Letter from the Special Issue Editors

The rapid progress of large language models has created a new class of data-management workloads. Users increasingly expect data systems to operate not only over relational tables, but also over documents, spreadsheets, images, logs, scientific articles, contracts, transcripts, and other heterogeneous collections. They also expect to express intent in natural language, and to obtain answers that are not only fluent, but grounded, auditable, and cost-effective. This special issue examines how the data engineering community is responding to this shift.

The articles in this issue focus on an emerging research area that we may broadly call semantic data systems: systems that treat language-based interpretation, extraction, ranking, joining, aggregation, and reasoning as first-class data-processing tasks. This perspective brings familiar database questions into a new setting. How should users express semantic intent? What are the logical operators and programming abstractions? How should systems choose among alternative model calls, prompts, indexes, caches, approximations, and execution strategies? How should quality, cost, latency, and provenance be exposed to users? And how should we evaluate systems whose behavior depends on models, data distributions, and user-defined semantics?

The contributions collected here approach these questions from complementary angles. Vitagliano, Russo, Cafarella, Madden, and Kraska discuss semantic query plan optimization for AI-powered analytics. Their article describes how semantic operators can support declarative programs over unstructured and multimodal data, and how systems such as Palimpzest, Abacus, and Carnot move from fixed semantic plans toward optimized agentic analytics. A central theme is that optimization must account not only for traditional resource costs, but also for model quality, uncertainty, latency, and monetary cost.

Shankar and Parameswaran take a user-centered view of LLM-powered data systems. Drawing on their experience with DocETL and DocWrangler, they show that users rarely arrive with a final, well-specified query. Instead, they explore data, revise prompts, validate intermediate results, and construct bespoke operators that are specific to their documents and domains. Their article is an important reminder that semantic data systems must be designed for iterative authoring and validation, not only for one-shot execution.

Patel, Guestrin, and Zaharia present the semantic operator model and the LOTUS system. Their work formalizes natural-language operators such as semantic filters, joins, top-k, grouping, and aggregation, and studies how such operators can be optimized while preserving accuracy guarantees with respect to reference implementations. This line of work highlights the need for model-data independence: users should be able to specify high-level semantic transformations while systems decide how to execute them accurately and efficiently.

Sanmartino, Urban, Binnig, and Papotti explore the physical design space of LLM data systems. Their article argues that the performance of semantic workloads depends heavily on execution-time and storage-time design choices, including caching, compression, batching, reuse, and materialized semantic representations. As with classical database systems, logical expressiveness alone is not enough: the physical layer determines whether workloads can scale in practice.

Liskowski, Zhao, Han, Datta, and Tsirogiannis study compositional online learning for semantic data processing systems. Their article considers how systems can adapt during execution, learning from feedback, intermediate results, and workload structure. This perspective is particularly relevant when semantic predicates and transformations are uncertain, task-specific, and expensive to evaluate.

Finally, Trummer introduces ThalamusDB and the idea of semantic approximate query processing. The article connects semantic query execution with approximation, sampling, and uncertainty-aware answers. As semantic workloads grow in scale and cost, approximate execution will be essential: users often need useful answers quickly, together with an understanding of confidence, error, and the cost of further refinement.

Taken together, these articles show that LLM-native data systems are not simply existing data systems with model calls added as external functions. Rather, they treat large language models as first-class citizens in the design of declarative query interfaces, logical operators, physical execution strategies, and interactive analysis workflows. Across the papers, LLMs appear in different roles: as implementations of semantic operators, as components of cost- and quality-aware query plans, as agents that rewrite and evaluate document pipelines, as building blocks for physical design and reuse, and as adaptive components in online and approximate processing.

Several common abstractions emerge from these contributions. Semantic operators provide a declarative way to express language-based filtering, mapping, joining, ranking, grouping, and aggregation over mixed structured and unstructured data. Model-in-the-loop planning makes optimization depend not only on CPU, memory, and I/O, but also on model selection, prompt design, token cost, latency, quality, and uncertainty. Agentic pipelines extend query processing beyond fixed plans, allowing systems to search, compute, revise, validate, and reuse intermediate results. Physical design techniques such as caching, compression, batching, materialization, indexing, and approximate execution then determine whether these abstractions can scale in practice.

The issue also surfaces a set of open challenges for the next generation of LLM-native data systems. Quality must be measured together with cost, latency, reproducibility, and robustness to model drift. Provenance and auditability must become part of the execution model, so that users can inspect not only the final answer, but also the evidence, prompts, model choices, intermediate decisions, and uncertainty behind it. Continual learning and feedback-driven adaptation are needed because semantic predicates and user intent evolve over time. Multi-tenant execution raises questions about isolation, fairness, resource allocation, and predictable cost. Finally, the community needs benchmarks and evaluation standards that reflect real workloads: multimodal data, bespoke user-defined operators, iterative query development, and deployment constraints.

We hope this special issue helps define LLM-native data systems as an emerging area at the intersection of data management, machine learning, and human-centered analytics. The challenge ahead is to build systems that make LLMs usable as reliable data-processing components: declarative enough for users to express intent, optimized enough to run at scale, transparent enough to support inspection and trust, and flexible enough to handle the heterogeneous and evolving data found in real applications.

We thank all authors for their contributions. We also thank Haixun Wang for the opportunity to organize this special issue and for his guidance throughout the process.

Carsten Binnig, Paolo Papotti, Gerardo Vitagliano
TU Darmstadt, EURECOM, MIT CSAIL