AgenticData: An Agentic Data System on Heterogeneous Data

Peiyao Zhou[1], Ji Sun[1], Yaoqiang Xu[2], Chen Qian[2]
Tsinghua University[1], East Branch of State Grid Corporation of China[2]
22371389@buaa.edu.cn,sunji@tsinghua.edu.cn,xuyq99@139.com,arielqc@163.com

ABSTRACT

While natural-language interfaces have advanced significantly in structured data querying, analyzing heterogeneous data remains a challenge due to the diversity in data formats and processing methods. To tackle these challenges, we introduce AgenticData, an agentic data system that enables natural-language (NL) query analytics over heterogeneous data sources. AgenticData employs a data profiling agent to generate a semantic data catalog for heterogeneous data and implements both relational operators and semantic operators to handle such data effectively. For an NL query, AgenticData utilizes an agentic planner to identify relevant data sources and generate a semantic query plan composed of relational and semantic operators. To enhance the quality of the query plan, AgenticData incorporates a validator that provides feedback on semantic plan checks (e.g., grammar validity and semantic consistency). Additionally, AgenticData introduces an optimizer to improve execution efficiency by reducing the cost and latency associated with LLMs. To further enhance the system, AgenticData includes a memory manager that maintains historical feedback and insights, offering contextual guidance to the planner. We implemented AgenticData and demonstrated its effectiveness and efficiency using real-world benchmarks. Furthermore, we showcased AgenticData in two distinct scenarios, illustrating its ability to handle diverse data sources accurately and efficiently.

VLDB Workshop Reference Format:
Peiyao Zhou[1], Ji Sun[1], Yaoqiang Xu[2], Chen Qian[2]. AgenticData: An Agentic Data System on Heterogeneous Data. VLDB 2026 Workshop: ADS 2026: The Joint Workshop on Agentic Data Systems and Data-Centric AI (The 1st ADS & 3rd DATAI).

1 INTRODUCTION

Organizations today manage and process data in a growing variety of formats. While structured data is typically stored in relational databases and tabular files, a substantial amount of valuable information is found in unstructured documents, including reports, manuals, and online resources. In many real-world scenarios, gaining meaningful insights requires integrating and analyzing multiple heterogeneous data sources [4, 6, 7, 9, 11, 15, 16, 19, 21, 23–25]. However, performing analytics across such diverse sources is challenging due to differences in data organization, access methods, and processing techniques.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

Large Language Models (LLMs) have demonstrated exceptional capabilities in natural-language understanding, information extraction, and semantic reasoning. These capabilities enable systems to process vast collections of textual documents, offering new opportunities for analyzing unstructured data. Recent systems such as [5, 6, 11–13, 17] have leveraged LLMs to build analytical pipelines. However, these systems typically require users to write specific declarative programs to orchestrate data processing, which demands domain expertise. As a result, natural-language interfaces for data analytics are becoming increasingly popular.

Beyond NL2SQL systems [3, 8, 20], which allow users to query relational databases using natural language, recent work has extended natural-language interfaces to unstructured document collections [15, 16, 18, 19]. However, these systems are generally designed to work with a single data source or access interface. In practice, analysts often need to integrate and analyze data from multiple heterogeneous sources [14]. Supporting natural-language analytics across heterogeneous data sources presents several key challenges:

Challenge 1: How can relevant data be discovered and selected from a vast number of heterogeneous sources? Heterogeneous data sources vary significantly in representation. Relevant information is often distributed across multiple sources and may require preliminary synthesis to facilitate query planning. Identifying relevant data sources while filtering out irrelevant ones is non-trivial, particularly given the limited context capacity of LLMs.

Challenge 2: How can complex queries be understood and translated into a high-quality data analytics pipeline? Natural-language queries frequently describe complex analytical tasks involving multiple reasoning steps. It is challenging for an LLM to fully interpret the query, understand the capabilities of supported operators, and correctly orchestrate a data analytics pipeline.

Challenge 3: How can efficient execution be ensured for analytics over large-scale heterogeneous data, minimizing cost and latency? Analyzing heterogeneous data often involves computationally expensive operations, such as semantic operators. Striking a balance between reducing costs and latency while maintaining analytical accuracy is a significant challenge.

To address these challenges, we propose AgenticData, an agentic data analytics system over heterogeneous data. AgenticData includes a data profiling agent that interprets data sources and generates a semantic catalog of the heterogeneous data [19]. AgenticData designs an agentic planner that parses an NL query into a semantic data query plan. It discovers, understands, selects relevant data sources using the data profiles. It then comprehends the query, plans query steps, and generates a semantic query plan. To ensure accuracy, an agentic validator detects grammatical and semantic errors in the query plan before execution. A memory manager

maintains historical context by tracking previous errors and accumulating insights, providing guidance to avoid similar mistakes in the future. An optimizer is designed to reduce the costs and latency associated with semantic operations while ensuring efficient execution. AgenticData is designed as a self-evolving system that continuously improves its data understanding, planning strategies and optimization decisions through historical tasks and execution feedback.

In summary, we make the following contributions.
(1) We propose AgenticData, a feedback-driven, agentic data analytics system that supports natural-language queries over heterogeneous data sources and performs analytics automatically.
(2) We introduce a planning framework that incorporates semantic profiling, query plan validation, and memory management to ensure accurate query plan generation and automatic correction.
(3) We present query optimization techniques at both plan and operator levels, enabling efficient execution of semantic query plans.
(4) We demonstrate AgenticData's versatility and effectiveness through real-world scenarios including financial rule analysis and conflict-of-interest detection for paper review. These examples highlight AgenticData's ability to handle diverse application domains accurately and efficiently.

2 SYSTEM DESIGN

2.1 Problem Formulation

We address the challenge of analyzing heterogeneous data sources, including structured databases, tabular files, and document collections, using natural-language (NL) queries. To meet these requirements, AgenticData provides a comprehensive set of relational operators (e.g., Join, Filter, Project, Aggregate, Sort, etc.) alongside semantic operators such as Semantic Scan, Semantic Join, Semantic Filter, and Semantic Aggregate, which are specifically designed for processing unstructured data or textual columns [6, 10]. Upon receiving an NL query, AgenticData identifies relevant data sources and orchestrates an analytical pipeline comprising a sequence of relational and semantic operators. The system then optimizes the pipeline to minimize the token cost associated with the use of LLMs and efficiently executes the pipeline to generate the final result.

2.2 System Architecture

To enable autonomous analytics over heterogeneous data, AgenticData leverages a multi-agent collaborative framework. The overall architecture is depicted in Figure 1. Offline, AgenticData includes a data profiling agent that constructs a semantic catalog for the heterogeneous data. This agent identifies semantic concepts and maps them to the corresponding data sources. Online, AgenticData operates through several specialized agents: An agentic planner interprets the data and translates natural-language (NL) queries into logical semantic plans. A validator checks the plans for grammatical errors and ensures semantic equivalence with the original query. A memory manager retains historical steps and integrates relevant guidance into the planner's context. An agentic optimizer rewrites the query plan to minimize LLM token costs and reduce latency. Finally, an executor executes the plan across the heterogeneous data sources. This collaborative system ensures efficient, accurate, and automated analytics for diverse and complex data environments. Rather than following a predefined execution workflow, AgenticData dynamically orchestrates agents according to the current execution state, intermediate results and feedback signals. Besides, AgenticData continuously evolves by incorporating execution feedback to refine data profiles, update long-term memory and adjust optimization strategies.

Data Profiling. The data profiling agent analyzes the data sources and generates comprehensive data profiles. These profiles are represented as a graph structure, where each node contains schema information, data descriptions, and sampled values from each dataset, while each edge describes the relationships between datasets. For structured data, nodes include schemas, descriptions, and sample tuples, with edges representing join relationships. For unstructured documents, the agent identifies core semantic dimensions and their interrelationships to construct a semantic catalog. Additionally, it builds a semantic index that maps relationships between documents and semantic concepts [19]. After initializing node profiles, the system encodes their descriptive summaries into vector representations to retrieve candidate related datasets. It then iteratively uses an LLM to infer semantic relationships among the candidates, and updates each node profile along with its vector representation using contextual information from its neighbors. These steps improve data understanding and facilitate cross-source analytics. Furthermore, AgenticData supports external web documents and incorporates Web APIs into data profiles, enabling seamless integration of web-based resources.

Agentic Planner. The planner converts an NL query into an executable pipeline composed of relational and semantic operators. It begins by selecting relevant data based on the data profiles. Next, the plan generation agent breaks down the query into simpler sub-steps using LLM-based reasoning and constructs a high-level semantic plan. This high-level plan preserves the original query's intent while simplifying the generation of specific query plans for each step. Guided by the high-level plan, the agent orchestrates the operators for each sub-step and merges them into a complete tree-based query plan. The planner is further enhanced by a data validator that provides feedback on the generated plan, allowing for iterative improvements. Additionally, the planner is supported by a memory system that retains historical plans and error information. This memory system helps guide the planner to correct previous mistakes in subsequent iterations, optimizes the use of context to maintain high performance, and recalls prior experiences to prevent similar errors during task planning and query plan generation.

Agentic Validator. Query plans generated by LLMs may contain errors due to hallucinations or inaccuracies. To promptly identify faulty plans and improve planning accuracy, we introduce a validator that detects grammatical issues and logical deviations from the original query. The validation process starts with a rule-based checker that examines the plan for basic errors. First, a parser converts the raw JSON into a structured representation, verifying operator types and required parameters. Next, a semantic analyzer infers the output schema for each node and checks the correctness of column references within the operators. During these stages, any validation failures and error locations are recorded to support subsequent plan refinement. If the rule-based checks are successful,

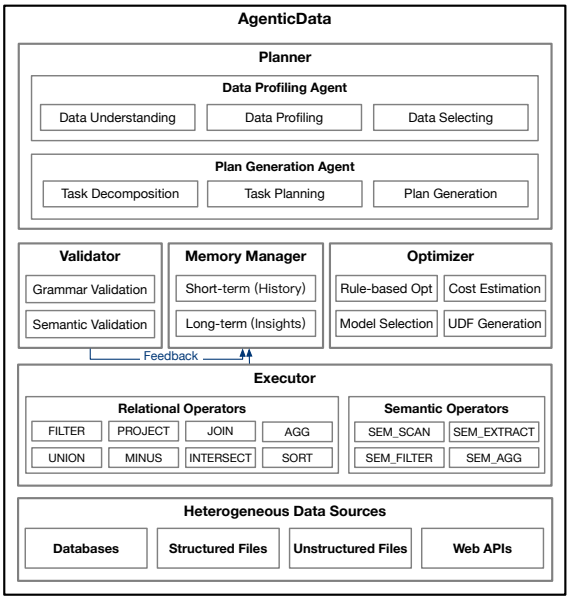

Figure 1: `AgenticData` **Overview Design**

an LLM-based logical checker is used to further assess the semantic alignment between the NL query and the generated plan. This advanced checker evaluates the logical correctness across several dimensions, including the appropriateness of data source selection, the fulfillment of all query conditions, and the consistency of the output schema with the original query's intent. If any grammatical or logical errors are identified during the validation, the validator stores the error types and detailed information in memory. These insights are then relayed to the planner, enabling it to address the issues and refine the query plan in the next planning iteration.

Agentic Optimizer. In heterogeneous data analytics scenarios, the cost associated with LLM invocations in semantic operators often accounts for the majority of execution expenses. To reduce both cost and latency while preserving accuracy, `AgenticData` incorporates an optimizer that transforms the logical query plan into an optimized physical plan for execution. The optimization process begins with rule-based logical optimization, which includes strategies such as filter pushdown and pruning. These steps are followed by reordering the operators based on cost estimations to improve efficiency. Next, physical optimization is applied to further minimize the costs associated with semantic operators. The optimizer selects appropriate models based on task complexity and estimated context length, striking a balance between accuracy and cost. It also finds opportunities to replace expensive LLM calls with more efficient user-defined functions, such as using regular expressions to extract specific fields. These optimizations ensure that our system remains cost-effective and efficient while maintaining accuracy.

Agentic Executor. The execution component runs the optimized query plan to generate the final result. It begins by retrieving data from various data sources through their respective interfaces, such as SQL queries for databases, the Pandas DataFrame API for local tabular files, and HTTP requests for online resources. During data retrieval, selection and filtering operations are applied whenever possible to reduce the data volume before further processing. Next, the component follows a bottom-up strategy to execute each operator in the plan in a systematic manner. Operators that do not have dependencies can be processed in parallel to improve efficiency. For runtime exceptions arising from APIs or internal operators, the executor aborts the task and logs the error details in memory for future analysis and troubleshooting.

Memory Manager. In agentic systems, the memory manager is response for the persistent storage, retrieval, and maintenance of historical experiences throughout agent execution, enabling the agent to compress long interaction histories while retaining key information across LLM invocations[22]. Specifically in `AgenticData`, the memory manager has two primary responsibilities: (1) maintaining a history of planning steps and feedback, and (2) synthesizing insights from past experiences to guide the current planning process. Compared to traditional ReAct-style frameworks, where historical actions and observations are simply appended to the context, our memory manager offers two key advantages. First, it compresses historical information to prevent the dilution of the LLM's attention while still providing clear guidance for the next planning step. Second, it enables the system to learn from past experiences and avoid repeating errors during planning. At each planning step, the validator and executor record the query plan along with the corresponding feedback. The memory manager analyzes these records to identify the root causes of errors and stores the entries as tuples ⟨query, plan, feedback, cause⟩ in a short-term memory buffer. Outdated entries in the buffer are periodically removed to maintain efficiency. Additionally, the memory manager groups records with similar error causes and identifies frequently recurring issues. When such patterns are detected, an LLM is employed to summarize query patterns, problematic plan segments, error causes, and the insights derived from them. These summarized records are then stored in a vector database with semantic indexes based on query patterns, forming the system's long-term memory. During subsequent planning iterations, the memory manager retrieves the latest plans and feedback from short-term memory to assist in immediate plan refinements. Simultaneously, long-term memory records are retrieved from the vector database by matching the semantic similarity between the current query and stored query patterns. This dual-layer memory approach guides the planner to avoid repeating similar mistakes and ensures more efficient and accurate planning.

3 EXPERIMENTS

We evaluate `AgenticData` on two real-world benchmarks, `Real-Bank` (https://anonymous.4open.science/r/AgenticData-benckmark-Real-Bank) and `Wikipedia` (https://dumps.wikimedia.org/), and compare it with state-of-the-art baselines. `Real-Bank` is a real-world customer dataset from a bank, consisting of 50 commonly asked data analytics questions related to banking business analysis. It includes a masked dataset of real deposits and loans, organized into 9 tables, along with 1 document describing the business rules. `Wikipedia` contains 100 natural-language queries on 1000 textual documents, and queries typically involve semantic filtering and aggregation over documents, such as counting documents related to a given concept and comparing the frequency of different topics. The results

Table 1: Accuracy Comparison on Real Benchmarks (%).

Dataset	Systems	LLM	Accuracy
Real-Bank	AgenticData	Qwen3	70.00
	ReFoRCE	Qwen3	50.25
Wikipedia	AgenticData	Qwen3	95.00
	Code Agent + Palimpzest	Qwen3	68.00

are shown in Table 1. On Real-Bank, AgenticData outperforms ReFoRCE [1] by 19.75% in accuracy. This improvement is mainly attributed to our enhanced data profiling and planning capabilities, which enable better understanding of ambiguous attributes and more accurate data source selection. On Wikipedia, we compare with a code agent (built on smolagents https://huggingface.co/docs/smolagents/index, a multi-step agent framework) combined with Palimpzest [6] (a Python library integrating semantic operators). Since Palimpzest focuses on semantic data processing rather than natural-language query understanding, the code agent translates natural-language queries into executable Python programs using the Palimpzest APIs. AgenticData achieves higher accuracy than the baseline. This demonstrates the effectiveness of our semantic query planning framework in handling unstructured data, where relevant information must be extracted and organized before analysis. *Overall, these results show that* AgenticData *can generate accurate query plans across both structured and unstructured data, while effectively integrating heterogeneous data sources.*

4 DEMONSTRATION SCENARIOS

We present the implementation and scenarios of AgenticData. Our demonstration aims to demonstrate the capability of AgenticData to perform joint analytics across heterogeneous data sources.

Financial Rule Analysis. We demonstrate the data analyzing workflow through a task in DABStep [2], a dataset concerning heterogeneous data analytics in a financial payment setting. It models a pricing system for payment processing, where fees are determined based on merchant attributes and transaction characteristics.

The data consists of both structured and unstructured sources. Four structured tables include history transaction records, fee rules, and lookup tables describing other dimensions. In addition, two textual documents provide essential business knowledge. They explain transaction characteristics such as merchant account types and Authorization Characteristics Indicators (ACIs), specify how transactions are matched to fee rules based on their characteristics, and describe how processing fees are calculated according to matched rules and transaction volumes.

Consider a query "For a credit transaction of 1 euro on NexPay, what would be the most expensive Authorization Characteristics Indicator (ACI)? In the case of a draw between multiple ACIs, return the ACI with the lowest alphabetical order." AgenticData performs the analysis and displays the execution progress through an interactive Web interface, as shown in Figure 2.

During the process, the planner first profiles the data sources (❶), identifying fee rule entries in fees.json, a description of how fees are evaluated in a chunk of manual.md, and a summary of

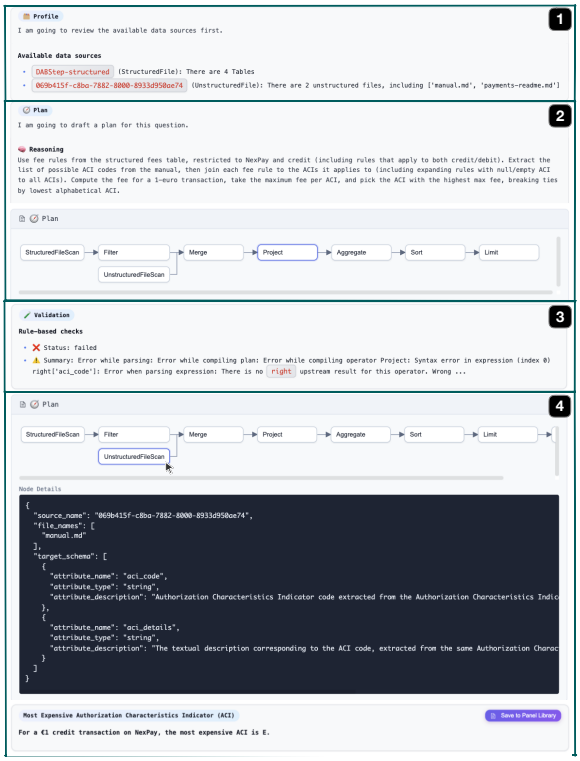

Figure 2: Financial Rule Analysis

ACI in another chunk of manual.md. Based on these profiles, the planner performs reasoning to understand the scenario, recognizing that ACI information is not explicitly available in structured tables but defined in the document, and that fee computation follows rule-based matching. After this understanding and locating the relevant data sources, the planner generates a query plan (❷), whose skeleton is shown on the page. However, the plan encounters an error that references a non-existent input. The validator detects it (❸) and the feedback is stored in memory. In the next turn, the previous plan and its error are loaded into the planner's context, guiding the planner to refine the plan until it passes validation. The final plan is then executed to produce the final result (❹). During the process, the details of each operator can be shown by clicking the corresponding node.

This scenario shows AgenticData's flexible approach to analyzing unstructured files. On the one hand, unstructured files can serve as background knowledge for the planner; on the other hand, AgenticData can extract relevant information through semantic operators and transform it into structured representations (as shown in step (❹) of Figure 2), avoiding the need for the LLM to ingest large volumes of raw text.

Conflict-of-Interest Detection. We consider a conflict-of-interest (COI) detection task to assist authors in identifying potential conflicts when submitting a paper. Given one or more target authors and a conference, AgenticData retrieves a reviewer list (e.g., from

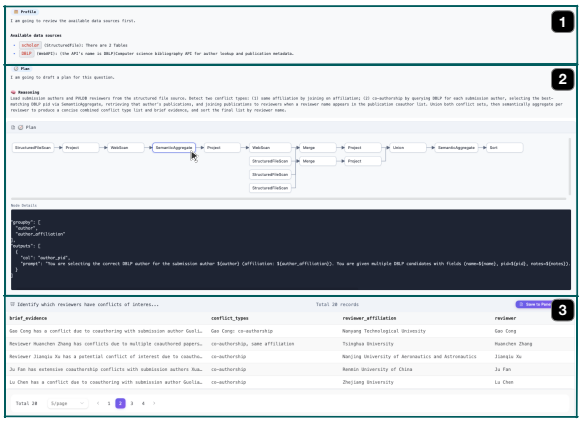

Figure 3: Conflict-of-Interest Detection

a provided URL or conference name) and preprocesses it into structured data. Users can flexibly define COI criteria in natural language, such as affiliation overlap or prior co-authorship, and query conflicts for one or multiple authors. `AgenticData` then identifies reviewers with conflicts and provides the corresponding details. For demonstration, we use the PVLDB reviewer list(https://vldb.org/pvldb/volumes/17/review-board/) and a paper submission (https://dl.acm.org/doi/10.1145/3722212.3725091), and ask: "Identify which reviewers have conflicts of interest with this submission (e.g., co-authorship or same affiliation)". The corresponding workflow is illustrated in Figure 3. During the data profiling stage, `AgenticData` identifies a suitable Web API provided by DBLP according to the query intent, and builds profiles of them including the request specification and the format of returned results. Based on the input data and the API specification, the planner generates a query plan to detect the two types of conflicts. For affiliation-based conflicts, the plan directly joins the list of reviewers and authors on affiliation. For co-authorship conflicts, the plan issues Web queries for each author name, and uses a `SemanticAggregate` operator to identify the most likely scholar profiles by considering contextual signals such as affiliation, resolving ambiguity caused by homonyms. The retrieved profiles are then used to collect publication and co-author information, which are matched against the reviewer list to identify overlaps. Finally, the results from different conflict types are merged and aggregated into a table, where each reviewer is associated with a list of detected conflicts and supporting evidence.

ACKNOWLEDGMENT

Ji Sun is the corresponding author. This paper was supported by National Key R&D Program of China (2023YFB4503600), NSF of China (62525202, 62232009), Shenzhen Project (CJGJZD20230724093403007), Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China(JYB2025XDXM509), China Railway Science Research Institute Group Co., Ltd, Zhongguancun Lab, Huawei, and Beijing National Research Center for Information Science and Technology (BNRist).

REFERENCES

[1] Minghang Deng, Ashwin Ramachandran, Canwen Xu, Lanxiang Hu, Zhewei Yao, Anupam Datta, and Hao Zhang. 2025. ReFoRCE: A Text-to-SQL Agent with Self-Refinement, Format Restriction, and Column Exploration. (2025).

[2] Alex Egg, Martin Iglesias Goyanes, Friso Kingma, Andreu Mora, Leandro von Werra, and Thomas Wolf. 2025. DABstep: Data Agent Benchmark for Multi-step Reasoning. *arxiv.org/abs/2506.23719* (2025). https://arxiv.org/abs/2506.23719

[3] Ruilin Hu, Yuyu Luo, Guoliang Li, Shuangqiao Wu, and Yun Luo. 2026. OpenSQL: Data-Efficient Text-to-SQL for Open-Source LLMs via Synthesized Intermediate Supervision. *Proc. VLDB Endow.* (2026).

[4] Guoliang Li, Jiayi Wang, Chenyang Zhang, and Jiannan Wang. 2025. Data+AI: LLM4Data and Data4LLM. In *Companion of the 2025 International Conference on Management of Data, SIGMOD/PODS 2025, Berlin, Germany, June 22-27, 2025*, Volker Markl, Joseph M. Hellerstein, and Azza Abouzied (Eds.). ACM, 837–843. https://doi.org/10.1145/3722212.3725641

[5] Guoliang Li, Xuanhe Zhou, and Xinyang Zhao. 2024. LLM for Data Management. *Proc. VLDB Endow.* 17, 12 (2024), 4213–4216. https://doi.org/10.14778/3685800.3685838

[6] Chunwei Liu, Matthew Russo, Michael J. Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael J. Franklin, Tim Kraska, Samuel Madden, Rana Shahout, and Gerardo Vitagliano. 2025. Palimpzest: Optimizing AI-Powered Analytics with Declarative Query Processing. In *CIDR*.

[7] Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuxin Zhang, Ju Fan, Guoliang Li, Nan Tang, and Yuyu Luo. 2025. A Survey of Text-to-SQL in the Era of LLMs: Where Are We, and Where Are We Going? *IEEE Trans. Knowl. Data Eng.* 37, 10 (2025), 5735–5754. https://doi.org/10.1109/TKDE.2025.3592032

[8] Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuxin Zhang, Ju Fan, Guoliang Li, Nan Tang, and Yuyu Luo. 2025. A Survey of Text-to-SQL in the Era of LLMs: Where Are We, and Where Are We Going? *IEEE TKDE* 37, 10 (2025), 5735–5754.

[9] Yuyu Luo, Guoliang Li, Ju Fan, and Nan Tang. 2026. Data Agents: Levels, State of the Art, and Open Problems. In *Companion of the International Conference on Management of Data, SIGMOD Companion 2026, Bengaluru, India, 31 May 2026- 5 June 2026*. ACM, 571–579. https://doi.org/10.1145/3788853.3801878

[10] Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2024. Semantic operators: a declarative model for rich, ai-based data processing. *arXiv preprint arXiv:2407.11418* (2024).

[11] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. 2025. DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. *Proc. VLDB Endow.* 18, 9 (2025), 3035–3048.

[12] Ji Sun, Guoliang Li, Peiyao Zhou, Yihui Ma, Jingzhe Xu, and Yuan Li. 2025. AgenticData: An Agentic Data Analytics System for Heterogeneous Data. arXiv:2508.05002 [cs.DB] https://arxiv.org/abs/2508.05002

[13] 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.* 49, 3 (2025), 79–95. http://sites.computer.org/debull/A25sep/p79.pdf

[14] Zhaoyan Sun, Shan Zhong, Daizhou Wen, Jiaxing Han, Guoliang Li, Ying Yan, Peng Zhang, Yu Su, Xiang Qi, Baolin Sun, Chengyuan Yang, Tao Fang, and Huaiyu Ruan. 2026. AgenticDataBench: A Comprehensive Benchmark for Data Agents. arXiv:2607.01647 [cs.DB] https://arxiv.org/abs/2607.01647

[15] Jiayi Wang and Jianhua Feng. 2025. Unify: An Unstructured Data Analytics System. In *ICDE 2025*. 4662–4674. https://doi.org/10.1109/ICDE65448.2025.00374

[16] Jiayi Wang and Guoliang Li. 2025. AOP: Automated and Interactive LLM Pipeline Orchestration for Answering Complex Queries. In *CIDR*.

[17] 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. http://sites.computer.org/debull/A25mar/p57.pdf

[18] Bangrui Xu, Qihang Yao, Zirui Tang, Xuanhe Zhou, Yeye He, Shihan Yu, Qianqian Xu, Bin Wang, Guoliang Li, Conghui He, and Fan Wu. 2026. MoDora: Tree-Based Semi-Structured Document Analysis System. *Proc. ACM Manag. Data* (2026).

[19] Sihui Xu, Jiayi Wang, and Guoliang Li. 2026. Bridging the Gap: Cardinality Estimation for Semantic Queries on Unstructured Data. In *SIGMOD*.

[20] Yuxin Zhang, Meihao Fan, Ju Fan, Mingyang Yi, Yuyu Luo, Guoliang Li, Bin Wu, and Wenchao Zhou. 2026. Reward-SQL: Boosting Text-to-SQL via Stepwise Execution-Aware Reasoning and Process-Supervised Rewards. *Proc. ACM Manag. Data* (2026).

[21] Xinyang Zhao, Xuanhe Zhou, and Guoliang Li. 2024. Chat2Data: An Interactive Data Analysis System with RAG, Vector Databases and LLMs. *Proc. VLDB Endow.* 17, 12 (2024), 4481–4484. https://doi.org/10.14778/3685800.3685905

[22] Wei Zhou, Xuanhe Zhou, Shaokun Han, Hongming Xu, Guoliang Li, Zhiyu Li, Feiyu Xiong, and Fan Wu. 2026. Are We Ready For An Agent-Native Memory System? arXiv:2606.24775 [cs.CL] https://arxiv.org/abs/2606.24775

[23] 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. *Proc. VLDB Endow.* 17, 10 (2024), 2514–2527. https://doi.org/10.14778/3675034.3675043

[24] Xuanhe Zhou, Guoliang Li, Jianming Wu, Jiesi Liu, Zhaoyan Sun, and Xinning Zhang. 2023. A Learned Query Rewrite System. *Proc. VLDB Endow.* 16, 12 (2023), 4110–4113. https://doi.org/10.14778/3611540.3611633

[25] Xuanhe Zhou, Zhaoyan Sun, and Guoliang Li. 2024. DB-GPT: Large Language Model Meets Database. *Data Sci. Eng.* 9, 1 (2024), 102–111. https://doi.org/10.1007/S41019-023-00235-6