Exploring the Semantic Gap in Agentic Data Systems: A Formative Study of Operationalization Failures in Analytical Workflows

Jalal Mahmud
Megagon Labs
California, USA
jalal@megagon.ai

Eser Kandogan
Megagon Labs
California, USA
eser@megagon.ai

ABSTRACT

Large language models (LLMs) are increasingly used to generate queries, invoke tools, and construct analytical workflows. Although recent advances have substantially improved workflow generation and execution, the semantic information required to operationalize analytical concepts often lies beyond what is explicitly represented in database schemas and data values. We present a cross-domain formative study of operationalization failures in agent-generated analytical workflows. Across 236 analytical intents spanning finance, human resources, and public safety domains, we identify 153 recurring failures in successfully executed SQL queries. Our analysis reveals five recurring classes of failures: comparative grounding, process reasoning, quantitative reasoning, role confusion, and policy grounding. These findings suggest a semantic gap between user-level analytical concepts and the information available to workflow-generation systems. More broadly, they raise questions about the admissibility of analytical operations and suggest that future agentic data systems may require richer semantic representations to bridge the gap between analytical intent and executable computation.

VLDB Workshop Reference Format:
Jalal Mahmud and Eser Kandogan. Exploring the Semantic Gap in Agentic Data Systems: A Formative Study of Operationalization Failures in Analytical Workflows. VLDB 2026 Workshop: Novel Optimizations for Visionary AI Systems.

1 INTRODUCTION

Large language models (LLMs) are rapidly becoming a core primitive of modern data systems. Modern AI-native systems increasingly rely on LLMs to generate queries, retrieve evidence, invoke tools, and construct analytical workflows. Examples include NL2SQL systems, retrieval-augmented systems, enterprise analytical assistants, and agentic planners that combine reasoning with database and retrieval operations [9, 10, 16, 20, 21]. Recent advances have substantially improved intent interpretation, workflow generation, and answer correctness [5, 12, 20, 21]. Nevertheless, analytical failures continue to occur even in workflows that execute successfully.

Consider an analytical assistant tasked with answering the request: *"Find unusually risky loans with sustained delinquency."* A generated workflow may operationalize *"unusually risky"* using a fixed threshold and *"sustained delinquency"* using a row-level predicate. However, unusual risk is inherently comparative and typically requires reasoning relative to a reference population, while sustained delinquency describes a temporal process that cannot be inferred from a single observation. Similarly, when asked to *"identify intersections with the highest fatality rate,"* the generated workflow may rank intersections using total fatalities rather than normalized fatality rates. Such a workflow conflates counts with rates, favoring high-volume intersections even when their fatality rate is lower. In both cases, the generated workflow is syntactically valid and executable, yet the selected operations do not faithfully capture the analytical concept expressed in the user's intent.

We refer to such cases as *operationalization failures*: successfully executed queries whose selected analytical operations do not faithfully instantiate the concept expressed in the user's intent. We distinguish these failures from syntax errors, execution failures, and incorrect schema linking. In an operationalization failure, the query may use plausible tables and attributes and return a result, but its threshold, analytical role, temporal operator, derived metric, or policy criterion is inappropriate for the requested concept.

These examples suggest a broader semantic gap in agentic data interaction. Existing data systems have traditionally focused on bridging gaps in intent interpretation, schema understanding, and query generation. Our findings suggest an additional semantic frontier concerned with the operationalization of analytical concepts. Concepts such as *unusual*, *persistent*, *high-risk*, and *rate* must ultimately be translated into executable computations, yet the information required to perform this translation is often not explicitly represented in database schemas or data values.

As shown in Figure 1, human analysts routinely draw on statistical context, process knowledge, metric definitions, analytical roles, and organizational policies when interpreting such concepts, whereas agents often have access only to database schemas, data values, and limited metadata.

To better understand this phenomenon, we conducted a cross-domain formative study spanning finance, human resources, and public safety analytics. Across 236 analytical intents, we identified 153 recurring failures in successfully executed SQL queries. Analysis revealed five recurring classes of failures: comparative grounding, role confusion, process reasoning, quantitative reasoning, and policy grounding. Collectively, these findings suggest a broader concern regarding whether the analytical operations selected by a workflow are appropriate for expressing the intended analytical concept. We refer to this concern as analytical admissibility and discuss its implications for future AI-native analytical systems.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

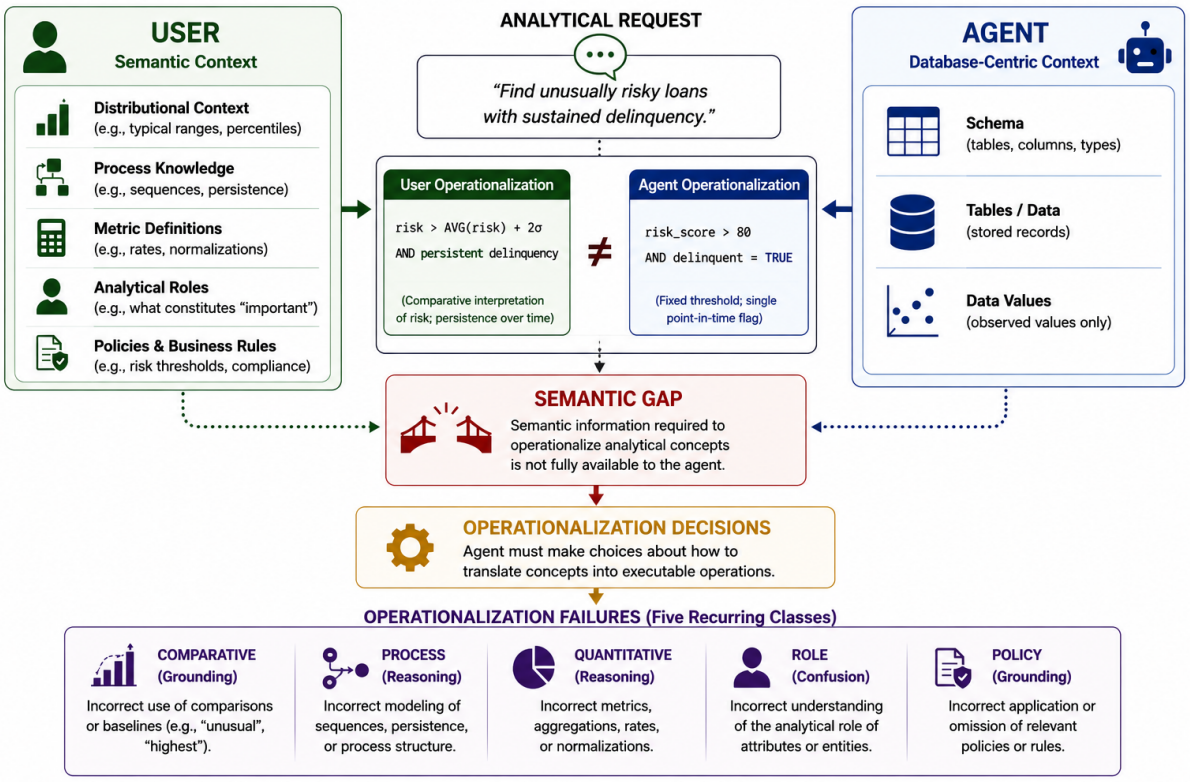

Figure 1: Illustration of the semantic gap in agentic data systems. Information required to operationalize analytical concepts may not be fully available to workflow-generation systems, leading to operationalization failures despite successful execution.

This paper makes three contributions. First, we present a cross-domain manual analysis of 236 successfully executed SQL queries and identify 153 operationalization failures. Second, we characterize five recurring failure classes and validate their presence in independently constructed BIRD workloads. Third, we discuss how the identified failure classes can inform future mechanisms for targeted clarification, semantic retrieval, validation, and workflow repair.

2 RELATED WORK

Our study relates to three areas: NL2SQL and agentic analytical systems, failure analysis of LLM-powered systems, and semantic representations for data systems.

NL2SQL and Agentic Analytical Systems Recent advances in large language models have substantially improved natural-language interfaces to structured data. Modern NL2SQL systems leverage schema linking, retrieval augmentation, in-context learning, self-correction, decomposition, and candidate selection to generate increasingly complex analytical queries [12, 13, 20, 21, 28].

More broadly, agentic systems extend query generation to multi-step planning, tool use, and workflow construction over heterogeneous data sources [9, 10, 16, 25, 29]. These systems have significantly improved the ability to interpret user intent and generate executable analytical workflows. GenEdit [18] demonstrates how enterprise-specific examples, instructions, schema elements, and user feedback can be maintained as an external knowledge set for Text-to-SQL generation. Such a knowledge set could supply some of the semantic information missing in the failures we observe. Our work is complementary: rather than proposing a new generation mechanism, we empirically characterize which kinds of operationalization knowledge may need to be represented, retrieved, or validated.

Failure Analysis of LLM-Powered Systems Prior work has examined failure modes in LLM-powered and agentic systems, including reasoning errors, planning failures, and tool-use failures [15, 24, 25]. Within NL2SQL, benchmarks such as Spider, BIRD, and Spider 2.0 expand evaluation from semantic parsing to database-grounded reasoning and enterprise-scale workflow generation [12, 13, 28].

Recent work has also highlighted gaps between benchmark performance and real-world analytical usability [5]. Gomm et al. [6] distinguish cooperative queries whose interpretations can be resolved through available context or interaction from uncooperative queries that lack sufficient specification. Our failure classes provide a complementary characterization: they identify the type of semantic dependency that must be resolved during operationalization. Some cases may require user clarification, whereas others may be resolved using data distributions, temporal structure, metric definitions, or external policies.

Our work contributes a cross-domain manual analysis of successfully executed queries to determine whether their analytical operations faithfully instantiate the concepts expressed by users. Rather than proposing a new query-generation method, we provide evidence that operationalization failures are frequent, structured, and recur across independently constructed workloads.

Semantic Representations for Data Systems Prior work in data management has explored schemas, metadata catalogs, semantic layers, ontologies, business glossaries, and data constraints as mechanisms for improving data interpretation and governance [1, 3, 11]. Modern metadata platforms such as DataHub [14], OpenMetadata [19], and Amundsen [17] enrich data assets with lineage, governance metadata, and business semantics. Contemporary semantic layers, including Snowflake Semantic Views and LookML [7, 22], make entities, dimensions, facts, metrics, relationships, and governed calculations explicit for analytical applications. These mechanisms can address portions of the observed gap, particularly analytical roles and quantitative definitions.

More broadly, measurement theory studies which operations are meaningful for different classes of quantities [23]. Recent work on schema-aware language models and table representation learning [4, 8, 26, 27] learns semantic representations of tables, columns, and relational context to improve tasks such as semantic parsing, retrieval, and table question answering. These approaches improve contextual understanding of structured data through learned semantic representations.

Our work is complementary to these efforts. Existing semantic representations can address portions of the observed gap by making business concepts, analytical roles, metrics, and relationships explicit. Our study empirically characterizes where such information is needed during workflow generation and highlights semantic requirements involving comparative baselines, temporal processes, quantitative constraints, and external policies. The observed failure classes therefore provide a diagnostic lens for determining which semantic information should be represented, retrieved, or validated for a given analytical concept.

3 FORMATIVE STUDY OF OPERATIONALIZATION FAILURES

To better understand how analytical failures arise in agent-generated workflows, we conducted a multi-domain formative study spanning finance, human resources, and public safety analytics.

Study Setup We selected three representative relational datasets covering finance, human resources, and public safety domains. The finance dataset was drawn from the BIRD benchmark [13]; the public safety dataset was derived from the NYC Motor Vehicle Collisions dataset publicly available via Kaggle [2]; and the human resources dataset was constructed from anonymized enterprise data. All experiments were conducted using a GPT-4o-based NL2SQL agent implemented within the Blue platform [9]. We use NL2SQL as a concrete and inspectable setting in which analytical operations can be directly examined.

The agent generated analytical queries using schema metadata, attribute descriptions, profiling statistics, and representative sample values, without handcrafted reasoning rules or domain-specific analytical guidance. We constructed 250 natural language intents reflecting realistic enterprise analytical tasks in finance, human resources, and public safety domains. After feasibility validation, 236 executable intents were retained. Representative examples include:

- Find unusually large withdrawals.
- Accounts with recurring overdraft cycles.
- Areas with worsening pedestrian injury trends.
- Intersections with the highest fatality rate.

The intents were formulated at the level of analytical tasks rather than schema-specific column names or literal value encodings. They span filtering, aggregation, comparison, temporal analysis, and multi-attribute reasoning commonly encountered in enterprise reporting and decision-support settings.

Failure Identification For each intent, the NL2SQL agent generated and executed a SQL query. We manually inspected the generated query and its result. A query was classified as an operationalization failure if it was (1) syntactically valid, (2) executed successfully, and (3) reflected a different analytical interpretation than the one expressed in the natural language intent.

Failure categories were derived through iterative qualitative analysis and each failure was assigned to a single dominant class.

Key Findings Our analysis yielded three principal findings.

Finding 1: Operationalization failures are common. Across 236 analytical intents, we identified 153 operationalization failures despite successful query execution. The remaining 83 intents were judged to be appropriately operationalized.

Finding 2: Operationalization failures generalize beyond the study workload. To assess external validity, we analyzed two independently constructed workloads from the BIRD benchmark [13].

First, we analyzed 31 analytical intents from the BIRD MiniDev financial benchmark [13]. Despite being independently constructed, 19 of 31 intents (61%) exhibited operationalization failures.

Second, we analyzed 145 intents labeled *challenging* by the BIRD benchmark authors [13], spanning 11 databases across diverse domains. Of the 130 intents that produced executable queries (15 were excluded due to API timeouts or SQL generation failures), 88 (68%) exhibited operationalization failures.

This suggests that operationalization failures are not specific to our study workload and may arise more broadly in analytical workflow generation tasks.

Finding 3: Operationalization failures are highly structured. Rather than arising from isolated model mistakes, the observed failures clustered into a small number of recurring categories that appeared consistently across domains and benchmarks.

4 RECURRING FAILURE CLASSES

Across the 153 identified operationalization failures, we observed five recurring classes summarized in Table 1. These categories are not intended as a complete taxonomy, but rather recurring operationalization patterns observed across domains and benchmarks.

Comparative Grounding Failures Comparative concepts such as *high*, *large*, *unusual*, *severe*, and *extreme* require reasoning relative to a reference population or distribution. Although these concepts are expressed qualitatively in natural language, their operationalization typically depends on comparative reasoning over observed data. Generated workflows frequently operationalize such concepts

Table 1: Recurring operationalization failure classes identified in the formative study. Percentages are computed over the 153 identified operationalization failures.

Failure Class	Description	Illustrative Example	%
Comparative Grounding	Comparative concepts are operationalized using arbitrary thresholds rather than distributional or domain-grounded interpretations	"severe collisions" → `injured >= 10`	28%
Role Confusion	Attributes are assigned inappropriate analytical roles, treating counts, frequencies, or identifiers as evidence for higher-level concepts	"most demanding job titles" → `ORDER BY COUNT(*)` (frequency ≠ intensity)	14%
Process-Level Reasoning	Process-oriented states emerging from event sequences are reduced to row-level predicates, ignoring persistence, recurrence, or temporal evolution	"sustained negative balance" → `balance < 0`	26%
Quantitative Reasoning	Derived quantities are constructed without respecting aggregation scope, functional dependencies, normalization requirements, or measurement constraints	"total amount paid back" → `amount + payments * duration` (principal counted twice)	13%
Policy Grounding	Institutionally defined concepts are operationalized using local heuristics or observed data statistics rather than externally defined criteria	"reportable incident" → severity threshold inferred from data	19%

Table 2: Illustrative examples of operationalization failures and knowledge-aware alternatives. The observed operationalization may be executable, but lacks the semantic context needed to faithfully express the intended analytical concept.

Failure Class	Intent	Observed Operationalization	Knowledge-Aware Operationalization
Comparative Grounding	Unusually large withdrawals	`WHERE amount > 10000`	`WHERE amount > AVG(amount) + 2*STDDEV(amount)`
Role Confusion	Most demanding jobs	`ORDER BY COUNT(*)`	Rank by qualification requirements
Process Reasoning	Recurring overdrafts	`WHERE balance < 0`	Temporal persistence analysis over account history
Quantitative Reasoning	Highest fatality rate	`ORDER BY SUM(fatalities)`	`SUM(fatalities) / COUNT(*)`
Policy Grounding	Reportable transactions	`WHERE amount > 50000`	Apply policy-defined reporting threshold

using arbitrary fixed thresholds rather than contextual or distributional interpretations. For example, *unusually large withdrawals* may be implemented using a fixed threshold without justification that the threshold corresponds to unusual behavior within the observed population.

Role Confusion Failures Many analytical concepts depend not only on an attribute's values but also on the analytical role the attribute plays within a computation. An attribute's role governs which operations are semantically meaningful or analytically admissible. Generated workflows frequently assign inappropriate roles to attributes, treating identifiers, frequencies, counts, or other observable quantities as direct evidence for higher-level concepts. For example, *most demanding job titles* may be operationalized as `ORDER BY COUNT(*)`, conflating job posting frequency with qualification intensity. Similar failures arise when counts, identifiers, or activity frequencies are used as proxies for concepts such as importance, influence, risk, or complexity.

Process-Level Failures Many analytical intents describe processes unfolding over time rather than properties of individual records. Meaningful states often emerge from patterns across events rather than single observations. Examples include recurring overdraft cycles, worsening injury trends, sustained delinquency, and accelerating transaction activity. Generated workflows frequently collapse such process-oriented concepts into row-level predicates, ignoring persistence, recurrence, trends, or temporal evolution. For example, *accounts with recurring overdraft cycles* may be operationalized as a single negative balance rather than recurrence across time.

Quantitative Reasoning Failures Many analytical requests require constructing derived quantities through aggregation, normalization, or metric composition. Generated workflows frequently ignore aggregation scope, functional dependencies, or measurement-level constraints, leading to semantically invalid derived metrics. For example, *total repayment* may be computed as `amount + payments * duration`, double-counting principal despite arithmetic validity. Similar failures arise when raw counts are substituted for rates, incompatible quantities are combined, or derived metrics are constructed without respecting their underlying semantics.

Policy Grounding Failures Certain analytical concepts are defined by external policies, regulations, or institutional standards rather than solely by observed data. Generated workflows frequently operationalize such concepts using arbitrary thresholds or local data statistics while ignoring externally defined criteria. For example, *reportable incidents* may be implemented using a severity

threshold inferred from data distributions rather than the governing policy definition.

Table 2 illustrates how operationalization failures can arise when the semantic information required to express an analytical concept is unavailable, not retrieved, or incorrectly applied during workflow generation.

5 IMPLICATIONS FOR AGENTIC DATA SYSTEMS

Our findings suggest that correctly identifying a user's intent, grounding it to the appropriate schema elements, and generating an executable workflow are often not sufficient. Even when these steps succeed, the resulting computation may fail to faithfully capture the analytical concept expressed by the user.

The five failure classes provide more actionable guidance than treating all such cases as missing specifications. Each class identifies a different semantic dependency and therefore suggests a different system response: comparative grounding requires a reference population or baseline; role confusion requires admissible attribute roles; process reasoning requires temporal structure and sequence operators; quantitative reasoning requires metric definitions and aggregation constraints; and policy grounding requires authoritative external criteria. The classes can therefore guide targeted clarification, semantic retrieval, validation, and workflow repair rather than treating every underspecified request uniformly.

One possible direction is to expose agents directly to additional sources of semantic information, including policies, documentation, process descriptions, and domain knowledge. However, reasoning over large collections of heterogeneous semantic artifacts introduces challenges in retrieval, grounding, consistency, and interpretation.

An alternative direction is to develop intermediary semantic representations that explicitly capture operationalization knowledge in reusable form. Such representations could encode comparative baselines, process semantics, metric definitions, policy criteria, or admissible analytical operations, allowing agents to bridge the gap between analytical concepts and executable computations more systematically. Understanding what constitutes an admissible operationalization therefore represents an important direction for future research on agent-generated analytical workflows and AI-native data systems.

6 CONCLUSION

We presented a formative study of operationalization failures in agent-generated analytical workflows. The recurring failure classes identified in our study suggest that future agentic data systems may require richer semantic representations capable of bridging the gap between user-level analytical concepts and executable computations. Our experiments use a specific NL2SQL system as a concrete, inspectable setting for studying the relationship between analytical concepts and their computational operationalization. The broader relevance of these failure classes to multi-tool and heterogeneous agent workflows is a hypothesis motivated by our findings rather than a result established by this study. Whether the identified failure classes transfer unchanged to multi-tool or heterogeneous agent workflows remains an empirical question.

An important direction for future work is therefore to assess the extent to which the identified operationalization failure classes persist across different workflow-generation architectures, language models, and metadata environments. Such studies would help distinguish failures arising from limitations of particular systems from those reflecting more fundamental challenges in bridging the semantic gap between analytical concepts and executable computations. As AI-native systems increasingly generate analytical workflows autonomously, understanding how analytical concepts should be operationalized over available data represents an important direction for future research.

REFERENCES

[1] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. 2015. Profiling relational data: a survey. *The VLDB Journal* 24, 4 (2015), 557–581.

[2] City of New York. 2023. NYC Motor Vehicle Collisions. https://www.kaggle.com/datasets/new-york-city/motor-vehicle-collisions. Accessed: 2026-02-17.

[3] E. F. Codd, S. B. Codd, and C. T. Salley. 1993. *Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate.* Technical Report. E. F. Codd and Associates.

[4] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. TURL: Table Understanding through Representation Learning. *Proceedings of the VLDB Endowment* 14, 3 (2020), 307–319. https://doi.org/10.14778/3430915.3430921

[5] Avrilia Floratou, Fotis Psallidas, Fuheng Zhao, Shaleen Deep, Gunther Hagleither, Wangda Tan, Joyce Cahoon, Rana Alotaibi, Jordan Henkel, Abhik Singla, Alex van Grootel, Brandon Chow, Kai Deng, Katherine Lin, Marcos Campos, Venkatesh Emani, Vivek Pandit, Victor Shnayder, Wenjing Wang, and Carlo Curino. 2024. NL2SQL is a Solved Problem... Not!. In *Conference on Innovative Data Systems Research (CIDR).* CIDR, Chaminade, USA.

[6] Daniel Gomm, Cornelius Wolff, and Madelon Hulsebos. 2025. Are We Asking the Right Questions? On Ambiguity in Natural Language Queries for Tabular Data Analysis. In *AI for Tabular Data Workshop at EurIPS 2025.* https://openreview.net/forum?id=Mm1CGvmm9Z

[7] Google Cloud. 2026. LookML: A Language for Semantic Data Modeling. https://cloud.google.com/looker/docs/what-is-lookml. Accessed: 2026-07-20.

[8] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL).* Association for Computational Linguistics, 4320–4333. https://doi.org/10.18653/v1/2020.acl-main.398

[9] Eser Kandogan, Nikita Bhutani, Dan Zhang, Rafael Li Chen, Sairam Gurajada, and Estevam Hruschka. 2025. Orchestrating Agents and Data for Enterprise: A Blueprint Architecture for Compound AI. arXiv:2504.08148 [cs.DB] https://arxiv.org/abs/2504.08148

[10] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling Declarative Language Model Calls into State-of-the-Art Pipelines. In *International Conference on Learning Representations (ICLR).*

[11] Ralph Kimball and Margy Ross. 2013. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3rd ed.). Wiley.

[12] Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. 2025. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. In *International Conference on Learning Representations (ICLR).* https://openreview.net/forum?id=XmProj9cPs

[13] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track.*

[14] LinkedIn. [n.d.]. DataHub: The Metadata Platform for the Modern Data Stack. https://datahubproject.io/. Open-source project documentation.

[15] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2024. AgentBench: Evaluating LLMs as Agents. In *The Twelfth International Conference on Learning Representations (ICLR).* https://openreview.net/forum?id=zAdUB0aCTQ

[16] Yifu Liu, Yin Zhu, Yingqi Gao, Zhiling Luo, Xiaoxia Li, Xiaorong Shi, Yuntao Hong, Jinyang Gao, Yu Li, Bolin Ding, and Jingren Zhou. 2025. XiYan-SQL:

A Novel Multi-Generator Framework for Text-to-SQL. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (2025).

[17] Lyft Engineering. [n.d.]. Amundsen: A Data Discovery and Metadata Engine. https://www.amundsen.io/. Project documentation and blog posts.

[18] Karime Maamari, Connor Landy, and Amine Mhedhbi. 2025. GenEdit: Compounding Operators and Continuous Improvement to Tackle Text-to-SQL in the Enterprise. In *Conference on Innovative Data Systems Research (CIDR)*. https://arxiv.org/abs/2503.21602

[19] OpenMetadata. 2025. OpenMetadata: An Open Standard for Metadata. https://github.com/open-metadata/OpenMetadata. Accessed: 2026-06-03.

[20] Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O. Arik. 2025. CHASE-SQL: Multi-Path Reasoning and Preference Optimized Candidate Selection in Text-to-SQL. In *International Conference on Learning Representations (ICLR)*.

[21] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. arXiv:2304.11015 [cs.DB] https://arxiv.org/abs/2304.11015

[22] Snowflake. 2026. Overview of Semantic Views. https://docs.snowflake.com/en/user-guide/views-semantic/overview. Accessed: 2026-07-20.

[23] S. S. Stevens. 1946. On the theory of scales of measurement. *Science* 103, 2684 (1946), 677–680.

[24] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R. Narasimhan. 2025. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=roNSXZpUDN

[25] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*.

[26] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 8413–8426. https://doi.org/10.18653/v1/2020.acl-main.745

[27] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2021. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=kyaIeYj4zZ

[28] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3911–3921. https://doi.org/10.18653/v1/D18-1425

[29] Sepanta Zeighami, Yiming Lin, Shreya Shankar, and Aditya Parameswaran. 2025. LLM-Powered Proactive Data Systems. *arXiv preprint arXiv:2502.13016* (2025). https://arxiv.org/abs/2502.13016