Bridging LLMs and Database Systems: A Deep Dive into Enhanced Relational Operators

Tianjing Zeng
Alibaba Group
zengtianjing.ztj@alibaba-inc.com

Yin Lin
Alibaba Group
yin.lin@alibaba-inc.com

Rong Zhu
Alibaba Group
red.zr@alibaba-inc.com

Yunxiang Su
Alibaba Group
suyunxiang.syx@alibaba-inc.com

Zhongjun Ding
Alibaba Group
dingzhongjun.dzj@alibaba-inc.com

Bolin Ding
Alibaba Group
bolin.ding@alibaba-inc.com

Jingren Zhou
Alibaba Group
jingren.zhou@alibaba-inc.com

ABSTRACT

Large language models (LLMs) are increasingly integrated into relational data processing via operator-like components—entity matchers, semantic filters, data imputers, and semantic rankers—that we call *LLM-Enhanced Relational Operators (LROs)*. A growing number of systems and components have proposed diverse LROs, yet no unified view exists. This tutorial establishes a formal taxonomy along three dimensions—operating logic, operand granularity, and implementation variant—distinguishes standalone LRO components for data preparation from multi-LRO semantic query systems, covers optimization techniques (model cascades, semantic batching, cost-based planning), examines benchmark methodology, and discusses open challenges toward deeper LLM–database co-design.

PVLDB Reference Format:
Tianjing Zeng, Yin Lin, Rong Zhu, Yunxiang Su, Zhongjun Ding, Bolin Ding, and Jingren Zhou. Bridging LLMs and Database Systems: A Deep Dive into Enhanced Relational Operators. PVLDB, 19(12): 4867 - 4871, 2026.
doi:10.14778/3827998.3828141

1 INTRODUCTION

Traditional relational databases excel at precise, value-based query processing but exhibit inherent limitations when handling queries requiring semantic understanding. Classical SQL operators, such as `JOIN`, `WHERE`, and `ORDER BY`, are fundamentally predicated on exact value matching: `JOIN` necessitates shared key values, `WHERE` requires syntactically precise predicates, and `ORDER BY` demands numerical comparability. These constraints preclude the effective execution of a broad class of semantic tasks that arise naturally in practice. For instance, traditional approaches struggle to join tables where the same entity appears under different names (*e.g.*, "Google LLC" in one relation and "Alphabet" in another), filter records based on subjective criteria (*e.g.*, "technically innovative companies"), or rank items by inherently non-numerical attributes like cultural influence. Large language models (LLMs) [2, 5, 30] offer a promising alternative, bringing world knowledge, semantic reasoning capabilities, and natural-language understanding to data processing—precisely addressing these limitations.

A growing body of work integrates LLMs into relational processing via *operator-like components*. Representative examples include `sem_filter` in LOTUS [6, 32], `LLMJoin` in BlendSQL [12], and entity matching in ComEM [41], among other LLM-backed operators in these systems. Despite targeting diverse tasks, these share a common pattern: (i) operate over relational data; (ii) accept a natural-language requirement; (iii) invoke LLMs for semantic inference during execution; (iv) produce relational outputs. We abstract such operator-like components as *LLM-Enhanced Relational Operators (LROs)*.

What Are LROs. An LRO extends classical relational operators by wrapping LLM calls within a relational operator interface. It takes as input one or two relations $\mathcal{R}$, an operand granularity $g \in \{\text{cell}, \text{row}, \text{col}, \text{table}\}$, and a natural-language requirement l. It invokes an LLM guided by l and materializes the result as output relation R': $\text{LRO}(\mathcal{R}, g, l) \mapsto R'$. LROs preserve the composability and closure properties of classical algebra while enabling semantics-driven query execution. For example, at row granularity, an LRO-Select might filter employee records to return only "senior engineers", while an LRO-Match might identify that "Apple Inc." and "AAPL" refer to the same company across two different tables.

Why LROs Matter: Bridging LLMs and Databases. LROs bridge traditional databases and LLMs by enabling semantic operations that go beyond traditional SQL operators. While classical SQL excels at structural operations, LROs add capabilities including natural language understanding (filtering by "atmosphere"), semantic-aware data processing (joining by semantically similar company names like "Google LLC" and "Alphabet"), world knowledge extraction (finding all countries in Asia), reasoning (predicting stock prices), and tool usage (accessing external APIs).

LROs maintain *relational closure*: both input and output are relations, enabling composition with classical SQL operators in a unified plan. LROs do not replace classical SQL—they extend it. In hybrid query processing pipelines, LROs and classical SQL operators work together, each handling the part of the computation it does best. For example, in Figure 1, a query "Find the male attendee from technical companies with the highest influence" requires: (i) a traditional filter ("Gender" = "M"), (ii) an LLM-based filter (the "Company" is a "technical company") via `LLM_SELECT`; (iii) an LLM-based ordering (ranking "Name" by their "influence")

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828141

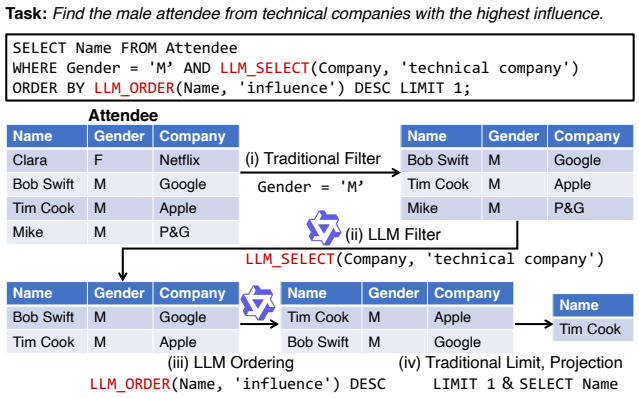

Figure 1: An example of integrating LROs with traditional SQL operators

via LLM_ORDER; (iv) traditional LIMIT and PROJECT operators to return the "Name" of the top-ranked attendee. This *hybrid execution model*—LROs for semantic steps, SQL for structural steps—is the hallmark of modern semantic query processing systems. Figure 1 visualizes this interleaving workflow.

Tutorial Motivation. A growing body of literature has proposed LROs across two main categories. *Standalone-LRO components* target data preparation tasks (*e.g.*, schema linking, entity matching, imputation, column generation) as end-to-end pipelines not designed for SQL composition, exemplified by ComEM [41], UniDM [33], and LLM-GDO [28]. *Multi-LRO systems* provide composable LRO suites for semantic query processing, orchestrating multiple LROs with classical SQL operators via automated planning or manual annotation, including LOTUS [32], BlendSQL [12], Palimpzest [23].

Despite this growing body of work, the field lacks a unified operator-level framework: (1) no systematic categorization of existing approaches; (2) no principled comparison of design choices; (3) no shared understanding of core challenges. Interestingly, we observe that the LRO abstraction provides a *critical unifying perspective* that reveals these diverse systems share a common foundation: they all accept relational inputs, perform semantic reasoning via LLMs guided by natural language, and produce relational outputs that preserve database closure properties. This unified view enables systematic categorization, principled comparison, and guidance for future LLM-enhanced database systems.

Tutorial Outline (1.5 hours). This tutorial provides a comprehensive introduction to LROs in three parts:

- **Part I—Foundations:** classical operator gaps, LRO definition, taxonomy covering operator logics and implementation variants;
- **Part II—Systems and Optimization:** standalone-LRO components, multi-LRO systems, query planning and optimization;
- **Part III—Evaluation and Frontiers:** benchmarks, evaluation and open challenges toward deeper LLM–database co-design.

Slides and a curated paper reading list will be released.

Related Tutorials. Recent tutorials cover adjacent topics: *LLM for Data Management* [20] (VLDB'24) broadly surveys LLM applications without focusing on the operator abstraction; *NL2SQL* [16, 27] (VLDB'23/'25) treats LLMs as query translators, not runtime operators; *DB Perspective on LLM Inference* [31] (VLDB'25) targets serving infrastructure; *Data Discovery* [1] (VLDB'25) focuses on join/union discovery. Our tutorial is the first to treat LROs as first-class citizens to explore the intersection fields of LLMs and databases, covering taxonomy, system design, and benchmarking.

Table 1: Unified LRO Taxonomy.

Logic	Gr.	Description	Representative Work
Select	row	Semantic filter	LOTUS [32], Aryn-Sycamore [3], BlendSQL [12], Palimpzest [23], DocETL [36], ThalamusDB [14]
	col	Schema linking	[15], RSL-SQL [7], C3 [11]
	table	Table identification	[15]
Match	cell	Semantic join	LOTUS [32], ThalamusDB [14], BlendSQL [12], Palimpzest [23], DocETL [36]
	row	Entity matching	ComEM [41], DataWrangle [29], Batcher [37], Jellyfish [43]
	col	Schema matching	Magneto [25], Agent-OM [34], DataWrangle [29]
Impute	cell	Missing value fill	UniDM [33], LLM-Prep [44]
	col	New col. gen.	BlendSQL [12], Palimpzest [23], SUQL [24], Aryn-Sycamore [3], DocETL [36], LLM-GDO [28], LOTUS [32], UniDM [33], HQDL [45], Binder [9]
	row	Record insertion	—
Cluster	row	Semantic grouping	LOTUS [32]
Order	row	Semantic ranking	LOTUS [32]

2 FOUNDATION AND TAXONOMY

LRO Definition and Distinction. Formally, as stated above, an LRO takes as input one or two relations $\mathcal{R}$, an operand granularity $g \in \{\text{cell}, \text{row}, \text{col}, \text{table}\}$, and a natural-language requirement l, producing output relation R' via LLM invocation: $\text{LRO}(\mathcal{R}, g, l) \mapsto R'$. The *relational closure* property—both input and output are relations—distinguishes LROs from Text-to-SQL systems (LLMs only generate SQL, no semantic operators at runtime) [21, 42], RAG (retrieves unstructured text, not relations) [19], and table understanding models (produce answers or embeddings, not relational outputs) [38].

Taxonomy Overview. The power of the LRO framework lies in its ability to provide a unifying lens through which to understand and categorize diverse LLM-enhanced systems. Our proposed three-dimensional taxonomy organizes LROs along: *operating logic* (what the operator computes, aligned with relational algebra), *operand granularity* (cell / row / col / table), and *implementation variant* (how elements are batched into LLM prompts). This taxonomy successfully encompasses a wide range of existing systems, revealing previously hidden commonalities across seemingly disparate approaches. For example, entity matching systems like ComEM [41] and DataWrangle [29] are unified as Match-Row operators, while semantic filtering in LOTUS [32] and schema linking [15] are unified under the Select operator at different granularities (row vs. column/table).

We now detail the five operating logics in Table 1. In the formulations below, $E_g(R)$ denotes the set of elements in R at granularity g (*e.g.*, all rows when g=row), and $e \models l$ means element e satisfies the natural-language requirement l as judged by the LLM.

Select. Retains elements that satisfy a semantic predicate. Formally, $\text{Select}(R, g, l) = \{e \in E_g(R) \mid e \models l\}$. Row-level Select (semantic filter) is supported by LOTUS [32], BlendSQL [12], ThalamusDB [14], Palimpzest [23], and DocETL [36]. Column/table-level Select (*i.e.*, schema linking) drives Text-to-SQL schema pruning and is addressed by [15] and RSL-SQL [7] as standalone components.

Match. Identifies semantically corresponding element pairs across two relations. Formally, $\text{Match}(R_1, R_2, g, l) = \{(e_1, e_2) \mid (e_1, e_2) \models l\}$. *Row-level Match (entity matching)* is the most studied standalone LRO: ComEM [41], DataWrangle [29], Batcher [37], and Jellyfish [43] achieve near-human accuracy on standard ER benchmarks. *Column-level Match (schema matching)* is addressed by Magneto [25] and Agent-OM [34] as standalone pipelines; DataWrangle handles both tasks in a unified framework. *Cell-level Match* (semantic table join) appears in BlendSQL [12] and ThalamusDB [14] as part of multi-LRO query engines.

Impute. Augments relations with LLM-generated content. *Cell-level* Impute fills missing values (UniDM [33], LLM-Prep [44], LLM-Forest [13]). *Column-level* Impute appends new computed columns—a key operator in semantic query engines (BlendSQL [12], SUQL [24], HQDL [45], LLM-GDO [28], Adda [26], SmartFeat [22]) and in data enrichment pipelines. *Row-level* Impute inserts records from LLM world knowledge.

Cluster. Groups elements into semantically coherent clusters. Formally, $\text{Cluster}(R, g, l) = \{C_1, \ldots, C_k\}$ where each $C_i \subseteq E_g(R)$ and the clusters are disjoint and exhaustive. For example, LOTUS's `sem_cluster_by` is the primary implementation. Column- and table-level variants (*e.g.*, schema segmentation) remain unexplored.

Order. Extends `ORDER BY` to qualitative criteria (*e.g.*, "rank by cultural influence"). LOTUS implements `sem_topk`.

Implementation Variants. Current LRO implementations can be categorized into three classes. `LLM-ALL` packs all candidates into one prompt ($O(1)$ calls): economical but sensitive to context length and positional bias. `LLM-ONE` processes each element individually ($O(n)$ for unary, $O(n^2)$ for binary operators): most widely adopted [23, 32] but costly at scale. `LLM-SEMI` (for Match) pairs one left element with all right elements ($O(n)$, used in BlendSQL).

The choice of variant depends on the operator logic. LROBench [4] reports that for Match, `LLM-ALL` outperforms `LLM-ONE` in both accuracy and cost (up to 40× cheaper), while `LLM-SEMI` offers a balanced alternative. For Cluster, `LLM-ALL` is preferred over `LLM-ONE`. For Select and Impute, the optimal choice depends on predicate complexity and data sparsity, and remains an open empirical question.

Order requires special treatment due to its unique computational structure. Four strategies exist: `LLM-ALL` (single-shot ordering), `LLM-PAIR` (pairwise comparisons, $O(n^2)$), `LLM-SORT` (quicksort with LLM comparison, $O(n \log n)$), and `LLM-SCORE` (assign scores then sort, $O(n)$). Empirically, `LLM-SCORE` achieves the best accuracy-cost trade-off [4].

3 LRO SYSTEM ARCHITECTURES

Standalone LRO components (*e.g.*, ComEM for entity matching, UniDM for imputation) apply a single LRO to a specific task and have been discussed alongside each operator in Section 2. This section focuses on *multi-LRO systems* that compose multiple LROs with classical SQL operators into unified query plans for *semantic query processing*—answering questions that require both relational reasoning and LLM semantic inference. Table 2 contrasts representative systems across key dimensions.

Table 2: Comparison of multi-LRO semantic query systems.

System	LRO Types	Planning	Optimization	Target
Binder [9]	Imp	Auto (LLM)	—	Struct. DB
SUQL [24]	Imp	Auto (LLM)	—	Conv. agent
HQDL [45]	Imp	Manual	—	DB enrich.
ThalamusDB [14]	Sel, Match	Manual	Batching	Multimedia
BlendSQL [12]	Sel, Match, Imp	Manual	—	Struct. DB
Aryn [3]	Sel, Imp	Auto (LLM)	—	Documents
Palimpzest [23]	Sel, Match, Imp	Manual	Cost model	Multimedia
DocETL [36]	Sel, Match, Imp	Auto (rewrite)	Rewriting	Documents
LOTUS/TAG [6, 32]	All 5	Manual	PAC bounds	Analytics
UQE [10]	Sel, Imp, Ord	Auto (agent)	—	Open-domain

Architecture Patterns. *SQL-extension systems* embed LROs as UDFs callable within SQL. BlendSQL [12] extends SQLite with `LLMMap`, `LLMValidate`, `LLMJoin`; ThalamusDB [14] integrates `NLFILTER`/`NLJOIN`; SUQL [24] adds `Answer` for conversational agents. *Declarative optimization systems* treat LRO pipelines as optimization problems: Palimpzest [23] and Abacus [35] enumerate physical plans via learned cost models; DocETL [36] rewrites plans automatically. *Agent-based systems* use LLM agents for dynamic planning: Aryn-Sycamore [3] builds multi-modal pipelines; UQE [10] decomposes NL questions into LRO subquery plans; CAESURA [40] extends to heterogeneous modalities.

Query Planning. Plans are obtained via (1) *manual annotation* (TAG [6], BlendSQL [12], HQDL [45], ThalamusDB [14]), (2) *LLM auto-planning* (Binder [9], Aryn [3]), or (3) *automated rewriting* (DocETL [36], Palimpzest [23]). LROBench [4] shows auto-planning systems score below 15% on complex queries *vs.* 86.67% for best-practice manual baselines, making planning the most critical unsolved problem.

Best Practices for Multi-LRO Systems. Building effective multi-LRO pipelines requires careful decisions on operator composition, input selection, and granularity determination. Practitioners must choose the most informative column(s) from R as semantic operator inputs, and construct precise, task-specific prompts—vague prompts l compound errors across stages. Granularity presents a key trade-off: decomposing a task into multiple LROs (e.g., Select followed by Impute) improves modularity and debuggability, while a single LLM call reduces latency but sacrifices error localizability. Critically, classical relational operators should be preferred over LROs wherever possible—predicates expressible by rules, structured lookups, or equi-joins should never be delegated to a semantic operator.

Since semantic operators are orders of magnitude more expensive than relational operators, minimizing cost and latency is a first-class concern in pipeline design. Key decisions include operator ordering, batching strategy, model selection, and implementation variant (`LLM-ALL` *vs.* `LLM-ONE`)—all of which significantly impact the accuracy–cost trade-off.

4 QUERY OPTIMIZATION FOR LRO SYSTEMS

LROs introduce new cost dimensions beyond classical I/O: token cost, LLM latency, and accuracy degradation from approximation.

Predicate Pushdown and Reordering. Cheap SQL predicates should execute before expensive LROs to reduce cardinality. Among LROs themselves, high-selectivity Select precedes Match. Semantic Integrity Constraints (SIC) [18] formalize inclusion/exclusion constraints on LRO outputs, enabling verification and propagation akin to classical optimization.

Semantic Batching. Packing multiple candidates per prompt amortizes overhead. Block-nested-loops batching reduces Match calls from $O(n^2)$ toward $O(n/B)$ for batch size B. Trummer [39] derives optimal batch sizes analytically as a function of predicate selectivity and model context length. ThalamusDB [14] and LOTUS [32] empirically confirm 40× cost reduction with no accuracy loss.

Model Cascades. Cascade strategies first apply a cheap model and escalate to a stronger one only for uncertain cases. FrugalGPT [8] provides general cascade optimization. LOTUS [32] applies this idea to several operators (Select, Match, Cluster, Order): a lightweight proxy model pre-filters candidates and importance sampling learns decision thresholds, providing PAC accuracy guarantees while accelerating single-operator execution by up to 1,000×.

Cost-Based Optimization. Palimpzest [23] and Abacus [35] frame LRO pipeline selection as Pareto optimization over accuracy–cost frontiers, scoring model choices, implementation variants (`LLM-ALL` *vs.* `LLM-ONE`), and caching strategies.

5 BENCHMARKS AND EVALUATION

Existing Benchmarks. *Text-to-SQL benchmarks* [21, 42] target natural language to SQL translation. They do not evaluate LROs because LLMs generate SQL queries without executing semantic operators at runtime.

SWAN [45] presents the first cross-domain benchmark for *beyond-database queries*—questions requiring information beyond stored data, answered via hybrid SQL-LLM integration. HQDL achieves 40.0% execution accuracy with GPT-4 Turbo on these queries, demonstrating both the potential and challenges of LLM-augmented querying. However, SWAN only considers Impute and lacks multi-LRO queries, resulting in limited LRO type coverage and query diversity.

TAG-Bench [6] designs semantically rich queries over relational databases that cannot be solved by classical SQL alone. It covers filtering and ranking tasks corresponding to LRO-Select and LRO-Order, but neglects Match and Cluster. Moreover, it includes no multi-LRO queries or query stratification.

SemBench [17] targets multi-modal semantic query processing across heterogeneous data sources. While it covers diverse semantic operations, its outputs violate relational closure, limiting its applicability to relational LRO evaluation.

LROBench [4] is designed specifically for operator-level LRO evaluation. It provides comprehensive coverage across four dimensions: (1) native relational I/O, (2) all five LRO logics, (3) implementation variants (`LLM-ALL` vs. `LLM-ONE`), and (4) multi-LRO query stratification. Key findings include: (1) `LLM-ALL` outperforms `LLM-ONE` for Match and Cluster; (2) `LLM-SCORE` dominates for Order; (3) chain-of-thought improves Select and Impute; (4) auto-planning systems score below 15% on complex queries.

Evaluation Challenges. Beyond benchmark design issues (query ambiguity, annotation errors), LRO evaluation faces system-level challenges: 1) LLM *non-determinism* necessitates multi-run aggregation; 2) *metric limitations* (Exact Match (EM) and Execution Accuracy (EX) false positives) and insufficient multi-dimensional coverage complicate assessment; 3) *data contamination* may inflate apparent performance.

6 OPEN CHALLENGES AND FUTURE DIRECTIONS

Reliable Automated Planning. Closing the gap between ∼15% auto-planning accuracy and 86.67% manual baselines is the most urgent problem; planners should fuse symbolic reasoning, execution feedback (*e.g.*, RL), and taxonomy-aware operator selection.

Unified Cost Model. One cost model spanning classical and LRO operators would unlock cross-operator reordering, predicate pushdown, and LLM-aware cardinality estimation.

Reliability and Adaptive Execution. Stochastic LLM outputs demand verification [18] and uncertainty-aware execution—still immature but critical for production. Model choice, LRO implementation variant, and batching should be chosen adaptively per query and data distribution.

Tighter LLM–Database Co-Design. The next step is deeper *co-design*: extend relational engines with LLM-backed components instead of replacing classical storage and optimization wholesale. Promising directions include (1) unified storage for heterogeneous and multi-modal data, (2) semantic access paths beyond B-trees and vanilla RAG [19], (3) adaptive assembly of prompt context from stored artifacts, and (4) holistic optimization jointly over storage, indexing, planning, and inference. LROs supply the bridging abstraction—relational closure at the boundaries, selective model invocation inside the plan.

7 PRESENTERS

Tianjing Zeng is a researcher at Alibaba Group. She is a lead contributor to LROBench and the LRO taxonomy.
Yin Lin is a researcher at Alibaba Group, focusing on LLM-based relational data processing.
Rong Zhu is a senior researcher at Alibaba Group. He is a corresponding author of the LROBench benchmark paper.
Yunxiang Su is a researcher at Alibaba Group. He is a co-lead contributor to LROBench and the LRO framework.
Zhongjun Ding is a researcher at Alibaba Group, focusing on LLM-enhanced relational data systems.
Bolin Ding is a senior research scientist at Alibaba Group, where he leads multiple research initiatives on database systems, large language models, and AI infrastructure.
Jingren Zhou is Senior Vice President at Alibaba, where he oversees research and engineering on large-scale AI systems and cloud database infrastructure.

ACKNOWLEDGMENTS

REFERENCES

[1] Ziawasch Abedjan, Mahdi Esmailoghli, and Sainyam Galhotra. 2025. Data Discovery in Data Lakes: Operations, Indexes, Systems. *Proc. VLDB Endow.* 18, 12 (2025), 5455–5458.

[2] Alibaba. 2025. Qwen3 Technical Report. *CoRR* abs/2505.09388 (2025).

[3] Eric Anderson, Jonathan Fritz, Austin Lee, Bohou Li, Mark Lindblad, Henry Lindeman, Alex Meyer, Parth Parmar, Tanvi Ranade, Mehul A. Shah, Benjamin Sowell, Dan Tecuci, Vinayak Thapliyal, and Matt Welsh. 2025. The Design of an LLM-powered Unstructured Analytics System. In *CIDR*.

[4] [Authors Anonymized]. 2025. LROBench: A Comprehensive Benchmark for LLM-Enhanced Relational Operators. Under review.

[5] Anthropic. 2025. Introducing Claude Sonnet 4.5. (2025). https://www.anthropic.com/news/claude-sonnet-4-5

[6] Asim Biswal, Siddharth Jha, Carlos Guestrin, Matei Zaharia, Joseph E. Gonzalez, Amog Kamsetty, Shu Liu, and Liana Patel. 2025. Text2SQL is Not Enough: Unifying AI and Databases with TAG. In *CIDR*.

[7] Zhenbiao Cao, Yuanlei Zheng, Zhihao Fan, Xiaojin Zhang, Wei Chen, and Xiang Bai. 2024. RSL-SQL: Robust Schema Linking in Text-to-SQL Generation. *CoRR* abs/2411.00073.

[8] Lingjiao Chen, Matei Zaharia, and James Zou. 2024. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *Trans. Mach. Learn. Res.* 2024 (2024). https://openreview.net/forum?id=cSimKw5p6R

[9] Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding Language Models in Symbolic Languages. In *ICLR*.

[10] Hanjun Dai, Bethany Yixin Wang, Xingchen Wan, Bo Dai, Sherry Yang, Azade Nova, Pengcheng Yin, Phitchaya Mangpo Phothilimthana, Charles Sutton, and Dale Schuurmans. 2024. UQE: a query engine for unstructured databases. In *Proceedings of the 38th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '24)*. Curran Associates Inc., Red Hook, NY, USA, Article 938, 32 pages.

[11] Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Lu Chen, Jinshu Lin, and Dongfang Lou. 2023. C3: Zero-shot Text-to-SQL with ChatGPT. *CoRR* abs/2307.07306 (2023).

[12] Parker Glenn, Parag Dakle, Liang Wang, and Preethi Raghavan. 2024. BlendSQL: A Scalable Dialect for Unifying Hybrid Question Answering in Relational Algebra. In *ACL (Findings)*. 453–466.

[13] Xinru He, Yikun Ban, Jiaru Zou, Tianxin Wei, Curtiss B. Cook, and Jingrui He. 2025. LLM-Forest: Ensemble Learning of LLMs with Graph-Augmented Prompts for Data Imputation. In *ACL (Findings)*. 6921–6936.

[14] Saehan Jo and Immanuel Trummer. 2024. ThalamusDB: Approximate Query Processing on Multi-Modal Data. *Proc. ACM Manag. Data* 2, 3, 186.

[15] George Katsogiannis-Meimarakis, Katsiaryna Mirylenka, Paolo Scotton, Francesco Fusco, and Abdel Labbi. 2026. In-depth Analysis of LLM-based Schema Linking. In *EDBT*. 117–130.

[16] George Katsogiannis-Meimarakis, Mike Xydas, and Georgia Koutrika. 2023. Natural Language Interfaces for Databases with Deep Learning. *Proc. VLDB Endow.* 16, 12 (2023), 3878–3881.

[17] Jiale Lao, Andreas Zimmerer, Olga Ovcharenko, Tianji Cong, Matthew Russo, Gerardo Vitagliano, Michael Cochez, Fatma Özcan, Gautam Gupta, Thibaud Hottelier, H. V. Jagadish, Kris Kissel, Sebastian Schelter, Andreas Kipf, and Immanuel Trummer. 2026. SemBench: A Benchmark for Semantic Query Processing Engines. *Proc. VLDB Endow.* (2026).

[18] Alexander W. Lee, Justin Chan, Michael Fu, Nicolas Kim, Akshay Mehta, Deepti Raghavan, and Ugur Cetintemel. 2025. Semantic Integrity Constraints: Declarative Guardrails for AI-Augmented Data Processing Systems. *Proc. VLDB Endow.* 18, 11 (2025), 4073–4080.

[19] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS*.

[20] Guoliang Li, Xuanhe Zhou, and Xinyang Zhao. 2024. LLM for Data Management. *Proc. VLDB Endow.* 17, 12 (2024), 4213–4216.

[21] 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 Chen-Chuan 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 *NeurIPS*.

[22] Yin Lin, Bolin Ding, H. V. Jagadish, and Jingren Zhou. 2024. SMARTFEAT: Efficient Feature Construction through Feature-Level Foundation Model Interactions. In *CIDR*.

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

[24] Shicheng Liu, Jialiang Xu, Wesley Tjangnaka, Sina J. Semnani, Chen Jie Yu, and Monica Lam. 2024. SUQL: Conversational Search over Structured and Unstructured Data with Large Language Models. In *NAACL-HLT (Findings)*. 4535–4555.

[25] Yurong Liu, Eduardo Peña, Aécio S. R. Santos, Eden Wu, and Juliana Freire. 2025. Magneto: Combining Small and Large Language Models for Schema Matching. *Proc. VLDB Endow.* 18, 8 (2025), 2681–2694. https://doi.org/10.14778/3742728.3742757

[26] Kuan Lu, Zhihui Yang, Sai Wu, Ruichen Xia, Dongxiang Zhang, and Gang Chen. 2025. Adda: Towards Efficient in-Database Feature Generation via LLM-based Agents. *Proc. ACM Manag. Data* 3, 3 (2025), 125:1–125:27. https://doi.org/10.1145/3725262

[27] Yuyu Luo, Guoliang Li, Ju Fan, Chengliang Chai, and Nan Tang. 2025. Natural Language to SQL: State of the Art and Open Problems. *Proc. VLDB Endow.* 18, 12 (2025), 5466–5469.

[28] Luyi Ma, Nikhil Thakurdesai, Jiao Chen, Jianpeng Xu, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2023. LLMs with User-defined Prompts as Generic Data Operators for Reliable Data Processing. In *IEEE Big Data*. 3144–3148.

[29] Avanika Narayan, Ines Chami, Laurel J. Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *Proc. VLDB Endow.* 16, 4 (2022), 738–746.

[30] OpenAI. 2025. OpenAI GPT-5 System Card. *CoRR* (2025).

[31] James Pan and Guoliang Li. 2025. Database Perspective on LLM Inference Systems. *Proc. VLDB Endow.* 18, 12 (2025), 5504–5507.

[32] Liana Patel, Siddharth Jha, Melissa Z. Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2025. Semantic Operators and Their Optimization: Towards AI-Based Data Analytics with Accuracy Guarantees. *Proc. VLDB Endow.* 18, 11 (2025), 4171–4184.

[33] Yichen Qian, Yongyi He, Rong Zhu, Jintao Huang, Zhijian Ma, Haibin Wang, Yaohua Wang, Xiuyu Sun, Defu Lian, Bolin Ding, and Jingren Zhou. 2024. UniDM: A Unified Framework for Data Manipulation with Large Language Models. In *MLSys*.

[34] Zhangcheng Qiang, Weiqing Wang, and Kerry Taylor. 2024. Agent-OM: Leveraging LLM Agents for Ontology Matching. *Proc. VLDB Endow.* 18, 3 (2024), 516–529. https://doi.org/10.14778/3712221.3712222

[35] Matthew Russo, Sivaprasad Sudhir, Gerardo Vitagliano, Chunwei Liu, Tim Kraska, Samuel Madden, and Michael J. Cafarella. 2025. Abacus: A Cost-Based Optimizer for Semantic Operator Systems. *CoRR* abs/2505.14661 (2025).

[36] 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, 3035–3048.

[37] Aaron Steiner, Ralph Peeters, and Christian Bizer. 2025. Fine-Tuning Large Language Models for Entity Matching. In *ICDEW*. 9–17.

[38] Aofeng Su, Aowen Wang, Chao Ye, Chen Zhou, Ga Zhang, Gang Chen, Guangcheng Zhu, Haobo Wang, Haokai Xu, Hao Chen, Haoze Li, Haoxuan Lan, Jiaming Tian, Jing Yuan, Junbo Zhao, Junlin Zhou, Kaizhe Shou, Liangyu Zha, Lin Long, Liyao Li, Pengzuo Wu, Qi Zhang, Qingyi Huang, Saisai Yang, Tao Zhang, Wentao Ye, Wufang Zhu, Xiaomeng Hu, Xijun Gu, Xinjie Sun, Xiang Li, Yuhang Yang, and Zhiqing Xiao. 2024. TableGPT2: A Large Multimodal Model with Tabular Data Integration. *CoRR* abs/2411.02059 (2024).

[39] Immanuel Trummer. 2025. Implementing Semantic Join Operators Efficiently. arXiv:2510.08489

[40] Matthias Urban and Carsten Binnig. 2024. CAESURA: Language Models as Multi-Modal Query Planners. In *CIDR*.

[41] Tianshu Wang, Xiaoyang Chen, Hongyu Lin, Xuanang Chen, Xianpei Han, Le Sun, Hao Wang, and Zhenyu Zeng. 2025. Match, Compare, or Select? An Investigation of Large Language Models for Entity Matching. In *COLING*. 96–109.

[42] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *EMNLP*. Association for Computational Linguistics, 3911–3921.

[43] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2024. Jellyfish: Instruction-Tuning Local Large Language Models for Data Preprocessing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 8754–8782. https://doi.org/10.18653/v1/2024.emnlp-main.497

[44] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2024. Large Language Models as Data Preprocessors. In *TaDA Workshop, VLDB*.

[45] Fuheng Zhao, Divyakant Agrawal, and Amr El Abbadi. 2025. Hybrid Querying Over Relational Databases and Large Language Models. In *CIDR*.