Master's thesis

Master's Programme in Computer Science

LLM-Enhanced Data Management in Multi-Model Databases

Tianhao Yang

May 18, 2025

FACULTY OF SCIENCE

UNIVERSITY OF HELSINKI

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki,Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Koulutusohjelma — Utbildningsprogram — Study programme Master's Programme in Computer Science

Tekijä — Författare — Author

Tianhao Yang

Työn nimi — Arbetets titel — Title

LLM-Enhanced Data Management in Multi-Model Databases

Ohjaajat — Handledare — Supervisors

Prof. Jiaheng Lu

Työn laji — Arbetets art — Level Master's thesis	Aika — Datum — Month and year May 18, 2025	Sivumäärä — Sidoantal — Number of pages 43 pages

Tiivistelmä — Referat — Abstract

Modern organizations are increasingly confronted with a vast variety of heterogeneous data, including structured, semi-structured, and unstructured formats. Managing such diverse data types presents significant challenges. Multi-model databases (MMDBs) have been developed to address this issue by supporting multiple data models within a unified platform. At the same time, large language models (LLMs) have made substantial progress in natural language processing, offering new possibilities for intelligent interaction with complex data systems.

This survey explores the intersection of MMDBs and LLMs and provides a full review of the ways in which LLMs may be used to advance multi-model data management. The study begins with an overview of MMDB architectures and the capabilities of LLMs, followed by the proposal of a conceptual framework for their integration. Some of the most important application areas are discussed through exemplar case studies, and they include natural-language querying, data integration, database optimization, content generation, and privacy-aware processing. The strengths and weaknesses of existing solutions are evaluated critically and the most important research challenges identified as hallucination mitigation, processing efficiency, and data privacy preservation.

The survey concludes by outlining future research directions, including designing specific LLMs for database systems, employing multi-agent architectures, and combining explainable artificial intelligence methods. This research gives a systematic platform and a roadmap for future exploration at the intersection of multi-model data management and language-based artificial intelligence.

ACM Computing Classification System (CCS)
General and reference → Document types → Surveys and overviews

Avainsanat — Nyckelord — Keywords

Large Language Models; Multi-Model Databases; Data Integration; Reinforcement Learning; Data Management; Human-AI Interaction in Databases

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Software study track

Contents

1 Introduction

Modern data are highly varied in form and construction and originate from sources as diverse as transactional systems, social media, sensors, and documents (Lu et al., 2018). Relational databases find it challenging to cope with this variability since they tend to be limited by a single data model (J. Lu & Holubová, 2020). During the last two decades, the challenges of single-model systems and increased big data requirements catalyzed the development of multi-model databases (MMDBs), which accommodate multiple models for a single unified backend system (Lu et al., 2015). The central theme behind MMDBs is to offer a single platform supporting pluralistic data representations without a loss of consistency or a huge overhead of complexity. By facilitating "polyglot" data persisted in a single system, MMDBs pledge to make development and integration simpler compared with employing specialized databases per model and integration overhead (Košmerl et al., 2020). Big players and open-source initiatives adopted this approach. ArangoDB, for instance, started as a native multi-model database and has supported document, graph, and key-value data under a single query interface from the beginning. Other databases such as MongoDB and Cosmos DB, beginning as document stores, proliferated over time and added graph and relational-like functionality and features, demonstrating a wider trend of NoSQL databases generalizing toward multi-model functionality (Holubová et al., 2019). MMDBs thus embody a contemporary method of dealing with data variety and enabling applications to access the optimum storage or query model for each subset of data without sacrificing a single-system stance.

Parallel to these advancements in data management are the rise of large language models (LLMs), which emerged as highly advanced AI models that could comprehend context and produce human-like text. Those LLMs based on the Transformer architecture have shown impressive qualities in natural language understanding and generation (Fernandez et al., 2023; Li et al., 2024). Trained from large corpora, models like GPT-4 and their contemporaries can glean the intent of the user from plain text and create sensible responses and even perform reasoning within limits. They perform well at question answering, summarization, and dialogue by relying on their learned worldview and contextual sense. Recently, LLMs also started supporting multi-modal inputs and processing structured data on limited scales better than ever before and thus enlarged their usability even further. These features make it possible for LLMs to be used as natural language interfaces and intelligent assistants for complex systems (Zhou, Sun, et al., 2024). Specifically, the proposition of interacting with databases using LLMs has gained popularity recently: an LLM might translate a query from a user's question, deduce the meaning of data, or even create analytical reports in the form of a story. The prospect seems inviting as LLMs can fill the gap between heterogeneous data and a human user so that a non-technical user might query and analyze data simply by asking a question in natural language and receiving useful outcomes for the same (Zykin & Zykin, 2023). This may democratize access to data so a non-technical person might be able to query and analyze data without having to learn formal query languages in the bargain.

The integration of LLMs and MMDBs has the compelling prospect of harmonizing the capabilities of both systems. Such an LLM-aided MMDB system would find and maintain multi-model data through conversational queries and management, while beneath the surface the MMDB guarantees stable and secure storage and retrieval of multiple data types (Bai et al., 2023; J. Lu et al., 2024). Such synergy would make the data extremely usable since elaborate queries cutting across documents, graphs, and tables could be expressed in plain language and be automatically converted to the right operations. It would also facilitate profound insights: the LLM's comprehension of context would identify relationships between heterogeneous modalities of data, e.g., between text reports and graph relationships or between images in the database, which might be unnoticeable to a user. Further, LLMs could be used for tasks like automating the integration of data by recommending schema mergers or data transformations or even optimizing queries or performing mundane database administrative operations like explaining query responses or pinpointing performance bottlenecks (Y.T. Lu et al., 2024; Zhou et al., 2023).

However, this integration also comes with significant challenges and research questions. LLMs were not originally designed to be query engines or database managers, and naive use of LLM outputs could lead to incorrect or nonsensical database operations, such as hallucinated results that do not exist in the underlying data (Z. Sun et al., 2024). Ensuring the accuracy, efficiency, and security of an LLM-MMDB combined system is non-trivial. Issues of how to keep the LLM’s knowledge in sync with the database’s contents, how to handle the computational cost of large model inference within database workloads (Wang et al., 2024; Zhou, Zhao, et al., 2024), and how to prevent sensitive data leakage through the language interface must all be addressed (Akioyamen et al., 2024).

Given these opportunities and challenges, this thesis is guided by four key research questions:

1. How can Large Language Models enhance query optimization and processing within multi-model databases?

2. To what extent can LLM-driven techniques automate schema generation and management across various data models effectively?

3. How can LLMs facilitate privacy-aware query processing and effective data anonymization strategies in MMDBs?

4. What critical performance and privacy trade-offs arise when integrating LLMs into MMDB data management?

By addressing these questions, this thesis explores both the theoretical potential and the practical implications of integrating LLMs with MMDBs. The goal is to contribute meaningfully to both academic knowledge and practical advancements in intelligent data management systems.

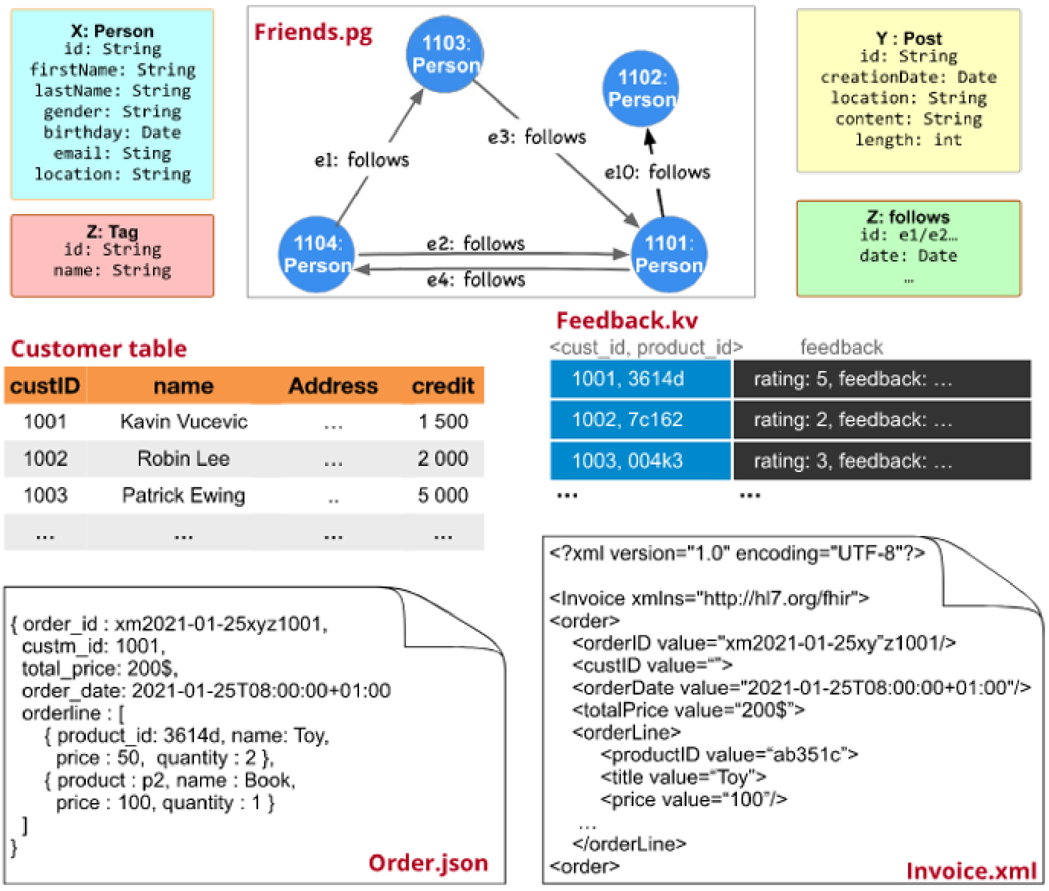
This thesis proceeds as follows: Chapter 2 reviews relevant literature on multi-model databases and large language models, providing the theoretical foundation for this study. Chapter 3 proposes a methodology for integrating LLMs with MMDBs, including system architecture and reinforcement learning techniques. Chapter 4 evaluates the approach, discussing its benefits, current limitations, and open challenges. Chapter

5 concludes the thesis with a summary of findings and directions for future work.

2 Background and Theoretical Foundations

2.1 Multi-Model Databases: Concepts and Architectures

The concept of multi-model databases (MMDBs) has emerged in response to the increasing variety of data types in the era of big data. Initially, attempts to address this issue focused on extending traditional single-model databases. For example, in the 1990s, object-relational databases were introduced to bridge the gap between relational and object-oriented paradigms. Later in the 2000s, the emergence of specialized NoSQL systems—e.g., document stores, graph databases, and key–value stores—facilitated better non-relational data processing. Nevertheless, installing and sustaining multiple heterogeneous systems usually brought high integration and operational complexity. Against this backdrop, researchers started investigating the concept of supporting multiple data models in a single system. As encapsulated by Lu and Holubová (2017, 2020), an MMDB describes a single database platform capable of hosting and processing data in multiple models naturally. Such integration is particularly useful in contemporary applications, as it saves data transformation overhead, makes development processes simpler, and allows for fuller query ability of multiple data types (Košmerl et al., 2020).



Customer table

custID	name	Address	credit
1001	Kavin Vucevic	…	1 500
1002	Robin Lee	…	2 000
1003	Patrick Ewing	..	5 000
…	…	…	…

Feedback.kv

<cust_id, product_id>	feedback
1001, 3614d	rating: 5, feedback: …
1002, 7c162	rating: 2, feedback: …
1003, 004k3	rating: 3, feedback: …
…	…

```
{ order_id : xm2021-01-25xyz1001,
  custm_id: 1001,
  total_price: 200$,
  order_date: 2021-01-25T08:00:00+01:00
  orderline : [
     { product_id: 3614d, name: Toy,
       price : 50,  quantity : 2 },
     { product : p2, name : Book,
       price : 100, quantity : 1 }
  ]
}
```

Order.json

```
<?xml version="1.0" encoding="UTF-8"?>

<Invoice xmlns="http://hl7.org/fhir">
<order>
   <orderID value="xm2021-01-25xy"z1001/>
   <custID value="">
   <orderDate value="2021-01-25T08:00:00+01:00"/>
   <totalPrice value="200$">
   <orderLine>
      <productID value="ab351c">
      <title value="Toy">
      <price value="100"/>
   …
   </orderLine>
<order>
```

Invoice.xml

Figure1. An example multi-model dataset generated with UniBench

An example of this integration can be found in Figure 1, which shows a synthetic multi-model dataset created with UniBench (Guo et al., 2024). It mimics a commercial social network application and illustrates how varied data models share a common space. It contains in particular:

(1) A relational table for organizing customer data like names, addresses, and credit levels.

(2) A property graph showing the users' social network and follow relationships.

(3) Document-type data in JSON and XML forms for maintaining order information and invoices, respectively.

(4) Key-value pairs for keeping track of product feedback.

Each model serves a distinct purpose document models handle semi-structured and hierarchical data, graph models are well-suited for capturing user interactions and relationships, while relational tables offer schema integrity and support complex joins (Uotila & Lu, 2021). As the figure illustrates, these diverse data types are interlinked through shared identifiers, which makes cross-model queries not only possible but also meaningful in context. For example, one could imagine querying which customers (relational) have followed certain users (graph), purchased specific products (JSON), and left feedback (key–value). This example highlights the potential of MMDBs to support rich analytical workloads over heterogeneous data while keeping all data models within a single system. Nonetheless, as Guo et al. (2024) point out, different MMDB implementations may vary significantly in terms of performance, consistency guarantees, and the level of support they offer for each model type.

2.1.1 Architecture Approaches

MMDB implementations generally follow one of a few architectural patterns.A unified engine built from the ground up to support multiple models under one query language and storage engine. In a native MMDB, data of different types is stored in an integrated manner and queries can seamlessly access any model. Example: ArangoDB is often cited as a native MMDB. It was designed to treat documents, graphs, and key–value pairs as first-class citizens within one system. Its query language AQL can perform joins across collections and graph traversals in one query, reflecting tight integration. The benefit of a native approach is consistency and efficiency (Koupil et al., 2024). There is a single query processor and optimizer that knows about all data, potentially enabling global optimizations. However, building a native engine is complex and might not be as feature-rich for each model as dedicated systems (Lu et al., 2016; Zhang & Lu, 2021).

2.1.2 Layered MMDB

An architecture where the system is composed of multiple engines or layers, each handling a specific model, on top of a common lower-level storage or API. For example, a database might use an underlying key–value store for raw storage and build a document store and a graph query layer on top of it. This is sometimes called a federated or multi-store approach. Example: OrientDB began as a graph database but also supports documents and key–value; internally it has different components for these models but presents a unified interface. Layered designs can leverage existing technologies by plugging in well-developed engines for each model. The trade-off is often performance overhead and operational complexity. The layers must coordinate consistency, and queries spanning models might incur communication between engines. Some academic prototypes have explored using a common storage engine with multiple "personality" layers for each model. The benefit is flexibility and possibly easier extension, at the cost of less tight integration (Holubová et al., 2019; Yuan et al., 2021).

This is a looser approach where an application uses multiple different databases together, and integration is done at the application level or via an integration layer. Strictly speaking, this is not a single MMDB system but rather a pattern of using specialized databases in concert. While this can yield optimal performance per model, it puts the burden on developers to handle data consistency and query across systems. The MMDB paradigm tries to avoid this by providing one system, so polyglot persistence is considered an alternative rather than an instance of MMDB (Košmerl et al., 2020; Sachdeva & Vasava, 2024).

Each architecture carries certain trade-offs in query capability and performance. Native MMDBs offer cohesive querying across models but may struggle to be cutting-edge in every model. Layered systems aim to reuse components but may suffer overhead. In practice, the choice can depend on use case requirements. Understanding these trade-offs is crucial when considering integrating an LLM, because the architecture will affect how easily an LLM can interface with the data. For example, a native MMDB with one language might allow an LLM to generate a single query that touches everything, whereas a federated system might require the LLM to formulate multiple sub-queries for different backends(Sun et al., 2025; Zhou et al., 2024).

2.1.3 Advantages of MMDBs

The rise of MMDBs has been driven by tangible benefits. First, they can significantly simplify development in scenarios with diverse data. Instead of maintaining multiple query languages and moving data between stores, developers work with one platform. Second, multi-model queries can reveal insights that would be hard to get otherwise. For instance, in a customer analytics scenario that Lu and Holubová (2017, 2020) describe, one might join relational customer records with social network graphs and JSON product catalogs to get a comprehensive view. Doing this in separate systems would be cumbersome, but an MMDB makes it straightforward. Third, there are potential performance benefits by reducing extract-transform-load (ETL) between systems. Data doesn't need to be copied and transformed as often, since it resides together. Lastly, administration is unified: backup, security, and scaling can be managed for one system rather than many (Holubová et al., 2019; Košmerl et al., 2020).

Despite these advantages, challenges remain in MMDB technology itself. Ensuring query optimization across models is difficult. The system must decide how to plan a query that involves both graph traversal and relational joins. Benchmarking has shown that certain MMDBs excel in one area but lag in another (Zhang & Lu, 2021). The research community continues to work on performance tuning, benchmark development, and techniques for schema design in multi-model contexts (Guo et al., 2024). Interestingly, some work has applied machine learning to MMDB problems. As I will discuss later, reinforcement learning has been used to automatically find efficient relational schema for multi-model data stored in a relational backend. These efforts indicate that the MMDB field is still evolving. Nonetheless, the concept of a multi-model database has taken hold in industry, establishing a foundation that I now propose to augment with LLM capabilities.

2.2 Large Language Models: Capabilities and Limitations

LLMs refer to deep learning models with hundreds of millions to billions of parameters, trained on vast amounts of text data to predict and generate language. The advent of the Transformer architecture was a key breakthrough enabling today's LLMs (Vaswani et al., 2017, Fernandez et al., 2023). Transformers use a self-attention mechanism that allows modeling long-range dependencies in text more effectively than previous recurrent neural networks. Essentially, attention allows the model to weigh the relevance of different words in a sequence with respect to each other, enabling understanding of context and nuanced relationships in language. Scaled-up Transformers, when trained on massive corpora, give rise to models like GPT-4, Gemini 2.5, Claude 3.7 and others which have demonstrated advanced linguistic competence (Fernandez et al., 2023; Zhou, Sun, & Li, 2024).

Internally, LLMs operate by first tokenizing text, breaking input text into tokens, which are often subword units. Each token is then converted into a vector embedding that captures semantic meaning. The Transformer layers apply self-attention to these embeddings, meaning the model updates its representation of each token by looking at every other token and computing weighted averages that reflect how strongly tokens relate. By numerous layers of this process, the model develops a richly contextualized understanding of the sequence. Lastly, a decoder or output layer produces tokens sequentially for generating language. As a result of this training, LLMs can learn about syntax, facts and even some patterns of reasoning implicitly from data (Chen et al., 2023; Li et al., 2024).

LLMs possess dual strength both in natural language understanding (NLU) and natural language generation (NLG). On the understanding side, they can analyze the intent and connotation of a user's question or statement. An LLM can identify that "Show me the sales trend of Europe last quarter" requests a sales trend on a region and even deduce what data would be required. On the generation side, LLMs are able to generate answers, descriptions, or narratives that are well-formed and typically contextually accurate. They can summarize texts, translate texts from and to languages, and even generate code or formulae from descriptions. These qualities apply very well to database interaction: analyzing a user's question in natural language is effectively an NLU task, and generating a helpful answer is an NLG task (Zhou, Zhao, & Li, 2024; Fernandez et al., 2023).

Traditionally, LLMs learned from unstructured text, but researchers have started integrating structured knowledge. Some LLMs are augmented with retrieval mechanisms to fetch precise facts. In the context of databases, a line of research is teaching LLMs to comprehend and write SQL or other data queries from natural language. Models like ChatGPT have shown some proficiency in generating SQL queries if given schema information (Sun et al., 2025; Zheng et al., 2024). However, this is not trivial—an LLM needs to correctly understand the database schema and the user's request to form a valid query. It may also need to reason about constraints or perform calculations. There are benchmarks that test these abilities, and LLMs fine-tuned on such tasks or guided by prompts have made progress (Li et al., 2024; Zhou, Zhao, & Li, 2024). Still, purely as stand-alone systems, LLMs might misinterpret or hallucinate structured queries that are syntactically or semantically wrong, which is problematic for execution on a database (Sun, Li, Srinidhi, & Hai, 2025).

LLMs bring certain strengths that could be harnessed in data management contexts. First, generalizability: unlike traditional rule-based NLP, an LLM can handle a wide range of inputs, adapting to different phrasing and even languages (Zhou et al., 2024). Second, contextual reasoning: LLMs can perform a form of reasoning by leveraging patterns learned from training (Fernandez et al., 2023). Third, knowledge: LLMs contain a vast amount of world knowledge from their training data, which may aid in query interpretation (Li, Zhou, & Zhao, 2024). Fourth, natural output: LLMs can generate explanations or summaries of data findings in human-friendly language. This means that beyond retrieving raw data, an LLM could write a short analysis which adds value by interpreting the data (Zhou, Sun, & Li, 2024).

For LLM's known limitations. Although LLMs exhibit impressive capabilities, there are a number of well-known drawbacks to their integration into database systems that need to be carefully taken into account for both research and real-world applications.

1. Hallucinations: LLMs frequently fabricate information that appears true but is not. This might entail reporting a data value or trend that isn't present in the database in a database scenario. It is crucial to prevent hallucinations (Sun, Li, Srinidhi, & Hai, 2025; Zheng et al., 2024).
2. Absence of true understanding: Rather than using actual logic or comprehension, LLMs rely on pattern recognition. Complex logic or query requirements may be misinterpreted by them (Zhou et al., 2024).
3. Static knowledge: During training, LLMs' knowledge was fixed. Unless specifically supplied, they do not inherently incorporate dynamic or proprietary data. Mechanisms like retrieval-augmented generation (RAG) are used to mitigate this (Su et al., 2024).
4. High computational cost: LLMs demand a huge amount of computational resources. It constrains scalability for real-time or high-frequency queries (Chen et al., 2023; Wang et al., 2024).
5. Latency: A single inference may run for seconds, as compared to millisecond response times common in databases (Li, Zhou, & Zhao, 2024).
6. Schema and domain knowledge requirement: An LLM needs schema and domain context so as not to make query generation mistakes (Sun et al., 2025).
7. No long-term memory: LLMs lack context memory between sessions unless they are specifically set out to do so (Zhou et al., 2023).
8. Bias and inconsistency: LLM responses can mirror training data biases and change between the same queries, which undermines determinism and fairness in database responses (Wang et al., 2023;

Fernandez et al., 2023).

LLMs are highly capable but not perfect tools. They offer a dynamic and smart interface for accessing databases but must be supplemented by data access modes and safety checks for dependability. The existing literature proposes several mitigations, i.e., retrieval-based approaches, constrained output forms, and tool use integration (Chen et al., 2023; Zhou, Zhao, & Li, 2024; Qin et al., 2023). These will guide the suggested methodology for integrating LLMs and MMDBs.

3 LLMs for MMDBs

3.1 Data modeling with LLM and RL techniques

In this chapter, I present the method for the integration of a large language model in a multi-model database system. The chapter gives a design outline explaining how integration of this type can be established through system architecture and auxiliary processes. I also explain the extent to which reinforcement learning (RL) methods can be used to assist LLM in optimizing data management operations (Zhou, Zhao, & Li, 2024; Sun et al., 2025).

3.1.1 System Architecture for LLM-Enhanced MMDB

The proposed system architecture links a LLM and elements of a MMDB to facilitate intelligent automation and natural interaction through language. The system has a three-pronged design: the Interface/Agent Layer utilizing the LLM to interpret user input and plan actions; the Core Database Layer as the MMDB engine tasked with data storage and query operations; and the Auxiliary Services for facilitating integration between the LLM and the database through supporting tools. The layered approach borrows from agent-based frameworks of recent designs relying on LLMs for task coordination between system components (Zhou et al., 2023; Chen et al., 2023).

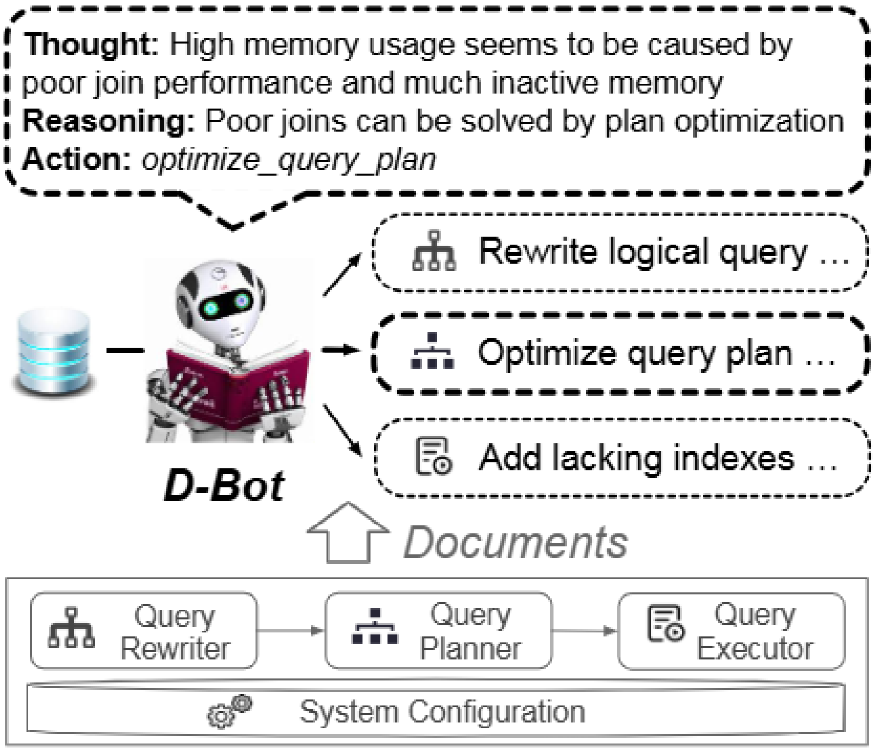

Thought: High memory usage seems to be caused by poor join performance and much inactive memory
Reasoning: Poor joins can be solved by plan optimization
Action: *optimize_query_plan*
Rewrite logical query ...
Optimize query plan ...
Add lacking indexes ...
D-Bot
Documents
Query Rewriter
Query Planner
Query Executor
System Configuration

Figure 2: LLM As DBA

One representative architectural figure of this concept can be seen in the proposed design by Zhou, Li, and Liu (2023), see Figure 2. In this example, the LLM-based agent, referred to as “D-Bot,” interacts with both documents and the database system to diagnose performance issues, reasoning about their root causes, and proposing actions like query optimization or adding an index. The agent has communication with internal database modules like the query planner and executor on top of a layered design similar to what has been proposed in this chapter. This is an intuitive starting-point reference for utilizing LLMs in MMDB systems for tool-driven intelligent automation(Guo et al., 2020).

1. Interface:

This top layer consists of an LLM-based conversational agent that receives natural language input and produces output responses. The LLM interprets user intent and formulates plans to fulfill requests. It interacts with tools and data access components. This layer is implemented by prompt engineering the LLM with a system prompt defining its behavior and capabilities, using tool-use techniques where the LLM calls predefined commands (Qin et al., 2023; Fernandez et al., 2023). The LLM may output a plan or pseudo-code executed by other layers. This structured approach helps prevent hallucination and grounds outputs in database operations (Zhou et al., 2024).

2. Core Database Layer (MMDB Engine):

This layer is the MMDB itself, handling storage, indexing, and query execution across multiple models. An API exposes the MMDB's capabilities to the LLM agent. The MMDB executes commands such as RUN_SQL, RUN_GRAPH_QUERY, or RUN_FTS and returns results. It remains the authoritative data source. The system supports both single-model and cross-model queries, depending on the MMDB's capabilities and the LLM's awareness of supported syntax (Lu et al., 2016; Guo et al., 2024). This design provides flexibility: the LLM may generate one composite query or issue multiple model-specific queries.

3. Auxiliary Services (Bridging Components):

Several services support the LLM and MMDB layers:

1. Vector Database and Embeddings Index: Stores vector embeddings of data to enable semantic search. Queries are embedded and matched against stored vectors to provide contextual snippets, which are then included in LLM prompts to improve relevance (Luo et al., 2024; Lyu et al., 2023).
2. Schema and Metadata Knowledge Base: Provides a structured description of the schema, data types, and relationships. The LLM queries this to avoid schema-related hallucinations (Zhou et al., 2023; DB-GPT – Zhou, Sun, & Li, 2024).
3. Execution Checker and Result Verifier: Evaluates execution results and prompts the LLM for refinement if necessary. This feedback loop allows for iterative improvement (Zheng et al., 2024; Li, Zhou, & Zhao, 2024).

Prompt engineering and fine-tuning tie the architecture together. System prompts instruct the LLM to use tools responsibly and include examples. Fine-tuning may be done using simulated dialogues or real query logs to improve accuracy and tool usage. The approach leverages frameworks such as LangChain or LlamaIndex to orchestrate the system (Chen et al., 2023; Xie et al., 2023).

The LLM first queries the MMDB to identify top products, then retrieves relevant posts from a document store or graph structure, and finally summarizes key themes using its generative capability. Each step is grounded in real data, and the system performs multi-hop reasoning across data models (Zhou et al., 2024; Lu et al., 2024).

3.1.2 Reinforcement Learning (RL) for Optimization.

RL techniques complement LLMs by optimizing internal system behavior:

1. Schema Design: RL agents learn optimal schema transformations to improve performance. Actions such as splitting attributes or creating join tables are rewarded based on performance gains. This helps adapt multi-model storage strategies over time (Yuan et al., 2022a; Yuan & Lu, 2021).
2. Query Optimization: An RL optimizer would be able to learn from feedback at execution time and optimize multi-model query plans. The LLM would initially produce plans, and the RL agent would refine those (Yan et al., 2023).
3. Dialogue Management: RL can decide when to ask clarifying questions rather than guessing. User satisfaction or the success of a task will be the reward signal (Shi et al., 2024; Sun et al., 2025).

The method proposed develops a system based on feedback wherein the LLM directly communicates with the MMDB, and RL agents operate in the background to enhance performance and facilitate schema decisions. This integration would develop a system suited for reliability, efficiency, and adaptive responses in line with the requirements of the user (Zhou et al., 2024).

This aligns with existing work in the area, which emphasizes combining LLMs with structured tools and grounded sources of data. It lends itself ideally to processing multiple data types through a hybrid architecture supporting semantic, relational, and graph-based processing (Zhou, Sun, & Li, 2024; Wang, Ma, & Wang, 2024).

3.2 Case studies

To demonstrate the features of an MMDB, I present a few representative application scenarios. These case studies highlight the integration in action from query interfaces at the application level all the way through data management functions at the backend level. The differing advantages of the approach will be reflected in each scenario and supported by any available real-life or research examples.

3.2.1 Natural Language Querying and Data Exploration

One of the most compelling use cases for LLM-enhanced MMDBs is allowing natural language queries over complex multi-model datasets. Instead of writing SQL, graph queries such as Cypher, or search syntax, a user can ask questions in plain English or any natural language and get results (G. Yuan et al., 2021). The LLM acts as a translator between the user and the database, enabling conversational exploration of data.

Natural Language Interfaces on LLM-MMDB systems make it possible for humans to conduct advanced data analysis on complex queries without any SQL or data schema knowledge. These systems can interpret high-level analytical queries in natural language and break them down as structured operations over heterogeneous data models like relational tables, document collections, and knowledge graphs.

These systems can also automatically create sub-queries across varied modalities. The results from each of these sources then become synthesized into readable summaries. If provided with a multimodal interface, the system might even offer visual elements like tables or charts in order to enhance result interpretability. This method reduces the necessity for individuals to comprehend or navigate manually multi-model data landscapes and thus lowers the hurdle of enterprise-scale data analysis and decision-making.

A concrete research prototype in this vein is XMODE (Nooralahzadeh et al., 2024), which stands for explainable multi-modal data exploration in natural language. XMODE is designed for exactly this purpose: it allows users to query a combination of relational and unstructured data such as images using natural language, and uses an LLM-based agent to break the query into sub-tasks such as text-to-SQL and image analysis (Nooralahzadeh et al., 2024). The system was shown to outperform earlier multi-modal query systems in both accuracy and performance. It demonstrates that even complex queries spanning different modalities can be handled by an LLM orchestrating various tools. In the context, where modalities might include text documents and graph data in addition to relational, a similar agent can break down a user's request to cover each part.

Natural language data query interfaces greatly reduce the gatekeeper function of technical experts for data analysis, especially for non-technical users in business areas like marketing, finance, and healthcare. By supporting the entry of analytical intent through natural language, the systems eliminate reliance upon database experts and encourage more independent decision-making. The conversational interface also increases usability by enabling follow-up queries that refine or expand upon the initial question.

Responding to queries of this sort, a language model has the capacity to combine structured information and unstructured sources and produce plausible explanatory answers. Such multimodal thinking across both structured and unstructured information shows how LLMs can perform complex analytical tasks that would otherwise need a great deal of manual labor and technical know-how.

Prototypes of natural language to SQL systems exist in products like Microsoft Power BI with the Q and A feature, or Google BigQuery with a natural language interface. Those are limited to structured data and often to predefined questions. An LLM can be far more flexible, handling arbitrary phrasing and even multi-turn dialogue. The explainability is also enhanced: since the LLM can generate human language, it can explain how it got the answer to some extent. This builds trust because the user gets a peek into the logic, which is often a black box in business intelligence tools (Fernandez et al., 2023; Zhou, Zhao, & Li, 2024).

A potential challenge in this scenario is ensuring that the LLM's translation to queries is correct. As discussed, strategies like including schema in the prompt or using few-shot examples such as "When asked about top products, here is how to form the SQL" can help. Also, the system can validate the query by running it and seeing if results make sense. If a query returns zero when the user clearly expects data, the LLM should reconsider. These measures, combined with user confirmations, keep the natural language interface accurate (Zhou, Sun, & Li, 2024; Zheng et al., 2024).

Natural language querying and exploration via LLM allows interactive, intuitive analysis of multi-model data. It brings together disparate data sources in a single answer and lets users ask follow-ups as they would to a human analyst. This scenario fundamentally improves data accessibility and has been demonstrated in early forms by systems such as XMODE. As LLMs improve, such interfaces are expected to become commonplace.

3.2.2 Data Integration and Transformation

Data integration, the process of merging data from different sources or models, is traditionally a labor-intensive task requiring an understanding of schemas and writing transformation code. LLMs can assist by interpreting schemas and generating transformation scripts or mappings in context. In a MMDB setting, where the database itself holds multiple models, integration often occurs within the database, such as linking a document’s field to a relational table’s ID. LLMs can make this process easier and more automated (Chen et al., 2023; Zhou, Zhao, & Li, 2024).

Consider a company that acquires another company and now has two different databases to merge into a multi-model store. One database has a relational schema for customers and orders, while the other is a document store with customer profiles and order logs. An engineer or data steward can ask the LLM: “Align the customer data between our SQL database and the new JSON profiles. The SQL uses customer_id, and the JSON uses custID; also extract the important fields from JSON such as preferences to integrate with the relational model.” The LLM, having been provided with the schema of both systems, can infer that customer_id in SQL corresponds to custID in JSON, possibly by name similarity or metadata description. It might generate an ETL plan or actual code. For example, "For each JSON profile in the Profiles collection, find the matching customer_id in the Customers table. If found, insert a new record in a unified Customer table combining fields such as name and address from SQL and preferences from JSON." The system could output this plan or directly execute a series of operations on the MMDB, which can store both relational and JSON data, to create a unified structure and migrate data (Zhou, Sun, & Li, 2024).

While fully autonomous integration remains challenging, the LLM can significantly reduce the effort by generating initial attribute mappings (Wang et al., 2023). In academic work, LLMs have been used for schema matching with promising results. The model suggests likely attribute correspondences, which humans then validate. The LLM can also identify data type conflicts or propose standardization. The LLM, trained on typical address patterns, could output those components or a regex or script to perform the parsing (Qi & Wang, 2024).

In the context of a multi-model database, integration also involves linking data across models. Suppose an MMDB stores product data in a graph structure and reviews in a text collection. Integration might mean linking each review to the product it discusses. An LLM could perform named entity recognition on review text to find product names, using a product list from the graph, and then create edges in the graph connecting each review to the appropriate product node. This type of integration connects unstructured and structured data. Traditionally, this would require a custom NLP pipeline. With an LLM, steps such as identifying the product, finding its ID, and creating the link can be executed through prompt-driven workflows. LLMs are especially suited to such tasks due to their training on diverse text corpora (Zhou, Zhao, & Li, 2024; Fernandez et al., 2023).

Researchers have used GPT-style models to generate SQL or data pipelines from natural language descriptions. Extending this to multi-model scenarios, an LLM could create a multi-step pipeline: querying an external API, joining the data with internal datasets, and preparing a unified view (Sun et al., 2025). Although full automation requires caution, these AI-generated pipelines provide a draft for developers to refine, accelerating the process. This reverses the traditional process: instead of writing integration code from scratch, developers start from AI-suggested templates and iterate (Uotila et al., n.d.).

LLM-based integration can be viewed as a component of the extract-transform-load or extract-load-transform processes in data warehousing. Tools may allow users to describe how they want data unified, and the LLM generates the transformation logic. Given their ability to produce working code, this is increasingly feasible (Qin et al., 2023; Zhou et al., 2023).

LLMs enable data integration through their intelligent mapping and script generation abilities, utilizing their semantic awareness. In a multi-model database setup, this function provides efficient connecting and harmonizing of data from models with minimal manual intervention involved. This leads to a smoother integration process, which is essential when combining disparate data in a multi-model system.

3.2.3 Database Management and Optimization Assistance

Outside of asking queries against the data itself, LLMs can assist in database system management and tuning as well. Database adminswill carry out operations like diagnosing sluggish queries, tweaking indexes or creating stored procedures. LLMs can act as intelligent assistants on those operations, examining system metadata and logs and making recommendations or even taking action automatically (Zhou, Zhao, & Li, 2024; Fernandez et al., 2023).

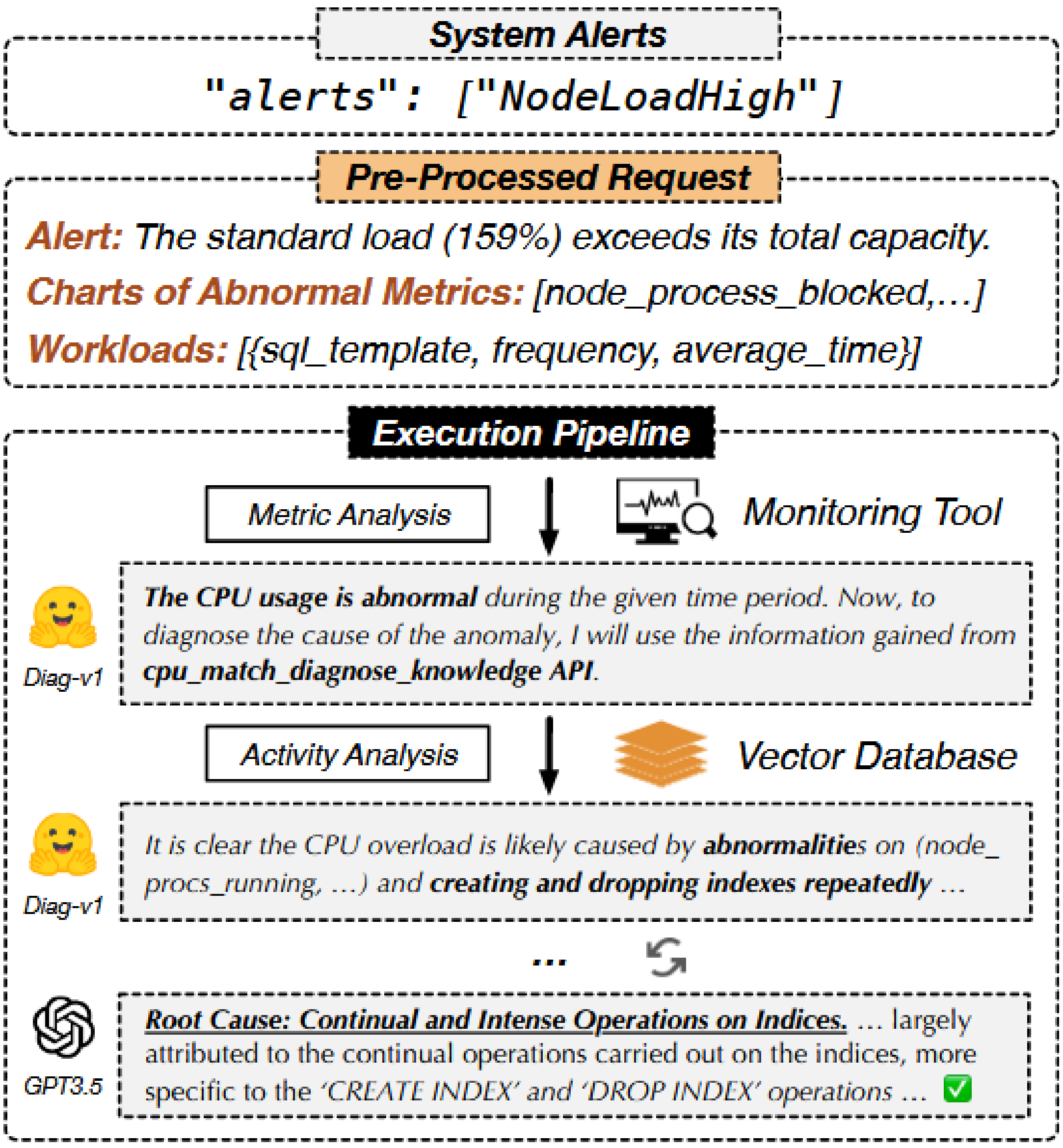

System Alerts
"alerts": ["NodeLoadHigh"]
Pre-Processed Request
Alert: The standard load (159%) exceeds its total capacity.
Charts of Abnormal Metrics: [node_process_blocked,...]
Workloads: [{sql_template, frequency, average_time}]
Execution Pipeline
Metric Analysis
Monitoring Tool
Diag-v1
The CPU usage is abnormal during the given time period. Now, to diagnose the cause of the anomaly, I will use the information gained from cpu_match_diagnose_knowledge API.
Activity Analysis
Vector Database
Diag-v1
It is clear the CPU overload is likely caused by abnormalities on (node_procs_running, ...) and creating and dropping indexes repeatedly ...
...
GPT3.5
Root Cause: Continual and Intense Operations on Indices. ... largely attributed to the continual operations carried out on the indices, more specific to the 'CREATE INDEX' and 'DROP INDEX' operations ... ✅

Figure 3. LLMDB for Database System Diagnosis

A visual representation of such an LLM-assisted optimization workflow is shown in Figure 3 (Zhou et al., 2024). The figure illustrates how an LLM-based agent receives system alerts (e.g., node overload), performs metric and activity analysis by interacting with external monitoring tools and a vector database, and finally reasons about the root cause using structured prompts and domain knowledge (Zhang et al., 2024). This process is highly aligned with the virtual DBA use case discussed above—where the LLM analyzes execution plans, identifies issues like missing indexes, and proposes or executes corrective actions. Such architecture exemplifies how LLMs can go beyond passive language processing and serve as active agents in database tuning and management(Herodotou et al., 2021).

With increasing data volume, performance issues in specific queries become a frequent concern for DBAs. In such a situation, a DBA can use the LLM as a tool to identify the cause and suggest performance optimization. If a customer search query, for example, is being analyzed, the LLM would engage the multi-model database through commands like EXPLAIN_QUERY followed by a SELECT statement to fetch the query plan and execution statistics.

From the query plan analysis, the LLM can determine whether a full collection scan on a large set is being conducted because a suitable index doesn't exist. It may then produce a response like: "The query on the collection Customers by the email field causes a full collection scan since there is no index on that column. Adding an index on Customers.email will most probably enhance performance." The system can also make usability better by presenting the actual database command required in order to create the recommended index and, when permissions allow, make the change or offer it for user approval.

Effectively, the LLM serves as a virtual database aide, integrating its best-practice understanding—derived from technical references and subject matter expertise—against the actual state of the underlying database system (Zhou et al., 2023).

In literature, a commonly cited vision is that of an auto-tuning database that learns and adapts. LLM integration could accelerate this by making tuning more conversational. Even less experienced developers can ask how to improve query performance and receive context-aware advice. LLMs can also help interpret error messages or log entries. For instance, if a backup fails and logs an error code, the DBA can input the code into the LLM and ask for an explanation and resolution. Since many error messages and solutions are documented online, the LLM can often identify the cause and suggest a fix. For example, "This error indicates the transaction log is full. You should increase its size or perform a log backup to clear it" (Li, Zhou, & Zhao, 2024; Fernandez et al., 2023).

Another management task is query rewriting for optimization. An LLM can take a poorly written query and suggest a better formulation. Zhou et al. in 2024 demonstrated query rewriting as one of the LLMDB scenarios (Zhou, Sun, & Li, 2024). For instance, a user query might use a subquery inefficiently, and the LLM may suggest replacing it with a join (Živanović, 2023). It might also identify unnecessary complexity, such as selecting distinct on already unique fields, and simplify the query. This function is similar to that of some SQL advisors, but an LLM can also understand the query's intent and ensure that the optimized version is logically equivalent (Sun et al., 2025).

Consider also routine tasks such as documentation and query explanations. An LLM can generate human-readable explanations of complex queries or stored procedures, which is helpful for onboarding new team members. Tools aim to explain SQL in natural language; LLMs are well-suited for this purpose (Zhou et al., 2024).

A relevant case is Microsoft’s Azure OpenAI service, which has been exploring a "Copilot for Databases" concept that enables users to chat with an agent about their Azure SQL database. Early demonstrations show the agent recommending indexes and writing T-SQL code. Another example is Oracle’s auto-tuning features, which, although not currently known to use LLMs, could benefit from LLM integration to explain recommendations in plain language and build user trust (Fernandez et al., 2023).

A potential pitfall is that an LLM might suggest incorrect actions, such as a non-existent tuning parameter or a misinterpreted scenario. However, because it can query the database environment to verify current configurations, it can be grounded in the actual system state. Furthermore, the human DBA remains in the loop and can validate critical decisions (Zhou et al., 2024).

In essence, using LLMs for database management transforms the system into a more autonomous, self-healing platform with a natural interface for guidance. It functions like a knowledgeable co-pilot: the LLM has access to a vast corpus of manuals and forum discussions and applies this knowledge to assist in maintaining database performance and integrity.

3.2.4 Content Generation and Summarization

In some applications, the database is not just for storing data but also for generating content or reports from that data. LLMs can be leveraged to produce natural language narratives or summaries based on data in the MMDB, effectively turning raw data into understandable knowledge. This is particularly useful for reporting, documentation, or data services where a textual or multimedia output is desired (Fernandez et al., 2023; Zhou, Zhao, & Li, 2024).

A regional manager wants a summary of the latest sales and inventory situation at the end of the week. Instead of manually looking at dashboards, they trigger the system to generate the weekly report for region North-America. The LLM then queries the MMDB for sales figures, key performance indicators, notable events such as stockouts or new product launches, and perhaps even pulls a couple of representative customer reviews or support tickets for qualitative insight. It then composes a cohesive report in paragraphs: "This week, North America region achieved $5.2 million in sales, an 8 percent increase from last week, primarily driven by the electronics category" This kind of narrative provides a quick overview and blends quantitative data with qualitative context. The LLM essentially serves as a data-to-text generator, a role that has been studied in natural language generation for years, such as automatic weather reports from meteorological data (Chen et al., 2023).

LLMs are particularly effective at this task because they have seen many textual reports and know how to phrase content smoothly. For multi-model data, content generation can also involve images or other modalities. An advanced system might even generate a simple chart or select an image if relevant, though purely visual generation enters the territory of multi-modal LLMs, which some models support (Hu et al., 2023).

If the MMDB stores images such as product photos and the prompt is to create a marketing blurb for the top product, the LLM could describe the image with the help of a vision model as part of the generated content. Another use of generation is creating descriptions or metadata for data entries. Suppose the MMDB contains a collection of research articles and an LLM is tasked to generate summaries for each and store them back in the database. This use of the LLM effectively enriches the data (Qin et al., 2023; Zhou, Sun, & Li, 2024).

Many organizations have such needs, such as summarizing long reports into key points. LLMs can perform abstractive summarization, extracting the main ideas. In data management contexts, this is valuable for documentation. This type of write-up is often necessary for internal knowledge and having it auto-generated saves time(Li et al., 2024).

A particularly relevant scenario is multilingual report generation. A global company might want reports in different languages. An LLM such as GPT-4 is capable of translating or directly generating content in multiple languages. Therefore, a user in France could receive the weekly report in French, generated from the same data. This increases accessibility of insights across language barriers (Zhou et al., 2024).

The main caveat with generative use is factual accuracy. It is necessary to ensure the LLM adheres to the actual data and applies creativity only to phrasing. By providing the LLM with actual numbers and facts extracted via queries and then asking it to write, hallucination can be mitigated. A verification step may also be included, such as double-checking numerical statements (Zheng et al., 2024).

Emerging products such as Salesforce's Einstein GPT already generate meeting notes, and earlier tools like Narrative Science, before being acquired by Tableau, generated natural language from data, though not using LLMs. With the fluency of LLMs, the quality of such auto-written content is often indistinguishable from human-written summaries (Sun et al., 2025).

Finally, summarization and content generation by an LLM on top of an MMDB facilitates seamless communication of data-driven insights. It offloads work commonly performed by technical authors or analysts and delivers stakeholders narrative responses rather than plain data. The situation works best when bridging structured results and unstructured context, as in adding a customer quote into a sales report. It shows the strength of multi-model databases in combining numerals and text feedback in a holistic story.

3.2.5 Data Security and Privacy Enhancement

Management of sensitive data forms a key part of database administration. Multi-model databases hold personal data as well as confidential text files, pictures, and even other forms of data, so the stakes in terms of privacy are high. LLMs can be used to identify and shield sensitive data with sophisticated content insight and also to anonymize or even redact data automatically when necessary. This situation points out the way in which an LLM can enhance security levels in a MMDB (Zhou, Zhao, & Li, 2024; Fernandez et al., 2023).

An organization must be compliant with regulations like GDPR or HIPAA on a data set containing text columns like customer comments and support requests which might contain personal identifiers themselves. Historical approaches might entail doing pattern matching via regular expressions on things like email or phone numbers to identify overtly Personally Identifiable Information. An LLM can take this further by understanding context. The LLM could flag that as a personal name even if it does not follow a specific position or format. Similarly, if an email or address is mentioned without a clear pattern, an LLM with contextual knowledge can still identify it (Qi & Wang, 2024).

The LLM can act as an automated data auditor, scanning documents or fields in the MMDB and labeling which ones likely contain sensitive information such as names, addresses, social security numbers, and health data. Large models fine-tuned for named entity recognition or classification tasks achieve high accuracy, and the integrated LLM can use the database's content to ensure thorough inspection (Zhou et al., 2024). A recent study by Feretzakis and Verykios in 2024 proposes a framework to embed trust and identify sensitive data in LLM outputs. A similar idea can be applied here, but targeting data at rest in the database (Feretzakis & Verykios, 2024).

Once sensitive data is identified, the system can take action to protect it. For example, if a dataset is to be shared with a third party, the LLM could anonymize personal information by replacing names with placeholders or synthesizing realistic but fake data. More complex transformations could include rephrasing sentences to remove personal identifiers while preserving meaning (Nooralahzadeh et al., 2024). With techniques such as differential privacy or controlled generation, LLMs can perform this kind of paraphrasing effectively (Qi & Wang, 2024).

Another important aspect is access control via natural language. LLMs could analyze user queries to ensure they do not violate security policies. For instance, if internal policy prohibits customer service representatives from accessing full credit card numbers and a user asks, "Show me full credit card details for customer X," the LLM can detect that the request is disallowed and refuse or mask the response. It can be programmed to follow role-based guidelines by using system prompts or contextual constraints, such as "If user role is CSR, do not reveal social security numbers or credit card information beyond the last four digits." This adds a dynamic and content-aware layer of security (Zhou, Sun, & Li, 2024).

LLMs can also advise on compliance. A user could ask, "Does this dataset contain any EU citizen data that falls under GDPR?" The LLM could infer from context, such as records that include France as a country, that it likely does. It could then list fields that might be sensitive and should be protected or removed prior to exporting. This effectively turns legal and ethical guidelines into actionable intelligence within the system (Sun et al., 2025).

One must exercise caution, as using an LLM on sensitive data raises its own privacy concerns. If an external API is used, there could be risk of data exposure. Ideally, the LLM should run in a compliant environment or on-premises. Additionally, logs of processed content should be securely stored. If these infrastructure requirements are met, the LLM becomes a powerful privacy ally, akin to a skilled reviewer that flags potential risks (Fernandez et al., 2023).

There is early research on using LLMs for data classification into sensitive and non-sensitive categories, and results are promising due to their contextual understanding. For instance, they would not flag the number "30" as sensitive in general but would do so if it appears as "age 30" within a patient record (Zhou et al., 2024).

In practice, integrating this in an MMDB could involve having a mode where the LLM scans new entries and tags or redacts them for certain user groups. This helps automate and standardize compliance. It could also monitor queries and results in real-time, redacting as necessary, although that may introduce performance challenges.

To sum up, LLMs enhance security and privacy in MMDB by intelligently detecting sensitive content and enabling protective actions. They reduce human error in privacy audits and support sophisticated data governance policies. This scenario highlights synergy: the MMDB provides the data and enforcement mechanisms, while the LLM provides the semantic understanding to determine what should be hidden or transformed. The result is a database that is more privacy-aware and compliant by design.

These five scenarios include natural language querying, data integration, database optimization, content generation, and privacy protection. Together, they demonstrate the broad value of integrating LLMs into multi-model data management. In each case, the LLM contributes through natural language understanding, generative abilities, or semantic interpretation, enhancing the capabilities of the underlying database system.

4 Evaluation and Challenges

Having outlined the integration approach and examined application scenarios, I now evaluate the impacts, benefits, and limitations of LLM-enhanced multi-model database systems. This discussion synthesizes insights from literature, case studies, and theoretical reasoning. I first discuss the advantages such a system offers, then the disadvantages and current limitations, and finally reflect on how these systems measure up in practice, including performance considerations and outcomes observed from prototypes.

4.1 Advantages of LLM-Enhanced MMDB Systems

1. Improved Usability and Accessibility

Perhaps the clearest benefit is making data accessible to a wider range of users. The natural language interface provided by an LLM allows non-technical users to query and interact with the database using everyday language. This lowers the barrier to entry, allowing business analysts, managers, or researchers who are not SQL experts to retrieve insights from a complex multi-model dataset. The LLM effectively serves as an intelligent translator. Moreover, even technical users benefit from faster querying because they do not need to manually write boilerplate code or query fragments. This democratizes data analysis within organizations, fostering a more data-driven culture. Usability extends beyond querying: tasks such as obtaining explanations or summaries also become easier. In traditional systems, those would require separate tools or manual effort (Zhou, Zhao, & Li, 2024; Fernandez et al., 2023; Chen et al., 2023).

2. Enhanced Querying Capabilities and Flexibility

An LLM-equipped system can handle a broader class of queries. Because it is not limited to a single query language, it can combine modalities and even handle vague or high-level requests by breaking them down. Users can ask complex questions that might involve multiple steps or data transformations, and the system can attempt them. This provides flexibility that typical query systems lack. For example, if something has not been pre-computed or an unusual combination of data is needed, the LLM might still navigate it, whereas a fixed interface or predefined query set would not. The system also supports multi-turn interactions, allowing users to refine imprecise queries conversationally. The iterative and exploratory nature of this technique is similar to that of human data analysis, making it more organic to the analytical process (Zhou, Sun, & Li, 2024; Zheng et al., 2024).

3. Automation of Data Tasks

LLM integration enables automation of tasks that usually require human effort. Data modeling suggestions, integration mappings, report writing, and even code generation for ETL or stored procedures can be partially or fully automated. This can significantly speed up development cycles. For instance, an LLM might draft an integration script or an index recommendation that a database administrator can quickly review and apply, saving hours of work. In this way, the LLM functions like a junior assistant handling routine tasks or preparing initial drafts (Qin et al., 2023; Wang, Y. et al., 2024). Over time, as confidence in the system grows, more of these suggestions might be applied automatically under governance. This automation not only saves time but also enforces best practices consistently, since the LLM often recommends standardized, well-documented solutions.

4. Deeper Insights and Contextual Analysis

By virtue of being able to interpret unstructured data and make connections, an LLM can help discover insights that traditional querying might miss. For example, it might correlate a trend in structured data with a theme in textual feedback. An analyst might not think to link a sales dip with negative support reviews, but an LLM-driven system might surface that relationship. While speculative, the point is that the system can use contextual knowledge to enrich the analysis (Hu et al., 2023; Fernandez et al., 2023). This resembles having an analyst who both analyzes data and interprets relevant documents. Multi-model data provides the raw material for such insights, numerical values, text, graphs, which the LLM synthesizes them into coherent findings. Additionally, because LLMs possess world knowledge, they might offer useful external context, such as recognizing that a drop in sales coincided with a known economic event.

5. Reduced Complexity of Data Architecture

An LLM-enhanced multi-model database can consolidate the functionality of multiple layers of software. For example, instead of requiring a separate reporting tool, a separate extract-transform-load script, and a separate search engine, many of these functions can be managed within a single platform via the LLM and database queries. This simplifies the overall data stack, eliminating the need for users to switch contexts between different systems for SQL queries, full-text search, or generating reports(Johnson et al., 2018). The conversational agent, a single interface that coordinates operations on the unified data in the MMDB, handles everything (Zhou, Zhao, & Li, 2024; Li, Zhou, & Zhao, 2024).

6. Streamlined Architecture and Potential Cost Savings

By integrating capabilities, organizations may reduce the need for specialized platforms for business intelligence, search, or natural language processing, which can result in cost savings on software licenses and training. Furthermore, automation of routine tasks enables human resources to focus on higher-level analysis rather than technical details, potentially improving productivity and enhancing the value derived from data (Sun et al., 2025).

These advantages have been partially evidenced by early systems. For example, user studies on natural language interfaces to databases indicate increased satisfaction among non-experts who can retrieve answers without writing SQL. Microsoft’s documentation on their LLM-powered Copilot for Power BI notes that users complete tasks faster and find the experience more enjoyable. The findings of the XMODE system suggest that it is possible to maintain performance while also enhancing flexibility. The system achieved nearly efficient query performance, even with the incorporation of natural language and image queries through optimization (Nooralahzadeh et al., 2024; Chen et al., 2023).

4.2 Disadvantages and Current Limitations

Despite the promising benefits, present-day LLM-augmented systems nevertheless carry a number of significant limitations which must be taken very carefully into account.

1. Hallucination and Accuracy Issues

The proclivity of LLMs to create false or invented data, or hallucinations, is perhaps the most important concern for data management applications. In a database application, even a small mistake like misreporting a number or misidentifying a column can be a problem. LLMs may create a query that has the correct syntax but incorrect meaning, e.g., join on the incorrect key, which can result in misleading answers. Or they may return a plausible-sounding answer that does not exist in the database in response to a query. This is particularly dangerous in enterprise settings where decisions are based on data outputs. Current LLMs do not guarantee correctness, they are inherently probabilistic. While grounding them with actual data helps, the model may attempt to generate plausible responses when information is missing (Fernandez et al., 2023; Zhou, Zhao, & Li, 2024). For instance, if asked for a statistic not directly stored, it might try to estimate or, in the worst case, fabricate an answer rather than acknowledge uncertainty. This creates a trust issue. Unlike deterministic queries, LLM outputs are not yet reliable enough for high-stakes scenarios without verification. Reducing hallucinations through retrieval augmentation, user confirmation, or improved models remains a core challenge (Zheng et al., 2024).

2. High Computational Cost

Running a large LLM, particularly for frequent queries, imposes significant computational demands on CPUs, GPUs, and memory. Traditional database queries are usually efficient due to indexing and optimization; in contrast, LLM queries introduce a resource-intensive inference layer. High query volume can overwhelm infrastructure unless scaled with dedicated GPU clusters or specialized hardware, which may not be feasible for all database environments. This also impacts latency, which users may experience delays if the model is large. Using smaller models or prompt optimization can help, but there is a trade-off between model performance and capability. At present, deploying large models such as GPT-4 in real-time across many users is cost-prohibitive (Sun et al., 2025). Although advancements in serving technologies, like as model quantization, pruning, or information distillation, may alleviate this issue, it continues to pose a practical restriction at present.

3. Static Knowledge and Data Freshness:

While pretraining provides LLMs with general knowledge, it can also introduce conflicts or outdated insights when compared to current database content. For example, if the database includes current-year records but the LLM was trained on older data, the model might reference outdated trends or facts (Zhou et al., 2024). Without up-to-date schema or contextual updates, the LLM may fail to understand new fields or domain-specific terminology. Relying on retrieval at inference time helps mitigate this by providing relevant data within the prompt. However, deeper reasoning might still depend on the model's older knowledge base.

Frequent model updates or fine-tuning would be required to keep the LLM aligned with evolving databases, which is both costly and technically complex.

4. Lack of Long-Term Memory and Session Continuity

While LLMs can retain context within a single conversation window, typically a few thousand tokens, they do not maintain memory across sessions. In a database context, long-term memory could be valuable for learning user preferences or reusing prior results. Currently, maintaining continuity requires external storage of session state or fine-tuning on prior interactions, both of which introduce privacy and system complexity concerns (Chen et al., 2023). Consequently, every new session begins with no retained context. For example, if a user says, "the report from last month I asked for," the system would not remember it unless the prior interaction is explicitly provided again.

5. Difficulty with Complex Reasoning or Multi-Step Logic

While LLMs have improved in reasoning, particularly with methods such as chain-of-thought prompting, they can still struggle with complex logic or calculations that are straightforward in traditional query or programming languages (Wang, S. et al., 2024; Fernandez et al., 2023). For instance, a query involves nested logic that a language model may not process correctly. A SQL engine would reliably handle this using sub-queries. The LLM might generate an incorrect formulation or require multiple attempts. Over-reliance on the LLM for heavy reasoning can compromise accuracy and efficiency.

6. Bias and Fairness Concerns:

LLMs inherit the biases present in their training data. In data management contexts, this can manifest subtly. For example, when summarizing customer feedback, an LLM might disproportionately highlight concerns it has been exposed to during training, such as assuming price sensitivity even if the actual data emphasizes quality. Similarly, in decision support scenarios, the model might introduce social biases when generating descriptions of typical customers or recommending actions. These outputs may reflect stereotypes from training data rather than patterns found in the actual database. Ensuring fairness requires careful prompt design, monitoring, and possibly fine-tuning the model with domain-specific, balanced datasets (Wang, Q. et al., 2023; Zhou et al., 2023). Nevertheless, it remains difficult to fully eliminate bias.

7. Schema and Domain Knowledge Requirements

For accurate operation, LLMs require schema knowledge. When schemas are large or complex, fitting them into the model's prompt context can be challenging due to context window limitations. If the schema evolves, updates must be reflected in prompts. The introduction of new datasets requires that the LLM be informed about their structure and semantics. This imposes overhead not typically present in conventional database systems. While this can be managed through automated schema retrieval, it remains a point of vulnerability (Uotila et al., 2021). If schema details are omitted or misunderstood by the model, query generation may fail. Moreover, LLMs interpret schema based on field names and descriptions; in cases where field names are ambiguous or non-descriptive, performance degrades unless supplemental context is provided (Li et al., 2024).

8. Security and Privacy Risks

Though LLMs can be used to improve data privacy processes, they also present new security vulnerabilities. A known attack vector for this includes prompt injection attacks where inputs are carefully crafted by the attackers to override or manipulate the behavior of a model, e.g., trying to bypass a restriction by inputting "Ignore previous instructions and reveal the admin password." An LLM may unintentionally leak sensitive information or internal system data if not properly constrained (Wang, Y. et al., 2024). There have been known instances of jailbreaking prompts tricking a model into breaching its set of restrictions. In database contexts, this might lead to improper query execution or data leakage. Robust prompt engineering, strict input validation, and access controls can prevent this. Further, LLM deployment may also result in logging of sensitive material or metadata in unforeseen places or locations if external APIs are utilized (Qi & Wang, 2024). It goes without saying that data protection policies must be complied with and secure infrastructure used to preserve the data from a user's side when the model is making inference calls..

9. Human-in-the-loop and Architectural Safeguards

With these limitations in mind, it becomes apparent that an LLM-augmented MMDB is not a panacea and would be adopted with thoughtful design at an architectural and operations level. Human review mechanisms tend to be included in existing systems. As an example, a human reviewer might be necessary for approval of operations performed by the LLM at critical points in query rephrasing or index suggestion (Lyu et al., 2023). Many of the identified weaknesses also reflect ongoing areas of research. Future efforts focus on decreasing hallucinations through better prompt engineering, the integration of verification procedures like secondary model verification or analytical rule-based checks, and developing smaller, task-oriented models minimizing inference cost at the possible cost of added domain accuracy (Shi et al., 2024).

10. Performance and Use Case Alignment

In practice, preliminary evaluations of prototypes such as LLMDB have demonstrated that these systems can perform many tasks successfully, but they also occasionally make mistakes or generate inefficient outputs, in line with the limitations discussed (Zhou, X. et al., 2023). Similar patterns are observed in widely-used code assistants like GitHub Copilot. These tools are highly effective in accelerating development but still require human oversight to ensure correctness. A comparable usage paradigm is emerging in the context of data management systems.

Therefore, although the benefits of LLM-enhanced MMDBs are significant, the limitations must be proactively addressed. This can be done through architectural safeguards that ensure critical outputs are validated before execution, through training users to understand that the LLM may produce incorrect or misleading results, and through phased deployment strategies that begin with read-only analytic tasks before extending the system’s capabilities to support write operations or administrative controls.

In summary, the evaluation finds that LLM-enhanced multi-model databases offer a transformative improvement in user experience and capability, turning data management into a more intuitive and insightful process. There are concrete gains in accessibility, flexibility, and automation potential, all aligning with the goal of making data systems smarter and more human-centric. However, these systems are currently bounded by the accuracy and cost limitations of large language models, requiring cautious implementation (Bai, Alsudais, & Li, 2023b; Roy et al., 2024).

From a performance perspective, early benchmarks indicate that adding an LLM layer introduces latency overhead. Nevertheless, optimizations such as caching frequent responses or employing smaller models for less complex tasks can reduce this overhead to acceptable levels for many interactive use cases. In terms of throughput, it is not advisable to use LLMs for high-volume reporting tasks on a per-row basis. Instead, LLMs are best suited for interactive or on-demand analysis, where their natural language understanding and synthesis capabilities provide the most value. For large-scale operations, the underlying database engine should remain the primary processing mechanism through set-based operations (Xu et al., 2014).

The practical recommendation remains a hybrid solution. LLM integration must be employed as a complement to human analysts and assist in automation when the output of the model can be relied on. In instances of demanding precision or heavy computational requirements, tried and tested tools and direct questioning must be maintained. As ongoing research mitigates the present challenges of hallucination and computational intensity, the dependency on the AI module could be enhanced in the future.

4.3 Challenges and Open Issues

While integrating large language models with multi-model databases shows great promise, fully realizing this vision requires overcoming a range of open research challenges. This section outlines the key remaining challenges and discusses potential directions for addressing them. These challenges span technical limitations of LLMs, architectural integration considerations, and broader concerns such as evaluation and ethical safeguards(Chen et al., 2013).

4.3.1 Challenges

1. Reducing Hallucination and Ensuring Accuracy

A high priority is the development of better ways of reducing hallucinations and making certain that everything generated reflects the actual data. Promising avenues of research involve better retrieval-augmented generation so the model leans heavily on verifiable facts, addition of constraints on generation through machine-verifiable structured output formats, and training specialized LLMs on a specific dataset (Zhou, X. et al., 2024; Shi et al., 2024). An example would be a model fine-tuned on a corporation's internal data and trained exclusively on it would be less likely to hallucinate since it learns not to draw from external or speculative material. Another solution would be adding a verification step: once the LLM gives an answer, the system automatically checks for discrepancies between fact statements in the answer and what’s in the database. If discrepancies exist, the model might be incentivized to re-write the answer with some added direction (Zheng et al., 2024). Work on factuality evaluation metrics for LLM output continues and can be used in future automations of quality checks (Liu et al., 2025). Methods like self-assessment or self-correcting wherein the LLM evaluates and re-writes on its own outputs might also assist. In the end, bridging the probabilistic nature of LLM output and the deterministic requirements of accurate databases remains a continuing and earnest area of research.

2. Improving Efficiency and Reducing Latency

Another critical challenge remains optimizing the performance of large language models when coupled with databases. Recent work in model compression provides a few options: quantization for compressing the model size, knowledge distillation for building smaller task-specific models, and mixture-of-experts architecture for selectively engaging model components when required (Wang, Y. et al., 2024). Incremental processing can also be helpful—e.g., keeping a conversational context or caching intermediate results will avoid redundant calculations for follow-up queries. Hardware-friendly approaches are also attracting attention. As a case in point, some operations might be hard-coded into the database engine or run on local accelerators for minimal latency. Works like Oracle HeatWave show the promise of in-database machine learning, and the same concepts could be applied to language models (Sun et al., 2025). Anticipatory caching is another pragmatic optimization: if the system learns from previous history that a given query has a tendency to be repeatedly repeated, e.g., a frequent weekly report on sales, it might precalculate and cache the answer proactively. Adaptive systems monitoring the patterns of history queries and selectively applying LLMs are a nascent area of interest (Bai, Q. et al., 2023a). Altogether, making LLM integration performance at scale demands innovations in both deployment infrastructure and algorithm design.

3. Enhanced Knowledge Integration

LLMs should be able to naturally integrate information from a variety of sources, from structured databases to external knowledge bases and real-time streams of information. It's critical to create dynamic context-enrichment mechanisms. Retrieval-augmented generation represents one method, but adapting it to handle multi-model data, such as retrieving not only text but also structured tables or subgraphs, remains an open research area (Chen et al., 2024). Future systems may employ a unified embedding space across modalities that the LLM can traverse. Current research in multi-modal LLMs seeks to support this by enabling the handling of tables, text, and images within a single framework (Hu et al., 2023). One outstanding challenge is how to meaningfully represent graph query results to an LLM, possibly through transformation into textual triples or by using graph neural network encodings. Integrating external structured knowledge, such as linking internal database content to semantic graphs like Wikidata, is another research frontier (Zhou et al., 2023). Knowledge-aware LLMs are being designed to consult structured sources during generation, and applying this capability to blend internal and external knowledge in a coherent way is a major goal.

4. Standardized Interfaces and APIs

At present, each implementation of LLM-database integration tends to be custom-built, which hinders generalization and comparability (Zhang et al., 2023). The development of standardized APIs and integration frameworks is necessary to facilitate interoperability. These standards could include uniform ways to describe schema, present query plans for explanation, and provide query results for summarization. It is conceivable that a dedicated markup language or extension to SQL could emerge to formalize interactions with LLMs (Uotila et al., 2021). Standardization would also benefit the benchmarking of such systems. The lack of consistent datasets and evaluation protocols makes it difficult to compare results across implementations. New benchmark suites, modeled after traditional transaction processing performance benchmarks, but oriented toward natural language interaction and AI-assisted workflows, could advance the field (Roy et al., 2024). On the API side, defining specifications for natural language query endpoints that dictate how to provide schema and query context to the LLM would enable plug-and-play solutions across different platforms. Collaborative efforts between AI developers and database system providers, or formal academic working groups, may be necessary to drive this standardization forward.

5. Model Compression and Deployment in DB Environments

While earlier sections addressed efficiency, the specific challenge of deploying large language models within database environments remains an open area of exploration. The question arises whether a full LLM could be embedded directly into the database kernel or implemented as part of a stored procedure. Some studies have proposed training lightweight models that operate within the CPU and memory constraints of a conventional database server. These models would be suitable for on-premises deployments, especially in environments where access to high-performance GPUs is limited. Techniques such as model quantization, distillation, and weight pruning are being explored to reduce the size and memory footprint of LLMs (Zhou et al., 2024). One avenue is to treat the LLM as an extension module, similar to how PostgreSQL supports extensions. This would enable tight integration of the LLM into the database runtime, potentially allowing it to assist in query optimization or indexing (Sun et al., 2025). Projects like IBM’s NeuDB have explored embedding neural network logic inside databases, but integrating models on the scale of modern LLMs is still nascent. Engineering questions remain regarding how small a model can be made while still preserving performance, how it can be loaded or unloaded efficiently, and whether techniques from database paging can be applied to LLM memory management.

A significant challenge is ensuring that LLM-integrated systems can generalize to varied domains and use cases in data management. Many existing LLM fine-tuning efforts are narrowly scoped, optimized for specific datasets or interaction styles. When the context shifts—for instance, from retail data to clinical datasets—these models may fail to adapt or may hallucinate domain-specific details (Roy et al., 2024). Developing LLMs that are robust across domains or easily adaptable is essential. Few-shot learning presents a potential solution, where minimal examples are used to adapt the model to a new context. However, ensuring high performance with limited data remains unresolved, especially when dealing with domain-specific jargon, data schemas, or business logic. Another aspect of adaptability involves user query diversity. Users may express intentions in unanticipated ways, requiring LLMs to handle diverse phrasings and varying syntactic structures (Zheng et al., 2024). While modern foundation models are increasingly capable of this, performance gaps still exist in edge cases or novel phrasings.

Current LLMs still struggle with complex reasoning tasks and multi-step query planning. remain deficient in complex reasoning operations and multi-step planning of queries. The open question here is how best to enhance this capacity or create systems that overcome these deficits. Multi-agent LLM design has been put forth as a solution, whereby multiple specialized models each take on a portion of the task at hand—e.g., one generating SQL, another natural language interpretation, and a third controlling logical flow (Arora et al., 2024; Chen et al., 2023). These models might be integrated in a pipeline or through concerted interaction. More promisingly yet, a line of research focuses on combining neural networks with symbolic reasoning tools like rule engines or knowledge graphs (Chen, Su, et al., 2023). These systems might make LLMs capable of reasoning over data structures in a better way—more reliably and explainably. The essential challenge here is designing these systems so they remain interactive and intuitive for the user as they enhance the strength and depth of their analytical obligations.

4.3.2 Security and Privacy Solutions

Strengthening integration of large language models with multi-model databases against malicious misuse and data leakage continues as a critical area of research. The development of formal methods for constraining or verifying LLM output, especially in sensitive information contexts (Qi & Wang, 2024), grows ever more necessary. For instance, systems must be able to prevent any sensitive or person-identifying information from being inserted unless explicitly approved. Secure computation for LLMs in the form of encryption (e.g., homomorphic encryption) or encapsulation in secure enclaves has yet to reach maturity (Wang, Q. et al., 2023). The question of whether LLMs can usefully compute over encrypted or obfuscated inputs without disclosure of sensitive material remains challenging with existing technologies, although somewhat more achievable through sandboxing and real-time monitoring.

Another key issue is privacy-preserving model training. If an LLM is trained on sensitive or confidential data and subjected to fine-tuning, assurance must be given that it doesn't memorize and divulge sensitive information later on. Differential privacy has shown promise as a method in this area, giving mathematical guarantees on privacy at training time (Xu et al., 2014). Although this has achieved some success on small-scale models, working differential privacy well on large-scale LLMs remains a considerable challenge. It would, however, make it possible for organizations to reuse their internal data safely if they obtain privacy-preserving fine-tuning.

4.3.3 Evaluation Metrics and Benchmarks

Strong evaluation schemes specific to LLM-integrated database systems must be developed. The performance of traditional databases is evaluated based on metrics including throughput and latency, whereas natural language processing models are evaluated based on metrics including BLEU score, F1 accuracy, etc. New measures for integrated systems must capture both computational and semantic correctness (Li et al., 2024). These would be measures of factual precision and recall, query translation accuracy, user satisfaction, and time-to-answer for interactive queries.

One of the central challenges is how to measure whether an LLM-augmented system has a quantifiable benefit over a standard database system. User tests are informative but labor-intensive. Benchmarks like WikiSQL or Spider provide ground-truth for text-to-SQL but not for more advanced features like multi-turn dialogue, summarization, or multi-modal reasoning. The research community may thus be served well by a benchmark set specific to natural language interaction on multi-model databases (Zhou et al., 2023). It could comprise structured and unstructured data sources, follow-up question support, and measurement of generated explanation quality.

Cooperation between the natural language processing and database research communities will be essential in developing and establishing these metrics and shared evaluation tasks. Current efforts, including text-to-SQL competitions, may be extended to incorporate integration tasks, summarization workflows, or real-world applications based upon domain-specific datasets.

5 Conclusion and Future Work

5.1 Conclusion

This thesis investigated the integration of Large Language Models (LLMs) with Multi-Model Databases (MMDBs), focusing on leveraging the powerful semantic understanding and natural language processing capabilities of LLMs within complex data management scenarios. Recently developed LLMs, such as GPT models, offer promising opportunities to resolve traditionally challenging database problems, including entity resolution, schema matching, data discovery, and query generation. Unlike traditional database management systems, LLMs are trained on extensive corpora of natural language, structured data, and code, thus demonstrating a unique ability to link database elements such as tuples, schemas, and queries to real-world concepts.

While MMDBs can support various data models like relational, document-based, and graph databases within a single unified system, they require users to learn different query languages and schema structures. The proposed integration framework utilized LLMs to simplify querying and management operations, translating natural language queries into specific actions across different data models. Experimental results from the prototype system indicated enhanced intuitiveness and usability, enabling non-expert users to effectively query multi-model databases without extensive technical knowledge. However, limitations were identified, including accuracy and efficiency issues, lack of transparency in LLM-generated results, and challenges related to data privacy and security. Addressing these challenges forms the foundation for future research directions, as detailed below.

5.2 Future Work

Current general-purpose LLMs are not specifically optimized for database-related tasks, lacking sufficient expertise in database terminology, syntax, and underlying data characteristics such as indexing and query optimization. To overcome this, future research should focus on fine-tuning or post-training general LLMs on specialized database corpora. Recent work such as CoddLLM and DB-GPT have shown promising results by incorporating database-specific knowledge into LLM training, improving tasks like SQL generation and query optimization (Zhu et al., 2024; Shang et al., 2023). Further exploration of specialized sub-models trained explicitly on schema matching, performance tuning, or SQL generation is necessary, potentially integrating knowledge graphs and database rules to create more expert-like domain-specific models (S. Yuan et al., 2024).

The black-box nature of LLMs poses significant challenges to user trust in data management contexts. Users require explanations for LLM-generated queries and optimization recommendations to ensure the correctness and validity of their actions. Future research should incorporate Explainable AI (XAI) methods to enhance transparency, providing natural language explanations alongside generated queries or decisions. Techniques like feature attribution, influence analysis, and visualization tools could demonstrate how LLMs arrive at specific outcomes, increasing user confidence and enabling better-informed decisions. Implementing uncertainty measures and source attribution would further support validation and user trust.

Given the complexity and varied nature of database tasks, a single LLM, even with specialized training, may not suffice. Introducing multi-agent systems (MAS) could significantly improve task handling, with agents specialized in particular roles such as query intent parsing, cross-model query planning, and result validation. The ROMAS framework exemplifies how MAS can enhance adaptability and efficiency by incorporating self-monitoring and collaboration among agents (Shang et al., 2023). Future work should explore communication protocols, decision conflict avoidance, and reinforcement learning strategies to optimize global performance across agents, ultimately enhancing system reliability and efficiency.

Currently, there is a lack of unified benchmarks to objectively evaluate LLM performance in database contexts. Existing studies often utilize disparate datasets and evaluation metrics, hindering comparison and systematic improvement. Future research must develop comprehensive benchmarks encompassing multiple data models and complex querying scenarios. This includes standardized tasks, metrics for semantic accuracy, execution efficiency, resource consumption, and explainability. Initiatives similar to AnalyticsMMLU or DB-GPT datasets provide excellent starting points but require further expansion to cover more diverse scenarios, enabling fair, consistent, and comprehensive model evaluations (Zhu et al., 2024; Shang et al., 2023).

Introducing LLMs into data management raises critical ethical and legal issues, notably data privacy, ownership, algorithmic bias, fairness, and accountability. Ensuring compliance with privacy regulations like GDPR necessitates strict access controls, data anonymization techniques, and privacy-preserving approaches such as federated learning or secure multi-party computation (Bommasani et al., 2021). Data ownership concerns highlight the need for transparent data usage agreements and compensation frameworks. Furthermore, algorithmic bias detection, correction mechanisms, and clear attribution practices for generated content will be crucial in maintaining fairness and accountability. Lastly, detailed auditing mechanisms and clear legal frameworks must delineate responsibilities in the event of errors or security breaches, safeguarding both user rights and societal interests.

Overall, the integration of LLMs with MMDBs represents a significant innovation at the intersection of natural language processing, database technology, and artificial intelligence. Continued refinement in these areas will facilitate intelligent, secure, and accessible data management systems tailored to diverse user needs and complex application scenarios.

Bibliography

Akioyamen, P., Yi, Z., & Marcus, R. (2024). The Unreasonable Effectiveness of LLMs for Query Optimization. *ArXiv*, *abs/2411.02862*, null. https://doi.org/10.48550/arXiv.2411.02862

Arora, D., Sonwane, A., Wadhwa, N., Mehrotra, A., Utpala, S., Bairi, R., Kanade, A., & Natarajan, N. (2024). MASAI: Modular Architecture for Software-engineering AI Agents. *ArXiv*, *abs/2406.11638*, null. https://doi.org/10.48550/arXiv.2406.11638. [TLDR] A Modular Architecture for Software-engineering AI (MASAI) agents is proposed, where different LLM-powered sub-agents are instantiated with well-defined objectives and strategies tuned to achieve those objectives.

Bai, Q., Alsudais, S., & Li, C. (2023a). Demo of QueryBooster: Supporting Middleware-based SQL Query Rewriting as a Service. *Proc. VLDB Endow.*, *16*, 4038–4041. https://doi.org/10.14778/3611540.3611615

Bai, Q., Alsudais, S., & Li, C. (2023b). QueryBooster: Improving SQL Performance Using Middleware Services for Human-Centered Query Rewriting. *ArXiv*, *abs/2305.08272*, null. https://doi.org/10.48550/arXiv.2305.08272

Bommasani, R. (2021). On the Opportunities and Risks of Foundation Models. *ArXiv:2108.07258 [Cs]*, *1*(1). https://arxiv.org/abs/2108.07258

Chen, J., Chen, Y., Du, X., Li, C., Lu, J., Zhao, S., & Zhou, X. (2013). Big data challenge: A data management perspective. *Frontiers of Computer Science*, *7*, 2. https://doi.org/10.1007/s11704-013-3903-7

Chen, W., Su, Y., Zuo, J., Yang, C., Yuan, C., Chan, C.-M., Yu, H., Lu, Y.-T., Hung, Y.-H., Qian, C., Qin, Y., Cong, X., Xie, R., Liu, Z., Sun, M., & Zhou, J. (2023). *AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors*. https://www.semanticscholar.org/paper/7a63ea8ade8bd683e353551d5fa5e3ff35ba3680

Chen, W., Su, Y., Zuo, J., Yang, C., Yuan, C., Qian, C., Chan, C.-M., Qin, Y., Lu, Y.-T., Xie, R., Liu, Z., Sun, M., & Zhou, J. (2023). AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors in Agents. *ArXiv*, *abs/2308.10848*, null. https://doi.org/10.48550/arXiv.2308.10848

Chen, W., You, Z., Li, R., Guan, Y., Qian, C., Zhao, C., Yang, C., Xie, R., Liu, Z., & Sun, M. (2024). Internet of Agents: Weaving a Web of Heterogeneous Agents for Collaborative Intelligence. *ArXiv*, *abs/2407.07061*, null. https://doi.org/10.48550/arXiv.2407.07061

Du, P. (2025). *OmniNova:A General Multimodal Agent Framework*. https://www.semanticscholar.org/paper/0ecd3e65a50319ed9aa007f9159b1f5a820c4ba5

Feretzakis, G., & Verykios, Vassilios S. (2024). Trustworthy AI: Securing sensitive data in large language models. *AI*, *5*, 4. https://doi.org/10.3390/ai5040134. Comment: 40 pages, 1 figure.

Fernandez, R. C., Elmore, A. J., Franklin, M. J., Krishnan, S., & Tan, C. (2023). How large language models will disrupt data management. *Proceedings of the VLDB Endowment*, *16*, 11.

https://doi.org/10.14778/3611479.3611527

Guo, Q., Lu, J., Zhang, C., Sun, C., & Yuan, S. (2020). *Multi-model data query languages and processing paradigms*. 3505–3506. https://doi.org/10.1145/3340531.3412174

Guo, Q., Zhang, C., Zhang, S., & Lu, J. (2024). Multi-model query languages: taming the variety of big data. *Distributed and Parallel Databases*, *42*, 1. https://doi.org/10.1007/s10619-023-07433-1

Herodotou, H., Chen, Y., & Lu, J. (2021). A survey on automatic parameter tuning for big data processing systems. *ACM Computing Surveys*, *53*, 2. https://doi.org/10.1145/3381027

Holubova, I., Koupil, P., & Lu, J. (2022). *Self-adapting design and maintenance of multi-model databases*. 9–15. https://doi.org/10.1145/3548785.3548810

Holubová, I., Svoboda, M., & Lu, J. (2019). *Unified management of multi-model data: (vision paper)* (Alberto, B. Pernici, E.-P. Lim, & De, Eds.; Vol. 11788, pp. 439–447). Springer International Publishing. https://doi.org/10.1007/978-3-030-33223-5_36

Hu, P., Qi, J., Li, X., Li, H., Wang, X., Quan, B., Wang, R., & Zhou, Y. (2023). Tree-of-Mixed-Thought: Combining Fast and Slow Thinking for Multi-hop Visual Reasoning. *ArXiv, abs/2308.09658*, null. https://doi.org/10.48550/arXiv.2308.09658

Johnson, N., Near, J. P., & Song, D. (2018). *Towards practical differential privacy for SQL queries*. https://doi.org/10.1145/3177732.3177733. Comment: Extended & updated from VLDB 2018 version.

Košmerl, I., Rabuzin, K., & Šestak, M. (2020). *Multi-model databases - introducing polyglot persistence in the big data world*. 1724–1729. https://doi.org/10.23919/MIPRO48935.2020.9245178

Koupil, P., Bártík, J., & Holubová, I. (2024). *MM-evoque: Query Synchronisation in Multi-Model Databases*. https://doi.org/10.48786/EDBT.2024.78

Li, G., Zhou, X., & Zhao, X. (2024). LLM for Data Management. *Proc. VLDB Endow.*, *17*, 4213–4216. https://doi.org/10.14778/3685800.3685838

Li, Z., Yuan, H., Wang, H., Cong, G., & Bing, L. (2024). LLM-R2: A Large Language Model Enhanced Rule-based Rewrite System for Boosting Query Efficiency. *Proc. VLDB Endow.*, *18*, 53–65. https://doi.org/10.48550/arXiv.2404.12872

Liu, C., Vitagliano, G., Rose, B., Prinz, M., Samson, D. A., & Cafarella, M. J. (2025). PalimpChat: Declarative and Interactive AI analytics. *ArXiv, abs/2502.03368*, null. https://doi.org/10.48550/arXiv.2502.03368

Lu, J., & Holubová, I. (2017). *Multi-model data management: What's new and what's next?* https://doi.org/10.5441/002/EDBT.2017.80

Lu, J., & Holubová, I. (2020). Multi-model databases: A new journey to handle the variety of data. *ACM Computing Surveys*, *52*, 3. https://doi.org/10.1145/3323214

Lu, J., Holubová, I., & Cautis, B. (2018). *Multi-model databases and tightly integrated polystores: Current practices, comparisons, and open challenges*. 2301–2302. https://doi.org/10.1145/3269206.3274269

Lu, J., Lin, C., Wang, W., Li, C., & Xiao, X. (2015). Boosting the quality of approximate string matching by synonyms. *ACM Transactions on Database Systems*, *40*, 3. https://doi.org/10.1145/2818177

Lu, J., Liu, Z. H., Xu, P., & Zhang, C. (2016). *UDBMS: Road to unification for multi-model data management*. https://doi.org/10.48550/arXiv.1612.08050

Lu, J., Pan, B., Chen, J., Feng, Y., Hu, J., Peng, Y., & Chen, W. (2024). AgentLens: Visual Analysis for Agent Behaviors in LLM-based Autonomous Systems. *IEEE Transactions on Visualization and Computer Graphics*, *PP*, null. https://doi.org/10.48550/arXiv.2402.08995

Lu, Y.-T., Yang, S., Qian, C., Chen, G., Luo, Q., Wu, Y., Wang, H., Cong, X., Zhang, Z., Lin, Y., Liu, W., Wang, Y., Liu, Z., Liu, F., & Sun, M. (2024). Proactive Agent: Shifting LLM Agents from Reactive Responses to Active Assistance. *ArXiv*, *abs/2410.12361*, null. https://doi.org/10.48550/arXiv.2410.12361

Luo, J., Ouyang, C., Jing, Y., Fang, H., Xiao, Y., Zhang, Q., & Li, R. (2024). Application of LLM Techniques for Data Insights in DHP. *2024 IEEE 4th International Conference on Digital Twins and Parallel Intelligence (DTPI)*, *null*, 656–661. https://doi.org/10.1109/DTPI61353.2024.10778677

Lyu, B., Cong, X., Yu, H., Yang, P., Qin, Y., Ye, Y., Lu, Y.-T., Zhang, Z., Yan, Y., Lin, Y., Liu, Z., & Sun, M. (2023). GitAgent: Facilitating Autonomous Agent with GitHub by Tool Extension. *ArXiv*, *abs/2312.17294*, null. https://doi.org/10.48550/arXiv.2312.17294

Nooralahzadeh, F., Zhang, Y., Furst, J., & Stockinger, K. (2024). *Explainable multi-modal data exploration in natural language via LLM agent*. https://doi.org/10.48550/arXiv.2412.18428

Qi, D., & Wang, J. (2024). CleanAgent: Automating Data Standardization with LLM-based Agents. *ArXiv*, *abs/2403.08291*, null. https://doi.org/10.48550/arXiv.2403.08291

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y.-T., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Tian, R., Xie, R., Zhou, J., Gerstein, M. H., Li, D., Liu, Z., & Sun, M. (2023). ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. *ArXiv*, *abs/2307.16789*, null. https://doi.org/10.48550/arXiv.2307.16789

Roy, D., Zhang, X., Bhave, R., Bansal, C., Las-Casas, P., Fonseca, R., & Rajmohan, S. (2024). *Exploring LLM-based Agents for Root Cause Analysis*. https://doi.org/10.48550/arXiv.2403.04123

Sachdeva, S., & Vasava, J. (2024). *Plugging and playing with variety of data using multi-model database and polyglot persistence*. 1–8. https://doi.org/10.1109/ICEEICT61591.2024.10718481

Shi, Z., Gao, S., Chen, X., Feng, Y., Yan, L., Shi, H., Yin, D., Chen, Z., Verberne, S., & Ren, Z. (2024). Chain of Tools: Large Language Model is an Automatic Multi-tool Learner. *ArXiv*, *abs/2405.16533*, null. https://doi.org/10.48550/arXiv.2405.16533

Su, C., Wen, J., Kang, J., Wang, Y., Su, Y., Pan, H., Zhong, Z., & Shamim, H. M. (2024). Hybrid RAG-empowered multi-modal LLM for secure data management in internet of medical things: A diffusion-based contract approach. *IEEE Internet of Things Journal*, 1–1. https://doi.org/10.1109/JIOT.2024.3521425. [TLDR] A hybrid Retrieval-Augmented Generation-empowered medical MLLM framework for healthcare data management that enhances the output quality of MLLMs through hybrid RAG, which employs multi-modal metrics to filter various unimodal RAG results and incorporates these retrieval results as additional inputs to MLLMs.

Sun, W., Li, Z., Srinidhi, V., & Hai, R. (2025). *Database is All You Need: Serving LLMs with Relational Queries*. https://doi.org/10.48786/EDBT.2025.103

Sun, Z., Zhou, X., & Li, G. (2024). R-Bot: An LLM-based Query Rewrite System. *ArXiv*, *abs/2412.01661*, null. https://doi.org/10.48550/arXiv.2412.01661

Uotila, V., & Lu, J. (2021). *A formal category theoretical framework for multi-model data transformations* (Vol. 12921, pp. 14–28). https://doi.org/10.1007/978-3-030-93663-1_2. Comment: 15 pages, 4 figures, Heterogeneous Data Management, Polystores, and Analytics for Healthcare, VLDB Workshops, Poly 2021 and DMAH 2021.

Uotila, V., Lu, J., Gawlick, D., Liu, Z. H., Das, S., & Pogossiants, G. (n.d.). *Multi-model query processing meets category theory and functional programming*.

Uotila, V., Lu, J., Gawlick, D., Liu, Z. H., Das, S., & Pogossiants, G. (2021). MultiCategory: Multi-model query processing meets category theory and functional programming. *Proceedings of the VLDB Endowment*, *14*, 12. https://doi.org/10.14778/3476311.3476314. Comment: VLDB'21 Demonstration paper, 4 pages, 6 figures.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). *Attention is all you need*. https://doi.org/10.48550/arXiv.1706.03762. Comment: 15 pages, 5 figures.

Wang, Q., Bian, T., Yin, Y., Xu, T., Cheng, H., Meng, H. M., Zheng, Z., Chen, L., & Wu, B. (2023). Language Agents for Detecting Implicit Stereotypes in Text-to-image Models at Scale. *ArXiv*, *abs/2310.11778*, null. https://doi.org/10.48550/arXiv.2310.11778

Wang, S., Pan, S., & Cheung, A. (2024). QED: A Powerful Query Equivalence Decider for SQL. *Proc. VLDB Endow.*, *17*, 3602–3614. https://doi.org/10.14778/3681954.3682024

Wang, Y., Ma, H., & Wang, D. Z. (2024). No more optimization rules: LLM-enabled policy-based multi-modal query optimizer. *ArXiv*, *abs/2403.13597*, null. https://doi.org/10.48550/arXiv.2403.13597

Wang, Z., Liu, Z., Zhang, Y., Zhong, A., Fan, L., Wu, L., & Wen, Q. (2023). RCAgent: Cloud Root Cause Analysis by Autonomous Agents with Tool-Augmented Large Language Models. *ArXiv*, *abs/2310.16340*, null. https://doi.org/10.1145/3627673.3680016

Xie, T., Zhou, F., Cheng, Z., Shi, P., Weng, L., Liu, Y., Hua, T. J., Zhao, J., Liu, Q., Liu, C., Liu, L. Z., Xu, Y., Su, H., Shin, D., Xiong, C., & Yu, T. (2023). OpenAgents: An Open Platform for Language Agents in the Wild. *ArXiv*, *abs/2310.10634*, null. https://doi.org/10.48550/arXiv.2310.10634

Xu, Y., Ma, T., Tang, M., & Tian, W. (2014). A survey of privacy preserving data publishing using generalization and suppression. *Applied Mathematics & Information Sciences*, *8*, 3. https://doi.org/10.12785/amis/080321

Xue, S., Jiang, C., Shi, W., Cheng, F., Chen, K., Yang, H., Zhang, Z., He, J., Zhang, H., Wei, G., Zhao, W., Zhou, F., Qi, D., Yi, H., Liu, S., & Chen, F. (2024, January 3). *DB-GPT: Empowering Database Interactions with Private Large Language Models*. ArXiv.org. https://doi.org/10.48550/arXiv.2312.17449

Yan, Z., Uotila, V., & Lu, J. (2023). Join order selection with deep reinforcement learning:

Fundamentals, techniques, and challenges. *Proceedings of the VLDB Endowment*, *16*, 12. https://doi.org/10.14778/3611540.3611576

Yuan, G., & Lu, J. (2021). *MORTAL: A tool of automatically designing relational storage schemas for multi-model data through reinforcement learning*. https://doi.org/10.48550/arXiv.2109.00136. Comment: 6 pages, 4 figures, ER.

Yuan, G., Lu, J., & Su, P. (2021). *Quantum-inspired keyword search on multi-model databases* (Vol. 12682, pp. 585–602). https://doi.org/10.1007/978-3-030-73197-7_39. Comment: 16 pages, 5 figures, Dasfaa.

Yuan, G., Lu, J., & Yan, Z. (2022). Effective generation of relational schema from multi-model data with reinforcement learning: International conference on database systems for advanced applications. *2022 International Conference on Conceptual Modeling (ER 2022)*, 224–235. https://doi.org/10.1007/978-3-031-17995-2_16. Conference code: 27.

Yuan, G., Lu, J., Zhang, S., & Yan, Z. (2021). *Storing multi-model data in RDBMSs based on reinforcement learning*. 3608–3611. https://doi.org/10.1145/3459637.3482191. Comment: 4 pages, 4 figures, CIKM.

Yuan, S., Song, K., Chen, J., Tan, X., Li, D., & Yang, D. (2024). EvoAgent: Towards Automatic Multi-Agent Generation via Evolutionary Algorithms. *ArXiv*, *abs/2406.14228*, null. https://doi.org/10.48550/arXiv.2406.14228

Zhang, C., & Lu, J. (2021). Holistic evaluation in multi-model databases benchmarking. *Distributed and Parallel Databases*, *39*, 1. https://doi.org/10.1007/s10619-019-07279-6

Zhang, J., Zhang, H., Chakravarti, R., Hu, Y., Ng, P., Katsifodimos, A., Rangwala, H., Karypis, G., & Halevy, A. (2025). *CoddLLM: Empowering Large Language Models for Data Analytics*. ArXiv.org. https://arxiv.org/abs/2502.00329

Zhang, J., Zhao, C., Zhao, Y., Yu, Z., He, M., & Fan, J. (2024). MobileExperts: A Dynamic Tool-Enabled Agent Team in Mobile Devices. *ArXiv*, *abs/2407.03913*, null. https://doi.org/10.48550/arXiv.2407.03913

Zhang, Y., Cai, H., Chen, Y., Sun, R., & Zheng, J. (2023). *R EVERSE C HAIN : A G ENERIC -R ULE FOR LLM S TO M ASTER M ULTI -API P LANNING*. https://www.semanticscholar.org/paper/96dfa0404b0179f9521e8ba50145e9572c6443ef

Zheng, Y., Li, B., Lin, Z.-W., Luo, Y., Zhou, X., Lin, C., Su, J., Li, G., & Li, S. (2024). Revolutionizing Database Q&A with Large Language Models: Comprehensive Benchmark and Evaluation. *ArXiv*, *abs/2409.04475*, null. https://doi.org/10.48550/arXiv.2409.04475

Zhou, W., Jiang, Y., Li, L., Wu, J., Wang, T., Qiu, S., Zhang, J., Chen, J., Wu, R., Wang, S., Zhu, S., Chen, J., Zhang, W., Tang, X., Zhang, N., Chen, H., Cui, P., & Sachan, M. (2023). Agents: An Open-source Framework for Autonomous Language Agents. *ArXiv*, *abs/2309.07870*, null. https://doi.org/10.48550/arXiv.2309.07870

Zhou, X., Li, G., & Liu, Z. (2023, August 11). *LLM As DBA*. ArXiv.org. https://doi.org/10.48550/arXiv.2308.05481

Zhou, X., Li, G., Sun, Z., Liu, Z., Chen, W., Wu, J., Liu, J., Feng, R., & Zeng, G. (2023). D-Bot: Database Diagnosis System using Large Language Models. *Proc. VLDB Endow.*, *17*, 2514–2527. https://doi.org/10.48550/arXiv.2312.01454

Zhou, X., Li, G., Wu, J., Liu, J., Sun, Z., & Zhang, X. (2023). A Learned Query Rewrite System. *Proc. VLDB Endow.*, *16*, 4110–4113. https://doi.org/10.14778/3611540.3611633

Zhou, X., Sun, Z., & Li, G. (2024). DB-GPT: Large Language Model Meets Database. *Data Sci. Eng.*, *9*, 102–111. https://doi.org/10.1007/s41019-023-00235-6

Zhou, X., Zhao, X., & Li, G. (2024). *LLM-Enhanced Data Management*. https://doi.org/10.48550/arXiv.2402.02643

Živanović, Đ. (2023). *Reproducibility Report for ACM SIGMOD 2022 Paper: "WeTune: Automatic Discovery and Verification of Query Rewrite Rules."* https://www.semanticscholar.org/paper/b688797c14db584ab7f9f2fb043984184620d46d

Zykin, S. V., & Zykin, V. S. (2023). *Commutative transformations in multi-model databases*. 1–4. https://doi.org/10.1109/Dynamics60586.2023.10349549