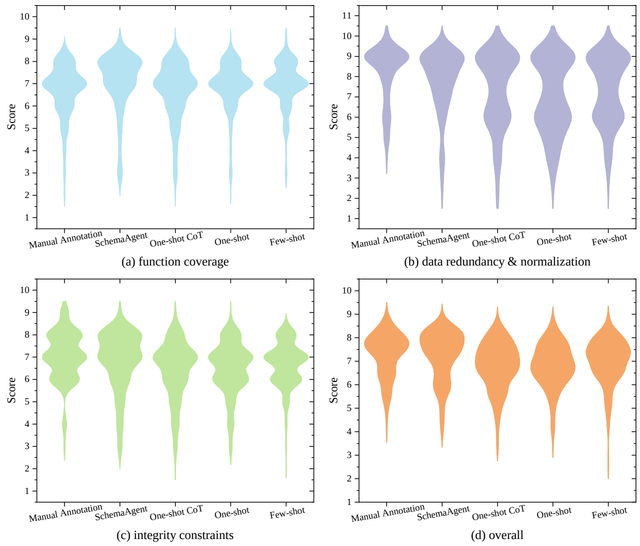
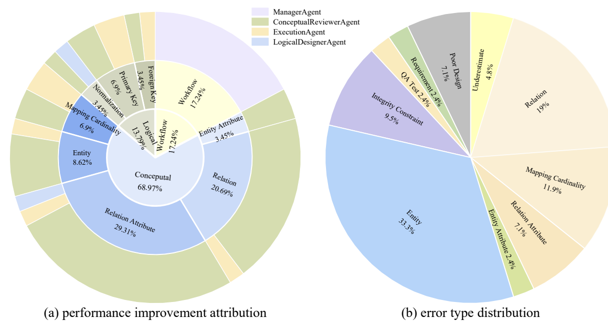


(a) function coverage

(b) data redundancy & normalization

(c) integrity constraints

(d) overall



(a) performance improvement attribution

(b) error type distribution

30%, respectively, reflecting the ease of user modification. Our analysis reveals that: (1) There is room for extensibility optimizations achieved by manual annotations, which are sometimes overlooked during manual process (e.g., setting an attribute instead of a table for expansion). (2) Overall, manual annotations perform strongly, validating the good quality of our dataset. In particular, scores for data redundancy and normalization are significantly higher for manual annotations, reflecting our design preference for consistency and elimination of redundancy. (3) SchemaAgent achieves the best results in all metrics in the baselines. Its strong performance in data redundancy and normalization further confirms our method's effectiveness in normal form control. (4) There exists some schemas from LLMs score higher than that from manual annotation, while, they account for a very small proportion, and we anticipate introducing extensibility optimizations in the future to address this issue.

6.6 Performance Enhancement and Error Analysis

To evaluate how SchemaAgent improves performance, we analyze 31 high-scoring cases (based on automated metrics and LLM-as-the-judge). Performance improvements are categorized into three main areas: workflow (without error feedback), conceptual model, and logical model. The conceptual model improvements involve enhanced recognition of entity, relation, and mapping cardinality. Logical model improvements cover primary/foreign key and normalization optimization. For each category, we trace the improvements back to the specific agents that provided feedback. Figure 10(a) shows 68.9% of improvements come from the improvements of the conceptual model, with the CMR agent contributing 85% of the feedback, and the LMD and TE agents provided the remaining ones. In 13.79% of the cases, with a correct conceptual model, LMD, TE, and CMR (and CMD) collaborate to optimize the

To further evaluate SchemaAgent, we also analyze 31 cases with low comprehensive scores, identifying ten error types. As shown in Figure 10(b), conceptual design as a critical phase in the database design shows significant room for improvement. (1) Entity errors included both overidentifying entities (e.g., creating a separate *recipient* table when recipients are already in the *users* table) and underidentifying entities, especially when entities have brief descriptions. (2) Relation errors involve missing relationships between entities. (3) Relation attribute errors primarily concern adding redundant auto-incrementing ID attributes to relationships. (4) Mapping cardinality errors often misclassify many-to-many relationships (e.g., between *supplier* and *product*) as one-to-many, particularly when the requirement description is ambiguous, challenging models' understanding of real-world scenarios. (5) Due to the inherent limitations of automatic evaluation, an underestimate is observed in 4.8% of the cases. (6) 7.1% of functionally complete schemas lack extensibility, leading to poor design. Furthermore, we observe that the model is more prone to errors when dealing with associative entities (entities with independent primary keys linked via foreign keys to other entities) and weak entities (entities whose existence depends on another entity). This suggests that the particular challenges in accurately discerning and representing complex entity relationships, especially those involving non-trivial primary key structures and existential dependencies, require advanced contextual understanding and external knowledge.

6.7 Case Study for Schema Generation

To demonstrate the practical effectiveness of our proposed SchemaAgent framework, we present a case study in the domain of warehouse management. As shown in Figure 11, the schema generated under CoT and the few-shot prompt settings include the *Cargo Name* directly within the *Task* table, which introduces a risk of inconsistency in the data (e.g. spelling errors in the *Cargo Names*). The output of SchemaAgent adheres to the principles of the 3NF. In the *Task* table, SchemaAgent correctly identifies the functional

Table 3: **Comparison with Text2SQL models. "Executable" indicates the execution using SQLite in DDL format.**

Method	Schema		Attribute		Primary key	Foreign key	Data type	Executable
	F1	Acc.	F1	Acc.	Acc.	Acc.	Acc.	Acc.
Arctic-Text2SQL-R1-7B	74.54	42.26	59.32	33.33	21.00	38.06	61.68	91.86
OminiSQL-14B	84.25	56.17	69.55	40.16	63.78	69.55	61.61	95.28
SchemaAgent	90.29	65.09	79.53	49.87	73.23	81.63	83.76	97.14

dependencies among attributes and accordingly determines a composite primary key. The normalized structure and composite key design significantly enhance data integrity.

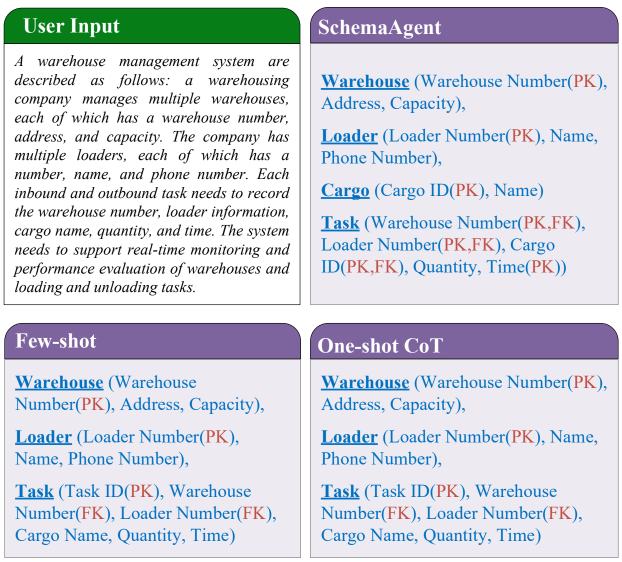


Figure 11: The schema output of SchemaAgent, few shot, and one shot with CoT in the warehouse management case.

Our method demonstrates significant performance for generating normalized schemas. Future work could integrate user query patterns to facilitate strategic denormalization, which enables a balance between query efficiency and data consistency.

6.8 Case Study for DDL Generation

We demonstrate that it is easy to covert schemas into DDL, as a fundamental part of physical design. Different database management systems (DBMS) employ diverse methodologies for physical design implementation. To verify the validity of the schema, we use a rule-based approach to generate SQLite-specific DDL, which is subsequently executed to empirically verify our methodology. This treatment results in a process similar to the Text2SQL task. Therefore, we select prominent Text2SQL works for evaluation, including **Arctic-Text2SQL-R1** [51]: a reinforcement learning-based Text2SQL model that optimizes for generating executable SQL queries through execution feedback. **OmniSQL** [21]: a Text2SQL model trained on a large synthetic dataset.

As shown in Table 3, existing Text2SQL models particularly struggle with relation schema identification. This highlights the crucial need for dedicated Text2DDL research, as it involves inferring complex structural design and relationships that existing Text2SQL approaches are not equipped to handle.

7 CONCLUSION & FUTURE WORK

The automation of database schema remains a significant challenge, due to its inherent dependence on specialized expertise and extensive accumulated practical experience. In this paper, we propose, for the first time, an LLM-based multi-agent framework for logical schema generation. We call this framework SchemaAgent, where we design six roles with a controllable interaction mechanism for error detection and correction in the workflow. In addition, we construct RSchema, a cross-domain dataset containing more than 381 pairs of user requirement texts and their corresponding logical schemas. We systematically evaluate the performance of mainstream LLMs and SchemaAgent on RSchema. Experimental results demonstrate that SchemaAgent achieves state-of-the-art and highly competitive performance across all metrics.

Future Work. This work is the first attempt to leverage LLMs for automated schema generation, which may give rise to several promising directions for future research, to name a few:

- **Physical Design Integration.** It would an interesting future work to incorporate physical design into our framework, which could make our work more comprehensive. Physical design may involve index generation, disk allocation, and so on.
- **Functional Dependency Discovery.** Identifying functional dependencies is a prerequisite for normalization. Traditional FDs discovery works are based on statistical analysis over datasets. However, with extensive world knowledge of LLMs, it is currently possible to discover FDs directly from natural language descriptions. Exploring LLM-based FD discovery opens up a new research avenue toward more intelligent and data-free schema normalization.
- **Query Efficiency and Normalization Balancing.** In this paper, we prioritize data consistency by requiring the generated schema to satisfy the 3NF. However, strict normalization may lead to suboptimal query performance in practice. A promising direction is to incorporate user query requirements into the schema generation process, aiming to achieve an optimal balance between data consistency and query efficiency.