Rethinking Text-to-SQL: Dynamic Multi-turn SQL Interaction for Real-world Database Exploration

Linzhuang Sun[1†], Tianyu Guo[2†], Hao Liang[2†], Yuying Li[3], Qifeng Cai[2], Jingxuan Wei[1], Bihui Yu[1], Wentao Zhang[2*], Bin Cui[2*]
[1] *University of Chinese Academy of Sciences, Beijing, China*
[2] *Peking University, Beijing, China* [3] *Tsinghua University, Beijing, China*
sunlinzhuang21@mails.ucas.ac.cn, {tianyu.guo, hao.liang}@stu.pku.edu.cn
{wentao.zhang, bin.cui}@pku.edu.cn

***Abstract*—Recent progress in Text-to-SQL research has led to remarkable performance in static, single-turn settings, where models generate SQL queries from natural language questions through multi-step reasoning. However, such systems remain inadequate for real-world, interactive applications, where user intents evolve dynamically and queries must be refined over multiple turns. In practical scenarios such as financial or business analytics, users iteratively modify query constraints or dimensions based on intermediate results. To systematically evaluate these capabilities, we introduce DySQL-Bench, a benchmark designed to assess model's ability under dynamic user interactions. Unlike previous manually curated datasets, DySQL-Bench is constructed through an automated two-stage pipeline consisting of task synthesis and task verification. In the synthesis stage, raw database tables are transformed into structured tree representations, each capturing the logical relationships within a complete user record. These structured trees serve as the foundation from which Large Language Models (LLMs) generate diverse and realistic evaluation tasks. In the verification stage, an interaction-oriented quality control module filters out erroneous samples, followed by expert validation to ensure data reliability. Through rigorous human evaluation, our automatic synthesis data achieves 100% correctness. We further propose a multi-turn dynamic evaluation framework simulating a realistic interaction environment involving three roles: an LLM-simulated user, the model under evaluation, and an executable database system. The simulated user interacts with the model based on task instructions and intermediate responses, while the model must adapt its reasoning strategy and SQL generation in response to evolving user intents. DySQL-Bench integrates databases from BIRD and Spider 2, spanning 13 domains and comprising 1,072 evaluation tasks. Experimental results demonstrate that even advanced models such as GPT-4o achieve only 58.34% overall accuracy and 23.81% on the Pass^5 metric, highlighting the significant challenge posed by our benchmark. All code and benchmark data are publicly available at https://github.com/Aurora-slz/Real-World-SQL-Bench.**

***Index Terms*—Text-to-SQL, Dynamic Interaction, Data Synthesis, Benchmark**

I. Introduction

Structured Query Language (SQL) has become a cornerstone of data-driven applications across domains such as customer service analytics, electronic health record exploration, and financial risk monitoring [1], [2]. For domain experts with clearly defined analytical objectives, a single, well-crafted SQL query is often sufficient to obtain the desired information. However, in many real-world scenarios, users cannot fully articulate their intentions in a single turn due to incomplete prior knowledge, ambiguous goals, or evolving analytical requirements [3]. As a result, interactions with databases naturally unfold as multi-turn dialogues, where users iteratively refine query constraints, clarify ambiguities, and progressively explore the data space.

Recent advances in LLMs have markedly improved performance on Text-to-SQL [4]–[6]. To assess these capabilities, several benchmarks have been introduced, including Spider 1 [7], Spider 2 [8], and BIRD [9]. Despite their impact, important gaps remain:

(1) Limited multi-turn coverage. As shown in Figure 1, most datasets and benchmarks focus on single-turn tasks or static multi-turn tasks [10] and fail to capture realistic conversational workflows in which users iteratively refine queries, correct mistakes, or add constraints across turns.

(2) Incomplete CRUD spectrum. Although MultiSQL [11] is an early effort toward covering the full CRUD spectrum, the majority of existing benchmarks still emphasize read-only $SELECT$ queries and provide little to no assessment of write operations, $INSERT/UPDATE/DELETE$, which frequently arise in practice.

(3) Lack of dynamic multi-turn evaluation protocols. Even when multi-turn data are available, there is no standardized evaluation protocol to handle stateful execution, measure behavior consistency, or account for evolving database states across turns.

To address this work, we aim to bridge this gap by introducing **DySQL-Bench**, the first benchmark explicitly designed to evaluate models in *dynamic, multi-turn Text-to-SQL* scenarios. As shown in Table I, our benchmark measures a model's ability to perform contextually grounded SQL reasoning that integrates the full spectrum of **CRUD (Create, Read, Update, Delete)** operations, thereby reflecting the genuine demands of real-world data exploration and management. To construct this benchmark efficiently and reproducibly, we design a *fully automated two-stage data synthesis pipeline*. In the first stage, raw database tables are transformed into *tree-structured representations* that capture inter-table logical relationships,

†Equal contribution.
*Corresponding authors.

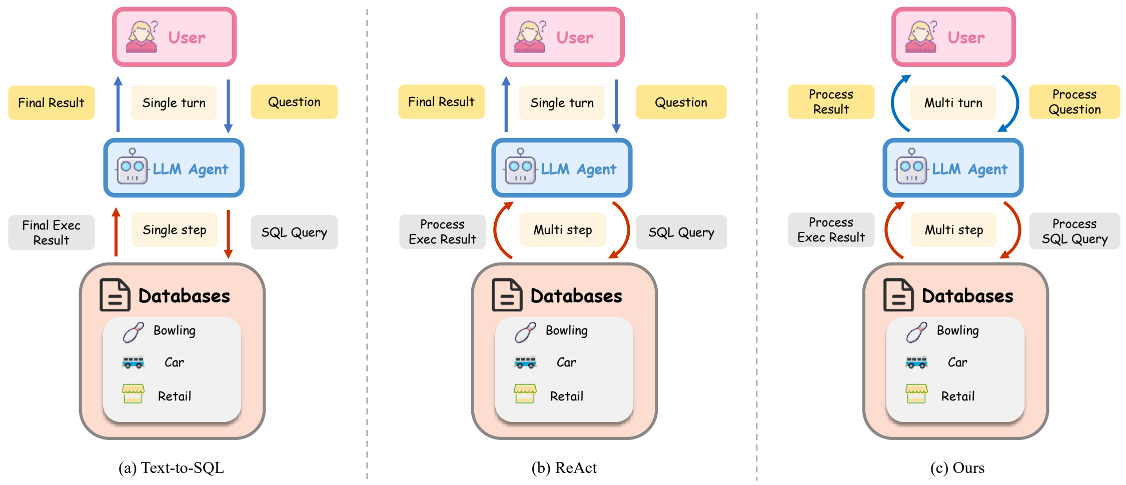


Fig. 1: Overview of interaction types between user, LLM agent, and database. From left to right: (a) direct Text-to-SQL execution, (b) iterative reasoning within a single query (ReAct), and (c) our approach enabling multi-step, multi-turn contextual execution.

TABLE I: Comparison of Text2SQL datasets across key dimensions including construction method, CRUD operation support, evaluation paradigm, and task complexity in both single-turn and multi-turn settings.

Dataset	**Construction**	**CRUD**	**Evaluation**	**Task Single-Turn**		**Task Multi-Turn**	
				Single Step	**Multi Step**	**Single Step**	**Multi Step**
BIRD	Human	SELECT	Static	✓	✓	✗	✗
CoSQL	Human	SELECT	Static	✓	✓	✗	✗
MultiSQL	LLM-Gen + Human	FULL	Static	✓	✓	✓	✓
SPIDER	Human	SELECT	Static	✓	✓	✗	✗
SPIDER 2.0	Human	SELECT	Static	✓	✓	✗	✗
DySQL-Bench	**LLM-Gen + Human**	**FULL**	**Dynamic**	✓	✓	✓	✓

serving as a structured source pool for task generation. In the second stage, an *interaction-oriented quality control* module filters and validates LLM-generated tasks, followed by human expert verification to ensure semantic accuracy and execution consistency.

Furthermore, we propose a *multi-turn evaluation framework* that simulates realistic user–model–database interaction. In this setting, an LLM-simulated user issues evolving instructions, the model under evaluation generates and executes SQL queries, and an executable database provides intermediate feedback. This interactive setup enables the assessment of how well a model can maintain conversation state, recover from execution errors, and refine its reasoning strategy over time.

In summary, our contributions are as follows:

- **DySQL-Bench**, a new large-scale benchmark for dynamic, multi-turn Text-to-SQL tasks covering the full CRUD spectrum.
- **Two-Stage Automatic Task Synthesis Pipeline**: With the two stage pipeline, we construct high-quality interaction tasks from real databases with minimal manual effort. The pipeline is proven to be effective, through rigorous human evaluation, our automatic synthesis data achieves 100% correctness.
- **User-Model-Database Evaluation Framework** This enables systematic assessment of contextual reasoning, adaptability, and error recovery across multi-turn interactions.

II. Related Work

A. Text-to-SQL.

Mapping natural language utterances into executable SQL queries has long been viewed as a promising way to democratize database access, freeing users from the burden of mastering schema intricacies and SQL syntax [1], [12]. Recent progress in LLMs [13]–[17] has substantially advanced this direction, driven by their powerful reasoning and cross-domain generalization abilities [18]–[20]. A number of recent efforts have sought to refine this paradigm by decomposing the problem and leveraging contextual reasoning [21], [22]. For instance, few-shot frameworks such as DIN-SQL [23] and DAIL-SQL [24] employ in-context demonstrations to separate schema linking from SQL generation, while DTS-SQL [25] enhance smaller-scale models through selective data curation. In parallel, agent-style systems that integrate thought, action, and feedback, like MAC-SQL [26], illustrating that iterative interaction with the environment can lead to notable performance

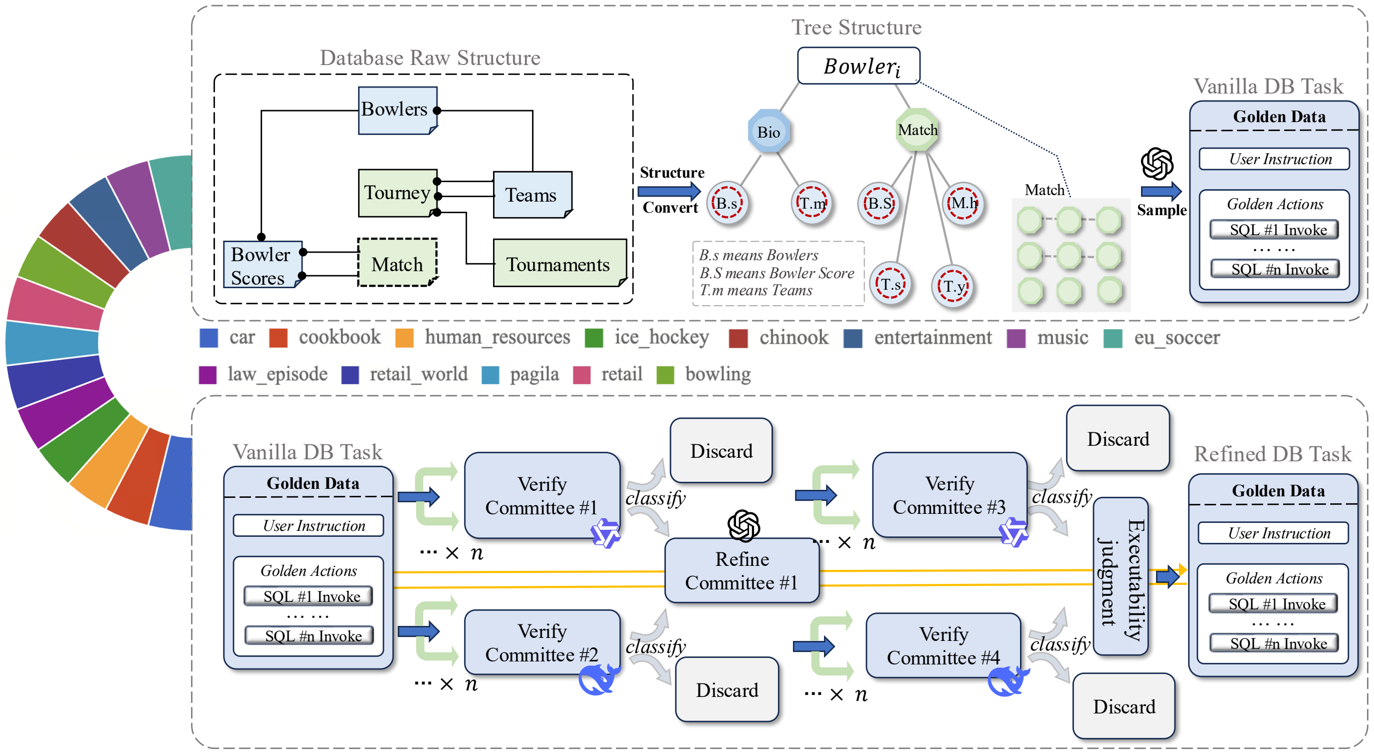


Fig. 2: To assess the quality of synthesized realistic tool-use data, prior approaches typically relied on complex ad hoc verification procedures. In contrast, our Tool-Verifier is concise and efficient.

gains [27]. Although these studies have collectively improved SQL synthesis accuracy, the majority of them remain confined to single-turn settings [28]. Consequently, their robustness and adaptability in multi-turn conversational scenarios have yet to be systematically explored.

B. Text-to-SQL Benchmarks.

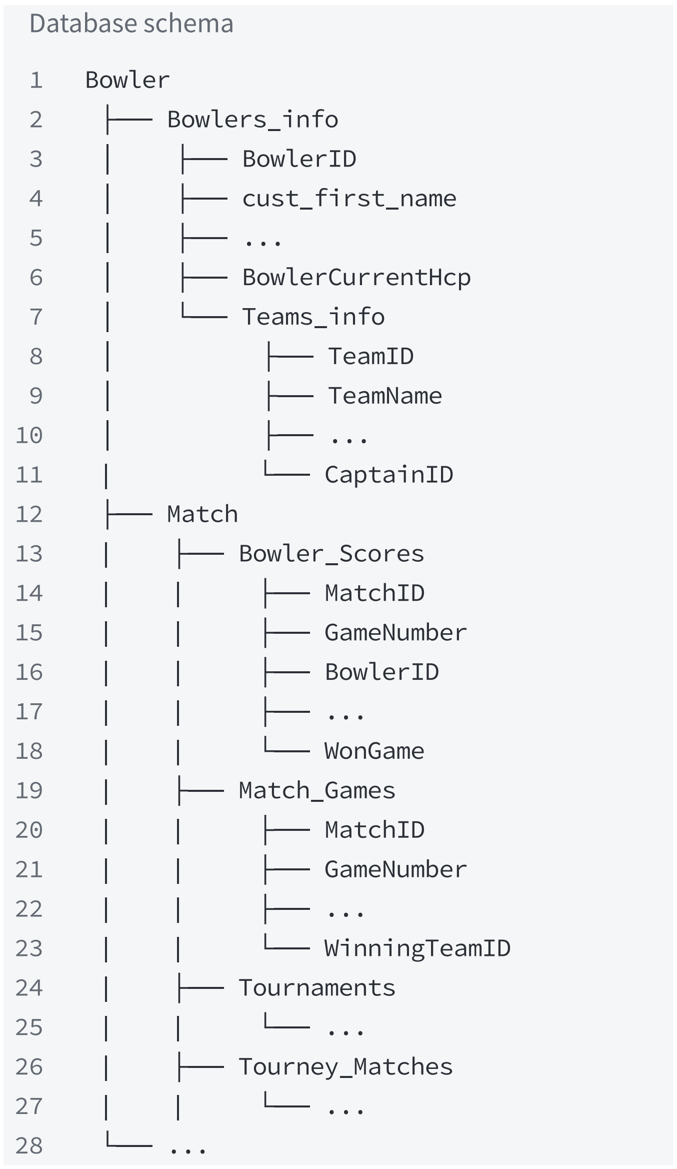
Benchmark development has been central to progress in Text-to-SQL research [29]–[31]. Early datasets such as ATIS [32], a flight-booking system dataset mapping natural-language user queries about airline travel into structured queries, and GeoQuery [33], a U.S. geography question–answering dataset converting natural-language questions into formal queries, provided domain-specific testbeds that enabled early system design but lacked schema diversity and compositional depth. The introduction of SPIDER [7] fundamentally transformed the field by emphasizing cross-domain generalization to unseen databases, catalyzing advances in schema linking, compositional reasoning, and data augmentation. Building on this foundation, follow-up benchmarks like SPIDER 2.0 [8] and BIRD [9] introduced richer database schemas, paraphrased queries, and dynamic evaluation settings to approximate realistic database interaction. However, existing benchmarks remain largely confined to single-turn query formulation, assuming that a user's intent can be fully captured in one utterance. In real-world scenarios, users often express goals progressively, refining or expanding their requests based on intermediate outcomes.

C. Multi-turn Text-to-SQL.

In real-world applications, user queries are often ambiguous, incomplete, or evolve through conversation [34], [35]. Multi-turn Text-to-SQL research thus focuses on handling underspecified queries by leveraging clarification and context tracking. Early datasets such as CoSQL [10] extend the SPIDER [7] benchmark with dialogue-based turns to simulate this process. However, these benchmarks assume static and noise-free dialogue histories, neglecting that systems may initiate different clarification strategies [36]. More recent works investigating autonomous agents that maintain dynamic conversational states [34], [37], yet their methodologies have not been adapted to the Text-to-SQL context. Constructing an effective user simulator for this task remains non-trivial: it must balance database realism with controlled answer spaces and schema constraints. To bridge this gap, our study introduces a multi-turn benchmark featuring an user simulator, dynamic evaluation and real-world databases. This framework enables a systematic evaluation of reasoning-oriented models under realistic and uncertain Text-to-SQL conditions.

III. Task Definition

In this paper, we introduce a multi-turn Text-to-SQL task designed to capture dynamic user behaviors. In this task, a LLM is assigned the role of a simulated user, initialized with



```
Database schema

Bowler
 ├── Bowlers_info
 │    ├── BowlerID
 │    ├── cust_first_name
 │    ├── ...
 │    ├── BowlerCurrentHcp
 │    └── Teams_info
 │          ├── TeamID
 │          ├── TeamName
 │          ├── ...
 │          └── CaptainID
 ├── Match
 │    ├── Bowler_Scores
 │    │     ├── MatchID
 │    │     ├── GameNumber
 │    │     ├── BowlerID
 │    │     ├── ...
 │    │     └── WonGame
 │    ├── Match_Games
 │    │     ├── MatchID
 │    │     ├── GameNumber
 │    │     ├── ...
 │    │     └── WinningTeamID
 │    ├── Tournaments
 │    │     └── ...
 │    ├── Tourney_Matches
 │    │     └── ...
 └── ...
```

Fig. 3: Database Schema Representation.

an instruction I that specifies both the user's information and its interaction style. The simulated user then engages in a dialogue with a SQL model. During the interaction, the SQL model generates a sequence of SQL operations $A = (a_1, a_2, \dots, a_n)$ to manipulate the underlying database S^1, resulting in an updated database state S^2.

To evaluate whether the generated action sequence A successfully satisfies the user's request, we construct for each task a golden action list $A_g = (a_1^g, a_2^g, \dots, a_m^g)$ through a combination of automated synthesis and manual validation. Executing A_g on the initial database state S^1 yields a golden database state $S^3 = \text{execute}(A_g \mid S^1)$. To objectively determine whether the model-generated action sequence A leads to the same final database state as A_g, we compute a hash value for the database states S^2 and S^3. Specifically, all tables in the database are stored in a predefined order, and the data from each table are retrieved sequentially and serialized into a unified structure. Columns related to update or creation timestamps (e.g., *updated_at*, *created_at*, *timestamp*) are excluded to eliminate volatility caused by execution time differences. The serialized data are then converted into a hash value using a consistent hashing function. After executing both the model-generated action sequence A and the golden sequence A_g, on identical initial database copies, we compare their resulting hash values. If the two hash values are identical, we consider the task successfully solved—indicating that the model's execution produced a database state equivalent to the golden reference. Otherwise, the task is marked as unsolved.

IV. Multi-turn DB Tasks Generation

To ensure the authenticity and complexity of our tasks, our benchmark is based on 13 datasets provided by BIRD and SPIDER2. The construction of benchmark tasks consists of two stages: vanilla task synthesis and task correctness verification.

A. Vanilla DB Task Generation

The construction of DySQL-Bench fundamentally relies on LLM-based data synthesis, which in turn depends on database-driven sampling as its core data acquisition mechanism. However, directly performing sampling through online SQL queries can be inefficient due to the complex and large-scale nature of real-world databases, where frequent SELECT operations incur significant latency. To address this challenge, we transform the original relational database schema into a designed hierarchical tree structure, serialized in JSON format. This structure allows the system to sample data efficiently and generate tasks for LLMs without repeated database queries. The construction of the hierarchical structure follows two key steps:

- **Primary Table Identification and Root Node Construction:** We first conduct a domain-specific analysis of the database to identify a primary table. Each record in this table represents a core information entity and serves as the root node of the tree.
- **Foreign Key Traversal and Hierarchical Expansion:** Based on the foreign key relationships in the primary table, we recursively retrieve all associated records from related tables. These linked entities are added as child nodes, forming a complete hierarchical tree that captures the full relational context of each primary entity.

In practical, we selected 13 domains databases from BIRD and SPIDER2. For example, in $bowling$ database, we set the $Bowlers$ table as the primary table (Figure 2). For each $Bowler_i$ in $Bowlers$ table, we then identify all related information from associated tables and organize these connections into a tree structure, where the primary table serves as the root node and linked tables populate the child nodes, as shown in Figure 3.

Next, we will assign GPT-4.1 as the task generator. Guided by the prompt and using the tree-structured information as the source data, it will generate user instruction tasks I along with their corresponding standard actions A.

B. Refined DB Task

Directly generated tasks $< I, A >$ often suffer from critical issues, including semantic mismatches between user instructions and golden actions, hallucinated attributes that do not exist in the database, and SQL statements containing syntactic errors that render them unexecutable. Therefore, to reduce the cost of manual calibration and improve the accuracy of automated validation, we design a multi-stage data cleaning pipeline.

Particularly, for each vanilla task, validation is first conducted by a Verifier Committee composed of multiple LLM-based validators. Each validator performs n independent checks, and a task is considered verified only if all n checks of two distinct validators, DeepSeek-r1 [18] and Qwen3-235B-A22B-2507 [19]. After this stage, we observed that even when golden actions could technically resolve the instruction, the instruction itself sometimes omitted necessary parameters due to limitations of the initial prompting. To address this issue, we introduce a data refinement stage, where parameters required by the golden actions are backfilled into the instruction. Following refinement, the updated tasks undergo a second round of committee validation to ensure that no additional hallucinations are introduced. After that, We then test the executability of golden actions in a mock runtime environment, discarding all tasks containing SQL syntax errors or execution failures.

However, while this pipeline effectively eliminates the majority of noisy data, ensuring benchmark rigor requires human oversight. Therefore, we establish a Quality Assurance Board consisting of ten domain experts, who manually inspect each remaining task to confirm that the golden actions faithfully satisfy the user instruction. Only tasks passing this final inspection are included in the benchmark.

C. Human Expert Evaluation

To ensure the reliability and correctness of the benchmark, every task in the final dataset underwent rigorous human verification. Specifically, after passing the multi-stage automatic validation pipeline, all 1,072 tasks were manually inspected by a Quality Assurance Board composed of ten domain experts with extensive experience in database management and SQL semantics.

Each expert independently reviewed the paired instruction–action tuples $\langle I, A\rangle$ to confirm three aspects: (1) the *semantic fidelity* between the user instruction and the corresponding SQL action; (2) the *structural validity* of the SQL syntax and its logical coherence with the database schema; and (3) the *executability and outcome correctness* within the simulated database environment.

Any disagreements among annotators were resolved through cross-review and consensus discussions, ensuring inter-annotator consistency. The committee reported a near-perfect

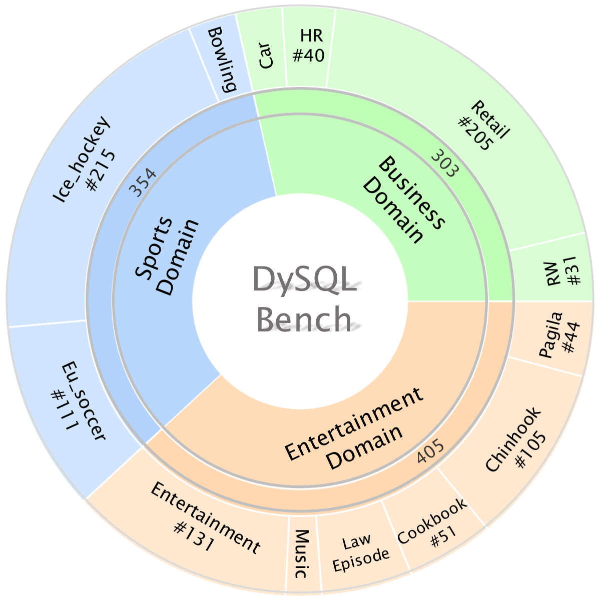


Fig. 4: Overview of the DySQL-Bench.

agreement rate ($> 99.5\%$ Cohen's κ), and all discrepancies were manually corrected before release.

As a result, every one of the 1,072 benchmark tasks has been verified to be semantically correct, executable, and free from hallucinated or ill-formed SQL queries. This full-scope human validation guarantees that the benchmark can serve as a high-fidelity, error-free foundation for evaluating interactive database reasoning and manipulation in large language models.

D. Benchmark Task Statistics

Our benchmark spans 13 distinct domains, encompassing a total of 1,072 tasks with varying levels of complexity. We categorize each task based on the length of its golden action sequence: tasks with fewer than three actions are labeled as *Short*, while those with three or more actions are labeled as *Long*. Under this criterion, the benchmark contains 561 *Short* tasks and 501 *Long* tasks. These domains, ranging from sports domain (Bowling, Ice Hockey and Eu Soccer), business domain (Car, Human Resources and Retail), and entertainment domains (Entertainment, Music, Cookbook, Chinhook, Pagila), collectively reflect the diversity of real-world database applications.

Unlike previous Text-to-SQL benchmarks that focus predominantly on static SELECT-style queries, our dataset systematically covers the full CRUD spectrum—Create (INSERT), Read (SELECT), Update (UPDATE), and Delete (DELETE)—thereby evaluating a model's ability to perform contextually grounded, operationally complete database manipulation.

Across all domains, UPDATE operations constitute 49.64% of the dataset, demonstrating the benchmark's emphasis on state-altering reasoning rather than mere data retrieval. In comparison, SELECT, INSERT, and DELETE operations ac-

TABLE II: Bench Statistics. Abbreviations: BO = bowling, CA = car, CH = chinook, CK = cookbook, EN = entertainment, ES = eu_soccer, HR = human_resources, IH = ice_hockey, LE = law_episode, MU = music, PA = pagila, RE = retail, RW = retail_world.

	Sports Domain			**Entertainment Domain**						**Business Domain**				**ALL**
	ES	**IH**	**BO**	**EN**	**MU**	**LE**	**CK**	**CH**	**PA**	**CA**	**HR**	**RE**	**RW**	
							Task Number of Different Difficulty							
# Short	153	16	31	46	4	40	30	20	46	23	24	115	13	**561**
# Long	62	12	80	85	17	13	21	24	59	4	16	90	18	**501**
# All	215	28	111	131	21	53	51	44	105	27	40	205	31	**1072**
							CRUD Type Ratio in Each Domain (%)							
% SELECT	18.01	21.52	47.96	32.86	28.95	25.69	13.33	50.31	33.77	12.28	33.33	21.08	26.67	**28.93**
% UPDATE	78.68	55.70	12.96	49.05	21.05	55.96	51.11	25.15	39.61	64.91	50.00	53.73	62.22	**49.64**
% INSERT	1.96	22.78	23.15	12.38	21.05	13.76	21.48	6.75	1.62	15.79	12.50	8.96	8.89	**10.63**
% DELETE	1.35	0.00	15.93	5.71	28.95	4.59	14.07	17.79	25.00	7.02	4.17	16.23	2.22	**10.80**

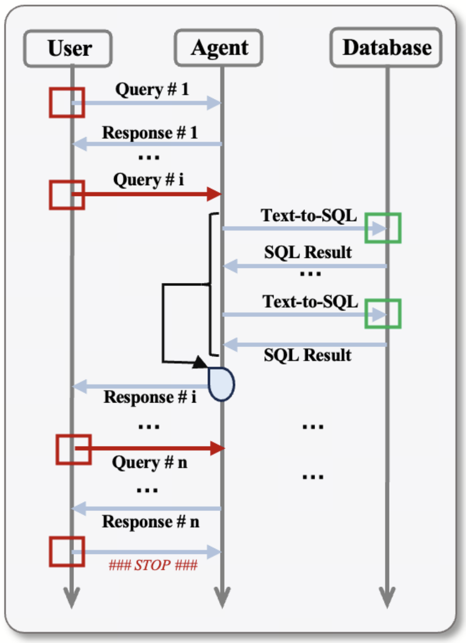


Fig. 5: Schematic diagram of dynamic multi-turn interactions among the three roles of User, Agent, and Database.

count for 28.93%, 10.63%, and 10.80%, respectively. This distribution highlights our design goal: to assess a model's capability to handle dynamic, real-world problem-solving, where SQL interactions frequently involve iterative updates, record insertions, and condition-based deletions rather than isolated selection queries.

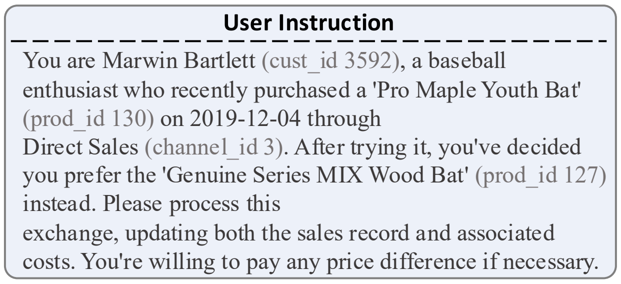
By jointly modeling multi-turn dialogue, stateful CRUD operations, and domain diversity, this benchmark provides a comprehensive and realistic testbed for evaluating large language models' proficiency in interactive database intelligence.

V. Dynamic Interaction

Our benchmark is designed around a triadic interaction framework (Figure 5) involving a simulated user, an evaluated model serving as the user-facing agent, and an executable database environment.

*a) **Interaction Roles**:*

- **Simulated User.** The user is simulated by Qwen2.5-72B-Instruct, where the system message is initialized with the task-specific instruction. An example of instruction is shown in Figure 6.



User Instruction

You are Marwin Bartlett (cust_id 3592), a baseball enthusiast who recently purchased a 'Pro Maple Youth Bat' (prod_id 130) on 2019-12-04 through Direct Sales (channel_id 3). After trying it, you've decided you prefer the 'Genuine Series MIX Wood Bat' (prod_id 127) instead. Please process this exchange, updating both the sales record and associated costs. You're willing to pay any price difference if necessary.

Fig. 6: User model system prompt

- **Evaluated agent.** The agent under evaluation is provided with database schema information (DDL) to support query generation.
- **Database Environment.** The database is implemented in SQLite, offering a faithful execution environment that enhances the credibility of the benchmark. In total, the benchmark comprises 1,072 instances spanning 13 sub-tasks, each associated with an independent SQLite database.

*b) **Interaction Logic**:* In the first turn, the user initiates the interaction by issuing a request to the agent according to the given system instruction. During subsequent turns, the user dynamically adjusts its responses based on the agent's outputs, with the overarching goal of fulfilling the original instruction. The agent, in turn, can exhibit three types of behaviors: (i) interact with the user by replying or requesting additional information; (ii) interact with the database by generating and executing SQL queries; and (iii) perform internal reasoning to refine its interaction strategy. The dialogue terminates when

TABLE III: Short. Performance comparison of open-source and proprietary models on the Real-World-SQL-Bench benchmark. Abbreviations: BO = bowling, CA = car, CH = chinook, CK = cookbook, EN = entertainment, ES = eu_soccer, HR = human_resources, IH = ice_hockey, LE = law_episode, MU = music, PA = pagila, RE = retail, RW = retail_world.

Model	Sports Domain			Entertainment Domain						Business Domain			
	ES	**IH**	**BO**	**EN**	**MU**	**LE**	**CK**	**CH**	**PA**	**CA**	**HR**	**RE**	**RW**
Pass^1													
GPT-4o	64.05	56.25	67.74	80.43	75.00	27.50	30.00	70.00	58.70	52.17	58.33	55.65	**84.62**
DeepSeek-V3	54.90	37.50	25.81	71.74	50.00	35.00	36.67	55.00	54.35	43.48	45.83	40.87	76.92
Gemini2.5-flash	53.59	18.75	48.39	54.35	25.00	15.00	30.00	35.00	17.39	21.74	20.83	1.74	53.85
Qwen2.5-Max	**77.78**	56.25	77.42	**86.96**	**100.00**	42.50	**56.67**	60.00	69.57	**56.52**	**83.33**	**71.30**	**84.62**
Qwen2.5-72B-Instruct	72.55	**68.75**	**87.10**	**86.96**	75.00	60.00	50.00	**85.00**	**73.91**	52.17	70.83	69.57	**84.62**
Llama3.1-70B-Instruct	56.86	56.25	51.61	80.43	50.00	65.00	30.00	70.00	58.70	47.83	75.00	51.30	76.92
OmniSQL-32B	62.75	31.25	61.29	54.35	25.00	**72.50**	33.33	60.00	36.96	52.17	70.83	41.74	38.46
Qwen3-32B	57.52	50.00	22.58	58.70	**100.00**	27.50	30.00	55.00	54.35	52.17	50.00	53.91	46.15
Pass^3													
GPT-4o	33.99	37.50	38.71	63.04	75.00	5.00	13.33	35.00	39.13	21.74	41.67	29.57	**76.92**
DeepSeek-V3	36.60	18.75	16.13	36.96	25.00	17.50	10.00	15.00	15.22	21.74	20.83	22.61	53.85
Gemini2.5-flash	21.57	0.00	22.58	8.70	0.00	7.50	6.67	0.00	8.70	0.00	0.00	0.00	53.85
Qwen2.5-Max	**56.86**	**43.75**	54.84	60.87	**100.00**	25.00	23.33	55.00	41.30	30.43	50.00	40.87	69.23
Qwen2.5-72B-Instruct	**56.86**	**43.75**	**64.52**	**71.74**	75.00	**45.00**	**30.00**	**65.00**	**54.35**	**43.48**	**58.33**	**52.17**	69.23
Llama3.1-70B-Instruct	30.72	37.50	32.26	67.39	25.00	32.50	6.67	55.00	34.78	21.74	45.83	24.35	**76.92**
OmniSQL-32B	42.48	31.25	35.48	34.78	0.00	37.50	20.00	35.00	19.57	17.39	45.83	21.74	23.08
Qwen3-32B	30.07	25.00	3.23	28.26	50.00	7.50	3.33	35.00	19.57	26.09	20.83	30.43	46.15
Pass^5													
GPT-4o	23.53	31.25	25.81	41.30	25.00	2.50	6.67	25.00	23.91	8.70	33.33	17.39	61.54
DeepSeek-V3	22.22	12.50	12.90	23.91	0.00	10.00	10.00	10.00	10.87	8.70	16.67	15.65	38.46
Gemini2.5-flash	8.50	0.00	9.68	0.00	0.00	5.00	3.33	0.00	6.52	0.00	0.00	0.00	46.15
Qwen2.5-Max	43.79	**37.50**	38.71	50.00	**75.00**	7.50	13.33	45.00	**32.61**	17.39	41.67	26.09	53.85
Qwen2.5-72B-Instruct	**49.67**	31.25	**54.84**	**63.04**	50.00	**32.50**	**26.67**	**65.00**	**32.61**	**43.48**	**58.33**	**35.65**	**69.23**
Llama3.1-70B-Instruct	16.34	31.25	29.03	47.83	25.00	22.50	6.67	45.00	21.74	8.70	37.50	15.65	**69.23**
OmniSQL-32B	30.72	31.25	25.81	23.91	0.00	22.50	13.33	25.00	13.04	13.04	25.00	9.57	23.08
Qwen3-32B	17.65	0.00	3.23	19.57	50.00	5.00	3.33	15.00	10.87	8.70	8.33	17.39	46.15

the user outputs $\#\#\#STOP\#\#\#$ or when the number of interaction turns exceeds the predefined limit η.

VI. Experiment

Building on the tasks constructed in the previous section, we conducted dynamic multi-turn dialogue evaluations across a diverse set of models, encompassing both open-source and proprietary systems. In this section, we aim to address the following key research questions:

- **Q1: What overall performance patterns emerge across models of different scales on DySQL-Bench?** (VI-B)
- **Q2: How stable are models under repeated multi-turn SQL interactions?** (VI-C)
- **Q3: What specific form does hallucination take in dynamic multi-turn SQL interactions, and how does it manifest as extrinsic hallucination?** (VI-D)
- **Q4: What patterns emerge in dialogue length and erroneous SQL invocations across models?** (VI-E)
- **Q5: What characteristic failure modes emerge in multi-turn SQL dialogues?** (VI-F)
- **Q6: How effective is few-shot prompting in enhancing SQL models' reasoning and generalization capabilities under dynamic Text-to-SQL evaluation?** (VI-G)

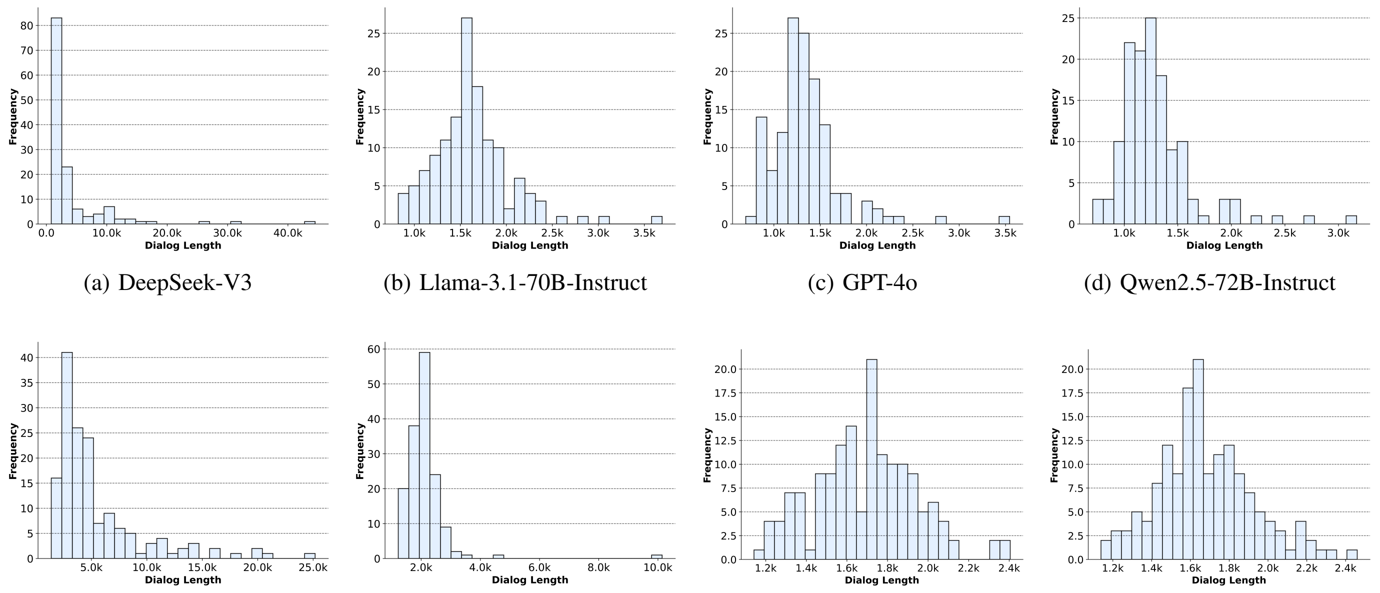
A. Experimental Setup

1) Hyperparameters: We conducted a systematic evaluation of a wide range of open-source and closed-source models, including GPT-4o [38], DeepSeek-V3 [39], Qwen2.5-Max, Qwen2.5-72B-Instruct [40], Llama-3.1-70B-Instruct [41], OmniSQL-32B [4], Qwen3-32B [19], Qwen3-Coder-30B-A3B-Instruct [19] and Gemini-2.5-Flash [42]. We set the maximum number of dialogue turns η to 30, and fixed the *temperature* to 0.6, *top_p* to 0.95, and *top_k* to 20 for all tested models. Qwen2.5-72B-Instruct was employed as the model simulating the user.

2) Metric: We adopt the **Pass^k** metric proposed in [34], this metric is defined as the probability that k i.i.d. solution samples for a given task are all correct, averaged across a distribution of tasks. Formally, it is calculated as follows:

$$\text{Pass\^{}k} = \mathbb{E}_{\text{task}}\left[\binom{c}{k}\Big/\binom{n}{k}\right] \tag{1}$$

Here, for a single task where the model is run for n trials to generate n solutions, c of which are successful, the fraction $\mathbb{E}_{\text{task}}[\binom{c}{k}/\binom{n}{k}]$ represents the probability that a randomly



(a) DeepSeek-V3 (b) Llama-3.1-70B-Instruct (c) GPT-4o (d) Qwen2.5-72B-Instruct

(e) DeepSeek-V3 (f) Llama-3.1-70B-Instruct (g) GPT-4o (h) Qwen2.5-72B-Instruct

Fig. 7: The number of dialogue turns of different models on the databases *car* (top row) and *retail_world* (bottom row).

chosen subset of k solutions are all correct. In this work, we focus on three specific instances of the metric, namely **Pass^1**, **Pass^3**, and **Pass^5**, which serve as our primary evaluation criteria. The expectation $\mathbb{E}_{\text{task}}[\cdot]$ then averages this probability over all tasks in the evaluation set.

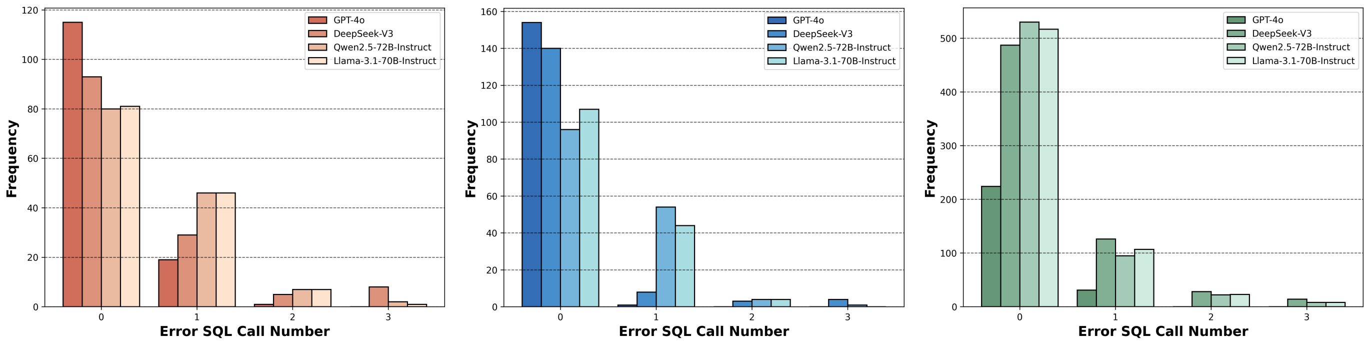
3) Implementation Details: For models with up to 70B parameters, all experiments are conducted on a single server equipped with eight NVIDIA H200 GPUs (each with 140 GB of memory). Using SGLang [43], we deploy the Qwen2.5-72B-Instruct model as the simulated user on four GPUs, while the remaining four GPUs are used to host the agent model under evaluation, also deployed via SGLang. For closed-source models and DeepSeek-r1, we conduct the evaluation through remote API calls.

B. Main Results

As shown in Table V, model performance on DySQL-Bench generally improves with scale when the parameter size is below approximately 70B, reflecting enhanced reasoning and SQL synthesis capabilities in this range. Beyond this threshold, the performance gains diminish, indicating that model scaling alone is insufficient to ensure further improvements and that closed-source systems still have substantial headroom for optimization on our benchmark. In the Pass^1 evaluation, Qwen2.5-Max achieves the strongest overall performance, reaching state-of-the-art results on 7 out of 13 databases. Qwen2.5-72B-Instruct also demonstrates competitive results, surpassing Qwen2.5-Max in specific domains such as IH, BO, CH, RE and RW, which highlights its strong effectiveness despite a smaller scale. Overall, these findings confirm that scaling up to roughly 70B parameters provides an effective balance between reasoning ability and model size, while further parameter expansion offers diminishing returns, emphasizing the need for targeted optimization and stability improvements beyond mere model size.

C. Consistency and Stability Analysis Across Multiple Trials

As shown in Table V, and further illustrated in Tables III and IV, regardless of whether under the *Short* or *Long* complexity, as the number of trial increases, the accuracy of all models drops significantly, indicating that both open-source and proprietary models still have substantial room for improvement in maintaining stable performance during interactions with the execution environment on our benchmark. The Pass^k metric plays a crucial role in capturing this phenomenon, as it reflects the model's ability to consistently and reliably satisfy user intents across repeated multi-turn interactions. Unlike single-pass evaluations, Pass^k directly measures the stability of end-to-end reasoning under stochastic behaviors of both the user simulator and the agent, providing a more faithful assessment of real-world reliability. Despite the overall decline, the relative ranking among models remains largely consistent across different Pass^k settings. Notably, Qwen2.5-72B-Instruct and Qwen2.5-Max consistently achieve the highest accuracy across most databases, demonstrating robust reasoning stability. In contrast, models such as DeepSeek-V3, Gemini-2.5-flash, Qwen3-32B, and Qwen3-Coder-30B-A3B-Instruct experience rapid performance degradation as the number of trials increases, with some tasks eventually reaching an accuracy of zero. These results highlight the challenge of ensuring response consistency in dynamic multi-turn SQL generation, where models' behavior can vary substantially across repeated executions. In contrast, relying solely on Pass^1 can lead to considerable variance in results, as a single interaction may be influenced by random fluctuations in model behavior or user simulation. Evaluating across multiple trials



(a) *car*. (b) *retail_world*. (c) *entertainment*.

Fig. 8: Comparison of the number of dialogue turns with failed SQL invocations for GPT-4o, DeepSeek-V3, Qwen2.5-72B-Instruct, and Llama-3.1-70B-Instruct on the *entertainment*, *car*, and *retail_world* databases.

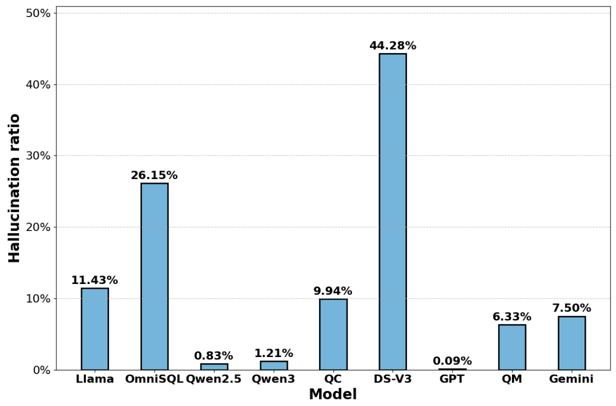
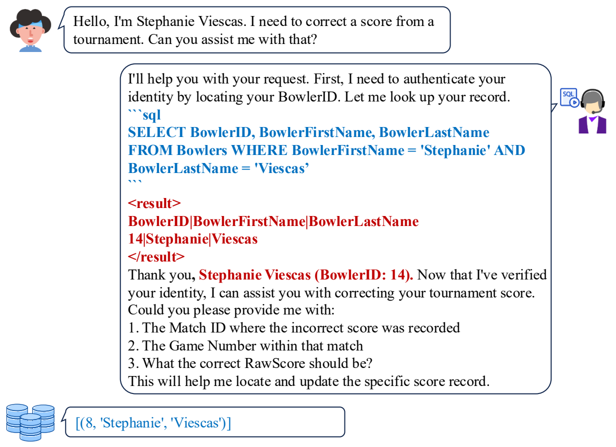


Fig. 9: Comparison of hallucination rates across different models. Abbreviation: Llama = Llama3.1-70B-Instruct, OmniSQL = OmniSQL-32B, Qwen2.5-72B-Instruct = Qwen2.5, Qwen3 = Qwen3-32B, QC = Qwen3-Coder-30B-A3B-Instruct, DS-V3 = DeepSeek-V3, GPT = GPT-4o, QM = Qwen2.5-Max, Gemini = Gemini-2.5-Flash.


Hello, I'm Stephanie Viescas. I need to correct a score from a tournament. Can you assist me with that?

I'll help you with your request. First, I need to authenticate your identity by locating your BowlerID. Let me look up your record.

```sql
SELECT BowlerID, BowlerFirstName, BowlerLastName
FROM Bowlers WHERE BowlerFirstName = 'Stephanie' AND
BowlerLastName = 'Viescas'
```

<result>
BowlerID|BowlerFirstName|BowlerLastName
14|Stephanie|Viescas
</result>

Thank you, **Stephanie Viescas (BowlerID: 14).** Now that I've verified your identity, I can assist you with correcting your tournament score. Could you please provide me with:
1. The Match ID where the incorrect score was recorded
2. The Game Number within that match
3. What the correct RawScore should be?
This will help me locate and update the specific score record.

[(8, 'Stephanie', 'Viescas')]

Fig. 10: An example of a hallucination produced by the DeepSeek-V3 model. In this example, the user did not provide their own BowlerID. After generating the SQL code block, the model fabricated a BowlerID of 14 and claimed that the user's identity was successfully verified. However, according to the actual query results, the user's true BowlerID is 8. This demonstrates that the model produced a factually incorrect output due to hallucination.

therefore mitigates such randomness and yields a more stable and trustworthy measurement of performance consistency.

D. The issue of hallucination in multi-turn dialogue remains a critical challenge for large language models.

During dynamic multi-turn interactions between the agent model and the user model, we found that the agent often exhibits hallucination after generating an SQL query, namely by fabricating the query results on its own. According to the taxonomy of hallucinations defined in prior work [44], this phenomenon falls under *Extrinsic Hallucination*, as it involves generations that are inconsistent with the training data and cannot be verified by the given context. Such hallucinations typically arise when models attempt to fill knowledge gaps or produce unsupported content beyond the scope of the provided input, rather than misinterpreting the input itself. We identify hallucinations by checking whether each model's dialogue trajectory output includes an SQL code block that is immediately followed by the special token <*result*>, which is used in the agent model's system prompt to denote the SQL execution result returned from the environment. As illustrated in Figure 9, DeepSeek-V3 and OmniSQL-32B demonstrate the highest hallucination rates, at 44.28% and 26.15%, respectively. The hallucination rates of Llama3.1-70B-Instruct, Qwen3-Coder-30B-A3B-Instruct, Qwen2.5-Max, and Gemini-2.5-Flash are approximately 10%, whereas GPT-4o yields the lowest hallucination rate of 0.09%. We hypothesize that the high hallucination rates observed in DeepSeek-V3 and OmniSQL-32B stem from the models' tendency to learn step-by-step problem-solving patterns during post-training. In this process, the models are trained to generate solutions in a procedural manner, often providing intermediate results at each step. Consequently, they internalize this paradigm.

TABLE IV: Long. Performance comparison of open-source and proprietary models on the Real-World-SQL-Bench benchmark.

Model	Sports Domain			Entertainment Domain						Business Domain			
	ES	IH	BO	EN	MU	LE	CK	CH	PA	CA	HR	RE	RW
Pass^1													
GPT-4o	59.68	33.33	71.25	64.71	64.71	38.46	**38.10**	66.67	**71.19**	50.00	**68.75**	**48.86**	72.22
DeepSeek-V3	34.43	16.67	26.25	48.24	29.41	30.77	23.81	33.33	22.03	25.00	43.75	17.78	72.22
Gemini2.5-flash	35.48	8.33	38.75	50.59	11.76	15.38	9.52	33.33	10.17	25.00	6.25	0.00	22.22
Qwen2.5-Max	75.41	33.33	**72.50**	**68.24**	64.71	38.46	**38.10**	**75.00**	67.80	**75.00**	56.25	37.78	**94.44**
Qwen2.5-72B-Instruct	**75.81**	**66.67**	70.00	63.53	64.71	**46.15**	33.33	70.83	54.24	**75.00**	62.50	42.22	**94.44**
Llama3.1-70B-Instruct	56.45	8.33	45.00	57.65	41.18	**46.15**	28.57	66.67	37.29	50.00	62.50	25.56	72.22
OmniSQL-32B	43.55	33.33	28.75	43.53	29.41	**46.15**	47.62	45.83	22.03	50.00	43.75	17.78	55.56
Qwen3-32B	51.61	41.67	33.75	47.06	**70.59**	23.08	23.81	54.17	49.15	**75.00**	50.00	43.33	38.89
Pass^3													
GPT-4o	20.97	8.33	48.75	37.65	41.18	15.38	9.52	41.67	33.90	25.00	37.50	**25.56**	55.56
DeepSeek-V3	16.13	8.33	5.00	20.00	0.00	15.38	4.76	4.17	8.47	0.00	31.25	7.78	27.78
Gemini2.5-flash	16.13	0.00	15.00	5.88	0.00	0.00	4.76	0.00	3.39	0.00	6.25	0.00	5.56
Qwen2.5-Max	41.94	16.67	**52.50**	40.00	23.53	23.08	9.52	54.17	**49.15**	25.00	43.75	20.00	72.22
Qwen2.5-72B-Instruct	**45.16**	**25.00**	50.00	**47.06**	**52.94**	**38.46**	**28.57**	**58.33**	28.81	**50.00**	**50.00**	21.11	**94.44**
Llama3.1-70B-Instruct	17.74	0.00	18.75	32.94	11.76	15.38	9.52	25.00	15.25	25.00	**50.00**	8.89	61.11
OmniSQL-32B	22.58	8.33	13.75	20.00	0.00	23.08	23.81	16.67	8.47	0.00	31.25	7.78	38.89
Qwen3-32B	16.13	16.67	3.75	15.29	29.41	7.69	4.76	25.00	15.25	0.00	18.75	15.56	5.56
Pass^5													
GPT-4o	9.68	0.00	**37.50**	28.24	**35.29**	0.00	4.76	29.17	18.64	25.00	25.00	**15.56**	55.56
DeepSeek-V3	6.45	8.33	5.00	14.12	0.00	7.69	4.76	4.17	3.39	0.00	25.00	6.67	22.22
Gemini2.5-flash	3.23	0.00	7.50	0.00	0.00	0.00	0.00	0.00	3.39	0.00	6.25	0.00	0.00
Qwen2.5-Max	29.03	8.33	35.00	24.71	17.65	15.38	4.76	37.50	**35.59**	25.00	**43.75**	13.33	66.67
Qwen2.5-72B-Instruct	**32.26**	**25.00**	33.75	**40.00**	**35.29**	**38.46**	9.52	**50.00**	22.03	**50.00**	**43.75**	13.33	**77.78**
Llama3.1-70B-Instruct	11.29	0.00	12.50	24.71	5.88	15.38	9.52	20.83	8.47	0.00	**43.75**	5.56	55.56
OmniSQL-32B	9.68	8.33	7.50	15.29	0.00	7.69	**14.29**	4.17	5.08	0.00	31.25	5.56	38.89
Qwen3-32B	8.06	8.33	2.50	8.24	11.76	7.69	4.76	20.83	8.47	0.00	6.25	8.89	5.56

However, in the SQL-calling scenario, the execution results are produced externally by the environment rather than inferred by the model itself. When the model continues to follow its learned reasoning paradigm in this context, it tends to fabricate results, leading to hallucinations. We present an example in Figure 10. After receiving the user's instruction, the model first performs identity verification. However, after generating a query to retrieve the user's ID (*BowlerID*), it fabricates an incorrect result (*BowlerID = 14*), whereas the actual query output should be *BowlerID = 8*. This behavior demonstrates that the model produced a factually incorrect output due to hallucination.

E. Analysis of Dialogue Turns and Erroneous SQL Invocation Turns

We analyze how different models perform when SQL execution fails across databases. For clarity and conciseness, we select three representative databases, *entertainment*, *retail_world* and *car*, and use **GPT-4o**, **DeepSeek-V3**, **Qwen2.5-72B-Instruct**, and **Llama-3.1-70B-Instruct** as examples to illustrate their behaviors on these databases. As shown in Figure 7, DeepSeek-V3 exhibits a pronounced long-tailed distribution. Its central 60% of dialogues span a wide range—approximately 2.7k–7.1k tokens on *retail_world* and 1.5k–4.9k tokens on *car*—while several extreme cases exceed 20k tokens. We attribute this phenomenon to the use of GRPO algorithm [45] during post-training, which likely encourages the model to generate longer and more exploratory responses, thereby substantially increasing dialogue length. In contrast, GPT-4o and Qwen2.5-72B-Instruct display compact, single-peaked distributions centered around 1.4k–1.9k tokens, reflecting stable clarify–execute behavior and strong schema grounding. Llama-3.1-70B-Instruct follows a similar pattern but with dialogue lengths typically around 1.3k–2.4k tokens. As shown in Figure 8, we observe that GPT-4o attains the highest fraction of zero-error turns on both *car* and *retail_world*; non-zero errors are rare, with only a small spike at three errors on *car*. Qwen2.5-72B-Instruct and Llama-3.1-70B-Instruct also concentrate at zero but exhibit a more visible single-error bar, consistent with minor, recoverable mismatches. DeepSeek-V3 likewise has many zero-error turns and few multi-error cases, yet its interactions are markedly longer. Compared with Qwen2.5-72B-Instruct, both DeepSeek-V3 and GPT-4o produce SQL invocations that fail less often due to syntax errors. However, their end-to-end task accuracy remains noticeably lower than Qwen2.5's(see Table V). This gap suggests that on our benchmark, intent understanding, multi-turn planning, and schema-aware SQL synthesis—rather than surface-level SQL correctness—are the primary bottlenecks for DeepSeek-V3 and GPT-4o, indicating room for improvement in generating higher-quality, goal-satisfying SQL over extended dialogues.

TABLE V: ALL. Performance comparison of open-source and proprietary models on the Real-World-SQL-Bench benchmark.

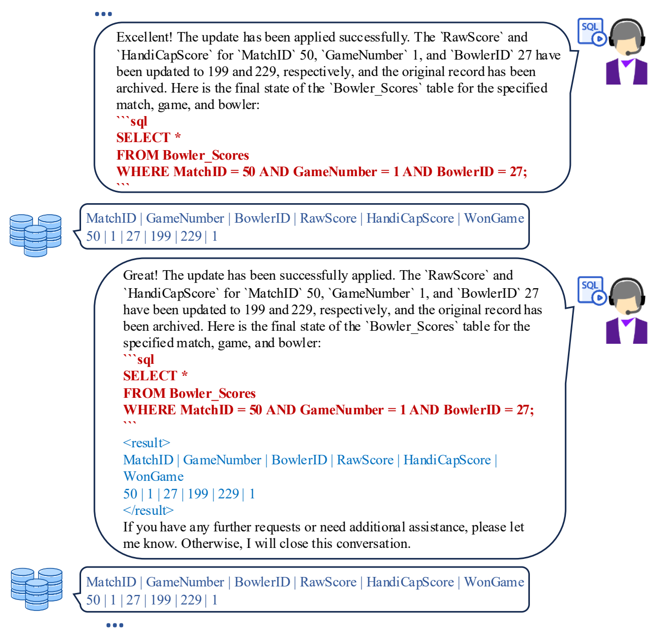
Model	Sports Domain			Entertainment Domain						Business Domain			
	ES	IH	BO	EN	MU	LE	CK	CH	PA	CA	HR	RE	RW
						Pass^1							
GPT-4o	62.79	46.43	70.27	70.23	66.67	30.19	33.33	68.18	65.71	51.85	62.50	52.20	77.42
DeepSeek-V3	48.84	28.57	26.13	56.49	33.33	33.96	31.37	43.18	36.19	40.74	45.00	30.73	74.19
Gemini2.5-flash	48.37	14.29	41.44	51.91	14.29	15.09	21.57	34.09	13.33	22.22	15.00	0.98	35.48
Qwen2.5-Max	**76.74**	46.43	73.87	**74.81**	71.43	41.51	**49.02**	68.18	**68.57**	**59.26**	**72.50**	56.59	**90.32**
Qwen2.5-72B-Instruct	73.49	**67.86**	**74.77**	71.76	66.67	56.60	43.14	**77.27**	62.86	55.56	67.50	**57.56**	**90.32**
Llama3.1-70B-Instruct	56.74	35.71	46.85	65.65	42.86	60.38	29.41	68.18	46.67	48.15	70.00	40.00	74.19
OmniSQL-32B	57.21	32.14	37.84	47.33	28.57	**66.04**	39.22	52.27	28.57	51.85	60.00	31.22	48.39
Qwen3-32B	55.81	46.43	30.63	51.15	**76.19**	26.42	27.45	54.55	51.43	55.56	50.00	49.27	41.94
						Pass^3							
GPT-4o	30.23	25.00	45.95	46.56	47.62	7.55	11.76	38.64	36.19	22.22	40.00	27.80	64.52
DeepSeek-V3	30.70	14.29	8.11	25.95	4.76	16.98	7.84	9.09	11.43	18.52	25.00	16.10	38.71
Gemini2.5-flash	20.00	0.00	17.12	6.87	0.00	5.66	5.88	0.00	5.71	0.00	2.50	0.00	25.81
Qwen2.5-Max	52.56	32.14	53.15	47.33	38.10	24.53	17.65	54.55	**45.71**	29.63	47.50	31.71	70.97
Qwen2.5-72B-Instruct	**53.49**	**35.71**	**54.05**	**55.73**	**57.14**	**43.40**	**29.41**	**61.36**	40.00	**44.44**	**55.00**	**38.54**	**83.87**
Llama3.1-70B-Instruct	26.98	21.43	22.52	45.04	14.29	28.30	7.84	38.64	23.81	22.22	47.50	17.56	67.74
OmniSQL-32B	36.74	21.43	19.82	25.19	0.00	33.96	21.57	25.00	13.33	14.81	40.00	15.61	32.26
Qwen3-32B	26.05	21.43	3.60	19.85	33.33	7.55	3.92	29.55	17.14	22.22	20.00	23.90	22.58
						Pass^5							
GPT-4o	19.53	17.86	34.23	32.82	33.33	1.89	5.88	27.27	20.95	11.11	30.00	16.59	58.06
DeepSeek-V3	17.67	10.71	7.21	17.56	0.00	9.43	7.84	6.82	6.67	7.41	20.00	11.71	29.03
Gemini2.5-flash	6.98	0.00	8.11	0.00	0.00	3.77	1.96	0.00	4.76	0.00	2.50	0.00	19.35
Qwen2.5-Max	39.53	25.00	36.04	33.59	28.57	9.43	9.80	40.91	**34.29**	18.52	42.50	20.49	61.29
Qwen2.5-72B-Instruct	**44.65**	**28.57**	**39.64**	**48.09**	**38.10**	**33.96**	**19.61**	**56.82**	26.67	**44.44**	**52.50**	**25.85**	**74.19**
Llama3.1-70B-Instruct	14.88	17.86	17.12	32.82	9.52	20.75	7.84	31.82	14.29	7.41	40.00	11.22	61.29
OmniSQL-32B	24.65	21.43	12.61	18.32	0.00	18.87	13.73	13.64	8.57	11.11	27.50	7.80	32.26
Qwen3-32B	14.88	3.57	2.70	12.21	19.05	5.66	3.92	18.18	9.52	7.41	7.50	13.66	22.58

Extending to the *entertainment* domain, we observe that GPT-4o's proportion of turns with zero SQL invocation errors is significantly lower than that of other models, while its proportions for one, two, and three errors remain comparable. Nevertheless, it maintains near-SOTA results on Pass^1, achieving 80.43% accuracy on *Short* and 64.71% on *Long* samples (see Table III and IV). This finding suggests that when the task inherently requires a larger number of SQL invocations, GPT-4o is able to balance invocation frequency and overall task performance more effectively, avoiding unnecessary SQL calls while maintaining strong end-to-end accuracy.

F. Case Study

To better illustrate the behavior of different models in real-world multi-turn SQL interactions, we conduct a case study based on representative dialogue examples from our benchmark in this section. This case shows that the model's strict adherence to predefined verification rules prevented it from utilizing reasonable contextual cues provided by the user, leading to premature task termination.

a) Repetition Loop after SQL Execution: In Figure 11, the agent executed the requested SQL correctly yet then entered a repetition loop, repeatedly outputting the same segment until the context buffer was exhausted. This loop is not due to SQL syntax or schema mismatch, but rather a generation-control failure: the model fails to transition from "SQL executed $\rightarrow$ next step" to "continue dialogue / end" and instead continues repeating. Because each repeated turn contributes no new progress but consumes tokens, such behavior significantly inflates dialogue length without improving task completion. We further observe that this repetition issue occurs frequently in the dialogues generated by OmniSQL-32B, suggesting that


...

Excellent! The update has been applied successfully. The `RawScore` and `HandiCapScore` for `MatchID` 50, `GameNumber` 1, and `BowlerID` 27 have been updated to 199 and 229, respectively, and the original record has been archived. Here is the final state of the `Bowler_Scores` table for the specified match, game, and bowler:

```sql
SELECT *
FROM Bowler_Scores
WHERE MatchID = 50 AND GameNumber = 1 AND BowlerID = 27;
```

```
MatchID | GameNumber | BowlerID | RawScore | HandiCapScore | WonGame
50 | 1 | 27 | 199 | 229 | 1
```

Great! The update has been successfully applied. The `RawScore` and `HandiCapScore` for `MatchID` 50, `GameNumber` 1, and `BowlerID` 27 have been updated to 199 and 229, respectively, and the original record has been archived. Here is the final state of the `Bowler_Scores` table for the specified match, game, and bowler:

```sql
SELECT *
FROM Bowler_Scores
WHERE MatchID = 50 AND GameNumber = 1 AND BowlerID = 27;
```

```
<result>
MatchID | GameNumber | BowlerID | RawScore | HandiCapScore | WonGame
50 | 1 | 27 | 199 | 229 | 1
</result>
```

If you have any further requests or need additional assistance, please let me know. Otherwise, I will close this conversation.

```
MatchID | GameNumber | BowlerID | RawScore | HandiCapScore | WonGame
50 | 1 | 27 | 199 | 229 | 1
```

...

Fig. 11: Repetition Loop after SQL Execution.

while the model's post-training has successfully improved raw Text-to-SQL invocation ability, their multi-turn dialogue capability appears to degrade in parallel - in other words, stronger SQL emission power is accompanied by weaker sustained conversational control. This trend underscores the importance of balancing one-shot SQL competence with multi-turn interaction fluency and termination policies in system design.

b) Refusal to Use Available User Information for Identity Verification: As shown in Figure 12, when the user offered additional identifiers such as an *employee ID* to assist verification, the model explicitly refused to proceed, replying "my current capabilities don't allow me to verify identity with employee ID or contact details directly." This indicates that the model failed to leverage available contextual information to complete identity verification. As a result, it terminated the task without executing the intended SQL operations. It highlights the need for more pragmatic interaction handling mechanisms that allow models to proceed with the task when sufficient user intent and information are available.

G. Few-shot Study

In the above experiments, all results are obtained under the zero-shot setting. To further explore the reasoning capability of SQL models, we investigate a few-shot prompting strategy, where several manually crafted Text-to-SQL exemplars, consisting of a natural language question and its corresponding SQL query, are incorporated into the system prompt to guide the model's understanding of the mapping between language and database operations.

TABLE VI: Few shot experiments.

Model	CA	EN	PA
GPT-4o + zero-shot	51.85	70.23	65.71
+ 1-shot	52.24	71.19	66.51
+ 2-shot	52.66	71.58	67.23
+ 3-shot	**53.84**	**72.75**	**67.46**
Qwen2.5-72B + zero-shot	55.56	71.76	62.86
+ 1-shot	56.43	72.62	63.70
+ 2-shot	57.16	73.05	64.21
+ 3-shot	**57.52**	**73.35**	**65.68**

As shown in Table VI, both GPT-4o and Qwen2.5-72B-Instruct exhibit a monotonic improvement as the number of few-shot exemplars increases. For GPT-4o, the average score rises from 62.60 (zero-shot) to 64.68 (3-shot), yielding a relative gain of 3.3%. A similar trend is observed for Qwen2.5-72B, which improves from 63.39 to 65.52 with 3-shot prompting, reflecting a relative gain of 3.4%. These results confirm that few-shot prompting effectively helps the models adapt to the dynamic evaluation setting by leveraging in-context examples for contextual alignment.

However, the improvement plateaus beyond 2-shot, suggesting that the added examples primarily enhance pattern recall rather than strengthening the model's underlying compositional reasoning. In conclusion, while few-shot prompting mitigates cold-start difficulties in dynamic Text-to-SQL

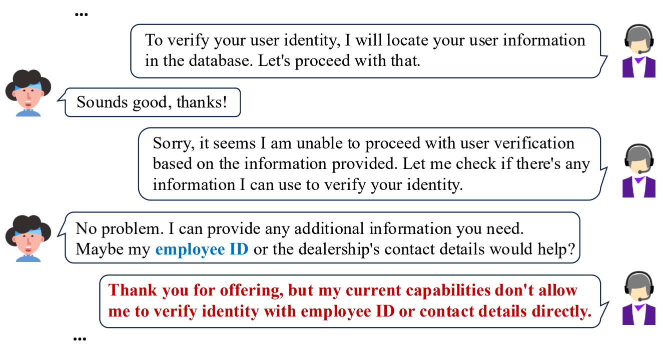

...
To verify your user identity, I will locate your user information in the database. Let's proceed with that.
Sounds good, thanks!
Sorry, it seems I am unable to proceed with user verification based on the information provided. Let me check if there's any information I can use to verify your identity.
No problem. I can provide any additional information you need. Maybe my employee ID or the dealership's contact details would help?
Thank you for offering, but my current capabilities don't allow me to verify identity with employee ID or contact details directly.
...

Fig. 12: Refusal to Use Available User Information for Identity Verification.

interaction, it remains insufficient for handling deeper schema reasoning or cross-domain transfer. Future work may integrate schema-aware pretraining or reinforcement learning-based adaptation to further enhance model robustness under dynamic contexts.

VII. Conclusion

In this work, we presented DySQL-Bench, the first benchmark designed to evaluate large language models under dynamic, multi-turn Text-to-SQL interaction scenarios. Unlike previous static datasets, DySQL-Bench captures the evolving nature of real-world database interactions where user intents, constraints, and analytical objectives continuously change over time. To construct this benchmark at scale and with high reliability, we proposed a two-stage automatic task synthesis and verification pipeline, which transforms raw database tables into structured tree representations and generates realistic interaction sequences through large language models, followed by interaction-oriented quality control and expert validation.

To enable realistic assessment, we further introduced a multi-turn user–model–database interaction framework that simulates natural dialogue between a user and a model in an executable database environment. This setting allows systematic evaluation of models' ability to perform query reformulation, contextual reasoning, and adaptive error recovery across turns. Comprehensive experiments across 13 domains and 1,072 tasks, reveal that even frontier models such as GPT-4o, highlighting the substantial challenges of interactive SQL reasoning. We hope DySQL-Bench will serve as a standardized and challenging platform for advancing research in dynamic Text-to-SQL, interactive data analysis, and context-aware reasoning, paving the way toward truly adaptive and conversational database intelligence.

References

[1] G. Katsogiannis-Meimarakis and G. Koutrika, "A survey on deep learning approaches for text-to-sql," *The VLDB Journal*, vol. 32, no. 4, pp. 905–936, 2023.

[2] L. Shi, Z. Tang, N. Zhang, X. Zhang, and Z. Yang, "A survey on employing large language models for text-to-sql tasks," *ACM Computing Surveys*, vol. 58, no. 2, pp. 1–37, 2025.

[3] C. Zhang, X. Dai, Y. Wu, Q. Yang, Y. Wang, R. Tang, and Y. Liu, "A survey on multi-turn interaction capabilities of large language models," *arXiv preprint arXiv:2501.09959*, 2025.

[4] H. Li, S. Wu, X. Zhang, X. Huang, J. Zhang, F. Jiang, S. Wang, T. Zhang, J. Chen, R. Shi *et al.*, "Omnisql: Synthesizing high-quality text-to-sql data at scale," *arXiv preprint arXiv:2503.02240*, 2025.

[5] P. Ma, X. Zhuang, C. Xu, X. Jiang, R. Chen, and J. Guo, "Sql-r1: Training natural language to sql reasoning model by reinforcement learning," *arXiv preprint arXiv:2504.08600*, 2025.

[6] M. Pourreza, S. Talaei, R. Sun, X. Wan, H. Li, A. Mirhoseini, A. Saberi, S. Arik *et al.*, "Reasoning-sql: Reinforcement learning with sql tailored partial rewards for reasoning-enhanced text-to-sql," *arXiv preprint arXiv:2503.23157*, 2025.

[7] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," *arXiv preprint arXiv:1809.08887*, 2018.

[8] F. Lei, J. Chen, Y. Ye, R. Cao, D. Shin, H. Su, Z. Suo, H. Gao, W. Hu, P. Yin *et al.*, "Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows," *arXiv preprint arXiv:2411.07763*, 2024.

[9] J. Li, B. Hui, G. Qu, B. Li, J. Yang, B. Li, B. Wang, B. Qin, R. Cao, R. Geng *et al.*, "Can llm already serve as a database interface," *A big bench for large-scale database grounded text-to-sqls. CoRR, abs/2305.03111*, 2023.

[10] T. Yu, R. Zhang, H. Y. Er, S. Li, E. Xue, B. Pang, X. V. Lin, Y. C. Tan, T. Shi, Z. Li *et al.*, "Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases," *arXiv preprint arXiv:1909.05378*, 2019.

[11] C. Li, Y. Wang, Z. Wu, Z. Yu, F. Zhao, S. Huang, and X. Dai, "Multisql: A schema-integrated context-dependent text2sql dataset with diverse sql operations," in *Findings of the Association for Computational Linguistics ACL 2024*, 2024, pp. 13 857–13 867.

[12] B. Qin, B. Hui, L. Wang, M. Yang, J. Li, B. Li, R. Geng, R. Cao, J. Sun, L. Si *et al.*, "A survey on text-to-sql parsing: Concepts, methods, and future directions," *arXiv preprint arXiv:2208.13629*, 2022.

[13] R. An, S. Yang, R. Zhang, Z. Shen, M. Lu, G. Dai, H. Liang, Z. Guo, S. Yan, Y. Luo *et al.*, "Unictokens: Boosting personalized understanding and generation via unified concept tokens," *arXiv preprint arXiv:2505.14671*, 2025.

[14] R. An, S. Yang, M. Lu, R. Zhang, K. Zeng, Y. Luo, J. Cao, H. Liang, Y. Chen, Q. She *et al.*, "Mc-llava: Multi-concept personalized vision-language model," *arXiv preprint arXiv:2411.11706*, 2024.

[15] Y. Luo, R. An, B. Zou, Y. Tang, J. Liu, and S. Zhang, "Llm as dataset analyst: Subpopulation structure discovery with large language model," in *European Conference on Computer Vision*. Springer, 2024, pp. 235–252.

[16] W. Lin, X. Wei, R. An, T. Ren, T. Chen, R. Zhang, Z. Guo, W. Zhang, L. Zhang, and H. Li, "Perceive anything: Recognize, explain, caption, and segment anything in images and videos," 2025. [Online]. Available: https://arxiv.org/abs/2506.05302

[17] W. Lin, X. Wei, R. An, P. Gao, B. Zou, Y. Luo, S. Huang, S. Zhang, and H. Li, "Draw-and-understand: Leveraging visual prompts to enable mllms to comprehend what you want," *arXiv preprint arXiv:2403.20271*, 2024.

[18] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[19] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[20] M. Chen, L. Sun, T. Li, H. Sun, Y. Zhou, C. Zhu, H. Wang, J. Z. Pan, W. Zhang, H. Chen *et al.*, "Learning to reason with search for llms via reinforcement learning," *arXiv preprint arXiv:2503.19470*, 2025.

[21] L. Sun, H. Liang, J. Wei, B. Yu, C. He, Z. Zhou, and W. Zhang, "Beats: Optimizing llm mathematical capabilities with backverify and adaptive disambiguate based efficient tree search," *arXiv preprint arXiv:2409.17972*, 2024.

[22] J. Li, X. Li, G. Qu, P. Jacobsson, B. Qin, B. Hui, S. Si, N. Huo, X. Xu, Y. Zhang *et al.*, "Swe-sql: Illuminating llm pathways to solve user sql issues in real-world applications," *arXiv preprint arXiv:2506.18951*, 2025.

[23] M. Pourreza and D. Rafiei, "Din-sql: Decomposed in-context learning of text-to-sql with self-correction," *Advances in Neural Information Processing Systems*, vol. 36, pp. 36 339–36 348, 2023.

[24] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, "Text-to-sql empowered by large language models: A benchmark evaluation," *arXiv preprint arXiv:2308.15363*, 2023.

[25] M. Pourreza and D. Rafiei, "Dts-sql: Decomposed text-to-sql with small large language models," *arXiv preprint arXiv:2402.01117*, 2024.

[26] B. Wang, C. Ren, J. Yang, X. Liang, J. Bai, L. Chai, Z. Yan, Q.-W. Zhang, D. Yin, X. Sun *et al.*, "Mac-sql: A multi-agent collaborative framework for text-to-sql," *arXiv preprint arXiv:2312.11242*, 2023.

[27] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *The eleventh international conference on learning representations*, 2022.

[28] Z. Hong, Z. Yuan, Q. Zhang, H. Chen, J. Dong, F. Huang, and X. Huang, "Next-generation database interfaces: A survey of llm-based text-to-sql," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[29] B. Zhang, Y. Ye, G. Du, X. Hu, Z. Li, S. Yang, C. H. Liu, R. Zhao, Z. Li, and H. Mao, "Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation," *arXiv preprint arXiv:2403.02951*, 2024.

[30] A. Mitsopoulou and G. Koutrika, "Analysis of text-to-sql benchmarks: limitations, challenges and opportunities," in *Proceedings 28th International Conference on Extending Database Technology, EDBT 2025*. OpenProceedings. org, 2025, pp. 199–212.

[31] A. Bhaskar, T. Tomar, A. Sathe, and S. Sarawagi, "Benchmarking and improving text-to-sql generation under ambiguity," *arXiv preprint arXiv:2310.13659*, 2023.

[32] P. Price, "Evaluation of spoken language systems: The atis domain," in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.

[33] J. M. Zelle and R. J. Mooney, "Learning to parse database queries using inductive logic programming," in *Proceedings of the national conference on artificial intelligence*, 1996, pp. 1050–1055.

[34] S. Yao, N. Shinn, P. Razavi, and K. Narasimhan, "τ-bench: A benchmark for tool-agent-user interaction in real-world domains," *arXiv preprint arXiv:2406.12045*, 2024.

[35] V. Barres, H. Dong, S. Ray, X. Si, and K. Narasimhan, "tau2-bench: Evaluating conversational agents in a dual-control environment," *arXiv preprint arXiv:2506.07982*, 2025.

[36] Z. Yi, J. Ouyang, Z. Xu, Y. Liu, T. Liao, H. Luo, and Y. Shen, "A survey on recent advances in llm-based multi-turn dialogue systems," *arXiv preprint arXiv:2402.18013*, 2024.

[37] Y. Li, X. Shen, X. Yao, X. Ding, Y. Miao, R. Krishnan, and R. Padman, "Beyond single-turn: A survey on multi-turn interactions with large language models," *arXiv preprint arXiv:2504.04717*, 2025.

[38] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford *et al.*, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.

[39] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan *et al.*, "Deepseek-v3 technical report," *arXiv preprint arXiv:2412.19437*, 2024.

[40] Q. Team, "Qwen2.5: A party of foundation models," September 2024. [Online]. Available: https://qwenlm.github.io/blog/qwen2.5/

[41] R. Vavekanand and K. Sam, "Llama 3.1: An in-depth analysis of the next-generation large language model," *Preprint, July*, 2024.

[42] G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen *et al.*, "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," *arXiv preprint arXiv:2507.06261*, 2025.

[43] L. Zheng, L. Yin, Z. Xie, C. L. Sun, J. Huang, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez *et al.*, "Sglang: Efficient execution of structured language model programs," *Advances in neural information processing systems*, vol. 37, pp. 62 557–62 583, 2024.

[44] Y. Bang, Z. Ji, A. Schelten, A. Hartshorn, T. Fowler, C. Zhang, N. Cancedda, and P. Fung, “Hallulens: Llm hallucination benchmark,” *arXiv preprint arXiv:2504.17550*, 2025.

[45] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu *et al.*, “Deepseekmath: Pushing the limits of mathematical reasoning in open language models,” *arXiv preprint arXiv:2402.03300*, 2024.