications with respect to these tasks.

This includes a major saving in time for running the ETL connector from the client premises, which is one of the biggest components of the solution. The

automatic adaptation of the data format change in the connector minimizes the unnecessary and time-consuming collaboration with the EVO customers. Automated data flow guarantees quickness and reduces the chances of error or non-aligned data. It eases the pre-processing and validation of data before the transformation that is necessary to allow the data to reach EVO, therefore, making the data integration process easier and effective.

ETL connector also boosts data security and integrity. In addition, the connector provides another layer of security by concealing EVO's data access credentials and sharing minimal data with the system. Proper handling of such confidential information should be consistent with good data management and compliance principles. In general, the thesis is about making alterations in the existing data exchange environment so that time gains and operational efficiency prevail at EVO for their customers whereby all the data processing becomes very smooth.

1.3 Utilized datasources

1.3.1 AdventureWorks

AdventureWorks [3] is a widely known and complete sample database provided by Microsoft, which we will used as the main data source in this thesis. We have made AdventureWorks to be used with MySQL database environment to suit well with the technical framework and research objectives that we have. This makes the database suitable for our analysis because it has many business entities that encompass the data of the customers, products, sales, and employees. Since it is for real business and deep on that manner it makes it a good choice for us, and that is what we need to show what happens in the real business and see the aim of the provided tool in our research. The AdventureWorks in a MySQL setting is also beneficial in terms of assessing the flexibility and efficiency of this dataset on diverse database management systems.

1.3.2 SalesForce RESTful API

Salesforce's REST API is a crucial tool for developers and businesses to interact with Salesforce's platform and access its data and functionality. The REST API allows for the integration of Salesforce with other systems and the development of custom applications that can interact with Salesforce. This is supported by, who highlight that web APIs, including RESTful services like Salesforce's REST API, are widely used by major websites to provide access to their data and functionality (Li et al., 2011) [4]. Additionally, 's comparative analysis of data reading performance from the Salesforce platform using different interfaces emphasizes the significance

of the REST API in enabling efficient data retrieval from Salesforce (Rogalski, 2023) [5].

Furthermore, the REST API provided by Salesforce offers a versatile and flexible means of interacting with the platform. It allows for the manipulation of Salesforce data, such as creating, updating, and deleting records, as well as performing various other operations. This flexibility is essential for developers seeking to build applications that seamlessly integrate with Salesforce. The use of Salesforce's Social Studio social listening platform to download tweets from known IRA accounts, as mentioned, demonstrates the practical application of the REST API in accessing and utilizing data from Salesforce for research and analysis purposes (Melykh & Korbut, 2020) [6].

Salesforce's REST API plays a pivotal role in enabling seamless integration with the Salesforce platform, facilitating data retrieval and manipulation, and empowering developers to build custom applications that interact with Salesforce. Its significance is underscored by its widespread use and practical applications in diverse domains, ranging from data analysis to custom application development.

1.4 Thesis structure

This thesis demonstrates its topic objectives and reason for needing it in the beginning. This section provides an overall review of the main needs and the proposed solution.

So far, we discussed the existing problem of the frequent changes of data sources and its inconsistency with EVO's ETL. Also, the role of the ETL connector is described.

In chapter two, we talk about the literature review. This section will review relevant articles and other studies that have been written by other researchers who have dealt with similar issues of the ETL process. It can help us understand the context and see what solutions other researchers have proposed and what frameworks they have introduced. Although, they do not completely share the same problem the whole idea is common, making the ETL process maintenance easier and more controllable.

Chapter three will focus on the technical structure and design perspective of the thesis' ETL connector. We will see how the application should be configured on both EVO's client's side and EVO's side. Meaning, how the connector should be configured to give access to the data sources that EVO needs and also the interface that enables EVO to extract data from those data sources. Moreover, the technology used for this purpose will be described completely. We will see two scenarios and compare the results when the process is done normally without the provided tool and another once with the provided tool. These results are useful

for understanding how the ETL connector can help to reduce the consumed time between these two scenarios and also the effort for the whole process.

Finally, there will be some other possible ways to improve and extend the functionality of the introduced tool for further improvement in future research that will be discussed in chapter five. The idea is to demonstrate the potential abilities that can be added to the ETL connector tool.

Chapter 2

Literature Review

Numerous publications have worked on the enhancement of ETL process to manage the steps of extracting the data from a data warehouse with various tools, They aim to simplify and streamline the maintenance process by providing algorithms and software to make the process more controllable and easy to maintain. These efforts focus on using advanced software solutions and methodologies to automate tasks within the ETL pipeline, reduce manual manipulation of the data, and increase efficiency. By integrating these enhancements, the ETL process becomes more agile and adaptable, significantly reducing the complexities and overhead associated with its maintenance. Consequently, these advancements represent a notable step in optimizing data warehousing and data integration practices.

The maintenance of ETL processes is challenging due to the complexity of data workflows and the evolving business requirements, which demand efficiency and automation (Theodorou et al., 2014) [7]. ETL processes involve dealing with a large amount of data extraction, transformation, and cleaning tasks, making their design, development, and implementation non-trivial (Belo et al., 2014) [8]. Additionally, the diversity of ETL users and developers, using different models and technologies, contributes to the complexity of ETL frameworks and processes, making them hard to analyze and harness (Theodorou et al., 2017) [9]. Furthermore, the traditional ETL process is time-consuming, and faulty implementations in any of the ETL steps can result in incorrect information in the target data warehouse, necessitating thorough validation (Talib et al., 2016) [10].

To address the difficulties in maintaining ETL processes, various frameworks for the automation of ETL processes have been proposed. For instance, a model-driven framework for the development of ETL processes has been introduced to overcome the drawbacks of traditional ETL processes (Akkaoui et al., 2011) [2]. This framework focuses on model-driven development, which can enhance the automation and efficiency of ETL processes. Moreover, a script-based automation ETL tool has been proposed, which combines scripting and database tools to implement the

ETL processes, proving to be faster than traditional methods (Li et al., 2016) [11]. Additionally, a framework for the design of the Data Warehouse (DW) back-stage and ETL processes has been presented, emphasizing the importance of dealing with the specificities of information at low levels of granularity, including transformation rules at the attribute level (Luján-Mora et al., 2004) [12].

Furthermore, some frameworks focus on testing and optimizing ETL processes. For example, a testing framework has been proposed to automate the testing of data quality at the ETL stage, ensuring the reliability of the ETL processes (Dakrory et al., 2015) [13]. Additionally, some frameworks aim to optimize ETL processes in data warehouses, providing algorithms to minimize the execution cost of ETL workflows (Skoutas et al., 2009). These frameworks contribute to the automation and optimization of ETL processes, addressing the challenges associated with their maintenance.

Overall, the maintenance of ETL processes is difficult due to the complexity of data workflows, evolving business requirements, and the diversity of users and developers. However, various frameworks have been proposed to automate and optimize ETL processes, aiming to enhance their efficiency and reliability.

Vijayendra, Nithin, Meiliu Lu in this article [14] have focused on the development and implementation of a web-based Extract, Transform, and Load (ETL) tool. The main aim of their project is to create a simple, web-based ETL tool accessible to anyone with internet access, supplemented by an online tutorial on the ETL process. This combination is designed to be an ideal self-paced, interactive learning tool for beginners in ETL, allowing for the extraction of data from text files or MySQL tables, integration, and loading into target MySQL tables with the application of various transformations.

The motivation behind this project stems from the challenges faced by learners interested in ETL tools, particularly the high cost and technical complexity of commercial ETL tools, which are often not open-source or web-based. These challenges make it difficult for small or medium-sized project developers, and students learning data warehousing fundamentals, to access and use these tools. The developed web-based ETL tool addresses these issues by being freely accessible, user-friendly, and specifically designed for beginning ETL developers. This tool allows learners to gain practical experience with ETL processes before delving into more complex commercial tools, contributing to the ETL learning community by enhancing the teaching and learning experience at a lower cost.

Another study by Fivien Nur Savitri [15], titled "Study of Localized Data Cleansing Process for ETL Performance Improvement in Independent Datamart" focuses on enhancing the efficiency of Extract-Transform-Load (ETL) processes in data warehousing through localized data cleansing. The article highlights that a significant

challenge in building data warehouses is the ETL process, which is complex due to varied and heterogeneous data sources. The author proposes a concept of localized data source cleansing. This approach involves identifying and addressing inconsistencies, non-formal expected existing data and duplicates in each data source's profile at the local level. The Localized data cleansing is expected to reduce and streamline the ETL process, thereby improving its performance. The study includes an investigation into the impact of localized versus non-localized heterogeneous data cleansing. Based on this investigation, an automatic localized data cleansing and integration system is defined. The proposed system involves cleansing processing for each data source profile, executed at the transactional data source site, before the data warehouse development stages. It is found that sequential execution of the Automatic Data Cleansing process and the Data Integrator process can significantly reduce the ETL processing workload in data warehouse development stages. This reduction is especially notable for data lacking integrity constraints and format-checking procedures. The research employs a bottom-up approach and focuses on independent datamarts. It considers that automatic data cleansing is highly required for unintegrated data and that data warehouse development is not a prerequisite condition. he study concludes that separating the automatic data cleansing system from the data warehouse development and performing it at the source location can improve ETL performance. This is achieved by dividing the ETL process workload among separate resources.

In a paper [16] from Alexander Albrecht, Best Practices and Strategies on best practices for managing ETL processes efficiently have been introduced. This might include strategies for data extraction, transformation techniques, and tips for effective data loading. The document likely addresses common challenges encountered in ETL processes and proposes solutions or workarounds. This could cover issues like data quality, integration of disparate data sources, and performance optimization. Although they have not introduced any tools in that paper they have delved into more advanced topics related to ETL, such as automation, scaling ETL processes, dealing with complex data types, and integrating ETL processes with other data management and analytics tools. The steps taken for the management of the ETL process in this paper, in summary, are Match which identifies similar ETL processes within a repository, aiding in managing large numbers of ETL processes. Similarity could be based on functionality or data sources/targets. Determining a suitable similarity measure is challenging due to the variety and heterogeneity of ETL features. Also, Merging was another operator that combined multiple ETL processes into one. It identifies common sub-processes and merges them, potentially improving resource utilization, reducing data transmission, and enhancing performance. Merged processes offer a unified view of information and reduce overhead and maintenance. And then in Rewrite step is to restructure an ETL

process, optimizing it and fixing design flaws. This could include reordering stages for consistency or efficiency.

Vangipuram Radhakrishna, in the paper "Automating ETL Process with Scripting" [17] discusses improving ETL process data flows for better business investment returns. It emphasizes the need for an enterprise-level scheduling solution that is user-friendly and can handle heterogeneous environments, focusing on automating the ETL process to optimize or enhance it. The paper proposes the use of scripting technologies for automating ETL processes, reducing manual effort, and potentially leading to the development of ETL tools with command-based interfaces.

The primary focus is on automating ETL processes using scripting technologies. This approach aims to reduce the manual effort required in running ETL processes and to handle them more efficiently. The paper suggests using scripting technologies to enable end-to-end processing automation of ETL tools. This could lead to the creation of more effective ETL tools that support command-based or script-based automation. It discusses the challenges in data warehouse operations, which involve handling large volumes of data and numerous data sources with unique access methods, content, and quality. The proposed automation aims to improve the functionality of ETL tools beyond just scheduling ETL processing. This includes better error handling, mapping issues, maintaining audit tables, and logging statistics. The paper anticipates a future need for ETL tools that support automated processing and a command-based user interface for faster and more efficient data processing.

Chapter 3

Technological architecture and design overview

3.1 Aims and objectives

This thesis is based on the development and showcasing of an infrastructure allowing EVO to log into customers' DBs before its ETL process. Consequently, it results in many new functions all through. We strive to enhance our goal of narrowing this communication barrier. operations, raise the accuracy of their data, as well as forge alliances among them. EVO and its clientele. This makes the process of data conversion more precise. this also paves the way for innovations that will be game-changing in the future.

1. The thesis aim is a new approach for this paper [1] That visualizes and manages the ETL pre-process in a web-based application.
   (a) Reliability and stability
   (b) Scheduling data ingestion
   (c) Security
   (d) Concurrency control
   (e) Compatibility
   (f) Supporting various data sources
   (g) Noise detection
2. This thesis proposes a complete ETL framework that supports SAP connectors for integrating SAP's many different data structures and business processes into the data warehouse. The compatibility makes it possible to extract durable, credible information from the ETL pipeline, hence an orderly pathway for delivery.
3. Finally, in the third phase, we will assess whether the benefits achieved are more than the input required in processing information before its implementation. It will be significant during the data life-cycle and will measure its effect on the operations' efficiency.

3.1.1 Reliability and stability

Reliability and stability: Throughout ETL operations, it has been observed that internet disruptions can halt the entire process, necessitating a restart after considerable delays. To mitigate this challenge, the web application described in this thesis has been developed. This application methodically extracts data in manageable CSV segments, ensuring a more resilient preparation phase. Thereafter, this segmented data is seamlessly incorporated into EVO's principal ETL process.

3.1.2 Scheduling data ingestion

Scheduling data ingestion: Another feature of the developed web application in this thesis is that we can define scheduling for data ingestion. This can be a simple query for the database, or it can be running a stored procedure on an interval. The result can be saved as a file in the FTP server that we define. The way to create the schedules is simple and easy. The scheduling is defined in YML format and can include all the needed parameters for data ingestion.

3.1.3 Security

Security: An aspect of the ETL process on the EVO's client side is that the data format changes occasionally. These changes of format are not propagated automatically and someone from the client's team should change the exported data for EVO's data model. As far as EVO does not have direct access to their client's data sources, EVO cannot apply these changes directly and they have to ask their clients to apply the change to the exported file. By the standalone application provided as a complimentary for this web application, the client can define the data sources and keep the credentials inside that application. This application provides a secure connection between EVO and their clients. In this way, EVO can directly connect to client's premises and query whatever they need without wasting time. Obviously, the database users that are created for this purpose are defined by their client so the permission is controlled.

3.1.4 Concurrency control

Concurrency control: The probable issues regarding concurrency are also addressed here. It has happened that the scheduling of data ingestion pipelines for one customer of EVO has encountered a problem. Because of this failure, all the other pipelines have failed. The relation between these pipelines is causing the problem and there is no recovery plan to avoid this data ingestion interruption. By using the scheduling feature of the web application introduced in this thesis, the control over the schedules will be done by EVO, and the recovery plan in case of probable failures can be defined.

3.1.5 Compatibility

Compatibility: Additionally, there is an expectation that some of the data will not follow up with the EVO data model. Therefore, a substantial part of this thesis is geared towards data compatibility. Checking for the right data format and where need be, transforming the ingested data to conform to a set and agreed upon data model. The compliance check supports the homogenous handling, and

usage and increases the validity of such empirical results. Overcoming this issue in compatibility will also enhance data integration and improve operational flow in the EVO model.

3.1.6 Supporting various data sources

Supporting various data sources: The application that is the interface of the connector in this thesis has also made it easy to connect to various data sources. Its versatility can be seen through its compatibility with SAP systems that are prevalent in the corporate world as an all-in-one solution set for various business needs, Moreover, it supports various data sources including SQL server and MySql, etc. Firstly, this is important to SAP as it forms bases for essential business operations, coupled with complicated information structures. Moreover, it can also handle the unique customizations and special protocols associated with the typical configured enterprise systems of SAP, making it even more flexible and adaptable. This helps the ETL framework fit into the normal SAP modules thus, extending its functionality even further to suit custom needs that may be required by any organization wishing to centralize its internal processes as well as data assets.

3.1.7 Noise detection

Noise detection: The integration of the EVO querying system has provided a crucial improvement within the refined paradigm of the ETL connector posed within this thesis in the domain of noise detection. However, traditional techniques of transformation very rarely were able to see 'noise' - discrepancies that include erroneous data formats, unrealistic value ranges, wrong information entries, etc. which evade routine detection instruments while transformations are on the go. The innovation of this framework has been in its capacity for pre-extraction querying that queries the datasets on their structural and interrelational compatibility before extraction. Through deep analysis of data patterns, EVO flags the deviant anomalies that do not correlate with the established data models. The system performs query-based inspection before the extraction phase, thus blocking any unwanted signals (noise) and allowing only the clean, validated data to enter the transformation phase. Such preemptive scrutiny lowers considerably the probability of corrupting data and makes the whole ETL process consistent with a modern concept of data quality.

3.2 Tools and Methodologies

This thesis tried to use an approach that is combined with automatic and manual control of the data flow. Considering the needs of both EVO and their client we

have two applications, one standalone that is on the client side which is called "connector" and the other one is the web application that uses those connectors to extract data. At a first stage we will focus on the first part "connector", then we will switch to the web application and describe all the features of it.

3.3 Expected outcomes

The project's outcome will be about measures and reduction of previous failures and prevention of data extraction interruption. The stability of the extraction process managing schedules of data extractions and also modification of the extracted data and prepare them for the expected data model should be guaranteed. The connector will make sure that the client's data source is reachable and ready to be queried and the web application should show how many connectors and of which type are available and start extracting the data from those connectors. Both apps are written in Python using open-faas and also vuejs for the frontend part. The requests both from the web application and the connector go through the API server. And then the result can either be saved on a File-server or returned directly to the front-end app.

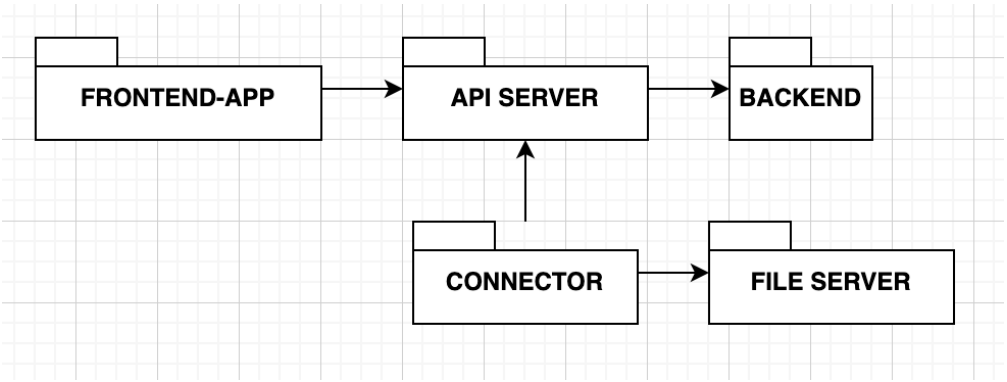


Figure 3.1: High level design

There are two kinds of users in the application, the first group can use the web application therefore, should be authenticated through the EVO's portal, in order to extract data from the client premises and the second group is the one who configures the connectors. The connectors application does not have a graphic user interface so all the connectors will be configured by a configuration file.

3.4 Functional Requirements

Functional requirements or FR dictate what a system should do. They are explicit descriptions detailing what kinds of services a system must offer, how it ought to respond to specified conditions, and how it needs to act in specific cases. In most cases, the details about different features needed in the software are usually included in the use cases, user stories, or system specifications where they explain what the software should be capable of doing, and how it interacts with the users and other systems. The connector as already mentioned, does not have a graphic user interface, hence, the configuration file needs to be set. These requirements are listed in table 3.1.

ID	Description
FR1	User logins to the EVO portal
FR2	User Adds a connector
FR3	User selects the connector and the data source
FR4	User writes the query and waits for the data
FR5	User retrieves the data and exports to CSV file
FR6	User creates schedules/tasks for data extraction
FR7	User monitors the ran query or the scheduled task
FR8	User provides the connection string of the database
FR9	User makes the connector available

Table 3.1: Functional Requirements

3.5 Non-Functional Requirements

Non-functional Requirements: NFRs represent the quality attributes, conditions of operation in its environment as well as standards of a system. They are specified for use for measuring how well the system should work in terms of usability, reliability, performance, maintainability, scalability, and security. As an example, an NFR for a web app would require the UI to be user-friendly, the service level should be guaranteed to a maximum uptime of 99.9%, the number of concurrent users to a minimum of 10k, and any personal NFRs be vital in that they delineate situations under which the system's functionalities take place, and hence influence user satisfaction, the lifespan of the system, and their ability to respond to dynamic needs. It can be difficult to verify non-functional requirements as opposed to functional requirements, but they determine the ultimate quality level of the system in operation. The main non-functional requirements are listed in table 3.2.

ID	Description
NFR1	The application should give the response in a reasonable time
NFR2	Anonymous users cannot and should not have access to either application
NFR3	Support the scalability for various data sources
NFR4	Guarantee the concurrent requests of the users

Table 3.2: Non-Functional Requirements

3.5.1 Functional Requirements Use Cases

Next, we will have the list of all use cases that are implemented in the web application with all the details of the actors that are involved. We will discuss the pre-conditions and post-conditions. Also, the scenarios are described for all the possible actions.

Actors	Involved Anonymous user
Pre-condition	The EVO portal is up and accessible
Post-condition	User is logged in and can see the app
Nominal Scenario	User logins to the portal

Table 3.3: Use case 1, UC1 - User logins to the EVO portal

Actors	Involved logged-in user
Pre-condition	User is logged in
Post-condition	The new connector is added
Nominal Scenario	The system generates a key for the created connector

Table 3.4: Use case 2, UC2 - User adds a connector

Actors	Involved logged-in user
Pre-condition	At least one connector is alive (accessible)
Post-condition	The data source is selected and is ready to be queried
Nominal Scenario	Selecting the exact connector and the data source

Table 3.5: Use case 3, UC3 - User selects the connector and the data source

3.6 Diagrams

Context diagram, use case diagram, and class diagram will be discussed in the next section. Followed by a sequence diagram. The level zero data flow diagram is another name for the context diagram. Structure of hardware/ software components,

Actors	Involved logged-in user
Pre-condition	The data source of the connector is selected
Post-condition	The query is written and sent
Nominal Scenario	Query is sent, refreshing until the data is ready

Table 3.6: Use case 4, UC4 - User writes the query and waits for the data

Actors	Involved logged-in user
Pre-condition	The query is sent and the loading is shown
Post-condition	The result of the query is ready
Nominal Scenario	The application automatically refreshes until data is ready

Table 3.7: Use case 5, UC5 - User retrieves the data and exports to CSV file

Actors	Involved logged-in user
Pre-condition	There is one live scenario
Post-condition	The task is defined and saved
Nominal Scenario	Depending on the configuration, the task can be run after definition

Table 3.8: Use case 6, UC6 - User creates tasks for data extraction

Actors	Involved logged-in user
Pre-condition	There is at least one query/task running
Post-condition	the result file is downloadable
Nominal Scenario	The application can show the list of queries and tasks that are running

Table 3.9: Use case 7, UC7 - User monitors the ran query or the scheduled task

Actors	Involved connector standalone app
Pre-condition	The database credential is available and configured
Post-condition	The connector has access to the database
Nominal Scenario	The configuration of the connector and its data sources are done

Table 3.10: Use case 8, UC8 - User provides the connection string of the database

Actors	Involved connector standalone app
Pre-condition	The connector is configured correctly
Post-condition	The connector is live and can be seen from the web app
Nominal Scenario	The configuration of the connector and its data sources are done

Table 3.11: Use case 9, UC9 - User makes the connector available

input/output resources, and databases. It is used to show how a system responds or interacts with its environment and to specify. The system's boundaries. The present context diagram is illustrated in 3.2. application.

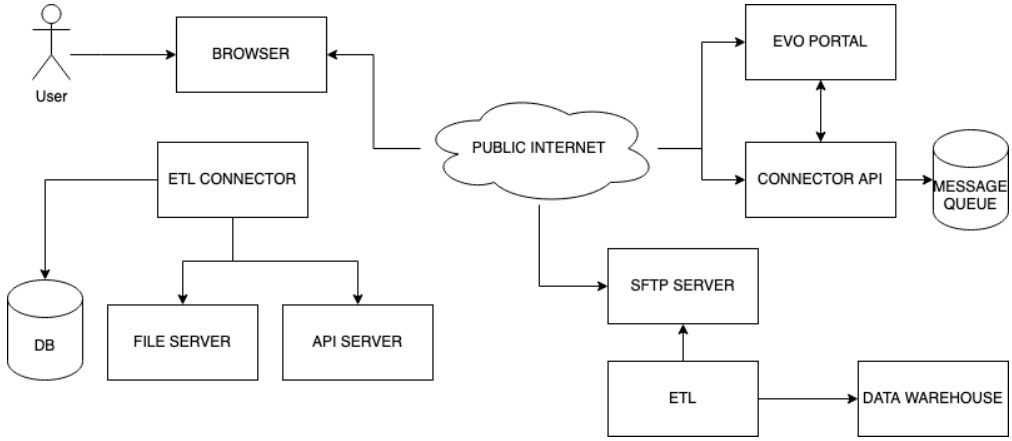


Figure 3.2: Context Diagram

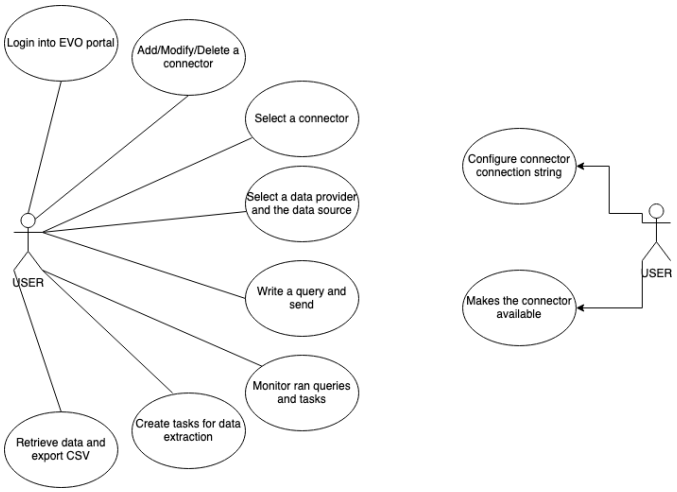


Figure 3.3: Use case diagram

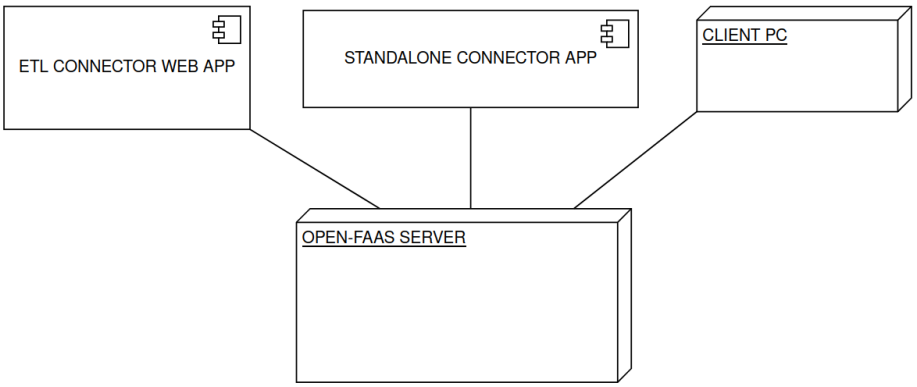


Figure 3.4: Deployment Diagram

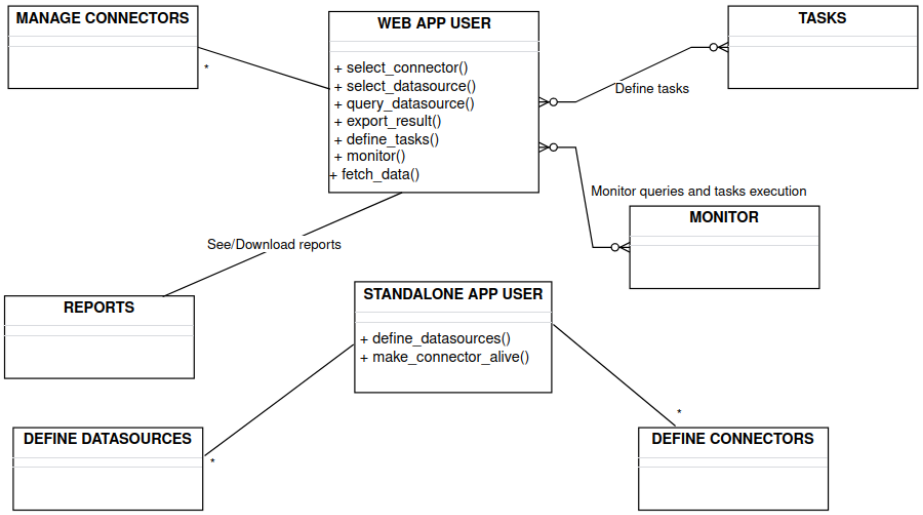


Figure 3.5: Class Diagram

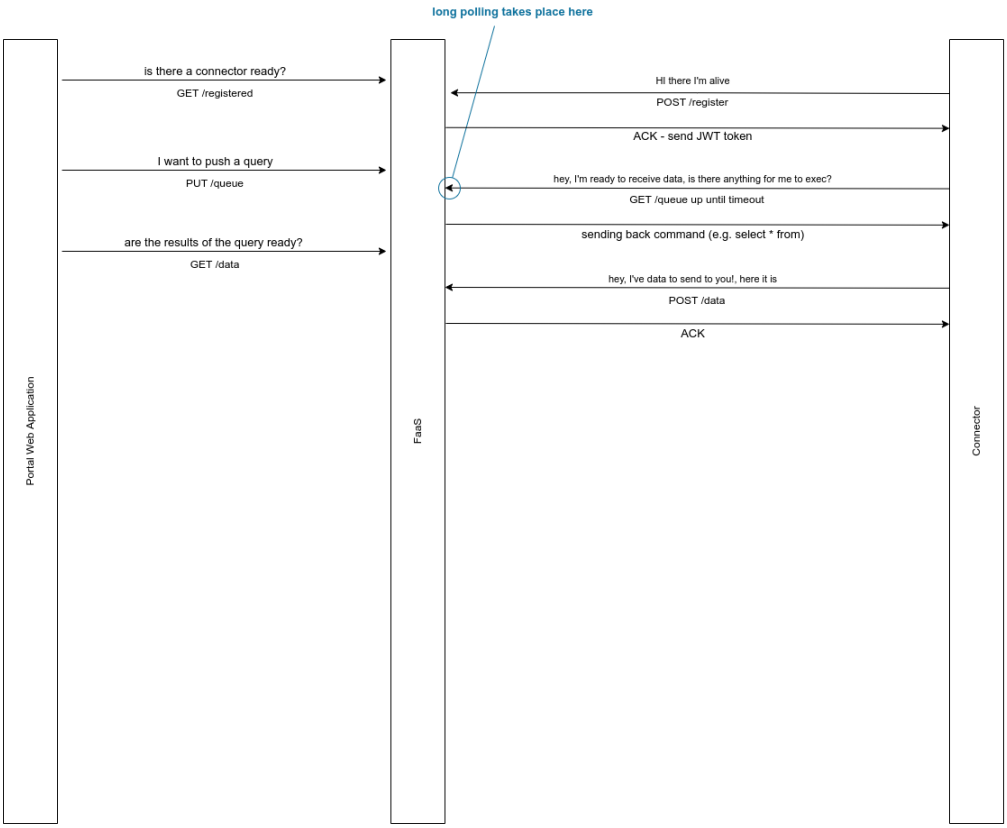


Figure 3.6: Sequence Diagram

Chapter 4

Technological design and using and configuring the application

4.1 Application Environment

In this chapter, the focus is on the application's environment. How to define connectors, make them available for the web application users, and manage the extraction of the data through the web application. Moreover, we will have a comparison between two scenarios of the flow in terms of efficiency and the problems that are addressed.

The application implemented for this thesis is divided into two parts. One is the standalone connector and the other is the web application. The client must configure the standalone app through a .yml file, providing the connector's URL and defining the data sources. The .yml file has a structure like below:

The name of the connector is what is shown in the web app, so a meaningful name identifies the data source. The URL as the default value shows, can even be localhost if it is available locally but usually, it is configured on a server. The tasks can also be defined in the format of a .yml file so that they are usable by the connector directly. Then, we have the data providers section. Depending on the diversity of the data sources, we should have a separate configuration section for any of them. Here, for example, we have provided an odbc data source that is a SQL database with the connection string and an OData [18] which is an open data protocol, recognized as a standard by ISO/IEC and governed by OASIS, establishes

```yaml
connector:
  name: test1234
  apiUrl: http://localhost:8081
  connectionKey: XXX
  connectionTimeout: 60
  localTempPath: /tmp
  tasksPath: /home/operatore/git-evo/etl-connector/etl-connector/config/tasks/task.example.yml


dataProviders:
  odbc:
    # https://docs.sqlalchemy.org/en/14/core/engines.html
    default: mysql-server
    dataSources:
      - name: mssql-server
        connectionString: DRIVER={ODBC Driver 17 for SQL Server};SERVER=server-url;DATABASE=Test;UID=XXX;PWD=XXX;
Authentication=SqlPassword;TrustServerCertificate=Yes
  odata:
    default: sap-sandbox
    dataSources:
      - name: sap-sandbox
        url: https://sandbox.api.sap.com/sap
        headers:
          - APIKey: xxx
```

Figure 4.1: Connector configuration 1

essential guidelines for crafting and utilizing RESTful APIs effectively. It allows developers to concentrate on core business functionality while streamlining the creation of RESTful APIs by standardizing headers, status codes, HTTP methods, URL conventions, media types, and payload structures. Additionally, OData advises on monitoring modifications, formulating reusable functions/actions, and managing asynchronous or batch requests. The protocol simplifies API consumption through its metadata, which provides a detailed model of the API's data structure, facilitating the development of versatile client tools and proxies.

In the end, the output providers should be defined. This will be used to store the results of queries or tasks. As the example refers, the credentials for the FTP server should be provided here.

The technology used for this standalone application is pure Python. This application uses the Open FaaS serverless functions to send the data in a queue that is implemented with RabbitMQ. The Open FaaS [19] is a framework that allows you to run stand-alone serverless functions independent of any particular cloud vendor. This is constructed on a platform based on container technologies that enable developers to bundle code with its dependencies in small and agile parts. Designed to be easy to use and efficient, OpenFaaS functions seamlessly by delivering all the intricate infrastructure services including scaling, networking, and health checks. A user can call on the same system for executing a function programmed in a language such as node.js, python or go, amongst others by simply issuing an HTTP request, subscription to a certain event, or through

```
outputProviders:
  localFilesystem:
    path: /home/operatore/git-evo/etl-connector/etl-connector/output
    compress: false
  sftp:
    default: upload.evofiles.com
    trust_all: true
    compress: true
    servers:
      - name: upload.evofiles.com
        # connection string format is username:password@host:port
        connectionString: XXX:XXX@upload.evofiles.com:22
        defaultUploadPath: /
```

Figure 4.2: Connector configuration 2

direct invocation. Therefore, open faas is ideal in the construction of Microservice, execution of asynchronous tasks, and generating workflow automation enabling the transition towards a modular and scalable approach to App development and provisioning.

The API's are written using the Flask framework [20]. It is a simple and powerful micro-framework in Python meant for rapidly creating web apps. However, it is easy to expand because of its simple structure of central core which does not involve any database abstraction. Rather than that, Flask enables extension with functions similar to those of Flask itself as an additional feature to applications. Unlike ORM built-in others, Flask allows developers free choice and straightforward development of services or any complicated applications. Developers find Django straightforward, flexible, and provides granular control, hence favoring the creation of lean maintainable Web apps with very little headache.

We have also used MongoDB [21] the leader among the non-relational databases is definitely MongoDB which has adopted cutting edge methods of storing and retrieving data. Unlike the classical relational databases using tables and rows, MongoDB employs the document-oriented model for data storage, which allows one to operate with the documents with the JSON layout. With reference to some categories of apps handling vast amounts of varying, loose and quickly evolving information, this makes the process of data integration easier and user-friendly. The ability of mongodb to adjust to the ever changing need for the data makes it the most valued aspect in the dynamic world today. Scalability in horizontal direction, strong indexing features, and strong aggregativeness increase its popularity among software engineers and enterprises looking for suitable options of effective processing of big amounts of information in one place. MongoDB's support system has been

built around its active community as well as its flourishing ecosystem containing vast number support tools and integrates that make it more than just a mere database, instead it becomes an essential building block of modern software applications development.

About the RabbitMQ [22] it is a more secure, scalable, maintainable messaging queue software than the existing ones. To be good at developing messaging programs, one should know well the efficiency of using RabbitMQ. With this guide for messaging using RabbitMQ, you can head straight towards building your messaging solutions. In this article, you follow the story of a hypothetical company named Clever Coney Media as they progress from the standard, synchronous approach to complex message routing and application monitoring methods using RabbitMQ. With RabbitMQ you get a feature to manage message queuing software and create distributive and scalable apps. With time you will move on to create custom messages inbox using special queues as well as different exchange types.

The technology used for the front-end is VueJS. After running the app based on the configuration of the project you will have the app running on localhost:port. Here is the first page of the application:

The tools and frameworks that are used in this project are all open-source. Open-source software has gained significant attention due to its collaborative and transparent nature, allowing for the sharing of source code and encouraging community involvement in software development (Kavanagh, 2004) [23]. The concept of open source is rooted in the "The Cathedral and the Bazaar" philosophy, which emphasizes the benefits of decentralized and community-driven software development ("The cathedral and the bazaar", 2000) [24]. This approach has been particularly valuable in fields such as computer science, where open-source software like OpenMEEG has been instrumental in advancing quasistatic bioelectromagnetics research (Gramfort et al., 2010) [25]. Additionally, open-source development has proven to produce reliable and innovative computer code at lower costs than proprietary software through sharing development responsibility with a large community of invested individuals (Buitenhuis & Pearce, 2012) [26].

Furthermore, the use of open-source software extends beyond computer science, with its application in fields such as renewable energy, where open-source development of solar photovoltaic technology has been explored (Buitenhuis & Pearce, 2012) [26]. The benefits of open source are not limited to technical domains, as it has also been adopted as a global sourcing strategy termed "open-sourcing," allowing commercial companies and open source communities to collaborate on software development of commercial interest (Ågerfalk & Fitzgerald, 2008) [27]. Moreover, the ethical and philosophical benefits associated with the use of open-source software have been recognized, further highlighting its significance in various domains (Omopupa et al., 2020) [28].

The impact of open source extends to academia, where it has facilitated research

and development by providing publicly accessible software and tools. For instance, in the field of deep learning, the use of open-source software such as Python, TensorFlow, and pyGDM has been crucial in pushing the limits of optical information storage (Wiecha et al., 2019) [29]. Additionally, the availability of open-source datasets and tools has been instrumental in advancing research in areas such as computer vision and artificial intelligence, enabling reproducible workflows and fostering collaboration within the scientific community (Neupane & Seok, 2020; Kitlasten et al., 2022) [30] [31]. Overall, the widespread adoption of open-source software has not only transformed software development but has also significantly contributed to advancements across various scientific and technological disciplines.

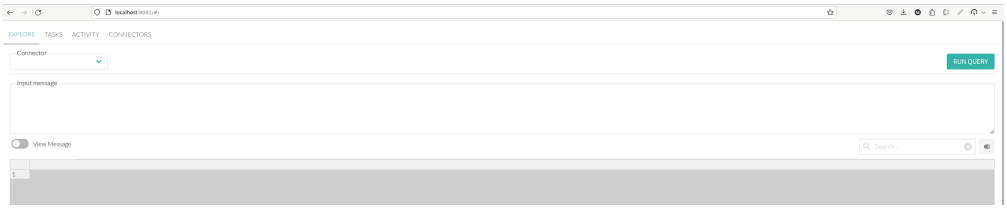


Figure 4.3: Application startup page

4.1.1 Connector configuration

As soon as the connector is configured correctly on EVO's client side we can see the status of all available connectors through the dropdown list in the first tab, or we can see them in the Connector's tab.

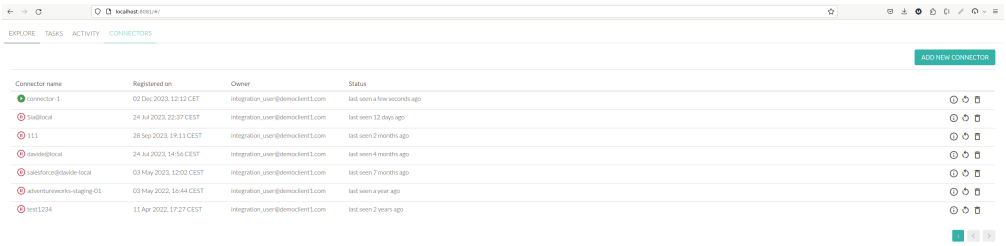


Figure 4.4: List of connectors

As you can see the first connector, the one that we have prepared for example, is alive and the other connectors which can be any data source for any clients of EVO are listed and their status can be tracked.

The way connectors are added is through the "ADD NEW CONNECTOR" button. First, a name should be given to the connector, better to be a name close

to the EVO's client data source. Then a key will be generated that will be used inside the connector app that is running in EVO's client's machine. Below you can see the steps to create a new connector, figures 4.5 and 4.6.

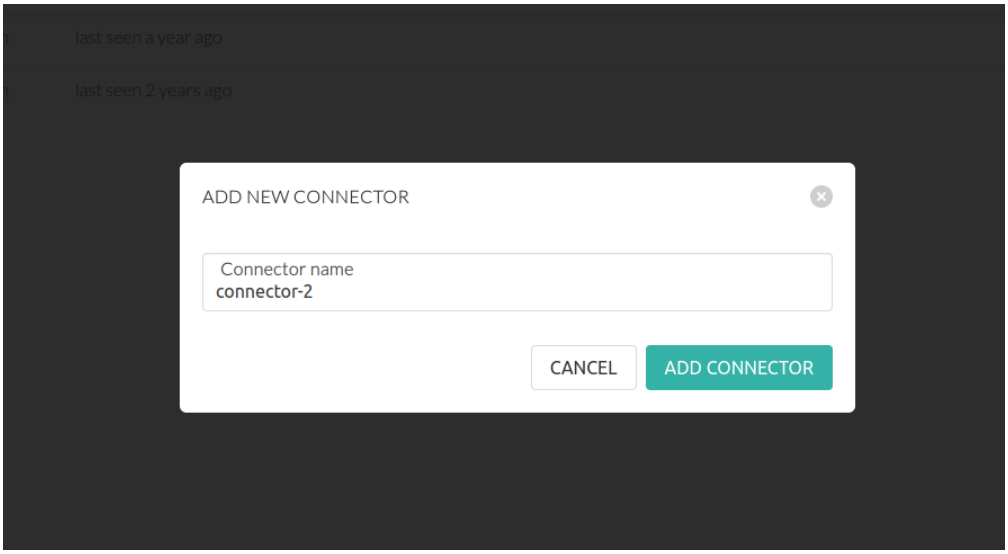


Figure 4.5: Add a connector

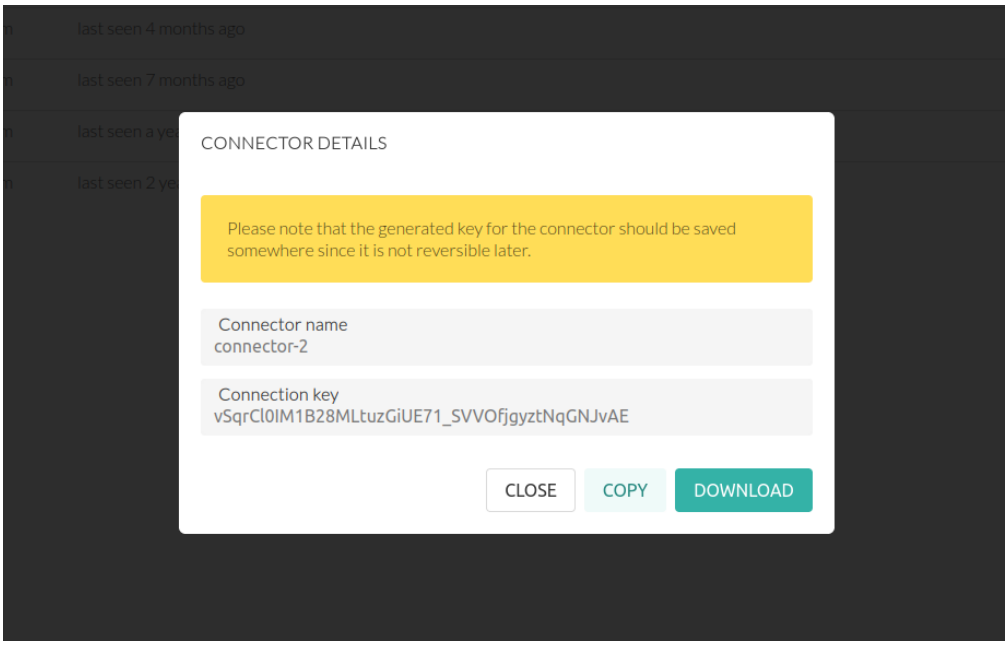


Figure 4.6: Add a connector - Generated key

The generated key will be used inside the connector.yml file on this line:
`connectionKey: vSqrCl0IM1B28MLtuzGiuE71_SVVOfjgyztNqGnJvAE`

Now going back to the first tab which is the main tab of our application. In the first tab, we can select an alive connector only and the data provider and the data sources that are provided for that connector. The type of operation for now is just running a query but as said earlier, this application can be extended to do other operations on the data sources.

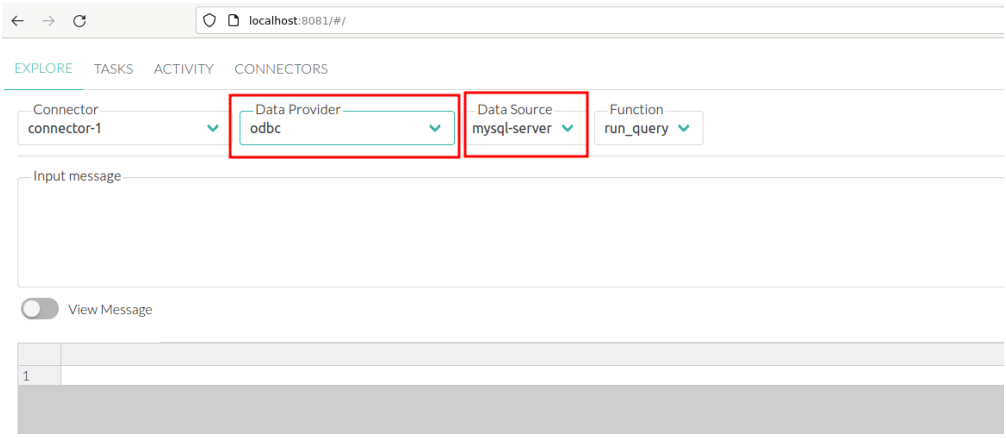


Figure 4.7: Prepare the connector

After selecting the data source we can start querying it. Starting from this query to see all available tables. See figure 4.8.

```
Show Tables;
```

We will see the list of tables that we have access to, we can easily write queries on any of them. For example, let's try querying the Person_Person table. After writing the query and pressing the "RUN QUERY" button, the polling will start and will continuously call an endpoint to see if the data from the client's machine is available or not. The user will see loading in the middle of the screen and as soon as the data is ready, it can be seen in the provided table. See figure 4.9

Also, the elapsed time will be shown on the title of the page in the browser and the operation can be stopped at any moment with the X button that appears next to the "RUN QUERY" button. This is useful when a query takes longer than expected. Below you can see the timer of the operation in figure 4.10

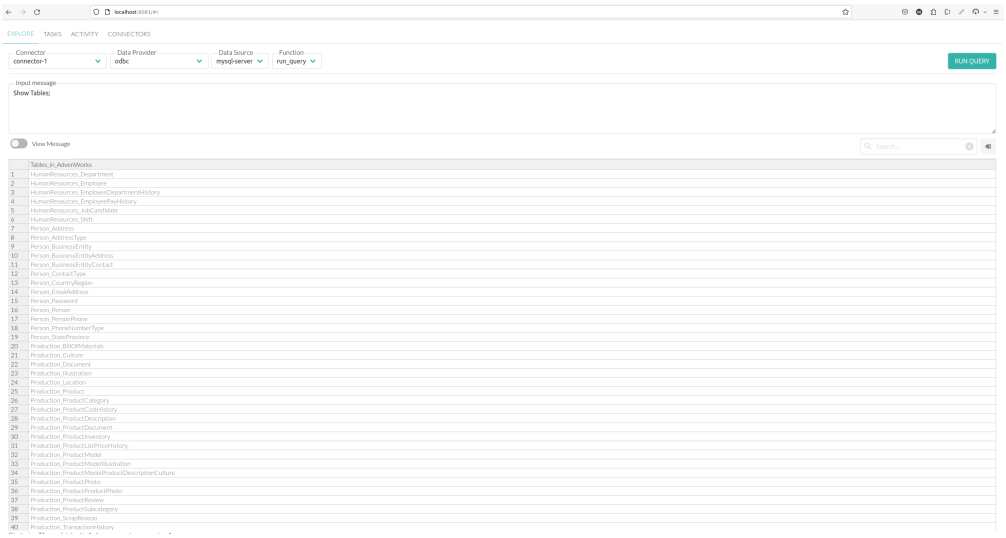


Figure 4.8: List of tables

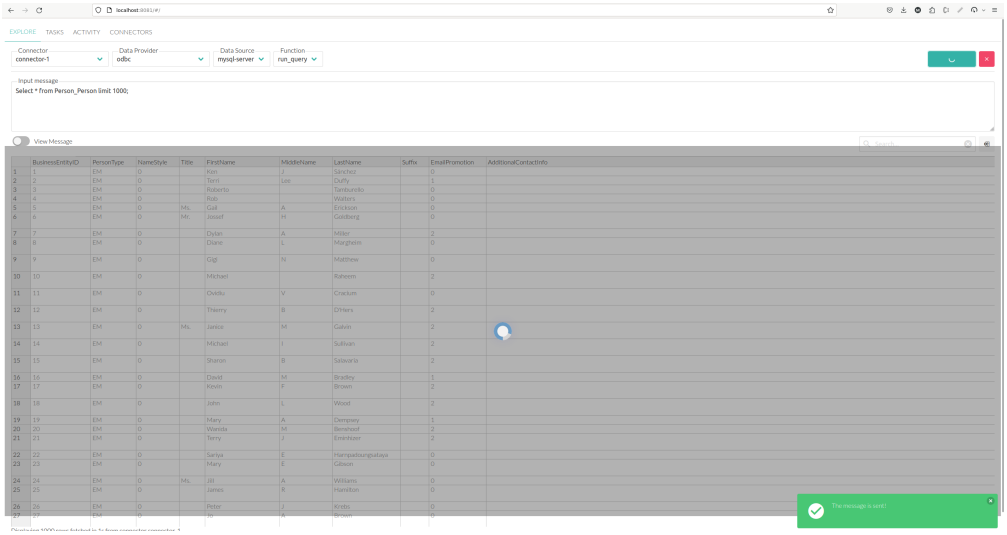


Figure 4.9: Query execution

4.2 Methods of data exporting

4.2.1 Export from the results table

When the result of the query is small enough that can be transmitted through the browser, by using the export button, we can download the content of query result. Otherwise, if the query is big enough that the size of the returned data is bigger than the size a browser can handle, it will be downloaded as a CSV file

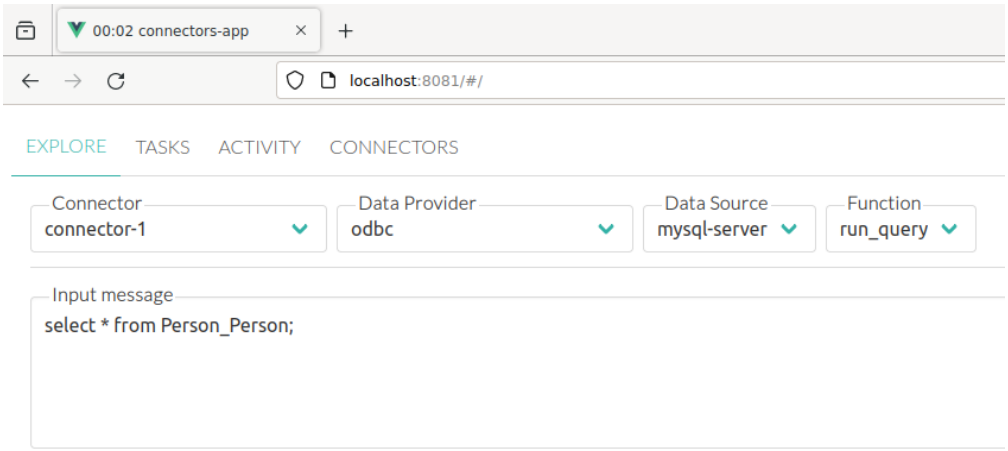


Figure 4.10: Query timer

automatically. For example, let's see these two queries again:

```
SELECT * FROM Person_Person limit 100;
SELECT * FROM Person_Person;
```

The first one selects the top 100 rows of the Person_Person table that can be seen inside the browser, see figure 4.11. On the other hand, the second query returns around 20K rows. Hence, it is downloaded automatically as a CSV file, see figure 4.12 and 4.13.

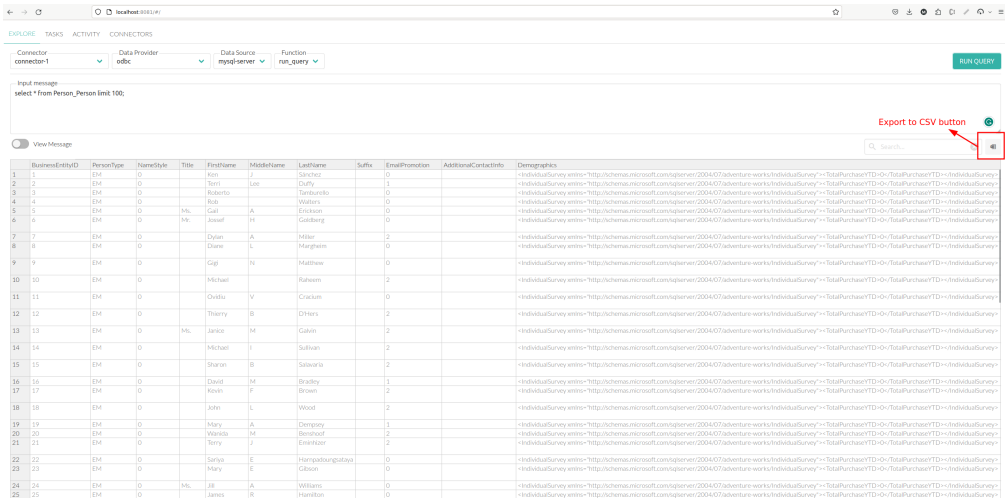


Figure 4.11: Small query result

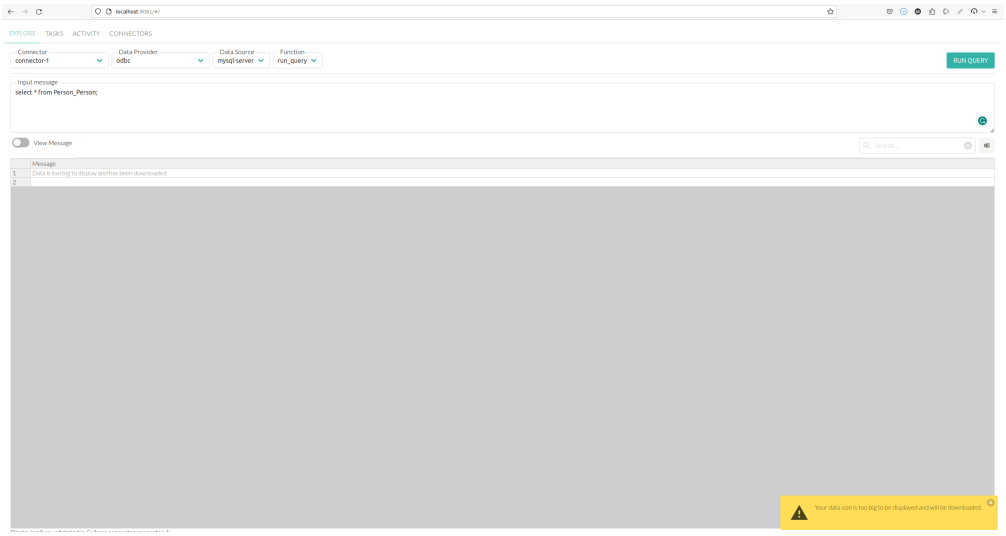


Figure 4.12: Big query result

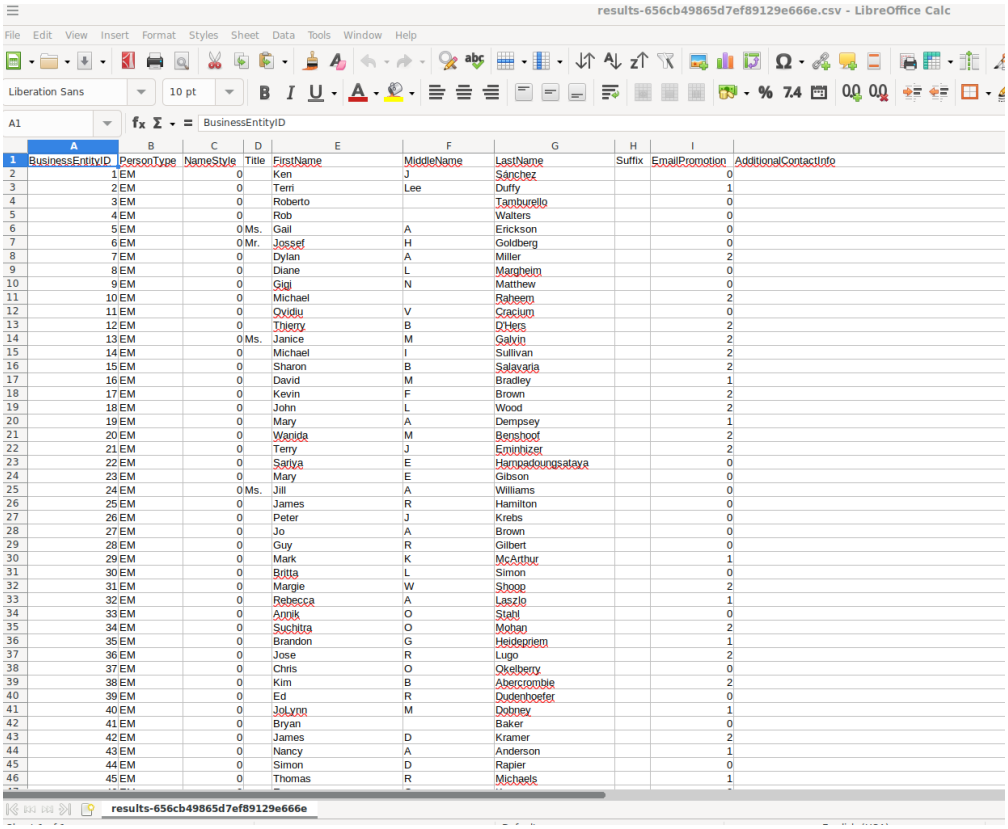
We can also see the time elapsed for the bigger query to understand how much time is needed to retrieve around 20K rows. Please see figure 4.14. As you can see the elapsed time for the whole operation is less than 10 seconds. So for 13,42MB file size we need around 10 seconds.

4.2.2 Export the data from the Activity tab

The result of the executed query can also be seen in the "ACTIVITY" -> "QUERIES" tab. If there are errors the messages are always there in case it is missed from the first tab. Also, there is information about the date and time of the execution and the content of the query itself. This information is useful when we want to see the logs of previous queries and monitor the data access. See figure 4.15. The data can be downloaded directly from this tab.

4.2.3 Tasks

Tasks are also useful for the continuous fetch of data. We can define tasks to automate the process of data extraction. The result of its execution is also accessible from the "ACTIVITY" -> "TASKS". Here, you will see the way to create tasks and download their execution result. For example, we will create a task to extract the top 1000 rows of the Person_Person table every 10 seconds. The



	A	B	C	D	E	F	G	H	I	
1	BusinessEntityID	PersonType	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	EmailPromotion	AdditionalContactInfo
2	1	EM	0		Ken	J	Sánchez		0	
3	2	EM	0		Terri	Lee	Duffy		1	
4	3	EM	0		Roberto		Tamburello		0	
5	4	EM	0		Rob		Walters		0	
6	5	EM	0	Ms.	Gail	A	Erickson		0	
7	6	EM	0	Mr.	Jossef	H	Goldberg		0	
8	7	EM	0		Dylan	A	Miller		2	
9	8	EM	0		Diane	L	Margheim		0	
10	9	EM	0		Gigi	N	Matthew		0	
11	10	EM	0		Michael		Raheem		2	
12	11	EM	0		Ovidiu	V	Cracium		0	
13	12	EM	0		Thierry	B	D'Hers		2	
14	13	EM	0	Ms.	Janice	M	Galvin		2	
15	14	EM	0		Michael	I	Sullivan		2	
16	15	EM	0		Sharon	B	Salavaria		2	
17	16	EM	0		David	M	Bradley		1	
18	17	EM	0		Kevin	F	Brown		2	
19	18	EM	0		John	L	Wood		2	
20	19	EM	0		Mary	A	Dempsey		1	
21	20	EM	0		Wanida	M	Benshoof		2	
22	21	EM	0		Terry	J	Eminhizer		2	
23	22	EM	0		Sariya	E	Harnpadoungsataya		0	
24	23	EM	0		Mary	E	Gibson		0	
25	24	EM	0	Ms.	Jill	A	Williams		0	
26	25	EM	0		James	R	Hamilton		0	
27	26	EM	0		Peter	J	Krebs		0	
28	27	EM	0		Jo	A	Brown		0	
29	28	EM	0		Guy	R	Gilbert		0	
30	29	EM	0		Mark	K	McArthur		1	
31	30	EM	0		Britta	L	Simon		0	
32	31	EM	0		Margie	W	Shoop		2	
33	32	EM	0		Rebecca	A	Laszlo		1	
34	33	EM	0		Annik	O	Stahl		0	
35	34	EM	0		Suchitra	O	Mohan		2	
36	35	EM	0		Brandon	G	Heidepriem		1	
37	36	EM	0		Jose	R	Lugo		2	
38	37	EM	0		Chris	O	Okelberry		0	
39	38	EM	0		Kim	B	Abercrombie		2	
40	39	EM	0		Ed	R	Dudenhoefer		0	
41	40	EM	0		JoLynn	M	Dobney		1	
42	41	EM	0		Bryan		Baker		0	
43	42	EM	0		James	D	Kramer		2	
44	43	EM	0		Nancy	A	Anderson		1	
45	44	EM	0		Simon	D	Rapier		0	
46	45	EM	0		Thomas	R	Michaels		1	

Figure 4.13: Big query result - CSV

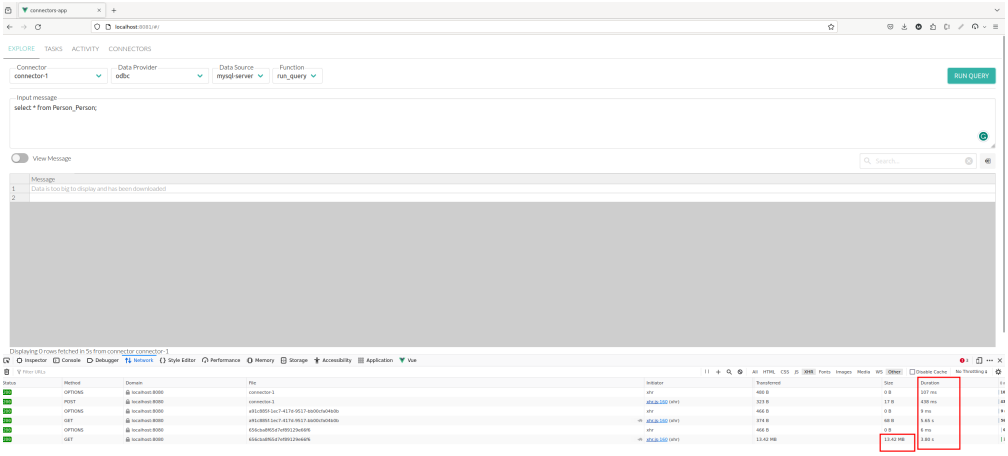


Figure 4.14: Big query elapsed time

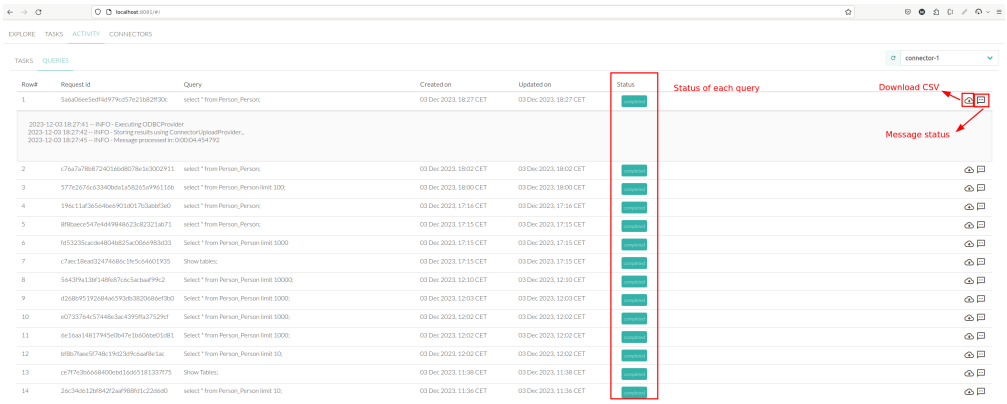


Figure 4.15: Activity queries

structure of the tasks is defined in YML format. In figure 4.16 we can see the connector of the task first needs to be defined, then we give our task a name and set its interval that is based on seconds. We can later set the "enable" status to true or false and finally in the "tasks" section we will give the details of the task, the data provider, the function, and the query are defined in this section. In the end, the output provider can be defined, it can be either the local file system or the FTP server. Then after saving the task, the connector can find it and execute it every 10 seconds and store the data in the "ACTIVITY"->"Tasks" tab. This is useful when a portion of data is updated frequently, so this is to make sure EVO always has the latest data. In figure 4.17 we can see the list of previously defined tasks and possibly manage them to set them enabled or disabled, change their data source or their output provider.

4.2.4 A usage of Salesforce RESTful API data source

Next, we are going to take a look at Salesforce APIs. EVO uses these APIs for their CustomerAI tool. As explained in the first chapter, Salesforce is a cloud-based software company that provides businesses with tools that help them find more prospects, close more deals, and provide a higher level of service to their customers.

Salesforce, Inc. is a famous American cloud-based software company that provides CRM services. Salesforce is a popular CRM tool for support, sales, and marketing teams worldwide.

The connector to use this data provider should be configured like below in figure 4.18. Then inside the application, we use this data provider to query the salesforce API. We can also use a SQL syntax to use their data. The following lines are some

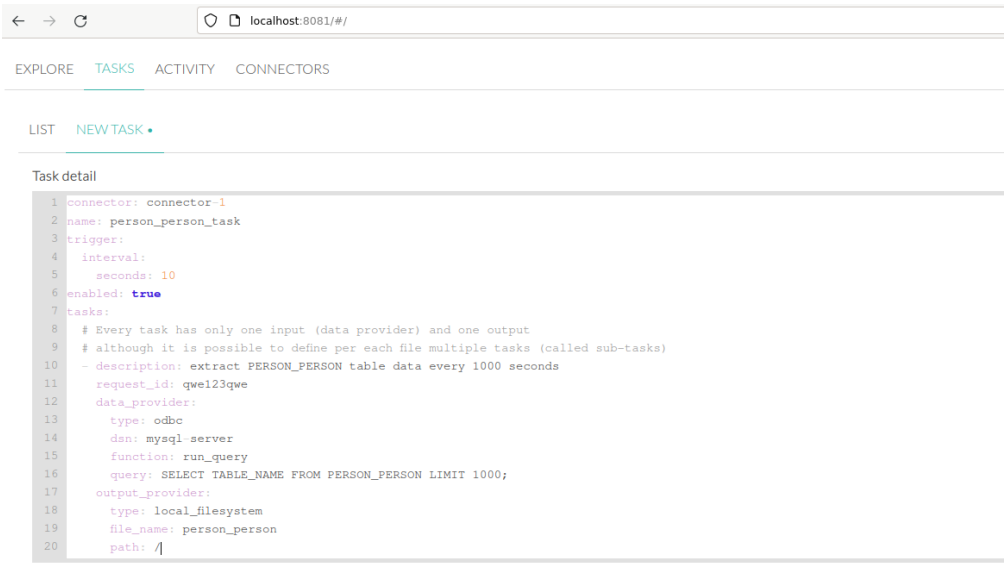


Figure 4.16: New task definition

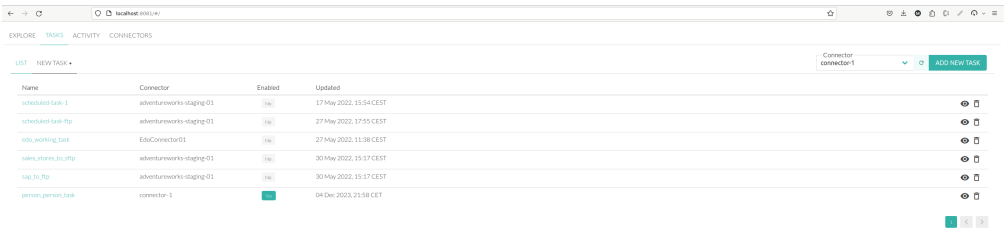


Figure 4.17: Tasks list

examples of how to write queries for the salesforce RESTful API [32].

```
"SELECT Id, Email, ParentAccount.Name FROM Contact WHERE LastName =
'Jones'"
```

```
"SELECT Id, Email FROM Contact WHERE LastName = 'Jones'"
```

```
"SELECT Id, Email FROM Contact WHERE Income > USD100"
```

```yaml
dataProviders:
  odbc:
    # https://docs.sqlalchemy.org/en/14/core/engines.html
    default: mysql-server
    dataSources:
      - name: mysql-server
        connectionString: DRIVER={MariaDB Unicode};SERVER=localhost;DATABASE=AdvenWorks;UID=sia;PWD=XXX
  odata:
    default: sap-sandbox
    dataSources:
      - name: sap-sandbox
        url: https://sandbox.api.sap.com/sap
        headers:
          - APIKey: xxx
  salesforce:
    default: evoeuropeltd-dev-ed
    dataSources:
      - name: evoeuropeltd-dev-ed
        username: salesforce@evopricing.com
        password: PASS
        security_token: TOKEN
```

Figure 4.18: Salesforce configuration

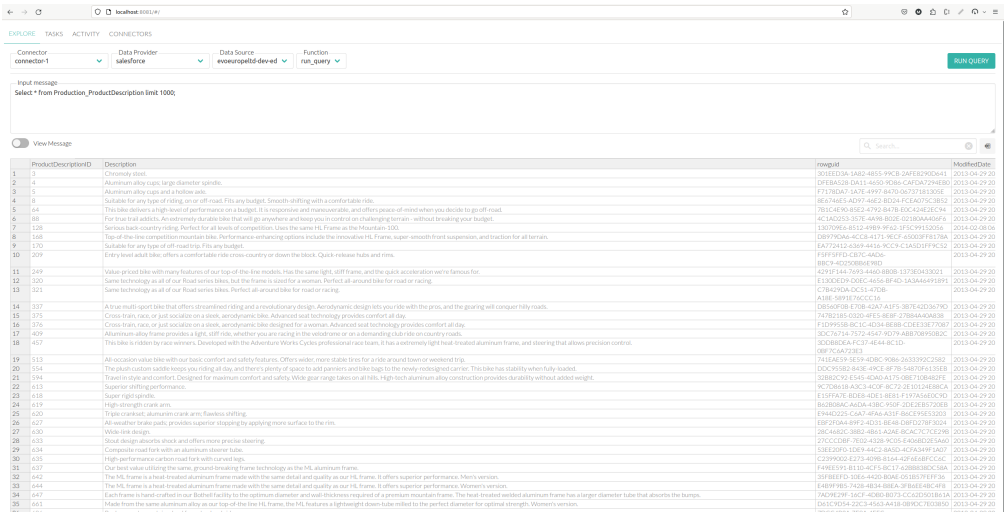


Figure 4.19: Salesforce API query

4.3 An intro to the current ETL process and the Connector application usage

4.3.1 Data extraction

Data extraction ETL: Based on the agreement between EVO and its clients, the raw data is usually uploaded to an FTP server daily or weekly. This data can have various formats including XLSX, CSV, TXT, etc. The data at first might be numerous rows in a raw format of CSV. See figure 4.20. Based on the agreement, the separator of the columns is defined, in here, the semicolon is considered as the

separator. Also, the number of columns MUST be agreed upon. For example, if the number of columns is agreed for the quantity of 10, the file cannot have only 9 columns. I emphasize that any tiny misalignment can lead the transformation process to failure. In the end, the line separator comes so that the scripts to process this file can use it, again as confirmed in the agreement, for data extraction. As you may have noticed, this data comes directly from EVO's clients so any change on them requires communication between EVO and its clients. Any misalignment of the row's content and expected value can again lead to failures during extraction. Usually, if there are any errors or misalignment, the back and forth between EVO and the client can take hours as it is not done automatically and the modification should be done by the client, hence, wasting a lot of time and resources in this process. So, by using the connector application, this wasted time can be omitted because EVO will have direct access to the data and can do the modification on them directly.

```
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
POS1;NR;1;;;01-Abiti;0102- VP;U-Uomo;ABC-ABC;AI2003;U1-.44.46.48.50.52.54.56.58.60.62.64.66;N;;LF00006-COSTANZO
```

Figure 4.20: Raw data

4.3.2 Data transformation

Data transformation ETL: Once the client's file is prepared and uploaded, it will become accessible on the FTP server. Then, EVO's ETL pre-processing procedure will be started by using the provided file in the path that is expected by the pre-processing. This critical step includes a set of data validation processes to ensure the integrity and accuracy of the information. It also verifies that the data columns adhere to the specified format and confirms that each column's data type aligns with the agreed requirements set for the transformation procedure. This approach ensures that the data is ready for the following processing stages. Again,

here, any failure because of corrupted data can lead to failures and re-running the process from the beginning. The connector application can also do this step before data extraction. Let's see an example of transforming data in the extraction phase in the connector application.

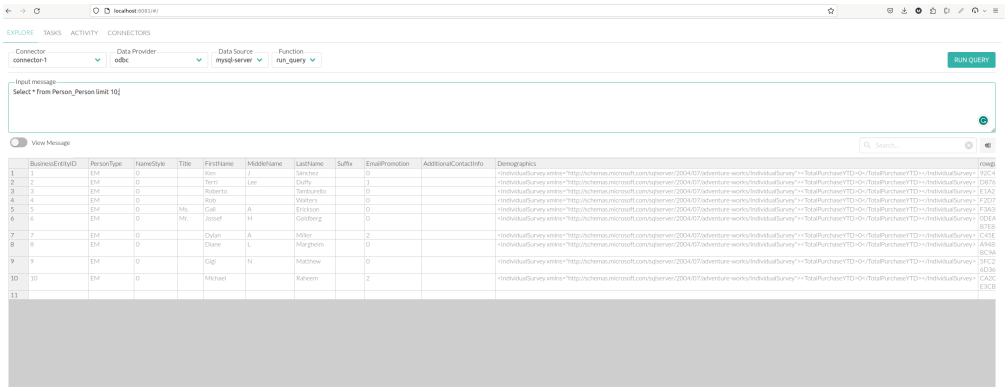


Figure 4.21: Normal select example

Now let's consider right here EVO needs to receive the data in the following format:

```
Title; Fullname; Person;
```

Simply, the transformation of the data can be done right inside the connector. See figure 4.22.

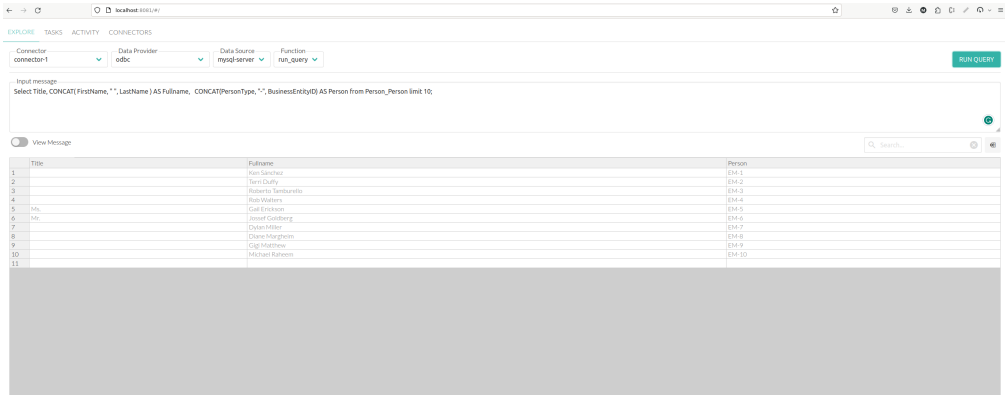


Figure 4.22: Transformed data extraction

If in real scenarios we have this kind of transformation on the raw data, the connector application can help reduce the pre-process steps for data transformation. This was just an example, in real cases, this step can take from thirty minutes

up to two or three hours, depending on the amount of rows, and number of transformations, and the database size. So, the more prepared the extracted data is, the faster the pre-processing is. Also, in some scenarios, we can completely omit this step and save a lot of time. This will also reduce the complexity of the EVO's ETL process. Below you can see an example of one of the existing pipelines of data transformation of EVO on the data. Figures 5.1, 4.24. By using the connector application wisely, we can omit this complex flow of pre-processing data. The maintenance of these pipelines requires skilled data engineers to continuously monitor and potentially fix the possible errors this is not an easy task and it is very time-consuming and requires a lot of resources.

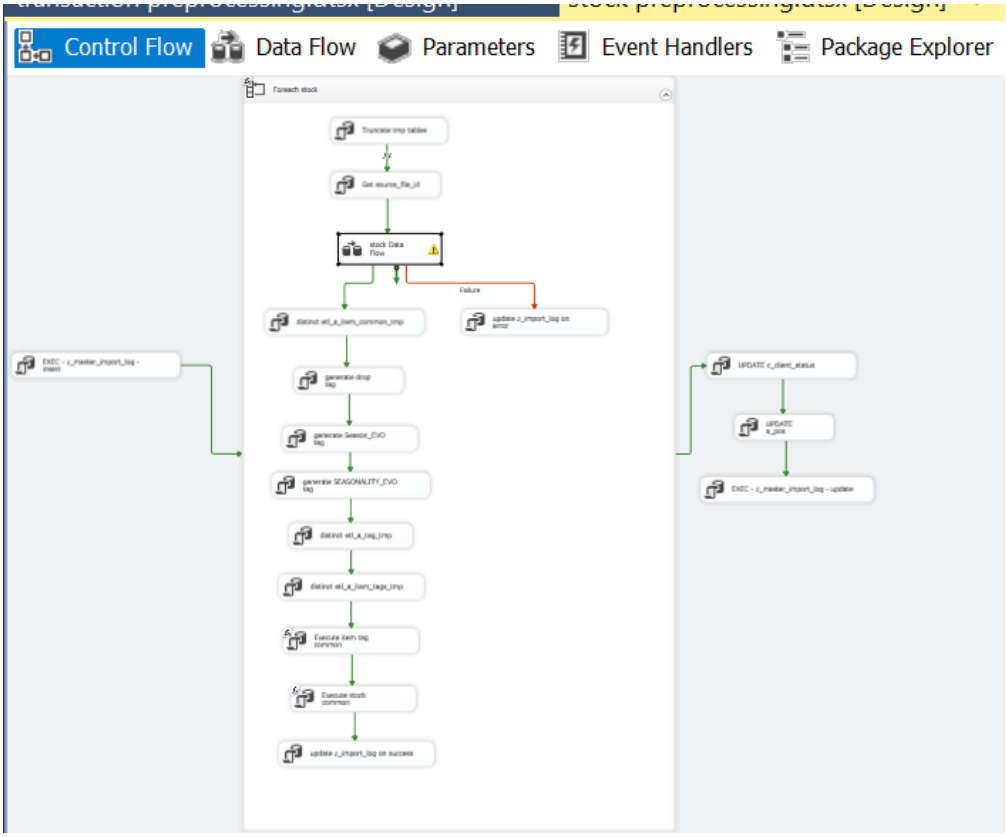


Figure 4.23: Transform pipeline 1

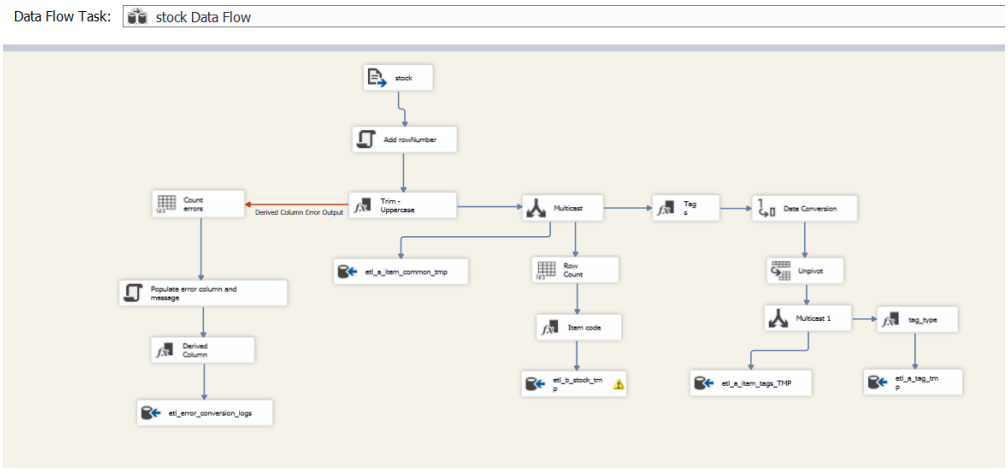


Figure 4.24: Transform pipeline 2

4.3.3 Data load

Data load ETL: The load process is not much changed. Here, we will just show the pipeline related to the main process of EVO's clients just to understand the complexity of the whole flow. See figure 4.25 The load process is not much changed. Here, we will just show the pipeline related to the main process of EVO's clients just to understand the complexity of the whole flow. See Figure 4.24. To recap the whole scenario and the way it has become more efficient, the process starts from receiving a file from the client FTP. This file usually contains rows and columns based on the agreement between EVO and its client. The file then goes to the SSIS server to go through the pre-processing step. At this step the transformation is done, the step that can also be done on the ETL connector application to reduce the number of operations and as a result omitting validation and data check step and this means saving the server resources. SSIS Packages that have DTSX extension will do the pre-processing and load operation. If we have a lookup operation, it can consume a lot of resources so this operation again can be done in the ETL connector application. Indeed that lookup operation can be done via a script but in that case, we should figure out a way for all the other operations as well to make them more efficient. So the server crashes because of this kind of operation when there are not many resources available. The potential usage of the ETL connector application can be also doing the insert operation that is done inside the pre-processing step. Meaning, the extracted and transformed data can be saved into a CSV file, then another task can take these CSV files and save the rows into the temp tables for the main ETL process of EVO making the most of the process automatic.

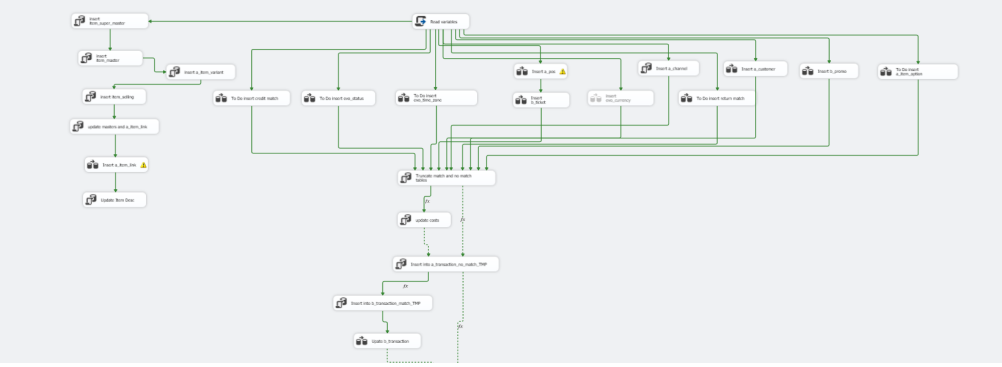


Figure 4.25: Main ETL pipeline

Chapter 5

Conclusions and potential future development

The ETL Connector application includes two microservices, one will run on EVO's client's side and the other is accessible through the WEB. As discussed in the previous chapter, many potential data sources can be used inside this application. The ones that were discussed so far were a normal MySQL database, a SAP data provider, or even an online RESTful API from SalesForce. Considering the application works perfectly with the SQL query syntax we may add as many data sources as we want. So, it is not important if it is an Oracle database or a MongoDB database. Even though, there are slight differences between some data sources syntax the ETL Connector can execute and do the data extraction by a person who is familiar with these languages. We talked about the tasks and scheduling that can help us achieve full automation of the ETL pre-process step which for us means the data transformation and preparing it for the EVO's main ETL process. Below you can see how ETL Connector has helped us to reduce the consumed resources and time for this process. The numbers are not completely accurate since they depend on many factors, but it is advised by the skilled data engineers who are responsible for EVO's ETL team.

In closing up our investigations about of the ETL connector application, it is important to note that through ETL it has been possible to improve the effectiveness and ease with which ETL runs are carried out. The ETL connector application has helped change how EVO and its customers handle and make ETL operations a lot. it has reduced EVO's server resource consumption. The significant decrease in servers' resource consumption is one of its outstanding accomplishments. The ETL process before the use of an ETL connector application used a lot of resources, causing server overload and wasted computing energy. This burden has been significantly reduced

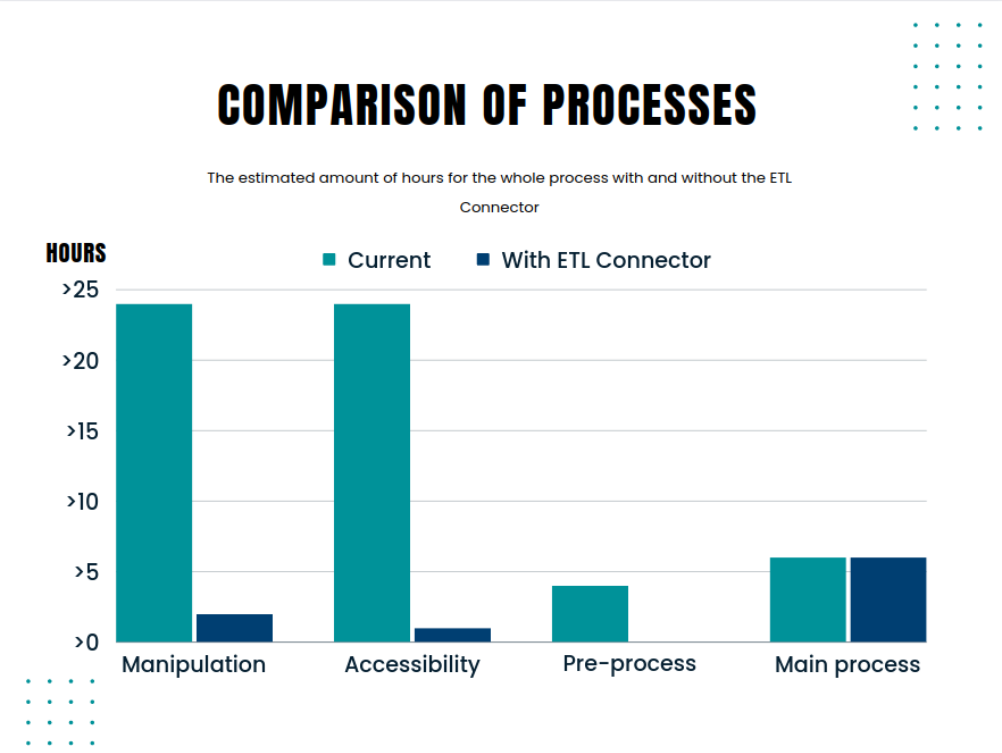


Figure 5.1: Time (hour) used for EVO pre-process

by using the unique design of the ETL connector application which is very efficient. The application maximizes on data extraction and transformation phases so that server resource use is economical. Optimization has various advantages including low operational cost, as well prolonging life span, and improving robustness of the server estate. The Enhanced Interactions between EVO and its customers is also another important step towards better performance is improved communication with EVO's customers. Effectively, the ETL connector application has reduced the hitherto conventional ETL process's back-and-forth communication. The application has made this process simple by automating most of the crucial steps of the ETL pipeline. This has created more transparent and speedy access to crucial data. The procedure has become faster and less prone to error than before because of the decline in manual intervention and high-level automation that this implementation has brought about.

It has enhanced control over ETL Processes; it has made EVO and customers have more authority in ETL operations. Besides, a user-friendly interface combined with strong backbone of the application permits real-time monitoring and adjustment. These control measures are important, especially in competitive business markets where agility in responding to evolving data needs is key. Enhanced control

is also provided to handle errors effectively, map data efficiently, and integrate with different data sets or sources without fuss.

Finally, ETL connector application has taken a big step in ETL process. Efficiency and effectivity have been raised to a new level as it impacts server resources consumption reduction, simpler interfaces, and greater control of ETL processes. Moving forward, future enhancements in such application will be pivotal to more adaptable, affordable, and reliable data management practices. This thesis highlights past achievement and offers a springboard for further innovation on the ETL landscape.

Bibliography

[1] Artur Wojciechowski. «E-ETL: Framework for Managing Evolving Etl Processes». In: *Proceedings of the 4th Workshop on Workshop for Ph.D. Students in Information & Knowledge Management*. PIKM '11. Glasgow, Scotland, UK: Association for Computing Machinery, 2011, pp. 59–66. ISBN: 9781450309530. DOI: 10.1145/2065003.2065016. URL: https://doi.org/10.1145/2065003.2065016 (cit. on pp. 1, 11).

[2] Zineb El Akkaoui, Esteban Zimànyi, Jose-Norberto Mazón, and Juan Trujillo. «A Model-Driven Framework for ETL Process Development». In: *Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP*. DOLAP '11. Glasgow, Scotland, UK: Association for Computing Machinery, 2011, pp. 45–52. ISBN: 9781450309639. DOI: 10.1145/2064676.2064685. URL: https://doi.org/10.1145/2064676.2064685 (cit. on pp. 1, 6).

[3] Microsoft. *Install and Configure AdventureWorks Sample Databases*. https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms. Accessed: 2023-11-25. 2022 (cit. on p. 3).

[4] N. Li, C. Pedrinaci, M. Maleshkova, J. Kopecky, and J. Domingue. «Omnivoke: a framework for automating the invocation of web apis». In: (2011). DOI: 10.1109/icsc.2011.72 (cit. on p. 3).

[5] R. Rogalski. «Comparative analysis of data reading performance from the salesforce platform using graphql, rest and soap interfaces». In: *Journal of Computer Sciences Institute* 27 (2023), pp. 171–177. DOI: 10.35784/jcsi.3601 (cit. on p. 4).

[6] O. Melykh and A. Korbut. «Entertainment media in the context of hybrid war in the post-soviet countries: the case of ukraine». In: *Economic Annals-i* 182 (3-4 2020), pp. 25–33. DOI: 10.21003/ea.v182-03 (cit. on p. 4).

[7] V. Theodorou, A. Abelló, M. Thiele, and W. Lehner. «A framework for user-centered declarative etl». In: (2014). DOI: 10.1145/2666158.2666178 (cit. on p. 6).

[8] O. Belo, A. Cuzzocrea, and B. Oliveira. «Modeling and supporting etl processes via a pattern-oriented, task-reusable framework». In: (2014). DOI: 10.1109/ictai.2014.145 (cit. on p. 6).

[9] V. Theodorou, A. Abelló, M. Thiele, and W. Lehner. «Frequent patterns in etl workflows: an empirical approach». In: *Data Knowledge Engineering* 112 (2017), pp. 1–16. DOI: 10.1016/j.datak.2017.08.004 (cit. on p. 6).

[10] R. Talib, M. Hanif, F. Fatima, and S. Ayesha. «A multi-agent framework for data extraction, transformation and loading in data warehouse». In: *International Journal of Advanced Computer Science and Applications* 7 (11 2016). DOI: 10.14569/ijacsa.2016.071146 (cit. on p. 6).

[11] J. Li, B. Kuang, and L. Jin-gang. «Script-based automation etl tool». In: (2016). DOI: 10.2991/meici-16.2016.201 (cit. on p. 7).

[12] S. Luján-Mora, P. Vassiliadis, and J. Trujillo. «Data mapping diagrams for data warehouse design with uml». In: (2004), pp. 191–204. DOI: 10.1007/978-3-540-30464-7_16 (cit. on p. 7).

[13] S. Dakrory, T. Mahmoud, and A. Ali. «Automated etl testing on the data quality of a data warehouse». In: *International Journal of Computer Applications* 131 (16 2015), pp. 9–16. DOI: 10.5120/ijca2015907590 (cit. on p. 7).

[14] N. Vijayendra and Meiliu Lu. «A web-based ETL tool for data integration process». In: *2013 6th International Conference on Human System Interactions (HSI)*. IEEE. 2013, pp. 434–438. DOI: 10.1109/HSI.2013.6577861 (cit. on p. 7).

[15] F. N. Savitri and H. Laksmiwati. «Study of Localized Data Cleansing Process for ETL Performance Improvement in Independent Datamart». In: *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*. IEEE. 2011, pp. 1–6. DOI: 10.1109/ICEEI.2011.6021806 (cit. on p. 7).

[16] Vasileios Theodorou, Alberto Abelló, and Wolfgang Lehner. «Quality measures for ETL processes». In: *Data Warehousing and Knowledge Discovery: 16th International Conference, DaWaK 2014, Munich, Germany, September 2-4, 2014. Proceedings 16*. Springer. 2014, pp. 9–22 (cit. on p. 8).

[17] Vangipuram Radhakrishna, Vangipuram SravanKiran, and K Ravikiran. «Automating ETL process with scripting technology». In: *2012 Nirma University International Conference on Engineering (NUiCONE)*. IEEE. 2012, pp. 1–4 (cit. on p. 9).

[18] *OData*. https://www.odata.org/. Accessed: 2023-11-08 (cit. on p. 21).

[19] *OData*. https://www.openfaas.com//. Accessed: 2023-11-09 (cit. on p. 22).

[20] *Flask*. https://pythonbasics.org/what-is-flask-python//. Accessed: 2023-11-09 (cit. on p. 23).

[21] *MongoDB*. https://www.mongodb.com/. Accessed: 2023-11-08 (cit. on p. 23).

[22] David Dossot. *RabbitMQ Essentials*. Birmingham: Packt Publishing, 2014 (cit. on p. 24).

[23] P. Kavanagh. «The open source definition». In: (2004), pp. 321–322. DOI: 10.1016/b978-155558320-0/50016-7 (cit. on p. 24).

[24] «The cathedral and the bazaar». In: *Computers Mathematics With Applications* 39 (3-4 2000), p. 263. DOI: 10.1016/s0898-1221(00)90039-7 (cit. on p. 24).

[25] A. Gramfort, T. Papadopoulo, E. Olivi, and M. Clerc. «Openmeeg: opensource software for quasistatic bioelectromagnetics». In: *Biomedical Engineering Online* 9 (1 2010), p. 45. DOI: 10.1186/1475-925x-9-45 (cit. on p. 24).

[26] A. Buitenhuis and J. Pearce. «Open-source development of solar photovoltaic technology». In: *Energy for Sustainable Development* 16 (3 2012), pp. 379–388. DOI: 10.1016/j.esd.2012.06.006 (cit. on p. 24).

[27] P. Ågerfalk and B. Fitzgerald. «Outsourcing to an unknown workforce: exploring opensurcing as a global sourcing strategy». In: *Mis Quarterly* 32 (2 2008), p. 385. DOI: 10.2307/25148845 (cit. on p. 24).

[28] T. Omopupa, A. Adedeji, A. Kehinde, A. Abdulsalam, and H. Abubakar. «Comparative study of koha usage in bowen university and university of ilorin libraries». In: *Üniversite Araştırmaları Dergisi* 3 (3 2020), pp. 98–106. DOI: 10.32329/uad.741713 (cit. on p. 24).

[29] P. Wiecha, A. Lecestre, N. Mallet, and G. Larrieu. «Pushing the limits of optical information storage using deep learning». In: *Nature Physics* 14 (3 2019), pp. 237–244. DOI: 10.1038/s41565-018-0346-1 (cit. on p. 25).

[30] D. Neupane and J. Seok. «Bearing fault detection and diagnosis using case western reserve university dataset with deep learning approaches: a review». In: *Ieee Access* 8 (2020), pp. 93155–93178. DOI: 10.1109/access.2020.2990528 (cit. on p. 25).

[31] W. Kitlasten, C. Moore, and B. Hemmings. «Model structure and ensemble size: implications for predictions of groundwater age». In: *Frontiers in Earth Science* 10 (2022). DOI: 10.3389/feart.2022.972305 (cit. on p. 25).

[32] *Simple Salesforce*. https://pypi.org/project/simple-salesforce//. Accessed: 2023-12-05 (cit. on p. 33).