# K-DECORE: Facilitating Knowledge Transfer in Continual Structured Knowledge Reasoning via Knowledge Decoupling

**Yongrui Chen**[1,2] **Yi Huang**[3,4,*] **Yunchang Liu**[1,2] **Shenyu Zhang**[1,2]
**Junhao He**[1,2] **Tongtong Wu**[5] **Guilin Qi**[1,2] **Tianxing Wu**[1,2,*]
[1]School of Computer Science and Engineering, Southeast University, China
[2]Key Laboratory of New Generation Artificial Intelligence Technology and its Interdisciplinary Applications (Southeast University), Ministry of Education, China
[3]JIUTIAN Research, China Mobile
[4]Department of Computer Science and Technology, Tsinghua University, China
[5]Department of Data Science & AI, Monash University, Australia
{yongruichen, tianxingwu}@seu.edu.cn, huangyi@chinamobile.com

## Abstract

Continual Structured Knowledge Reasoning (CSKR) focuses on training models to handle sequential tasks, where each task involves translating natural language questions into structured queries grounded in structured knowledge. Existing general continual learning approaches face significant challenges when applied to this task, including poor generalization to heterogeneous structured knowledge and inefficient reasoning due to parameter growth as tasks increase. To address these limitations, we propose a novel CSKR framework, K-DECORE, which operates with a fixed number of tunable parameters. Unlike prior methods, K-DECORE introduces a knowledge decoupling mechanism that disentangles the reasoning process into task-specific and task-agnostic stages, effectively bridging the gaps across diverse tasks. Building on this foundation, K-DECORE integrates a dual-perspective memory consolidation mechanism for distinct stages and introduces a structure-guided pseudo-data synthesis strategy to further enhance the model's generalization capabilities. Extensive experiments on four benchmark datasets demonstrate the superiority of K-DECORE over existing continual learning methods across multiple metrics, leveraging various backbone large language models.

## 1 Introduction

Structured Knowledge Reasoning (SKR) aims to translate the natural language question into structured queries over discrete, structured knowledge —such as relational databases, knowledge graphs, and dialogue states. It is essential for a wide range of real-world applications, including legal judgment prediction [1], clinical decision support [2, 3], and financial analysis [4]. Recent advances in Large Language Models (LLMs) have demonstrated impressive capabilities for structured reasoning when provided with appropriate prompting or fine-tuning [5, 6, 7].

However, most existing methods operate under a static assumption that SKR tasks are singular in type and remain unchanged throughout both the training and deployment phases. This assumption is misaligned with real-world scenarios, where models must continuously adapt to new reasoning tasks across various forms of structured knowledge. For instance, a virtual assistant like Siri or Alexa must

*Corresponding authors.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

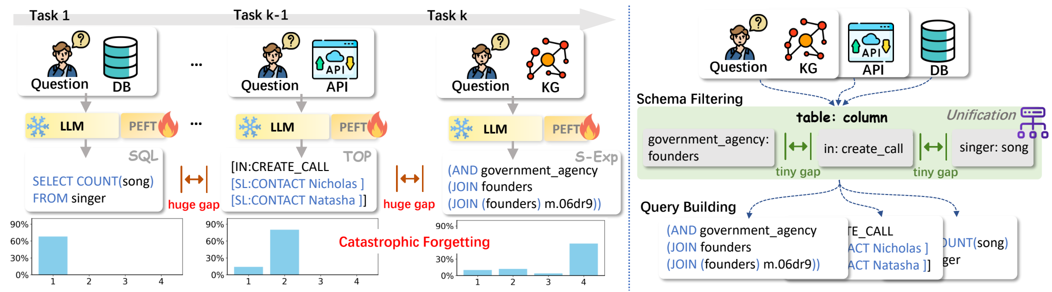


Figure 1: a) Overview of the continual SKR task. The backbone LLM is frozen and only the PEFT module if tunable. b) SKR based on knowledge decoupling. Schema filtering refines the scope of a given schema and impacts various SKR performance. The differences between schema filtering in these SKR tasks are minimal, making it potential for *knowledge transfer*.

dynamically adjust its knowledge base to handle a variety of tasks such as managing calendar events, to-do lists, and smart home controls, each characterized by distinct schemas and query languages. Therefore, it is essential to empower the model to leverage knowledge transfer across different tasks during the continual learning process [8].

Recent works to continual SKR have primarily focused on designing the training strategy for a single type of structured knowledge (e.g., continual text-to-SQL) [9, 10] or employing parameter-efficient fine-tuning (PEFT) techniques to allocate task-specific parameters [11, 12, 13]. However, these methods often fail to generalize across heterogeneous structured knowledge and suffer from parameter growth proportional to the number of tasks, ultimately hindering reasoning efficiency.

We observe that schema filtering, i.e., identifying schema elements relevant to query construction, (Figure 1(b)) is a shared, reusable component across diverse SKR tasks and significantly influences overall performance. We hypothesize that decoupling pattern filtering from downstream stages and unifying its input-output format across tasks can enhance robustness to task-specific variations. This consistency facilitates knowledge transfer and improves SKR performance without requiring parameter growth or task-specific reviews. For query construction stages with high task variance, their contribution to SKR depends on the structural richness of replayed queries. We propose to automatically synthesize structurally diverse replay samples, enabling more informative replay memory usage within the same capacity budget.

In this paper, we propose K-DECORE (Knowledge DEcoupling for COntinual REasoning), a novel continual SKR framework. K-DECORE comprises a backbone LLM and two lightweight PEFT modules: a schema filter and a query builder, responsible for capturing task-specific and task-agnostic knowledge, respectively. This decoupled design fosters forward and backward knowledge transfer. To mitigate the model's tendency to forget previously learned tasks, K-DECORE incorporates a dual-perspective memory mechanism (Section 3.2) designed to retain replay samples from both schema and query structure viewpoints. Specifically, the schema memory captures representative schema instances, ensuring the filter maintains comprehensive coverage across tasks, while the query memory emphasizes preserving diverse query structures inherent to SKR tasks. Furthermore, to enhance the model's generalization to unseen patterns, we introduce a query synthesis strategy that generates novel structured queries for replay. We evaluate K-DECORE on task streams comprising four diverse SKR tasks, where it consistently outperforms strong baselines and achieves state-of-the-art results across multiple metrics. In summary, the contributions of this paper include:

- We propose a novel continual structured knowledge reasoning framework, leveraging knowledge decoupling to enable effective knowledge transfer across diverse tasks. To the best of our knowledge, this is the first work to explore continual learning across heterogeneous SKR tasks.
- We propose a dual-perspective memory construction mechanism that specifically synthesizes pseudo queries for novel structures, with the goal of enhancing the LLM's generalization capability.
- We curate task streams from four SKR benchmarks and conduct comprehensive experiments. Empirical results demonstrate that K-DECORE consistently surpasses strong baselines across multiple evaluation metrics, establishing its effectiveness in continual structured knowlegde reasoning.

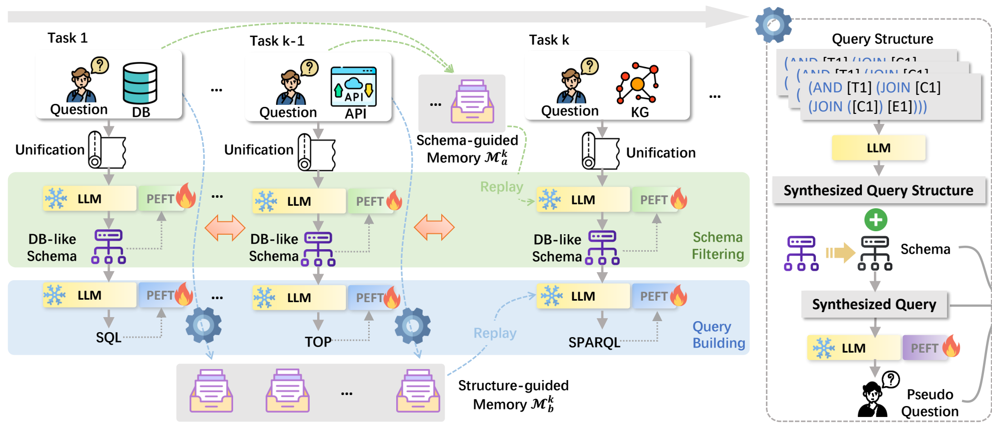


Figure 2: The left panel presents the K-DeCore training framework, organized into two key stages for each SKR task: schema filtering and query building, each supported by specialized PEFT modules. By unifying the schema, the framework aims to bridge the gap between tasks, effectively enabling the *knowledge transfer*. The right panel illustrates the creation process of structure-guided synthetic pseudo samples, designed to offer a more structurally diverse set of examples.

## 2 Preliminary

**Structured Knowledge Reasoning** Given a natural language question $\mathcal{Q}$ and accessible structured knowledge $\mathcal{S}$, such as a database, a dialogue state, or a knowledge graph, the goal of Structured Knowledge Reasoning is to generate a structured query $\mathcal{Y}$. Formally, this can be represented as: $\mathcal{Y} = f_\theta(\mathcal{Q}, \mathcal{S})$, where $f_\theta$ denotes a LLM-based reasoner parameterized by $\theta$.

In this paper, we focus on three commonly used types of structured knowledge: a) A *database* is represented as $(\mathcal{C}, \mathcal{T})$, where $\mathcal{C} = c_1, \ldots, c_n$ denotes the column names and $\mathcal{T} = \{t_1, \ldots, t_m\}$ denotes the table names. b) A *knowledge graph* is generally composed of subject-predicate-object triples, expressed as $\{\langle s, p, o\rangle \,|s \in \mathcal{E}, p \in \mathcal{R}, o \in \mathcal{E} \cup \Gamma\}$, where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations, and $\Gamma$ is the set of types. c) A *dialogue state* is represented as a collection of predefined intents paired with their corresponding slots, indicated by $\{\langle \mathcal{I}_1, s_1\rangle, \ldots, \langle \mathcal{I}_n, s_n\rangle\}$, where each $\mathcal{I}_i$ represents an intention and $s_i$ denotes the associated slots.

**Problem Formulation** Let $f_\theta$ be trained sequentially on $K$ SKR tasks, denoted by $\{\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^K\}$. Each task $\mathcal{D}^i$ consists of a training set and a test set: $\mathcal{D}^k = \mathcal{D}^k_{\text{train}} \cup \mathcal{D}^k_{\text{test}} = \{(\mathcal{Q}_i, \mathcal{S}_i, \mathcal{Y}_i)\}_{i=1}$. To encourage $f_\theta$ to continually acquire reasoning capabilities over diverse types of structured knowledge, for tasks $\mathcal{D}^k$ and $\mathcal{D}^j$ $(k \neq j)$, the corresponding structured knowledge $\mathcal{S}(\mathcal{D}^k)$ and $\mathcal{S}(\mathcal{D}^j)$ are of different types, i.e., $\text{Type}(\mathcal{S}(\mathcal{D}^k)) \neq \text{Type}(\mathcal{S}(\mathcal{D}^j))$. Our objective is to ensure that $f_\theta$ performs well on each $\mathcal{D}^k_{\text{test}}$ after sequentially learning on all $\mathcal{D}^k_{\text{train}}$, for all $k \in \{1, \ldots, K\}$.

## 3 K-DeCore

Figure 2 illustrates the overall architecture of the proposed K-DeCore. It is built upon a backbone LLM $f$ with parameter $\theta$, and incorporates three PEFT modules: $\mathbf{P}_a$, $\mathbf{P}_b$, and $\mathbf{P}_c$. Throughout training, the backbone parameters $\theta$ remain fixed; only the parameters of the PEFT modules are updated, enabling efficient adaptation to the task while preserving the pretrained knowledge of the LLM. The entire method can be divided into two parts: *continual knowledge decoupling* (Section 3.1) and *dual-perspective memory construction* (Section 3.2), which are described in detail below.

### 3.1 Continual Knowledge Decoupling

Unlike traditional continual learning methods [13, 14] in general fields, our proposed K-DeCore innovatively bifurcates the reasoning process over the structured knowledge into two distinct yet synergistic phases: *schema filtering* and *query building*. The schema filtering phase is designed to distill the essential schema components $\mathcal{S}^* \subseteq \mathcal{S}$ necessary for constructing the final query $\mathcal{Y}$ from the initial schema $\mathcal{S}$. Subsequently, the query building phase focuses on generating the query $\mathcal{Y}$ using both the original schema $\mathcal{S}$ and the refined schema $\mathcal{S}^*$ obtained from schema filtering. This decoupling strategy offers dual advantages: firstly, within-task pattern filtering has been empirically demonstrated to enhance the performance of query generation by reducing the search space; secondly, the relatively stable output format of schema filtering across diverse SKR tasks enables the model to facilitate more robust knowledge transfer from preceding tasks during the schema filtering phase. In the following sections, we will delve into a detailed exploration of these two components.

#### 3.1.1 Task-agnostic Continual Schema Filtering

K-DECORE trains an independent PEFT module, denoted as $\mathbf{P}_a$, for the schema filtering stage across all tasks. This module is integrated with the backbone LLM $f_\theta$ to predict the relevant schema set $\mathcal{S}^*$, i.e., $\mathcal{S}^* = f(\mathcal{Q}, \mathcal{S}; \theta, \mathbf{P}_a)$. During the training process for task $k$, the module $\mathbf{P}_a^k$ is initialized with parameters from the checkpoint of the preceding task, represented as $\mathbf{P}_a^{k-1}$. To ensure that $\mathbf{P}_a$ perceives a consistent style akin to previous tasks while learning new SKR tasks, thereby mitigating forgetting due to training, the knowledge schema $\mathcal{S}$ from various SKR tasks is standardized into a database-like (DB-like) *unified schema representation*. This standardization further reduces task-specific discrepancies and enhances effective knowledge transfer. The DB style is chosen as the unified format for two primary reasons: (a) it is likely the most prevalent form of $\mathcal{S}$ and is familiar to LLMs, and (b) its straightforward relational structure simplifies the conversion of other schema types. Concretely, for each $(\mathcal{Q}, \mathcal{S}, \mathcal{Y})$, $\mathcal{S}$ is first converted to $\Omega = (\Phi, \Psi)$, where $\Phi = \{\phi_1, \ldots, \phi_{|\Phi|}\}$ denotes the set of abstract table names, and $\Psi = \{\psi_1, \ldots, \psi_{|\Psi|}\}$ denotes the set of abstract column names. Then, $f(\theta, \mathbf{P}_a)$ predicts the useful schema $\Omega^*$ from $\Omega$, i.e., $\Omega^* = f(\mathcal{Q}, \mathcal{Z}; \theta, \mathbf{P}_a)$, where $\widetilde{\Omega}$ denotes the textual representation of $\Omega$:

$$\widetilde{\Omega} = \phi_1 : \psi_1^1, \ldots, \psi_1^n \mid \phi_2 : \psi_2^1, \ldots, \psi_2^n \mid \ \ldots \ \mid \phi_m : \psi_m^1, \ldots, \psi_m^n. \tag{1}$$

During training, the following loss function is minimized to optimize $\mathbf{P}_a$, while keeping $\theta$ frozen:

$$\mathcal{L}(\mathcal{D}^k; \theta, \mathbf{P}_a) = -\sum_{i=1}^{|\mathcal{D}^k|} \sum_{j=1}^{|\widetilde{\Omega}_i^*|} \log P(\omega_j^* \mid \mathcal{Q}_i, \widetilde{\Omega}_i, \omega_{<j}^*; \theta, \mathbf{P}_a) + \sum_{k'=1}^{k-1} \mathcal{L}(\mathcal{M}_a^{k'}; \theta, \mathbf{P}_a), \tag{2}$$

where $P(z_j^* \mid \mathcal{Q}_i, \widetilde{\Omega}_i, \omega_{<j}^*; \theta, \mathbf{P}_a)$ denotes the probability of each token $z_j^*$ generated by autoregression. To further mitigate the model's forgetting, we incorporate a review loss $\mathcal{L}(\mathcal{M}_a^{k'}; \theta, \mathbf{P}_a)$ computed on the memory $\mathcal{M}_a^{k'}$ from task $k'$, which will be detailed in Section 3.2.

**How to convert to DB style?** In this paper, various forms of structured knowledge can be easily transformed into $\bar{\mathcal{S}}$. For a DB $(\mathcal{T}, \mathcal{C})$, each table name $t_i \in \mathcal{T}$ is mapped to $\phi$, while each column name $c_i \in \mathcal{C}$ is mapped to $\psi$. For a KG subgraph $\{\langle s, p, o\rangle \mid s \in \mathcal{E}, p \in \mathcal{R}, o \in \mathcal{E} \cup \Gamma\}$, each entity type $\gamma \in \Gamma$ and each relation name $t_i \in \mathcal{T}$ are treated as $\phi$, while each relation or property $r \in \mathcal{R}$ and each column name $c_i \in \mathcal{C}$ are treated as $\psi$. For a dialogue state $\{\langle \mathcal{I}_1, s_1\rangle, \ldots, \langle \mathcal{I}_n, s_n\rangle\}$, each intention $\mathcal{I}_i$ is considered as $\phi$, and each slot name $s_i$ is considered as $\psi$. See Appendix A for details.

#### 3.1.2 Task-specific Continual Query Building

Similar to schema filtering, K-DECORE trains another separate PEFT module, denoted as $\mathbf{P}_b$, specifically for the query building phase across all tasks. To avoid error propagation within the pipeline, we ensure that $f(\theta, \mathbf{P}_b)$ also considers the complete schema $\mathcal{S}$ during reasoning, expressed as $\mathcal{Y} = f(\mathcal{Q}, \mathcal{S}^*, \mathcal{S}; \theta, \mathbf{P}_b)$. At this stage, the primary focus of the function $f(\theta, \mathbf{P}_b)$ is to capture the semantics of the problem and the logical and structural features of queries associated with various SKR tasks during reasoning. As shown in Figure 1, the query structures for different SKR tasks are highly task-specific and can vary significantly, which heightens the risk of forgetting at this stage. To address this risk, we include a small number of query samples from previous tasks during training. The following loss function is minimized to optimize $\mathbf{P}_b$:

$$\mathcal{L}(\mathcal{D}^k; \theta, \mathbf{P}_b) = -\sum_{i=1}^{|\mathcal{D}^k|} \sum_{j=1}^{|\mathcal{Y}_i|} \log P(y_j | \mathcal{Q}_i, \mathcal{S}_i^*, \mathcal{S}_i, y_{<j}; \theta, \mathbf{P}_b) + \sum_{k'=1}^{k-1} \mathcal{L}(\mathcal{M}_b^{k'}; \theta, \mathbf{P}_b), \tag{3}$$

where $y_j$ denotes the $j$-th token of the target query $\mathcal{Y}_i$ and $\mathcal{M}_b^{k'}$ represents the memory dedicated to storing structural information of queries, which will be elaborated on in Section 3.2. Notably, here we directly use the raw schema format $\mathcal{S}$ to enable $f(\theta, \mathbf{P}_b)$ to generate the executable queries.

**Algorithm 1** Structure-guided Query Synthesis

**Require:** The training set of the $k$-th task, represented as $\mathcal{D}^k_{\text{train}} = \{(\mathcal{Q}_i, \mathcal{S}_i, \mathcal{Y}_i)\}_{i=1}^{N^k}$; The set of all query structures present in $\mathcal{D}^k_{\text{train}}$ is denoted by $\mathcal{M}_{\mathcal{Y}} = \{\mathcal{Y}^*_1, \dots, \mathcal{Y}^*_{N_1}\}$, while the set of all available schema sets is denoted by $\mathcal{M}_{\mathcal{S}} = \{\mathcal{S}_1, \dots, \mathcal{S}_{N_2}\}$.

$\mathcal{M}^k_{\text{pseudo}} \leftarrow \emptyset$
**for** $|\mathcal{M}^k_{\text{pseudo}}| < \mathcal{N}$ **do**
  $\mathcal{S}^+ \leftarrow \textsc{Random}(\mathcal{M}_{\mathcal{S}}, 1), \mathcal{Y}^* \leftarrow \textsc{Random}(\mathcal{M}_{\mathcal{Y}}, T)$ ▷ Sampling schema and structures.
  $\mathcal{Y}^*_{\text{pseudo}} \leftarrow f(\mathcal{Y}^*; \theta)$ ▷ Synthesize a pseudo query structure.
  $\mathcal{Y}^+ \leftarrow \textsc{FillSchema}(\mathcal{Y}^*_{\text{pseudo}}, \mathcal{S}^+)$ ▷ Fill the schema into the slots of the structure.
  **if** $\textsc{Execute}(\mathcal{Y}^+, \mathcal{S}^+) =$ `"success"` **then**
    $\mathcal{Q}^+ \leftarrow f(\mathcal{Y}^+, \mathcal{S}^+; \theta, \mathbf{P}_c)$ ▷ Generate a pseudo natural language question.
    $\mathcal{M}^k_{\text{pseudo}} \leftarrow \mathcal{M}^k_{\text{pseudo}} \cup \{(\mathcal{Q}^+, \mathcal{S}^+, \mathcal{Y}^+)\}$
  **end if**
**end for**
**return** $\mathcal{M}^k_{\text{pseudo}}$

### 3.2 Dual-perspective Memory Construction

**Schema-guided Memory Sampling.** The goal of this sampling strategy is to construct memory $\mathcal{M}^k_a$ for each task $\mathcal{D}^k$ to relieve $f(\theta, \mathbf{P}_a)$ from the burden of memorizing schema filtering knowledge. Since we have unified the representation of structured knowledge and tasks, the primary gap between different tasks lies in the domain of their schemas. Therefore, it is essential to select samples with **representative schemas**. Formally, for each training sample $\mathcal{X} = (\mathcal{Q}, \mathcal{S}, \mathcal{Y}) \in \mathcal{D}^k_{\text{train}}$, the process begins with the extraction of the relevant schema set $\mathcal{S}^*$ from $\mathcal{Y}$. Subsequently, all samples in $\mathcal{D}^k_{\text{train}}$ are partitioned into $\mathcal{N}$ clusters. The distance between two samples $\mathcal{X}_1$ and $\mathcal{X}_2$ is defined by the cosine similarity $d_1(\mathcal{X}_1, \mathcal{X}_2) = \cos\big(g(\mathcal{S}^*_1), g(\mathcal{S}^*_2)\big)$, where $g$ denotes an encoder-only LLM. Finally, the sample closest to the center of each cluster is selected, and $(\mathcal{Q}, \mathcal{S}, \mathcal{S}^*)$ is added to $\mathcal{M}^k_a$.

**Structure-guided Memory Construction.** This strategy aims to preserve the sample set $\mathcal{M}^k_b$ with diverse query structure features, which are utilized for the rehearsal process of $f(\theta, \mathbf{P}_b)$. The memory $\mathcal{M}^k_b$ consists of two parts, denoted as $\mathcal{M}^k_b = \mathcal{M}^k_{\text{real}} \cup \mathcal{M}^k_{\text{pseudo}}$. First, $\mathcal{M}^k_{\text{real}}$ comprises representative samples selected from $\mathcal{D}^k_{\text{train}}$. This selection process is analogous to the one used for constructing $\mathcal{M}^k_a$, with the main difference being that the sample distance within a cluster is defined by $d_2(\mathcal{X}_1, \mathcal{X}_2) = \cos\big(g(\mathcal{Y}^*_1), g(\mathcal{Y}^*_2)\big)$. Here, $\mathcal{Y}^*$ denotes the structure of the query $\mathcal{Y}$, which is formed by substituting the schema elements with placeholders. The structure of the s-expression in Figure 1 is represented as `(AND [T1] (JOIN [C1] (JOIN ([C1]) [E1])))`, where `[T1]`, `[C1]`, and `[E1]` serve as placeholders for an abstract table, an abstract column, and an entity, respectively.

Unlike existing methods [9, 10] which primarily preserve training-time structures (like $\mathcal{M}^k_{\text{real}}$) seen by $f_\theta$, our K-DeCore employs $\mathcal{M}^k_{\text{pseudo}}$ to boost zero-shot reasoning by introducing novel structures absent from $\mathcal{D}^k_{\text{train}}$. This is achieved through a structure synthesis process, detailed in Algorithm 1. The procedure begins by identifying the repertoire of query structures present in $\mathcal{D}^k_{\text{train}}$, denoted as $\mathcal{M}_{\mathcal{Y}} = \{\mathcal{Y}^*_1, \dots, \mathcal{Y}^*_{N_1}\}$, and the available schema sets, $\mathcal{M}_{\mathcal{S}} = \{\mathcal{S}^*_1, \dots, \mathcal{S}^*_{N_2}\}$. Iteratively, our backbone model $f_\theta$ is leveraged to generate novel structures: $T$ distinct structures $\mathcal{Y}^* \in \mathcal{M}_{\mathcal{Y}}$ are sampled and synthesized into a new structure $\mathcal{Y}^*_{\text{pseudo}}$ guided by carefully curated demonstrations. Here $\mathcal{Y}^*_{\text{pseudo}}$ can be complex SQL queries with nested subqueries or S-expressions featuring multi-step compositional reasoning; detailed examples and prompts of these synthesized structures can be found in the Appendix B.3. This query structure $\mathcal{Y}^*_{\text{pseudo}}$ is then instantiated into a concrete query $\mathcal{Y}_{\text{pseudo}}$ by populating it with a randomly sampled schema $\mathcal{S}^* \in \mathcal{M}_{\mathcal{S}}$. Only generated queries $\mathcal{Y}_{\text{pseudo}}$ that execute correctly, thereby ensuring legality and semantic validity, are retained. Finally, a PEFT module, denoted as $\mathbf{P}_c$, designed to generate the pseudo NLQ $\mathcal{Q}^+$ and is trained by optimizing

$$\mathcal{L}(\mathcal{D}^k; \theta, \mathbf{P}_c) = -\sum_{i=1}^{|\mathcal{D}^k|} \sum_{j=1}^{|\mathcal{Q}_i|} \log P(q_j | \mathcal{Y}, q_{<j}; \theta, \mathbf{P}_c), \tag{4}$$

where $q_j$ denotes the $j$-th token of real $\mathcal{Q}$. Notably, $\mathbf{P}_c^k$ is tailored specifically for each task $\mathcal{D}^k$. Once $\mathcal{M}_b^k$ is constructed, $\mathbf{P}_c^k$ serves as the initialization for training $\mathbf{P}_c^{k+1}$. Since $\mathbf{P}_c^k$ is constructed only once per task, it does not introduce the forgetting problem. To construct $\mathcal{M}_b^k$, we combined the real memory $\mathcal{M}_{\text{real}}^k$ with the synthetic memory $\mathcal{M}_{\text{pseudo}}^k$ in varying proportions. In our experiments, we systematically investigated different synthetic sample percentage to identify the optimal balance.

# 4 Experiments

## 4.1 Experimental Setup

**Datasets.** We conduct experiments on four widely-used SKR datasets, covering DB, KG, DS reasoning. **Spider** [15] is a DB reasoning benchmark where each NLQ is translated into a complex SQL query. These queries often require multi-table `JOIN` operations, `GROUP BY` clauses, and nested subqueries, demanding sophisticated database reasoning capabilities. **ComplexWebQuestions (CWQ)** [16] is a KG reasoning dataset where each NLQ corresponds to an executable SPARQL query. It involves up to 4-hop reasoning over a knowledge graph with complex constraints such as comparisons, aggregations, and nested conditions. **GrailQA** [17] is a KG reasoning benchmark featuring NLQs that require complex multi-hop reasoning to build s-expressions over the Freebase knowledge graph. **MTOP** [18] is designed for multilingual semantic parsing in task-oriented dialogue systems. Each NLQ is parsed into a TOP representation, modeling nested intents and slots.

As detailed in Section 2, we utilize the aforementioned four datasets to construct three distinct sequential task streams, where each task $\mathcal{D}_{\text{train}}$ comprises a single dataset. The input and output spaces vary across different tasks, as illustrated in Figure 1. Additionally, to emulate scenarios with limited training data, each individual task $\mathcal{D}_{\text{train}}^k$ is constrained to $|\mathcal{D}_{\text{train}}^k| = 1000$ and $|\mathcal{D}_{\text{test}}^i| = 300$.

**Evaluation Metrics.** In line with existing studies [13, 12], we employ three metrics to evaluate the performance of the methods: a) $\text{AA}_\text{a} = \frac{1}{K}\sum_{k=1}^{K} acc_{k,K}$, which assesses the overall accuracy across all tasks; b) $\text{BWT} = \frac{1}{K-1}\sum_{k=1}^{K-1}(acc_{k,K} - acc_{k,k})$, which measures the extent of forgetting knowledge from previous tasks; c) $\text{FWT} = \frac{1}{K-1}\sum_{k=2}^{K}(acc_{k,k} - acc_{k,0})$, which evaluates the ability to forward transfer knowledge from past tasks to new ones. Here, $acc_{k,j}$ represents the test accuracy on $\mathcal{D}_{\text{test}}^k$ after training on $\mathcal{D}^j$, and $acc_{k,0}$ denotes the test accuracy on $\mathcal{D}_{\text{test}}^k$ only training on $\mathcal{D}_{\text{train}}^k$.

**Compared Methods.** We conduct a comprehensive comparison against the following three types of baselines: **a) FINE-TUNING**: Represents the performance of a standard, vanilla model without any continual learning enhancements. **b) Rehearsal-based Methods:** Includes methods such as EMAR [19] and SFNET [9], which require replaying real historical examples or utilizing additional unlabeled data. **c) PEFT-based Methods:** Covers techniques like PROGPROMPT[14], O-LORA [11], C3 [12], and SAPT [13]. Additionally, we establish an upper-bound baseline, Multi-Task, where for each task $\mathcal{D}^k$, the model $f_\theta$ is trained jointly on all data of $\mathcal{D}_{\text{train}}^{(0:k)}$.

**Backbone LLMs.** To thoroughly assess the performance across different backbone models, we employed three widely used LLMs of varying sizes: 1) T5-LARGE, 2) LLAMA3-8B-INSTRUCT, and 3) QWEN2.5-7B-INSTRUCT. Most existing methods in the literature are tailored for encoder-decoder architectures, such as T5, which inherently limits their applicability to the current generation of LLMs, the majority of which adopt a decoder-only design. Consequently, when employing LLaMA3 and QWEN2.5 as the backbone, these methods are excluded from consideration due to architectural incompatibility. This distinction underscores the novelty of our approach, as it represents the first systematic exploration of leveraging decoder-only LLMs for the continual SKR task.

**Implementation Details.** Our experiments were conducted using a single NVIDIA RTX 4090 GPU. The hyperparameters were configured as follows (see Appendix C for further details): a) We employed LoRA [20] as the PEFT module. b) The memory size for each task was set to $|\mathcal{M}_a^k| = 5$ and $|\mathcal{M}_b^k| = 5$. c) The ratio of real to pseudo memory samples, $|\mathcal{M}_{\text{real}}^k|$ to $|\mathcal{M}_{\text{pseudo}}^k|$, was maintained at $4:1$. d) We used a batch size of 12 and set the learning rate to $5 \times 10^{-5}$. e) The number of training epochs was fixed at 5. f) $T$ in Algorithm 1 is set to 5. g) The backbone LLM for pseudo samples synthesis is set to QWEN2.5-7B-INSTRUCT. All of our data and codes are publicly available.[2]

[2]https://github.com/SEU-COIN/K-DeCore

Table 1: Experimental results for comparison with baselines.

| Backbone | Method | SKR stream1 | | | SKR stream2 | | | SKR stream3 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | AA | BWT | FWT | AA | BWT | FWT | AA | BWT | FWT |
| T5-LARGE | FINE-TUNING | 2.9 | −31.1 | 6.3 | 10.2 | −24.0 | 7.6 | 6.8 | −23.3 | 4.2 |
| | EMAR [19] | 12.6 | −18.5 | 4.8 | 12.5 | −19.5 | 7.2 | 8.7 | −20.6 | 3.0 |
| | O-LoRA [11] | 1.4 | −30.5 | −2.3 | 7.8 | −17.9 | −5.4 | 5.4 | −27.8 | 0.3 |
| | PROGPROMPT[14] | 9.7 | −23.5 | 1.8 | 7.8 | −17.9 | 4.6 | 15.9 | −19.3 | 1.9 |
| | SFNET [9] | 27.3 | −14.7 | 3.6 | 18.2 | −27.5 | 4.2 | 24.3 | −18.0 | 3.1 |
| | C3 [12] | 26.6 | - | −1.9 | 30.8 | - | 4.2 | 29.8 | - | 3.2 |
| | SAPT [13] | 18.5 | −15.8 | 3.7 | 24.5 | −10.2 | 5.4 | 25.6 | −20.7 | 3.5 |
| | K-DECORE (Ours) | 31.8 | −9.6 | 8.6 | 31.4 | −7.5 | 8.4 | 30.1 | −6.9 | 7.7 |
| | MULTI-TASK | 37.4 | 9.1 | 8.8 | 38.4 | 12.6 | 10.1 | 36.6 | 11.5 | 6.8 |
| LLAMA3 -8B-INSTRUCT | FINE-TUNING | 22.8 | −29.4 | 4.2 | 26.3 | −22.9 | 1.8 | 16.4 | −29.1 | 1.6 |
| | EMAR [19] | 34.0 | −13.2 | 3.3 | 34.5 | −18.4 | 3.7 | 29.8 | −19.9 | 0.5 |
| | SFNET [9] | 36.1 | −14.5 | 3.7 | 35.8 | −17.1 | 4.2 | 35.4 | −12.3 | 4.2 |
| | C3 [12] | 39.7 | - | 2.3 | 40.7 | - | 2.6 | 36.6 | - | 2.1 |
| | K-DECORE (Ours) | 40.5 | −16.7 | 5.9 | 41.1 | −17.1 | 6.1 | 37.0 | −19.3 | 4.2 |
| | MULTI-TASK | 54.5 | 5.7 | 10.0 | 54.7 | 12.8 | 5.1 | 54.3 | 12.2 | 4.8 |
| QWEN2.5 -7B-INSTRUCT | FINE-TUNING | 19.6 | −26.9 | 5.1 | 28.8 | −15.9 | 4.6 | 25.3 | −22.8 | 1.9 |
| | EMAR [19] | 35.6 | −9.2 | 5.4 | 35.4 | −10.8 | 5.3 | 30.7 | −14.8 | 2.0 |
| | SFNET [9] | 40.3 | −10.5 | 4.5 | 38.5 | −13.2 | 4.8 | 34.1 | −15.0 | 5.1 |
| | C3 [12] | 38.9 | - | 1.9 | 38.8 | - | 1.8 | 35.9 | - | 2.6 |
| | K-DECORE (Ours) | 43.2 | −8.2 | 6.9 | 40.1 | −9.8 | 5.5 | 36.8 | −16.7 | 6.9 |
| | MULTI-TASK | 52.0 | 9.2 | 10.4 | 51.8 | 9.9 | 9.7 | 55.0 | 15.6 | 7.3 |

## 4.2 Overall Results

Table 1 presents a comprehensive performance analysis of our proposed K-DECORE framework against several baselines across three continual SKR streams, utilizing various LLM backbones. With T5 as the backbone, K-DECORE uniformly outperforms all competing methods across all evaluation metrics. This superior performance is largely maintained when employing Llama3 and QWEN2.5 as backbones, where K-DECORE establishes new state-of-the-art results in both AA and FWT.

In contrast, while PEFT-based methods such as C3 and SAPT achieve competitive results by dedicating independent parameters to each task, their per-task efficacy is hampered by the inherent convergence limitations of prompt-tuning. K-DECORE, however, attains superior performance without introducing additional parameters, offering a more efficient and scalable solution through its novel knowledge decoupling mechanism. Furthermore, our synthetic sample generation strategy, by creating more diverse query structures, enhances the model's generalization capabilities, as evidenced by its strong FWT performance. Notably, since C3 trains and loads a distinct PEFT module for each task, thereby precluding knowledge transfer to previous tasks, we do not report its BWT.

## 4.3 Ablation Study

To explore the contributions of each component of our proposed K-DECORE, we compared the performance of the following settings: **a) w/o Decoupling**: We omit the *knowledge decoupling* process and instead treat SKR as a standalone stage. In this configuration, we use a single model $f_\theta$ equipped with one PEFT module $\mathbf{P}$ for inference, while maintaining the replay process unchanged. **b) w/o Unification**: To evaluate the contribution of the *unified schema representation*, we only used the original schema format specific to each SKR task during the schema filtering stage. **c) w/o Replay**: We employ continual knowledge decoupling without replaying any samples, which involves removing the second term in both Equations (2) and (3). **d) w/o** $\mathcal{M}_a^k$: We removed the memory $\mathcal{M}_a^k$ in the continual schema filtering. **e) w/o** $\mathcal{M}_b^k$: We removed the memory $\mathcal{M}_b^k$ in the continual query building. **f) w Random Memory**: We randomly sample from $\mathcal{D}_a^k$ to construct $\mathcal{M}_a^k$ and $\mathcal{M}_b^k$.

Table 2 presents the results of our ablation study, investigating the contribution of key components of K-DECORE. Removing the entire replay mechanism or specifically the query structure memory (**w/o**

Table 2: Experimental results of ablation studies.

| Backbone | Method | SKR stream1 | | | SKR stream2 | | | SKR stream3 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | AA | BWT | FWT | AA | BWT | FWT | AA | BWT | FWT |
| LLAMA3-8B-INSTRUCT | K-DECORE | 40.5 | −16.7 | 5.9 | 41.1 | −17.1 | 6.1 | 37.0 | −19.3 | 4.2 |
| | w/o Decoupling | 38.8 | −13.8 | 2.6 | 37.5 | −19.8 | 2.7 | 33.6 | −19.6 | 2.1 |
| | w/o Unification | 37.8 | −17.7 | 3.4 | 37.1 | −15.2 | 0.7 | 34.6 | −15.7 | −0.8 |
| | w/o Replay | 20.5 | −41.2 | 4.3 | 28.8 | −34.4 | 6.8 | 17.9 | −43.8 | 3.5 |
| | w/o $\mathcal{M}_a^k$ | 39.4 | −17.1 | 5.1 | 38.2 | −20.9 | 6.2 | 35.6 | −20.2 | 3.6 |
| | w/o $\mathcal{M}_b^k$ | 20.8 | −42.1 | 5.3 | 28.6 | −32.8 | 5.3 | 18.5 | −44.5 | 3.7 |
| | w Random Memory | 39.9 | −17.4 | 5.6 | 37.7 | −20.6 | 5.4 | 36.3 | −21.3 | 3.9 |
| QWEN2.5-7B-INSTRUCT | K-DECORE | 43.2 | −8.2 | 6.9 | 40.1 | −9.8 | 5.5 | 36.8 | −16.7 | 6.9 |
| | w/o Decoupling | 37.9 | −8.8 | 5.4 | 39.3 | −5.5 | 4.0 | 36.1 | −17.1 | 4.8 |
| | w/o Unification | 34.4 | −13.3 | 1.8 | 37.4 | −10.5 | 2.6 | 36.1 | −11.7 | 2.4 |
| | w/o Replay | 16.8 | −41.4 | 5.3 | 29.7 | −23.8 | 5.5 | 18.7 | −40.3 | 6.5 |
| | w/o $\mathcal{M}_a^k$ | 42.8 | −9.7 | 5.3 | 39.5 | −9.8 | 4.9 | 35.9 | −17.5 | 6.6 |
| | w/o $\mathcal{M}_b^k$ | 19.1 | −39.2 | 6.0 | 28.3 | −24.4 | 4.7 | 16.3 | −43.8 | 6.8 |
| | w Random Memory | 42.1 | −7.7 | 5.3 | 39.4 | −11.8 | 6.3 | 36.2 | −17.9 | 5.7 |

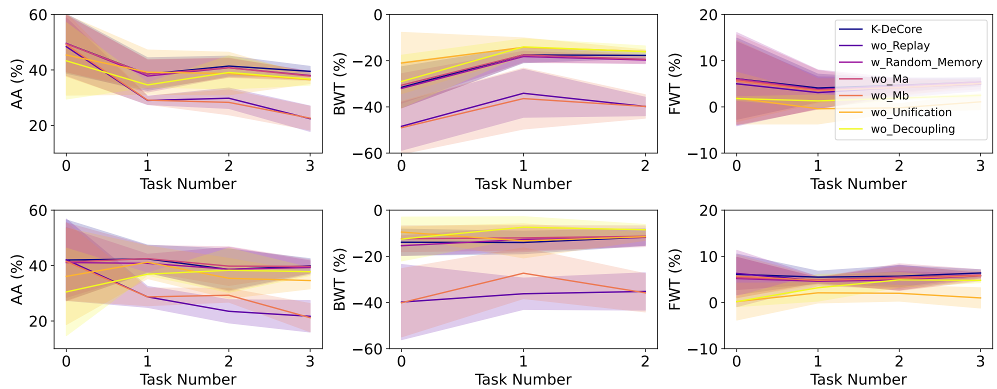


Figure 3: AA (%), BWT (%), and FWT (%) till the seen tasks after learning on each task, using LLAMA3-8B (top row) and QWEN2.5-8B (bottom row). Solid lines represent the mean values across three distinct task sequences, while shaded regions indicate the standard deviation.

$\mathcal{M}_b^k$) leads to drastic performance drops in AA and significantly worse BWT, highlighting the critical role of memory, particularly the storage of diverse query structures, in mitigating forgetting and maintaining overall performance. Disabling the knowledge decoupling (**w/o Decoupling**) or removing the unified schema representation (**w/o Unification**) also results in lower AA and reduced FWT, demonstrating that separating reasoning into distinct stages and standardizing schema representation are vital for effective knowledge transfer and handling heterogeneous tasks. While removing the schema memory (**w/o** $\mathcal{M}_a^k$) or using random memory sampling shows less severe degradation compared to removing query memory, performance still drops relative to the full model, confirming the benefits of dedicated schema memory and our proposed memory construction strategy.

### 4.4 Performance Till the Seen Tasks

Figure 3 presents the performance trajectory of the evaluated methods on previously seen tasks across different LLM backbones. Our proposed K-DECORE framework consistently demonstrates superior overall performance, with its advantage becoming more pronounced as the number of sequential tasks increases. The model's robust forward transfer capabilities, a key contributor to this success, can be attributed to our strategies for pseudo-sample synthesis and continuous query construction, which collectively enhance its adaptability to new and unseen scenarios. An interesting trade-off is revealed in our ablation study. While omitting the knowledge decoupling component results in a higher BWT, this apparent benefit comes at the cost of increased inter-task interference. This interference ultimately diminishes the model's adaptability, leading to poorer performance in FWT and overall AA, thereby underscoring the critical role of our decoupling strategy.

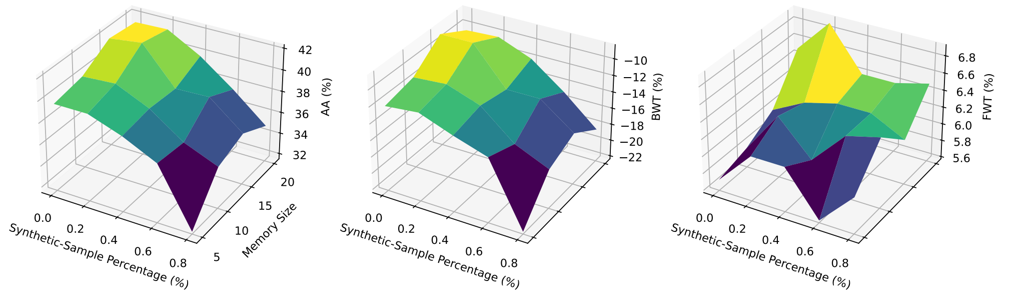


Figure 4: Peformance of K-DECORE with varing memory sizes and synthetic sample percentages.

Table 3: Experimental results on imbalanced task streams.

| Backbone | Method | SKR stream1 | | | SKR stream2 | | | SKR stream3 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | AA | BWT | FWT | AA | BWT | FWT | AA | BWT | FWT |
| QWEN2.5 -7B-INSTRUCT | FINE-TUNING | 18.5 | −15.2 | 4.1 | 22.2 | −11.8 | 4.8 | 26.3 | −5.1 | 3.4 |
| | EMAR [19] | 28.2 | −3.1 | 5.4 | 30.7 | −1.1 | 5.1 | 28.3 | −1.6 | 3.2 |
| | K-DECORE (Ours) | 31.8 | −10.9 | 8.5 | 33.5 | −8.5 | 7.3 | 30.1 | −6.5 | 6.2 |

## 4.5 Effects of Varying Memory Sizes and Synthetic Sample Percentages

Figure 4 shows the AA, BWT, and FWT of various methods across seen tasks using different backbone LLMs. When the proportion of pseudo samples is relatively low, increasing the memory size tends to enhance AA and BWT. Conversely, with a fixed memory size, incorporating pseudo samples at a proportion of 20% frequently yields optimal results in terms of AA, BWT, and FWT, outperforming scenarios with either no pseudo samples or a higher proportion of them. Interestingly, for FWT, a larger memory capacity does not inherently translate to improved generalization capabilities.

## 4.6 Performance on Imbalanced Task Streams

To further assess our model's robustness in more realistic scenarios, we conducted an additional experiment simulating an imbalanced data stream, a common challenge where tasks have a non-uniform number of training samples. Specifically, we configured the training set with the following sample distribution: Spider (500), GrailQA (800), CombWebQ (300), and MTOP (600). Here we use QWEN2.5-7B-INSTRUCT as the backbone model. As shown in Table 3, our method, KE-DECORE, demonstrates clear superiority in the imbalanced data stream experiment. It achieves the highest AA on all the streams, significantly outperforming the EMAR baseline.

## 4.7 Training Efficiency

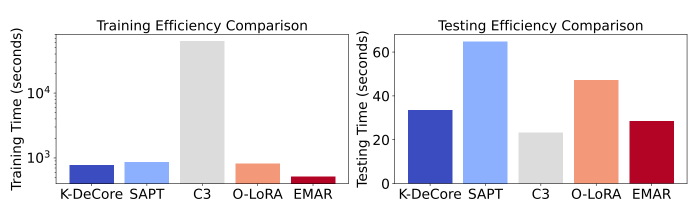


Figure 5: Training and testing time of various methods.

Figure 5 illustrates the average training duration for each approach across various tasks. Notably, C3 exhibited a considerably longer training time compared to other methods, primarily due to its prompt tuning necessitating an extensive number of epochs. The primary source of time overhead for SAPT and O-LoRA lies in the merging process of multiple LoRA modules. This bottleneck becomes increasingly pronounced as the number of tasks scales up. In contrast, our K-DECORE demonstrated a favorable balance between efficiency and performance, with its training time only marginally exceeding that of the rehearsal-based EMAR.

## 5 Related Work

### 5.1 Structured Knowledege Reasoning.

Structured Knowledge-intensive Retrieval (SKR) tasks, which include text-to-SQL [21], Knowledge Graph Question Answering (KGQA), and dialogue parsing, are centered on converting natural language questions (NLQs) into structured, executable queries. In the database domain (DB SKR), the process of translating NLQs into SQL queries has advanced significantly. Initial methods relied on specialized model architectures [22] and the use of intermediate representations [23]. More recently, the field has shifted towards leveraging Large Language Models (LLMs) enhanced with sophisticated techniques such as task decomposition, chain-of-thought prompting [24, 25], and self-consistency [26]. This modern approach has led to substantial performance gains, as demonstrated in several recent studies [27, 28, 29, 30]. Similarly, SKR over knowledge graphs (KG SKR), or KGQA, has evolved from traditional semantic parsing [31, 32] and embedding-based methods [33]. Recent advancements, such as DecAF [34] and KB-BINDER [35], now integrate the generation of logical forms with direct answer retrieval, markedly improving performance. In the domain of dialogue parsing, tasks like TOP [36] and MTOP [18] involve deciphering complex, multi-turn interactions. These have also been advanced by LLMs, which provide a more nuanced understanding of the required structured representations.

While APEX [10] and LECSP [37] also constitute related work, they were excluded from our comparative analysis. These methods are specifically tailored for the text-to-SQL task and do not generalize to the other SKR domains we address. Furthermore, their computational overhead when implemented with LLM backbones exceeds the practical limits of our experimental setup.

### 5.2 LLM Continual Learning.

In the context of continual learning for LLMs [38], Parameter-Efficient Fine-Tuning (PEFT) methods, including prompt tuning [39], adapters [40], and LoRA [20], have become instrumental. These techniques are particularly effective for adapting models to new tasks with limited data [41, 10] by introducing a small number of trainable parameters while keeping the base model frozen. This parameter-efficient approach allows for the sequential integration of new knowledge, mitigating catastrophic forgetting and adapting to distribution shifts without the need for replaying historical data [12, 13]. A common strategy in continual learning is to train a separate PEFT module for each new task. This compartmentalizes task-specific knowledge, effectively preventing interference between tasks [42, 13, 43].

However, this one-to-one mapping of modules to tasks presents a significant drawback: it inherently struggles with generalization to novel samples that do not neatly fit into the learned task distributions, thereby limiting its practical utility in dynamic, real-world scenarios. In stark contrast, our proposed K-DECORE framework utilizes a fixed set of PEFT modules throughout the entire learning process. This design offers a more efficient and scalable solution for both training and inference in evolving, iterative environments.

## 6 Conclusion

In this paper, we introduced K-DECORE, a novel framework for Continual Structured Knowledge Reasoning (CSKR) that effectively addresses the challenges of catastrophic forgetting and parameter growth in heterogeneous structured knowledge environments. By leveraging knowledge decoupling, K-DECORE decouples schema filtering from query construction, enabling efficient knowledge transfer across diverse tasks without increasing model parameters. Our dual-perspective memory mechanism further enhances knowledge retention by maintaining replay samples from both schema and query structure perspectives, while our query synthesis strategy enriches memory content with complex structured queries. Through extensive evaluations on task streams from four SKR benchmarks, K-DECORE consistently demonstrated superior performance over strong baselines across multiple metrics. However, due to time and cost constraints, we did not employ reasoning-based LLMs (like QWEN3) as the backbone for our framework. This remains a limitation of our current work. Future research will explore the integration of such models to potentially enhance the applicability and robustness of K-DECORE in real-world scenarios.

## 7 Acknowledgements

This work was partially funded by National Nature Science Foundation of China (Grant No. U21A20488, No. 62376058, No. 62476058, and No. 6250072425), and funded by Southeast University-China Mobile Research Institute Joint Innovation Center and the Southeast University Interdisciplinary Research Program for Young Scholars. We thank the Big Data Computing Center of Southeast University for providing the facility support on the numerical calculations in this paper.

## References

[1] Junyun Cui, Xiaoyu Shen, and Shaochun Wen. A survey on legal judgment prediction: Datasets, metrics, models and challenges. *IEEE Access*, 11:102050–102071, 2023.

[2] Linfeng Li, Peng Wang, Jun Yan, Yao Wang, Simin Li, Jinpeng Jiang, Zhe Sun, Buzhou Tang, Tsung-Hui Chang, Shenghui Wang, and Yuting Liu. Real-world data medical knowledge graph: construction and applications. *Artif. Intell. Medicine*, 103:101817, 2020.

[3] Sabbir M. Rashid and Deborah L. McGuinness. Designing and evaluating an ensemble reasoning-based clinical decision support system. *Data Intell.*, 7(1):1–39, 2025.

[4] Chao Zhang, Yuren Mao, Yijiang Fan, Yu Mi, Yunjun Gao, Lu Chen, Dongfang Lou, and Jinshu Lin. Finsql: Model-agnostic llms-based text-to-sql framework for financial analysis. In Pablo Barceló, Nayat Sánchez-Pi, Alexandra Meliou, and S. Sudarshan, editors, *Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago AA, Chile, June 9-15, 2024*, pages 93–105. ACM, 2024.

[5] Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete, editors, *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 174–184. ACM, 2023.

[6] Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. Codes: Towards building open-source language models for text-to-sql. *Proc. ACM Manag. Data*, 2(3):127, 2024.

[7] Zhijie Nie, Richong Zhang, Zhongyuan Wang, and Xudong Liu. Code-style in-context learning for knowledge-based question answering. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 18833–18841. AAAI Press, 2024.

[8] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.

[9] Yongrui Chen, Xinnan Guo, Tongtong Wu, Guilin Qi, Yang Li, and Yang Dong. Learn from yesterday: A semi-supervised continual learning method for supervision-limited text-to-sql task streams. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 12682–12690. AAAI Press, 2023.

[10] Ruiheng Liu, Jinyu Zhang, Yanqi Song, Yu Zhang, and Bailong Yang. Discarding the crutches: Adaptive parameter-efficient expert meta-learning for continual semantic parsing. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 3560–3578. Association for Computational Linguistics, 2025.

[11] Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 10658–10671. Association for Computational Linguistics, 2023.

[12] Yongrui Chen, Shenyu Zhang, Guilin Qi, and Xinnan Guo. Parameterizing context: Unleashing the power of parameter-efficient fine-tuning and in-context tuning for continual table semantic parsing. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[13] Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. SAPT: A shared attention framework for parameter-efficient continual learning of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 11641–11661. Association for Computational Linguistics, 2024.

[14] Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[15] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics, 2018.

[16] Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 641–651. Association for Computational Linguistics, 2018.

[17] Yu Gu, Sue Kase, Michelle Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond I.I.D.: three levels of generalization for question answering on knowledge bases. In Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia, editors, *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 3477–3488. ACM / IW3C2, 2021.

[18] Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 2950–2962. Association for Computational Linguistics, 2021.

[19] Xu Han, Yi Dai, Tianyu Gao, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Continual relation learning via episodic memory activation and reconsolidation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 6429–6440, 2020.

[20] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[21] Shuqin Li, Kaibin Zhou, Zeyang Zhuang, Haofen Wang, and Jun Ma. Towards text-to-sql over aggregate tables. *Data Intell.*, 5(2):457–474, 2023.

[22] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. Grappa: Grammar-augmented pre-training for table semantic parsing. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[23] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-sql in cross-domain database with intermediate representation. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4524–4535. Association for Computational Linguistics, 2019.

[24] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[25] Songlin Chen, Weicheng Wang, Xiaoliang Chen, Peng Lu, Zaiyan Yang, and Yajun Du. Llama-lora neural prompt engineering: A deep tuning framework for automatically generating chinese text logical reasoning thinking chains. *Data Intell.*, 6(2):375–408, 2024.

[26] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[27] Mohammadreza Pourreza and Davood Rafiei. DIN-SQL: decomposed in-context learning of text-to-sql with self-correction. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

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

[29] Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. CHESS: contextual harnessing for efficient SQL synthesis. *CoRR*, abs/2405.16755, 2024.

[30] Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan Ö. Arik. CHASE-SQL: multi-path reasoning and preference optimized candidate selection in text-to-sql. *CoRR*, abs/2410.01943, 2024.

[31] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL, 2013.

[32] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1321–1331. The Association for Computer Linguistics, 2015.

[33] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[34] Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[35] Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. Few-shot in-context learning on knowledge base question answering. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 6966–6980. Association for Computational Linguistics, 2023.

[36] Arash Einolghozati, Panupong Pasupat, Sonal Gupta, Rushin Shah, Mrinal Mohit, Mike Lewis, and Luke Zettlemoyer. Improving semantic parsing for task oriented dialog. *CoRR*, abs/1902.06000, 2019.

[37] Ruiheng Liu, Jinyu Zhang, Yanqi Song, Yu Zhang, and Bailong Yang. Filling memory gaps: Enhancing continual semantic parsing via SQL syntax variance-guided llms without real data replay. In Toby Walsh, Julie Shah, and Zico Kolter, editors, *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 24641–24649. AAAI Press, 2025.

[38] Huajun Chen. Large knowledge model: Perspectives and challenges. *Data Intell.*, 6(3):587–620, 2024.

[39] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[40] Clifton Poth, Hannah Sterz, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan Vulić, Sebastian Ruder, Iryna Gurevych, and Jonas Pfeiffer. Adapters: A unified library for parameter-efficient and modular transfer learning. *arXiv preprint arXiv:2311.11077*, 2023.

[41] Huanxuan Liao, Shizhu He, Yao Xu, Yuanzhe Zhang, Yanchao Hao, Shengping Liu, Kang Liu, and Jun Zhao. From instance training to instruction learning: Task adapters generation from instructions. *Advances in Neural Information Processing Systems*, 37:45552–45577, 2024.

[42] Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. *arXiv preprint arXiv:2301.12314*, 2023.

[43] Martin Menabue, Emanuele Frascaroli, Matteo Boschini, Enver Sangineto, Lorenzo Bonicelli, Angelo Porrello, and Simone Calderara. Semantic residual prompts for continual learning. In *European Conference on Computer Vision*, pages 1–18. Springer, 2024.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We propose a novel CSKR framework, K-DeCore, which facilitates the knowledge transfer using knowledge decoupling.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed this in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: There is no theoretical analysis in our paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The implementation details of all experiments are detailed in Section 4.1 and the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide open access to data and code and provide adequate explanations.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The implementation details of all experiments are detailed in Section 4.1 and the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the mean and standard deviation of the experimental results for different orders in Section 4.4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computer resources required to reproduce the experiments are provided in Section 4.1 and Section 4.6 (Efficiency).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research complies with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our work has no social impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The method in this article is a training mechanism, not a specific language model, so it does not involve this risk.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing datasets and evaluation metrics we use are clearly referenced.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The datasets used and processed in our experiments and the corresponding documentation will be publicly released.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not have crowdsourced experiments and research on humans.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We don't have this risk.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: Our use of LLM does not compromise the core methodology, scientific rigor or originality of the research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

# A DB-style Input and Output Format

We emphasize that our DB-style unification is designed to fully preserve all critical information, ensuring that the flattened representation remains faithful to the original structures. Below, we explain how our design achieves this preservation while enabling effective covering across tasks.

For relation direction, our mapping explicitly preserves the subject-object orientation inherent in triples (subject-predicate-object). Specifically, we designate the content in the primary key of the transformed table as the subject, while the contents in other columns represent the corresponding objects. This structured encoding ensures that directional information is retained and can be reliably interpreted during query generation, preventing ambiguities that might arise from undirected representations.

Regarding edge types, we incorporate them directly into the schema by prefixing each relation (or edge) with its type and treating it as a dedicated column in the flattened table. This simple yet effective augmentation embeds the type information within the schema itself, allowing the model to access and utilize it without requiring additional modifications to the backbone architecture.

For multi-level structures, such as hierarchical slots or nested tables, we adopt flattening techniques. This involves recursively expanding multi-level columns into single-level ones, ensuring that hierarchical relationships are linearized but not discarded. Our framework is flexible enough to accommodate this, as the flattening occurs preprocessing and does not alter the core continual learning mechanism.

More crucially, our DB-style unification is applied only during the schema filtering stage, where the primary objective is akin to entity linking—namely, identifying the relevant tables (entity types) and columns (relations) for the given query. In this context, not all granular details (e.g., full hierarchical depth) are necessary, as the filter's role is to select pertinent schema elements rather than reconstruct the entire original structure. The subsequent query builder module, operating on the filtered schema, can then draw on the preserved essential information.

## A.1 GrailQA

**Schema Filtering Stage:**

```
INPUT:
Identify the exact schema component from the given schema that would correctly answer the
↪  following question.

schema:

book.series_editor : book_edition_series_edited | book.book_edition : book_edition_series,
↪  place_of_publication | book.book_edition_series : editions_in_this_series, series_editor

question:

a people's history of christianity was edited by what series editor?


OUTPUT:
book.series_editor : book_edition_series_edited
```

**Query Building Stage:**

```
INPUT:
Generate an s-expression that can be used to find the answer to the following question using the
↪  knowledge graph schema items provided.

schema:

a people's history of christianity: m.012bphrj | book.series_editor : book_edition_series_edited |
↪  book.book_edition : book_edition_series, place_of_publication | book.book_edition_series :
↪  editions_in_this_series, series_editor


question:

a people's history of christianity was edited by what series editor?


OUTPUT:
(AND book.series_editor (JOIN book.series_editor.book_edition_series_edited m.012bphrj))
```

### A.2 Spider

#### Schema Filtering Stage:

```
INPUT:
Identify the exact schema component from the given schema that would correctly answer the
↪  following question.

schema:

department : department_id , name , creation , ranking , budget_in_billions , num_employees | head
↪  : head_id , name , born_state , age | management : department_id , head_id , temporary_acting


question:

How many heads of the departments are older than 56 ?


OUTPUT:
head : age
```

#### Query Building Stage:

```
INPUT:
Your task is to convert the following question to an SQL query using the following database
↪  schema.

schema:
department : department_id , name , creation , ranking , budget_in_billions , num_employees | head
↪  : head_id , name , born_state , age | management : department_id , head_id , temporary_acting

question:
How many heads of the departments are older than 56 ?

OUTPUT:
select count(*) from head where age > 56
```

### A.3 CWQ

#### Schema Filtering Stage:

```
INPUT:
Identify the exact schema component from the given schema that would correctly answer the
↪  following question.

schema:

type.object : type | location.location : time_zones, containedby, contains, people_born_here |
↪  common.topic : notable_types | geography.river : mouth | travel.tourist_attraction :
↪  near_travel_destination | kg.object_profile : prominent_type |
↪  education.educational_institution : colors | location.country : languages_spoken,
↪  administrative_divisions

question:What language is spoken in the country that has Southern Peninsular?

OUTPUT:
location.country : languages_spoken, administrative_divisions
```

### Query Building Stage:

```
INPUT:
Given the list of schema items, write an SPARQL query that can be used to find the answer to the
↪  following question.

schema:
Southern Peninsula: m.08kmfj | type.object : type | location.location : time_zones, containedby,
↪  contains, people_born_here | common.topic : notable_types | geography.river : mouth |
↪  travel.tourist_attraction : near_travel_destination | kg.object_profile : prominent_type |
↪  education.educational_institution : colors | location.country : languages_spoken,
↪  administrative_divisions

question:
What language is spoken in the country that has Southern Peninsular?


OUTPUT:
PREFIX ns: <http://rdf.freebase.com/ns/>SELECT DISTINCT ?xWHERE {?c
↪  ns:location.country.administrative_divisions ns:m.08kmfj .?c
↪  ns:location.country.languages_spoken ?x .}
```

## A.4 MTOP

### Schema Filtering Stage:

```
INPUT:

Identify the exact schema component from the given schema that would correctly answer the
↪  following question.

schema:
in:get : message, weather, alarm, info_recipes, stories_news, reminder, recipes, event, call_time,
↪  life_event, info_contact, contact, timer, reminder_date_time, age, sunrise, employer,
↪  education_time, job, availability, category_event, call, employment_time, call_contact,
↪  location, track_info_music, sunset, mutual_friends, undergrad, reminder_location,
↪  attendee_event, message_contact, reminder_amount, date_time_event, details_news,
↪  education_degree, major, contact_method, life_event_time, lyrics_music, airquality, language,
↪  gender, group | in:send_message | in:set : unavailable, rsvp_yes, available,
↪  default_provider_music, rsvp_interested, default_provider_calling, rsvp_no | in:delete :
↪  reminder, alarm, timer, playlist_music | in:create : alarm, reminder, call, playlist_music,
↪  timer | in:question : news, music | in:play : music, media | in:end_call | in:ignore_call |
↪  in:update_call | in:update_reminder_date_time | in:pause : music, timer | in:answer_call |
↪  in:snooze_alarm | in:update_reminder_todo | in:is_true_recipes | in:remove_from_playlist_music
↪  | in:add : time_timer, to_playlist_music | in:share_event | in:prefer | in:start_shuffle_music
↪  | in:silence_alarm | in:switch_call | in:subtract_time_timer | in:update_timer |
↪  in:previous_track_music | in:hold_call | in:skip_track_music | in:update_method_call |
↪  in:update_alarm | in:like_music | in:restart_timer | in:resume : timer, call, music |
↪  in:merge_call | in:replay_music | in:loop_music | in:stop : music, shuffle_music |
↪  in:unloop_music | in:update_reminder_location | in:cancel : message, call | in:update_reminder
↪  | in:rewind_music | in:repeat : all_music, all_off_music | in:fast_forward_music |
↪  in:dislike_music | in:disprefer | in:help_reminder | in:follow_music

question:
Has Angelika Kratzer video messaged me ?

OUTPUT:

IN:GET : MESSAGE
```

### Query Building Stage:

```
INPUT:
Convert the following natural language query to an API call in Task Oriented Parsing (TOP)
↪  representation using the following API specification.

API specification:
IN:GET : MESSAGE, WEATHER, ALARM, INFO_RECIPES, STORIES_NEWS, REMINDER, RECIPES, EVENT, CALL_TIME,
↪  LIFE_EVENT, INFO_CONTACT, CONTACT, TIMER, REMINDER_DATE_TIME, AGE, SUNRISE, EMPLOYER,
↪  EDUCATION_TIME, JOB, AVAILABILITY, CATEGORY_EVENT, CALL, EMPLOYMENT_TIME, CALL_CONTACT,
↪  LOCATION, TRACK_INFO_MUSIC, SUNSET, MUTUAL_FRIENDS, UNDERGRAD, REMINDER_LOCATION,
↪  ATTENDEE_EVENT, MESSAGE_CONTACT, REMINDER_AMOUNT, DATE_TIME_EVENT, DETAILS_NEWS,
↪  EDUCATION_DEGREE, MAJOR, CONTACT_METHOD, LIFE_EVENT_TIME, LYRICS_MUSIC, AIRQUALITY, LANGUAGE,
↪  GENDER, GROUP | IN:SEND_MESSAGE | IN:SET : UNAVAILABLE, RSVP_YES, AVAILABLE,
↪  DEFAULT_PROVIDER_MUSIC, RSVP_INTERESTED, DEFAULT_PROVIDER_CALLING, RSVP_NO | IN:DELETE :
↪  REMINDER, ALARM, TIMER, PLAYLIST_MUSIC | IN:CREATE : ALARM, REMINDER, CALL, PLAYLIST_MUSIC,
↪  TIMER | IN:QUESTION : NEWS, MUSIC | IN:PLAY : MUSIC, MEDIA | IN:END_CALL | IN:IGNORE_CALL |
↪  IN:UPDATE_CALL | IN:UPDATE_REMINDER_DATE_TIME | IN:PAUSE : MUSIC, TIMER | IN:ANSWER_CALL |
↪  IN:SNOOZE_ALARM | IN:UPDATE_REMINDER_TODO | IN:IS_TRUE_RECIPES | IN:REMOVE_FROM_PLAYLIST_MUSIC
↪  | IN:ADD : TIME_TIMER, TO_PLAYLIST_MUSIC | IN:SHARE_EVENT | IN:PREFER | IN:START_SHUFFLE_MUSIC
↪  | IN:SILENCE_ALARM | IN:SWITCH_CALL | IN:SUBTRACT_TIME_TIMER | IN:UPDATE_TIMER |
↪  IN:PREVIOUS_TRACK_MUSIC | IN:HOLD_CALL | IN:SKIP_TRACK_MUSIC | IN:UPDATE_METHOD_CALL |
↪  IN:UPDATE_ALARM | IN:LIKE_MUSIC | IN:RESTART_TIMER | IN:RESUME : TIMER, CALL, MUSIC |
↪  IN:MERGE_CALL | IN:REPLAY_MUSIC | IN:LOOP_MUSIC | IN:STOP : MUSIC, SHUFFLE_MUSIC |
↪  IN:UNLOOP_MUSIC | IN:UPDATE_REMINDER_LOCATION | IN:CANCEL : MESSAGE, CALL | IN:UPDATE_REMINDER
↪  | IN:REWIND_MUSIC | IN:REPEAT : ALL_MUSIC, ALL_OFF_MUSIC | IN:FAST_FORWARD_MUSIC |
↪  IN:DISLIKE_MUSIC | IN:DISPREFER | IN:HELP_REMINDER | IN:FOLLOW_MUSIC

question:
Has Angelika Kratzer video messaged me ?

OUTPUT:

[IN:GET_MESSAGE [SL:CONTACT Angelika Kratzer ] [SL:TYPE_CONTENT video ] [SL:RECIPIENT me ] ]
```

# B Prompts for Synthesizing Query Structures

## B.1 Prompt for Pseudo Structure Synthesis

```
You are an expert in logical query expression synthesis. Your task is to merge two simple query
↪  skeletons into a single, more complex skeleton that logically integrates their structure and
↪  meaning. The result should preserve the semantics of both inputs and follow the same
↪  structural style and syntax. Output only the composed skeleton. Do not include any explanation
↪  or additional text.

Simple skeleton 1:\{example1\}

Simple skeleton 2:\{example2\}

Composed skeleton:
```

## B.2 Prompt for Pseudo Question Generation

```
You are an AI assistant that specializes in transforming structured query statements into fluent,
↪  contextually accurate natural language questions. Your task is to read a given formal query
↪  and its associated schema context, and then write a corresponding question in clear, natural
↪  language that would retrieve the correct answer if the formal query were executed on a
↪  database.

question:
{query}

schema:
{schema}

Follow these steps carefully:

1. Understand the Query Semantics:
Analyze the query logic, including the target variables, the filtering conditions, the joins, and
↪  the structure of the query. Infer what specific information the query is intended to retrieve.

2. Incorporate Schema Semantics:
Use the provided schema elements (such as table names, entity types, or relation labels) to guide
↪  the terminology and phrasing in the question. Map schema components to human-interpretable
↪  phrases as necessary.

3. Avoid Direct Translation:
Do not simply convert the query structure into a rigid, mechanical form. Instead, aim for a fluent,
↪  natural-sounding question that a human might ask when seeking the same information.

4. Preserve Answer Equivalence:
Ensure that the generated question is logically equivalent to the formal query in terms of the
↪  expected answer. The question should not broaden, narrow, or alter the scope of the query.

5. Maintain Clarity and Brevity:
The question should be unambiguous and concise, while preserving all the critical information
↪  needed to align with the query.

6. Output Format Constraint:
Only output the final question, without any explanatory text, metadata, or formatting symbols. The
↪  output should consist solely of a single fluent question in English.
```

## B.3 Examples of Structure Synthesis

```
Structure 1:
(ARGMAX [T1] [C1])

Structure 2:
(AND [T1] (JOIN [C1] [E1]))

Synthetic Structure:
(ARGMAX (AND [T1] (JOIN [C1] [E1])) [C2])
```

```
Structure 1:
SELECT [C1] FROM [T1] WHERE [C2] = [V1];

Structure 2:
SELECT [C1] FROM [T] WHERE [C3] IN [V1];

Synthetic Structure:
SELECT * FROM [T1] WHERE [C1] IN (SELECT [C2] FROM [T2] WHERE [C3] = [V1]);
```

# C Hyperparameter settings

Our hyperparameter settings are as follows:

Table 4: Hyperparameter Configuration for Experiments

| Hyperparameter | Value |
|---|---|
| **GPU** | NVIDIA RTX 4090 |
| **PEFT Module** | LoRA [20] |
| **LoRA Target** | gate_proj, down_proj, up_proj, q_proj, v_proj, k_proj, o_proj |
| **LoRA Rank** | 8 |
| **LoRA $\alpha$** | 16 |
| **Memory Size per Task** | $\lvert\mathcal{M}_a^k\rvert = 5$, $\lvert\mathcal{M}_b^k\rvert = 5$ |
| **Real to Pseudo Memory Ratio** | $4:1$ |
| **Batch Size** | 12 |
| **Global Batch Size** | 32 |
| **Learning Rate** | $5 \times 10^{-5}$ |
| **Optimizer** | Adam |
| **Training Epochs** | 5 |
| **Maximum Input Length** | 1024 |
| **Maximum Output Length** | 256 |
| **Parameter $T$ in Algorithm 1** | 5 |