MoMQ: Mixture-of-Experts Enhances Multi-Dialect Query Generation across Relational and Non-Relational Databases

Anonymous ACL submission

Abstract

The improvement in translating natural language to structured query language (SQL) can be attributed to the advancements in large language models (LLMs). Open-source LLMs, tailored for specific database dialects such as MySQL, have shown great performance. However, cloud service providers are looking for a unified database manager service (e.g., Cosmos DB from Azure, Amazon Aurora from AWS, Lindorm from AlibabaCloud) that can support multiple dialects. This requirement has led to the concept of multi-dialect query generation, which presents challenges to LLMs. These challenges include syntactic differences among dialects and imbalanced data distributions across them. To address these issues, we propose MoMQ, a novel Mixture-of-Experts-based multi-dialect query generation framework across both relational and non-relational databases. MoMQ incorporates dialect-specific expert groups to capture syntax features of individual dialects while minimizing cross-dialect interference in query generation. Additionally, we propose a multi-level routing mechanism enhanced by Dialect Router Loss (DRL) and a shared expert group architecture, facilitating common knowledge transfer from high-resource dialects to low-resource ones. Furthermore, we have developed a high-quality multi-dialect query generation benchmark that covers relational and non-relational databases such as MySQL, PostgreSQL, Cypher for Neo4j, and nGQL for NebulaGraph. Extensive experiments have shown that MoMQ performs effectively and robustly, even in resource-imbalanced scenarios.

1 Introduction

The ability to convert natural language into structured query language (SQL) has made it much easier to interact with relational database management systems. In recent years, the use of large language models (LLMs) has significantly improved SQL generation tasks (Li et al., 2024b; Gao et al., 2024). This shift has enhanced the quality of generated queries, moving away from encoder-decoder-based approaches (Li et al., 2023b; Fu et al., 2023; Li et al., 2023a) to those driven by LLMs. Open-source LLMs (Li et al., 2024b; Bai et al., 2023; Rozière et al., 2024) that have been fine-tuned through supervision have become the primary method due to their lower data privacy risks and cost compared to closed-source LLMs (Achiam et al., 2023; Bai et al., 2023; Anil et al., 2023). These LLMs are typically designed to work best with a specific database dialect, like MySQL.

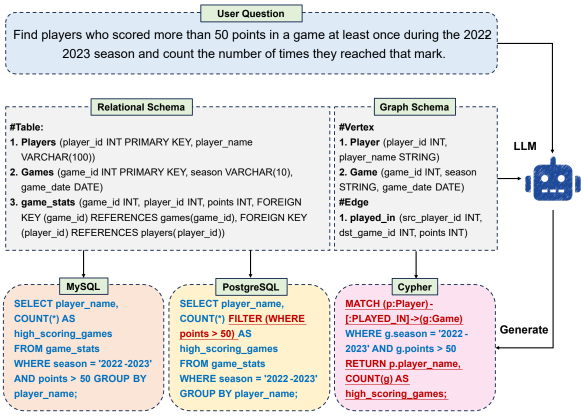

Figure 1: Database queries exhibit notable syntax variations. For instance, PostgreSQL offers a distinctive FILTER clause compared to MySQL, and Cypher's MATCH statement contrasts with the SELECT queries used in SQL databases like MySQL and PostgreSQL.

However, for general database management services in cloud computing, LLMs that support most dialects are needed. Therefore, SQL generation LLMs should not only cover major dialects like MySQL and PostgreSQL but also non-relational graph databases such as Neo4j (Neo4j, 2012) and NebulaGraph (Wu et al., 2022). Figure 1 illustrates the similarities and key distinctions in query syntax across different databases. The primary variations within relational databases stem from the use of different keywords, while the contrast between relational and non-relational databases is more fundamental, reflecting differences in query logic. These differences are collectively referred to as the database dialect issue(Zmigrod et al., 2024). Past research (Zhang and Yang, 2022; Standley et al., 2020; Crawshaw, 2020) has demonstrated that multi-task learning enables models to integrate knowledge from diverse sources, leading to improved performance. However, directly fine-tuning dense LLMs on multi-dialect data encounters several challenges: (1) Syntax variations across relational database dialects, such as the use of distinct keywords in MySQL and PostgreSQL, and more pronounced differences between relational and non-relational databases, such as "SELECT" versus "MATCH", may hinder accurate query generation. (2) The cost of annotating natural language to database query language is significant, and data for most dialects is limited. Importantly, the similarities in natural language questions and database schemas across dialects represent transferable knowledge that is not fully utilized.

To tackle these challenges, we propose MoMQ, a Mixture-of-Experts (MoE)-based multi-dialect query generation framework that unifies query generation for both relational and non-relational databases. Given the high cost of pre-training MoE structures from scratch, we build our MoE framework on top of a dense model. Unlike MoE Upcycling (Komatsuzaki et al., 2022), we incorporate multiple Low-Rank Adaptation (LoRA) modules (Hu et al., 2021) to design a detailed MoE structure, as in Wu et al. (2024); Li et al. (2024a); Feng et al. (2024), while freezing the original model to retain pre-trained knowledge. We introduce specialized dialect expert groups to isolate dialect-specific knowledge, minimizing interference during query generation. To tackle the imbalance in multi-dialect data, we introduce a shared expert group visible to all dialects, which facilitates knowledge transfer from high-resource to low-resource dialects. Furthermore, we propose a multi-level routing mechanism that includes a dialect router and an expert router. The dialect router directs dialect-specific tokens to their respective expert groups while distributing common tokens across all groups with the assistance of the Dialect Router Loss, enabling token-level knowledge transfer. The expert router activates the top-k experts in each dialect expert group for input tokens. To prevent routing collapse (Shazeer et al., 2017), we employ an Expert Balance Loss to mitigate load imbalance.

We construct a comprehensive evaluation benchmark covering both relational and non-relational graph databases, such as MySQL, PostgreSQL, Cypher for Neo4j, and nGQL for NebulaGraph. Experimental results show that MoMQ outperforms existing methods, achieving 3-5% improvements in full-data scenarios and 4-6% in data-imbalanced settings. Ablation studies further confirm the framework's robustness across different expert configurations and validate its dialect transfer capabilities through expert weight analysis.

Overall, we summarize our contributions as follows:

- We introduce MoMQ, a MoE-based framework for unified query generation across relational and non-relational databases, enhancing open-source LLMs' multi-dialect capabilities.
- We propose a novel dialect-specialized architecture and a multi-level routing mechanism that effectively mitigate potential interference in multi-dialect generation and alleviate data imbalance through enhanced knowledge transfer.
- A high-quality multi-dialect generation benchmark has been constructed, providing an evaluation standard for related research.
- We also conducte empirical experiments and analysis to validate the effectiveness and robustness of MoMQ under different settings.

2 Related Work

MoMQ is a multi-dialect query generation framework that is highly related with the tuning-based text-to-SQL task and Mixture-of-Experts.

Tuning-Based Text-to-SQL. Before the era of large-scale models, the text-to-SQL task typically employs an encoder-decoder-based architecture. Research in this domain primarily focuses on enhancing the encoder's ability to understand the question and schema (Li et al., 2023b; Fu et al., 2023; Li et al., 2023a). With the advent of large language models (LLMs), the text-to-SQL task has gradually shifted towards LLM-based approaches(Achiam et al., 2023; Bai et al., 2023; Anil et al., 2023; Li et al., 2024b,d; Pourreza and Rafiei, 2023). Supervised fine-tuning of open-source large models such as LLama (Touvron et al., 2023), StartCoder (Li et al., 2023c), and Qwen (Bai et al., 2023)

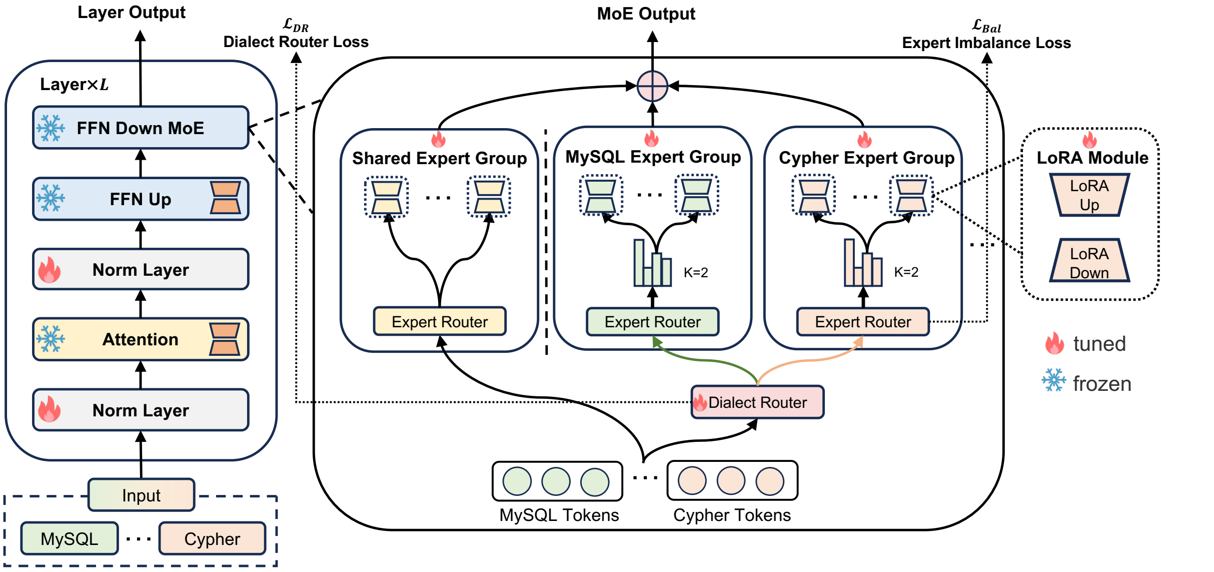

Figure 2: The overall structure of MoMQ. The original feed-forward network (FFN) is transformed into a MoE structure, which consists of Shared Expert Group, Dialect Expert Group, and Multi-Level Routing Mechanism. The pre-trained weights are frozen and LoRA modules inserted into Attention and FFN are fine-tuned for rapid adaptation to multi-dialect query generation. The normalization layer is unfrozen due to its observed improvement. The Dialect Router Loss and Expert Balance Loss are added to the training objectives to adjust multi-dialect routing and mitigate routing collapse respectively.

has significantly improved both natural language understanding and SQL query generation capabilities.

Mixture-of-Experts. In recent years, Mixture-of-Experts (MoE) has emerged as an effective structure for reducing inference computational costs and enhancing multi-task learning capabilities in scenarios where model parameters are continuously scaled up (Dai et al., 2024; Jiang et al., 2024; Zhao et al., 2024; Xue et al., 2024; Zoph, 2022; Fedus et al., 2022; Lepikhin et al., 2020; Du et al., 2022). A gate unit is then utilized for expert activation. However, training MoE models from scratch or converting dense models into MoE through upcycling requires substantial pre-training costs (Komatsuzaki et al., 2022; Lin et al., 2024; Xue et al., 2024). Recent studies have proposed the construction of MoE based on the Low-Rank Adaptation (LoRA) module (Wu et al., 2024; Li et al., 2024a; Feng et al., 2024). This approach not only mitigates catastrophic forgetting of pre-trained knowledge in dense models but also enhances the multi-task learning capability.

3 Methodology

Task Definition. Given $\mathcal{N}$ natural language questions $\mathcal{Q} = \{q_1, q_2 \cdots, q_\mathcal{N}\}$, a set of target query language dialects $\mathcal{L} = \{l_1, l_2 \cdots, l_\mathcal{M}\}$ and a database schema $\mathcal{D} = \{\mathcal{C}, \mathcal{T}\}$, where $\mathcal{C}$ and $\mathcal{T}$ represent columns and tables. Formally, the goal is to learn a mapping function as:

$$\mathcal{F} : (\mathcal{Q}, \mathcal{L}, \mathcal{D}) \rightarrow \mathcal{S}, \tag{1}$$

where for each input question $q_i \in \mathcal{Q}$, schema $\mathcal{D}$ and dialect $l_j \in \mathcal{L}$, the function $\mathcal{F}$ generates a valid database query $\mathcal{S}_{i,j}$.

Architecture Overview. The overall architecture of MoMQ is illustrated in Figure 2. MoMQ constructs MoE structure by leveraging LoRA modules as fine-grained experts. Dialect expert groups for each dialect and a shared expert group visible to all dialects are further introduced. The multi-level routing mechanism, composed of a dialect router and an expert router, ensures the correct routing of tokens between different expert groups and within each group. When the tokens of different dialects are input, they are initially directed through the dialect router to the appropriate dialect expert groups. Within these groups, the tokens are further routed by the expert router to the final activated experts. Notably, all tokens are fully processed by the shared expert group, which is beneficial for the fusion and transfer of multi-dialect knowledge. Besides, to better assist the dialect router in effectively routing tokens of various dialects to the appropriate expert group, we introduce a novel Dialect Router Loss.

3.1 MoE Construction

MoE structures are constructed in a variety of ways, such as pre-training from scratch or replicating multiple FFNs followed by a step of continual pre-training, all of which require additional pre-training to inject knowledge into the MoE (Komatsuzaki et al., 2022; Lin et al., 2024; Xue et al., 2024). Inspired by recent works (Wu et al., 2024; Li et al., 2024a; Feng et al., 2024). We use a simple but efficient way to construct MoE, freezing the original LLMs and inserting multiple Low-Rank Adaptation (LoRA) (Hu et al., 2021) modules into FFN as fine-grained experts. Specifically, a transformer FFN consists of two stacked layers, an up-projection layer and a down-projection layer. We replace the down-projection layer with multiple LoRA modules and deploy vanilla LoRA modules in both the up-projection layer and the attention layer to facilitate the model's rapid adaptation to multi-dialect query generation. The LoRA module consists of two matrices, $\mathbf{B} \in \mathbb{R}^{m \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times n}$, where $r \ll \min(m, n)$, and defines the adapted weight matrix $\mathbf{W}'$ as:

$$\mathbf{W}' = \mathbf{W_0} + \mathbf{BA}, \tag{2}$$

where $\mathbf{W_0} \in \mathbb{R}^{m \times n}$ is the original pre-trained weight matrix that remains fixed during the adaptation. All LoRA modules work parallel to the original layer to fully utilize the pre-trained knowledge.

3.2 Dialect Expert Group

In multi-dialect generation, there are non-trivial syntax differences between two database dialects. For example, in MySQL, the "DATE_ADD" function can be used to add a specified time interval to a date or datetime value. While PostgreSQL uses the "INTERVAL" keyword along with the "+" operator to achieve a similar result. These differences will interfere with learning dialect-specific knowledge.

To address this issue, we design multiple dialect expert groups to isolate dialect-specific knowledge, thereby mitigating interference and improving the quality of generated queries. The expert group consists of multiple LoRA experts and a top-k expert router. Each database dialect has a separate expert group to learn knowledge of the corresponding query syntax. Concretely, given multiple dialect expert groups $\mathcal{G} = \{G_1, G_2, \cdots, G_M\}$ in a specific Transformer layer, the output of the i-th expert group is calculated as:

$$\mathbf{H}_e^i = \sum_{k=1}^{N} g_k^i \cdot \mathbf{E}_k^i, \tag{3}$$

$$g_k^i = \begin{cases} s_k^i, & s_k^i \in \text{TopK}(\{s_k^i | 1 \le j \le N\}, K) \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

$$s_k^i = \text{softmax}(\mathbf{W}_e^i \cdot \mathbf{H})_k, \tag{5}$$

where $\mathbf{E}_k^i$ is the k-th LoRA expert in the i-th expert group, and N is the total number of experts in the group, g_k^i denotes the k-th gate value for the expert, s_k^i denotes the token-to-expert affinity, $\text{TopK}(\cdot, K)$ denotes the set comprising K highest affinity scores among those calculated for the tokens in all N experts, $\mathbf{W}_e^i \in \mathbb{R}^{d \times K}$ is a trainable matrix of the expert router, and $\mathbf{H}$ is the hidden states of all input tokens input. Note that g_k^i is sparse, indicating that only K out of N gate values are nonzero. This sparsity property ensures computational efficiency within an expert group, i.e., each token will be assigned to and computed in only K experts.

The routing mechanism within the expert group faces the problem of load imbalance (Shazeer et al., 2017). This can lead to a situation known as routing collapse, where the expert router continually selects a limited set of experts, thereby inhibiting adequate training for the others. In order to mitigate the risk of routing collapse, we employ an expert-level balance loss, which is computed as follows:

$$\mathcal{L}_{Bal} = N \cdot \sum_{i=1}^{N} f_i \cdot P_i, \tag{6}$$

$$f_i = \frac{1}{KT} \sum_{t=1}^{T} \mathbb{1}(Token\, t\, selects\, Expert\, i), \tag{7}$$

$$P_i = \frac{1}{T} \sum_{t=1}^{T} s_{i,t}, \tag{8}$$

where T is the number of processed tokens, $\mathbb{1}$ is the indicator function, f_i is the fraction of tokens dispatched to expert i and P_i is the fraction of the router probability allocated for expert i.

3.3 Shared Expert Group

There is a lack of inherent information communication between different dialects by using only a separate expert group for each dialect. When encountering data imbalance, it can negatively impact the common knowledge transfer from high-resource dialects to low-resource dialects. Meanwhile, multiple experts may converge in acquiring shared knowledge in their respective parameters, thereby resulting in redundancy in expert parameters. If there are shared experts dedicated to capturing and consolidating common knowledge across varying dialects, knowledge transfer will be more efficient and parameter redundancy will be alleviated.

Toward this objective, we further add a shared expert group to integrate information across multiple dialects at the sentence level. Regardless of the router module, all tokens in a sentence will be deterministically assigned to experts in the shared expert group. Formally, the MoE output in the complete MoMQ architecture is formulated as follows:

$$\mathbf{H}_o = \sum_{i=1}^{M} \mathbf{H}_e^i + \sum_{k=1}^{N_s} \mathbf{E}_k^s, \tag{9}$$

where M is the number of dialect expert groups, N_s is the number of shared experts and $\mathbf{E}_k^s$ is the output of k-th shared expert.

3.4 Dialect Router

After the construction of multiple dialect expert groups, how to route the tokens of different dialects to the appropriate group remains to be addressed. Intuitively, the dialect router is able to make correct routing under the guidance of sentence-level dialect hard labels, thus forming a complete isolation between different dialect expert groups. However, there may exist certain similarities between different dialects, e.g., "LIMIT" and "ORDER BY" in relational and non-relational database dialects are both valid tokens. Moreover, different dialects exhibit a high degree of similarity in the understanding of natural language questions and database schemas. If these similar tokens have the opportunity to enter multiple dialect expert groups, especially from high-resource dialects to low-resource ones, may further facilitate token-level common knowledge transfer.

To this end, we have designed a novel dialect router trained with a Dialect Router Loss (DRL) incorporating dialect smoothing. Dialect smoothing is employed to further reduce dialect isolation by replacing hard dialect labels with a smooth distribution. This distribution assigns a lower value to the true dialect while allocating a portion of the value mass to other dialects. Under the constraint of DRL, tokens excluding dialect-specific ones have a higher probability of being routed to various dialect experts group upon the output weight of the dialect router, thus facilitating more comprehensive dialect information exchange and knowledge transfer. We add the DRL to the training objective and it is formally defined as:

$$\tilde{y}_t = y_t(1-\epsilon) + \frac{\epsilon}{M}, \tag{10}$$

$$\mathcal{L}_{DR} = -\frac{1}{LT}\sum_{l=1}^{L}\sum_{t=1}^{T} r_t^l \log(\tilde{y}_t), \tag{11}$$

where y_t represents one-hot vector label among M dialect groups, $\epsilon \in [0, 1]$ is the smoothing factor, $\tilde{y}_t$ is the label after smooth, r_t^l denotes the output logits of dialect router in the l-th Transformer layer for the t-th token, L is the total number of layers, T it the total number of input tokens, and y_t is the dialect class label.

3.5 The Training Objectives

Finally, we formulate the multi-dialect query generation task as a text-to-text problem. The training objective is to minimize the negative log-likelihood of output $\mathbf{y}$ conditioned on the input question $\mathbf{x}$ and the task prompt $\mathbf{P}$. The fine-tuning loss on the task is defined as:

$$\mathcal{L}_{FT} = -\sum_j \mathcal{P}(y_j|\mathbf{y}_{<j}; \mathbf{x}, \mathbf{P}) + \alpha\mathcal{L}_{DR} + \lambda\mathcal{L}_{Bal}, \tag{12}$$

where α and λ are hyper-parameters that are used to adjust the impact of auxiliary losses.

4 Experiments

4.1 Datasets

There is currently no unified benchmark for evaluating the performance of LLMs in multi-dialect query generation. For text-to-SQL tasks, benchmarks like Spider (Yu et al., 2018) and BIRD (Li et al., 2024c) are widely used in English, while Chase (Guo et al., 2021) is used for Chinese. These benchmarks are based on SQLite. Additionally, multilingual benchmarks like MultiSpider (Dou et al., 2023) assess multilingual comprehension. To address this gap, we developed a training and evaluation dataset for converting natural language to query language for both relational and non-relational databases in English and Chinese. As shown in Table 2, our benchmark includes four dialects, with dataset construction details provided in Appendix A. All datasets

Backbone	Method	Execution Accuracy					Executable				
		MySQL	PG	Cypher	nGQL	Avg.	MySQL	PG	Cypher	nGQL	Avg.
Qwen2-1.5B	ICL	19.92	20.43	7.63	4.16	10.74	37.60	46.95	75.00	40.62	50.04
	Full Fine-Tuning	43.30	27.00	23.61	22.92	29.21	64.21	66.67	91.55	65.97	72.10
	Full Fine-Tuning*	27.80	24.30	21.30	9.61	20.75	45.63	67.37	87.15	62.38	65.64
	LoRA	49.20	32.16	25.12	26.97	33.36	67.28	69.84	**92.94**	71.53	75.40
	MoMQ (Ours)	**52.15**	**32.75**	**27.55**	**30.21**	**35.66**	**70.23**	**71.01**	91.78	**82.18**	**78.80**
Qwen2-7B	ICL	54.98	45.87	16.31	7.63	31.19	73.80	78.85	67.70	40.27	65.15
	Full Fine-Tuning	63.71	46.24	39.12	36.57	46.41	83.52	82.75	**96.53**	85.07	86.97
	Full Fine-Tuning*	58.92	44.37	**41.44**	38.66	45.84	78.72	**85.09**	95.95	85.65	86.35
	LoRA	63.35	44.95	38.31	27.31	43.48	84.87	83.80	95.83	82.52	86.76
	MoMQ (Ours)	**66.30**	**48.12**	40.97	**41.20**	**49.15**	**87.21**	84.51	95.83	**87.38**	**88.73**
Qwen1.5-14B	ICL	41.69	42.29	19.79	3.12	26.72	61.62	68.81	69.79	37.15	59.34
	Full Fine-Tuning	46.68	45.95	36.46	28.65	39.43	67.16	80.99	93.23	80.90	80.57
	Full Fine-Tuning*	34.93	44.01	37.85	28.24	36.26	53.51	**82.51**	93.52	68.87	74.60
	LoRA	50.55	45.89	36.11	28.59	40.29	70.85	81.93	91.67	79.17	80.90
	MoMQ (Ours)	**61.75**	**47.65**	**39.47**	**34.26**	**45.78**	**81.55**	81.69	**94.91**	**82.18**	**85.08**

Table 1: Results in the full data setting. PG refers to PostgreSQL and * denotes single-dialect full fine-tuning.

underwent manual inspection and filtering to ensure query correctness and executability.

Dialect	Split	Source	Count
MySQL	Train	Spider + BIRD + Chase	3,000
	Test	BIRD	280
PG	Train	BIRD	3,000
	Test	SQL-Eval[1]	304
Cypher	Train	Text-To-Cypher[2]	3,000
	Test		288
nGQL	Train	NL2GQL(Zhou et al., 2024)	3,000
	Test		288

Table 2: The statistics of data for each dialect.

4.2 Experimental Setup

Evaluation Metric. We consider two prevalent evaluation metrics: execution accuracy (EX) and executable (EXEC). The EX metric evaluates whether the predicted and ground-truth queries yield the same execution results on the database. The EXEC metric evaluates whether the generated query can be executed correctly in the corresponding database without syntax errors.

Models. MoMQ can be used on a variety of open-source LLMs. We select three model sizes of current SOTA models, namely Qwen2-1.5B, Qwen2-7B, and Qwen1.5-14B from HuggingFace[3], as backbones respectively to validate MoMQ's multi-dialect query generation capabilities and robustness. All models use the Instruct or Chat version instead of the Base version to obtain better performance. The implementation details are shown in Appendix B.

[1]https://github.com/defog-ai/sql-eval
[2]https://github.com/neo4j-labs/text2cypher
[3]https://huggingface.co/

Baselines. We compare MoMQ with the following methods: (1) In-context Learning (ICL), which adds one example into the prompt without updating any model parameters; (2) Single-dialect full fine-tuning, which fine-tunes all parameters of the model in each dialect data; (3) Multi-dialect full fine-tuning, which fine-tunes in a mixed dataset of multiple dialects; (4) Vanilla LoRA, which freezes the pre-trained model and replaces all linear layers with LoRA modules in a mixed dataset of multiple dialects. All the baselines use the same prompt template, as shown in Appendix C.

4.3 Results

Full Data. Experimental results in Table 1 indicate that MoMQ significantly outperforms both single-dialect and multi-dialect full fine-tuning over nearly 3-5% on EX and EXEC in the full data setting. Additionally, MoMQ shows consistent improvement as the model size increases from 1.5B to 14B. Notably, for the 1.5B model, the EX for MySQL and nGQL improves by nearly 10%. This demonstrates the capability of MoMQ to effectively isolate dialect-specific knowledge and leverage generalized knowledge across multiple dialects for improved performance. Meanwhile, MoMQ is robust enough to achieve stable improvements across various backbones. We further conducted a case study on the 7B model, as shown in Appendix D.

Imbalanced Data. We evaluate MoMQ's transfer capabilities in two data imbalance settings. The first, MySQL high-resource, samples the entire MySQL dataset and 128 samples from each of the other three dialects. As shown in Table 3, MoMQ consistently outperforms full fine-tuning

Backbone	Method	Execution Accuracy					Executable				
		MySQL	PG	Cypher	nGQL	Avg.	MySQL	PG	Cypher	nGQL	Avg.
Qwen2-1.5B	ICL	19.92	20.43	7.63	4.16	10.74	37.60	46.95	75.00	40.62	50.04
	Full Fine-Tuning	30.87	23.76	15.74	3.13	18.37	48.46	56.50	80.79	59.38	61.28
	LoRA	35.55	**27.30**	17.82	4.17	21.21	53.01	**60.28**	85.42	48.73	61.86
	MoMQ (Ours)	**39.73**	24.35	**19.21**	**6.25**	**22.39**	**60.27**	60.05	**86.46**	**59.72**	**66.62**
Qwen2-7B	ICL	54.98	45.87	16.31	7.63	31.19	73.80	78.85	67.70	40.27	65.15
	Full Fine-Tuning	60.27	45.39	27.66	**11.46**	36.20	78.84	**82.03**	92.36	**50.35**	**75.90**
	LoRA	61.38	43.62	27.32	3.94	34.06	**81.30**	79.43	91.90	33.91	71.64
	MoMQ (Ours)	**63.22**	**46.93**	**28.59**	9.95	**37.17**	**81.30**	81.32	**92.82**	45.37	75.21
Qwen1.5-14B	ICL	41.69	42.29	19.79	3.12	26.72	61.62	68.81	69.79	37.15	59.34
	Full Fine-Tuning	39.98	41.25	26.04	5.79	28.26	57.69	71.75	**90.28**	62.04	70.44
	LoRA	43.05	41.02	23.03	1.74	27.21	60.39	70.45	82.41	31.48	61.18
	MoMQ (Ours)	**53.38**	**45.39**	**26.50**	**13.54**	**34.70**	**74.91**	**75.77**	87.50	**64.58**	**75.69**

Table 3: Results in the MySQL high-resource setting. All available data from MySQL is utilized, while 128 examples are sampled from each of the other dialects.

Backbone	Method	Execution Accuracy					Executable				
		MySQL	PG	Cypher	nGQL	Avg.	MySQL	PG	Cypher	nGQL	Avg.
Qwen2-1.5B	ICL	19.92	20.43	7.63	4.16	10.74	37.60	46.95	75.00	40.62	50.04
	Full Fine-Tuning	28.41	**27.42**	21.76	1.62	19.80	44.40	54.61	87.73	49.65	59.10
	LoRA	31.98	25.77	24.77	3.47	21.50	48.71	59.46	91.90	49.19	62.31
	MoMQ (Ours)	**33.09**	22.81	**26.04**	**5.79**	**21.93**	**49.20**	**64.18**	**92.59**	**55.90**	**65.47**
Qwen2-7B	ICL	54.98	45.87	16.31	7.63	31.19	73.80	78.85	67.70	40.27	65.15
	Full Fine-Tuning	61.13	47.99	43.17	5.67	39.49	78.48	81.68	**96.18**	47.57	75.98
	LoRA	**62.12**	**50.35**	38.66	4.17	38.82	**80.32**	**83.69**	94.56	36.46	73.76
	MoMQ (Ours)	**62.12**	48.46	**44.68**	**10.53**	**41.45**	78.84	82.39	95.72	**56.02**	**78.24**
Qwen1.5-14B	ICL	41.69	42.29	19.79	3.12	26.72	61.62	68.81	69.79	37.15	59.34
	Full Fine-Tuning	36.65	**47.99**	38.08	3.70	31.61	54.12	78.49	94.33	60.30	71.81
	LoRA	**46.99**	43.50	31.02	2.43	30.98	**63.10**	72.22	90.39	50.00	68.93
	MoMQ (Ours)	43.42	46.45	**40.05**	**10.19**	**35.03**	57.32	**79.67**	**94.79**	**65.86**	**74.41**

Table 4: Results in the Cypher high-resource setting. All available data from Cypher is utilized, while 128 examples are sampled from each of the other dialects.

and LoRA, achieving nearly a 5% improvement in average EX for the 14B model.

The second, Cypher high-resource, samples the entire Cypher dataset and 128 samples from each of the other three dialects. As shown in Table 4, MoMQ generally outperforms other methods, particularly excelling in Cypher and nGQL due to their high syntactical similarity, enabling effective transfer of dialect-common knowledge. However, other methods occasionally perform better, such as in MySQL and PG, primarily due to insufficient training data for relational dialects and the greater differences between relational and non-relational databases, which hinder knowledge transfer.

5 Analysis

To further validate the effectiveness, robustness, and interpretability of MoMQ, we design a variety of analytical experiments. All experiments are conducted using the Qwen2-7B backbone in the full data setting.

Ablation Study. To evaluate the effectiveness and impact of different components of MoMQ, we conduct ablation studies on Qwen2-7B. As shown in Table 5, we remove different components from MoMQ and evaluate the average EX. Upon conducting the ablations, notable decreases in performance are observed. Specifically, the exclusion of the shared expert group results in a drop in average EX from 49.15% to 48.57%. Further removal of the dialect router loss leads to an additional decline, with the average EX decreasing to 47.08%. In this case, tokens are routed with hard dialect labels at the sentence level, resulting in complete dialect isolation. The most significant reduction occurs when the dialect router is removed, resulting in an average EX of 44.79%. In this case, tokens are randomly assigned to the dialect expert groups without any supervision. Finally, eliminating dialect expert groups causes the entire structure to revert to LoRA, further reducing the EX to 43.48%. These results

Component	Ablation				
Shared Expert Group	✓	×	×	×	×
Dialect Router Loss	✓	✓	×	×	×
Dialect Router	✓	✓	✓	×	×
Dialect Expert Group	✓	✓	✓	✓	×
Avg. EX	**49.15**	48.57	47.08	44.79	43.48

Table 5: Results of ablation studies on Qwen2-7B.

underscore the critical contributions of the shared expert group, dialect router loss, dialect router, and dialect expert group to the overall performance of MoMQ.

Effect of Expert Dimension. We further analyze the impact of the expert dimension on the performance of MoMQ, where the expert dimension refers to the LoRA module's rank. As illustrated in Table 6, MoMQ demonstrates a consistent increase in average EX as the expert dimension increases from 16 to 128. Specifically, with an expert dimension of 128, MoMQ achieves the highest average EX of 49.15%. It is interesting to note that when the expert dimension is further increased to 256, there is a slight decrease in the average EX. This decline suggests a potential issue where a large expert dimension may lead to inadequate training within 3 epochs and consequently impact MoMQ's performance negatively.

Expert	Execution Accuracy				
Dimension	MySQL	PG	Cypher	nGQL	Avg.
16	65.56	44.33	39.93	34.49	46.08
64	64.95	46.10	**42.82**	36.00	47.47
128	**66.30**	48.12	40.97	**41.20**	**49.15**
256	65.19	**48.23**	41.20	39.24	48.46

Table 6: Results with different expert dimensions on Qwen2-7B.

Effect of Expert Number. To comprehensively analyze the impact of the number of experts, we evaluate MoMQ with configurations of 8, 16, 32, and 64 experts, respectively. As shown in Table 7, MoMQ demonstrates robust performance across all configurations, particularly excelling with 8 and 32 experts. When the number of experts reaches 64, there is also a certain decline in MoMQ's performance. This is attributed to a similar reason as when the expert dimension reaches 256, indicating that the experts are not sufficiently trained.

Expert	Execution Accuracy				
Number	MySQL	PG	Cypher	nGQL	Avg.
8	**66.67**	46.93	**42.25**	38.77	48.65
16	65.68	48.11	40.74	38.19	48.18
32	66.30	**48.12**	40.97	**41.20**	**49.15**
64	65.68	47.40	37.62	38.89	47.40

Table 7: Results with different expert numbers on Qwen2-7B.

Expert Weight Distribution. To further analyze the effect of the multi-level routing mechanism, we collect the output logits of the dialect router and the expert router from all Transformer layers when generating the nGQL query in the case study. As illustrated in Figure 3, under the constraint of the Expert Balance Loss, expert weights within each dialect expert group remain balanced after training. At the same time, each dialect expert group has activated experts, indicating that the expert groups are not entirely isolated. Tokens from the nGQL dialect have the opportunity to influence other expert groups, suggesting a degree of knowledge transfer. Furthermore, we observe that the weights of the nGQL expert group are significantly greater than those of the other groups, demonstrating that the nGQL experts play a dominant role in this generation process. This observation further validates the effectiveness of the Dialect Router Loss.

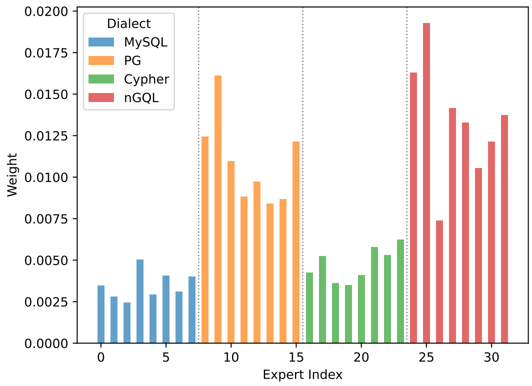

Figure 3: Expert weight distribution of generating the nGQL query. A certain number of experts are activated in each expert group and the nGQL expert group plays a dominant role in this generation process.

6 Conclusion

In this paper, we introduce MoMQ, a Mixture-of-Experts framework for multi-dialect query generation across relational and non-relational databases. MoMQ assigns dedicated expert groups for each dialect to isolate dialect-specific knowledge and mitigate interference. We further design a novel multi-level routing mechanism comprising a dialect router and an expert router to ensure correct token-level routing and enhance knowledge sharing through the Dialect Router Loss and a shared expert group. We have also constructed a comprehensive multi-dialect dataset (covering MySQL, PostgreSQL, Cypher, and nGQL) to support further research in this area.

Limitations

MoMQ adopts a sparse MoE structure, where increasing the number of expert groups and experts per group leads to a larger model size and reduced training efficiency. Additionally, since the MoE structure is built on a dense model, inference requires activating all original model parameters, resulting in higher computational costs. Future work can explore parameter sparsification and pruning within expert groups to remove redundant dialect knowledge, enhancing both training and inference efficiency.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *Preprint*, arXiv:2009.09796.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *Preprint*, arXiv:2401.06066.

Longxu Dou, Yan Gao, Mingyang Pan, Dingzirui Wang, Wanxiang Che, Dechen Zhan, and Jian-Guang Lou. 2023. Multispider: towards benchmarking multilingual text-to-sql semantic parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12745–12753.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Preprint*, arXiv:2101.03961.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. 2024. Mixture-of-loras: An efficient multitask tuning for large language models. *Preprint*, arXiv:2403.03432.

Han Fu, Chang Liu, Bin Wu, Feifei Li, Jian Tan, and Jianling Sun. 2023. Catsql: Towards real world natural language to sql applications. *Proceedings of the VLDB Endowment*, 16(6):1534–1547.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-sql empowered by large language models: A benchmark evaluation. *Proc. VLDB Endow.*, 17(5):1132–1145.

Jiaqi Guo, Ziliang Si, Yu Wang, Qian Liu, Ming Fan, Jian-Guang Lou, Zijiang Yang, and Ting Liu. 2021. Chase: A large-scale and pragmatic Chinese dataset for cross-database context-dependent text-to-SQL. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2316–2331, Online. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. 2022. Sparse upcycling: Training mixture-of-experts from dense checkpoints. In *The Eleventh International Conference on Learning Representations*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *Preprint*, arXiv:2006.16668.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. 2024a. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *Preprint*, arXiv:2404.15159.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024b. Codes: Towards building open-source language models for text-to-sql. *Proc. ACM Manag. Data*, 2(3).

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023b. Graphix-t5: mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024c. 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*, 36.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023c. Starcoder: may the source be with you! *Preprint*, arXiv:2305.06161.

Zhishuai Li, Xiang Wang, Jingjing Zhao, Sun Yang, Guoqing Du, Xiaoru Hu, Bin Zhang, Yuxiao Ye, Ziyue Li, Rui Zhao, and Hangyu Mao. 2024d. Pet-sql: A prompt-enhanced two-round refinement of text-to-sql with cross-consistency. *Preprint*, arXiv:2403.09732.

Bin Lin, Zhenyu Tang, Yang Ye, Jiaxi Cui, Bin Zhu, Peng Jin, Junwu Zhang, Munan Ning, and Li Yuan. 2024. Moe-llava: Mixture of experts for large vision-language models. *arXiv preprint arXiv:2401.15947*.

Neo4j. 2012. Neo4j - the world's leading graph database.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. 2020. Which tasks should be learned together in multi-task learning? In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9120–9132. PMLR.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Min Wu, Xinglu Yi, Hui Yu, Yu Liu, and Yujue Wang. 2022. Nebula graph: An open source distributed graph database. *Preprint*, arXiv:2206.07278.

Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of lora experts. *Preprint*, arXiv:2404.13628.

Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. 2024. Openmoe: An early effort on open mixture-of-experts language models. *Preprint*, arXiv:2402.01739.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. 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*, pages 3911–3921.

Yu Zhang and Qiang Yang. 2022. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609.

Xinyu Zhao, Xuxi Chen, Yu Cheng, and Tianlong Chen. 2024. Sparse moe with language guided routing for multilingual machine translation. In *The Twelfth International Conference on Learning Representations*.

Yuhang Zhou, Yu He, Siyu Tian, Yuchen Ni, Zhangyue Yin, Xiang Liu, Chuanjun Ji, Sen Liu, Xipeng Qiu, Guangnan Ye, and Hongfeng Chai. 2024. r^3-NL2GQL: A model coordination and knowledge graph alignment approach for NL2GQL. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13679–13692, Miami, Florida, USA. Association for Computational Linguistics.

Ran Zmigrod, Salwa Alamir, and Xiaomo Liu. 2024. Translating between sql dialects for cloud migration. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '24, page 189–191, New York, NY, USA. Association for Computing Machinery.

Barret Zoph. 2022. Designing effective sparse expert models. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1044–1044.

A Dataset Construction Details

For MySQL, we each sampled 1,500 examples from the training sets of Spider and Chase for training. Additionally, we selected the "superhero" and "student_club" databases from the dev set of BIRD as the test set. The syntax of all samples was accurately converted from SQLite to MySQL.

Regarding PG, we obtained 3,000 samples from BIRD, transforming the original SQLite syntax into PostgreSQL syntax for the training set. As for the test set, we directly used SQL-Eval, an evaluation set released by Defog. It's based on the schema from Spider but with a new set of hand-selected questions and queries grouped by query category.

For Cypher, the samples were acquired from the open-source text-to-cypher dataset, which includes over 16 different graph schemas, along with graph information for evaluation. We sampled "fincen" and "movies" databases as the test set and used the remainder for training.

In the case of nGQL, we utilized a dataset published by Zhou et al. (2024). It involves matching data from different Knowledge Graphs to the NebulaGraph format, as well as generating training and testing data.

B Implementation Details

Our experiments are conducted using PyTorch 2.3.1 on a computer running the Ubuntu 20 operating system, equipped with 8 NVIDIA A100 80GB GPUs. We establish a shared expert group with 2 LoRA experts and four dialect-specific expert groups corresponding to MySQL, PostgreSQL, Cypher, and nGQL. Each dialect expert group comprises 8 LoRA experts, with each input token activating the top-2 experts. For models with different parameter sizes, we employ varying expert dimensions: 64 dimensions for the 1.5B model, 128 dimensions for the 7B model, and 256 dimensions for the 14B model. To optimize the objectives, we use the AdamW optimizer with parameters set to $\beta_1 = 0.9$ and $\beta_2 = 0.95$. The learning rate is set to $1e^{-6}$ for full fine-tuning and $1e^{-6}$ for the others, accompanied by a weight decay of 0.1 and $\epsilon = 0.1$ for the smoothing factor. We set $\alpha = 0.1$ and $\lambda = 0.001$ to adjust the impact of auxiliary losses. All experiments are run for 3 epochs. Each experiment is repeated three times with different random seeds, and the mean values are reported. The random seed is shared by all compared methods for a fair comparison.

C Prompt Templates

As shown in Figure 4, the prompt template used in multi-dialect query generation consists primarily of the dialect identifier (blue), user question, database schema, and references. The ICL method additionally includes an examples section (red) to activate the model's latent query generation capabilities.

D Case Study

Figure 5 shows the comparison of different methods for generating nGQL and MySQL queries. From the above results, only MoMQ generates the correct queries. For nGQL dialect, the multi-dialect full fine-tuning method uses the "MATCH" statement instead of the "GO FROM" statement, which is interfered by the Cypher syntax. Single-dialect full fine-tuning and LoRA methods are unable to generate the correct query, although it uses the right nGQL syntax. For MySQL dialect, both full fine-tuning and LoRA do not perform a join operation on the table, indicating a lack of understanding of the question and schema. These cases further illustrate that MoMQ has a stronger resistance to interference and a better understanding of common information compared to other methods.

Prompt

You are a [PostgreSQL, MySQL, Cypher,nGQL] expert. You need to read and understand the following [Database Schema], as well as any applicable [Reference]. Then, using your [PostgreSQL, MySQL, Cypher,nGQL] knowledge, generate the corresponding query that answers the [UseQuestion].

[Database schema]

Table: author……

[Reference]

……

[Example]

User Question : Which authors are not part of any organization?

Expected Output: SELECT DISTINCT name, aid FROM author WHEREoid IS NULL

[User Question]

Which conference published the most publications in the last 15 years? Give the conference name and publication count.

[Query]

Figure 4

Dialect: nGQL **Question:** Find all the entities that like Steve Nash and how they relate to the target entity? **Schema**: # Relationship: like (likeness, src_vid, dst_vid, …) … **Gold Query**: GO FROM "Steve Nash" OVER like REVERSELY YIELD like._dst	Full Fine-Tuning ✗	MATCH (v1) -[:like]-> (v) WHERE id(v1) == 'Steve Nash' RETURN (v1)-[:like]->(v)
	Full fine-Tuning* ✗	GO FROM "Steve Nash" OVER like YIELD like._dst
	LoRA ✗	GO FROM "Steve Nash" OVER like YIELD like._dst, like._src | GO 1 STEPS FROM $-.like._dst OVER like YIELD like._src
	MoMQ ✓	GO FROM "Steve Nash" OVER like REVERSELY YIELD like._dst
Dialect: MySQL **Question:** How many members of Business have the Medium size of tee shirt? **Schema**: #Table: member (member_id, link_to_major, …) #Table: major (major_id, …) … **Gold Query**: SELECT COUNT(T2.member_id) FROM major AS T1 INNER JOIN member AS T2 ON T1.major_id = T2.link_to_major WHERE T1.major_name = 'Business' AND T2.t_shirt_size = 'Medium'	Full Fine-Tuning ✗	SELECT count(*) FROM member WHERE major_name = 'Business' AND t_shirt_size = 'Medium'
	Full fine-Tuning* ✗	SELECT count(*) FROM member WHERE major_name = 'Business' AND t_shirt_size = 'Medium'
	LoRA ✗	SELECT count(*) FROM member WHERE major_name = 'Business' AND t_shirt_size = 'Medium'
	MoMQ ✓	SELECT count(*) FROM member AS T1 JOIN major AS T2 ON T1.link_to_major = T2.major_id WHERE T2.major_name = "Business" AND T1.t_shirt_size = "Medium"

Figure 5: Case study of generating nGQL and MySQL queries of different methods in the full data setting.