Unify: A System For Unstructured Data Analytics

Jiayi Wang
Tsinghua University
jiayi-wa20@mails.tsinghua.edu.cn

Yuan Li
Tsinghua University
2453942388li@gmail.com

Jianming Wu
Tsinghua University
wjmwjmwb@yeah.net

Shihui Xu
Tsinghua University
xsh23@mails.tsinghua.edu.cn

Guoliang Li
Tsinghua University
liguoliang@tsinghua.edu.cn

ABSTRACT

Unstructured data comprises over 80% of today's information, yet no specialized system effectively supports its semantic analytics. Traditional SQL-based approaches rely on predefined schemas, making them unsuitable. While large language models (LLMs) enable semantic analysis of unstructured data, manually orchestrating execution plans remains inefficient. This raises a critical question: *how can we automate unstructured data analytics?* In this demonstration, we present `Unify`, a system that automates unstructured data analytics for natural language queries. `Unify` defines a set of core operators for unstructured data processing, with both preprogrammed and LLM-based implementations. It guides LLMs to decompose queries into logical steps and map them to appropriate operators for accurate execution. Our demonstration showcases `Unify` by real-world scenarios, highlighting its ability to bridge the gap between unstructured data and actionable analytics.

PVLDB Reference Format:
Jiayi Wang, Yuan Li, Jianming Wu, Shihui Xu, and Guoliang Li. Unify: A System For Unstructured Data Analytics. PVLDB, 18(12): 5287 - 5290, 2025.
doi:10.14778/3750601.3750653

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/TsinghuaDatabaseGroup/Unify.

1 INTRODUCTION

Organizations across diverse domains generate vast amounts of unstructured data daily, holding immense potential for insights. However, the lack of predefined schemas makes traditional analytics methods ineffective — structured query languages like SQL are unsuitable, and manual analysis is both time-consuming and error-prone [4, 11–13]. Prior systems [5, 6] for unstructured data analytics either rely on manual query planning or are limited to simple retrieval tasks [3, 8].

To address these challenges, we present `Unify` [7, 9], an end-to-end system that automates unstructured data analytics by integrating the semantic understanding capabilities of LLMs with efficient query planning and execution strategies. Unlike existing systems, `Unify` *automates the entire analytics pipeline, from natural language query understanding to plan generation, optimization, and execution, without requiring predefined schemas or user intervention.*

At its core, `Unify` offers the following key functionalities:
Index Construction. `Unify` employs a set of predefined operators commonly used in unstructured data analytics to orchestrate query plans. To map natural language queries to these operators, `Unify` builds embedding indexes over the usage of the operators described in natural language. For unstructured data, sentences are transformed into embedding vectors and indexed using structures such as hierarchical navigable small world (HNSW) to enable efficient retrieval.
Logical Plan Generation. `Unify` decomposes a natural language query iteratively by matching it to a set of predefined semantic operators, ultimately constructing a plan composed of these operators. By carefully selecting and combining these operators, `Unify` ensures that the generated plan accurately represents the user's intent, and can be executed to produce the correct answer.
Physical Plan Optimization: `Unify` employs a cost model and a semantic cardinality estimation method to convert logical plans into efficient physical execution plans, improving execution efficiency.
Interactive Execution: To compute the final answer, `Unify` executes operators of the plan in parallel when possible and dynamically adjusts the plan at runtime based on intermediate results.

`Unify` stands out for its unique advantages:
1. User-Friendly Interaction. `Unify` provides a natural language interface, enabling non-expert users to interact with `Unify` easily.
2. High Accuracy. By iteratively matching appropriate operators to solve partial tasks, the logical plan ensures high accuracy.
3. High Efficiency. By leveraging optimized physical plans and parallel execution, `Unify` significantly reduces query processing time.

In summary, `Unify` is the first end-to-end system that automates efficient unstructured data analytics for natural language queries, delivering both efficiency and accuracy for real-world applications.

2 SYSTEM OVERVIEW

As shown in Figure 1, `Unify` consists of the following modules: operator management, index construction, logical plan generation, physical plan optimization and interactive plan execution.
Operators. Traditional relational operators are inadequate for unstructured data analytics due to their lack of semantic processing capabilities. To address this, as shown in Table 1, `Unify` defines a set of common operators, including `Filter`, `Extract`, `GroupBy`, and `Compare`, each with well-defined input-output specifications and multiple execution implementations.

To enable accurate operator-query matching, `Unify` indexes operators by the **logical representations** of their natural language

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750653

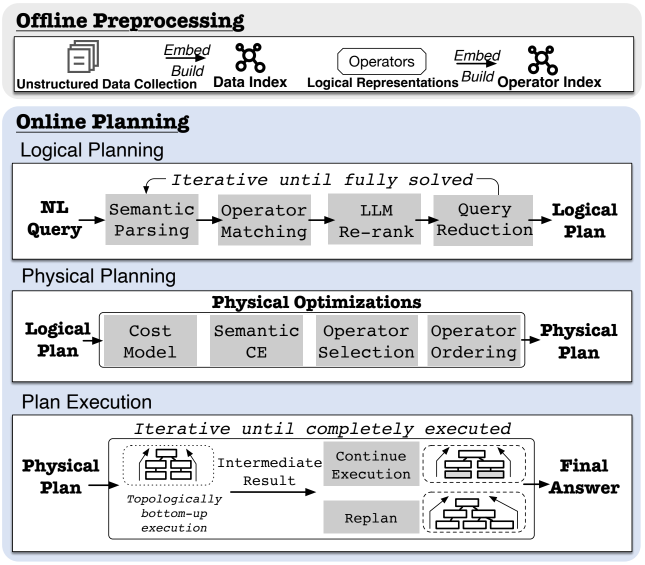


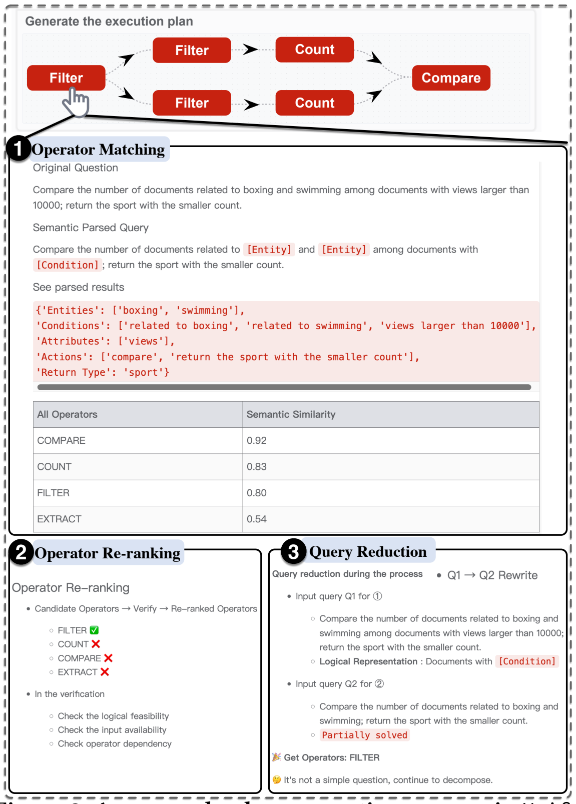
Figure 1: Architecture of Unify.

Table 1: The logical operators, their inputs, outputs, and example logical representations.

Operator	Input	Output	Example Logical Representation
Scan	List	List	documents satisfy [Condition]
Filter	List	List	[Entity] that [Condition]
Compare	A, B, Condition	A/B	larger in [Entity] and [Entity]
GroupBy	List	List of List	aggregate [Entity] by [Attribute]
Count	List	Number	number of documents [Condition]
Sum	List	Number	the total sum of [Entity]
Max	List	Number	the maximum of [Entity]
Min	List	Number	the minimum of [Entity]
Average	List	Number	the mean of [Entity]
Median	List	Number	the median of [Entity]
Percentile	List	Number	the k-th percentile for [Entity]
OrderBy	List	List	Sort [Entity] [Condition]
Classify	Text	Class	The type of [Entity]
Extract	Text	Text	get [Entity] from documents
TopK	List	List	the top [Number] [Entity]
Join	List, List	List	[Entity] that also occurs in [Entity]
Union	Set, Set	Set	set union of [Entity] and [Entity]
Intersection	Set, Set	Set	in set [Entity] and in [Entity]
Complementary	Set, Set	Set	in set [Entity] not in [Entity]
Compute	List	Number	sum of squares of [Entity]
Generate	Text	Text	explain the result

(NL) functionality descriptions. A logical representation is *a structured template that abstracts semantic elements into placeholders, such as Entity and Condition, each representing distinct semantic roles*. For instance, the Filter operator can be represented as *"[Entity] that [Condition]"*, which can match with queries like *"movies that were made in 1990s"*.

Each logical operator can be realized through one or more physical implementations, which are categorized into two types: (1) **Pre-programmed Implementations**, which leverage predefined algorithms similar to traditional database operators, and (2) **LLM-based Implementations**, which employ LLMs for tasks requiring deeper semantic reasoning. For example, the Filter operator can be implemented either by keyword-matching functions or by prompting an LLM to evaluate whether a document satisfies the filtering criteria. Most logical operators support both types, allowing Unify to balance efficiency and semantic comprehension.



Original Question

Compare the number of documents related to boxing and swimming among documents with views larger than 10000; return the sport with the smaller count.

Semantic Parsed Query

Compare the number of documents related to [Entity] and [Entity] among documents with [Condition]; return the sport with the smaller count.

See parsed results

```
{'Entities': ['boxing', 'swimming'],
'Conditions': ['related to boxing', 'related to swimming', 'views larger than 10000'],
'Attributes': ['views'],
'Actions': ['compare', 'return the sport with the smaller count'],
'Return Type': 'sport'}
```

All Operators	Semantic Similarity
COMPARE	0.92
COUNT	0.83
FILTER	0.80
EXTRACT	0.54

Operator Re–ranking

- Candidate Operators → Verify → Re–ranked Operators
  - FILTER ✅
  - COUNT ❌
  - COMPARE ❌
  - EXTRACT ❌
- In the verification
  - Check the logical feasibility
  - Check the input availability
  - Check operator dependency

Query reduction during the process • Q1 → Q2 Rewrite

- Input query Q1 for ①
  - Compare the number of documents related to boxing and swimming among documents with views larger than 10000; return the sport with the smaller count.
  - **Logical Representation** : Documents with [Condition]
- Input query Q2 for ②
  - Compare the number of documents related to boxing and swimming; return the sport with the smaller count.
  - Partially solved

🎉 **Get Operators: FILTER**

🤔 It's not a simple question, continue to decompose.

Figure 2: An example plan generation process in Unify.

If existing operators are insufficient, users can extend Unify by defining custom logical representations for planning and corresponding physical implementations for execution.

Index Construction. Before query processing, Unify conducts offline preprocessing to organize operators and data for efficient online planning and execution (Figure 1): Unify first employs a text embedding model to convert unstructured data into vector representations. These vectors are organized using a vector index, enabling efficient retrieval of required data during query execution. The logical representations of operators are also computed and indexed by their semantic embeddings for efficient matching.

Logical Plan Generation. Upon receiving an NL query, Unify decomposes it into smaller, manageable sub-queries by iteratively matching segments of the query with the predefined operators. As illustrated in Figure 2, this involves:

Operator Matching. The first step in logical planning is selecting the most relevant operator for a segment of the query. Relying solely on LLMs for operator selection using prompts can be inefficient and error-prone due to the large LLM token consumption and the large number of possible operators [10]. To mitigate this, Unify minimizes LLM involvement, using it only when necessary.

The main challenge in operator matching is to accurately interpret the intent of the query. To address this, Unify converts the query into a *logical representation* by replacing concrete values with semantic elements like Condition, Entity, through a few-shot prompt. This abstraction reduces the query to its logical essence independent of specific values, thus enabling a clearer focus on its logical structure. For example, in step ❶ of Figure 2, the original question *"Compare the number of documents related to boxing and swimming among documents with views larger than 10000; return the sport with the smaller count"* can be parsed as *"Compare the number of documents related to [Entity] and [Entity] among documents with [Condition]; return the sport with the smaller count"*.

Our **key insight** is that an operator is relevant if its logical representation closely matches the query's. Based on this, we leverage semantic similarity as a coarse-grained metric, comparing the query's logical representation embeddings with those of the operators and only selecting the operators with the highest similarities. This approach significantly improves both accuracy and efficiency by rapidly eliminating operators that are unlikely to solve any part of the query. For example, in step ❶ of Figure 2, after selecting only the operators with highest similarities, the number of candidate operators is greatly reduced.

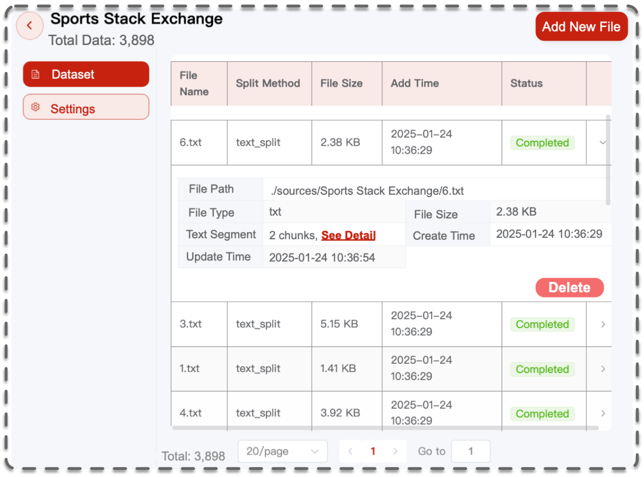
Operator Re-ranking. After identifying candidate operators, Unify employs the LLM to re-rank them based on their applicability to the query. Specifically, Unify maintains text descriptions of processed data, and only those operators whose inputs match these available variables are considered. In this phase, the LLM evaluates each operator's applicability by determining whether its inputs are available and whether it can address part of the query, categorizing operators as fully solving, partially solving, or not applicable. Operators are then ranked primarily by their ability to resolve the query, with semantic similarity serving as a secondary consideration. For example, in step ❷ of Figure 2, after re-ranking, the Filter operator is prioritized despite not having the highest semantic similarity. This is because it is the only operator that can be directly applied to the raw data (*i.e.,* filtering by *views larger than 10,000*).

Query Reduction. Once an appropriate operator is selected, the LLM applies it to reduce the matched segment of the query. For example, in step ❸ of Figure 2, the original query is first reduced by removing "*with views larger than 10000*" and transformed into *"Compare the number of documents related to boxing and swimming; return the sport with the smaller count."*, applying the Filter operator. Instead of relying on rigid keyword-based matching and replacement, the reduction is conducted flexibly by the LLM with a few-shot prompt that incorporates placeholders for the query, the operator information, and the expected output of the operator. After each reduction step, the LLM verifies whether the query has been fully reduced to determine if further reduction is needed.

Iterative Query Decomposition. The above process is repeated until the query is fully decomposed. In each iteration, Unify selects an operator, verifies its applicability, and applies it to reduce the query further. The resulting plan is represented as a directed acyclic graph (DAG), where nodes correspond to selected operators and edges represent dependencies between operators.

Physical Plan Optimization. Once a logical plan is generated, Unify converts it into an optimized physical plan by selecting the most efficient physical implementation for each operator and determining the optimal operator execution order. This relies on a semantic cost model and a semantic cardinality estimation method to estimate execution time and intermediate result sizes [9].

Cost Model. Unlike relational databases, where operator costs are dominated by I/O factors, Unify focuses on computational costs due to the frequent involvement of resource-intensive LLMs. To



Sports Stack Exchange
Total Data: 3,898

Add New File

Dataset
Settings

File Name	Split Method	File Size	Add Time	Status
6.txt	text_split	2.38 KB	2025–01–24 10:36:29	Completed

File Path	./sources/Sports Stack Exchange/6.txt		
File Type	txt	File Size	2.38 KB
Text Segment	2 chunks, See Detail	Create Time	2025-01-24 10:36:29
Update Time	2025-01-24 10:36:54		

Delete

File Name	Split Method	File Size	Add Time	Status
3.txt	text_split	5.15 KB	2025–01–24 10:36:29	Completed
1.txt	text_split	1.41 KB	2025–01–24 10:36:29	Completed
4.txt	text_split	3.92 KB	2025–01–24 10:36:29	Completed

Total: 3,898 20/page 1 Go to 1

Figure 3: Unstructured data management in Unify.

facilitate plan optimization, Unify constructs a unified cost model that estimates the execution time of different physical operators.

Semantic Cardinality Estimation. Operator costs are directly determined by cardinality. Existing methods estimate cardinality for semantic predicates using uniform sampling [5], which can be inaccurate and may mislead optimization decisions. Instead, Unify employs an importance sampling approach that selects data points based on their embedding distance to the query [9]. By prioritizing samples with smaller embedding distances, Unify better captures relevant data that are more likely to satisfy the query, significantly improving the overall estimation accuracy.

Plan Execution. Unify efficiently executes the plan through:

Parallel Topological Execution. In a DAG-structured plan, independent operators are executed in parallel whenever possible. Execution follows the plan's topological order, ensuring that an operator runs once all dependencies are satisfied. For example, in Figure 2, after the first Filter operator, the two subsequent paths can proceed concurrently, as they have no interdependencies. This parallelism improves resource utilization and reduces execution time.

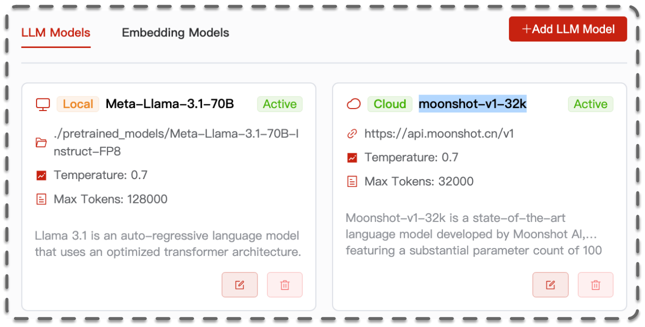
Dynamic Plan Adjustment. During execution, intermediate results are used to adapt the plan dynamically. For instance, if an operator fails to yield the expected output, Unify dynamically replans from the failed query to adjust the execution plan. This allows Unify to adapt the plan based on intermediate results throughout execution, thus obtaining higher robustness.

3 DEMONSTRATION SCENARIOS

Datasets. By default, Unify provides datasets from Wikipedia [2] and Stack Exchange [1].

Scenario. Consider a data analyst seeking to analyze social media data to identify trends and patterns, which are crucial for data-driven decision-making. Given the large volume and unstructured nature of web data, answering such queries directly is challenging due to the absence of structured metadata.

However, with Unify, users can perform automated data analytics easily through natural language queries. Prior to analytics, users can manage datasets within Unify (Figure 3), *e.g.,* modifying them by adding or removing entries related to Sports Stack Exchange. Additionally, Unify allows users to configure the LLM



Local Meta-Llama-3.1-70B Active
./pretrained_models/Meta-Llama-3.1-70B-Instruct-FP8
Temperature: 0.7
Max Tokens: 128000
Llama 3.1 is an auto-regressive language model that uses an optimized transformer architecture.

Cloud moonshot-v1-32k Active
https://api.moonshot.cn/v1
Temperature: 0.7
Max Tokens: 32000
Moonshot-v1-32k is a state-of-the-art language model developed by Moonshot AI,... featuring a substantial parameter count of 100

Figure 4: Model management in `Unify`.

and embedding model for semantic analytics, such as setting hyper-parameters such as temperature (Figure 4). The LLM can be locally deployed or cloud-based and the embedding model for semantic data representation is also customizable.

After configuration, users can submit natural language queries. For example, the data analyst may issue the query: *Compare the number of documents for boxing and swimming among those with more than 10,000 views; return the sport with fewer documents.* `Unify` generates a logical plan (Figure 2) that sequentially filters documents by view count, identifies boxing-related and swimming-related questions, counts occurrences for each sport, and compares the counts to identify the sport with fewer related documents.

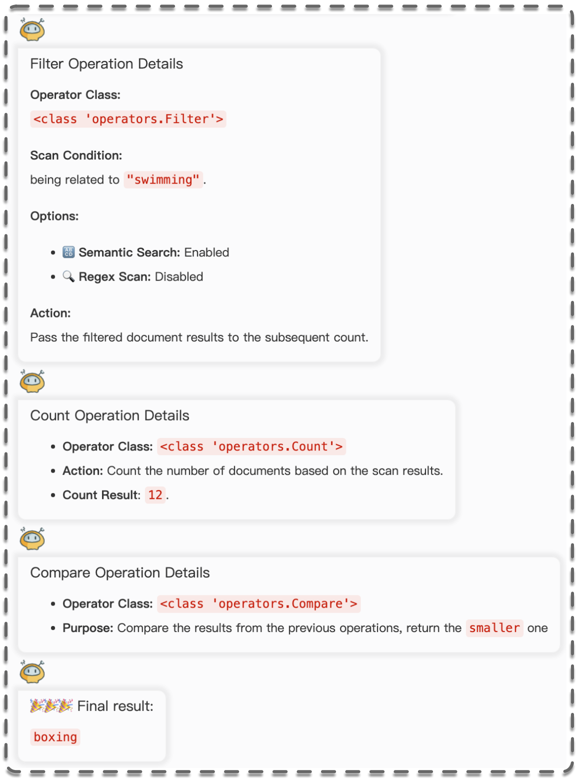
As shown in Figure 2, `Unify` visualizes the generated plan and provides an interactive interface to examine the plan orchestration process. Each operator in the plan is explained with details about how it is selected. For instance, the first `Filter` operator is selected via: (1) a coarse-grained *operator matching* step, where the logical representation of the query is first computed and the semantic similarity is computed based on the embedding distance between the logical representations of the query and operators, returning the top-4 relevant operators with the highest semantic similarities (step ❶); and (2) a fine-grained operator *re-ranking* step, where LLM verifies input availability and operator feasibility of the top-4 candidate operators, ultimately selecting `Filter` as the most appropriate operator (step ❷). The query is then progressively reduced with the selected operator (step ❸), and this iterative process continues until the query is fully decomposed into an executable plan.

During plan execution, `Unify` provides real-time visualizations. For example, Figure 5 illustrates the execution of the last few operators in Figure 2, displaying the executed physical implementations and intermediate results. These details allow users to verify correctness and understand the applied physical execution strategy.

In summary, `Unify` provides an intuitive interface and an automated framework for data analytics over unstructured datasets. It enables users to issue natural language queries, observe the process of transforming queries into executable plans, and track execution details to ensure accuracy in deriving insights.

4 CONCLUSION

In this paper, we propose `Unify`, which supports automatic unstructured data analytics, bridging the gap between complex unstructured data and actionable insights. By automating the entire analytics pipeline, `Unify` empowers users to derive meaningful insights from unstructured data with minimal technical expertise.



Filter Operation Details
Operator Class:
`<class 'operators.Filter'>`
Scan Condition:
being related to `"swimming"`.
Options:
- **Semantic Search:** Enabled
- **Regex Scan:** Disabled

Action:
Pass the filtered document results to the subsequent count.

Count Operation Details
- **Operator Class:** `<class 'operators.Count'>`
- **Action:** Count the number of documents based on the scan results.
- **Count Result:** `12`.

Compare Operation Details
- **Operator Class:** `<class 'operators.Compare'>`
- **Purpose:** Compare the results from the previous operations, return the `smaller` one

🎉🎉🎉 Final result:
`boxing`

Figure 5: A plan execution example of `Unify`.

ACKNOWLEDGMENTS

This paper was supported by National Key R&D Program of China (2023YFB4503600), NSF of China (62525202, 62232009), Shenzhen Project (CJGJZD20230724093403007), Zhongguancun Lab, and Beijing National Research Center for Information Science and Technology (BNRist). Guoliang Li is the corresponding author.

REFERENCES

[1] 2025. Stack Exchange Data Archive. https://archive.org/download/stackexchange. Last accessed: 2025-07-11.
[2] 2025. Wiki Data Archive. https://dumps.wikimedia.org.
[3] Zui Chen, Zihui Gu, Lei Cao, et al. 2023. Symphony: Towards Natural Language Query Answering over Multi-modal Data Lakes. In *CIDR*.
[4] Guoliang Li, Jiayi Wang, Chenyang Zhang, and Jiannan Wang. 2025. Data+AI: LLM4Data and Data4LLM. In *SIGMOD 2025*. ACM, 837–843.
[5] Chunwei Liu, Matthew Russo, Michael Cafarella, et al. 2025. Palimpzest: Optimizing AI-Powered Analytics with Declarative Query Processing. CIDR.
[6] Liana Patel, Siddharth Jha, Carlos Guestrin, and Matei Zaharia. 2024. LOTUS: Enabling Semantic Queries with LLMs Over Tables of Unstructured and Structured Data. *CoRR* (2024).
[7] Zhaoyan Sun, Jiayi Wang, Xinyang Zhao, Jiachi Wang, and Guoliang Li. 2025. Data Agent: A Holistic Architecture for Orchestrating Data+AI Ecosystems. *IEEE Data Eng. Bull.* 50, 1 (2025), 57–69.
[8] Matthias Urban and Carsten Binnig. 2024. CAESURA: Language Models as Multi-Modal Query Planners. In *CIDR*.
[9] Jiayi Wang and Jianhua Feng. 2025. Unify: An Unstructured Data Analytics System. ICDE.
[10] Jiayi Wang and Guoliang Li. 2025. AOP: Automated and Interactive LLM Pipeline Orchestration for Answering Complex Queries. *CIDR* (2025).
[11] Jiayi Wang, Guoliang Li, and Jianhua Feng. 2025. iDataLake: An LLM-Powered Analytics System on Data Lakes. *IEEE Data Eng. Bull.* 49, 1 (2025), 57–69.
[12] Xuanhe Zhou, Guoliang Li, Zhaoyan Sun, Zhiyuan Liu, Weize Chen, Jianming Wu, Jiesi Liu, Ruohang Feng, and Guoyang Zeng. 2024. D-Bot: Database Diagnosis System using Large Language Models. *VLDB* 17, 10 (2024), 2514–2527.
[13] Xuanhe Zhou, Zhaoyan Sun, and Guoliang Li. 2024. DB-GPT: Large Language Model Meets Database. *Data Sci. Eng.* 9, 1 (2024), 102–111.