Towards a Neural Database Execution Engine

Christos Tsapelas[1,2]

Supervised by Georgia Koutrika[2]
[1]Department of Informatics and Telecommunications, National and Kapodistrian University of Athens
[2]Archimedes, Athena Research Center, Greece

Abstract
Recent advances in natural language understanding have heightened the interest in AI systems capable of answering queries across multiple data modalities, such as structured database tables and unstructured text. Current approaches typically rely on Large Language Models (LLMs) to facilitate queries between these modalities, which incurs substantial computational costs and often yields suboptimal performance. To this direction, this research introduces a novel query execution engine designed to bridge diverse data modalities, leveraging the high-efficiency querying capabilities of database systems with the advanced reasoning capacities of LLMs. This paper presents a prototype architecture for such a multi-modal database system, detailing its core components and their functionalities to demonstrate how it can achieve effective, scalable query processing across structured and unstructured data.

Keywords
database systems, large language models, memory networks, hybrid query execution, virtual knowledge bases

1. Introduction

The evolution of modern data warehouses has introduced unprecedented challenges and opportunities, with data volumes now encompassing multiple modalities, such as structured table data, unstructured text, and images. Each data type possesses a unique structure, necessitating tailored querying methods that effectively harness the properties of each modality. However, despite these advances, existing systems struggle to generalize queries across multiple modalities, presenting a key limitation in addressing the needs of diverse, cross-modal data integration tasks.

Database management systems (DBMS) excel in performing rapid, efficient, and precise queries on extremely large data volumes at scale. However, their primary focus remains on exact computation at scale, with limited reasoning capabilities [1]. In contrast, Large Language Models (LLMs) excel at processing natural language data across massive textual corpora, offering logical reasoning over unstructured data due to the model's ability to embed knowledge within its weights [2]. This distinction highlights a crucial gap between traditional DBMS architectures and the reasoning and flexibility capabilities that LLMs bring to unstructured data processing.

Numerous contemporary applications demand complex queries that integrate information across multiple modalities [3]. Current dominant approaches for multi-modal data integration include retrieval-augmented generation (RAG), similarity-based search, and Text2SQL. These techniques, however, exhibit limitations in both the diversity of query types they can accommodate and their query execution performance. Text2SQL methods, for instance, are effective for natural language queries that have a direct SQL equivalent, whereas RAG systems are constrained to point lookups involving only a limited number of records, requiring an LLM to execute the join operation.

To this direction, my doctoral research seeks to bridge the gap between the approaches of LLMs and traditional database systems to enable efficient, flexible hybrid search queries. The objective is to develop a prototype neural query execution engine equipped with novel algorithms for efficient data access and join operations, leveraging the strengths of both learned models and traditional database methods, for rapid and precise query execution across multiple data modalities. The proposed neural execution engine seeks to empower database systems with the flexibility to handle diverse data modalities and complex query types, addressing a critical need in the field of data management and paving the way for next-generation data retrieval solutions.

2. Related Work

Recent works in natural language understanding require retrieving and reasoning, like question answering. For such knowledge-intensive tasks, it is required to assimilate information from different sections of large different inputs such as books and article collections [4]. To this direction, the notions of Virtual Knowledge Bases (VKBs) [5, 6, 4] and Memory Networks [7, 8] are proposed, in which entity mentions in text are transformed into dense representations to represent properties or relations expressed with text passages.

Moreover, the advanced reasoning capabilities of LLMs using RAG in question answering, has emerged a new area of research where the system takes as input both structured and unstructured data for reasoning over different modalities [1, 9, 3, 10, 11, 12, 13, 14], or use the LLMs as a query engine to pose SQL queries [2].

Simultaneously, the database community has introduced an innovative research direction involving learning-based techniques to enhance query execution. Advances such as learned sorting [15] and learned scans and joins algorithms [16, 17] have demonstrated highly promising results in optimizing traditional query processes. These methods indicate that machine learning techniques can significantly improve fundamental database operations, reinforcing the potential for a hybrid approach that integrates both database and LLM methodologies.

3. Research Questions

The main purpose of this research proposal is the development of a prototype execution engine able to execute queries combining different data modalities. The implementation of such a novel system, emerges a set of research questions:

Published in the Proceedings of the Workshops of the EDBT/ICDT 2025 Joint Conference (March 25-28, 2025), Barcelona, Spain

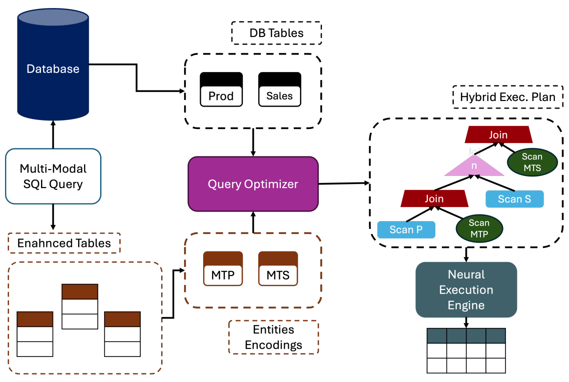

Figure 1: Architecture of Neural Execution Engine. The system accepts a multi-modal SQL query and fetches the related database and mention tables. Next the optimizer generates a hybrid execution plan with: database scans (rectangles), mention tables scans (circles), database join (triangle) database-mention table join (trapezoid)

RQ1 How can structured database tables and unstructured data sources, like text documents, be effectively associated to form a unified querying framework?

RQ2 How can dense vector representations be constructed to preserve both the semantic richness of text and the structural integrity of database entities?

RQ3 What new operators are required and how can database operators be adapted to process queries involving structured and unstructured data?

RQ4 How can a cost-based query optimizer be designed to generate efficient execution plans for hybrid queries involving structured and unstructured data?

RQ5 What techniques can be employed to ensure the query engine scales efficiently for large datasets across multiple modalities?

3.1. Research Opportunities

Building upon the related work, the proposed query engine represents a significant advancement in addressing the previously outlined research questions.

Virtual Knowledge Bases (VKBs) generate dense representations of real-world entities, such as those found within Wikipedia, to enable querying. However, these representations have not been applied within the context of database systems. A key component of this research involves establishing connections between database entities, as defined by the data model of each database, and external text corpora or additional modalities, such as images.

Furthermore, the current state-of-the-art approach for integrating multiple data modalities relies on Multi-Modal Large Language Models (MLLMs). This methodology typically employs large-scale LLMs to process queries, a strategy that is computationally expensive and constrained by the input size limitations inherent to LLMs.

The primary objective of this research is to enable efficient execution of multi-modal queries capable of managing large-scale data in a manner aligned with traditional database systems. To achieve this, the research will extend conventional database operators, such as scans and joins, by developing novel implementation algorithms designed to process diverse data modalities.

4. A Prototype of A Neural Database Engine

In this section, an overview of the proposed neural database execution engine is provided. Suppose the example query: *"Find all customers who purchased 'Product X' and had positive experience regarding the quality of the product, within the past six months."*. This query is transformed into a multi-modal SQL query and sent to the engine for execution. The example query will assist to describe several aspects of the proposed system, along with the execution flow of a hybrid query between database tables and a text corpora.

In Figure [1], we present the architecture of our query engine. The example query is parsed and the system fetches the *Product* and *Sales* database tables and their related mention tables. Then, the optimizer is invoked to generate an optimal execution plan for the given query. The optimizer faces many challenges like selecting the appropriate scan and join operators within and across modalities, while predicting their optimal order in the execution plan. In Figure 1, different physical operators are separated to make clear the different processing steps. Finally, the generated execution plan is submitted to the neural engine for execution.

Before the query engine can execute queries across both data modalities, a preparatory phase, referred to as *mention tables construction*, is required. In the provided example, mentions of *'Product X'* must be recognized within text passages. In this phase, the system generates a series of key-value (KV) tables that bridge the information within database tables and text documents stored in blob storage.

The objective of these KV tables is to create dense vector representations of entities (keys) that encapsulate the knowledge embedded in the text corpus (values). These representations are structured to seamlessly integrate with a Transformer model, enabling efficient and effective processing by the query engine in subsequent stages.

Upon initializing these learned tables, the execution engine is ready to process queries. When a query is posed, the system parses it and generates an optimal execution plan. The plan selection process resembles that of traditional database systems, wherein the optimizer explores the space of possible execution plans and evaluates candidate plans based on a cost model.

Given the hybrid nature of the proposed query engine, which supports queries across multiple data modalities, it is necessary to define new hybrid operators capable of handling data from both structured and unstructured sources, like scans, projections, joins etc. These operators are designed to facilitate seamless integration and processing of data across the diverse modalities enclosed by the system.

During the next subsections, the main components of the proposed query engine are describes. Initially, the process of *mention tables* is described, a methodology to associate table data with the available text corpus. Next, the core of the execution engine is detailed, focusing on the needed operators and the query optimizer of the system.

4.1. Mention Tables

As previously noted, it is essential to establish associations between data from database tables and the available text documents, defining the specific types of information to be retrieved and assimilated across these data sources [4]. The database schema provides a structured representation of the entities within the database, with clearly defined properties for each entity type.

Thus, an initial processing step is proposed between the different data sources, where each passage in text documents is annotated with the main entities (fact tables) from the database and we highlight entity mentions in the passage with special tokens. Figure [2] shows the construction of mention tables. Representations of these tokens are later used to generate entity encodings.

The goal of *mention tables* is to gather these database entity encodings into matrices constructing key-value stores containing the dense vector representations for each entity in text documents forming a virtual knowledge base of the available text documents, like [7, 4, 18, 8]

4.2. Neural Query Execution

To extend the querying capabilities of traditional database systems across multiple data modalities, it becomes necessary to adapt and expand conventional database operators to effectively manage and process data from both structured and unstructured sources.

Within a neural database system, operators are categorized into two distinct types: **a) single-modal operators**, which are designed to process a single data modality (e.g., structured table data or unstructured text passages), such as scan operations, and **b) multi-modal operators**, which are capable of processing and associating inputs across multiple data modalities, such as join and aggregation operations, that integrate information from both structured and unstructured sources. Thus, there is an emerging need to extend traditional relational algebra used in database systems, to describe the new neural operators for different modalities.

In database systems, all operators in relational algebra take as input a relation and the output is the result of the operator applied on the input relation, which is again a relation. In the case of the proposed query engine, we need to define the neural operators regarding the scan, filter and project of mention tables, as well the join implementation between the a database table and a mention table.

Scan Mention Tables Scan operations over mention tables enable efficient querying and processing of entity-associated text passages. These include *entity retrieval scans*, which extract passages linked to specific entities, like *'Product X'* in example query, and *mention highlight scans*, which identify all occurrences of targeted entities. *Contextual similarity scans* rank passages based on semantic relevance to a query vector, while *entity-to-entity relationship scans* reveal co-occurrences within text. Additional methods, such as *aggregated entity statistics scans* and *temporal or categorical filters*, allow for deeper insights by analyzing mention frequency, context diversity, or filtering by specific attributes. Advanced operations, like *neighborhood scans* for exploring entity connections and *multi-modal entity scans* for linking to database tables, further enhance the querying capabilities of mention tables. These methods leverage the dense vector representations of entities to facilitate robust and flexible data exploration.

Join Mention & Database Tables For join operations in the proposed query engine, two types of joins are possible: **a)** joins within mention tables and **b)** hybrid joins between database and mention tables.

Joins within mention tables enable the discovery of relationships between entities based on shared textual contexts or semantic relevance. These include *entity co-occurrence joins*, which retrieve passages where multiple entities are mentioned together, and *contextual similarity joins*, which link entities based on the similarity of their dense vectors and *passage-level joins*, which connect text passages that reference related entities, enabling richer narratives

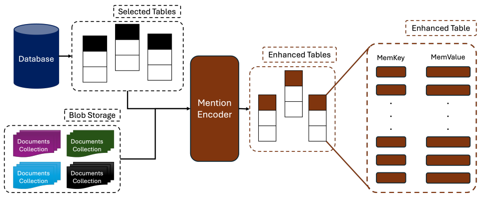

Figure 2: Mention Tables represent entities inside database tables into vector representations.

Joins between mention tables and database tables integrate structured and unstructured data to provide a unified query interface. *Entity-ID joins* link entities in mention tables to their corresponding database records, while *property-based joins* combine entities based on shared attributes, such as linking customer mentions with their structured profiles. In the example query, the first join of the execution plan is an *Entity-ID* join between *Product* database and mention tables. *Aggregated knowledge joins* enrich database records with insights from text passages, and *hybrid semantic joins* bridge structured relationships in the database with semantic similarity in mention tables, enabling advanced querying across diverse data modalities.

While the traditional database operators are well-defined, the landscape of neural operators for execution is an active area of research. There are efforts from the database community enhancing traditional operators [15, 16, 13] with neural models for faster query processing, while there are approaches that propose learned operators, e.g learned scans and joins [17]. Further, the proposed query engine can utilize the reasoning capabilities of LLMs to provide reasoning or summaries on the results of the aforementioned operators at the end of query results or on some intermediate step of query processing. In the provided example, the LLM is invoked to evaluate all reviews text passages per product.

4.3. Query Optimization

The query optimizer for the proposed neural execution engine bridges the gap between structured and unstructured data processing, enabling efficient query execution across database tables and mention tables. By integrating traditional database strategies with neural processing, it ensures scalability and adaptability for hybrid, multi-modal queries.

A core characteristic of the optimizer is its cost-based approach, which evaluates potential query execution plans based on resource consumption, including computation time, cardinality estimation on both database and mention tables, memory usage, and I/O overhead. For neural operators, additional factors such as the cost of vector similarity computations and embedding generation are built-in the cost model, ensuring an accurate evaluation of query plans.

Moreover, two very important aspects are the query decomposition and the cross-modal data flow. The optimizer decomposes complex queries into into modality-specific sub-queries, ensuring efficient processing of structured and unstructured data by simultaneously selecting the most appropriate operators, as well as, their optimal order in the execution plan. In this context, the proposed execution plan should minimizes redundant computations and intermediate results, optimizing data transfer between operators, for efficient cross-modal data flow.

Finally, the optimizer is designed to adapt to dynamic query workloads and evolving data characteristics by supporting runtime re-optimization. It monitors operator performance during execution and adjusts plans as needed. Furthermore, it integrates pre-trained or fine-tuned neural models for unstructured data processing, ensuring their effective and efficient use in query execution.

5. Conclusions and Future Work

This paper presents a prototype query engine designed to execute queries across multiple data modalities efficiently and at scale. A central proposition of this work is the initial association of key entities from the database with their corresponding references within the text corpus. The system then constructs mention tables, key-value tables containing dense vector representations of entities and their related textual passages. The results of this approach have the potential to inspire new algorithms that link structured database information with external unstructured data sources.

Furthermore, this novel tabular representation of unstructured text enables the development of specialized operators that the query engine must support. The design and implementation of these operators establish the foundational components of the envisioned query engine. Additionally, the integration of these operators calls for the development of a new generation of query optimizers capable of generating efficient execution plans across both structured and unstructured data modalities. This research lays the fundamentals for a novel approach to querying multi-modal data and opens new avenues for future exploration in hybrid query optimization and execution strategies.

Acknowledgments

This work has been partially supported by DataGEMS, funded by the European Union's Horizon Europe Research and Innovation programme, under grant agreement No 101188416 and by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program.

References

[1] A. Biswal, L. Patel, S. Jha, A. Kamsetty, S. Liu, J. E. Gonzalez, C. Guestrin, M. Zaharia, Text2sql is not enough: Unifying ai and databases with tag, arXiv preprint arXiv:2408.14717 (2024).

[2] M. Saeed, N. De Cao, P. Papotti, Querying large language models with sql, arXiv preprint arXiv:2304.00472 (2023).

[3] L. Patel, S. Jha, C. Guestrin, M. Zaharia, Lotus: Enabling semantic queries with llms over tables of unstructured and structured data, arXiv preprint arXiv:2407.11418 (2024).

[4] Y. Zemlyanskiy, J. Ainslie, M. de Jong, P. Pham, I. Eckstein, F. Sha, Readtwice: Reading very large documents with memories, in: Proceedings of NAACL, 2021.

[5] B. AlKhamissi, M. Li, A. Celikyilmaz, M. Diab, M. Ghazvininejad, A review on language models as knowledge bases, 2022. URL: https://arxiv.org/abs/2204.06031. arXiv:2204.06031.

[6] M. de Jong, Y. Zemlyanskiy, N. A. FitzGerald, F. Sha, W. W. Cohen, Mention memory: incorporating textual knowledge into transformers through entity mention attention, in: 10th International Conference on Learning Representations, ICLR 2022, 2022.

[7] S. Sukhbaatar, J. Weston, R. Fergus, et al., End-to-end memory networks, Advances in neural information processing systems 28 (2015).

[8] Z. Zhong, T. Lei, D. Chen, Training language models with memory augmentation, in: Proceedings of the 2022 Conference on EMNLP, Association for Computational Linguistics, 2022, pp. 5657–5673. URL: https://aclanthology.org/2022.emnlp-main.382/. doi:10.18653/v1/2022.emnlp-main.382.

[9] L. Patel, P. Kraft, C. Guestrin, M. Zaharia, Acorn: Performant and predicate-agnostic search over vector embeddings and structured data, Proceedings of the ACM on Management of Data 2 (2024) 1–27.

[10] A. Dargahi Nobari, D. Rafiei, Dtt: An example-driven tabular transformer for joinability by leveraging large language models, Proceedings of the ACM on Management of Data (SIGMOD) 2 (2024). URL: https://doi.org/10.1145/3639279. doi:10.1145/3639279.

[11] G. Badaro, M. Saeed, P. Papotti, Transformers for tabular data representation: A survey of models and applications, Transactions of the Association for Computational Linguistics 11 (2023) 227–249. URL: https://aclanthology.org/2023.tacl-1.14/. doi:10.1162/tacl_a_00544.

[12] M. J. Cafarella, C. Re, D. Suciu, O. Etzioni, M. Banko, Structured querying of web text, in: 3rd Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, California, USA, 2007.

[13] C. Liu, M. Russo, M. Cafarella, L. Cao, P. B. Chen, Z. Chen, M. Franklin, T. Kraska, S. Madden, G. Vitagliano, A declarative system for optimizing ai workloads, arXiv preprint arXiv:2405.14696 (2024).

[14] Y. Lin, M. Hulsebos, R. Ma, S. Shankar, S. Zeigham, A. G. Parameswaran, E. Wu, Towards accurate and efficient document analytics with large language models, 2024. URL: https://arxiv.org/abs/2405.04674. arXiv:2405.04674.

[15] A. Kristo, K. Vaidya, U. Çetintemel, S. Misra, T. Kraska, The case for a learned sorting algorithm, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, SIGMOD '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 1001–1016. URL: https://doi.org/10.1145/3318464.3389752. doi:10.1145/3318464.3389752.

[16] I. Sabek, T. Kraska, The case for learned in-memory joins, Proc. VLDB Endow. 16 (2023) 1749–1762. URL: https://doi.org/10.14778/3587136.3587148. doi:10.14778/3587136.3587148.

[17] M. Urban, C. Binnig, Eleet: Efficient learned query execution over text and tables, volume 17, VLDB Endowment, 2024, pp. 4867–4880.

[18] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, J. Weston, Key-value memory networks for directly reading documents, in: Proceedings of the 2016 Conference on EMNLP, Association for Computational Linguistics, Austin, Texas, 2016. URL: https://aclanthology.org/D16-1147. doi:10.18653/v1/D16-1147.