Computers and Artificial Intelligence; Vol.3, No.6; 2026
E-ISSN: 3006-3728 P-ISSN: 3006-371X
DOI: https://doi.org/10.70267/cai.v3n6.261276
Published by: Zeus Press

A Review of Schema Representation and Compression Techniques in Large Language Models

Qibin He*

School of Information Engineering, Minzu University of China, Beijing 100081, China

**Corresponding author: Qibin He.*

Abstract

With the increasing application of large language models (LLMs) in text-to-Structured Query Language (Text-to-SQL) and intelligent data querying, database schemas have become an important context that affects query understanding, schema linking, and SQL generation. Based on benchmarks including WikiSQL, Spider, BIRD, and Spider 2.0, this paper reviews six directions: natural-language representation and prompt construction, semantic representation and embedding, relevance modeling and dynamic selection, schema compression and token optimization, structural relation modeling and auxiliary information enhancement, and LLM-driven Text-to-SQL applications. This paper compares the publicly reported results of representative methods, including RESDSQL, CHESS, RSL-SQL, CodeS, DAIL-SQL, and MAC-SQL, as well as RAT-SQL, SDSQL, and LGESQL, and points out that schema compression is not simple text deletion. Instead, it combines retrieval, ranking, pruning, structural constraints, and dynamic sub-database construction to balance information retention and context cost. Existing research still has several prominent problems: relevance ranking has difficulty considering both semantic matching and join-path completeness; compression effectiveness lacks unified and comparable evaluation metrics; and some methods do not adequately report token costs and end-to-end latency. In addition, the generalization ability of related methods in complex, real-world business databases remains to be verified. In response to these problems, this paper summarizes research gaps concerning dynamic K values, unified compression metrics, end-to-end efficiency, and real-world database validation, and proposes directions such as a closed loop of retrieval, compression, generation, and execution verification.

Keywords

large language model, schema representation, schema compression, embedding, schema linking, Text-to-SQL, token optimization

1. Introduction

1.1 Research Background

Natural-language database querying aims to enable users to express query requirements in natural language, with the system automatically understanding query intent, matching the database structure, and generating Structured Query Language (SQL) statements. Traditional Text-to-SQL methods usually rely on specially designed encoders, decoders, or intermediate representations and use database schemas in a relatively fixed

© 2026 The Author(s). Authors retain copyright in their articles.

manner. With the development of large language models (LLMs), database querying has gradually shifted from task-specific model training to in-context learning and prompt engineering that rely on general language understanding. Existing studies show that the organization of schema information directly affects the model's judgment of query intent. In research on Chinese natural-language-to-SQL (NL2SQL), Yaodong Zheng et al. incorporated schema-link information into the encoding process to strengthen the association between natural-language questions and tables and fields [1]. Yumeng Liu et al. also noted in their review of intelligent synthesis techniques for structured database query languages that database structural information is an indispensable basic condition in converting natural-language queries [2].

Against this background, the role of schemas has changed significantly. Traditional database systems focus on whether a schema accurately describes the tables, fields, data types, and constraint relationships in a database, whereas LLMs must also understand the semantic role of these structures in a user's question. For example, when a user asks to "query the customer with the highest sales amount in the past three months," providing only field names such as customer_id and amount may not be sufficient. The model must also determine which fields correspond to the customer, sales time, and amount, and which relationships should be used to join different tables. Therefore, a schema is no longer only database metadata; it has become intermediate knowledge that connects a natural-language query with the actual database structure.

At the same time, schema size has become an important factor restricting the application of LLMs. When a database contains only a small number of tables and fields, the complete schema can be placed directly in a prompt. However, enterprise databases, hospital databases, and multi-business data platforms often contain many tables and fields. Directly entering all schema information into the model increases token consumption and inference cost and also expands the range of candidates that the model must distinguish. Therefore, how to reduce irrelevant schema information while ensuring the completeness of the information required for the task has gradually become an important issue in schema representation research.

1.2 Role of Schemas in Large Language Model Tasks

From the perspective of the Text-to-SQL process, a schema performs at least three types of functions. The first is structural identification, which tells the model which tables and fields the database contains and the field types. The second is semantic mapping, which helps the model map business concepts in a query to specific fields. The third is relational constraint, which helps the model determine join paths in SQL by using primary keys, foreign keys, and inter-table relationships. Different studies place different levels of emphasis on these three types of information. Natural-language- and prompt-based methods usually focus more on semantic descriptions and context organization [3,4,5], embedding and retrieval methods focus more on semantic relevance between a query and a schema [6,7], and structure-aware methods further use field dependencies and inter-table relationships to improve SQL generation quality [8,9].

Therefore, schema representation is essentially a problem of organizing information at multiple levels: it must express the objective structure of a database, highlight semantic information relevant to the current task, and control the input size. For an LLM, a complete schema is not necessarily an effective schema. What the model actually needs is schema context that is highly relevant to the current query, can support SQL generation, and does not cause excessive interference. Thus, schema representation and schema compression are two aspects of the same technical problem.

1.3 Main Problems in Schema Representation and Compression

The first problem facing existing research is the difficulty of unifying representation granularity. Table and field names are the most compact, but they may contain abbreviations and semantic ambiguities. Adding field comments can improve comprehensibility but increases text length. Adding instance values can help the model understand the actual contents of fields but may introduce substantial redundancy. Primary- and foreign-key relationships can supplement structural information but may form long relationship descriptions in complex databases. Therefore, more information is not always better; different types of information need to be selected according to the task.

The second problem is relevance judgment. The tables and fields that are actually relevant to a query usually account for only part of the complete schema, but relevance cannot depend entirely on string matching. Users may use natural-language business terms, whereas a database may use abbreviations, internal codes, or

© 2026 The Author(s). Authors retain copyright in their articles.

inconsistent naming. Embeddings can reduce lexical differences, but vector similarity may also ignore primary- and foreign-key relationships and compositional semantics in complex queries. Therefore, pure semantic retrieval and structural relationships need to be further coordinated.

The third problem is information loss after compression. If schema compression only pursues a reduction in the number of tokens, it may mistakenly remove key fields or join relationships required for SQL generation. If too much information is retained to ensure accuracy, the value of compression is difficult to demonstrate. Existing research has gradually developed from complete-schema input to retrieval, Top-K selection, and dynamic compression, but a unified solution is still lacking for automatically determining the retained range according to query complexity, database structure, and model capability.

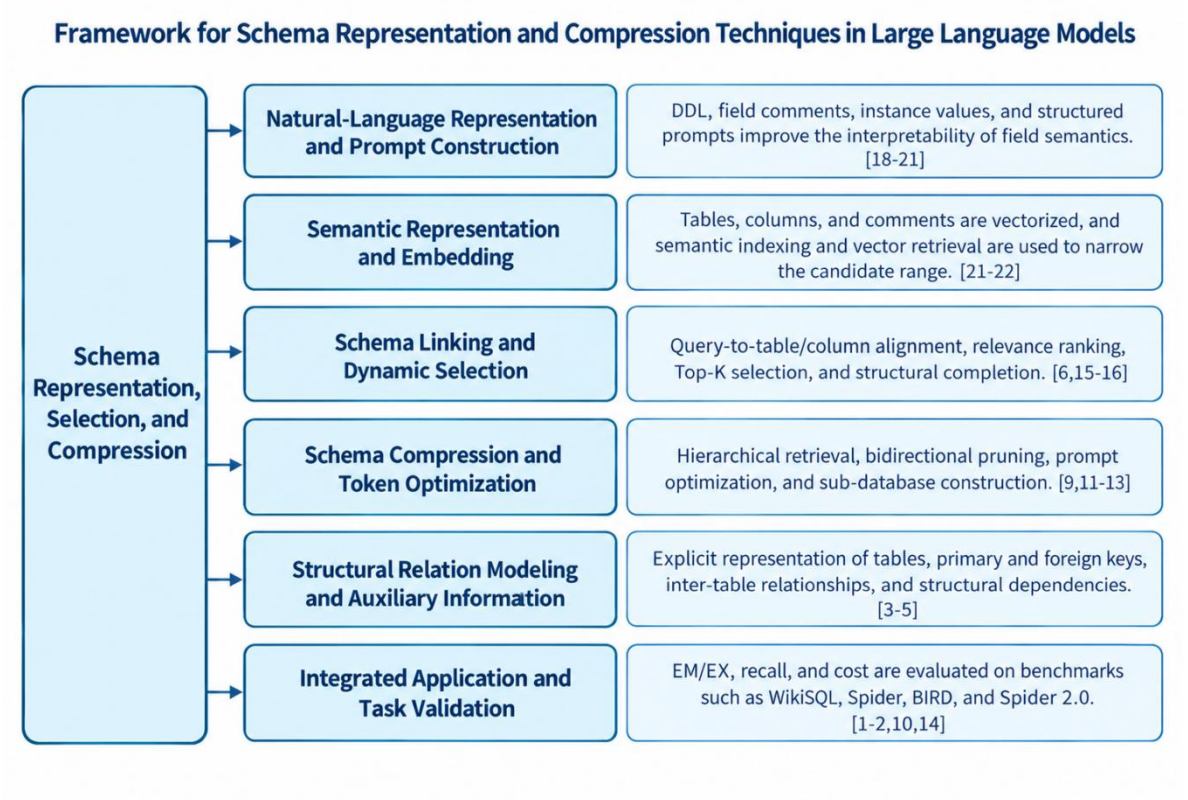
1.4 Research Content and Organization of This Paper

This paper classifies the research into six directions: natural-language representation and prompt construction, semantic representation and embedding, relevance modeling and dynamic selection, schema compression and token optimization, structural relation modeling and auxiliary information enhancement, and LLM-driven Text-to-SQL applications.

This paper successively introduces the theoretical foundations, six categories of key techniques, cross-method comparisons, and future directions. It focuses on distinguishing schema encoding, candidate screening, and context compression to avoid conflating methods and metrics at different levels.

The main contributions of this paper are as follows: it establishes a classification framework covering representation, retrieval, screening, compression, and structural constraints; discusses benchmark evolution through WikiSQL, Spider, BIRD, and Spider 2.0; and proposes evaluating compression effectiveness from four aspects: input size, schema fidelity, downstream quality, and end-to-end cost.

Figure 1: Classification System for Six Research Directions in Schema Representation and Compression (Drawn Based on the Relevant Literature)



© 2026 The Author(s). Authors retain copyright in their articles.

2. Theoretical and Technical Foundations of Schema Representation

2.1 Basic Concepts and Information Components of a Schema

A schema describes the logical structure of a database, including tables, fields, data types, primary keys, foreign keys, and constraints. In Text-to-SQL, it also serves as a semantic bridge between natural language and the database structure and usually needs to be converted into text, vectors, or model context.

Schema information can be divided into basic structural information (table names, field names, and data types) and auxiliary semantic information (field comments, instance values, primary and foreign keys, and inter-table relationships). Field names are compact but may have unclear meanings. Comments and instance values facilitate understanding but increase token use, so they need to be selected according to the task.

2.2 Main Forms of Schema Representation

Common schema representations include structured text, natural-language descriptions, vector representations, and retrieved local schemas. Data Definition Language (DDL) has a clear and compact structure; natural language facilitates the interpretation of business semantics; vectors are suitable for similarity retrieval; and a local schema retains a relevant subset according to the query.

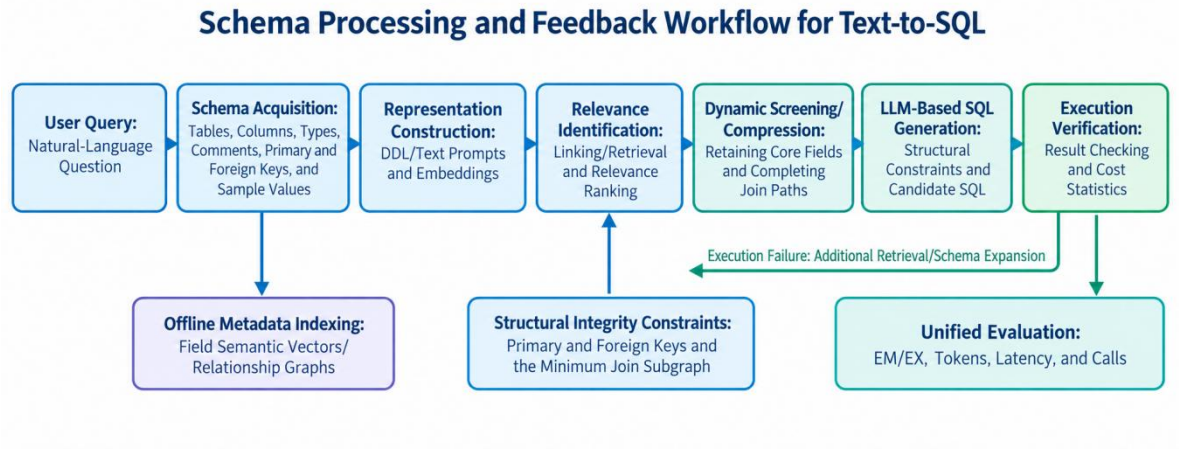
These forms can be used in a cascade: schema text is first generated from structured metadata, embeddings are then used to retrieve candidates, and finally the relevant fields and necessary relationships are organized into a prompt. Therefore, they are different levels of the processing workflow rather than mutually exclusive schemes.

2.3 Schema Processing Workflow in Large Language Models

The participation of a schema in Text-to-SQL usually includes metadata acquisition, representation construction, query understanding, relevance judgment, context organization, and SQL generation. The system first obtains tables, fields, and relationships; selects DDL, natural-language, or hybrid representations; screens relevant items and organizes the input; and, when necessary, makes corrections through feedback from execution verification.

Compression effectiveness must be evaluated together with the downstream task. If tokens are reduced but key fields or join relationships are omitted, the compression is unsuccessful. Practical value is demonstrated only when SQL performance is maintained while irrelevant schema information is reduced.

Figure 2: Schema Processing and Feedback Workflow for Text-to-SQL (Drawn Based on the Relevant Literature)



2.4 Schema Redundancy and Compression

Schema redundancy includes structural redundancy (tables and fields that the current query will not access), semantic redundancy (repeated descriptions), and task-irrelevant redundancy. All of these increase context and disperse the model's attention.

© 2026 The Author(s). Authors retain copyright in their articles.

Therefore, schema compression should reduce irrelevant context while retaining the information necessary for the task, rather than simply reducing characters or tokens. Related research has gradually shifted from static complete-schema input to retrieval, matching, and task-driven dynamic input [6,7,2,10,11,12].

WikiSQL represents single-table tasks with restricted SQL; Spider extends to complex multi-table queries across databases; BIRD emphasizes large-scale databases and real data values; and Spider 2.0 targets enterprise-level, multi-step SQL workflows. These benchmarks reflect the continuing increase in schema size and task complexity.

Table 1: Comparison of Typical Text-to-SQL Benchmark Datasets

Benchmark (Reference)	Release	Scale	Schema/Task Characteristics	Evaluation Focus
WikiSQL [13]	2017	80,654 question-SQL pairs; 24,241 Wikipedia tables	Primarily single-table; restricted SQL forms	Suitable for validating basic NL2SQL and column selection
Spider [14]	2018	10,181 questions; 5,693 complex SQL statements; 200 databases	Cross-database, multi-table joins and nesting; covers 138 domains	Development set of 1,034 questions: Easy 248, Medium 446, Hard 174, Extra Hard 166
BIRD [15]	2023	12,751 question-SQL pairs; 95 databases; approximately 33.4 GB; 37 domains	Large-scale real databases; emphasizes database content values and query efficiency	Development set of 1,534 questions: Simple 925, Moderate 465, Challenging 144
Spider 2.0 [16]	2025	632 enterprise-level Text-to-SQL workflow questions	Common databases contain more than 1,000 columns; involves BigQuery/Snowflake and long workflows	Emphasizes enterprise-level multi-step workflows; does not use the four difficulty levels of Spider 1.0

Note: The numbers at each difficulty level for Spider and BIRD are statistics from the development sets. Spider 2.0 targets enterprise workflows and does not use the four difficulty levels of Spider 1.0.

Benchmark evolution shows that the focus of schema processing has shifted from encoding the complete structure to activating relevant parts while retaining the necessary query paths. WikiSQL focuses on column matching, Spider examines cross-database generalization, BIRD emphasizes understanding data values, and Spider 2.0 adds the challenges of enterprise metadata and long workflows [13,14,15,16].

3. Natural-Language-Based Schema Representation and Prompt Construction

3.1 Natural-Language and Structured-Text Descriptions of Schemas

Natural language or structured text is the direct entry point through which an LLM understands a schema. Original DDL is intended for database systems, and field names and types may not fully express business semantics. Comments or descriptions therefore often need to be added to reduce differences involving abbreviations, synonyms, and business terms.

Table and field names provide basic entity and attribute information, but real databases may use abbreviations or internal codes. Field comments can explain business meanings. The multi-level indexing method of Junhui Kang and Jianbin Wang also demonstrates the value of jointly organizing databases, tables, fields, and comments [6].

Instance values can supplement field semantics. For example, the values “completed” and “processing” in a status field can help determine its meaning. However, a large number of instance values increases the input and introduces irrelevant information, so values should be selected on demand rather than always being entered in full.

© 2026 The Author(s). Authors retain copyright in their articles.

3.2 Auxiliary Schema Information and Semantic Organization

Cross-table queries require not only field identification but also determination of the join method. Primary keys, foreign keys, and inter-table relationships can constrain join paths. Research that incorporates schema-link information into encoding also demonstrates the role of structural associations in mapping a query to database elements [1].

A schema prompt can combine four levels of information: table and field names, types and comments, instance values, and primary-key, foreign-key, and inter-table relationships. Simple queries usually do not require all information to be enabled, whereas complex multi-table queries should retain necessary relationships. Prompt organization is itself a process of selecting information according to the task.

3.3 Schema Organization in Prompt Engineering

Prompt engineering emphasizes the organization of task instructions, context, and reasoning steps. The study by Jun Huang et al. noted that prompt design involves task descriptions, context organization, and other components [3]. In Text-to-SQL, a clear hierarchical structure helps the model distinguish instructions, the schema, and the query.

A structured prompt can organize the input as "task requirements-database structure-user query-output format." Related research on chain-of-thought prompting also supports the explicit presentation of task steps [4], but a complex prompt does not mean that text should be added without limit.

3.4 Comparison of Different Schema Text Representations

A complete schema provides comprehensive information, but its token cost rises as the database grows. A natural-language-enhanced representation can reduce field ambiguity but adds text. A task-relevant schema uses less context but depends on screening quality.

The three approaches form a continuum from completeness to efficiency: a complete schema emphasizes recall, natural-language enhancement emphasizes comprehensibility, and a local schema emphasizes efficiency. A practical system can first retrieve relevant items and then add comments and relationships as needed.

3.5 Section Summary

This section shows that natural-language representation and structured prompts can improve schema comprehensibility but remain limited by input size and redundancy. Large-scale databases need to combine embeddings, relevance modeling, and dynamic compression.

4. Semantic Schema Representation and Embedding Methods

4.1 Basic Idea of Schema Embeddings

Embeddings map queries, tables, and fields into a vector space and retrieve relevant schema information by semantic similarity. This approach reduces lexical differences between natural-language expressions and database naming and lowers the context cost of full-text matching.

Embedding objects can include table names, field names, comments, instance values, or schema fragments. Multi-level indexing organizes databases, tables, fields, and comments hierarchically, enabling the system to narrow the candidate range of databases and tables before locating specific fields [6].

Vector embeddings focus on semantic retrieval, whereas RAT-SQL, SDSQL, and LGESQL explicitly encode tables and columns, primary and foreign keys, and dependencies among elements. The latter are not token compression algorithms, but they can help determine whether the necessary join paths are still retained after screening [17,18,19].

© 2026 The Author(s). Authors retain copyright in their articles.

4.2 Alignment of the Semantic Spaces of Queries and Schemas

A query and a schema may use different terms. For example, a user may say “customer purchase amount,” whereas the database field is named purchase_amount. Embeddings can establish a connection at the overall semantic level and compensate for the limitations of pure string matching.

Haifeng He et al. used an LLM, embeddings, and retrieval-augmented generation (RAG) for intelligent hospital operations data querying and supported query generation by retrieving metadata [7]. This shows that embeddings are mainly responsible for candidate discovery and context construction rather than replacing SQL generation.

4.3 Embedding-Based Schema Retrieval

A common workflow is to encode a schema and build an index offline, and then encode a query online to retrieve candidate tables and columns, avoiding the need to enter the complete database each time. This two-stage architecture is particularly valuable for large-scale databases.

Pure vector retrieval still has limitations: field-level similarity may miss compositional semantics, semantic relevance does not equal SQL usability, and vector distance has difficulty expressing primary- and foreign-key constraints. Structural information is therefore usually required for secondary screening.

4.4 Comparison of Different Semantic Representation Methods

Field-level embeddings facilitate precise retrieval but can easily lose table-level context. Table-level representations retain entity semantics but may introduce irrelevant fields. Schema-fragment-level representations can be used directly as context but have higher encoding and indexing costs.

Embeddings can be used for one-time Top-K retrieval or multi-stage retrieval. The latter first recalls candidates and then combines rules, structural relationships, or an LLM for fine screening, making it more suitable for complex multi-table queries.

4.5 Limitations and Development Trends of Embedding Methods

Embedding effectiveness depends on the representation model, indexing granularity, and candidate strategy. Similar fields may cause false recalls, and implicit business logic is also difficult to express completely through similarity. Semantic and structural information therefore need to be modeled jointly.

5. Schema Relevance Modeling and Dynamic Selection

5.1 Basic Problem of Query-Schema Relevance

Relevance modeling identifies tables, fields, and relationships related to the current query from the complete database and is a prerequisite for schema compression. Irrelevant items in a large-scale database increase the burden on the model.

Relevance includes at least three dimensions: lexical, semantic, and structural. Lexical matching focuses on name overlap, semantic matching focuses on similar meanings, and structural matching determines whether candidate items can jointly support the query through relationships.

5.2 Schema Linking and Relevance Matching

Schema linking maps entities, attributes, and conditions in a query to database tables and columns. It can be completed by an explicit algorithm or judged by an LLM. Related Chinese NL2SQL research strengthens the association between a query and tables and columns through schema-link information [1].

Schema linking answers “which database elements correspond to the concepts in the query,” whereas dynamic selection determines “which elements enter the model input” while considering the context budget. The former provides mappings, and the latter further implements screening.

© 2026 The Author(s). Authors retain copyright in their articles.

RESDSQL connects schema linking and SQL skeleton parsing through relevance ranking [20]. RSL-SQL further uses bidirectional pruning and context supplementation to reduce the risk of omitting key fields [21]. The algorithms and experimental metrics of both methods are compared uniformly in Section VI.

5.3 Schema Screening Based on Relevance Ranking

Candidate schemas are usually ranked according to embedding similarity, keyword matching, field types, and inter-table relationships, after which Top-K tables, fields, or fragments are selected. Ranking can be adjusted according to the context budget more easily than direct deletion.

A fixed K value has difficulty adapting to query complexity. A simple single-table query requires only a few fields, whereas a complex multi-table query requires more fields and join relationships. A K that is too small loses information, whereas a K that is too large weakens compression effectiveness.

5.4 Query-Driven Dynamic Schema Selection

Dynamic selection determines the schema range in real time according to the current query. The workflow includes requirement understanding, candidate retrieval, relevance ranking, and selection within a budget, enabling the same database to construct different contexts for different queries.

Dynamic selection must also maintain structural completeness. A highly relevant field still cannot support correct SQL generation if its table or the foreign key required for a join is missing. Therefore, the selection objective should be the necessary fields and relationships that support the query, rather than isolated Top-K fields.

5.5 Limitations of Relevance Modeling

Current relevance modeling still has difficulty completely distinguishing semantic relevance from SQL usability and expressing multi-hop relationships through single-field similarity. Evaluations should jointly report schema recall, SQL accuracy, token compression ratio, and inference latency.

6. Schema Compression and Token Optimization

6.1 Operational Definition and Evaluation Metrics of Schema Compression

This paper defines schema compression as follows: given a complete schema S and a query q, construct a subschema S′ without significantly reducing SQL correctness, thereby reducing the number of input tokens, tables, or columns. The token compression ratio is CR=(T_full−T_keep)/T_full×100%; field pruning can be reported by the column retention/reduction ratio, and task quality should be evaluated together with schema linking recall, exact match (EM), execution accuracy (EX), latency, and the number of LLM calls.

6.2 Selective Compression Based on Relevance Ranking

RESDSQL ranks tables and columns through Ranking-enhanced Encoding and uses Skeleton-aware Decoding to first generate an SQL skeleton and then fill in fields [20]. RESDSQL-3B+NatSQL achieves 80.5% EM and 84.1% EX on the Spider development set and 72.0% EM and 79.9% EX on the test set. The original paper does not provide a uniformly comparable token compression ratio, and execution accuracy cannot replace the compression ratio.

6.3 Retrieval-Driven Hierarchical Compression and Query-Driven Filtering

CHESS organizes retrieval, schema selection, candidate generation, and unit testing into a multi-stage workflow. Public results show that the Schema Selector reduces LLM token use to approximately one-fifth of the original amount and improves overall accuracy by approximately 2%. Under a high-compute budget, its execution accuracy on the BIRD test set is 71.10%, and the number of LLM calls is reduced by approximately 83% [22]. CodeS incorporates query-driven schema filtering into the workflow of open-source SQL models and combines column names and comments with coarse-to-fine BM25 retrieval [23]. Its uniform cross-model compression ratio has not been fully disclosed.

© 2026 The Author(s). Authors retain copyright in their articles.

6.4 Bidirectional Pruning and Robust Selection Under Structural Constraints

RSL-SQL identifies relevant tables and columns through forward pruning and then uses backward pruning to check whether the candidates can form a usable query path. It combines binary selection, context enhancement, and multi-round self-correction to reduce the risk of excessive pruning [21]. The paper reports an 83% reduction in the number of input columns and a strict schema linking recall of 94%. Under the GPT-4o setting, its execution accuracies on BIRD and Spider are 67.2% and 87.9%, respectively. Column reduction ratio, recall, and execution accuracy have different meanings and cannot replace one another.

6.5 Prompt-Level Context Optimization and Task-Level Sub-Database Construction

DAIL-SQL optimizes question representation, example selection, and organization and uses SQL skeleton similarity and compact structural knowledge to reduce the overall prompt cost. It is prompt-level optimization rather than pure schema pruning [24]. Its prompt on the Spider development set uses approximately 1,600 tokens per question, and its execution accuracy on the test set is 86.6%. MAC-SQL uses multiple agents to obtain a local sub-database and corrects SQL according to error feedback. Under the GPT-4 setting, its execution accuracy on the BIRD test set is 59.59% [25]. The two methods optimize different objects, and their cost measures should not be compared directly.

6.6 Comparison and Applicable Boundaries of Different Compression Mechanisms

The methods act at different levels: RESDSQL performs ranking, CHESS combines retrieval and testing, CodeS integrates filtering, RSL-SQL emphasizes bidirectional pruning, DAIL-SQL optimizes prompts, and MAC-SQL constructs sub-databases. A mechanism should be selected according to the compression object, and the results should not be combined into a single compression ratio.

6.7 Unified Evaluation Framework for End-to-End Tasks

Compression should be evaluated from four aspects: input size (tokens, number of tables, and number of columns); schema fidelity (required-field recall, strict linking recall, and join-path completeness); downstream quality (EM and EX); and end-to-end cost (retrieval and generation time, number of LLM calls, and inference cost). Reporting only token reduction cannot reveal information loss, and reporting only EX cannot prove that the benefit comes from compression.

(1) **Token Compression Rate:** The proportion of the number of tokens retained after compression to the original number of tokens.

$$CR_{\text{token}} = \frac{T_{\text{full}} - T_{\text{keep}}}{T_{\text{full}}} \times 100\% \tag{1}$$

(2) **Column Reduction Rate:** The proportion of the number of columns retained after compression to the original number of columns.

$$CR_{\text{col}} = \frac{C_{\text{full}} - C_{\text{keep}}}{C_{\text{full}}} \times 100\% \tag{2}$$

(3) **Table Retention Rate:** The proportion of the number of tables retained after compression to the original number of tables.

$$R_{\text{col}} = \frac{C_{\text{keep}}}{C_{\text{full}}} \times 100\% \tag{3}$$

(4) **Relationship Link Retention Rate:** The proportion of the number of primary–foreign key relationships retained after compression to the original number of relationships.

$$R_{\text{link}} = \frac{|G_{\text{gold}} \cap G_{\text{pred}}|}{|G_{\text{gold}}|} \times 100\% \tag{4}$$

(5) **Execution Accuracy:** The proportion of generated SQL queries that are correctly executed on the query set.

$$EX = \frac{N_{\text{correct-execution}}}{N_{\text{queries}}} \times 100\% \tag{5}$$

Here, T_full/T_keep denote the numbers of tokens before and after compression, respectively; C_full/C_keep denote the numbers of columns; G_gold is the required schema set; and G_pred is the retained set. EM measures exact SQL matching, and EX measures the correctness of execution results. Metrics not

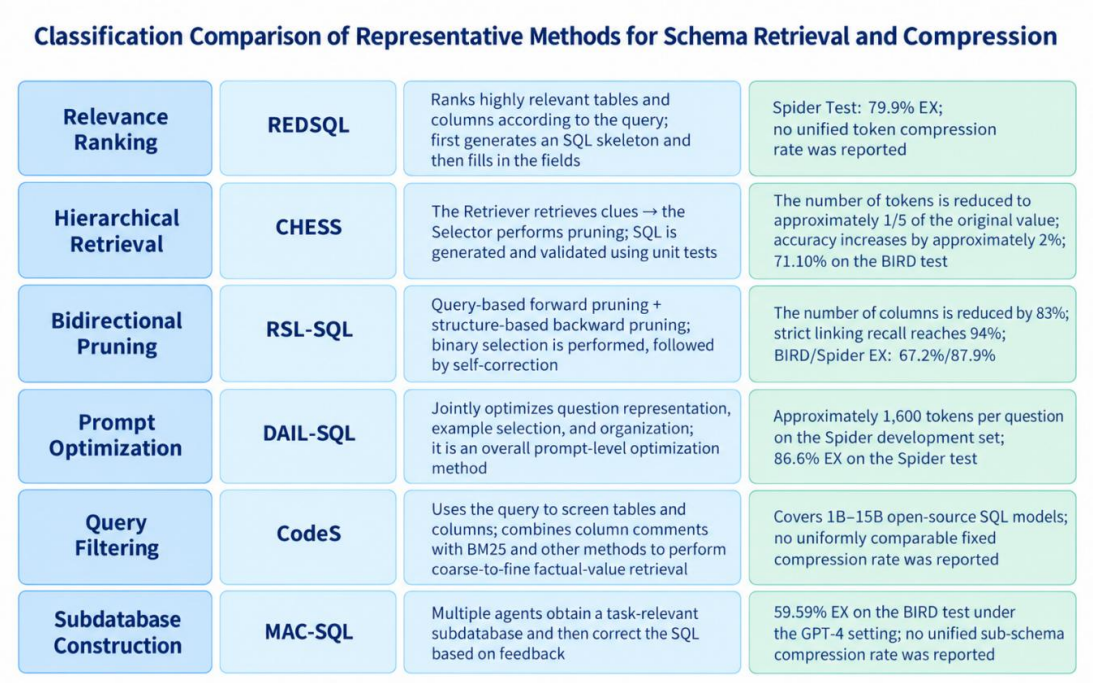
© 2026 The Author(s). Authors retain copyright in their articles.

reported in the original papers are not fabricated and should be marked as "not reported/not directly comparable."

Table 2: Comparison of Representative Schema Retrieval, Screening, and Compression Methods

Method (Category)	Compression Object and Core Mechanism	Publicly Reported Task Performance	Efficiency Evidence and Scope
RESDSQL (relevance ranking)	Ranks tables and columns by relevance; generates an SQL skeleton before filling in fields.	Spider test: 72.0% EM; 79.9% EX.	No uniform token compression ratio is reported; EX cannot be treated as a compression ratio.
CHESS (hierarchical retrieval)	Retrieves clues, performs stepwise pruning, and verifies through generation and unit testing.	BIRD test under the high-budget setting: 71.10% accuracy.	Tokens reduced to approximately 1/5; calls reduced by approximately 83%; accuracy increased by approximately 2%.
RSL-SQL (bidirectional pruning)	Forward and structural backward pruning, binary selection, and self-correction.	GPT-4o: BIRD 67.2% EX; Spider 87.9% EX.	Columns reduced by 83%; strict linking recall of 94%; model setting is specified.
DAIL-SQL (prompt optimization)	Optimizes question representation and example organization and compresses the overall prompt.	Spider test: 86.6% EX.	Approximately 1,600 tokens/question on the Spider development set; not pure schema pruning.
CodeS (query filtering)	Screens tables and columns according to the query and combines comments with BM25 retrieval.	Evaluates open-source SQL models from 1B to 15B and Spider/BIRD.	No uniform cross-model compression ratio is given; original and retained sizes should be reported.
MAC-SQL (sub-database)	Constructs a sub-database with multiple agents and corrects SQL according to feedback.	BIRD test under the GPT-4 setting: 59.59% EX.	No uniform subschema compression ratio is reported; the cost of agent calls should also be counted.

Figure 3: Classification and Comparison of Representative Schema Retrieval and Compression Methods (Data Compiled from the Publicly Reported Results of the Papers)



© 2026 The Author(s). Authors retain copyright in their articles.

7. Modeling Schema Structural Relationships and Enhancing Auxiliary Information

7.1 Primary Keys, Foreign Keys, and Inter-Table Relationships

A plain-text schema has difficulty fully expressing the join methods of complex queries. Primary keys, foreign keys, and inter-table relationships provide structural constraints for multi-table SQL and can also support compression: necessary related tables are expanded along relationships from relevant fields to retain the minimum usable query path.

Relational constraints change compression from simple field deletion to structural subgraph selection and can retain the join paths required by a query in a smaller context.

7.2 Dependency-Based Structure-Aware Methods

Qing Shu et al. used a dependency-graph attention network to enhance SQL generation [8], indicating that database dependencies can be modeled explicitly rather than relying entirely on an LLM to infer them from text.

Structure-aware methods pay more attention than pure embeddings to how schema elements jointly participate in a query and are also more explicit than pure prompts. Their cost is the need to maintain relationship representations, and system updates become more complex when the database structure changes.

RAT-SQL uses relation-aware self-attention to encode tables and columns, primary and foreign keys, and query-column relationships [17]. SDSQL learns dependencies among elements by using Schema Dependency as an auxiliary task [18]. LGESQL uses line-graph propagation and graph pruning to improve encoding discriminability [19]. Compression should retain the minimum relational subgraph that supports SQL generation rather than retaining only relevant field names.

7.3 Auxiliary Information Enhancement and Domain Knowledge

Real-world business databases often require field comments, business rules, instance values, and domain knowledge to supplement semantics. Research on knowledge management and domain knowledge graphs also shows that external knowledge can help models understand specialized information [11,26].

Auxiliary information should likewise be screened according to the query. Only information that can improve field identification, relationship judgment, or SQL generation is worth retaining. The richer the knowledge, the greater the requirements for retrieval and compression.

7.4 Limitations of Structure Enhancement Methods

The limitations of structure enhancement are complexity and generalization. Enterprise databases may have weak constraints, implicit relationships, or missing foreign keys, and database definitions alone have difficulty covering business logic. Excessive reliance on explicit relationships may also overlook implicit semantics.

A more feasible direction is joint modeling: embeddings discover semantic relevance, structural relationships constrain query paths, and an LLM performs integrated understanding and generates SQL.

8. Combined Role of Schema Techniques in Text-to-SQL and Task Validation

8.1 Integrated Application Pattern of the Schema Processing Chain

From the perspective of the complete task chain, the input stage organizes tables and columns, types, comments, and relationships. The intermediate stage recalls candidates through embeddings or schema linking and then reduces the input through ranking, pruning, or sub-database construction. The generation stage retains necessary join paths and uses execution verification to check whether screening has caused information loss.

This paper includes in the comparison methods that directly change schema representation or the input range: RESDSQL, CHESS, RSL-SQL, CodeS, and MAC-SQL use ranking, staged retrieval, bidirectional pruning, query filtering, and local sub-databases, respectively. DAIL-SQL is used as a reference for overall prompt optimization [20,24,25,23,22,21].

© 2026 The Author(s). Authors retain copyright in their articles.

DIN-SQL and C3 improve Text-to-SQL performance through task decomposition, prompt calibration, and structured output, but they are not specialized schema compression algorithms. They are therefore discussed as task organization methods and are not directly compared with column pruning or token compression [27,28].

DIN-SQL decomposes the task into schema linking, query classification/decomposition, SQL generation, and self-correction. It achieves 85.3% EX on the Spider test set and 55.9% on the BIRD test set [27]. C3 builds a zero-shot workflow through Clear Prompting, Hints calibration, and consistent output and achieves 82.3% EX on the Spider test set [28]. These methods embody the approach of first organizing and constraining the input and then generating SQL.

8.2 Influence of Schema Representation on SQL Generation

Schema representation affects field identification, SQL structure, and the understanding of business semantics. Missing key fields directly restricts generation, missing foreign keys or table relationships may cause join errors, and ambiguity in field naming also affects the judgment of business meanings.

8.3 Coordination of Schema Representation, Retrieval, and Compression

The coordinated workflow can be summarized as follows: text descriptions supplement semantics, retrieval recalls candidates, schema linking establishes mappings, structural constraints complete join paths, and the compression module selects the retained set according to the budget. When execution fails, incremental retrieval is performed.

8.4 Typical Application Scenarios

Intelligent data querying has combined LLMs, embeddings, and RAG to retrieve database metadata and support hospital operations queries [7]. Research on multi-level indexing in chemical enterprises demonstrates the need to organize schemas hierarchically in specialized databases [6].

Spatial databases contain special data types and domain operations. Related research shows that schema representation needs to extend from fields in general relational tables to specialized semantics and domain constraints [12].

8.5 Comprehensive Evaluation and Task Validation

Existing studies often report SQL accuracy, execution results, and system efficiency. Schema compression research should also disclose the tokens, candidate schema size, and inference latency before and after compression to determine whether the same task is completed with less context.

9. Comprehensive Comparison and Problem Analysis of Existing Research

9.1 Characteristics of Different Schema Representation Methods

Table 3 summarizes representative methods and distinguishes structural encoding foundations, direct compression methods, and LLM task organization methods.

Table 3: Summary and Comparison of Representative Methods in the Six Research Directions

Research Direction	Representative Methods/Benchmarks	Core Schema Processing Mechanism	Main Conclusion
Natural-language representation and prompt construction	Structured DDL, field comments/instance values; prompt organization [3,4,5,6]	Organizes tables and columns hierarchically and adds comments, examples, and relationships as needed	Highly interpretable, but natural-language supplements increase tokens, so redundancy must be controlled
Semantic representation and embedding	Multi-level indexing, embedding + RAG [6,7]	Encodes tables, columns, comments, or instance values, with coarse retrieval followed by fine screening	Reduces lexical differences, but pure vector similarity still requires structural constraints

© 2026 The Author(s). Authors retain copyright in their articles.

Research Direction	Representative Methods/Benchmarks	Core Schema Processing Mechanism	Main Conclusion
Schema linking and dynamic selection	Query-table/column alignment, relevance ranking, Top-K dynamic selection, and structural completion [20,21,29]	Query-table/column alignment, ranking, bidirectional pruning, and self-correction	RSL-SQL reports an 83% column reduction and 94% strict recall; RESDSQL has 79.9% EX on the Spider test set
Schema compression and token optimization	CHESS, DAIL-SQL, CodeS, MAC-SQL [24,25,23,22]	Hierarchical retrieval, prompt optimization, query filtering, and sub-database construction	CHESS reduces tokens to approximately 1/5; other efficiency metrics use different measures and cannot be compared directly
Structural relation modeling and auxiliary information	RAT-SQL, SDSQL, LGESQL [17,18,19]	Relation-aware attention, schema dependency learning, line graphs, and graph pruning	Provides a basis for retaining necessary join paths after compression; field-level semantic similarity alone is insufficient
Integrated application and task validation	WikiSQL, Spider, BIRD, Spider 2.0 [13,14,15,16]	Validation on benchmarks ranging from single-table tasks to cross-database, multi-table, real-value, and enterprise workflows	As tasks become more complex, schema recall, EX/EM, tokens, and end-to-end cost need to be reported together

9.2 Relationship Between Schema Compression Ratio and Task Performance

Schema compression and SQL performance usually have a nonlinear relationship. Removing irrelevant items at the initial stage has little effect on performance, whereas further compression may mistakenly remove necessary information. Therefore, performance changes under different compression ratios should be examined rather than a single compression point.

Public results show that high compression does not necessarily reduce performance. CHESS reports that token use is reduced to approximately one-fifth and accuracy improves by approximately 2%. RSL-SQL retains a strict linking recall of 94% when the number of columns is reduced by 83%. However, these metrics come from different tasks and settings and cannot be ranked directly.

9.3 Dynamic K Values and Adaptive Selection

A fixed K has difficulty adapting to different queries: complex queries usually require more fields and relationships, whereas simple queries can use a smaller context.

9.4 Token Overhead and Computational Efficiency

Efficiency cannot be judged only by LLM input tokens. Embeddings, retrieval, structural graph construction, and reranking all incur costs. An overly heavy front-end module may reduce tokens but increase total latency. End-to-end time and total cost should be reported.

9.5 Interpretability and Generalization

Dynamic selection should explain the basis for retaining or deleting fields and verify generalization in scenarios involving irregular naming, implicit rules, and missing foreign keys.

10. Future Research Directions

10.1 Adaptive Schema Retrieval and Dynamic K Values

For enterprise-level long-schema scenarios such as Spider 2.0, dynamic K values should jointly consider join depth, candidate-field scores, and execution feedback. Simple queries should use a small K, multi-hop or

© 2026 The Author(s). Authors retain copyright in their articles.

high-uncertainty queries should expand the candidate range, and schema information should be supplemented when verification fails.

10.2 Efficient Schema Representation Combining Semantics and Structure

Future research can combine text, vectors, and graph-structured representations: text supplements semantics, embeddings perform retrieval, and graph structures retain relationships. Multi-level indexes can be built in advance for frequently used databases, and only relevant parts need to be activated at query time.

10.3 Closed Loop of Schema Retrieval, Compression, and Agents

A database agent can form a closed loop of schema acquisition, retrieval, SQL generation, execution verification, and error correction, and can incrementally supplement fields or relationships according to execution feedback, avoiding irreversible information loss caused by one-time compression.

10.4 Schema Representation for Complex Databases

Spatial databases, knowledge bases, and specialized business databases have special data types, hierarchical relationships, and domain rules. Related research [12,9] indicates that schema representation should extend from traditional table fields to domain operations and constraints.

10.5 Comprehensive Evaluation System for Real-World Applications

The evaluation system should simultaneously measure accuracy, schema recall, compression ratio, tokens, latency, cost, and interpretability, and should distinguish between “shorter input” and “a genuinely faster system.”

11. Conclusion

This paper reviews the main routes of schema representation and compression in LLM environments. Research shows that a schema has become a key context connecting a natural-language query and the database structure. Effective methods need to coordinate semantic descriptions, relevance retrieval, structural constraints, and dynamic compression. In addition, tree-based models and intelligent database natural-language query parsing architectures provide supplementary perspectives from query-structure organization and parsing workflows [30,31].

References

[1] Yaodong Zheng, Xufeng Li, Heping Chen. Research on the Application of Chinese NL2SQL Technology Based on Deep Learning Models[J]. Microcomputer Applications, 2025, 41(04): 295-298, 303.

[2] Yumeng Liu, Yijing Zhao, Bicong Wang, et al. Research Progress in Intelligent Synthesis Technology for Structured Database Query Languages[J]. Computer Science, 2024, 51(07): 40-48.

[3] Jun Huang, Fei Lin, Jing Yang, et al. Large-Model Prompt Engineering for Generative Artificial Intelligence: Methods, Current Status, and Prospects[J]. Chinese Journal of Intelligent Science and Technology, 2024, 6(02): 115-133.

[4] Mengke Chen, Yun Bian, Yunhao Liang, et al. WH-CoT: A 6W2H-Based Chain-of-Thought Prompting Framework for Large Language Models[J]. Journal of Computer Applications, 2024, 44(Suppl. 2): 1-6.

[5] Lu Rong. Research on Large Language Model Prompt Learning for Automatic Book Classification[J]. Research on Library Science, 2024(01): 86-103.

[6] Junhui Kang, Jianbin Wang. Research on an Intelligent Query Method for Special-Operation Information in Chemical Enterprises Based on Multi-Level Indexing[J]. Safety, Health & Environment, 2026, 26(06): 28-34.

© 2026 The Author(s). Authors retain copyright in their articles.

[7] Haifeng He, Jide Xu, Han Lü. Construction of an Intelligent Data Querying System Based on Large Language Models and Its Application in Hospital Operations Management[J]. China Medical Equipment, 2026, 23(07): 77-83.

[8] Qing Shu, Xiping Liu, Zhao Tan, et al. An SQL Generation Method Based on a Dependency-Graph Attention Network[J]. Journal of Zhejiang University (Engineering Science), 2024, 58(05): 908-917.

[9] Yu Fu, Haoran Li. SQL Generation Based on Structure Awareness and Monte Carlo Tree Search[J]. Computer Technology and Development, 2026, 36(03): 118-123, 117.

[10] Yuhan Peng, Shaojie Qiao, Qi Xue, et al. A Review of Text Information Processing Techniques for Text-to-SQL[J]. Radio Engineering, 2024, 54(05): 1053-1062.

[11] Yang Zhou, Peihan Cai, Zhenjiang Dong. Large-Model Knowledge Management System[J]. ZTE Technology Journal, 2024, 30(02): 63-71.

[12] Mengyi Liu, Jianqiu Xu, Yongxin Tong. Natural-Language Query Conversion for Spatial Databases Based on Large Language Models[J]. Journal of Software, 2026, 37(03): 1121-1142. DOI: 10.13328/j.cnki.jos.007514.

[13] V. Zhong, C. Xiong, R. Socher. Seq2SQL: Generating Structured Queries from Natural Language Using Reinforcement Learning[EB/OL]. arXiv:1709.00103, 2017.

[14] T. Yu, R. Zhang, K. Yang, et al. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task[C]//Proceedings of EMNLP. 2018: 3911-3921.

[15] J. Li, B. Hui, G. Qu, et al. Can LLM Already Serve as a Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs[C]//Advances in Neural Information Processing Systems. 2023, 36.

[16] F. Lei, J. Chen, Y. Ye, et al. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows[C]//International Conference on Learning Representations. 2025.

[17] B. Wang, R. Shin, X. Liu, et al. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers[C]//Proceedings of ACL. 2020: 7567-7578.

[18] B. Hui, X. Shi, R. Geng, et al. Improving Text-to-SQL with Schema Dependency Learning[EB/OL]. arXiv:2103.04399, 2021.

[19] R. Cao, L. Chen, Z. Chen, et al. LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations[C]//Proceedings of ACL-IJCNLP. 2021: 2541-2555.

[20] H. Li, J. Zhang, C. Li, H. Chen. RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2023, 37(11): 13067-13075.

[21] Z. Cao, Y. Zheng, Z. Fan, X. Zhang, W. Chen, X. Bai. RSL-SQL: Robust Schema Linking in Text-to-SQL Generation[EB/OL]. arXiv:2411.00073, 2024.

[22] S. Talaei, M. Pourreza, Y. C. Chang, et al. CHESS: Contextual Harnessing for Efficient SQL Synthesis[EB/OL]. arXiv:2405.16755, 2024.

[23]Z. Li, S. Zhang, J. Liu, et al. CodeS: Towards Building Open-source Language Models for Text-to-SQL[J]. Proceedings of the ACM on Management of Data, 2024, 2(3): 127.

[24] D. Gao, H. Wang, Y. Li, et al. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation[J]. Proceedings of the VLDB Endowment, 2024, 17(5): 1132-1145. DOI: 10.14778/3641204.3641221.

[25] B. Wang, C. Ren, J. Yang, et al. MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL[EB/OL]. arXiv:2312.11242, 2023.

[26] Shengda Wu, Hui Yu. Research on a Hierarchical Prompt Engineering Method for Automatic Construction of Domain Knowledge Graphs[J]. Data Communications, 2026(04): 19-25.

[27] M. Pourreza, D. Rafiei. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction[C]//Advances in Neural Information Processing Systems. 2023.

© 2026 The Author(s). Authors retain copyright in their articles.

[28] X. Dong, C. Zhang, Y. Ge, et al. C3: Zero-shot Text-to-SQL with ChatGPT[EB/OL]. arXiv:2307.07306, 2023.

[29] Y. Gan, X. Chen, M. Purver. Re-appraising the Schema Linking for Text-to-SQL[C]//Findings of the Association for Computational Linguistics: ACL 2023. 2023: 835-852.

[30] Meng Zhao, Ke Chen, Lidan Shou, et al. Research on Converting Complex Natural-Language Queries to SQL Based on a Tree Model[J]. Journal of Software, 2022, 33(12): 4727-4745.

[31] Xiaoliang Cao. Design of a Natural-Language Query Parsing Architecture for Intelligent Databases[J]. Information Recording Materials, 2026, 27(18): 100-102. DOI: 10.16009/j.issn.1009-5624.2026.18.032.

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgment

This paper is an output of the science project.

Copyrights

Copyright for this article is retained by the author (s), with first publication rights granted to the journal. This is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).

© 2026 The Author(s). Authors retain copyright in their articles.