LLMs and Databases: A Synergistic Approach to Data Utilization

Fatma Özcan, Yeounoh Chung, Yannis Chronis, Yu Gan, Yawen Wang
Carsten Binnig, Johannes Wehrstein, Gaurav Kakkar, Sami Abu-el-haija
Google Inc

Abstract

Large language models (LLMs) are not merely changing computing; they are igniting a revolution across industries, from healthcare to finance. Their prowess in tackling complex problems, demonstrated by breakthroughs in chatbots, translation, and code generation, is undeniable. A notable breakthrough in data management, driven by LLMs, is the advancement of natural language to SQL translation. This technology has fueled substantial progress, making database interactions more accessible and enabling the deployment of numerous real-world applications. Yet, even these powerful models falter with latency-sensitive regression problems, a critical need in database performance optimization. Inspired by the foundational principles of LLMs, we are developing pre-trained cardinality estimation and foundation database models, bridging this gap and unlocking the next generation of database optimization.

1 Introduction

Large Language Models (LLMs), a key part of GenAI, have captured the attention of many people, technical and non-technical alike. LLMs are bringing significant changes to the field of computing and this is not just an incremental step; it's a fundamental shift on how to approach computing problems and data. Notably, LLMs have demonstrated a powerful capability to tackle challenging problems across diverse domains, including data management. For instance, LLMs have greatly improved NL2SQL (converting natural language to SQL queries), leading to high accuracy solutions and making commercial applications more feasible. Beyond NL2SQL, LLMs have the potential to help with other difficult data management tasks like data integration, data discovery, and semantic understanding of tables and schemas.

LLMs, trained with all data available on the Internet, are very adept at question answering over unstructured context. Combining LLMs with databases allows us to analyze structured and unstructured data together in innovative ways [1–4]. These emerging systems explore how to augment RDBMS with pre-trained knowledge of LLMs to drive insights from a broader knowledge base.

For years, the seamless integration of structured and unstructured data has remained an elusive goal for enterprises. Traditional approaches, relying on ETL and predefined entity extraction, suffer from inherent limitations: rigid schemas and batch-oriented processing lead to significant delays. However, the advent of Large Language Models (LLMs) is ushering in a paradigm shift. LLMs, with their inherent ability to process unstructured data, combined with database capabilities, are unlocking unprecedented analytical power. Innovative systems, such as Lotus [1] and Palimzest [2], are leveraging LLM-powered operators to facilitate on-demand information extraction from unstructured sources. This eliminates the need for predefined entities and minimizes processing latency, unlocking new possibilities for AI-driven data analysis.

The synergy between databases and LLMs extends far beyond their respective handling of structured and unstructured data. Databases operate within a closed-world paradigm, relying solely on their stored information, while LLMs possess vast, pre-trained knowledge derived from the entire internet. Furthermore, their primary access mechanisms, SQL and natural language, respectively, offer distinct advantages. To realize the full potential of data analytics, we must explore more innovative ways to integrate these contrasting strengths.

LLMs have shown promise in addressing certain database challenges, including data wrangling[5] and semantic query equivalence checking[6]. These initial studies highlight the potential of LLMs for data management. However, LLMs currently struggle with regression tasks[7] and latency-sensitive problems, which are prevalent in database performance optimization. For example, cardinality estimation is one such latency sensitive regression problem, which can be solved more accurately using traditional ML models.

Early work on ML-based cardinality estimation focused on instance-based models, where a new model is trained for each data set and workload. However, one significant learning from LLMs's success is the importance of pre-trained models that can generalize to new tasks and data sets, with or without fine-tuning. Building on this observation, we build pre-trained cardinality estimation models. Inspired further by foundation models, we also explore the building blocks for database foundation models, and build data and query experts, which can be combined to solve many database performance problems.

In this paper, we first describe our findings and insights using LLMs to solve natural language to SQL. Then, we present our research on pre-trained cardinality estimation models, followed by foundation database models, which are inspired by foundation language models. Finally, we conclude with future research directions.

2 Natural Language to SQL with LLM

One particular domain that stands out to benefit from recent advancements in LLMs is Natural Language to SQL (NL2SQL). NL2SQL is a challenging task that requires translating ambiguous natural language questions into structured SQL queries over complex data schema [8]. Early approaches in NL2SQL relied on rule-based semantic parsing or seq-to-seq language models treating NL2SQL as a machine translation problem [9–11] with limited success. Recently, the latest LLMs with their superior language understanding and code generation capabilities have revolutionized the NL2SQL landscape. Since mid-2023, LLM-based approaches have dominated popular NL2SQL benchmarks, significantly outperforming earlier methods based on semantic parsing and machine translation [12–17].

LLMs lack the pre-trained knowledge of database schemas. Hence, each new database schema, with its different data model and semantics, pose a new challenge. This missing information needs to be provided as context to the LLM model. As such, we focus on schema linking and creating the right context for LLMs in our NL2SQL research.

2.1 Schema linking for NL2SQL

While LLMs excel at queries with less semantic ambiguity and simple schema, they can still struggle with more nuanced and ambiguous questions asked over complex data schema. Rather than directly translating natural language questions into SQL queries, LLMs can link relevant schema elements (e.g., tables, columns) to the natural language questions and generate SQL queries that align with users' intent. This so called schema linking is crucial for generating accurate and executable SQL queries. Many LLM based NL2SQL systems have focused on improving schema linking accuracy via careful relevant table and column retrieval [18, 19]. Recent studies [12, 20] showed that LLMs can generate accurate SQL queries given the entire database schema. The enhanced reasoning and "near-perfect" retrieval capabilities [21] of the latest LLMs, such as *gemini-1.5*, enable accurate schema linking during inference.

Table 1: Schema Selection Performance (TBR: Table Retrieval, CR: Column Retrieval) and Execution Accuracy on BIRD dev. Using the entire DB table schema without any relevant column selection yields comparable performance to using one of the state-of-the-art schema linking method; the ground truth TBR and CR results in the prompt boosts the SQL generation accuracy significantly.

	Filtered Schema [19]	All DB Schema [20]	Ground Truth Schema
Ex Acc (%)	64.08	64.80	72.43
TBR Recall (%)	97.69	100	100
TBR Precision(%)	89.72	34.29	100
CR Recall(%)	97.12	-	100
CR Precision(%)	69.43	-	100

Table 1 illustrates that the perfect relevant table and column retrievals (Ground Truth Schema) lead to accurate schema linking and give a significant boost to the generated SQL accuracy. It is important to note that such perfect table retrieval and column retrieval are infeasible in practice due to the question ambiguity as well as the semantic complexity of the underlying schema. High quality, but expensive, table and column selection can be done via multiple LLM calls (Filtered Schema [19]), and alternatively, one can pass the entire DB schema to the prompt, skipping relevant table and column retrievals entirely. In [20] we further study this alternative approach leveraging the extended context window of the latest LLMs, and evaluate the trade-offs of passing more contextual information for in-context learning.

2.2 In-context learning for NL2SQL

While providing table and column schema aids schema linking, it does not ensure accurate SQL generation. The model must comprehend the semantic correlation between the user's question and actual SQL query. This is further complicated by the absence of domain knowledge. For instance, consider a query such as "find patients with normal blood pressure". "Normal" is a domain-specific term that requires context and interpretation. Moreover, text search semantics frequently deviate from SQL semantics. A user might ask "find me all account holders in south bohemia," but it remains ambiguous how to translate "south bohemia" into a SQL predicate. Depending on the actual literal stored in the table, "South Bohemia", "south Bohemia", or "SOUTH BOHEMIA" are all possible options. Entity recognition, especially with proper nouns, introduces additional complexity.

One promising approach to address these challenges is in-context learning (ICL). ICL is a popular technique to guide the LLM to perform new tasks by providing a few-shot examples in the prompt. It has been demonstrated to improve the performance of LLMs in many challenging NLP tasks, such as math and reasoning [22]. We have found that in-context example selection is critical to NL2SQL accuracy. By selecting text2SQL example pairs similar to the user question, we can provide additional references about the schema, domain knowledge, and even help resolve literal issues.

Figure 1 illustrates the impact of example selection on NL2SQL generation. Our findings indicate that semantically similar examples within the prompt context lead to significant accuracy improvements. While even random examples from the same database schema provide some benefit, likely by exposing semantic relationships between question types and query structures, similarity-based example selection yields superior results. This improvement suggests that relevant examples convey valuable, schema-specific knowledge that enhances the capabilities of off-the-shelf LLMs. The highest accuracy is observed when the ground truth example is included, indicating that LLMs are adept at recognizing and leveraging correct examples but are less proficient at generating accurate queries without such guidance. However, it is important to note that even with ground truth examples, perfect accuracy is not achieved. This implies a potential for LLM overconfidence, where pre-training biases can lead to incorrect generations despite the availability of correct examples.

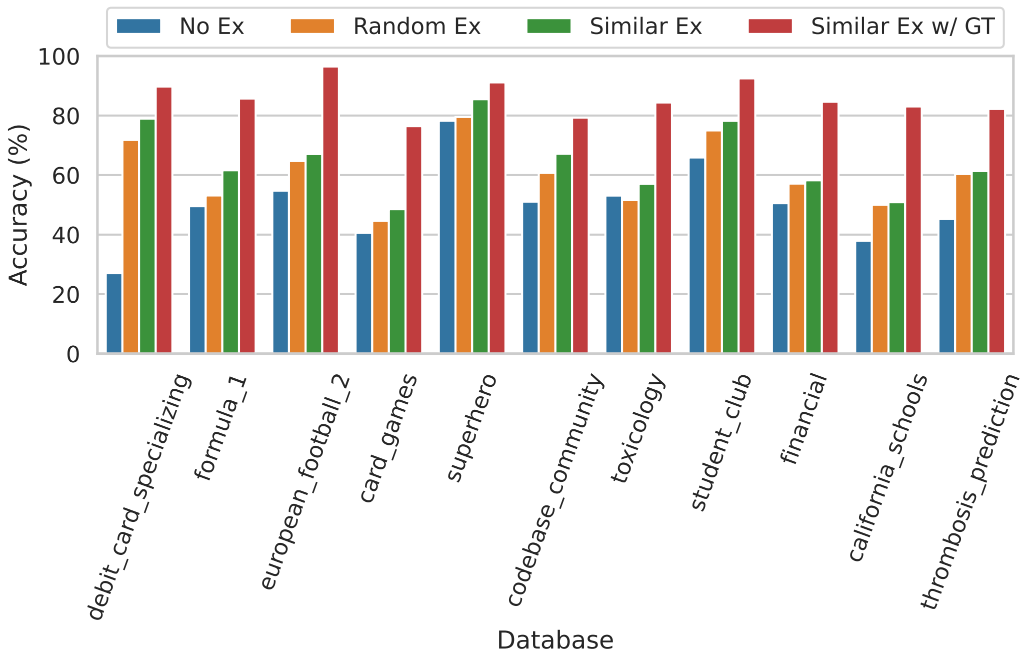


Figure 1: Generation accuracy with different ICL strategies on different databases from BIRD Bench. "No Ex": No examples are provided in the prompt. "Random Ex": 10 random examples from the same database schema are included in the prompt. "Similar Ex": The top 10 examples most similar to the question, based on question embedding similarity, are provided in the prompt. "Similar Ex w/ GT": The ground truth example is provided along with 9 other similar examples based on question embedding similarity.

2.3 Self-consistency for NL2SQL

Another technique that we see in the latest state-of-the-arts in the BIRD benchmark [13] leaderboard is self-consistency [14, 23]. The idea is to generate multiple output candidates via repeated runs and the most common (majority voting) or the most likely (specialized picker) output is selected. In [14], we use three different NL2SQL pipelines to generate a more diverse set of candidate pool, which result in a higher overall accuracy paired with a fine-tuned picker LLM model. While the strong results demonstrate how the stochastic nature of LLM generation can be exploited to address more ambiguous user queries and tasks, the ambiguous user queries and complex data schema still remain as unsolved challenges. Self-consistency also raises concerns about the increased number of LLM calls and the cost and latency of NL2SQL output generation. Therefore, improving the efficiency of these multi-step LLM interactions is a critical research priority.

2.4 Long context for NL2SQL

Much of LLM-based NL2SQL research and solutions assumed a limited context size (typically smaller than 8k tokens, which is just enough hold a few select table schema) and also degrading performance over increasing amount of contextual information [24]. As such, many NL2SQL pipelines focused on retrieving a handful of top-K table schema along with 3-5 examples or chain-of-thought (CoT) demonstrations [15, 25]. Alternatively, in a recent study [20] we explored the potential of leveraging the long context LLMs, like *gemini-1.5* with 2-million tokens context limit, with lots of extra contextual information(e.g., entire DB table schema, hundreds of examples for ICL). In particular, we explored including entire database schemas, hints, tens of example queries, sample column values, as well as entire distinct values of columns. We study in detail the utility and cost of various long context-based strategies in [20]. The results indicate that the long context model exhibits a very strong retrieval capability over the extended context window for NL2SQL and is robust to extra irrelevant information in the prompt. This means that NL2SQL generation does not need to be impaired by the imperfect performance of retrieval services – we can retrieve more to ensure higher recall at the cost of lower precision.

3 Using Traditional ML for Latency-sensitive Database Problems

Recently, machine learning models have been used effectively in many database problems[26], showing a lot of promise. These early works show that there are many database problems where ML can benefit database systems, including run-time prediction [27], and query optimization [28]. All such internal database optimization tasks have in common that they have strict latency requirements as they happen at query time and they are data dependent. Cardinality estimation (CE), as one of the key building block in modern databases for optimizing performance, has been studied extensively as a target for many ML based approaches [29, 30].

Traditional CE techniques used in modern database systems have well-known limitations, as they make simplistic data modeling assumptions, such as data uniformity and independence of columns in the tables. Recent research has explored learned CE, employing machine learning for improved accuracy. Learned CE model following an instance-based approach have been shown to improve upon the state-of-art techniques used by database systems today [29]. Instance-based models are trained and evaluated on a specific database instance, requiring retraining for each new dataset. This category encompasses workload-driven models that are trained using a representative set of queries executed on the target database, incurring significant overhead and data-driven models that learn the data distribution within the database, avoiding query execution but still requiring retraining upon data updates. Despite their better accuracy, such learned CE models have not been adapted in practice, due to their high training overheads, among other reasons. As a result, pre-trained CE models, which are trained on a corpus of diverse datasets on transferable features, are highly desirable to alleviate the training costs and increase adaption.

Zero-shot models: This emerging paradigm aims for generalizability across diverse datasets without retraining. Pre-trained on a variety of databases, these models leverage transferable features to adapt to unseen data and are more robust to data modifications. We developed such pre-trained CE estimation models, which we describe next.

3.1 Cardinality Estimation Model

Recognizing the inherent graph representation of SQL queries, we explore the following two model architectures for the cardinality estimation task: Graph Neural Network (GNN) and Graph Transformer.

The design of the GNN model is based on the GNN cost model from [31]. Each node's features are passed through a separate Multi-Layer Perceptron (MLP) based on the node type. The GNN then mirrors the query execution order to propagate information in a topological order from leaf to the root node. Finally a learned graph-level embedding from the root node is used to predict query cardinality.

In addition to GNN, we build a Graph Transformer model which extends the attention mechanism to graph-structured data. Inspired by Graphormer [32], we make several modifications on top of the traditional transformer architecture to better capture graph complexity. These changes include heterogeneous input embedding to accommodate node-type specific feature types, shortest-path spatial encoding as an attention bias to capture structural relationships between nodes, directional causal masking to enforce topological ordering, and a virtual node connected to all other nodes used for a comprehensive graph-level representation for the final cardinality prediction.

3.2 CardBench

To train and test the pre-trained CE models, we need a new benchmark. Existing benchmarks for cardinality estimation models often rely on limited datasets, like the Join Order Benchmark (JOB) with its single IMDb dataset. While JOB offers realistic data distributions, a single dataset is not sufficient for zero-shot training. Recent efforts have introduced new benchmarks with additional datasets, but they still fall short in representing the wide range of real-world data required for training and testing pre-trained models.

To close this gap, we have developed and open-sourced CardBench[33][1], a benchmark containing thousands of queries on 20 distinct databases, and scripts to compute data summary statistics and generate queries. CardBench contains 20 distinct datasets and thousands of queries of different complexities. Our goal is to foster further research in the area of learned CE, with a focus on enabling the training and testing of pre-trained zero-shot CE models.

CardBench datasets were chosen to be diverse, complex and cover a wide range of data distributions to stress CE models. All the datasets are publicly available or are based on publicly available datasets (we list the sources in the CardBench github repository). In comparison to existing CE benchmarks, CardBench includes a much higher number of datasets and training data (i.e., queries and cardinalities) to experiment with and compare different types of learned CE approaches. Our training data are SQL queries represented as graphs annotated with dataset statistics and the query cardinality as the label (Table 2, right). As shown in Figure 2, we represent the SQL query as a graph. The graph nodes are annotated with relevant the features described in [33]. The query graph has table nodes (in blue), column nodes (in green), operator nodes (in red), predicates (in yellow) and correlations across column of the same table (white circles with red outline).

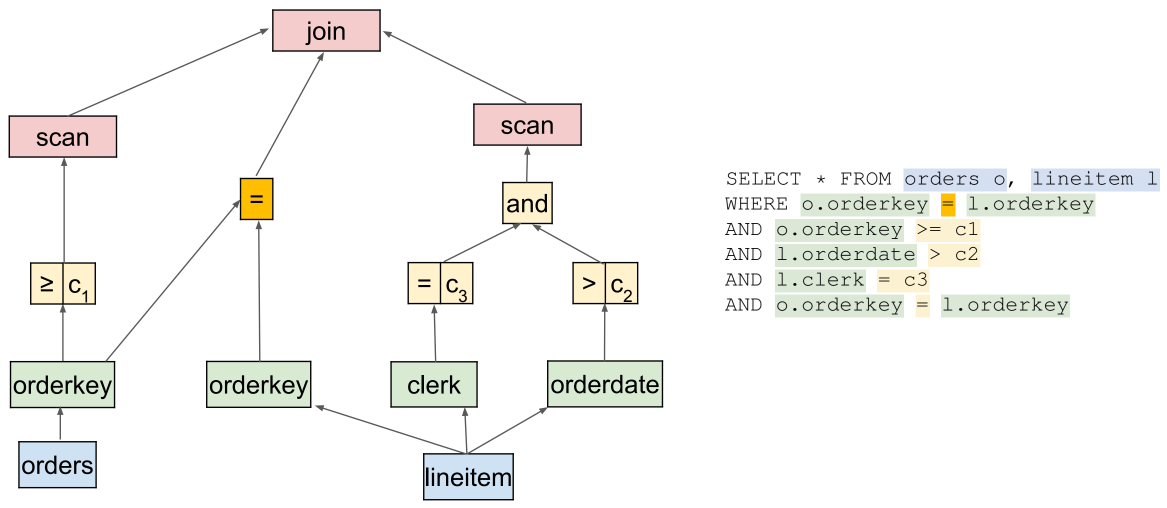
CardBench includes three sets of training data, Single Table, Binary Join and Multi-Join. Single Table queries filter a single table using 1 to 4 filter predicates. Binary Join queries join two tables that are also filtered with 1 to 3 filter predicates per table. Multi Join queries perform 0 to 8 joins and also apply 0 to 2 filter predicates per table. The three training dataset configurations can demonstrate the challenges of more complex queries on CE models. Next, we present our experiments using Cardbench, with the two pre-trained CE estimation models that we trained.

3.3 Experiments with CardBench

In this section, we present our empirical findings, evaluating the accuracy of various model configurations for cardinality estimation. We focus on the *Binary Join* CardBench dataset that contain queries with 0 to 1 joins, and 1-3 predicates on each table, as often the majority of queries fall in this category (in the recently released dataset by Amazon [34] more than 90% of the queries have 0 to 1 joins).

We use the q-error, which is a common metric in database systems research for evaluating the accuracy of CE models[30]. It calculates the relative deviation of the predicted cardinality from the true cardinality for a given query. The q-error for a single query is calculated as follows:

[1]https://github.com/google-research/google-research/CardBench_zero_shot_cardinality_training



```
SELECT * FROM orders o, lineitem l
WHERE o.orderkey = l.orderkey
AND o.orderkey >= c1
AND l.orderdate > c2
AND l.clerk = c3
AND o.orderkey = l.orderkey
```

Figure 2: Foundation database models can generalize across tasks and datasets which is very different from current approaches where we need to train multiple models either per task (Zero-shot) or per dataset (Multi-Task) or even per combination (Instance-specific).

$$\text{q-error} = \max\left(\frac{\hat{y}}{y}, \frac{y}{\hat{y}}\right) \in [1, +\infty), \tag{1}$$

where y and $\hat{y}$ represent true and predicted cardinalities, respectively.

We assess CE performance using three distinct ML model configurations: instance-based, zero-shot, and fine-tuned.

Instance-based Models: In this configuration, we train and evaluate individual models using a single dataset. For each dataset, queries are randomly partitioned into training and validation sets with 85:15 train-validation split. An additional 500 queries are reserved as a standalone test set to evaluate final model accuracy.

Zero-shot Models: This configuration investigates the generalizability of cardinality prediction models to unseen datasets, i.e., data from another dataset apart from the training data. We train and validate the model on queries from 19 datasets, maintaining the 85:15 train-validation split. The model is then tested on the 20th dataset (not used in training). This test set contains the same queries used in the instance-based setting to ensure a fair comparison.

Fine-tuned Models: This configuration investigates whether fine-tuning a pre-trained zero-shot model improves accuracy and sample efficiency. We fine-tune the model initially trained on 19 datasets using the 20th dataset. The fine-tuned model's accuracy is then evaluated using the same 500 holdout queries from this 20th dataset.

The average training and inference times under each configuration are 2. We observe that training the zero-shot model over extended epochs can often result in lower accuracy due to over-fitting to the training set. Therefore, we cap the maximum number of training epochs for zero-shot training to 20, and set a maximum training epochs of 100 for the other two configurations. The graph transformer model size is 33.6MB, with 8.4M parameters in total, the GNN model size is 7.5MB, with 1.88M parameters in total.

We compare the learned cardinality approaches against a the cardinality estimation of PostgreSQL[2].

[2]https://www.postgresql.org/

Model	Training Time	Inference Time
Instance-Based (GNN)	1.3hr	
Zero-Shot (GNN)	1.5hr	35ms
Fine-Tuned (GNN)	11min	
Instance-Based (Transformer)	11.1hr	
Zero-Shot (Transformer)	11.1hr	97ms
Fine-Tuned (Transformer)	2.1hr	

Table 2: Training and Inference times for difference model types. The inference time of the zero-shot, fine-tuned and instance-based configurations of the same model type are the same since they share the same architecture and model size. For training a V100 GPU was used.

To get the estimates we load the CardBench data in PostgreSQL and use the *explain* command, similar to [31]. We call this baseline *Postgres*.

Cardinality prediction for queries with binary joins is challenging due to cross-table distribution and multi-column correlation, which are difficult to model, significantly impact join cardinality. Table 3 shows the P50, P75, P90 and P95 q-errors for queries with binary joins. The baseline algorithm exhibits significant inaccuracy in binary join query cardinality estimation, with a median q-error of 55.54 and P95 q-error of 4.4×10^6, respectively.

Learning-based models significantly outperform the baseline though, particularly at the 95th percentile. Instance-based models, which learn cross-table distributions within a dataset, achieve low median q-errors of 1.16 (GNN) and 1.20 (transformer). While their P95 q-errors are higher (37.55 for GNN, 43.96 for transformer), the results suggest that estimating join cardinality via learned distributions is feasible. However, out-of-distribution cardinality estimation for binary joins proves more challenging, as zero-shot models experience a dramatic increase in q-errors (up to 20x at the median, 5300x at P95). However, even a small amount of data used for fine-tuning improves the accuracy of pre-trained models considerably. Hence, we conclude that pre-trained models with fine-tuning is a viable approach to cardinality estimation.

Model	Q^{50}_{err}	Q^{75}_{err}	Q^{90}_{err}	Q^{95}_{err}
Postgres	5.29	21.03	38557.10	411511.45
GNN Instance	1.17	1.82	10.92	37.55
GNN Zeroshot	22.66	102.12	3430.23	16271.84
GNN Finetune	**1.32**	**2.45**	**13.62**	**109.77**
Transformer Instance	1.20	1.99	11.93	43.96
Transformer Zeroshot	24.88	264.06	6000.99	228499.79
Transformer Finetune	79.52	1.57	4.19	47.14

Table 3: Average q-error percentiles for Binary Join Queries, the best accuracy are in bold.

4 Foundation DB Models

As discussed in the previous section, the state-of-the-art is one-off models that need to be trained individually per task and even per dataset, which causes extremely high training overheads. Hence, a new learning paradigm is needed that moves away from such one-off models towards generalizable

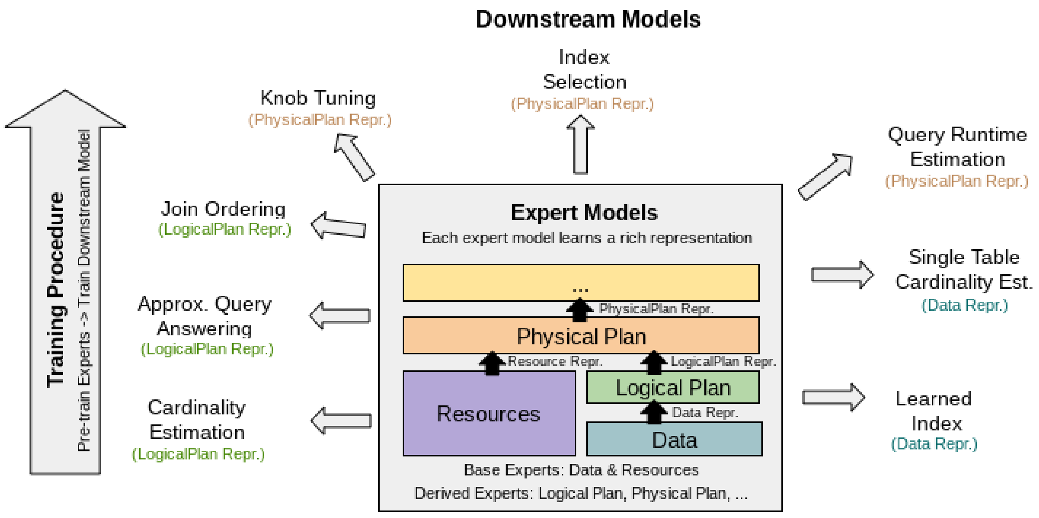


Figure 3: Foundation database models build on a mixture pre-trained experts where some experts learn representations independently (e.g., the data expert) and experts that enrich representations (e.g., the logical plan and physical plan expert). Shallow downstream models take these representation as input and solve a particular database task.

models that can be used with only minimal overhead for an unseen dataset on a wide spectrum of tasks. Recent advances in LLMs have proven that generalizable (i.e., foundation) models for text, coupled with fine-tuning, can be used to solve a wide variety of NLP problems [35]. In this section, inspired by LLMs, we propose a new direction which we call *foundation database models*[36], which are pre-trained in both task-agnostic and dataset-agnostic manner, making it possible to use the model with low overhead to solve a wide spectrum of downstream tasks on unseen datasets. In [36], we propose a vision for foundation database models, and describe our initial prototype and findings.

We make the observation that a set of foundational experts – data, resource, logical and physical query plan – could be used to solve many database problems, including cardinality estimation, index selection, run-time estimation, materialized view selection, partitioning and clustering key selection, etc. (see . Figure 3).

The insight to realize such a foundation model for database problems is that we use a mixture of pre-trained expert models as shown in Figure 3, which enable generalizability along the two dimensions: (1) To generalize across datasets, we provide a data expert that learns to summarize databases into learned embeddings which represent the characteristics of a given dataset. (2)Foundation database models come with additional pre-trained experts that enrich the data expert to solve a wide range of downstream tasks with only low overheads.

We pre-train expert models in a modular manner, each with a set of inputs that could be raw data, engineered features or learned embedding vectors from other expert models. This is where we differ from foundation language models that are trained end-to-end.

For our initial validation, we developed three expert models: the data expert, a logical plan expert, and a physical plan expert. The data expert does not use database specific information such as particular constants in the data or attribute and table names to learn the embedding, but it uses a new transferable-encoding of databases which allows the data expert to learn general data characteristics of a database such as data distributions and correlations across columns. The logical plan expert takes embeddings produced by the data expert as input, in addition to other features, and learns how various query operators modify their input data. The logical plan representation as such enriches the data representation and can be used to solve tasks such as cardinality estimation or even approximate query answering where this information is needed. Finally, the physical plan expert also learns how logical operators are executed using various implementations.

In [36], we report our initial findings for cardinality and run-time estimation. For cardinality estimation at P50, our model achieves a q-error of 2.12 without fine-tuning, and 1.69 with fine-tuning compared to the q-error of 1.98 for Postgres, for a workload of queries with up to 2 joins and 5 filters. Our model does even better at the tail (P95), with q-error of 92.92 without fine-tuning and 26.08 with fine-tuning compared to the q-error of 294.15 for Postgres. Once again, we observe that fine-tuning is essential to achieve high accuracy.

We believe our initial results are quite promising, and more work is needed to investigate more experts, and a wider spectrum of downstream database problems. We also argue that different pre-training tasks need to be explored to identify the most effective way of training these models.

5 Concluding Remarks

In this paper, we described our experiences and insights using LLMs for natural language to SQL generation. While significant progress has been possible with the use of LLMs, there is still a knowledge gap in understanding enterprise schemas and data semantics for high accuracy.

We also argued using traditional ML models for solving latency sensitive regression problems, such as cardinality estimation. We presented two approaches, both inspired by LLMs. The first one pre-trains a model using a number of diverse datasets, and using database agnostic input features, and is able to generalize to an unseen dataset with fine-tuning, using small amounts of training data. The second one takes this one step further and argues for foundational database models that are generalizable both across datasets as well as across multiple database problems.

We posit that both foundational database models and LLMs hold significant promise for future research and the development of novel data management solutions.

Acknowledgement

We would like to thank to our many collaborators, who helped with various parts of the work reported in this paper; including Shobha Vasudevan, Chelsea Lin, Jiaxun Wu, Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Sercan Arik, Brenton Milne, Reza Sherkat, Per Jacobsson, Aleksandras Surna, and Mahadevan Sankar.

References

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

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

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

[4] M. Urban and C. Binnig, "Caesura: Language models as multi-modal query planners," in *Conference on Innovative Data Systems Research*, 2024.

[5] A. Narayan, I. Chami, L. Orr, S. Arora, and C. Ré, "Can foundation models wrangle your data?" *arXiv preprint arXiv:2205.09911*, 2022.

[6] B. Haynes, R. Alotaibi, A. Pavlenko, J. Leeka, A. Jindal, and Y. Tian, "Geqo: Ml-accelerated semantic equivalence detection," *Proceedings of the ACM on Management of Data*, vol. 1, no. 4, pp. 1–25, 2023.

[7] M. Lukasik, H. Narasimhan, S. Kumar, F. Yu, and A. Menon, "Regression aware inference with llms," in *EMNLP Findings*, 2024.

[8] A. Floratou, F. Psallidas, F. Zhao, S. Deep, G. Hagleither, W. Tan, J. Cahoon, R. Alotaibi, J. Henkel, A. Singla, A. V. Grootel, B. Chow, K. Deng, K. Lin, M. Campos, K. V. Emani, V. Pandit, V. Shnayder, W. Wang, and C. Curino, "Nl2sql is a solved problem... not!" in *Conference on Innovative Data Systems Research*, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:266729311

[9] A. Wong, L. Pham, Y. Lee, S. Chan, R. Sadaya, Y. Khmelevsky, M. Clement, F. W. Y. Cheng, J. Mahony, and M. Ferri, "Translating natural language queries to sql using the t5 model," in *2024 IEEE International Systems Conference (SysCon)*. IEEE, 2024, pp. 1–7.

[10] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "Tabert: Pretraining for joint understanding of textual and tabular data," *arXiv preprint arXiv:2005.08314*, 2020.

[11] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[12] K. Maamari, F. Abubaker, D. Jaroslawicz, and A. Mhedhbi, "The death of schema linking? text-to-sql in the age of well-reasoned language models," *arXiv preprint arXiv:2408.07702*, 2024.

[13] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Geng, N. Huo *et al.*, "Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[14] M. Pourreza, H. Li, R. Sun, Y. Chung, S. Talaei, G. T. Kakkar, Y. Gan, A. Saberi, F. Özcan, and S. Ö. Arık, "Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql," *arXiv preprint arXiv:2410.01943*, 2024.

[15] M. Pourreza and D. Rafiei, "Din-sql: Decomposed in-context learning of text-to-sql with self-correction," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[16] H. Li, J. Zhang, C. Li, and H. Chen, "Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 11, 2023, pp. 13 067–13 075.

[17] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018.

[18] H. A. Caferoğlu and Ö. Ulusoy, "E-sql: Direct schema linking via question enrichment in text-to-sql," *arXiv preprint arXiv:2409.16751*, 2024.

[19] S. Talaei, M. Pourreza, Y.-C. Chang, A. Mirhoseini, and A. Saberi, "Chess: Contextual harnessing for efficient sql synthesis," *arXiv preprint arXiv:2405.16755*, 2024.

[20] Y. Chung, G. T. Kakkar, Y. Gan, B. Milne, and F. Ozcan, "Is long context all you need? leveraging llm's extended context for nl2sql," *arXiv preprint arXiv:2501.12372*, 2025.

[21] M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, T. Lillicrap, J.-b. Alayrac, R. Soricut, A. Lazaridou, O. Firat, J. Schrittwieser *et al.*, "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context," *arXiv preprint arXiv:2403.05530*, 2024.

[22] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, T. Liu, B. Chang, X. Sun, L. Li, and Z. Sui, "A survey on in-context learning," 2024. [Online]. Available: https://arxiv.org/abs/2301.00234

[23] Y. Gao, Y. Liu, X. Li, X. Shi, Y. Zhu, Y. Wang, S. Li, W. Li, Y. Hong, Z. Luo, J. Gao, L. Mou, and Y. Li, "Xiyan-sql: A multi-generator ensemble framework for text-to-sql," *arXiv preprint arXiv:2411.08599*, 2024. [Online]. Available: https://arxiv.org/abs/2411.08599

[24] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024. [Online]. Available: https://aclanthology.org/2024.tacl-1.9

[25] L. Nan, Y. Zhao, W. Zou, N. Ri, J. Tae, E. Zhang, A. Cohan, and D. Radev, "Enhancing few-shot text-to-sql capabilities of large language models: A study on prompt design strategies," *arXiv preprint arXiv:2305.12586*, 2023.

[26] G. Li, X. Zhou, and L. Cao, "Machine learning for databases," in *Proceedings of the First International Conference on AI-ML Systems*, 2021, pp. 1–2.

[27] T. Kraska, U. F. Minhas, T. Neumann, O. Papaemmanouil, J. M. Patel, C. Ré, and M. Stonebraker, "Ml-in-databases: Assessment and prognosis." *IEEE Data Eng. Bull.*, vol. 44, no. 1, pp. 3–10, 2021.

[28] C. Anneser, N. Tatbul, D. Cohen, Z. Xu, P. Pandian, N. Laptev, and R. Marcus, "Autosteer: Learned query optimization for any sql database," *Proceedings of the VLDB Endowment*, vol. 16, no. 12, pp. 3515–3527, 2023.

[29] P. Negi, Z. Wu, A. Kipf, N. Tatbul, R. Marcus, S. Madden, T. Kraska, and M. Alizadeh, "Robust query driven cardinality estimation under changing workloads," *Proceedings of the VLDB Endowment*, vol. 16, no. 6, pp. 1520–1533, 2023.

[30] X. Wang, C. Qu, W. Wu, J. Wang, and Q. Zhou, "Are we ready for learned cardinality estimation?" *arXiv preprint arXiv:2012.06743*, 2020.

[31] B. Hilprecht and C. Binnig, "One model to rule them all: Towards zero-shot learning for databases," in *12th Conference on Innovative Data Systems Research, CIDR 2022, Chaminade, CA, USA, January 9-12, 2022*. www.cidrdb.org, 2022. [Online]. Available: https://www.cidrdb.org/cidr2022/papers/p16-hilprecht.pdf

[32] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu, "Do transformers really perform badly for graph representation?" *Advances in neural information processing systems*, vol. 34, pp. 28 877–28 888, 2021.

[33] Y. Chronis, Y. Wang, Y. Gan, S. Abu-El-Haija, C. Lin, C. Binnig, and F. Özcan, "Cardbench: A benchmark for learned cardinality estimation in relational databases," *arXiv preprint arXiv:2408.16170*, 2024.

[34] A. van Renen, D. Horn, P. Pfeil, K. E. Vaidya, W. Dong, M. Narayanaswamy, Z. Liu, G. Saxena, A. Kipf, and T. Kraska, "Why tpc is not enough: An analysis of the amazon redshift fleet," in *VLDB 2024*, 2024. [Online]. Available: https://www.amazon.science/publications/why-tpc-is-not-enough-an-analysis-of-the-amazon-redshift-fleet

[35] A. Narayan, I. Chami, L. J. Orr, and C. Ré, "Can foundation models wrangle your data?" *Proc. VLDB Endow.*, vol. 16, no. 4, pp. 738–746, 2022.

[36] J. Wehrstein, C. Binnig, F. Ozcan, S. Vasudevan, Y. Gan, and Y. Wang, "Towards foundation database models," in *Conference on Innovative Data Systems Research*, 2025. [Online]. Available: https://vldb.org/cidrdb/2025/towards-foundation-database-models.html