A systematic review of tools to create ETL pipelines

MASTERS THESIS

Mujeeb Ahmed

Submitted on 25 April 2025

Friedrich-Alexander-Universität Erlangen-Nürnberg
Faculty of Engineering, Department Computer Science
Professorship for Open Source Software

Supervisor:
Philip Heltweg, M. Sc.
Prof. Dr. Dirk Riehle, M.B.A.

Friedrich-Alexander-Universität
Faculty of Engineering

Declaration of Originality

I confirm that the submitted thesis is original work and was written by me without further assistance. Appropriate credit has been given where reference has been made to the work of others. The thesis was not examined before, nor has it been published. The submitted electronic version of the thesis matches the printed version.

Erlangen, 25 April 2025

License

Erlangen, 25 April 2025

Abstract

ETL data pipelining tools are used to create pipelines that extract data from a source, transform it, and load it into a destination. A variety of approaches exist for building these pipelines, ranging from visual modeling tools tailored to non-technical users, to general-purpose programming solutions preferred by developers, and domain-specific languages that aim to bridge the gap between the two. Selecting the right approach depends on multiple factors. To develop a deeper understanding, it is important to examine the design considerations of these tools, along with the assessment approaches and benchmarks used to evaluate their effectiveness. Beyond aiding tool users, understanding these factors is also valuable for developers aiming to create new ETL tools, such as those working on Jayvee. This thesis presents a Systematic Literature Review (SLR), conducted in accordance with the guidelines of Kitchenham and Charters, 2004, to examine ETL pipeline creation tools based on their type, target audience, evaluation methods, and monetization models. Additionally, we analyze the trade-offs associated with different design choices. The results show that developers have a wide range of options depending on their goals, with each involving trade-offs that highlight distinct strengths and weaknesses. We also identify and discuss the intended users of these systems, along with evaluation methods and key metrics explored in the literature. However, monetization models remain largely unexplored.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Problem Statement . . . 1
- 1.3 Scope of the Study . . . 2

2 Related Work **3**

3 Research Design **5**
- 3.1 Planning the Review . . . 6
  - 3.1.1 Research Questions . . . 6
  - 3.1.2 Search Strategy . . . 6
  - 3.1.3 Inclusion and Exclusion Criteria . . . 7
- 3.2 Conducting the Review . . . 8
  - 3.2.1 Data Extraction and Synthesis . . . 8
- 3.3 Reporting the Review . . . 9

4 Research Results **10**
- 4.1 Search Results . . . 10
- 4.2 Tools Discussed in the Literature . . . 14
- 4.3 Categorization of the Tool Type . . . 15
  - 4.3.1 Metadata-Driven SQL-Based tools . . . 15
  - 4.3.2 General-Purpose Programming Languages . . . 16
  - 4.3.3 Domain-Specific Languages . . . 17
  - 4.3.4 Open-Source Orchestration . . . 18
  - 4.3.5 Cloud-Based Tools . . . 19
  - 4.3.6 Semantic Web-Based Tools . . . 20
- 4.4 Target Audience . . . 21
- 4.5 Evaluation Methods Used to Evaluate Tools . . . 23
  - 4.5.1 System-Centric Evaluations . . . 23
  - 4.5.2 Demonstration-Based Evaluations . . . 24
  - 4.5.3 Informal Interviews . . . 25
  - 4.5.4 User-Centered Evaluations . . . 25

4.5.5 Comparative Case Study Evaluation . . . 25
4.5.6 Telemetry-Based Evaluations . . . 26
4.6 Metrics Used in Evaluation Methods . . . 27
4.6.1 Development Effort in Pipeline Creation . . . 27
4.6.2 Portability in Tool's Functionality . . . 28
4.6.3 Extensibility of Tool . . . 28
4.6.4 Functional Suitability . . . 28
4.6.5 Productivity Metrics . . . 28
4.6.6 Validation and Accuracy Metrics . . . 28
4.6.7 Usability and Interface Metrics . . . 29
4.7 Trade offs Exist in the Development of the Tools . . . 30
4.7.1 Metadata-Driven Design vs Hardcoded Logic . . . 30
4.7.2 Code-Based vs GUI-Based Tools . . . 30
4.7.3 Declarative vs. Procedural Approach . . . 31
4.7.4 Development simplicity vs Optimization . . . 32
4.7.5 Ontology-Based Integration vs Simpler Schema Models . . . 32
4.7.6 Functional simplicity vs Versatility . . . 33
4.7.7 In-house build vs buy/open source . . . 33
4.8 Monetization Models for the Tools . . . 36

5 Discussion **37**
5.1 Tool Categorization and Target Audience . . . 37
5.2 Evaluation Methods and Metrics . . . 37
5.3 Trade-Offs in ETL Tool Creation . . . 38
5.4 Monetization Model . . . 38

6 Limitation **40**
6.1 Credibility . . . 40
6.2 Transferability . . . 40
6.3 Dependability . . . 40
6.4 Conformability . . . 41

7 Conclusion **42**

Appendices **44**
A Excel Dataset: Initial Paper Pool and Review Process . . . 45
B Reproducible Package Overview . . . 46

References **49**

List of Figures

3.1 Flowchart showing the key steps in Barbara Kitchenham's systematic literature review process. 5

4.1 Screening and Selection Process for Scientific Papers 11
4.2 Distribution of Included Papers by Publication Year 12
4.3 Total count of each element after analyzing every article. 12
4.4 Distribution of Tool Categories. 15
4.5 Distribution of target audience of the tools. 21
4.6 Frequency of evaluation methods used in ETL tool studies as observed in the reviewed literature. 23

List of Tables

4.1 List of ETL Tools and References 14
4.2 Overview of Evaluation Methods and Addressed Metrics 27
4.3 Trade-Offs and Impact of the Choices 35

Acronyms

DSL Domain Specific Language

DW Data Warehouse

EMF Eclipse Modeling Framework

ETL Extract, Transform, Load

ETLCL ETL Control Language

GPL General Purpose Programming Languages

GUI Graphical User Interface

OWL Web Ontology Language

RDF Resource Description Framework

SLR Systematic Literature Review

SPARQL SPARQL Protocol and RDF Query Language

SQL Structured Query Language

ERP Enterprise Resource Planning

3PL Third-Party Logistics

XML eXtensible Markup Language

CSV Comma-Separated Values

FTP File Transfer Protocol

1 Introduction

1.1 Background

Data pipelines are at the heart of data engineering. They manage the entire process, from collecting raw data from various sources to cleaning, organizing them into a structured format, and transferring it to the appropriate systems. This process, called ETL, short for Extract, Transform, Load, is a key part of how data engineering works. It serves as the foundation for data analysis and supports numerous applications operating behind the scenes.

These pipelines automate the process of extracting raw data from scattered sources, transforming them into a structured format, and loading them into a target location, such as data warehouses, where it can be efficiently queried and analyzed. Without ETL pipelines, the information remains in a state of disorganization and in an unworkable form. ETL pipelines make information accessible, reliable, and ready for reporting, predictive analysis, and performance analysis.

ETL data pipelines can be created in various ways, depending on the user's needs. Visual tools strengthen accessibility for non-technical users, whereas code-based solutions provide greater flexibility for individuals with technical expertise. There are also domain-specific languages that combine the best of both worlds. These options allow users to select the approach that best aligns with their needs, whether they prioritize ease of use, scalability, or greater control.

1.2 Problem Statement

One of the biggest challenges in building an ETL pipeline is finding tools that work well for both technical and non-technical users. If a tool is too simple, it might not handle the complexity of real-world data processes. But if it is too technical, it can become a barrier for team members without a coding background. When tools lean too far in either direction, collaboration suffers, and the whole process becomes less efficient. That's why choosing a tool that strikes a good balance between ease of use and technical flexibility is so important.

There are many tools out there that can be used to create ETL data pipelines, but surprisingly little work has been done to group them in meaningful ways, such as by type, target audience, or product model. Because of this, it is hard to understand the trade-offs between different kinds of tool, especially for users trying to pick the one that fits them best. This thesis aims to fill this gap. By categorizing tools and examining how they are used, along with the compromises they involve, it provides a clearer path for users to make more informed choices based on their specific needs. Another key motivation behind this work is to support developers who are building ETL tools, such as Jayvee, by offering insights into how different design choices affect usability, adoption, and trade-offs. The goal is not only to guide users in choosing tools, but also to help tool creators understand the needs of their audience and design more balanced, effective solutions.

1.3 Scope of the Study

This study will focus on the tools and approaches used to create ETL pipelines, specifically examining the tools described in the scientific literature.

2 Related Work

A number of earlier studies provided a solid foundation for understanding and analyzing ETL tools, particularly by identifying key quality attributes such as performance, scalability, reliability, and data quality. These contributions were instrumental in shaping the criteria by which ETL tools were commonly evaluated. Although these works highlighted important evaluation criteria, they paid comparatively less attention to the specific methodologies used to assess tools in practice.

Nwokeji and Matovu (2021) explored how ETL evolved as a method for integrating data from different sources into a central warehouse. The authors analyzed how ETL tools were implemented, which factors were important when selecting an approach, how ETL research showed up across different fields and parts of the world, and what common challenges developers faced. Based on their findings, the authors shared takeaways that could be useful to both researchers exploring ETL and practitioners applying it in real-world settings.

Mukherjee and Kar (2017) also took a closer look at different well-known ETL tools, such as Informatica, Datastage, Ab Initio, Oracle Data Integrator and SSIS, to understand how they performed on real-world data storage tasks. The authors compared these tools based on key features including performance optimization, data lineage, real-time processing, cost, and language support. In addition to the technical comparison, the paper integrated insights from the data science industry to explain how these tools were valued and applied in practice. It also demonstrated how traditional data warehousing was evolving, particularly with the rise of big data, cloud platforms, and the growing need to handle both structured and unstructured data.

Patel and Patel (2020) discussed the role of ETL as a important part of data warehousing. The authors grouped various categories of ETL tools, such as code-based, cloud-based, real-time, and batch processing, and discussed the importance of selecting right tools based on their pros and cons. The study focused on both reviewing existing ETL tools and helping organizations identify the most important functions to consider when selecting a tool suited to their needs.

In our study, we built upon this foundation by offering a more focused examination of the evaluation methods applied in ETL tool research. We identified and discussed various approaches, such as system-centric evaluations, user studies, and comparative case studies, and connected them with performance indicators like usability, validation accuracy, and productivity. We also categorized the tools used to create ETL data pipelines and identified their target audiences. By doing so, we sought to complement the existing body of work with a structured perspective on how evaluations were conducted in practice.

In addition, we addressed trade-offs present in ETL pipeline development, contributing to a more nuanced understanding of tool design, which were not often discussed in earlier studies. While many studies centered on comparisons and theoretical analyses, our work highlighted practical aspects, including the balance between simplicity and flexibility, along with the kinds of trade-offs that often came up when designing ETL tools. With this contribution, we support both researchers and practitioners in making sense of the often complex process of developing and evaluating tools.

3 Research Design

The research design we drafted for this study uses a systematic review of the literature (SLR) following Kitchenham and Charters (2004) guidelines to proceed with systematic review processes in collecting and analyzing relevant scientific studies in the development of ETL data pipeline tools. Figure 3.1 shows three basic pillars of Kitchenham and Charters (2004) guidelines with steps involved in each step.

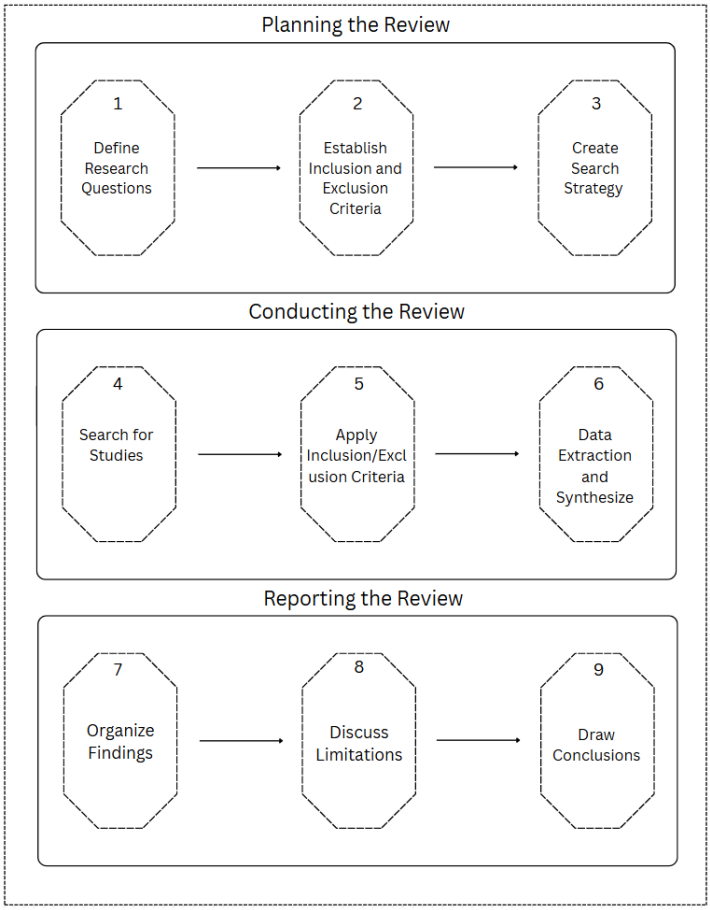

Figure 3.1: Flowchart showing the key steps in Barbara Kitchenham's systematic literature review process.

3.1 Planning the Review

Planning the review initiated with clearly defining the research questions and then refining them to make them more explicit and focused. Once the research questions were defined, the next step was to set up criteria for what to include and exclude from the review. Inclusion criteria referred to the boundaries set to select papers to be included in the final dataset from the initial pool of articles. Similarly, exclusion criteria defined the boundaries to exclude papers from the initial dataset. After establishing these criteria, the search strategy was developed. In the search strategy, we created the search query and selected the data sources from which we retrieved articles.

3.1.1 Research Questions

This research aims to fill the gaps by addressing the following research questions.

1. What tools have been described in the scientific literature to create ETL pipelines?
2. How can these tools be classified according to their type, target audience, and monetization/business model?
3. How have these tools been evaluated?
4. What tradeoffs exist in the development of the ETL pipeline creation tool and how do these tradeoffs influence the strengths and weaknesses of the tools?

3.1.2 Search Strategy

We constructed a search strategy with a range of sources for ETL data pipeline tools with collected literature through Google Scholar with a custom-made search query. Strict inclusion and exclusion rules were applied to ensure that the selected articles were relevant, peer-reviewed, written in English, and focused on actual ETL software tools. After this careful filtering process, we condensed the initial 308 articles to 27 relevant ones and analyzed them to study ETL data pipeline tools, their features, strengths, and trade-offs.

We designed a search scheme to span a range of sources interested in the development of ETL data pipeline tools. We used Google Scholar as our source because of its broad coverage of computer science databases and its ability to retrieve relevant literature effectively. The following search query was used:

```
allintitle:("ETL" OR "data pipeline" OR "extract transform
load") AND ("tool" OR "software" OR "solution" OR "platform"
```

```
OR "application" OR "framework" OR "programming language" OR
"domain specific language" OR "DSL" OR "domain-specific language")
```

This process involved choosing synonyms for significant terms in the field of ETL data pipeline in an effort to span a variety of terms such as 'tool', 'software', 'solution', 'platform', 'application', 'framework', 'programming language', 'domain specific language', and 'DSL' as synonyms for tools and software when dealing with the data.

To ensure an effective search, terms 'tool,' 'software,' 'solution,' 'platform,' and 'framework' were combined with use of the OR operator in an attempt to include a variety of terms, and then combined with AND with principal ETL data pipeline terms 'ETL,' 'data pipeline,' and 'extract transform load.' In a deliberate and purposeful manner, a variety of studies pertinent and prevalent with regard to ETL data pipeline tool development was captured to provide a sound basis for reviewing the literature in a systematic and purposeful manner.

3.1.3 Inclusion and Exclusion Criteria

To ensure the relevance of the articles, the following inclusion and exclusion criteria were applied:

Inclusion Criteria:

1. Papers describing actual software tools for creating ETL or data modeling pipelines.

Exclusion Criteria:

1. Exclude papers that are case studies with data or focus on optimization approaches or methodologies rather than presenting a fully developed ETL pipeline tool/software.
2. Non-peer-reviewed sources.
3. Papers not written in English.
4. Papers that could not be accessed in full due to paywalls or restrictions.

3.2 Conducting the Review

Conducting the review referred to the execution of the search strategy created in the previous step. We searched for data using the query that we developed, then applied the inclusion and exclusion criteria to the dataset obtained from the search. After we had our final dataset, the next step was to begin data extraction from the literature. To facilitate this process, a data extraction form was created following Kitchenham and Charters (2004) guidelines, guaranteeing that all relevant fields were captured.

3.2.1 Data Extraction and Synthesis

We designed our data extraction for this study to classify ETL data pipeline tools according to types, target groups, evaluation approaches, performance factors, tool development trade-offs, and monetization strategies mentioned in the authors' works. To make it feasible, we developed an in-depth data extraction form in compliance with Kitchenham's (2004) guidance for a systematic review. The form was designed in a manner that captured all pertinent factors in relation to our study objectives and questions, in a systemic manner.

To facilitate and promote extraction of data, we have utilized software MAXQDA in developing traceability between source and extracted data in analyzed articles. We structured the fields in the extraction form systematically to ensure consistency and traceability throughout the extraction process. As the extraction process continued, we expanded the form into a more complex structure to capture more nuanced findings when deemed necessary.

For example, when it comes to ETL tool types, we extended the 'Type' field through information analysis in articles. We grouped similar technologies into broader categories for added integrity and clarity. For instance, tools coded in Python and PHP were classified under 'General-Purpose Programming Languages'.

In addition to tool types, we conducted similar operations for our other research objectives. We document each extracted value in detail to present a clear and contextual picture, ensuring a thorough understanding of the information uncovered.

3.3 Reporting the Review

Reporting the review involved synthesizing and reporting the findings. Once all data was extracted, the results were organized and synthesized into a consistent storyline.

Data were extracted from the 27 selected articles. Most of them provided information directly relevant to the research objectives. During the extraction process, we reached a point where no new themes or insights emerged from the remaining articles. This reflects the concept of theoretical saturation Fusch and Ness (2015), where continued analysis no longer contributes additional information. As a result, the analysis captured a comprehensive range of perspectives, forming a solid foundation for the study's findings.

4 Research Results

This section begins with a discussion of search results for a review of the literature in Section 4.1, and a variety of tools discussed in the scientific literature for creating ETL pipelines are determined in Section 4.2. The results serve as a basis for classifying these tools based on their type in Section 4.3, target group in Section 4.4, and monetization approaches in Section 4.7. In addition, we review the evaluation methodologies used to assess such tools, taking into account alternative methodologies in the literature in Section 4.5. Finally, we examine the trade-offs in creating ETL pipeline tools and how these trade-offs influence the strengths and weaknesses of the tools in Section 4.6.

4.1 Search Results

We export our query results from Google Scholar into an Excel sheet, which we use to manage and screen all the articles. The spreadsheet includes basic information such as the title, authors, publication year, and abstract. It also contains columns where we apply and track our inclusion and exclusion criteria. This setup helps us stay organized and ensures that our review process is consistent and transparent.

We begin by checking each article's title to assess its relevance. If the title appears to match our research focus, we then read the abstract for a more detailed evaluation. Articles are excluded if they are duplicates, not peer-reviewed, not in English, or not accessible. All decisions are clearly recorded in the Excel sheet, making it easy to revisit or justify choices as we move forward with the review.

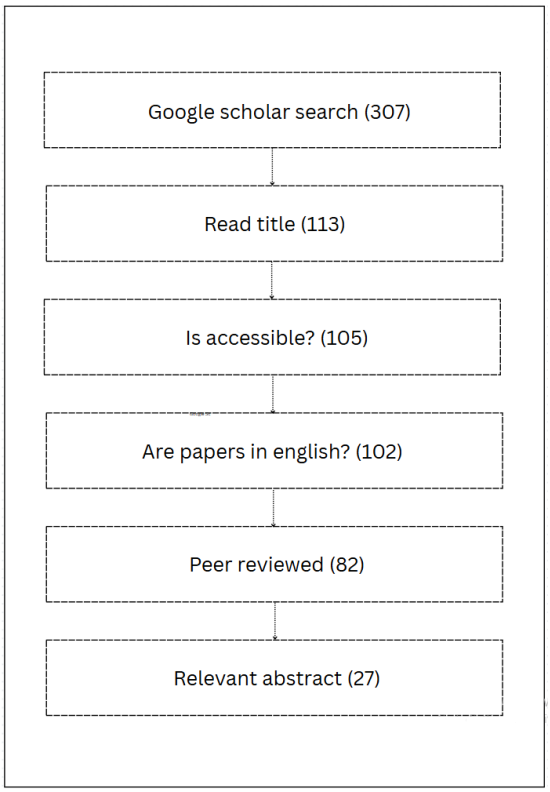

Figure 4.1: Screening and Selection Process for Scientific Papers

Figure 4.1 demonstrates that the initial search yielded 308 results on Google Scholar. After screening titles, this number was reduced to 113 articles. An accessibility check left 105 articles, and the exclusion of non-English publications brought the total down to 102. Verifying peer-reviewed status was confirmed by means of metadata filtered down the selection to 82 articles. Finally, after reviewing the abstracts, 27 articles were selected for analysis. The years of publication of these articles vary, with a notable trend of continuous research on this topic, reflecting sustained interest from the research community in this domain.

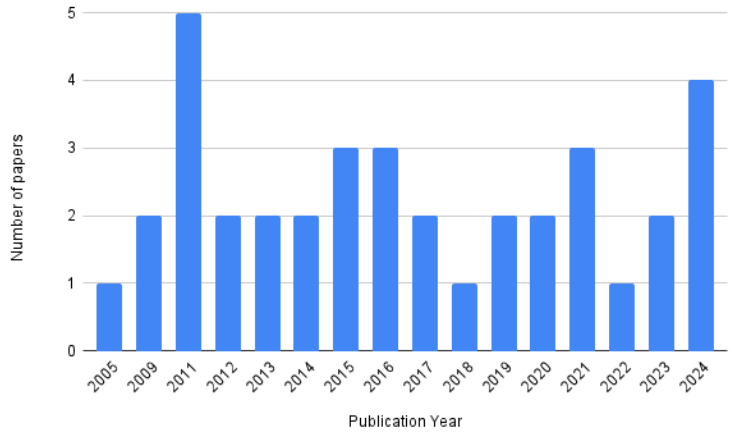

Figure 4.2: Distribution of Included Papers by Publication Year

Figure 4.2 shows the distribution of included papers by their publication year. While the dataset covers publications from 2005 to 2024, there has been a trend of continuous publication since 2011. This consistent pattern of publications indicates a continued and evolving interest in the area of ETL tools. Although some years had relatively fewer contributions, the steady presence of publications over more than a decade highlights the ongoing relevance and importance of this research area in the data engineering and software tools community.

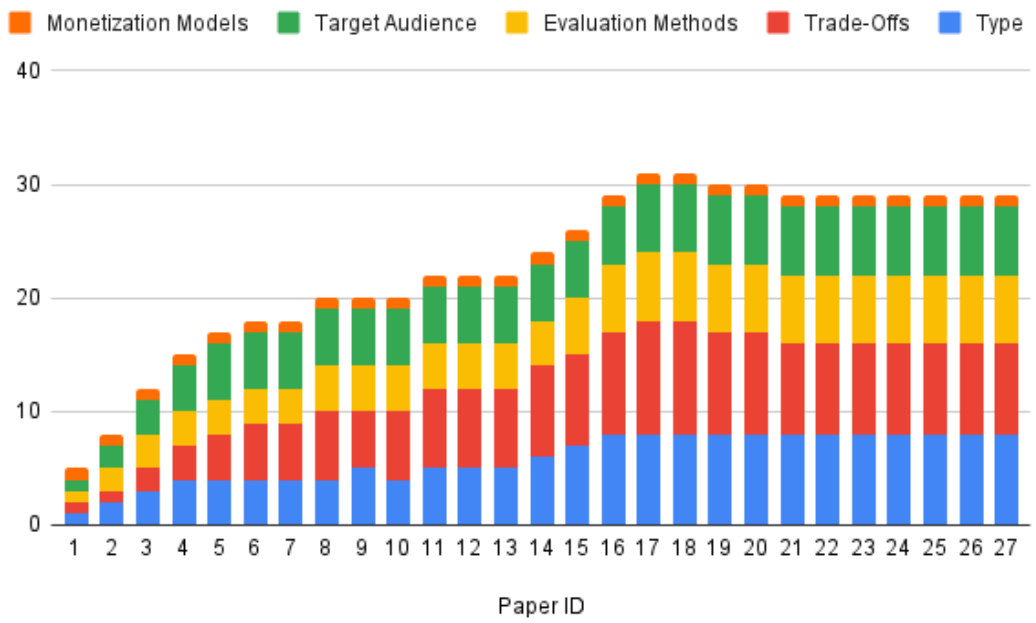

Figure 4.3: Total count of each element after analyzing every article.

We track the cumulative count of each required element, such as type, target group, trade-offs, and evaluation methods, across the analyzed articles, as shown in Figure 4.3. From Paper ID 21 onward, we observe no substantial changes in the identified categories, which indicates that theoretical saturation has been reached. While the overall trend shows a steady accumulation of categories, the stacked graph occasionally dips downward. This is due to the iterative nature of our process: as we review more papers, we refine our framework by adding, removing, or merging certain categories to better reflect the emerging themes.

4.2 Tools Discussed in the Literature

Each paper discusses a single tool, as shown in Table 4.1. This table presents the ID of the paper, along with the tool name. Some papers do not name the tool they discuss, while others explicitly name the tool. In addition, the table includes the citation of the article in which the tool is discussed.

#	Tool Name	Citation
1	BIcenter	Almeida et al. (2021)
2	NewTL	Debroy et al. (2018)
3	GENUS	Souissi and BenAyed (2017)
4	ETLator	Radonić and Mekterović (2017)
5	pygrametl	Thomsen and Pedersen (2009)
6	Bifrost	S. S. Shah et al. (2024)
7	ETL^2	R. Oliveira and Ramos (2015)
8	Informatica	N. Shah and Sawant (2015)
9	NMSTREAM	Xiao et al. (2018)
10	Web ETL Tool	Novak and Rabuzin (2010)
11	Database Agnostic ETL Tool	Sirsikar et al. (2015)
12	ETL NoSQL	Yangui et al. (2017)
13	Semantic ETL Framework	Bansal (2014)
14	Programmable Semantic ETL	Nath et al. (2015)
15	Web-based tool for data integration	Vijayendra and Lu (2013)
16	Framework for ETL Systems	da Silva and Times (2018)
17	Web-Based ETL Tool	Al-Rahman and Hasan (2014)
18	ETL for Health Databases	Quiroz et al. (2022)
19	Framework for ETL Process	Akkaoui et al. (2011)
20	ETL Tool for Structured Data	Wang and Liu (2020)
21	ETLCL	Popović et al. (2023)
22	A New Tool for ETL process	Chen and Zhao (2012)
23	Implementation of ETL Tool	Ying-lan and Bing (2009)
24	DSL for ETL	B. Oliveira and Belo (2016)
25	Script-Based ETL Tool	Li et al. (2016)
26	DSL to Enrich ETL Schemas	Belo et al. (2016)
27	ETL Tool for Cancer Data	Lakhan and Singh (2021)

Table 4.1: List of ETL Tools and References

4.3 Categorization of the Tool Type

The first area we explore in the literature review focuses on how ETL pipeline creation tools are classified. This involves understanding how these tools are grouped based on the technologies they use and the design approaches they follow.

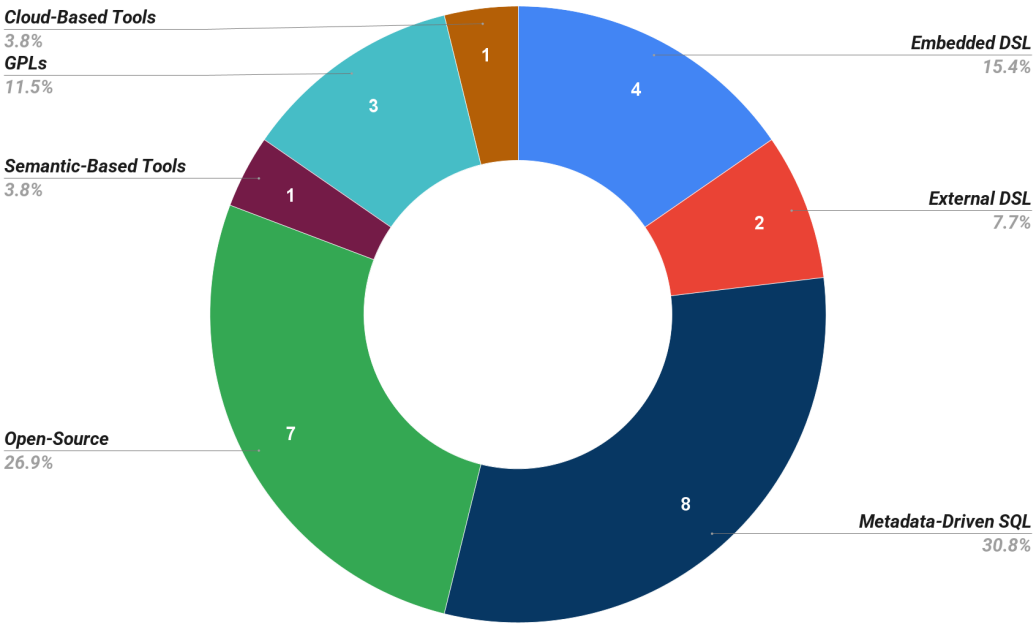

Figure 4.4: Distribution of Tool Categories.

4.3.1 Metadata-Driven SQL-Based tools

Among the several types of ETL tools identified in the literature, metadata-driven SQL-based tools are the most commonly discussed. These tools work with a metadata-driven approach using SQL, where information related to the ETL process such as source links, transformation rules, and target links can be stored in metadata tables. This information is then used to perform several tasks, such as generating automated queries and keeping everything consistent. It also allows us to edit, update, or add new information from a single location.

In the literature, several ETL tools are described that follow a metadata-driven SQL-based approach, each applying the concept in slightly different ways. Informatica (8, N. Shah and Sawant (2015)) is a tool developed for multi-database environments, where metadata is used to capture source and target table definitions, mappings, and transformation rules. This information is then used to dynamically generate SQL queries tailored to platforms like DB2, Oracle, and

MySQL, enabling flexible cross-database ETL execution. Another tool applies the same principle in a telecom setting, with metadata not only driving mappings but also embedding data cleaning rules directly, such as removing unwanted characters or reformatting product fields. A third case involves a database-agnostic system designed to replace commercial tools by using a metadata repository and template-based SQL generation to support different database backend.

An ETL tool for Health Databases (18, Quiroz et al. (2022)) demonstrates how this metadata-driven model can be applied to healthcare data integration. Their tool transforms source data into the OMOP Common Data Model using a combination of SQL and structured YAML configuration files. Instead of relying on lengthy, monolithic SQL scripts, this approach defines transformation logic in a column-by-column format within YAML, making the ETL logic clearer and easier to maintain. Each YAML file represents one OMOP table, specifying primary keys and transformation rules using PostgreSQL syntax. The system also includes a lightweight web interface to help users create and manage these configuration files, which are then compiled into SQL scripts for execution. By separating configuration from execution and structuring the logic in a transparent way, this tool improves maintainability, reusability, and collaboration, particularly in sensitive domains like healthcare.

4.3.2 General-Purpose Programming Languages

In contrast to the previous approach, data pipelines developed using general-purpose programming languages (GPLs), such as Java or Python. GPLs let developers build custom transformations tailored to specific needs from scratch.

For instance, A Web-Based ETL Tool (17, Al-Rahman and Hasan (2014)) is implemented entirely in Java, where the developers create custom classes to handle data extraction, transformation, and loading. Using JDBC, the tool interacts with databases directly, and transformation logic, such as field mapping and data validation, is hardcoded within the source files. Another tool, a Web-based tool for data integration (15, Vijayendra and Lu (2013)) combines PHP scripts, Korn shell scripting, and MySQL to perform ETL tasks via a basic web interface. While users can define inputs through the frontend, all processing steps are carried out through backend scripts that handle file parsing and data loading. A third example follows an object-oriented design pattern, where each ETL phase, like extraction, filtering, and loading, is coded as a separate software component. These components are assembled programmatically without any use of graphical workflow tools or DSLs, and all changes to the pipeline have to be made directly in code.

4.3.3 Domain-Specific Languages

Building on the flexibility of general-purpose programming languages, domain-specific languages (DSLs) emerged as a way to balance customization with structure. Two types of DSLs were identified in the literature: embedded and external DSLs (Fowler, 2010). Embedded DSLs provided specific features within familiar host languages like Python or SQL, helping developers keep the versatility of GPLs while making aspects more structured. External DSLs, on the other hand, were standalone tools with their own syntax tailored for ETL tasks, especially for domain experts. However, this increased usability often came at the cost of flexibility, developers might find it harder to implement complex logic outside the DSL's scope, and integrating such tools into broader systems could be more challenging. Additionally, learning a new syntax introduces a learning curve and may reduce adoption if community support is limited.

Embedded DSLs

In the literature, the ETLator (4, Radonić and Mekterović (2017)) framework is discussed as an embedded domain-specific language implemented within Python to define ETL workflows programmatically. Rather than relying on visual tools or external configuration files, ETLator leverages Python's native syntax and object-oriented features to express ETL logic through a collection of predefined classes and scripts. This embedded DSL makes it easier for users to break down ETL processes into clear stages like setup, flow, and finalization, using simple naming rules to control the order of execution. It treats each row of data as a Python dictionary, which makes it straightforward to work with, and it follows the PEP 249 standard for connecting to databases. The approach not only supports the implementation of reusable and parameterized scripts but also supports complex tasks such as handling Slowly Changing Dimensions (SCDs). Additionally, ETLator integrates features for automatic logging and documentation generation, including visual flow diagrams. By using an embedded DSL, it offers developers both flexibility and structure, allowing them to build and manage ETL processes in a lightweight, script-based environment.

pygrametl (5, Thomsen and Pedersen (2009)) is another example where an embedded DSL is employed to simplify ETL programming within a general-purpose language (Python). It provides a library of ETL-specific classes and functions that allow developers to describe data extraction, transformation, and loading tasks using Python code. Through its embedded DSL, pygrametl introduces abstractions such as dimension and fact table objects, which encapsulate the underlying SQL operations required for tasks like lookups and inserts. Developers can combine these abstractions with Python's native control structures and data types to construct flexible, reusable, and maintainable ETL pipelines. The framework also supports key ETL features such as snowflaked dimensions and Slowly

Changing Dimensions, offering both object-oriented and functional styles. By embedding the DSL in Python, pygrametl empowers users with full programming flexibility while reducing the need to manually write and maintain raw SQL queries.

External DSLs

An external domain-specific language (DSL) named ETL Control Language (ETLCL) specifically designed to manage and orchestrate ETL processes in data warehouse (DW) environments were discussed in the literature. The authors identified challenges in maintaining platform-specific ETL systems, especially when orchestrating tasks across different tools and operating systems. To address this, they propose ETLCL (21, Popović et al. (2023)) as a platform-independent language that abstracts common ETL concepts, such as tasks, dependencies, schedulers, and execution environments, into a unified syntax. This language is implemented externally using the Eclipse Modeling Framework (EMF) and xText, which allow model-to-text transformation and code generation for specific platforms. ETLCL programs are written in a textual syntax reminiscent of SQL and can be transformed into platform-specific executable code. Through a metadata repository and orchestration services, ETLCL enables centralized ETL process definitions that are reusable and maintainable. The authors demonstrate that ETLCL reduces the complexity and time required to modify ETL workflows, manage dependencies, and switch between execution platforms. This makes ETLCL particularly valuable in enterprise contexts where ETL processes must be portable, maintainable, and abstracted from tool-specific implementations.

Another study proposes an external DSL (24, B. Oliveira and Belo (2016)) to define configurations of ETL patterns based on BPMN conceptual models. The DSL is text-based and enables users to specify reusable pattern configurations through domain-level instructions like Header, Input, Output, Rule, and Exception blocks. Each pattern can be translated into execution primitives for commercial ETL tools, enabling automatic code generation. The use of BPMN provides abstraction, and the DSL formalizes how the ETL tasks are mapped from conceptual to executable layers. The syntax is designed for end users (not embedded in a general-purpose language), and the authors propose that code generators can transform these definitions into tool-specific formats.

4.3.4 Open-Source Orchestration

Open-source orchestration tools, such as Apache Beam and Talend Open Studio, were noted for their emphasis on pipeline-level coordination, scalability, and support from active user communities. These tools often complemented other categories by acting as execution engines or control layers. Unlike DSLs or code-

based approaches, orchestration tools were more about managing the flow of ETL steps than handling the transformations themselves.

NMStream (9, Xiao et al. (2018)) and Bifrost (6, S. S. Shah et al. (2024)) are two ETL tools highlighted in the literature that reflect open-source efforts to simplify ETL orchestration using well-known components from the Apache ecosystem. Both aim to make data integration more accessible and configurable, with a particular focus on real-time processing. NMStream, for example, is designed as a flexible, event-driven system that can handle real-time ETL tasks, particularly useful in settings where data comes from sensors or spatial monitoring systems. It combines several familiar Apache tools: Flume to gather and process the data, Quartz Scheduler to manage when tasks run, and Cassandra to store everything in a scalable way. One of the most practical aspects of NMStream is how easily users can adjust their workflows using a simple drag-and-drop interface, no coding needed, and no need to restart the system to apply changes.

Bifrost takes a slightly different route. It focuses on creating a no-code environment where users can build ETL pipelines from modular building blocks. It is not built specifically for real-time data like NMStream, but it emphasizes transparency and simplicity. Users can visually connect different steps in the process and easily see how the data moves and transforms. Both tools steer away from requiring traditional programming or DSLs, instead offering more intuitive, graphical interfaces that make them easier to use, even for people without deep technical backgrounds. By building on top of proven Apache technologies, they make scalable and distributed ETL processing more approachable for users without deep technical backgrounds.

4.3.5 Cloud-Based Tools

The literature also identified growing interest in cloud environments. The cloud is not only a storage solution; it has evolved into a comprehensive environment capable of executing logic, automating workflows, and managing infrastructure at scale. With the help of cloud resources, we can store data, run transformation logic, and automate different steps of the ETL workflow within a complete and flexible environment.

NewTL (2, Debroy et al. (2018)) is a cloud-based, in-house ETL tool developed by Varidesk Inc. to manage data integration between their ERP system and multiple 3PL partners. Built on Microsoft Azure, it replaces commercial solutions like Dell Boomi, offering better performance, simplicity, and lower cost. NewTL handles scheduled and manual data transfers using Azure services like WebJobs, Queue, Blob, and Table Storage, supporting both inbound and outbound workflows. It processes XML and CSV files over FTP, transforms data between ERP and 3PL formats, and provides features like manual re-transmission, alerting, and

dashboard monitoring. The system proves reliable and scalable, achieving high availability and significant cost savings.

4.3.6 Semantic Web-Based Tools

Some tools used semantic web technologies to develop ETL data pipelines, such as the Resource Description Framework (RDF) and SPARQL to give data a clear structure and meaning that machines can interpret. This made it easier to connect information coming from different sources. RDF stores information using triples, which are simple statements made up of a subject, predicate, and object (for example, 'Jayvee is a DSL'). SPARQL is the query language used to pull and work with this data. In the context of ETL pipelines, these technologies made it easier to bring together data from different places and apply transformations in a more practical and flexible way.

A paper discusses and introduces a semantic ETL framework (13, Bansal (2014)) that integrates heterogeneous data sources using semantic web technologies, focusing on the "Variety" challenge of Big Data. Unlike traditional ETL systems, this approach enhances the transform phase by generating semantic data models using ontologies and transforming raw data into RDF triples. These triples are then queried using SPARQL, enabling more meaningful integration and analysis. The framework utilizes tools such as Protege for ontology modeling, Oxygen XML for converting CSV to OWL instances, and Apache Jena for running semantic queries.

4.4 Target Audience

In this section, we discuss the target audience of the tools mentioned in the literature, as shown in Figure 4.5. The majority of users had strong technical skills in areas like scripting, programming, and database management. The second largest group includes semi-technical users with foundational technical skills, such as SQL proficiency. This group, which includes students and new learners, benefits from ETL tools that combine ease of use with hands-on learning opportunities.

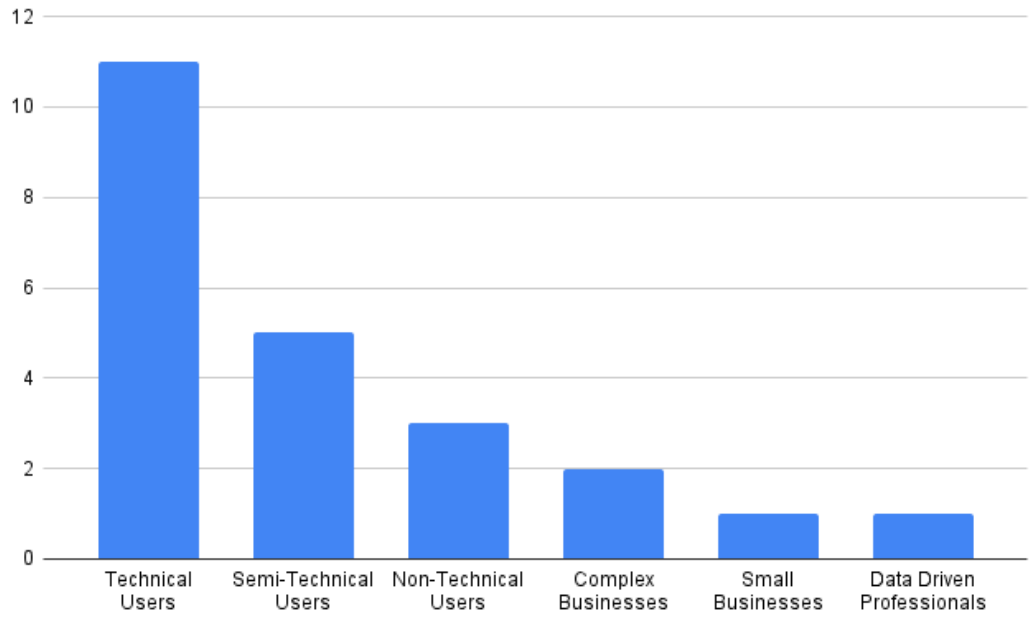

Figure 4.5: Distribution of target audience of the tools.

Non-technical users drive ETL tool adoption with simple, GUI-based interfaces featuring drag-and-drop functionality. Managers, business analysts, and other professionals rely on these tools for simplified workflows that require minimal technical expertise. Similarly, data-driven professionals look for ETL tools that help them gain insights and support decision-making, prioritizing features like analytics and visualization over technical complexity

Among businesses, complex organizations depend on ETL tools designed for scalability and customization to handle high volumes of data operations and intricate workflow configurations. Larger organizations usually need tools that can connect with a lot of different data sources and help manage data across the whole organization. Smaller businesses, on the other hand, tend to opt for ETL solutions that are affordable, easy to use, and low maintenance.

The distribution of these user groups shows the variety of needs that ETL tool development has to address. Each group brings its own priorities and skill levels, which influence how ETL tools are built and adopted in different sectors.

4.5 Evaluation Methods Used to Evaluate Tools

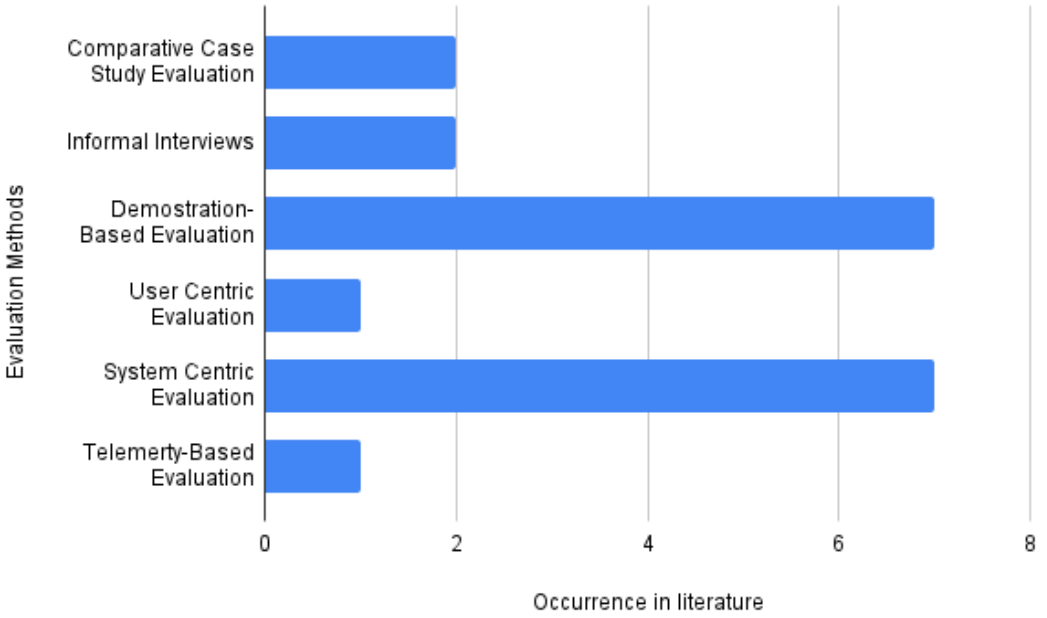

Figure 4.6: Frequency of evaluation methods used in ETL tool studies as observed in the reviewed literature.

The literature covers several ways to evaluate ETL tools. As shown in Figure 4.6, system-centric and demonstration-based evaluations are the most frequently used, each appearing in seven studies. Other methods like comparative case studies, informal interviews, user-centered evaluations, and telemetry-based assessments are used less often, typically appearing in one or two studies. This suggests a dominant focus on performance and functionality, with relatively less emphasis on user experience.

4.5.1 System-Centric Evaluations

System-centric evaluations use quantitative benchmarks such as execution speed, system responsiveness, or alignment with business rules, carried out through load testing or scripted data workflows.

Many tools are evaluated using system-centric methods in the literature, with a focus on accuracy, validation of outcomes, and tool-level efficiency. In the case of GENUS (3, Souissi and BenAyed (2017)), the authors design a validation process that tests whether data extracted and transformed from various sources, such as text, images, and videos, retain their original content after processing. For text

data, they implement a return-validation algorithm that cross-checks extracted concepts against the original source to confirm semantic preservation. For binary formats like images and videos, the tool uses base64 decoding to reconstruct the original files from their transformed states, verifying that no data are lost or corrupted during processing. The system also validates the structural consistency of the resulting data warehouse by checking that all generated dimension and fact tables align correctly with the expected schema design.

In the case of SETL (13, Bansal (2014)), the evaluation involves output quality and productivity measurements. To assess data quality, they compare the resulting RDF-based knowledge base (SETLKB) against a manually created benchmark dataset (ExtBIKB), examining the completeness and correctness of the output. Additionally, they evaluate productivity by tracking factors such as the number of lines of code required, the time needed for setup and execution, and the level of manual effort involved in creating the mappings. This helps to quantify how the tool streamlines development compared to more traditional approaches.

4.5.2 Demonstration-Based Evaluations

Demonstration-based evaluations, on the other hand, show how the tools work in action, either through example use cases or in real or simulated scenarios, like working with sensor data or business applications.

ETLator (4, Radonić and Mekterović (2017)) is developed using a Python-based scripting framework for ETL and is validated by applying the tool to the Northwind database schema, a standard dataset used for teaching and testing data operations. They show how various scripts can be organized within a directory structure to execute ETL tasks such as populating dimension and fact tables, handling slowly changing dimensions, and enabling parallel execution. Their demonstration includes automated logging and the generation of data flow diagrams, illustrating the tool's operational logic and helping to convey usability and performance characteristics in practice.

The Bifrost (6, S. S. Shah et al. (2024)) system is evaluated by implementing it in a real-world ETL scenario and walking through how users can visually construct ETL workflows using a no-code, block-based interface. The paper details a case in which data are transformed and loaded through drag-and-drop blocks, giving insights into how the tool supports modularity and step-by-step visibility. The demonstration aims to show how even users without programming knowledge can create and monitor ETL tasks using the GUI.

4.5.3 Informal Interviews

Evaluations involving direct human feedback are less common but offer distinct value. Few studies include informal interviews, but these give a closer look at how users interact with the tools, how easy they are to use, and how much work goes into building them.

A tool discussed in one of the papers, ETL2 (7, R. Oliveira and Ramos (2015)) developed as an educational aid for teaching ETL processes, is iteratively refined over five years of classroom use. During this period, students provide continuous feedback about the tool's usability, debugging features, and testing mechanisms. The feedback is gathered informally throughout the semester and is considered especially valuable since a portion of the students have relevant industry experience. This iterative feedback loop enables the developers to adjust the tool's complexity and improve the interface and workflow in response to user's real-time struggles and suggestions. Similarly, the ETLCL (21, Popović et al. (2023)) emphasizes the role of domain experts (including the authors themselves) in shaping the language features. While not based on formal interviews, their evaluation leans heavily on the authors' dual expertise in ETL system design and DSL engineering, blending practical experience and internal critique to assess the language's usability and completeness.

4.5.4 User-Centered Evaluations

User-centered evaluations, which focus on interaction with the tool, are notably rare, and appear only in studies where user experience is a primary concern. This pattern shows an uneven focus—most evaluations lean heavily on performance metrics or controlled tests, with far fewer looking at how real users interact with the tools.

In the literature, a Web-Based ETL Tool (17, Al-Rahman and Hasan (2014)) is evaluated using a user-centered approach that emphasizes usability and practical interaction in real-world settings. Rather than relying on formal interviews or structured usability studies, the evaluation is based on observations of user interaction, system responsiveness, and overall workflow support. Particular attention is given to the tool's interface clarity, ease of use, and its ability to integrate smoothly into daily operations. The focus of the evaluation is on how effectively the system meets users' needs and maintains accuracy throughout routine tasks.

4.5.5 Comparative Case Study Evaluation

Comparative case study approach is used to evaluate ETL tools by applying them to similar scenarios and comparing their outcomes. This method allows researchers to examine differences in aspects such as performance, development time, and

ease of use under consistent conditions. It is particularly useful in highlighting how the proposed tools perform relative to existing solutions, helping to illustrate their practical advantages and limitations through real-world examples.

The papers on pygrametl (5, Thomsen and Pedersen (2009)) and the Script-based Automation ETL Tool (25, Li et al. (2016)) use comparative case studies as an evaluation method, contrasting their proposed tools with existing solutions to assess practical benefits. In the case of pygrametl, the authors implement the same ETL workflow using both pygrametl and a graphical ETL tool (Pentaho Data Integration), measuring development time and ease of implementation. Their comparison shows that coding with pygrametl is faster and more expressive, especially for complex tasks like managing snowflakes and slowly changing dimensions. Similarly, the Script-based Automation ETL Tool is compared against Apache Sqoop in a scenario involving data transfer between Oracle and Impala. The authors evaluate performance metrics such as speed and efficiency across various dataset sizes. Their results indicate that their script-based tool outperforms Sqoop, especially for larger datasets, offering a faster and more streamlined alternative for data extraction and loading. These comparative studies provide a grounded way to demonstrate how the proposed tools perform in realistic use cases.

4.5.6 Telemetry-Based Evaluations

Telemetry-based evaluations are also rarely used. These rely on collecting data from system logs to keep track of things like response time or data accuracy during real-time use.

newTL (2, Debroy et al. (2018)), a system described in the literature, is a cloud-based ETL solution developed in-house by Varidesk Inc., and it is evaluated using telemetry-based methods. This approach involves the use of telemetry data collected through Azure Application Insights to continuously monitor the system's performance and reliability. Key performance indicators such as average response times and system availability are tracked over time. For instance, the dashboard's average response time is consistently under 500 milliseconds, and up-time is reported to be uninterrupted throughout the year. Additionally, extensive availability testing is conducted from multiple regions to simulate real-world access patterns, and the results consistently confirm high responsiveness and zero downtime. This kind of telemetry-driven evaluation enables the authors to validate the system's real-world robustness, scalability, and efficiency using live performance data rather than controlled or simulated tests.

4.6 Metrics Used in Evaluation Methods

#	**Evaluation Method**	**Addressed Metrics**
1	System-centric evaluation (7 occurrences)	Portability, Functional Suitability, Productivity
2	Demonstration-based evaluation (7 occurrences)	Functional Suitability, Productivity
3	Informal interviews (2 occurrences)	Portability, Extensibility, Usability and Interface
4	Comparative case study (2 occurrences)	Development Effort
5	User-centered evaluation (1 occurrences)	Usability and Interface
6	Telemetry-based evaluation (1 occurrences)	Productivity

Table 4.2: Overview of Evaluation Methods and Addressed Metrics

Table 4.2 summarizes the evaluation methods used in ETL tool studies and the metrics they emphasized. Most studies focused on productivity, functional suitability, and validation accuracy, especially within system-centric and demonstration-based evaluations. While some metrics, like usability, development effort, and interface quality, were subjective or experience-based assessments, others such as accuracy were more quantitative and outcome-driven.

4.6.1 Development Effort in Pipeline Creation

Development effort is used to understand how much time and complexity goes into building ETL pipelines using a tool. In many cases, researchers look at factors like how much code needs to be written, how difficult it is to set up the environment, or how quickly users can create working pipelines. Tools that reduce repetitive tasks, offer reusable components, or have simple configuration options are generally seen as less demanding. This metric helps reveal how beginner-friendly or scalable a tool might be, especially when development teams have limited resources.

4.6.2 Portability in Tool's Functionality

Portability refers to how easily an ETL tool can be used across different systems or environments. If a pipeline can run on multiple platforms (like Windows and Linux) or connects to different types of databases without much change, that counts as a plus. In some papers, tools are tested in different settings to show how flexible their outputs are. Portability is particularly useful for teams working in diverse IT ecosystems, as it reduces the need to rebuild workflows from scratch when switching platforms.

4.6.3 Extensibility of Tool

Extensibility is about whether the tool can grow with the project's needs. This comes up in evaluations where the tool allows users to add new transformation steps, plug in custom code, or support new data sources without overhauling the entire system. If developers can tweak or extend the tool easily, that is considered a strength. This metric shows how future-proof a tool might be, especially valuable when working on long-term or evolving data integration projects.

4.6.4 Functional Suitability

Functional suitability focuses on whether the tool can actually perform its intended tasks, such as connecting to data sources, applying transformations, and loading data reliably. Researchers often check if the tool handles complex tasks like joining data from different formats or applying advanced filters. Tools that support reusable workflows and adapt to a variety of data formats score well here. This metric is at the core of most evaluations, as a tool that cannot perform its basic job, even if it is easy to use, will not be of much help in practice.

4.6.5 Productivity Metrics

Productivity is all about how quickly and efficiently developers can get work done with the tool. In several papers, this is measured by looking at how long it takes to complete a pipeline, whether tasks are automated, or how much manual coding is needed. Some tools are shown to speed up development by offering visual interfaces or pre-builtt templates. Others highlight time savings by simplifying debugging or reusing configurations. Productivity metrics give a clear sense of whether a tool helps reduce effort in day-to-day ETL development.

4.6.6 Validation and Accuracy Metrics

Validation and accuracy metrics are used to check whether the pipelines built with the tool actually produce correct results. This means checking if data are

transformed properly, if mapping rules are applied as expected, and if outputs match the defined schema or benchmarks. In some cases, tools are tested by comparing their output to a manually prepared gold standard or by verifying data consistency after processing. These metrics are especially important for tools handling sensitive or complex data, where errors could lead to serious downstream issues.

4.6.7 Usability and Interface Metrics

Usability and interface metrics focus on how easy and intuitive the tool is for users—especially those without deep technical backgrounds. Evaluations that look at this often involve observing user interactions, collecting informal feedback, or noting how clearly the interface guides users through tasks. Features like step-by-step configuration, visual feedback on errors, or support for drag-and-drop workflows make tools stand out in this area. These metrics help identify tools that are not only powerful, but also accessible and pleasant to use.

4.7 Trade offs Exist in the Development of the Tools

Several trade-offs exist in the development of ETL pipelines, and the literature discusses how each choice between alternatives carries both positive and negative impacts.

4.7.1 Metadata-Driven Design vs Hardcoded Logic

A foundational trade-off is observed between metadata-driven design and hardcoded logic. Metadata-driven tools, often built on SQL, store essential ETL information such as the source of the data, transformation rules, and target destination of the data in metadata tables. This supports automatic query generation and reduces the need for manual scripting. But the generated queries are not optimized, because they prioritize versatility over task-specific precision. In contrast, hardcoded logic provides greater control and easier debugging but involves consistent code changes. Maintaining the system takes considerable effort and increases the likelihood of errors.

For example, the SETL tool proposed by Chen and Zhao (2012) automatically generates SQL transformation procedures from metadata mappings, which improves scalability and makes the system easier to maintain over time. It also allows configuration changes without touching the code, making it more adaptable to evolving data requirements. Similarly, Wang and Liu (2020) introduced a metadata-centric model that supports custom data cleaning functions and stores SQL expressions as metadata to avoid recompilation. This improves efficiency in recurring operations but makes optimization and debugging more difficult, since the logic is not directly exposed. In contrast, earlier tools like the one described in Sirsikar et al. (2015) rely on hardcoded logic, giving developers greater control and better performance in task-specific use cases. These implementations are easier to debug due to their transparency but are less flexible and require frequent code changes for updates, which increases the maintenance burden.

4.7.2 Code-Based vs GUI-Based Tools

Some of the limitations of hardcoded approaches are addressed in the trade-off between code-based and GUI-based tools. Code-based tools maintain flexibility and allow the implementation of complex logic, making them suitable for advanced users. On the other hand, since these tools rely on coding, they may be difficult for users who lack programming experience. GUI-based tools attempt to minimize this by offering visual interfaces that boost development time and make the tools more accessible. While these tools are easy to use, they can be

limiting, as users must work within the boundaries of the predefined interface.

Bifrost S. S. Shah et al. (2024) offers a drag-and-drop interface where users can build data pipelines without writing any code at all. This makes it great for quick setup and collaboration across teams with mixed technical skills, but GUI tools can quickly become limiting when you need to perform complex or highly customized transformations that go beyond the built-in components. On the other hand, scripting-based tools like pygrametl Thomsen and Pedersen (2009) and ETLator Radonić and Mekterović (2017) give developers much more control. These frameworks allow you to define ETL logic directly in Python, which makes it easier to reuse code, handle edge cases, and optimize performance. For example, ETLator lets you organize workflows using folders and simple scripts, making it easy to automate repetitive tasks or manage large-scale pipelines. While this approach does require more programming knowledge, it offers better scalability, flexibility, and transparency when things go wrong. So, the choice often comes down to whether you need ease of use and fast on-boarding, or greater precision and long-term control.

4.7.3 Declarative vs. Procedural Approach

A similar trade-off can be identified when comparing declarative and procedural approaches. Declarative approaches highlight specifying the targeted output, while the system takes care of the underlying operations. This is often done using configuration files like YAML, which streamline development by encapsulating internal logic. Compared to GUI-based tools, declarative designs still require technical input but reduce the effort needed for development. With increasing workflow complexity, the layers of abstraction may hinder effective debugging. Although procedural designs are more detailed and require additional time, they provide developers with full visibility and control—capabilities that are often essential when working with non-standard or changing data requirements.

ETLCL Popović et al. (2023) introduces a domain-specific language that lets users define ETL tasks and control flows in a platform-agnostic way. Similarly, the approach described by Yangui et al. (2017) models ETL logic using BPMN diagrams, enabling developers to specify transformations as high-level processes. These declarative models make ETL logic easier to maintain and transfer across systems, especially when execution is handled by external services or connectors. On the other hand, tools like SETL Nath et al. (2015) follow a more procedural approach. While SETL leverages semantic web concepts like ontologies, the actual transformation steps are scripted explicitly in a fixed sequence, requiring users to define the logic in a structured and controlled manner. This procedural model provides fine-grained control and can be more familiar to developers used

to scripting, but it tends to be less flexible and harder to scale or adapt when requirements change.

4.7.4 Development simplicity vs Optimization

Whereas the previous trade-offs emphasize control versus ease of use, others deal more directly with performance versus simplicity. Tools optimized for simplicity are faster to adopt and easier to set up, making them appealing in fast-paced or resource-constrained environments. However, these tools may lack the tuning and configuration required for high-performance tasks. A comparable issue is noted with declarative approaches, where ease of use sometimes reduces both clarity and adaptability.

The tool developed by Chen and Zhao (2012) is a clear example of this trade-off. Instead of relying on hand-written, highly optimized queries, this tool uses metadata to automatically generate SQL procedures. This reduces the need for manual coding, simplifies updates, and makes the system more adaptable to change. But, this also means that the generated queries may not be as performance-efficient as those crafted by experienced developers. The authors acknowledge that while optimization could improve execution speed, the tool is intentionally designed to favor simplicity, reusability, and long-term maintainability.

4.7.5 Ontology-Based Integration vs Simpler Schema Models

Moving into data integration, another trade-off emerges between ontology-based integration and simpler schema models. Ontologies provide a structured, semantic approach to data consistency and cross-system linking. Despite their ability to ensure consistent data and facilitate integration across platforms, they require advanced tools and specialized domain knowledge for effective configuration and maintenance. In contrast, simpler schema models are more user-friendly and quicker to deploy, which enhances their practical accessibility. This ease of use is accompanied by limitations, especially in terms of adapting to more intricate or evolving datasets. This dynamic reflects a similar pattern seen in the declarative versus procedural trade-off, where ease of use can limit adaptability over time.

An example of this trade-off can be seen in the semantic ETL framework proposed by Bansal (2014). Instead of relying on a traditional schema-based model, where source and target data structures are mapped directly, the authors introduce an ontology-based approach. In their system, domain ontologies are used to

describe concepts and relationships across multiple heterogeneous data sources. This allows the ETL process to not only transform data structurally, but also align it semantically. While this adds complexity, requiring tools like Protege for ontology design and additional reasoning steps, the benefit lies in increased flexibility and the ability to perform richer, cross-source queries using technologies like RDF and SPARQL. The authors acknowledge that schema-based models are simpler to implement, but argue that ontology-based modeling is better suited for scalable, meaningful integration in complex data environments.

4.7.6 Functional simplicity vs Versatility

A related but broader trade-off questions simplicity versus versatility. Systems developed for limited-scope tasks place a higher value on usability and cost efficiency. Their limited functionality can lead to challenges when adapting to evolving project needs. More versatile tools address this constraint by supporting a wider variety of use cases. However, they tend to introduce more complexity and require technical skill, echoing earlier trade-offs between GUI and code-based tools, and between declarative and procedural approaches.

The trade-off between functional simplicity and versatility appears in the web-based ETL tool developed by Novak and Rabuzin (2010). Rather than aiming to support a wide array of transformation features or data source types, the authors focused on creating a tool that is easy to access and use, especially for beginners. The tool includes a clean interface, a limited set of connectors (e.g., text files and MySQL), and a small but practical collection of transformation functions like string formatting and date extraction. While this limited scope sacrifices some versatility, it lowers the learning curve significantly and makes the tool ideal for educational use or small-scale ETL tasks. The authors deliberately chose functional simplicity over advanced customization, arguing that accessibility and ease of use were more valuable in their context than a broader or more flexible feature set.

4.7.7 In-house build vs buy/open source

Finally, the build versus buy dilemma captures many of these tensions at the strategic level. Developing tools in-house allows for complete customization and long-term savings, though it requires a strong technical team and high upfront cost. While these solutions can ease development and speed up deployment, they often fall short when it comes to accommodating specific domain requirements.

An example of the trade-off between building an in-house system and adopting a commercial or open-source alternative can be found in the development of NewTL

Debroy et al. (2018). Initially, the team evaluated several off-the-shelf tools, but found them either too expensive, too complex for their needs, or insufficiently customizable to integrate seamlessly with their ERP system. Despite the common perception that building ETL systems from scratch is prohibitively difficult or costly, the authors decided that an in-house solution offered better alignment with their business requirements. While building the system involved greater initial development effort, it ultimately gave the company a solution tailored exactly to their data integration needs, proving that for some organizations, in-house development can offer advantages that outweigh the convenience of prebuilt tools.

Table 4.3 provides an overview of the key trade-offs explored throughout the previous discussions. For each case, it summarizes the design choices made by different ETL tools, how frequently those trade-offs appeared, and the associated positive and negative impacts. This serves as a consolidated snapshot of the themes and decisions analyzed across the selected tools, helping to tie together the detailed examples and highlight recurring priorities.

#	**Trade-Off exists A vs B (Occurrence Frequency)**	**Choice made (Occurrence Frequency)**	**Impacts of the choice A (Positive/Negative Impact)**	**Impacts of the choice B (Positive/Negative Impact)**
1	Metadata-Driven Design vs Hardcoded Logic (Frequency: 4, In tool: 11,22,20,18)	Metadata-Driven Design (4)	Positive: *Automation, Performance efficiency, Controlled integration* Negative: *Optimization issues, Requires expertise*	Impacts not discussed explicitly
2	Declarative vs. Procedural Approach (Frequency: 3, In tool: 14,21,12)	Declarative Approach (2) Procedural Approach (1)	Positive: *Controlled Integration, Development Efficiency, Performance Efficiency* Negative: *Requires Expertise*	Positive: *Controlled Integration* Negative: *Increased Development Time*
3	Code-Based vs GUI-Based Tools (Frequency: 3, In tool: 4,5,6)	Code-Based (2) GUI-Based Tools (1)	Positive: *Functional flexibility* Negative: *Requires expertise*	Positive: *Development Efficiency* Negative: *Technical Limitations*
4	Development simplicity vs Optimization (Frequency: 2, In tool: 22,18)	Development simplicity (2)	Positive: *Ease of adoption, Development efficiency*	Positive: *Scalability* Negative: *Requires Expertise*
5	Ontology-Based Integration vs Simpler Schema Models (Frequency: 2, In tool: 13,14)	Ontology-Based Integration (2)	Positive: *Interoperability and sharing* Negative: *Requires Expertise*	Positive: *Development Efficiency*
6	Functional simplicity vs Versatility (Frequency: 2, In tool: 21,15)	Functional simplicity (2)	Positive: *Ease of adoption, Cost effective* Negative: *Limited functionality*	Positive: *Functional flexibility* Negative: *Requires Expertise*
7	In-house build vs buy/open source (Frequency: 1, In tool: 2)	In-house build (1)	Positive: *Cost effective, Independent Development*	Positive: *Development Efficiency* Negative: *Lack of Customization*

Table 4.3: Trade-Offs and Impact of the Choices

4.8 Monetization Models for the Tools

In the literature, it is not explicitly discussed which monetization models are applied to the ETL tools that are created. While various tools are introduced and evaluated, details about their business models, licensing structures, or revenue strategies are largely absent. However, at some points, it is mentioned that the tools are open source and freely available for public use.

5 Discussion

In this chapter, we discuss points that emerge from our analysis of the literature regarding the tools. Subsection 5.1 discusses how tools are classified based on their underlying technologies and intended users. Subsection 5.2 reflects the ways in which they are evaluated, and subsection 5.3 interprets the trade-offs developers face between usability, flexibility, and performance. In subsection 5.4, we identify a gap in the literature regarding attention to monetization models.

5.1 Tool Categorization and Target Audience

Metadata-driven SQL-based tools and open source orchestration are the most prominent tools used in data pipeline creation. To use these tools, users need a strong technical background to handle the complexity of the metadata-driven approach and to understand how to use different open source tools to create ETL data pipelines. Working with these tools often requires solid technical skills, which matches the largest user group we identify in our analysis. In contrast, semi-technical users, the second most common group in the literature, tend to go for tools that strike a balance between control and simplicity. Non-technical users tend to go for visual tools that are straightforward and easy to use. This mix of user needs shows why it is important for ETL tools to work across different experience levels, from hands-on, code-based setups to simple, accessible interfaces.

5.2 Evaluation Methods and Metrics

Most tools are evaluated with a strong focus on how they perform technically, such as speed, reliability, and how well they adapt across systems. System-centric and demo-based evaluations tend to dominate, highlighting performance over user experience. On the other hand, approaches that focus on actual users, like user-centered evaluations, informal interviews, or telemetry data, are used far less often. Even when informal interviews are included, they are usually done with non-expert users, which limits how much insight they can offer about deeper usability issues. This shows a gap in how tools are assessed. Focusing

mostly on technical benchmarks tends to miss the real-world challenges that users, especially those without a technical background, often run into when trying to use these tools. Shifting some attention toward more user-focused and qualitative evaluations could bring out issues that are not always visible in performance tests, such as how smoothly a tool fits into any workflow, how easy it is to use, and whether people actually find it adaptable or frustrating to work with.

5.3 Trade-Offs in ETL Tool Creation

Tools tend to reflect a bigger push in the industry, trying to balance performance, flexibility, and ease of use. We often see a trade-off between making things faster and making them harder to use. Approaches like metadata-driven setups or declarative methods work well, but they are not easy for everyone to handle. Most of these tools are clearly built with skilled developers in mind.

Another trend that comes up is the growing push to make tools easier to use. A lot of newer tools now come with visual interfaces, clearly built for people who do not have technical background but once trying to do anything more advanced often leads to limitations, as flexibility and control are traded off for ease of use. This is not surprising. It is something we are seeing across the industry. Low-code and no-code tools are becoming more popular, but when users hit their limits, especially in more complex projects, they often shift back to fully code-based solutions that give them more freedom.

One of the key trade-offs observed in ETL tool development is between development effort and control. Simpler tools definitely help people with less technical background get started, but they often cannot handle more complex tasks. That seems to be part of why some businesses end up creating their own tools, not just to customize things, but because the problems they are dealing with are too specific or complicated for ready-made solutions. It makes us wonder if current tools can really strike that balance between being flexible enough for advanced users and still usable for everyone else. It is not an easy thing to pull off, but it is probably the direction future tools need to move in.

5.4 Monetization Model

Discussion of monetization models for the tools is missing from the literature. Every article that discusses a tool for ETL data pipeline creation states that the tool is completely open source. This suggests a strong focus on technical features, without giving attention to how these tools could be positioned in the market, and overlooking questions of economic viability and market strategy. Whether

these tools continue to grow often depends on funding or paid support. Without considering that, it's hard to judge their future potential.

6 Limitation

We evaluate the quality of our research using the trustworthiness criteria proposed by Guba and Lincoln (1985), which include credibility, transferability, dependability, and conformability. This approach is chosen because our research is qualitative in nature, and these criteria provide a structured and transparent framework for evaluating the rigor and overall quality of qualitative studies.

6.1 Credibility

Credibility reflects confidence in the accuracy of the findings of this study. Although the research is grounded in peer-reviewed literature, it does not include direct input from ETL tool developers or users. As a result, certain practical insights may be absent. Including expert interviews or user feedback in future research could strengthen the credibility of the findings by ensuring that the identified trade-offs and tool evaluations are more closely aligned with real world industry practices.

6.2 Transferability

Transferability concerns how well the study's findings may hold true in different contexts. While this study provides a structured categorization of key components in ETL data pipelines, its conclusions may not be fully generalizable to all industries or to emerging ETL technologies that are not yet widely represented in the current literature. Since the study focuses mainly on academic sources, industry-specific applications might be underrepresented. Future research could improve transferability by examining real-world case studies or conducting cross-industry comparisons.

6.3 Dependability

Dependability refers to the consistency of the research process and whether the findings would hold up if the study is repeated under similar conditions. In

this study, we use a systematic review approach to ensure clarity in how data are collected and categorized. That said, some external factors, like shifts in database indexing, changes to search algorithms, or the rapid evolution of ETL tools, could affect how easily the results can be reproduced. To strengthen the dependability of our work, we follow a clear and structured process, publish our raw data, and document each step so that others can retrace and verify the analysis.

6.4 Conformability

Conformability concerns the neutrality and objectivity of the research. In this study, the classification of tools, evaluation methods, trade-off analysis, and target audience categorization are grounded in published literature to reduce personal bias. Nevertheless, variations in how existing studies are interpreted may still influence the classification and assessment of tools. To strengthen the conformability, future work could involve inter-researcher validation or support findings through triangulation with multiple data sources.

7 Conclusion

We aim to classify ETL (Extract, Transform, Load) pipeline creation tools through a systematic literature review. Our goal is to explore the different types of tools used to build ETL data pipelines, understand who they are built for, and look into how they are monetized, based on what the literature describes. We also set out to review how these tools are evaluated, which metrics are commonly used, and what trade-offs are discussed in their development. This thesis provides a structured framework for addressing our research goals.

The findings reveal that metadata-driven SQL-based frameworks and open-source orchestration tools are the most frequently used in ETL data pipeline development. In contrast, tools built with domain-specific languages, general-purpose programming, semantic web technologies, GUI-based design, and cloud-based tools appear less often. The findings also indicate that most tools are designed for users with some technical knowledge, making them less accessible to those with limited technical expertise. When it comes to evaluating the tools, the focus is mostly on system-level performance, with methods like system-centric and demonstration-based evaluations being the most common. In contrast, approaches that involve factors from user's perspective such as user-centered evaluations, case studies, or interviews are mentioned far less often, showing that usability and user experience are not a primary concern. The analysis also brings up several trade-offs in tool creation, especially the balance between using metadata-driven approach versus hardcoded logic, and between declarative and procedural approaches. These decisions often reflect deeper choices around flexibility, control, and how easy a tool is to maintain.

With this study, developers of ETL data pipelines have the opportunity to learn key factors that can help them plan and design an ETL tool such as Jayvee in a nutshell. Although we comprehensively discuss various aspects regarding the tools that can create ETL data pipelines, future research could look into areas that are less explored in literature, such as how these tools are monetized. Alongside peer-reviewed sources, insights from gray literature could help fill these gaps.

Acknowledgment

All results and the review process presented in this thesis were conducted by the author, who takes full responsibility for their accuracy and integrity. To improve grammar and sentence structure, language refinement tools such as ChatGPT and Overleaf's integrated Writefull feature were used where necessary.

Appendices

A Excel Dataset: Initial Paper Pool and Review Process

The accompanying Excel file, `etl-review-data.xlsx` (Download), documents the complete workflow of the systematic literature review conducted in this thesis. It provides a transparent and reproducible overview of how papers were identified, filtered, and analyzed. The file includes the following sheets:

- **Sheet 1: Initial Pool and Screening Process:** This sheet contains the complete list of papers retrieved during the initial search. In addition to standard metadata (title, authors, year, source), this sheet also documents how inclusion and exclusion criteria were applied iteratively for each paper.

- **Sheet 2: Codebook:** This sheet contains a *codebook section* that tracks evolving category labels such as tool type, evaluation methods, target audience, and trade-offs. It reflects the iterative process of coding and classification carried out during paper screening.

  It was used primarily for summarizing trends and generating tables and visualizations in the thesis.

- **Sheet 3: Data Extraction Form:** This sheet contains the final structured data extracted from the 27 included studies. Each row corresponds to one paper, and columns represent standardized fields derived from the research questions.

  This form was based on a protocol aligned with the guidelines by Kitchenham and Charters, 2004

The Excel file is included in the digital supplementary materials submitted with this thesis and serves as a detailed trace of how the literature was reviewed and categorized.

B Reproducible Package Overview

This appendix documents the contents and structure of the reproducible package created for this thesis, titled *"A Systematic Review of Tools to Create ETL Pipelines."* The package provides all necessary files and instructions to replicate the review process and analysis described in the thesis.

B.1 Project Scope and Motivation

The project investigates tools designed to support the creation of ETL (Extract, Transform, Load) pipelines. It analyzes their design approaches—ranging from GUI-based systems for non-programmers to general-purpose and domain-specific programming languages—and compares their strengths, weaknesses, and evaluation methods. The reproducible package allows readers to follow the same workflow used to select and analyze literature.

B.2 Structure and Replication Steps

The reproducible package is organized into the following components:

- **Search Query**
  Contains the original search strategy used to retrieve relevant papers.
  File: `Search query/SearchReport_Publish_or_perish`
- **Raw Data**
  Includes the unfiltered result set from the literature search.
  File: `Raw data/raw_data.csv`
- **Inclusion/Exclusion Filtering**
  Documents how the raw dataset was filtered using predefined criteria.
  File: `Filtered data/Inclusion_exclusion criteria.csv`
  Output: `Filtered data/filtered_papers.csv`
- **Data Extraction Form**
  Contains the structured form used to extract relevant data from each paper.
  File: `forms/data_extraction_form.csv`
- **Qualitative Analysis**
  Describes the import of filtered papers into MAXQDA and the coding process applied for thematic analysis.
  (Note: MAXQDA project file not included due to proprietary format.)
- **Results**
  Includes summary tables and visualizations generated from the extracted data and coding results.
  Folder: `Results/`

B.3 Replication Workflow Summary

1. **Download the ZIP file of the reproducible package** and extract it to a desired location. (Download)
2. **Run the Search Query** using `Publish or Perish` with the provided search report file.
3. **Filter the data** using the inclusion/exclusion criteria and save the filtered dataset.
4. **Extract relevant data** from the included papers using the provided data extraction form.
5. **Perform qualitative analysis** by importing the filtered papers into MAXQDA and coding them.
6. **Compile results** into summary tables and visualizations, saved in the `Results/` folder.

This package ensures full transparency of the literature review process and enables others to replicate or extend the analysis presented in this thesis.

References

Akkaoui, Z. E., Zimányi, E., & Mazón, J. N. (2011). Framework for etl process. *ACM*. https://dl.acm.org/doi/abs/10.1145/2064676.2064685

Almeida, J. R., Coelho, L., & Oliveira, J. L. (2021). Bicenter. *ScienceDirect*. https://www.sciencedirect.com/science/article/pii/S2352711021001497

Al-Rahman, S. Q. A., & Hasan, E. H. (2014). Web-based etl tool. *Semantic Scholar*. https://pdfs.semanticscholar.org/055f/187912653380de276fc3e2ed66a8c087c2c1.pdf

Bansal, S. K. (2014). Semantic etl framework for big data integration. *IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/6906824/

Belo, O., Gomes, C., Oliveira, B., & Marques, R. (2016). Domain-specific language to enrich etl schemas. *Springer*. https://link.springer.com/chapter/10.1007/978-3-319-23201-0_4

Chen, Z., & Zhao, T. (2012). A new tool for etl process. *IEEE Explore*. https://ieeexplore.ieee.org/abstract/document/6425038/

da Silva, M. S., & Times, V. C. (2018). Framework for etl systems development. *JIDM*. https://journals-sol.sbc.org.br/index.php/jidm/article/view/1455

Debroy, V., Brimble, L., & Yost, M. (2018). Newtl. *ACM Digital Library*. https://dl.acm.org/doi/abs/10.1145/3167132.3167300

Fowler, M. (2010). *Domain-specific languages*. Addison-Wesley Professional. https://martinfowler.com/books/dsl.html

Fusch, P. I., & Ness, L. R. (2015). Are we there yet? data saturation in qualitative research. *The Qualitative Report*, *20*(9), 1408–1416.

Guba, E. G., & Lincoln, Y. S. (1985). *Naturalistic inquiry*. SAGE Publications.

Kitchenham, B. A., & Charters, S. (2004). *Guidelines for performing systematic literature reviews in software engineering*. Software Engineering Group, Keele University.

Lakhan, A., & Singh, R. (2021). Implementation of etl tool for data warehousing. *ResearchGate*. https://www.researchgate.net/profile/Anwar-Sathio/publication/353212846_Implementation_of_ETL_Tool_for_Data_warehousing_for_Non-Hodgkin_Lymphoma_NHL_Cancer_in_Public_Sector_Pakistan/links/6106fd001ca20f6f86f239cd/Implementation-of-ETL-Tool-for-Data-warehousing-for-Non-Hodgkin-

Lymphoma - NHL - Cancer - in - Public - Sector - Pakistan . pdf ? origin = journalDetail&_tp=eyJwYWdlIjoiam91cm5hbERldGFpbCJ9

Li, J., Kuang, B., & Liu, J. (2016). Script-based automation etl tool. *Atlantis Press*. https://www.atlantis-press.com/proceedings/meici-16/25863795

Mukherjee, R., & Kar, P. (2017). A comparative review of data warehousing etl tools with new trends and industry insight [Cited by 72 in Google Scholar: https://scholar.google.com/scholar?cites=12779357461268076662as$_sdt$ = 2005$sciodt$ = 0, 5hl = en]. *2017 IEEE 7th International Advance*. https://ieeexplore.ieee.org/abstract/document/7976926/

Nath, R. P. D., Hose, K., & Pedersen, T. B. (2015). Programmable semantic etl framework for semantic data warehouses. *ACM Digital Library*. https://dl.acm.org/doi/abs/10.1145/2811222.2811229

Novak, M., & Rabuzin, K. (2010). Web etl tool. *CiteSeerX*. https : / / citeseerx . ist . psu . edu / document ? repid = rep1 & type = pdf & doi = fed403faa4bd9871347f617d32f18dd3f892bf13#page=110

Nwokeji, J. C., & Matovu, R. (2021). A systematic literature review on big data extraction, transformation and loading (etl) [Cited by 24 in Google Scholar: https://scholar.google.com/scholar?cites=14124771777261562 25as$_sdt$ = 2005$sciodt$ = 0, 5hl = en]. In *Intelligent computing: Proceedings of the 2021*. Springer. https://link.springer.com/chapter/10.1007/978-3-030-80126-7_24

Oliveira, B., & Belo, O. (2016). A domain-specific language for etl. *Springer*. https://link.springer.com/chapter/10.1007/978-3-319-23485-4_60

Oliveira, R., & Ramos, J. (2015). Etl^2. *IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/7218021/

Patel, M., & Patel, D. B. (2020). Progressive growth of etl tools: A literature review of past to equip future [Cited by 20 in Google Scholar: https://scholar.google.com/scholar?cites=3752107420142125187as$_sdt$ = 2005$sciodt$ = 0, 5hl = en]. In *Rising threats in expert applications and solutions*. Springer. https://link.springer.com/chapter/10.1007/978-981-15-6014-9_45

Popović, A., Ivković, V., Trajković, N., & Luković, I. (2023). Domain-specific language for managing etl processes. *PeerJ Computer Science*. https://peerj.com/articles/cs-1835/

Quiroz, J. C., Chard, T., Sa, Z., Ritchie, A., & Jorm, L. (2022). Etl for the conversion of health databases to omop. *PLoS One*. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0266911

Radonić, M., & Mekterović, I. (2017). Etlator. *IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/7973632/

Shah, N., & Sawant, V. (2015). Informatica. *CiteSeerX*. https : / / citeseerx . ist . psu . edu / document ? repid = rep1 & type = pdf & doi = 8874676f528fad9be5e8d387926b718767ebf326

Shah, S. S., Koranne, V. V., & Ahmed, K. N. A. (2024). Bifrost. *IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/10511886/

Sirsikar, S., Jain, A., Raina, D., & Munot, R. (2015). Database agnostic etl tool. *NCRDBN*. http://researchpublications.org/NCRDBN-15/NCRDBN_21.pdf

Souissi, S., & BenAyed, M. (2017). Genus. *IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/7945615/

Thomsen, C., & Pedersen, T. B. (2009). Pygrametl. *ACM Digital Library*. https://dl.acm.org/doi/abs/10.1145/1651291.1651301

Vijayendra, N., & Lu, M. (2013). Web-based etl tool for data integration process. *IEEE Explore*. https://ieeexplore.ieee.org/abstract/document/6577861/

Wang, J., & Liu, B. (2020). Etl tool for structured data based on data warehouse. *ACM*. https://dl.acm.org/doi/abs/10.1145/3424978.3425101

Xiao, F., Li, C., Wu, Z., & Wu, Y. (2018). Nmstream. *ISPRS Annals*. https://isprs-annals.copernicus.org/articles/IV-4/243/2018/

Yangui, R., Nabli, A., & Gargouri, F. (2017). Etl nosql. In *Springer*. https://link.springer.com/chapter/10.1007/978-3-319-65930-5_4

Ying-lan, F., & Bing, H. (2009). Implementation of etl tool. *IEEE Explore*. https://ieeexplore.ieee.org/abstract/document/5362117/