International Journal of Engineering Research in Big Data Systems (IJERBDS)
ISSN (Online): 3144-5179
Volume- 3 Issue-1 || Jan – Mar 2026 || PP. 1 – 5
https://ijerbds.org/

Self-Configuring ETL Pipelines Using AI-Generated DAG Structures

DOI: https://doi.org/10.63345/ijerbds.v3.i1.301

Justin Rajakumar Maria Thason

BCBSA - Ajace, Inc

AI Thought Leadership, IEEE Leadership & Enterprise AI Research

http://www.ijerbds.org/ || Vol. 3 No. 1 (2026): March Issue

Date of Submission: 04-02-2026	Date of Acceptance: 16-02-2026	Date of Publication: 03-03-2026

Abstract : ETL pipelines are manually created, brittle in case of schema evolution of sources, and require significant effort on the part of data engineers. In this study, we propose a system that automatically configures ETL pipelines: the profiler infers schema and statistics from sources, the semantic mapper finds correspondences of the schema to a target model, the AI planner generates the pipeline as a directed acyclic graph (DAG) in Apache Airflow format, and the sandbox runs the DAG with data contracts in order to validate its correctness before deployment. Among 90 integration scenarios at three complexity levels, our system achieved the correct output in 77 cases (85.6%), compared with 51 (56.7%) for the template generator and 49 (54.4%) for the zero-shot language model ($p < 0.001$). The average time to deploy a pipeline decreased from 187 minutes in manually built pipelines to 11.8 minutes. In case of introducing 60 schema-drift events, 47 of them (78.3%) were handled correctly, compared with 12 (20.0%) for the template generator ($p < 0.001$), and the mean recovery time decreased from 142 to 6.4 minutes. The combination of profiling, validation, and repair rather than just generation made it possible to achieve a reliable self-configuration.

Keywords: ETL, data pipelines, DAG generation, Apache Airflow, schema drift, data contracts, large language models, data integration

Introduction

Companies ingest data from various operational systems, files, and external services into data warehouses and data lakes. Each data integration usually requires an engineer to look at a source, understand how its fields are related to the target model, create extraction and transformation logic, schedule execution, and define quality checks. This is a tedious task, yet a complicated one because of the brittleness of its outputs: renaming a column or changing a data type in an upstream system might cause failure of downstream jobs without any notification. The workflow engines like Apache Airflow use DAGs of interdependent tasks to represent pipelines and make their structure explicit and machine-readable. This makes it possible to generate the structure of a pipeline, its configuration, and its tests automatically: given a source, a target, and some constraints, an appropriate planner can create the DAG. Large language models are a good fit for this kind of task due to the fact that they can interpret documentation, field names, and sample

@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijerbds.org

International Journal of Engineering Research in Big Data Systems (IJERBDS)
ISSN (Online): 3144-5179
Volume- 3 Issue-1 || Jan – Mar 2026 || PP. 1 – 5
https://ijerbds.org/

values. However, they cannot guarantee the correctness of the generated pipeline by themselves. In this study, we investigate whether the combination of profiling, AI planning, and validation can produce a correctly working pipeline. We provide (i) system architecture, (ii) a benchmark comprising 90 integration scenarios and 60 schema-drift events, and (iii) analysis that helps to discover the factors that affect the performance and reliability of the process.

Literature Review

The ETL processes have been studied by Vassiliadis (2009) who described the stages and design issues of extraction, transformation, and loading, while the problems associated with data cleaning were analyzed by Rahm and Do (2000). Kleppmann (2017) places batch and stream pipelines in a bigger picture of data-intensive applications, pointing out the problem of schema evolution. The analysis of the evolution of the Wikipedia's database schema showed the frequent occurrence of changes and necessity of corresponding adaptation of applications and queries, therefore, making the problem of graceful schema evolution an important one (Curino et al., 2008).

The automatic data integration relies on the schema matching, where Rahm and Bernstein (2001) classify the approaches by names, structures, instances, and context, and the integration of machine learning and integration tasks (Dong & Rekatsinas, 2018). Doan et al. (2012) provide an overview of data integration principles. Recently, Narayan et al. (2022) showed that foundation models can execute data cleaning and integration tasks such as entity matching and imputation even without examples.

Data quality assurance moved towards declarative checks. Schelter et al. (2018) describe the automation of data quality verification at scale via constraints and metrics, and Breck et al. (2019) analyze data validation for machine learning, which detects anomalies and drift by evaluating schemas and distributions on ingestion. Polyzotis et al. (2018) survey the data lifecycle challenges in production machine learning, including pipeline fragility. The generation of code by language models is evaluated via the execution of the generated code against tests (Chen et al., 2021), and agents that integrate reasoning with action are able to use feedback for repairing errors (Yao et al., 2023).

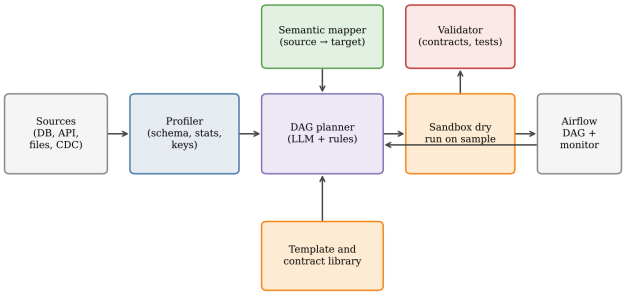
There has been little integration of different strands to produce automatically generated and validated pipeline definitions, nor much research into the problem of schema drift in operational setting. The present study addresses this gap.

Research Methodology

System design

Figure 1 shows the system architecture. Profiler takes a source and calculates column types, null rate, distinct counts, value patterns, candidate keys, and, in the case of relational databases, foreign-key relationships. The semantic mapper creates the mappings of source fields to target ones based on name similarity, value statistics, and interpretation of documentation by the language model, assigns confidence score. Planner uses profile, mapping, and data contract of the target to generate an Airflow DAG: extraction tasks, a staging stage, validation tasks, transformations (typing, deduplication, joins, slowly changing dimension type 2 management, change data capture merges), and loading tasks with idempotent writes. Planning uses the repository of validated task templates, which the model parameterizes and connects to, rather than generates arbitrary code. Sandbox runs the DAG on a sample, and the validator evaluates row counts, satisfaction of constraints, key uniqueness, and distributional tests. Failures are reported to planner to perform up to three repair attempts, and mappings with low confidence score are marked for human review.

Figure 1. System for self-configuration of ETL pipelines: profiling, semantic mapping, AI DAG planning, sandbox validation, and monitored deployment.



Benchmark

@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijerbds.org

International Journal of Engineering Research in Big Data Systems (IJERBDS)
ISSN (Online): 3144-5179
Volume- 3 Issue-1 || Jan – Mar 2026 || PP. 1 – 5
https://ijerbds.org/

The benchmark includes 90 integration scenarios, with 30 examples at each of three complexity levels: simple (with one or two sources and direct mapping), medium (with three to five sources and joins and type conversion), and complex (with six to ten sources and change data capture and type 2 dimensions). Each scenario includes the reference pipeline created by an experienced engineer and a reference output. Four approaches were evaluated: hand-built reference (for time measurement only), the template generator that fills fixed pipeline templates using the profile, the zero-shot language model that generates DAG from the description, and the proposed system. Correctness was achieved if the pipeline was executed successfully and its output was correct with respect to the reference in terms of row count, key set, and checksums of columns.

Schema drift and analysis

60 events of schema drift were introduced to the running pipelines, with 15 instances in each of four categories: column addition, column renaming, column type modification, and column removal. The event was counted as non-failing if the pipeline continued delivering the correct output after the adaptation or quarantine of change. The proportions of correctness were compared using two-proportion z-test and chi-square test, times of building the pipelines were compared using one-way ANOVA and Tukey's test, while times of recovering were compared using Welch's t-test.

Results

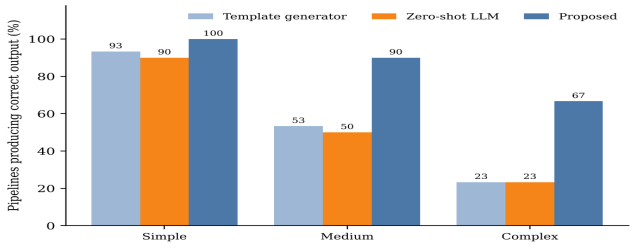
Correctness

Table 1. Pipelines achieving correct output by complexity (30 per level)

Approach	Simple	Medium	Complex	Overall (%)	p vs proposed
Template generator	28	16	7	56.7	<0.001
Zero-shot LLM	27	15	7	54.4	<0.001
Proposed	30	27	20	85.6	—

The correctness differed significantly among the three automated approaches ($\chi^2(2) = 24.0$, $p < 0.001$). The proposed system reached correctness in 85.6% of scenarios, outperforming the template generator by 28.9 percentage points and the zero-shot language model by 31.1 points. All approaches were performing well in simple scenarios, but with complexity the performance of baselines declined dramatically (Figure 2): in complex scenarios, the template generator and the zero-shot language model reached the correct output in only 7 of 30 cases each, while the proposed system achieved success in 20. Failures of the zero-shot language model were caused mostly by incorrect join keys and mishandling of type 2 history, while failures of the proposed system were caused by the ambiguity of the mappings, requiring human review.

Figure 2. Percentage of scenarios resulting in correct output on different complexity levels and approaches.



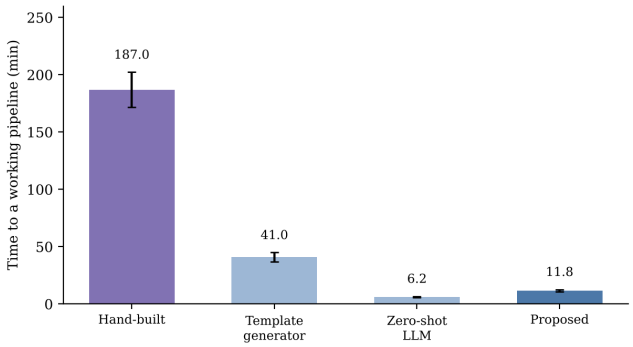
Time to build a pipeline

The time required to build a working pipeline differed depending on the approach ($F(3, 356) = 444$, $p < 0.001$; Figure 3). Hand-built pipelines took 187 ± 74 minutes, templates – 41 ± 19, zero-shot language model – 6.2 ± 2.8, and proposed system – 11.8 ± 4.6 minutes, resulting in 94% decrease in time relative to manual approach. The time for proposed system was worse comparing to zero-shot model due to profiling, validation, and repair ($p < 0.001$, Welch's test), but it resulted in higher correctness mentioned above. The time for proposed system included the time for human review of mappings marked for review, which happened in 19% of cases.

Figure 3. Time to a working pipeline by approaches (error bars show 95% confidence intervals).

OPEN ACCESS
@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijerbds.org

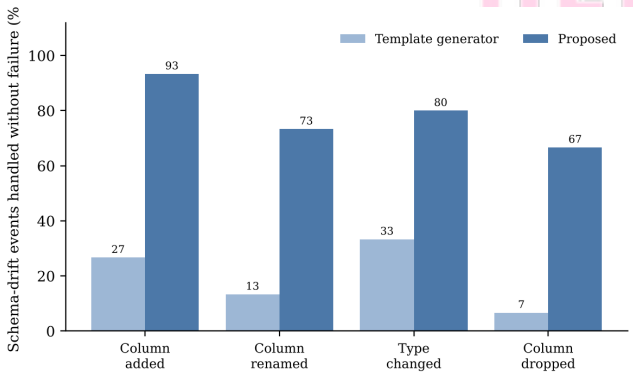
International Journal of Engineering Research in Big Data Systems (IJERBDS)
ISSN (Online): 3144-5179
Volume- 3 Issue-1 || Jan – Mar 2026 || PP. 1 – 5
https://ijerbds.org/



Resilience to schema drift

The proposed system handled 47 out of 60 drift events (78.3%), compared with 12 events (20.0%) for the template generator ($p < 0.001$; Figure 4). The most easily handled events were the additions of columns (14 out of 15), since they could be quarantined or propagated using the type inferred from the profile, while the least tractable were dropped columns (10 out of 15), since downstream dependency could not be fulfilled without a business decision regarding default values. The mean time to recover from a failure decreased from 142 ± 58 minutes with manual repair to 6.4 ± 3.1 minutes with automated adaptation (Welch $t = 18.1$, $p < 0.001$).

Figure 4. Number of schema-drift events handled without failure by types of change.



Discussion

The obtained results show that generation, validation, and structural considerations each contribute to system performance. Zero-shot language model generated plausible DAGs quickly, but without checks against source data, it misjudged the keys and history handling of type 2 dimensions – consistent with the results showing that models perform excellently on well-specified tasks, but they benefit from execution feedback when correctness depends on data (Chen et al., 2021; Yao et al., 2023). The profiler provided the missing information like uniqueness and null rates, while the library of templates prevented the planner from generating the arbitrary code.

The drift results confirm the fact that schema evolution occurs quite often and that systems need to adapt to changes (Curino et al., 2008). The automated validation at ingestion as suggested by Breck et al. (2019) and Schelter et al. in 2018 was used in this study, and changes were detected prior to being propagated to downstream tables. The ability of planner to regenerate the task from the updated profile contributed to faster recovery times.

The difficulty of handling dropped columns points to a more general issue: certain changes are semantic and require human business decision, which cannot be made by any planner. This is why the system escalates such cases to human review, which is a design choice, rather than an error.

For companies, the most promising near-term application is not full autonomy but acceleration of the process of operation under governance. Simple and medium-complexity integrations can be automated and then reviewed, freeing the engineers to solve the complex integrations, while the confidence score of mapping will help to direct attention to uncertain fields. This recognizes the limits of automated schema matching, which has been known for a long time to be inherently ambiguous (Rahm & Bernstein, 2001; Doan et al., 2012).

Limitations

The benchmarking used the synthetic and anonymized sources based on common patterns of enterprises; real-world systems are more irregular. The reference pipelines were generated by a single engineer, so the concept of correct output reflects his design choices. The study covers only the batch pipelines on Apache Airflow; streaming pipelines, very large tables, and data sensitivity handling were not evaluated. The behavior of language model differs depending on the version, while

@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijerbds.org

International Journal of Engineering Research in Big Data Systems (IJERBDS)
ISSN (Online): 3144-5179
Volume- 3 Issue-1 || Jan – Mar 2026 || PP. 1 – 5
https://ijerbds.org/

the costs of model calls were not measured. The drift events were generated artificially and may not reproduce the actual occurrence.

Conclusion

The ETL system, which profiles sources, plans a directed acyclic graph (DAG) using the language model constrained to the library of templates, and validates and repairs it using sandbox, reached correct pipelines for 85.6% of 90 scenarios, reduced the time to the deployment by 94% compared to the manual creation, and handled 78.3% of schema-drift events without failure. Thus, the automation seems to be applicable for most integration tasks when combined with validation and human review for ambiguous decisions. Future work needs to cover streaming pipelines, learned cost-based optimization of the generated DAGs, and richer data-contract languages. Proceedings of the ACM SIGMOD International Conference on Management of Data, 1645–1650.

References

1. Kleppmann, M. (2017). Designing data-intensive applications. O'Reilly Media.
2. Narayan, A., Chami, I., Orr, L., Arora, S., & Ré, C. (2022). Can foundation models wrangle your data? Proceedings of the VLDB Endowment, 16(4), 738–746.
3. Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2018). Data lifecycle challenges in production machine learning: A survey. ACM SIGMOD Record, 47(2), 17–28.
4. Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. The VLDB Journal, 10(4), 334–350.
5. Rahm, E., & Do, H. H. (2000). Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin, 23(4), 3–13.
6. Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2018). Automating large-scale data quality verification. Proceedings of the VLDB Endowment, 11(12), 1781–1794.
7. Vassiliadis, P. (2009). A survey of extract–transform–load technology. International Journal of Data Warehousing and Mining, 5(3), 1–27.
8. Yao, S., Zhao, J., Yu, D., et al. (2023). ReAct: Synergizing reasoning and acting in language models. International Conference on Learning Representations.

@2026. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijerbds.org