

Plugging Schema Graph into Multi-Table QA: A Human-Guided Framework for Reducing LLM Reliance

Downloaded from: https://research.chalmers.se, 2026-11-06 02:35 UTC

Citation for the original published paper (version of record):

Wang, X., Nobre da Costa, M., Kovaceva, J. et al (2025). Plugging Schema Graph into Multi-Table QA: A Human-Guided Framework for Reducing LLM Reliance. Emnlp 2025 2025 Conference on Empirical Methods in Natural Language Processing Findings of Emnlp 2025: 5829-5842. http://dx.doi.org/10.18653/v1/2025.findings-emnlp.311

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004. research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

Plugging Schema Graph into Multi-Table QA: A Human-Guided Framework for Reducing LLM Reliance

Xixi Wang[1] **Miguel Costa**[1] **Jordanka Kovaceva**[2]
Shuai Wang[2] **Francisco C. Pereira**[1]

[1]Technical University of Denmark, Kgs. Lyngby, Denmark
[2]Chalmers University of Technology, Gothenburg, Sweden
s232253@dtu.dk, migcos@dtu.dk, jordanka.kovaceva@chalmers.se,
shuaiwa@chalmers.se, camara@dtu.dk

Abstract

Large language models (LLMs) have shown promise in table Question Answering (Table QA). However, extending these capabilities to multi-table QA remains challenging due to unreliable schema linking across complex tables. Existing methods based on semantic similarity work well only on simplified hand-crafted datasets and struggle to handle complex, real-world scenarios with numerous and diverse columns. To address this, we propose a graph-based framework that leverages human-curated relational knowledge to explicitly encode schema links and join paths. Given a natural language query, our method searches on graph to construct interpretable reasoning chains, aided by pruning and sub-path merging strategies to enhance efficiency and coherence. Experiments on both standard benchmarks and a realistic, large-scale dataset demonstrate the effectiveness of our approach. To our knowledge, this is the first multi-table QA system applied to truly complex industrial tabular data.

1 Introduction

Table Question Answering (Table QA) involves answering natural language questions using structured tabular data. This capability is increasingly critical in numerous practical domains, including healthcare (Singhal et al., 2025), finance (Srivastava et al., 2024), and transportation (Zhang et al., 2024c), where data is commonly stored in relational databases (Wang and Yu, 2025) or spreadsheet-like tables. Effective Table QA systems allow non-expert users to query structured data, thus enhancing accessibility, facilitating data-driven decision-making, and reducing the reliance on manual data manipulation or complex query languages such as SQL.

Unlike traditional text-based QA (Jin et al., 2022), Table QA requires models to understand the structural semantics of tabular data, including column types, row-level content, and inter-cell relationships. In single-table QA, all relevant information resides within one table (see the top of Figure 1). Existing methods typically solve this by either translating natural language questions into executable programs (e.g., SQL via semantic parsing) (Nan et al., 2022; Liu et al.; Cai et al., 2022), or by treating the table as unstructured input for neural models (Herzig et al., 2020; Zhu et al., 2021; Ma et al., 2022; Pal et al., 2022).

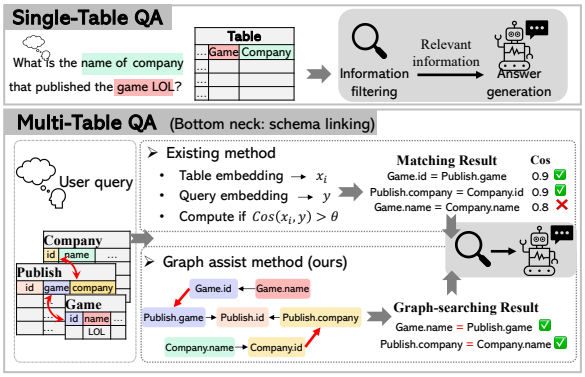


Figure 1: Overview of three Table QA paradigms. **Top**: Single-table QA, where all relevant information is contained within one table. **Middle**: Existing multi-table QA approaches that rely on semantic similarity for schema linking across tables. **Bottom**: Our proposed method, which leverages graph-based, human-curated relational knowledge to explicitly guide schema linking and inter-table reasoning.

Multi-table QA introduces greater complexity, as answering a question often requires reasoning over multiple related tables (middle of Figure 1). This setting poses unique challenges such as identifying relevant tables, aligning schemas, and inferring join paths, which are not present in single-table QA. Existing methods attempt to address these through flattening and concatenating multiple tables (Pal et al., 2023; Wang et al., 2024), leveraging key constraints for semantic alignment (Chen et al., 2024), or adopting multi-step retrieval pipelines (Wu et al., 2025a).

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 5829–5842
November 4-9, 2025 ©2025 Association for Computational Linguistics

However, existing multi-table QA methods have been evaluated only on simplified benchmarks. For instance, the Multi-table and Multi-hop Question Answering (MMQA) benchmark (Wu et al., 2025a) contains tables with an average of just five columns. These are far less complex than real-world datasets like Crash Investigation Sampling System (CISS) (National Highway Traffic Safety Administration, 2024), where tables can have up to 207 columns. These methods typically rely on heavy model training to learn schema linking or semantic alignment, which may suffice for small benchmarks but becomes highly impractical for large-scale tables. Even state-of-the-art LLMs struggle with accurate column-level matching at this scale (Zhang et al., 2024a,b). As schema linking across complex tables remains a major bottleneck, current approaches fail to generalize to realistic industrial scenarios.

We argue that multi-table QA does not necessarily require complex model-based or semantic matching techniques to resolve schema linking across tables. In real-world table QA system, the set of tables and their structures are typically fixed. This allows us to leverage domain knowledge to manually construct an explicit graph that captures both schema linking and join path inference. By doing so, we bypass the major bottleneck of implicit schema alignment during inference. Moreover, this approach enhances transparency and controllability of schema linking, making the overall multi-table QA pipeline more robust and easier to adapt to new domains without needing extensive retraining. Importantly, the graph-based schema representation can be implemented as a plug-and-play module, enabling seamless integration with both closed-source and open-source LLMs.

In this paper, we introduce **SGAM** (Schema Graph-Assisted Multi-table QA), a graph-based framework that integrates human-curated schema knowledge to support multi-table QA, as shown in Figure 1 (bottom). Specifically, we construct a relational graph capturing the semantic and logical dependencies among table columns. Each graph node encodes essential schema details, such as column name, associated table, and primary key status, while edges represent both intra-table semantic relationships and inter-table joinable or derivation-based links. Given a natural language query, our method searches directed paths within this graph to derive clear and interpretable reasoning chains, significantly simplifying multi-table reasoning tasks. Furthermore, we introduce efficient path pruning and sub-path merging strategies to minimize redundant joins and enhance inference coherence.

Our work offers three main contributions. First, we propose a novel graph-based framework for multi-table QA that leverages human-curated relational knowledge to explicitly represent schema linking and join paths, thereby greatly simplifying reasoning over complex real-world tables. Second, we build and publicly release a new multi-table QA dataset based on the real-world CISS, which is substantially more realistic and challenging than existing benchmarks. Finally, we achieve state-of-the-art performance on both standard multi-table QA benchmarks and our new dataset. To the best of our knowledge, this is the first application of multi-table QA to truly large-scale, real-world tabular data.

2 Problem Description

Multi-table question answering involves answering a natural language question by retrieving and reasoning over multiple structured tables. Let the table corpus be defined as $\mathcal{C} = \{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$, where each table $\mathcal{T}_i$ is represented as $\mathcal{T}_i = \{C_1 : A_1, \ldots, C_d : A_d\}$. Here, C_j denotes the name of the j-th column and A_j its associated values, with d being the number of columns in $\mathcal{T}_i$. Given a natural language question Q, the system must identify a subset of relevant tables $E(Q) = \{\mathcal{T}_{Q,1}, \ldots, \mathcal{T}_{Q,k}\} \subseteq \mathcal{C}$ and determine how to join them through appropriate conditions $\bowtie_{c_j}$, such as foreign key relationships or semantically inferred links. These joins form a join path:

$$\mathcal{T}_{Q,1} \bowtie_{c_1} \mathcal{T}_{Q,2} \bowtie_{c_2} \cdots \bowtie_{c_{k-1}} \mathcal{T}_{Q,k}. \quad (1)$$

The system must also retrieve a set of relevant attributes $A(Q) = \{a_1, a_2, \ldots, a_n\}$, where each a_j belongs to one of the selected tables, i.e., $a_j \in \mathcal{T}_{Q,j}$. The retrieved attributes and joined tables together must provide sufficient and relevant information to generate a natural language answer $\mathcal{A}$ to query Q.

3 Method

3.1 Attribute-Level Retrieval

Query decomposition: Traditional retrieval-augmented generation (RAG) methods handle natural language queries well but struggle in multi-table settings. Queries often embed multiple implicit

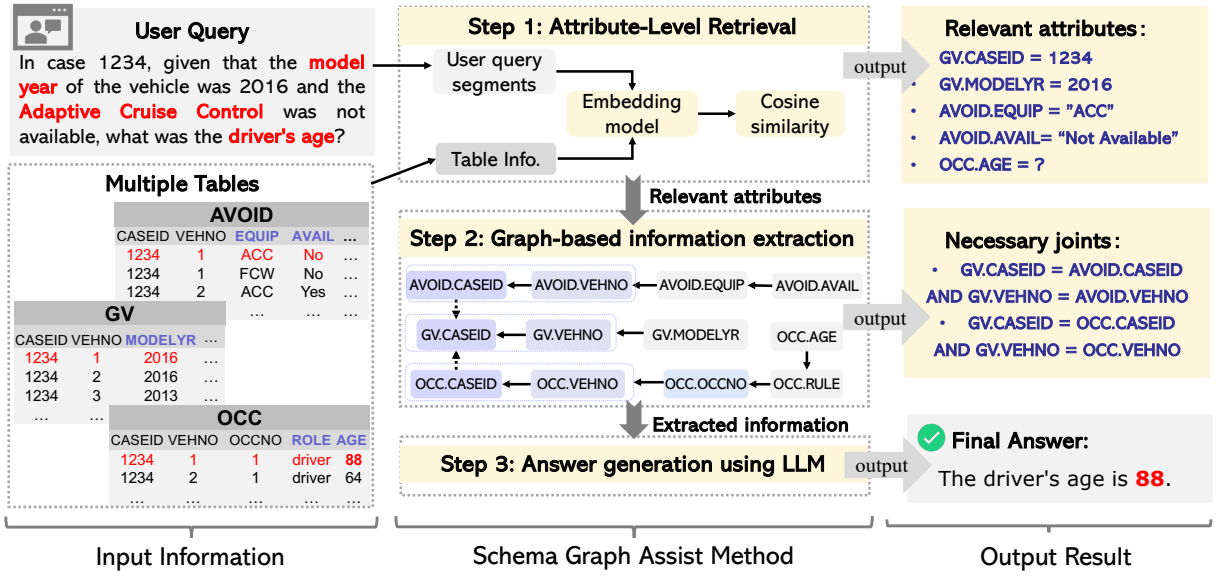


Figure 2: Illustration of our Schema Graph Assist Multi-table QA framework. Given a complex user query and multiple tables, Step 1 retrieves relevant attributes from the tables using embedding-based semantic similarity. Step 2 uses a schema graph to automatically identify and traverse the minimal set of necessary join paths, such as GV.CASEID = AVOID.CASEID and GV.VEHNO = AVOID.VEHNO, to connect related information across tables. Step 3 uses LLM to generate a final answer based on extracted information. This graph-guided reasoning enables the system to accurately extract the target attribute (OCC.AGE) by aligning semantically related fields and resolving multi-hop dependencies. In this case, the system correctly infers the driver's age as 88.

constraints (e.g., *vehicle models*, *case IDs*) within a single sentence, making it difficult to identify all relevant filters. This can lead to incomplete or imprecise retrieval. Moreover, connective phrases (e.g., *at the time of*, *related to*) enhance fluency but add little semantic value, introducing noise into the embedding space and increasing the likelihood of irrelevant matches.

To overcome these challenges, we propose a decomposition-based approach, explicitly segmenting each query into two distinct components: *constraints* and *question focus*. The *constraints* specify conditions to filter relevant records (e.g., particular vehicle models or case identifiers), while the *question focus* identifies the core information sought (e.g., activation of safety features, driver's age). Handling these components separately allows more accurate retrieval and reduces interference from non-essential linguistic elements.

Retrieval tabular attributes: To support accurate attribute identification, our RAG framework constructs structured keyword sets for each attribute. These keywords, derived from attribute names, descriptions, and representative values, fall into two categories: *uniqueness-based keywords* that uniquely identify an attribute, and *frequency-based keywords* that frequently occur within an attribute's values but rarely elsewhere. For instance, the term `animal` is typically associated with variables related to `CRASHTYPE` (crash type), while terms like `truck` and `bus` frequently occur in the `BODYCAT` (vehicle's body category) attribute. This design enables the retrieval process to harness both semantically distinctive signals and statistically salient patterns.

Given a query segment q, we embed it alongside each attribute's keyword set K_i using a pre-trained embedding model, such as OpenAI's `text-embedding-3-small`. The cosine similarity between the query and each keyword set is then computed as $s_i = \cos(\text{emb}(q), \text{emb}(K_i))$. To better distinguish relevant from irrelevant attributes, similarity scores are squared and normalized:

$$s_i' = \frac{s_i^2 - \min_k s_k^2}{\max_j s_j^2 - \min_k s_k^2} \quad (2)$$

We select keywords exceeding a threshold (i.e., $s_i' \geq \tau$) and map them back to candidate attributes, forming the retrieval context for subsequent RAG processing. This method effectively balances recall and precision by exploiting both semantic relevance and frequency-based indicators, enhancing retrieval performance on noisy datasets.

3.2 Graph based information extraction

Graph Construction. We integrate relational database knowledge into our graph construction to preserve semantic structures and relational contexts across multiple tables. Attributes from these tables are represented as nodes within a directed graph $G = (V, E)$. Each node $v \in V$ is uniquely identified by a tuple $v = (C, t, \pi)$, where C is the column name, t the table name, and π a boolean indicating whether C is the primary key. For instance, the node ("ACTIVATE", "AVOID", 0) denotes the column `ACTIVATE` in the `AVOID` table, not being a primary key.

Edges $e = (v_i, v_j) \in E$ represent attribute dependencies and are classified into intra-table edges and inter-table edges based on their connections.

Intra-table Edges. We define intra-table edges to capture both structural keys and semantic relationships among attributes within the same table. These edges fall into two categories:

(1) *Primary key-based edges*: Directed edges are created from non-key attributes to primary key attributes within a table, reflecting inherent structural dependencies. For example, the edge from `("OCCNO", "OCC", 1)` to `("VEHNO", "OCC", 1)` encodes the association between occupants and their vehicles:

$$E^{\text{key}}_{\text{intra}} = \{(v_i, v_j) \in E \mid v_i.t = v_j.t \wedge v_j.\pi = 1\} \tag{3}$$

(2) *Semantic context edges*: Certain attributes lack standalone meaning and must be interpreted in combination with others. For instance, `ACTIVATE` (indicating system activation) is meaningful only when contextualized with `EQUIP` (the system type) and `AVAIL` (system availability). To maintain semantic integrity, we manually introduce edges among such interdependent attributes:

$$E^{\text{context}}_{\text{intra}} = \{(v_i, v_j) \in E \mid v_i.t = v_j.t\} \tag{4}$$

Inter-table Edges. Inter-table edges capture relationships between attributes across different tables, supporting schema linking and multi-hop reasoning. These edges fall into two categories:

(1) *Key-based edges*: These connect attributes with identical names (e.g., `CASEID`, `VEHNO`) that serve as primary keys in their respective tables, indicating joinable relationships:

$$E^{\text{key}}_{\text{inter}} = \{(v_i, v_j) \in E \mid v_i.t \neq v_j.t \wedge\ v_i.a = v_j.a \wedge v_i.\pi = v_j.\pi = 1\} \tag{5}$$

(2) *Derivation edges*: For attributes whose values are algorithmically derived from attributes from other tables, we explicitly add edges to reflect their relationship. For example, the attribute `ALCOHOL INVOLVEMENT` in table `CRASH` is computed based on the values of `ALCOHOL TEST RESULT FOR DRIVER` in table `GV`. Therefore, in the graph, the node `("ALCINV", "GV", 0)` will have outcoming edges to `("ALCTEST", "GV", 0)`, representing its derivation logic.

$$E^{\text{derive}}_{\text{inter}} = \{(v_i, v_j) \in E \mid v_i.t \neq v_j.t \wedge v_i.a = v_j.a\} \tag{6}$$

To unify structural and semantic reasoning, we merge intra-table edges and inter-table edges into a single graph structure:

$$E = (E^{\text{key}}_{\text{intra}} \cup E^{\text{context}}_{\text{intra}}) \wedge (E^{\text{key}}_{\text{inter}} \cup E^{\text{derive}}_{\text{inter}}) \tag{7}$$

We also manually assign labels for different edges to clearly represent both extraction cues and reasoning logic, forming the backbone for downstream multi-table query interpretation.

Path Search for Attribute Dependency Chains. Given a query involving one or more target attributes, we perform a directed path search over the graph G to identify semantic dependency chains. Each search begins from a query-referenced attribute node and proceeds along dependency edges until reaching a terminal node, typically a primary key. For tables with multiple primary keys, we define a single terminal key v^* as the most global (i.e., top-level) anchor point:

$$v^* = \arg\max_{v_k \in \Pi} \text{depth}(v_k) \tag{8}$$

where Π is the set of a table's primary keys and $\text{depth}(\cdot)$ measures the topological distance from the key node to the attribute node. This guarantees that the selected path terminates at the most globally joinable key. Each resulting path $P = (v_1, v_2, \ldots, v_n)$ represents a coherent attribute dependency chain. The full answer is computed by aggregating all valid paths:

$$Answer(Q) = \phi(\mathbb{P}),\ \mathbb{P} = \{P_1, P_2, \ldots, P_n\} \tag{9}$$

Path Pruning and Sub-path Merging. To reduce redundancy and improve reasoning efficiency, we introduce a pruning and merging strategy over identified attribute dependency paths. For each

target attribute node $v_i \in T_j$, the graph search typically yields multiple dependency paths $P(v_i, v_c)$ ending at a terminal key node v_c (often a primary key) in either the same or different tables. These paths may span within or across tables, leading to overlapping semantics and redundant retrieval when they converge on the same key.

To address this, we apply the following pruning procedure to each path in $\mathbb{P}$. For a path P, let e^* denote the last inter-table edge from the attribute node. We truncate the path to end at e^*, retaining only the sub-path $P' \subseteq P$ that lies entirely within a single table:

$$P' = P(e^*.\text{dst}, v_c),\ e^*.\text{dst}.t = v_c.t \qquad (10)$$

where $e^*.dst$ denotes the target node of the last inter-table edge. This ensures each retained sub-path P' stays within a single table and links a relevant attribute to its terminal key. Merging these P' yields a minimal, non-redundant set of attribute-to-key chains, avoiding duplicate extraction, maintaining contextual coherence, and reducing computational cost

4 Experiments

We evaluate our graph-based method through two experiments: SQL generation on the BIRD benchmark and end-to-end QA on the real-world CISS dataset. We now detail the setup and results [1].

4.1 Datasets

BIRD: To evaluate the effectiveness of approach in multi-table information retrieval, we first used the public BIRD[2] dataset (Li et al., 2024). BIRD is composed of 95 table corpus ranging from healthcare, transportation, sports, and retail and each corpus contains 7.4 tables on average. From BIRD, we selected two complex corpus available: *Olympics* and *Financial*, because BIRD was originally designed for text-to-SQL tasks, rather than multi-table question answering. The difference in task settings means that the full dataset is not directly compatible with our method, which requires graph construction for multi-table reasoning. Therefore, we manually constructed schema graphs for two representative and complex subsets, *Olympics* and *Financial*, to adapt them to our QA framework. *Olympics* contains 11 tables with an average of 2.9 attributes per table and Financial is formed by 8 tables with and average of 6.7 attributes per table. For each corpus, we built individual graphs corresponding to the specific domain.

Properties	Benchmark			Real Crash
	Olympics	Financial	BIRD	CISS
Total Tables	11	8	7.4	38
Properties per table				
Avg attributes/columns	2.9	6.7	-	**30.2**
Avg foreign keys	1.8	2.3	-	2.1
Avg primary keys	1.0	1.0	-	2.1
Question Category				
Numerical	78	46	-	5
List	71	39	-	75
Select	11	12	-	548
Count	9	9	-	0
Total	169	101	12751	628

Table 1: Descriptive statistics and question types for benchmark datasets (Olympics, Financial, BIRD) and the real-world CISS dataset. Notably, CISS tables contain significantly more attributes per table compared to the benchmark datasets.

CISS: To showcase the real-world applicability of our approach, we construct a new multi-table question answering dataset based on the CISS. All tables in this dataset are domain-specific and task-relevant, with well-defined key relationships. CISS provides in-depth crash data from the United States, including detailed information on crash types, accidents' causes, injury severity, vehicle types, and more. The dataset covers approximately 3000 to 4500 crashes annually. The recorded crash cases span from accident year 2017 to 2023. We began by converting structured tabular data into textual summaries and use GPT-4o to automatically generate question-answer pairs. Each question is manually verified for clarity, relevance, and correctness, resulting in a verified accuracy of 98%. The final dataset consists of more than 620 high-quality QA pairs covering diverse aspects of traffic accidents, including crash types, vehicle conditions, driver and pedestrian demographics, and injury outcomes. To encourage varied reasoning complexity, the dataset includes 220 one-hop, 300 two-hop, and 100 multi-hop questions, where a *hop* refers to a distinct reasoning step required to connect different pieces of information across attributes or tables. This CISS-based QA dataset also serves as a case study to evaluate the effectiveness of our graph-enhanced approach in complex, domain-driven applications.

4.2 Baselines

To ensure an up-to-date comparison on the BIRD dataset, we selected the highest ranking models

[1] https://github.com/hiXixi66/SGAM-Multi-table-QA
[2] https://bird-bench.github.io/

from the BIRD leaderboard which tracks submitted results proposed in recent literature. All the selected baseline models use GPT-4o as backbone and demonstrate diverse strategies for SQL generation and multi-table reasoning. We compare our results against Distillery+GPT-4o (Maamari et al., 2024), RSL-SQL+GPT-4o (Cao et al., 2024), OpenSearch-SQL (Xie et al., 2025), and AskData+GPT-4o (AT&T CDO-DSAIR), with the latter ranking first among all. The first three methods are all among the top-performing systems on the leaderboard and rely on oracle knowledge, which significantly simplifies the task. AskData+GPT-4o also reports results without oracle knowledge, providing a more realistic comparison scenario.

4.3 Implementation details

For embedding-based retrieval, we use the `text-embedding-3-small` model made available by OpenAI (OpenAI, 2024b). For model inference, we use LLaMA-3 70B (AI, 2024), GPT-3.5 (OpenAI, 2023), GPT-4o (OpenAI, 2024a), Qwen2.5-7B (Yang et al., 2025), and DeepSeek-R1-Distill-Qwen-32B (DeepSeek-AI, 2025) models on a high-performance computing cluster with 4 NVIDIA A100 GPUs (40GB). All main prompts used throughout the experiments are listed in the Appendix A.

4.4 Tasks and Evaluation Metrics

We conduct two types of evaluation using each of the aforementioned datasets.

Graph-aware evaluation (using BIRD): Designed to directly assess the impact of introducing our graph-based schema representation into SQL generation process, we compare model performance with and without graph guidance to understand how structured schema information influences query accuracy. We report execution accuracy and SQL generation metrics, precision, recall and F1 across different datasets for both variants.

End-to-end evaluation (using CISS): Unlike traditional Text2SQL tasks that assume clean, well-structured data, the CISS dataset presents real-world challenges such as mixed structured and textual fields, inconsistent values, and complicated connected tables. Many attributes can only be used when combined with other attributes, and reasoning often goes beyond simple table joins. To demonstrate that a schema graph can be used beyond guide SQL generation tasks, such as resolving inconsistencies and modeling cross-table semantics, we conduct experiments on a real-world QA task over CISS.

We assess each generated answer by evaluating whether it correctly addresses the question's core information need, and report performance across different hop categories (1-hop, 2-hop, and $\geq$3-hop) to assess reasoning complexity. The backbone models vary across experiments, but the question decomposer model (Q-d) remains fixed per block to isolate the impact of the encoder. No oracle evidence is provided; models must retrieve and reason over raw table data guided by graph-enhanced or baseline retrieval mechanisms. We use GPT-4o to automatically judge whether the generated answer correctly addresses the question, based on semantic correctness. To ensure reliability, we manually reviewed 30% of the results, confirming strong agreement between the model's judgment and human evaluation.

4.5 Results: Graph-Aware Evaluation

To evaluate the effectiveness of our proposed graph-based approach for assisting large language models in SQL generation, we conduct experiments on the BIRD dataset. Our task is framed as multi-table question answering, where the model must interpret a question and retrieve relevant information across complex relational schemas. Unlike prior work that assumes access to oracle knowledge—i.e., explicit evidence annotations such as `Charter schools` refers to `Charter School (Y/N)` ` = 1` in table `fprm`—our setting reflects real-world applications, where such privileged mappings are typically not readily available. To simulate this scenario, we do not provide any evidence annotations to the model in our approach. Instead, we focus on complex corpus that involve multiple interrelated tables and construct a relational graph for each topic to encode schema and semantic dependencies. Questions are grouped by topic and then split into three subsets: 20 examples for training, 30 examples for development, and the remaining for testing.

As shown in the upper part of Table 2, our method achieves strong execution accuracy (EX), reaching 83.19% at pass@1 and up to 87.61% at pass@10 for the *Olympics* dataset and up to 78.57% for the *Financial* dataset. The average is higher than AskData + GPT-4o, which didn't use oracle knowledge as ours. Importantly, most of baseline

Model	Oracle Knowledge	Execution on Eval Datasets	
Comparative Method (pass@1)		**Dev**	**Test**
Distillery + GPT-4o Maamari et al. (2024)	✓	67.21*	71.83*
RSL-SQL + GPT-4o Cao et al. (2024)	✓	68.12*	70.21*
OpenSearch-SQL Xie et al. (2025)	✓	69.30*	72.28*
AskData + GPT-4o (AT&T CDO-DSAIR)	✓	75.36*	77.14*
AskData + GPT-4o (AT&T CDO-DSAIR)	✗	65.91* (**-9.45**)	67.41* (**-9.73**)
Our Method		**Olympics**	**Financial**
SGAM+ GPT-4o pass@1	✗	83.19	55.36
pass@5	✗	86.73	73.21
pass@10	✗	87.61	78.57

Table 2: Execution accuracy on the BIRD Text2SQL benchmark. Scores with * are from the official leaderboard. Since BIRD covers a wide range of domains, we select two representative domains for evaluation. Method *AskData.* shows a large performance drop without Oracle Knowledge (schema linking hints), while our method performs well without using it. Green highlights indicate our method outperforms non-Oracle baselines.

methods listed in the lower part of the table depend on oracle annotations and yield lower performance. These results highlight the effectiveness of our human knowledge graph-guided framework in enabling LLMs to generate accurate SQL in a fully automated and without explicit oracle knowledge.

While GPT-4o plays an important role, its performance degrades significantly without the support of the graph structure. To verify it, we compare the models performance with and without graph-assisted guidance in generating SQL. Table 3 reports the precision, recall, and F1 score on the selected BIRD datasets. As shown, using our approach significantly improves performance. On the *Olympics* dataset, F1 score increases by 51.26% (from 33.29% to 84.55%), and on the *Financial* dataset by 39.19% points (from 17.68% to 56.87%). These significant improvements demonstrate that by using a graph-based approach we can help guide the model to key attribute relationships between different tables.

4.6 Results: End-to-end Evaluation

To validate the applicability of our method in real-world, noisy, and semantically entangled data settings, we further evaluate it on the CISS dataset.

Table 4 presents a detailed comparison across different backbone models under varying reasoning depths (1-hop, 2-hop, and 3-hop), both with and without question decomposition (Q-d). The results clearly demonstrate the critical role of question decomposition in enhancing multi-hop reasoning performance. Without decomposition, even strong models like GPT-3.5 and GPT-4o struggle on 3-hop questions, achieving only 68.92% and 62.16% accuracy respectively. However, when equipped with our question decomposition module (e.g., GPT-4o as the Q-d model), their performance improves substantially, reaching 86.11% and 91.89% respectively. This trend holds consistently across other backbone models and hop depths, confirming that decomposition provides a structured representation of complex questions—such as explicit constraints and a focused main query—that significantly reduces reasoning ambiguity.

More importantly, these results reveal that our framework reduces the dependency on the intrinsic reasoning capacity of the backbone LLM. For instance, the performance gap between smaller, instruction-tuned models (e.g., Qwen2.5-7B) and larger proprietary models (e.g., GPT-4o) is significantly narrowed or even eliminated when question decomposition is applied. This suggests that our framework acts as a form of "reasoning scaffolding," allowing even lightweight models to perform competitively on complex, multi-table QA tasks. Furthermore, the decomposition-based pipeline enables better modularity and control in system design, as reasoning steps are made explicit and interpretable.

In sum, the consistent improvements observed across all reasoning depths and model configurations underscore the dual benefits of our approach: (1) improved reasoning accuracy through structured question understanding, and (2) reduced reliance on large-scale proprietary models, thereby enhancing the practicality and scalability of multi-table QA in real-world applications.

5 Related Work

Single-Table Question Answering. Early work treats table QA as semantic parsing: a question is translated into executable SQL or `pandas` code that is then run on a single table (Nan et al., 2022; Liu et al.; Cai et al., 2022). Extractive models instead read the whole table as text and select the answer cell (Herzig et al., 2020; Zhu et al., 2021), while generative models directly produce the answer without an explicit program (Ma et al., 2022; Pal et al., 2022). Recent studies leverage large language models (LLMs): some guide LLMs to emit higher-quality SQL (Zhang et al., 2024a; Ye

	P		R		F1	
Dataset	w/	w/o Δ	w	w/o Δ	w	w/o Δ
Olympics	84.15 ± 1.20	-50.98	85.71 ± 1.47	-50.96	84.55 ± 1.26	-51.26
Financial	58.93 ± 3.32	-40.75	56.43 ± 3.36	-38.01	56.87 ± 3.35	-39.19
Avg	71.54	-45.87	71.07	-44.49	70.71	-45.23

Table 3: Performance comparison for precision (P), recall (R), and F1 score (F1). We compare the performance of GPT-4o with and without graph assistance for the selected BIRD datasets.

Q-d model	-			*GPT-4o*		*LLaMA-3 70B*		*Qwen2.5-32B*	
Answer model	**1-hop**	**2-hop**	**≥3-hop**	**2-hop**	**≥3-hop**	**2-hop**	**≥3-hop**	**2-hop**	**≥3-hop**
Qwen2.5-7b-instruct-1m	84.10	65.25	60.25	85.57	91.04	91.09	82.76	68.06	64.10
DeepSeek-R1-Distill-Qwen-32B	89.94	61.03	66.21	87.51	89.47	88.23	86.57	87.50	84.31
Llama-3.3-70B-Instruct	89.58	64.63	62.16	90.70	88.89	89.14	88.89	86.81	84.58
GPT-3.5	90.48	68.18	68.92	88.37	86.11	91.35	82.28	88.37	80.56
GPT-4o	91.63	66.89	62.16	90.58	91.89	89.92	88.89	89.28	81.08

Table 4: Evaluation of QA performance under different backbone configurations, varying both the Q-d and Answer models. "–" indicates no question decomposition applied.

et al., 2023; Zhang et al., 2024b), whereas others let the LLM read the table and form the answer end-to-end (Wu et al., 2025b).

Multi-Table Question Answering. Multi-table QA focuses on answering queries that require reasoning over multiple interrelated tables. A core challenge lies in identifying implicit join paths, i.e., determining which columns across tables are semantically or relationally aligned—since such connections are rarely explicit in the query. Early work by Pal et al. (Pal et al., 2023) introduced a large-scale multi-table QA benchmark, where all relevant tables are flattened and concatenated with the question, followed by Transformer-based modeling and multi-table pretraining. Chen et al. (Chen et al., 2024) proposed a join-aware method that scores column-level joinability based on name semantics, value overlaps, and schema constraints. Wu et al. (Wu et al., 2025a) utilized LLMs to decompose complex questions into sub-questions, iteratively retrieving and linking tables based on column overlaps. While effective in controlled settings, these methods struggle with real-world scenarios such as CISS (National Highway Traffic Safety Administration, 2024), where tables may contain over 200 columns. The resulting complexity severely hinders accurate table linking and cross-table reasoning.

To address these limitations, we propose a graph-assist approach that encodes column-level relations across tables using schema-derived structure. This enables interpretable and scalable reasoning by selectively retrieving relevant columns based on query intent. To the best of our knowledge, this is the first system evaluated on the realistic tables, demonstrating scalability to real-world industrial data.

6 Conclusion

In this work, we present a novel graph-based framework for multi-table question answering that integrates explicit, human-curated relational knowledge to assist in schema linking and reasoning. By constructing a structured relational graph that captures both intra- and inter-table dependencies, our method enables interpretable reasoning paths for complex queries, significantly improving robustness and scalability in real-world settings. We further introduce pruning and sub-path merging strategies to reduce redundancy and enhance inference coherence. To support real-world evaluation, we release a challenging multi-table QA dataset based on the CISS corpus. Experiments on both standard benchmarks and our newly constructed dataset demonstrate the effectiveness of the proposed approach. By offloading structural and semantic understanding to a lightweight schema graph, our framework provides a scalable and interpretable solution that advances the practical deployment of multi-table QA in complex real-world industrial settings.

7 Limitations

Limitations of Domain Knowledge. While our method demonstrates strong performance with open-source models, we do not fine-tune them on domain-specific data. As a result, the models may occasionally produce factual errors due to gaps in vehicle safety domain knowledge. Incorporating domain-adaptive fine-tuning could further improve answer accuracy in these expert settings.

Dependency on Accurate Attribute Retrieval. While our method significantly improves multi-table QA accuracy by using schema-guided reasoning over a graph after attributes retrieval, it still relies on correctly identifying most of relevant attributes during the retrieval phase. If AttributeRAG fails to extract the very necessary columns corresponding to semantic components of the question, the downstream graph reasoning process may lack critical information, ultimately leading to incorrect or incomplete answers. In future work, we plan to explore more robust and semantically grounded methods for mapping each part of a natural language question to its corresponding variables during the initial retrieval stage.

References

Meta AI. 2024. Llama 3: Open foundation and instruction-tuned models.

Zefeng Cai, Xiangyu Li, Binyuan Hui, Min Yang, Bowen Li, Binhua Li, Zheng Cao, Weijie Li, Fei Huang, Luo Si, et al. 2022. Star: Sql guided pre-training for context-dependent text-to-sql parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1235–1247.

Zhenbiao Cao, Yuanlei Zheng, Zhihao Fan, Xiaojin Zhang, Wei Chen, and Xiang Bai. 2024. Rsl-sql: Robust schema linking in text-to-sql generation. *arXiv preprint arXiv:2411.00073*.

Peter Baile Chen, Yi Zhang, and Dan Roth. 2024. Is table retrieval a solved problem? exploring join-aware multi-table retrieval. *arXiv preprint arXiv:2404.09889*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333.

Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. 2022. A survey on table question answering: recent advances. In *China Conference on Knowledge Graph and Semantic Computing*, pages 174–186. Springer.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. Tapex: Table pre-training via learning a neural sql executor. In *International Conference on Learning Representations*.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2022. Open domain question answering with a unified knowledge interface. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1605–1620.

Karime Maamari, Fadhil Abubaker, Daniel Jaroslawicz, and Amine Mhedhbi. 2024. The death of schema linking? text-to-sql in the age of well-reasoned language models. *arXiv preprint arXiv:2408.07702*.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, et al. 2022. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

National Highway Traffic Safety Administration. 2024. Crash investigation sampling system (ciss). Accessed: 2025-05-07.

OpenAI. 2023. Gpt-3.5 series.

OpenAI. 2024a. Gpt-4o technical report.

OpenAI. 2024b. text-embedding-3-small model.

Vaishali Pal, Evangelos Kanoulas, and Maarten Rijke. 2022. Parameter-efficient abstractive question answering over tables or text. In *Proceedings of the Second DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering*, pages 41–53.

Vaishali Pal, Andrew Yates, Evangelos Kanoulas, and Maarten de Rijke. 2023. Multitabqa: Generating tabular answers for multi-table question answering. *arXiv preprint arXiv:2305.12820*.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Mohamed Amin, Le Hou, Kevin Clark, Stephen R Pfohl, Heather Cole-Lewis, et al. 2025. Toward expert-level medical question answering with large language models. *Nature Medicine*, pages 1–8.

Pragya Srivastava, Manuj Malik, Vivek Gupta, Tanuja Ganu, and Dan Roth. 2024. Evaluating llms' mathematical reasoning in financial document question answering. *arXiv preprint arXiv:2402.11194*.

Bing Wang, Chunhao Wang, Xingpeng Zhang, Chunlan Zhao, Xiaoling Yang, and Kuan Guo. 2024. Multi-table question answering method based on correlation evaluation and precomputed cube. In *International Conference on Knowledge Science, Engineering and Management*, pages 393–405. Springer.

Shuai Wang and Yinan Yu. 2025. iQUEST: An iterative question-guided framework for knowledge base question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 15616–15628.

Jian Wu, Linyi Yang, Dongyuan Li, Yuliang Ji, Manabu Okumura, and Yue Zhang. 2025a. MMQA: Evaluating LLMs with multi-table multi-hop complex questions. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, et al. 2025b. Tablebench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25497–25506.

Xiangjin Xie, Guangwei Xu, Lingyan Zhao, and Ruijie Guo. 2025. Opensearch-sql: Enhancing text-to-sql with dynamic few-shot and consistency alignment. *arXiv preprint arXiv:2502.14913*.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, and Zipeng Zhang. 2025. Qwen2.5-1m technical report. *arXiv preprint arXiv:2501.15383*.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, pages 174–184.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024a. Tablellama: Towards open large generalist models for tables. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6024–6044.

Xiaokang Zhang, Sijia Luo, Bohan Zhang, Zeyao Ma, Jing Zhang, Yang Li, Guanlin Li, Zijun Yao, Kangli Xu, Jinchang Zhou, et al. 2024b. Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios. *arXiv preprint arXiv:2403.19318*.

Zijian Zhang, Yujie Sun, Zepu Wang, Yuqi Nie, Xiaobo Ma, Peng Sun, and Ruolin Li. 2024c. Large language models for mobility in transportation systems: A survey on forecasting tasks. *arXiv preprint arXiv:2405.02357*.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics.

A Prompts

A.1 Question decomposition

We use the following prompt to ask LLMs to decompose a question into (conditions, main question).

```
You are a helpful assistant
for question understanding and
decomposition.  Your task is to
rewrite a complex question into:  -
A list of conditions (such as case
ID, vehicle model, etc.)  - A main
question that describes what needs to
be answered
Return the result in the following
format:
Condition 1:  ...
Condition 2:  ...
...
Main Question:  ...

Example:
Question:  For the vehicle model Bolt
EV, were any active safety systems
activated in case 20335?
Answer:
Condition 1:  the vehicle model is
Bolt EV
Condition 2:  case ID is 20335
Main Question:  were any active safety
systems activated?

Example:
Question:  In case 17390, what was the
model year of the vehicle with most
severe damage to the front?
```

```
Answer:
Condition 1:  Case ID is 17390
Condition 2:  The damage to the front
is the most severe
Main Question:  What was the model
year of the vehicle?

Now decompose this question:
```

A.2 Answer generation on CISS

We use the following prompt to ask LLMs to generate answers for questions on CISS dataset:

```
  Question:  What was the direction of
travel of Vehicles in CASE 13803?
  In case 13803, the crash summary is:
Vehicle 1 and vehicle 2 were traveling
east approaching an intersection.
Vehicle 2 was stopped ahead of Vehicle
1.  Vehicle 1's front plane contacted
Vehicle 2 rear plane.  The number of
in-transport vehicles is:  2
  ...
  **Please read all the information
provided above carefully and focus on
question related information, answer
the above question:**
```

A.3 Attributes select for SQL

We use the following prompt to ask LLMs to select attributes from tables:

```
  You are a helpful assistant for
aligning natural language question
phrases with database variables.

You are given:
1.  A database schema table with
sheet_name (table), column_name
(field), and key_words (keywords
associated with the field).
2.  A natural language question.
Your task is:  Split the question
into key semantic fragments (e.g.,
person name, gender, event, year,
etc.).  For each fragment, return
the most relevant column_name and its
corresponding sheet_name based on the
key_words.  Ensure that each value
in the output matches the database
**exactly**, including full names,
suffixes (e.g., "II", "Jr."), and
punctuation (e.g., commas).
Tips:
Do not shorten names (e.g., avoid
"Michael Phelps").
You must ...  ...
Output format:

[
"fragment":  "...",
"column_name":  "...", //...
"gender", not "person.gender") //...
"sheet_name":  "..." //...
]
```

A.4 SQL generation

We use the following prompt to ask LLMs to generate SQLs:

```
Given the following reasoning paths
and natural language question,
generate the corresponding SQL query.
Note:  Some paths may be irrelevant
to answering the question.  Please
carefully identify and ignore
unrelated paths.
When assigning values to variables in
the WHERE or SELECT clause, always
prioritize using the variable at the
**start of the path**, not internal
IDs or primary keys.
Only join tables that contribute
directly to the SELECT, WHERE, or
GROUP BY clauses.  Avoid unnecessary
joins (e.g., 'games_competitor')
if the question does not involve
participation, age, or game-specific
attributes.  Unused joins may
introduce incorrect row duplication.
  ...
Example:

Path 1:  full_name[person], id[person],
person_id[games_competitor],
id[games_competitor]

Path 2:  games_name[games],
id[games], games_id[games_competitor],
id[games_competitor]

Path 3:  age[games_competitor],
id[games_competitor]
```

```
Question: What was the name of the
Olympic game that John Aalberg took
part in when he was 31?

‘‘‘sql SELECT games.games_name FROM
person JOIN games_competitor ON
person.id = games_competitor.person_id
JOIN games ON games.id =
games_competitor.games_id
JOIN competitor_event ON
games_competitor.id =
competitor_event.competitor_id WHERE
person.full_name = 'John Aalberg' AND
games_competitor.age = 31; ‘‘‘
```

B Case Study

B.1 1-hop question

The following example highlights how our graph enhanced retrieval process. While the attribute CRASH.MANCOLL provides the target answer field, the graph also links to CRASH.SUMMARY, which offers contextual evidence describing the crash scenario. By incorporating this additional information, the model generates not only the correct answer but also a reliable explanation, making the response more trustworthy and grounded in the original data.

```
Question: What was the manner of the
collision in CASE 6916?

Attributes retrieval result:
MANCOLL[CRASH], CASEID[CRASH]

Graph searching result:
Path 1: MANCOLL[CRASH],
SUMMARY[CRASH], CASEID[CRASH]

Information extraction result: In
case 6916, the crash summary is:
Vehicle #1 was traveling west on an
interstate highway, departed the road
to the right, and impacted a guardrail
with the front plane.
The MANCOLL is: Not Collision with
Vehicle in Transport.

Answer: In case 6916, the manner of
the collision is: Not Collision with
Vehicle in Transport. This means
that Vehicle #1 did not collide with
another vehicle in transport; instead,
it departed the road and collided with
a guardrail.
```

B.2 2-hop question

This example demonstrates how the schema graph enhances the robustness of the multi-table QA pipeline. In this case, the attribute-level RAG module successfully retrieves DVLONG (longitudinal speed) but overlooks DVTOTAL (total speed), which is also essential to answer the question. However, our schema graph captures the semantic dependency between DVLONG and DVTOTAL by modeling them as connected nodes within the same table and linking them through shared keys such as VEHNO and CASEID. As a result, both attributes are extracted during graph traversal, effectively compensating for the RAG omission and ensuring that the model obtains complete information for accurate answering.

```
Question: For the case 27187, for the
vehicle model Grand Prix, what is the
speed of the vehicle?
Attributes retrieval result:
DVLONG[GV], Model[VPICDECODE],
CASEID[CRASH]

Graph searching result:
Path 1: DVTOTAL[GV], DVLONG[GV],
VEHNO[GV], CASEID[GV]
Path 2: Model[VPICDECODE],
VEHNO[VPICDECODE], CASEID[VPICDECODE]
Path 3: CASEID[VPICDECODE],
CASEID[GV]

Information extraction result: In
case 27187, for vehicle NO.2, the
total speed is: Actual value 38, the
longitudinal speed is: Actual value
-37 In case 27187, for vehicle NO.2,
the model is: Grand Prix
...

Answer: The total speed of the
vehicle is 38.
```

B.3 3-hop question

This 3-hop question requires reasoning over multiple conditions: identifying the correct occupant based on age (25 years), gender (female), and belt use (lap and shoulder belt), and then determining which body region made contact with the left door panel. While attribute-level retrieval alone may

struggle to consistently extract all relevant fields, our schema graph enables the model to follow structured paths linking semantic and relational dependencies across tables.

```
In case 16215, for the driver who
is a 25-year-old female using a lap
and shoulder belt, which body region
contacted the left door panel?

Attributes retrieval result:
ROLE[OCC], AGE[OCC], BELTUSE[OCC],
BODYREGION[OCCONTACT], CONTAREA
[OCCONTACT], CASEID[CRASH]

Graph searching result:
Path 1: BELTUSE[OCC], ROLE[OCC],
OCCNO[OCC], VEHNO[OCC], CASEID[OCC]
Path 2: BODYREGION[OCCONTACT],
CONTAREA[OCCONTACT], OCCNO[OCCONTACT],
VEHNO[OCCONTACT], CASEID[OCCONTACT]
Path 3: AGE[OCC], ROLE[OCC],
OCCNO[OCC], VEHNO[OCC],
CASEID[OCC] Path 3: CASEID[OCC],
CASEID[OCCONTACT]

Information extraction result: In
case 16215, for vehicle NO.2, occupant
NO.1, the occupant's role is: Driver,
the belt-use determination is: Lap
and shoulder belt.
In case 16215, for vehicle NO.2,
occupant NO.1, the vehicle component
the occupant is suspected to have
contacted is: Left rear upper
quadrant, the CONTACTED COMPONENT
AREA is: Left Door Panel, the the
suspected body region that made
contact to the Contacting Component
is: Elbow - Left.
In case 16215, for vehicle NO.2,
occupant NO.1, the occupant's role
is: Driver, the occupant's age is:
Actual age 25 years.

Answer: In case 16215, the left
shoulder was the body region that
contacted the left door panel.
```

B.4 Fail cases

These failure cases describe a key limitation of LLM-based reasoning in domain-specific contexts. Although the graph-based system successfully extracted all relevant information from tables, the LLM failed to aggregate these information appropriately due to a lack of domain-specific knowledge.

```
Question: What was the general area
of damage on the first vehicle in CASE
18765?

Information extraction result: In
case 18765, for event NO.1, the
general damage area of first vehicle
is: Front
In case 18765, for event NO.2, the
general damage area of the first
vehicle is: Back/Truck Back
In case 18765, for event NO.3, the
general damage area of the first
vehicle is: Front

Answer: The general damage area on
the first vehicle (V1) in case 18765
was in the Front.
Ground true answer: The general area
of damage on the first vehicle (V1) in
CASE 18765 includes both the front and
the back.
```

This above example demonstrates a key limitation: the model failed to consolidate multiple related mentions (e.g., "Front", "Back/Truck Back") into a unified interpretation, defaulting to the first mention instead.

```
Question: Was the vehicle equipped
with any safety systems in CASE 31120?

Answer:In Case 31120, the vehicles
(NO.1 and NO.2) were equipped with
some safety systems, which are systems
designed to prevent accidents or help
reduce severity. From the information
provided, the active safety systems
available are: Automatic Crash
Notification (Yes for both Vehicle
NO.1 and Vehicle NO.2)

Information extraction result:In case
31120, for vehicle NO.1, the avoidance
equipment to check is: Automatic
Crash Notification, the availability
```

```
condition is:  Yes
In case 31120, for vehicle NO.1,
the avoidance equipment to check
is:  Daytime Running Lamps, the
availability condition is:  Yes
In case 31120, for vehicle NO.1,
the avoidance equipment to check
is:  Rearview Video System, the
availability condition is:  Yes
In case 31120, for vehicle NO.2, the
avoidance equipment to check is:
Automatic Crash Notification, the
availability condition is:  Yes ...
Ground true answer:  Yes, in case
31120, both Vehicle No.1 and Vehicle
No.2 were equipped with safety systems.
Specifically, they both had the
following safety systems available:
- Automatic Crash Notification
- Daytime Running Lamps
- Rearview Video System

  Additionally, Vehicle No.  2 also
had Advanced Lighting available."
```

The answer is incorrect due to the language model's implicit assumption about the definition of safety systems. The model misclassified Daytime Running Lamps and Rearview Video System as not being part of the safety systems, likely due to its general knowledge. However, the table explicitly includes it under active safety features, leading to an incomplete answer.