SQLAgent：学会在生成前像数据工程师一样探索

Wenjia Jiang[1] **Yiwei Wang**[2] **Boyan Han**[1] **Joey TianyiZhou**[3] **ChiZhang**[11]AGI Lab，西湖大学，中国 [2]加州大学默塞德分校，美国 [3]前沿人工智能研究中心（CFAR）与 CATOS，A*STAR，新加坡

jwj57920@gmail.com

arXiv:2602.01952v1 [cs.DB] 2 Feb 2026

摘要

大语言模型最近在推理和代码生成方面展现出令人印象深刻的能力，使其成为关系型数据库自然语言接口的有前景工具。然而，由于SQL推理高度依赖数据库特定知识，需要深入熟悉独特的模式、模糊的语义以及复杂的连接路径，现有方法往往难以在复杂的真实场景中泛化。为应对这一挑战，我们提出一种新颖的两阶段大语言模型框架，将知识获取与查询生成解耦。在探索阶段，系统采用受蒙特卡洛树搜索启发的策略自主导航模式，构建数据库专属知识库，并生成由模式片段、可执行查询和自然语言描述组成的三元组作为使用示例。在部署阶段，双智能体系统将所收集的知识作为上下文示例，迭代检索相关信息，并针对用户问题生成准确的SQL 查询。该设计使智能体能够主动熟悉未见过的数据库，并处理复杂的多步推理。在大规模基准测试上的大量实验表明，我们的方法相较强基线显著提升了准确率，凸显了其有效性与泛化能力。

1 引言

大语言模型（LLM）在复杂推理任务中展现出了卓越的能力（OpenAI，2023 年；Bubeck等人，2023；Mirchandani等人，2023；Wu等人，2023；Meta基础人工智能研究（FAIR）外交团队等人，2022）。高价值信息主要存储在关系型数据库中（Verbitski等人，2017 年；Yavuz等人，2018），因此，利用大语言模型与结构化数据交互受到了广泛关注。这一兴趣推动了文本到 SQL 研究的发展，其目标是将自然语言问题转换为可执行的 SQL 查询，从而让更多人能够访问数据（Liu等人，2025年；Katsogiannis-Meimarakis和Koutrika，2023；Kobayashi等人，2025年；Malekpour等人，2024年；Shi等人，2025年；Deng等人，2022）。其首要目标是在无需用户掌握复杂语法的前提下，弥合人类语言与数据库之间的鸿沟。

然而，当前方法难以泛化到复杂场景（Lei 等人，2025年）。在真实世界的数据库上，性能往往会下降，暴露出显著的泛化差距（Pourreza 和 Rafiei，2024年；Gao等人，2024年；Talaei等人，2024年）。这一局限源于 SQL 推理本质上针对特定数据库，而非具备可移植性。有效查询严格依赖独特的模式结构，这与通用代码生成不同。因此，模型在浏览复杂模式结构以及根据特定列名消解语义歧义方面面临挑战。此外，推理还需要与独特的数据库连接路径和关系精确对齐。

如果不熟悉数据库，通用预训练模型就难以理解新数据库独特的结构和含义（Lei 等人，2025年）。相比之下，人类专家会先深入熟悉数据库独特的模式和关系，从而成功完成任务。这一核心洞见启发了

我们的方法：在尝试最终的翻译任务之前，先通过一个预备流程构建这种基础知识。为实现这一目标，我们提出了一种新颖的基于大语言模型的智能体框架，分两个关键阶段运行。首先，框架自主探索陌生数据库，生成丰富的数据库专属知识库。随后，这些知识会作为上下文示例提供给智能体（Dong 等人，2024年；Rubin 等人，2022；Zhang 等人，2022b）。这一流程引导智能体生成最终的复杂 SQL 查询。

探索阶段会自主构建一个结构化的数据库专属知识库。该阶段的核心目标是为数据库模式的每个关键组件（例如其表和列）生成丰富的使用示例。每个示例都被形式化为一个三元组：一个模式子结构、一条相应的可执行 SQL 查询及其自然语言描述。为了系统地在整个模式中生成这些三元组，我们首先将数据库表示为一棵可遍历的树结构，其中表和列等实体被组织为节点。这棵树为我们的探索提供了一张地图。为了生成三元组，智能体需要探索这棵树，以找到有意义的节点并将其组合成有效的查询。为此，我们随后采用一种受蒙特卡洛树搜索（Kocsisand Szepesvári，2006年；Coulom，2007年）启发、并由作为核心推理引擎的智能体引导的搜索策略。智能体智能地遍历这棵树，以构建并测试新的查询。它通过选择一系列动作来实现这一点，例如引入连接或添加筛选条件。每条成功的探索路径都会产生一个新的三元组。这些三元组被收集起来以构建我们的知识库，使下一阶段能够更好地编写准确的 SQL 查询。总体而言，这一过程使系统能够在无需任何人工干预的情况下，主动熟悉未知的数据库。

在部署阶段，我们的目标是有效利用探索阶段获得的知识来处理复杂的用户查询。为此，我们提出了一种双智能体框架，由 InfoAgent 和 GenAgent 分工协作。InfoAgent 根据用户的问题，从先前构建的数据库中检索最相关的知识三元组。随后，GenAgent 使用这一模式片段，检索探索阶段存储的相关知识三元组。然后，GenAgent 将检索到的三元组纳入推理上下文，并将其作为上下文示例来引导最终的查询生成。如果生成的查询执行失败，GenAgent 会向 InfoAgent 提供反馈，促使其开启新一轮信息检索。之后，两个智能体通过迭代循环协同工作，不断完善 SQL 查询，循环过程包括示例检索、生成和执行反馈。这种协作式多步骤设计使系统能够拆解问题，从而实现高准确性和可靠性。

为验证我们方法的有效性，我们开展了大量实验，结果表明，该方法在复杂的大规模基准测试集上显著优于强基线。

主要贡献

1. 我们提出了一种新颖的两阶段大语言模型框架，将知识获取与查询生成解耦。2. 我们设计了一种自主的智能体驱动的探索策略，无需人工标注即可构建结构化、可执行的知识库。3. 我们引入了一种双智能体推理机制，通过迭代检索和整合上下文示例，生成准确且可执行的 SQL 查询。

2 相关工作

基于大语言模型的文本到 SQL 方法。 文本到 SQL 领域的研究已从早期的神经解析器发展到现代由大语言模型驱动的方法（Shi等人，2025年；Deng等人，2022）。开创性的神经方法引入了基于图的编码器等技术，以利用数据库模式（Wang et al.，2020），并采用受约束解码来确保语法正确性（Scholaket al.，2021）。随着大语言模型的出现，该领域取得了显著进展。众多微调方法（Li 等人，2024年）和先进的大语言模型提示技术（Dong et al.，2023年；Wang et al.，2023a；Zhang et al.，2023年；Talaei等人，2024年；

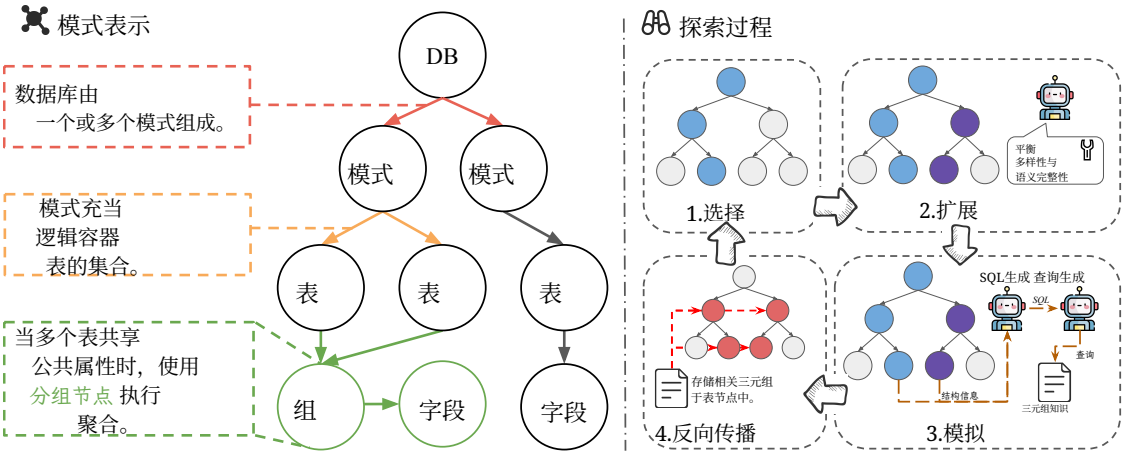


图1： **数据库表示与探索阶段。** 该图左侧展示了我们对数据库结构的表示与处理过程。右侧展示了我们在现有结构上使用树搜索进行探索的示意图。这包括四个阶段：选择、扩展、模拟和反向传播。该方法能够收集一系列三元组。

Pourreza 和 Rafiei，2024年；Gao等人，2024年）在现有基准测试集上取得了出色的表现。然而，这些方法仍面临诸多挑战，例如上下文窗口有限，以及难以适应未见过的数据库领域。

用于代码生成的多智能体。生成模型与交互式问题求解的交汇，推动了旨在增强语言模型推理能力的基于智能体的框架迅速发展，尤其是在代码生成方面（Yao 等人，2023年；Zhang 等人，2022a；Chen 等人，2023年；Wang 等人，2023b；Shinn 等人，2024年；Zhang 等人，2024年；Xia 等人，2024年）。为了克服单一模型的局限，一种重要策略是部署多个分工协作的智能体。在这种多智能体范式中，各智能体承担专门角色并相互交互，通常采用协作式分解（Li 等人，2023a）或分层任务委派（langgenius，2023年）等模式。为了使这些交互更加稳健，其他研究则致力于设计专门的动作空间，以规范智能体操作（Wang 等人，2024年；Yang 等人，2024年）。受这些成果启发，我们的方法采用了类似策略，为文本到SQL转换配备专门的探索智能体和生成智能体。

3 方法论

我们的工作通过提出一种新颖的两阶段框架，解决将文本到 SQL 系统泛化到陌生数据库这一挑战。首先，**探索阶段** 会自主构建数据库专属知识库，无需人工监督。随后，**部署阶段** 采用双智能体框架，有效利用所获取的知识，实现准确且稳健的查询合成。

3.1 数据库模式表示

我们的数据库探索方法首先将传统的关系模式转换为可遍历的树状结构。数据库、其中的表及其对应的字段都被视为树中的独立节点，以明确呈现它们的嵌套关系。然而，大型数据库中常见的结构冗余带来了重大挑战。例如，基于时间的分片通常会生成许多模式相同的表，例如每日日志或每小时快照。若天真地将每个表单独建模，就必须将每个表节点都连接到各自的字段节点，从而导致表示复杂度和冗余度很高。为了解决这一问题，我们引入了一种名为**共享字段组**的抽象。该组件充当规范且可复用的模板，用于封装一组结构相同的表所共有的字段结构。形式化地，设 $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ 为

数据库中的表集合。每个表 T_i 都由其模式签名 $S_i = \{(f_{i,j}, \tau_{i,j})\}_{j=1}^{m}$定义，其中 $f_{i,j}$ 和 $\tau_{i,j}$ 分别表示第 j 个字段的名称和数据类型。我们将**共享字段组**$\mathcal{G}_k$ 定义为满足以下条件的表的等价类：

$$\mathcal{G}_k = \{T_i \in \mathcal{T} \mid \text{Hash}(\text{sort}(S_i)) = \text{Hash}(\text{sort}(S_k))\} \tag{1}$$

其中 Sort$(\cdot)$ 确保签名对列顺序保持不变，而 Hash$(\cdot)$ 为结构模式提供唯一的 128 位标识符。因此，每个表节点不再维护与其字段节点之间的大量独立连接，而是与适当的**共享字段组**建立单一链接。如图 1所示，这种设计显著简化了整体模式表示，并阐明了具有相同结构的表之间固有的关系。通过抽象共同结构，这种方法将表示表与字段之间关系的复杂度从$O(N \times M)$降低至 $O(N + M)$，其中 N是分片表的数量，而 M是共享字段的数量。这种抽象不仅简化了模式，还提高了后续探索过程的可解释性和效率。该结构的具体定义和识别算法详见附录 C 和附录D。

3.2 探索阶段：LLM引导的树搜索

探索阶段的目标是系统地获取关于目标数据库的先验知识。这些知识不仅应捕捉静态的数据库模式结构，还应描述其组件如何组合以形成有意义的查询。具体而言，我们旨在构建一个知识库，将可行的结构模式转化为可执行的 SQL 查询，并使其与自然语言语义对齐。我们将探索结果形式化为一组三元组（S， Q， U），其中 S表示数据库模式结构的一个子集， Q是相应的可执行 SQL 查询，而 U 是与 Q在自然语言语义上对齐的内容。

为应对众多模式组件及其组合可能性所带来的巨大复杂性，我们提出了一种受蒙特卡洛树搜索（KocsisandSzepesvári，2006年）启发的新型探索框架，并以大语言模型作为语义推理引擎，取代传统的基于启发式的搜索策略。与依赖树上的置信上限（UCT）进行标量奖励反向传播的传统蒙特卡洛树搜索不同，我们的框架采用**大语言模型即策略与评估器**范式。这一选择源于 SQL 推理的内在特性：SQL有效性虽然可以通过执行判断，但其语义空间具有高度连续性，并且取决于上下文。因此，大语言模型提供的高维自然语言反馈和错误跟踪信息，比稀疏的数值奖励更能充当信息丰富的启发式引擎，使智能体能够以更深的语义层次探索复杂模式。整个探索过程将持续进行，直到满足预先设定的终止条件，例如生成目标数量的有效三元组或达到最大迭代次数。这种大语言模型驱动的方法保留了结构化的迭代探索周期，使系统能够在模型理解的引导下逐步构建复杂查询，而不再仅仅依赖数值奖励。该过程如图1所示，包含四个不同阶段，分别是大语言模型引导的选择与扩展、模拟，以及对结果进行反向传播。

LLM引导的选择与扩展。 此阶段的目标是从数据库模式中逐步构建复杂且语义多样的 SQL 查询。我们的方法通过利用大语言模型，在单个经过推理的步骤中同时执行选择与扩展来实现这一目标。在搜索树的每个节点处，都会向大语言模型提供必要的上下文，以便其做出决策。该上下文包括当前查询状态（一系列先前的操作）以及可用的模式上下文（由可访问表和列构成的简化 JSON 结构）。大语言模型的任务是从预定义的离散动作空间中选择最有希望的后续操作，选项包括 选择列、 添加约束和 应用聚合。附录 E中提供了每个操作的详细描述。大语言模型的选择会直接创建一个新节点，从而扩展搜索树。此次扩展的目标是引导搜索树不断发展，构建更加复杂的查询。该策略确保生成语义丰富的示例，覆盖广泛的数据库操作和模式交互，超越简单的单列检索。

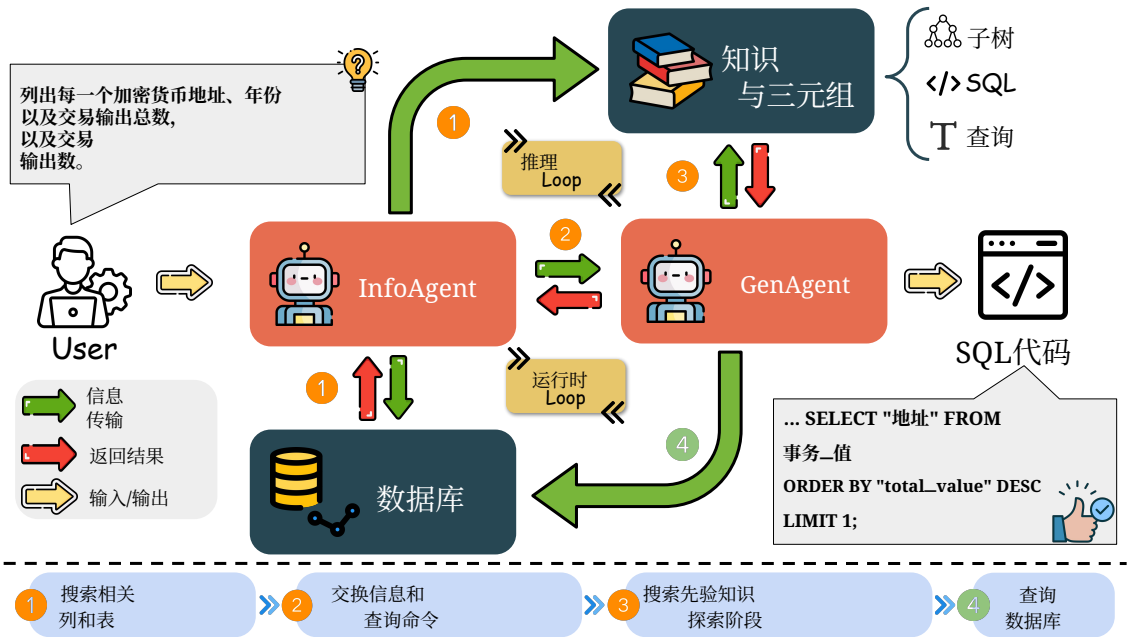


图 2：SQL 部署阶段。在此阶段，利用探索阶段获取的数据库信息和用户的实际查询来生成 SQL。双智能体架构负责控制智能体的信息获取上下文和 SQL 生成上下文，使系统能够处理复杂的 SQL 语句，同时保持较高的查询准确率。

SQL 模拟。 从新扩展的节点开始，模拟阶段利用大语言模型生成完整且可执行的查询。系统会提示模型构建一个与当前节点所表示结构在语义上保持一致的 SQL 查询。为了引导这一过程生成有意义的查询，大语言模型会被要求优先使用具有相关文档或注释的表和列。这些元数据为构建更相关的查询提供了有价值的语义线索。该过程基于动态约束的模式节点集合运行，并排除那些已经纳入当前查询路径的节点（ID 等键列除外），以确保广泛探索。随后，大语言模型利用扩展节点中的上下文构建正确的 SQL 查询。例如，如果扩展节点包含用户表中的列 age、height 和 gender，大语言模型可能会通过生成 SQL 查询 SELECT * FROM users WHERE age > 20 AND gender = 'Male'; 来补全该结构，并生成相应的自然语言描述："查找所有年龄大于 20 岁的男性用户。"

生成的查询 Q 将经过多层验证流程。首先，通过 SQL 解析器进行**语法正确性检查**，确保其符合数据库特定的方言。其次，在实时数据库上执行查询，以验证其**执行有效性**。我们仅在模拟满足以下三个条件时才判定其成功：(1) 语法有效；(2) 执行时没有运行时错误；(3) 返回非空且有意义的结果集。若满足这些条件，我们便将结构信息 S 和 SQL 查询 Q 作为输入提供给大语言模型，生成相应的自然语言内容 U，从而确保高质量的语义对齐。如果查询执行失败或返回空结果，模拟便会结束，并将这一结果传递至下一阶段。

我们的方法首先定义有效且局部化的模式结构，然后据此生成相应的 SQL 和自然语言描述，从而产出准确可靠的示例。由于每个示例都以真实的数据子结构为基础，因此能够确保高度的语义对齐，并在部署阶段充当可靠的参考。

反向传播。 反向传播阶段会根据模拟结果，系统地记录探索结果，并更新路径上每个节点的历史上下文，以指导部署阶段。对于成功的探索，如果得到有效的 (S, Q, U) 三元组，就会将其记录为正向反馈，并存储在所有相关实体节点上，例如表和列。这个过程会为每个模式

组件积累其在有意义组合中成功使用的历史记录。相反，如果探索因查询无效或结果为空而失败，就会传播负向反馈，降低该路径的选择优先级，以避免重复尝试无效组合。这些累积的信息对于自适应学习至关重要：在后续步骤中，LLM提示会加入来自可见节点的反馈。通过从过去行动的结果中学习，大语言模型会不断完善策略，优先选择能够得到有效且非空结果的路径，从而形成自我优化的探索过程。

3.3 部署阶段：双智能体 SQL 合成

为了弥合用户查询与复杂模式之间的语义差距，我们提出了一种双智能体框架（图2），并通过迭代优化上下文并生成 SQL。

InfoAgent：模式定位与上下文管理。收到查询后，InfoAgent 会构建精确的模式上下文。首先，它通过对模式名称、类型和描述的预计算嵌入进行语义搜索，检索排名前 k 的相关列，以完成模式定位。为弥补简单检索可能遗漏隐式依赖的问题，该智能体会执行上下文扩展步骤。在此过程中，大语言模型会结合初始检索结果与用户查询，推断逻辑上必需的组件，例如连接所需的外键，从而确保上下文可执行。

GenAgent：知识驱动合成。GenAgent 利用优化后的模式上下文和检索到的知识合成 SQL 查询。它通过将当前用户查询的语义意图与向量化查询进行匹配来检索示例 Q ，这些查询k存储在我们的三元组知识库中。检索到的排名前列的三元组可作为特定数据库的上下文示例，GenAgent 会将其与用户查询和模式上下文结合，生成最终 SQL。详细策略和模板见附录 B。

协作式细化循环。该流程采用反馈循环来处理错误。如果生成的 SQL 执行失败，InfoAgent **会通过移除未使用的模式项来**精简上下文，并重新分析查询以找出被忽略的关键词。反之，执行成功后会触发**语义忠实度检查**，以验证结果是否符合用户意图。该循环会持续进行，直至确认查询有效或达到迭代次数上限，详情见附录F。

4 实验

本节将全面评估我们提出的两阶段方法。我们的主要目标是评估 SQLAgent 的有效性和效率。我们首先详细介绍实验设置，随后深入分析对比实验和消融研究的结果。完整的实验细节请参见附录 A 。

4.1 实验设置

基准测试集。我们在 Spider 2.0-Snow 基准测试集（Lei 等人，2025年）上评估所提出的方法。该基准测试集包含大量企业级 SQL 查询，其中一些查询超过 100 行，复杂度显著高于传统的文本到 SQL 任务。每个子任务包含 547 个示例，涵盖 150 多个数据库，每个数据库平均约有 800 列。与Lei 等人（2025年）一致，我们根据词元数量对 SQL 查询的难度进行分类：简单（少于 80 个词元）、中等（80 至 159 个词元）和困难（160 个或更多词元）。

评估指标。 我们使用广泛采用的执行准确率（EX）指标评估所提方法的性能（Li 等人，2023b；Yu 等人，2018；Lei 等人，2025年）。执行准确率通过比较给定数据库实例上预测的 SQL 查询与真实查询的执行结果来计算。该指标能够更精确地估计模型性能，因为可能存在多个有效的

表1：不同方法的性能比较。 此表详细比较了执行成功率（EX）和效率指标。EX 按简单、中等和困难难度等级以及总体得分以百分比形式报告。效率以每个查询平均调用大语言模型（LLM）和数据库（DB）的次数衡量。我们评估了所提出的 SQLAgent 及其不同组件，并将其与其他基线方法和最先进方法进行比较。

方法	策略	执行成功率 (%)				效率	
		Easy	中等	Hard	总体	大语言模型调用次数	数据库调用次数
Spider-Agent	智能体式	24.22	11.38	6.94	12.98	11	3
ReFoRCE	共识	49.22	17.00	5.23	20.84	3.5	3.9
SQLAgent	基线	32.81	14.57	0.00	14.26	3.0	4.2
	含实验	41.40	**19.03**	5.81	20.10	4.1	5.2
	完整方法	**57.81**	18.62	**12.21**	**25.78**	5.2	3.6

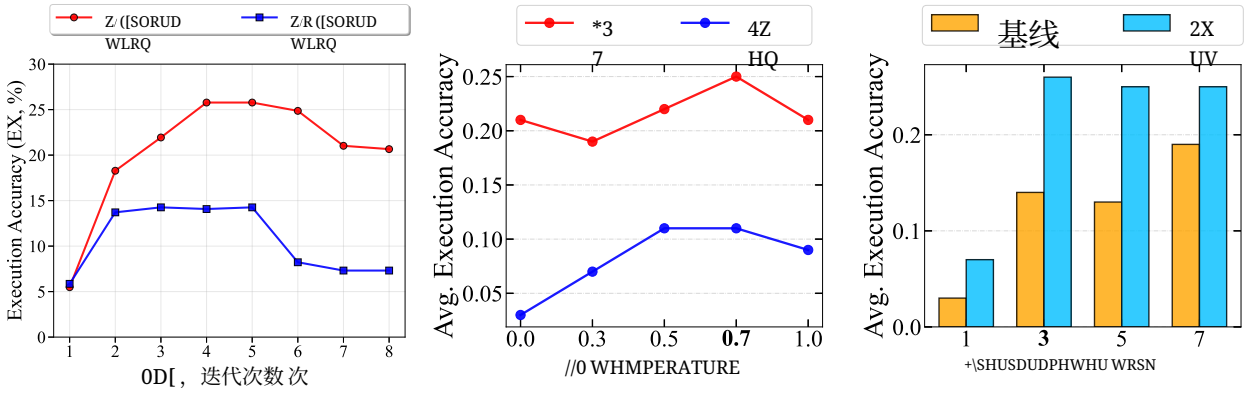
针对单个问题生成的 SQL 查询。为考虑大语言模型（LLM）的随机性，我们还报告 PASS@K，该指标衡量在 K 次运行中能否获得正确结果，从而评估重复执行时输出的稳定性。为评估模型的整体效率，我们统计对大语言模型的调用次数以及发送到数据库的执行次数。随后，我们报告基准测试集中所有任务的上述次数的平均值。

基线。 我们的基线方法受到Deng等人(2025年)的启发； Yu 等人(2018)采用自我细化机制来处理大型数据库模式。首先，我们通过拼接表名和列名构建模式的文本索引。用户提交查询后，系统会从该索引中检索最相关表的数据库定义语言（DDL），并将其作为上下文提供给大语言模型。如果执行尝试返回空结果或产生语法错误，大语言模型便会启动细化循环。它会利用执行反馈反复尝试修正 SQL 查询，而不会再次查询数据库结构。此过程会持续进行，直至生成有效查询或满足预设的停止条件。

4.2 结果与分析

本节评估所提出的两阶段框架，重点考察其有效性、效率和稳健性。除非另有说明，准确率均以执行成功率（EX）表示，效率则通过实际运行时间和模型令牌成本进行评估。

对比结果。 为评估所提出框架的有效性和效率，我们将 SQLAgent 与两个强大的基线进行了比较，`Spider-Agent` 和 ReFoRCE，详细结果见表1。我们的方法显著优于两个基线，整体执行准确率（EX）达到**25.78%**，而 ReFoRCE 为 20.84%，Spider-Agent 为 12.98%。在复杂查询上的性能差距尤为明显；在“Hard”子集上，我们的方法执行成功率（EX）达到**12.21%**，表明其相较于难以处理这些任务的竞争方法有明显提升。除了这些数值结果，附录例如H 中还对 SQLAgent 与其他近期推理时框架（如 Din-SQL、C3 和 Chess）进行了详细的概念比较，附录I.2 还对剩余失败模式进行了全面分析。为单独评估框架各组件的贡献，我们进行了消融实验。基线模型的整体执行成功率（EX）为 14.26%，在困难查询上则完全无法成功。加入探索阶段（w/ Exp. Stage）获得的知识后，整体准确率提升至**20.10%**，证实了主动构建知识库的价值。最后，整合部署阶段双智能体框架的完整方法取得了最高准确率**25.78%**。这一最终性能提升验证了双智能体架构能够有效利用所获取的知识，实现稳健的 SQL 合成。在效率方面，我们的完整方法平均需要**5.2**次大语言模型调用和**3.6** 次数据库调用，详见表**1**。这一配置明显比 Spider-Agent 更高效，后者需要调用大语言模型 11 次。虽然其大语言模型调用次数多于 ReFoRCE（3.5 次），但为达到更高的准确率，所需的数据库交互次数更少（3.9 次）。消融



(a)迭代期间的性能。 **(b)** 对两个关键超参数的分析

图3： 探索阶段与关键超参数分析。 左图展示了执行准确率随智能体迭代次数的变化，表明探索阶段始终能够提升性能。中图展示了大语言模型温度对不同模型的影响，其中 GPT 模型的准确率在温度为 0.7 时达到峰值。右图评估了 top-k 模式检索参数，结果显示，我们的方法在所有设置下均优于基线，且当 k 值大于 3 时，性能增益趋于饱和。

实验进一步阐明了这一权衡：引入探索阶段会使成本略高于基线。然而，随后加入双智能体框架可将数据库调用次数从 5.2 次减少至 3.6 次。这表明，该框架的结构化推理不仅提升了准确率，还减少了不必要的查询执行，从而验证了整体设计的有效性。

迭代期间的性能。 图 3a 追踪了双智能体循环中执行成功率随最大迭代预算的变化，并比较了我们的框架在有无探索阶段知识库时的性能。在此分析中，我们控制 InfoAgent 与 GenAgent 之间的协作轮次；达到迭代上限后，GenAgent 必须生成最终 SQL 查询。

结果清楚地表明了探索所得知识的价值。对于配备了这些知识的模型（红色曲线），准确率与迭代次数呈显著正相关，从 5.5% 上升至四次迭代时的峰值 25.8%。这一趋势表明，在每一轮循环中，智能体都能有效利用知识库，逐步优化上下文并检索相关示例，从而构建出更准确的查询。为定性展示这些获取的知识如何帮助解决复杂结构（例如嵌套 VARIANT 字段和按时间分片的关系），我们在附录 I 中提供了详细的案例研究。相比之下，未使用这些知识的模型（蓝色曲线）提升有限，准确率在远低得多的 14.1% 峰值处趋于平稳。更显著的是，其性能在五次迭代后急剧下降。这一现象表明，缺少经过验证的先验知识时，迭代优化过程的稳定性较差。智能体难以区分有效与无效的优化路径，而冗余或模糊反馈的累积甚至会导致它们将原本正确的查询修改为错误查询。

超参数分析。 我们通过实验确定两个关键超参数的最优设置：模式检索top-k和大语言模型温度。 k 参数定义了从数据库中检索的最相关列的数量，以处理用户的查询。为分析其影响，我们在不同的 k 值下，将我们的方法与基线的性能进行比较。如图3b右侧面板所示，我们的方法性能迅速提升，并在$k = 3$处趋于饱和。将 k 提高到3以上时，对于我们的方法而言，收益逐渐减少，而计算成本更高。相比之下，基线需要更大的 k，但受到大语言模型上下文窗口的限制。因此，我们选择 $k = 3$ 作为在准确性与效率之间取得平衡的最优设置。温度参数控制大语言模型输出的随机性（Peeperkorn等人，2024年）。图 3b 中间面板展示了温度对执行准确率的影响。我们观察到，较高的温度（例如，0.7）使

LLM	Cost	EX@8 (%)		Δ
		基线	完整方法	
Qwen2.5-7B	20.8k	7.12	14.99	+7.87
GPT-4o	15.2k	21.57	31.99	+10.42
GPT-5	12.5k	22.49	32.90	+10.70
Claude-Sonnet-4.0	11.6k	29.97	39.12	+9.15

表2： **不同大语言模型骨干的性能比较。** 该表比较了我们完整方法与基线在各种大语言模型上的表现。我们报告每查询成本（k 词元）和执行准确率（EX@8，%）。

大语言模型能够生成更多样且有效的搜索词。因此，我们在实验中将大语言模型温度设为0.7。

大语言模型骨干分析。 表 2 比较了我们完整方法与基线在四种不同大语言模型（LLM）骨干上的表现，并报告了每查询词元成本及经过 8 轮后执行准确率（EX@8）（OpenAI，2023年；Qwen 等人，2025年；Li 等人，2023b）。数据显示，无论底层大语言模型的能力如何，我们的框架都能带来稳定且显著的提升。如 Δ列所示，我们的方法在能力更强的模型上实现了约 **+10%** 的稳健绝对准确率提升。这种稳定的性能提升表明，我们结合主动知识探索与双智能体推理的两阶段架构，可作为适用于任何配套大语言模型的模型无关增强。值得注意的是，这种提升在较小模型上尤为显著：我们的方法使 Qwen2.5-7B-Instruct 的 EX@8 提升了一倍以上，从 7.12% 增至 **14.99%**。这表明，我们的结构化方法能够有效弥补较小模型推理能力的不足，凸显了其价值与可扩展性。

5 结论

我们提出了 SQLAgent，这是一个模拟人类数据工程师解决问题过程的框架。SQLAgent 首先自主探索数据库，以构建特定上下文知识，随后利用这些知识进行部署，从而为传统方法难以应对的大规模企业级数据库提供可扩展且自适应的解决方案。大量实验验证了这一方法，并证明它在具有挑战性的基准测试集上有效且稳健。最终，我们的工作标志着向打造更自主、能力更强的数据交互智能体迈出了重要一步，推动实现复杂结构化数据访问普及化的目标。

参考文献

Sébastien Bubeck、Varun Chandrasekaran、Ronen Eldan、Johannes Gehrke、Eric Horvitz、Ece Kamar、Peter Lee、Yin Tat Lee、Yuanzhi Li、Scott Lundberg 等。《通用人工智能的曙光：GPT-4 的早期实验》。*arXiv* 预印本 *arXiv:2303.12712*，2023年。

Xinyun Chen、Maxwell Lin、Nathanael Schaerli 和 Denny Zhou。教导大语言模型进行自我调试。载于计算语言学协会第*61*届年会，2023年。

Rémi Coulom。蒙特卡洛树搜索中的高效选择性与备份算子。载于 H. Jaap van den Herik、Paolo Ciancarini 和 H. H. L. M. (Jeroen) Donkers（编），计算机与游戏，第72–83页，柏林、海德堡，2007年。Springer Berlin Heidelberg。国际标准书号 978-3-540-75538-8。

Minghang Deng、Ashwin Ramachandran、Canwen Xu、Lanxiang Hu、Zhewei Yao、Anupam Datta 和 Hao Zhang。ReFoRCE：一种具备自我优化、共识强制和列探索功能的文本转 SQL 智能体，2025年。网址 https://arxiv.org/abs/2502.00675。

Naihao Deng、Yulong Chen 和 Yue Zhang。《文本到 SQL 的最新进展：现状与展望》。载于 Nicoletta Calzolari、Chu-Ren Huang、Hansaem Kim、James Pustejovsky、Leo Wanner、Key-Sun Choi、Pum-Mo Ryu、Hsin-Hsi Chen、Lucia Donatelli、Heng Ji、Sadao Kurohashi、Patrizia Paggio、Nianwen Xue、Seokhwan Kim、Younggyun Hahm、Zhong He、Tony Kyungil Lee、Enrico Santus、Francis Bond 和 Seung-Hoon Na（编），第*29*届国际计算语言学大会论文集，第 2166–2187 页，庆州，大韩民国，2022年10月。国际计算语言学委员会。网址https://aclanthology.org/2022.coling-1.190/。

Qingxiu Dong、Lei Li、Damai Dai、Ce Zheng、Jingyuan Ma、Rui Li、Heming Xia、Jingjing Xu、Zhiyong Wu、Baobao Chang、Xu Sun、Lei Li 和 Zhifang Sui。上下文学习综述。由 Yaser Al-Onaizan、Mohit Bansal 和 Yun-Nung Chen 编辑，*2024*年自然语言处理实证方法会议论文集，第 1107–1128 页，美国佛罗里达州迈阿密，2024年11月。计算语言学协会。数字对象标识符：10.18653/v1/2024.emnlp-main.64。网址：https://aclanthology.org/2024.emnlp-main.64/。

Xuemei Dong、Chao Zhang、Yuhang Ge、Yuren Mao、Yunjun Gao、Jinshu Lin、Dongfang Lou 等。C3：使用 ChatGPT 的零样本文本到 SQL。*arXiv* 预印本 *arXiv:2307.07306*，2023年。

Dawei Gao、Haibin Wang、Yaliang Li、Xiuyu Sun、Yichen Qian、Bolin Ding 和 Jingren Zhou。大语言模型赋能的文本到 SQL：基准评估。*VLDB Endowment* 会议录，17（5）：1132–1145，2024年。

Jeff Johnson、Matthijs Douze 和 Hervé Jégou。利用图形处理器进行十亿规模的相似性搜索。*IEEE Big Data* 汇刊，7（3）：535–547，2019年。

George Katsogiannis-Meimarakis 和 Georgia Koutrika。文本到 SQL 深度学习方法综述。*VLDB* 杂志，32（4）：905–936，2023年1月。ISSN 1066-8888。数字对象标识符：10.1007/s00778-022-00776-8。网址：https://doi.org/10.1007/s00778-022-00776-8。

Hideo Kobayashi、Wuwei Lan、Peng Shi、Shuaichen Chang、Jiang Guo、Henghui Zhu、Zhiguo Wang和Patrick Ng。You Only Read Once（YORO）：学习为文本到SQL内化数据库知识。载于Luis Chiruzzo、Alan Ritter和Lu Wang主编，计算语言学协会美洲国家分会*2025*年会议论文集：人类语言技术（第*1*卷：长篇论文），第1889–1901页，阿尔伯克基，新墨西哥州，2025年4月。计算语言学协会。国际标准书号979-8-89176-189-6。数字对象标识符：10.18653/v1/2025.naacl-long.94。网址 https://aclanthology.org/2025.naacl-long.94/。

Levente Kocsis 和 Csaba Szepesvári。基于赌博机的蒙特卡洛规划。载于 Johannes Fürnkranz、Tobias Scheffer 和 Myra Spiliopoulou 编，《机器学习：*ECML 2006*》，第 282–293 页，柏林、海德堡，2006年。Springer Berlin Heidelberg。国际标准书号 978-3-540-46056-5。

LangChain AI。 *LangGraph*：使用大语言模型构建有状态、多智能体应用，2024年。网址 https://github.com/langchain-ai/langgraph。

langgenius. *Dify*：开源大语言模型应用开发平台，2023年。网址https://github.com/langgenius/dify。

Fangyu Lei、Jixuan Chen、Yuxiao Ye、Ruisheng Cao、Dongchan Shin、Hongjin Su、Zhaoqing Suo、Hongcheng Gao、Wenjing Hu、Pengcheng Yin、Victor Zhong、Caiming Xiong、Ruoxi Sun、Qian Liu、Sida Wang 和 Tao Yu。Spider 2.0：评估语言模型在真实世界企业级文本到 SQL 工作流中的表现，2025年。网址 https://arxiv.org/abs/2411.07763。

Guohao Li、Hasan Abed Al Kader Hammoud、Hani Itani、Dmitrii Khizbullin 和 Bernard Ghanem。Camel：用于探索"大语言模型社会"的交流型智能体。载于第三十七届神经信息处理系统大会，2023a。

Haoyang Li、Jing Zhang、Hanbing Liu、Ju Fan、Xiaokang Zhang、Jun Zhu、Renjie Wei、Hongyan Pan、Cuiping Li 和 Hong Chen。《Codes：迈向构建用于文本到 SQL 的开源语言模型》。*ACM* 数据管理学报，2(3):1–28，2024年。

Jinyang Li、Binyuan Hui、Ge Qu、Jiaxi Yang、Binhua Li、Bowen Li、Bailin Wang、Bowen Qin、Ruiying Geng、Nan Huo、Xuanhe Zhou、Chenhao Ma、Guoliang Li、Kevin C.C. Chang、Fei Huang、Reynold Cheng 和 Yongbin Li。《大语言模型是否已能充当数据库接口？面向大规模数据库语境文本到 SQL 的大型基准测试》。载于神经信息处理系统第 *37* 届国际会议论文集，NIPS '23，美国纽约州雷德胡克，2023b。Curran Associates Inc.

Xinyu Liu、Shuyu Shen、Boyan Li、Peixian Ma、Runzhi Jiang、Yuxin Zhang、Ju Fan、Guoliang Li、Nan Tang 和 Yuyu Luo。《大语言模型时代的文本到 SQL 综述：我们身处何处，又将走向何方？》*IEEE* 知识与数据工程汇刊，37(10):5735–5754，2025年。数字对象标识符：10.1109/TKDE.2025.3592032。

Mohammadhossein Malekpour、Nour Shaheen、Foutse Khomh 和 Amine Mhedhbi。通过大语言模型路由优化 SQL 生成，2024年。网址 https://arxiv.org/abs/2411.04319。

Meta基础人工智能研究（FAIR）外交团队、Anton Bakhtin、Noam Brown、Emily Dinan、Gabriele Farina、Colin Flaherty、Daniel Fried、Andrew Goff、Jonathan Gray、Hengyuan Hu 等人。结合语言模型与战略推理，在外交游戏中达到人类水平的表现。《科学》，378(6624):1067–1074，2022年。

Suvir Mirchandani、Fei Xia、Pete Florence、Brian Ichter、Danny Driess、Montserrat Gonzalez Arenas、Kanishka Rao、Dorsa Sadigh 和 Andy Zeng。大语言模型作为通用模式机器。*arXiv*预印本 *arXiv:2307.04721*，2023年。

Neo4j公司 *Neo4j*图数据库，2024年。网址https://neo4j.com/。访问日期：2025年12月20日。

OpenAI。GPT-4技术报告。技术报告，OpenAI，2023年。

OpenAI。全新嵌入模型和 API更新，2024a。网址 https://openai.com/index/new-embedding-models-and-api-updates/。访问日期：2024年5月20日。

OpenAI。你好，GPT-4o，2024b。网址https://openai.com/index/hello-gpt-4o/。访问日期：2025年12月20日。

Max Peeperkorn、Tom Kouwenhoven、Dan Brown 和 Anna Jordanous。温度是大语言模型的创造力参数吗？2024年。网址 https://arxiv.org/abs/2405.00492。

Mohammadreza Pourreza 和 Davood Rafiei。Din-SQL：带自我纠正的文本到 SQL 分解式上下文学习。神经信息处理系统进展，36，2024年。

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, MingfengXue, Pei Zhang, Qin Zhu, RuiMen, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang

Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, 和 Zihan Qiu。Qwen2.5 技术报告，2025年。网址https://arxiv.org/abs/2412.15115。

Ohad Rubin、Jonathan Herzig 与 Jonathan Berant。学习检索提示以用于上下文学习。载于 Marine Carpuat、Marie-Catherine de Marneffe 与 Ivan Vladimir Meza Ruiz 编，*2022* 年计算语言学协会北美分会会议论文集：人类语言技术，第 2655–2671 页，美国西雅图，2022 年 7 月。计算语言学协会。数字对象标识符：10.18653/v1/2022.naacl-main.191。网址 https://aclanthology.org/2022.naacl-main.191/。

Torsten Scholak、Nathan Schucher 与 Dzmitry Bahdanau。PICARD：为从语言模型进行受约束的自回归解码而增量解析。载于 Marie-Francine Moens、Xuanjing Huang、Lucia Specia 与 Scott Wen-tau Yih 编，*2021* 年自然语言处理实证方法会议论文集，第 9895–9901 页，线上及多米尼加共和国蓬塔卡纳，2021 年 11 月。计算语言学协会。数字对象标识符：10.18653/v1/2021.emnlp-main.779。网址 https://aclanthology.org/2021.emnlp-main.779/。

Liang Shi、Zhengju Tang、Nan Zhang、Xiaotong Zhang 和 Zhi Yang。关于在文本到 SQL 任务中使用大语言模型的综述。*ACM* 计算机调查，58(2)，2025 年 9 月。ISSN 0360-0300。数字对象标识符：10.1145/3737873。网址 https://doi.org/10.1145/3737873。

Noah Shinn、Federico Cassano、Ashwin Gopinath、Karthik Narasimhan 和 Shunyu Yao。Reflexion：运用语言强化学习的语言智能体。神经信息处理系统进展，36，2024年。

Shayan Talaei、Mohammadreza Pourreza、Yu-Chen Chang、Azalia Mirhoseini 和 Amin Saberi。Chess：面向高效 SQL 合成的上下文利用。*arXiv* 预印本 *arXiv:2405.16755*，2024年。

Alexandre Verbitski、Anurag Gupta、Debanjan Saha、Murali Brahmadesam、Kamal Gupta、Raman Mittal、Sailesh Krishnamurthy、Sandor Maurice、Tengiz Kharatishvili 和 Xiaofeng Bao。Amazon Aurora：高吞吐量云原生关系数据库的设计考量。载于 *2017* 年 *ACM* 数据管理国际会议论文集，SIGMOD '17，第 1041–1052 页，纽约州纽约，2017 年。计算机协会。国际标准书号 9781450341974。数字对象标识符：10.1145/3035918.3056101。网址https://doi.org/10.1145/3035918.3056101。

Bailin Wang、Richard Shin、Xiaodong Liu、Oleksandr Polozov 和 Matthew Richardson。RAT-SQL：面向文本到 SQL 解析器的关系感知模式编码与链接。载于 Dan Jurafsky、Joyce Chai、Natalie Schluter 和 Joel Tetreault 编，第*58*届计算语言学协会年会论文集，第 7567–7578 页，线上，2020年7月。计算语言学协会。数字对象标识符：10.18653/ v1/2020.acl-main.677。网址https://aclanthology.org/2020.acl-main.677/。

Bing Wang、ChangyuRen、Jian Yang、Xinnian Liang、Jiaqi Bai、Qian-WenZhang、Zhao Yan 和 Zhoujun Li。Mac-sql：用于文本到 SQL 的多智能体协作。*arXiv* 预印本 *arXiv:2312.11242*，2023a。

Lei Wang、Wanyu Xu、Yihuai Lan、Zhiqiang Hu、Yunshi Lan、Roy Ka-Wei Lee 和 Ee-Peng Lim。《计划并求解提示：通过大语言模型改进零样本思维链推理》。载于第*61*届计算语言学协会年会论文集（第*1*卷：长篇论文），第2609–2634页，2023b。

Xingyao Wang、Boxuan Li、Yufan Song、Frank F. Xu、Xiangru Tang、Mingchen Zhuge、Jiayi Pan、Yueqi Song、Bowen Li、Jaskirat Singh、Hoang H. Tran、Fuqiang Li、Ren Ma、Mingzhang Zheng、Bill Qian、Yanjun Shao、Niklas Muennighoff、Yizhe Zhang、Binyuan Hui、Junyang Lin、Robert Brennan、Hao Peng、Heng Ji 和 Graham Neubig。《OpenDevin：面向 AI 软件开发者的通用型智能体开放平台》，2024年。网址 https://arxiv.org/abs/2407.16741。

Qingyun Wu、Gagan Bansal、Jieyu Zhang、Yiran Wu、Shaokun Zhang、Erkang (Eric) Zhu、Beibin Li、Li Jiang、Xiaoyun Zhang 和 Chi Wang。《Autogen：通过多智能体对话实现下一代大语言模型应用》。技术报告 MSR-TR-2023-33，微软，2023年8月。

Chunqiu Steven Xia、Yinlin Deng、Soren Dunn 和 Lingming Zhang。《Agentless：揭秘基于大语言模型的软件工程代理》。*arXiv* 预印本 *arXiv:2407.01489*，2024年。

John Yang、Carlos E. Jimenez、Alexander Wettig、Kilian Lieret、Shunyu Yao、Karthik Narasimhan 和 Ofir Press。SWE-agent：代理-计算机接口助力自动化软件工程，2024年。网址https://arxiv.org/abs/2405.15793。

Shunyu Yao、Jeffrey Zhao、Dian Yu、Nan Du、Izhak Shafran、Karthik Narasimhan 和 Yuan Cao。ReAct：在语言模型中协同推理与行动，2023年。网址 https://arxiv.org/abs/2210.03629。

Semih Yavuz、Izzeddin Gur、Yu Su 和 Xifeng Yan。如何在 WikiSQL 上实现 100% 的条件准确率。编者 Ellen Riloff、David Chiang、Julia Hockenmaier 和 Jun'ichi Tsujii，*2018*年自然语言处理实证方法会议论文集，第 1702–1711 页，比利时布鲁塞尔，2018 年 10 月至 11 月。计算语言学协会。数字对象标识符：10.18653/v1/D18-1197。网址 https://aclanthology.org/D18-1197/。

Tao Yu、Rui Zhang、Kai Yang、Michihiro Yasunaga、Dongxu Wang、Zifan Li、James Ma、Irene Li、Qingning Yao、Shanelle Roman、Zilin Zhang 和 Dragomir Radev。《Spider：用于复杂跨领域语义解析和文本到 SQL 任务的大规模人工标注数据集》。载于 Ellen Riloff、David Chiang、Julia Hockenmaier 和 Jun'ichi Tsujii 编，《*2018* 年自然语言处理实证方法会议论文集》，第 3911–3921 页，比利时布鲁塞尔，2018 年 10 月至 11 月。计算语言学协会。数字对象标识符：10.18653/v1/D18-1425。网址：https://aclanthology.org/D18-1425/。

Hanchong Zhang、Ruisheng Cao、Lu Chen、Hongshen Xu 和 Kai Yu。《Act-SQL：通过自动生成的思维链进行文本到 SQL 的上下文学习》。载于《计算语言学协会论文集：*EMNLP 2023*》，第 3501–3532 页，2023年。

Shun Zhang、Zhenfang Chen、Yikang Shen、Mingyu Ding、Joshua B Tenenbaum 和 Chuang Gan。利用大语言模型进行代码生成规划。载于第十一届国际学习表征会议，2022a。

Yiming Zhang、Shi Feng 和 Chenhao Tan。用于上下文学习的主动示例选择。载于 Yoav Goldberg、Zornitsa Kozareva 和 Yue Zhang 编，*2022*年自然语言处理实证方法会议论文集，第 9134–9148 页，阿布扎比，阿拉伯联合酋长国，2022年12月b。计算语言学协会。数字对象标识符：10.18653/v1/2022.emnlp-main.622。网址 https://aclanthology.org/2022.emnlp-main.622/。

Yuntong Zhang、Haifeng Ruan、Zhiyu Fan 和 Abhik Roychoudhury。《Autocoderover：自主程序改进》。载于第*33*届*ACM SIGSOFT*软件测试与分析国际研讨会论文集，第1592–1604页，2024年。

附录

A 实验细节

在所有实验中，我们检索最相关的前 3 个数据库表（$k = 3$）以填充上下文，并将自我优化尝试的最大次数设为 5。为确保公平比较，基线方法和我们提出的方法均采用相同的超参数设置。

除非另有说明，所有实验均使用 GPT-4o 作为骨干大语言模型OpenAI (2024年b）。所有大语言模型的温度均设为 0.7。对于涉及本地模型推理（例如开源基线方法）的实验，我们使用了一台配备两块 NVIDIA RTX 4090 GPU 的服务器。
多智能体框架基于 LangGraph 库构建 LangChain AI (2024年）。向量嵌入由`text-embedding-3-small` 模型生成，该模型来自 OpenAI OpenAI (2024年a），随后所有向量化任务均由 Faiss 库处理 Johnson 等人 (2019年）。我们的数据建模框架完全基于 Neo4j 构建 Neo4j公司 (2024年），所有图遍历均使用 Cypher 查询语言。

制品许可与条款。 我们使用 Spider2.0 基准Lei 等人 (2025年)进行评估，该基准以 **CC BY-SA4.0** 许可证发布。我们确认，对该数据集的使用符合其用于研究的预期用途。SQLAgent 的代码及构建的知识库均依据 **MIT 许可证** 发布，并可在补充材料中获取。

B 部署阶段的实现细节

B.1 模式与查询向量化

如第 3.3 节所述，我们采用语义检索来弥合自然语言与数据库模式之间的差距。

模式嵌入。 为实现高效检索，数据库中的每一列 c都会按照以下模板序列化为描述性字符串：`"[名称]: 名称; [类型]: 类型; [描述]: 注释"`。

知识库嵌入。 对于在探索阶段构建的知识库，我们对每个三元组（S, Q, U）中的 SQL 组件（Q）进行向量化。我们使用代码专用嵌入模型（例如 text-embedding-3-small 或 uni-xcoder）来捕获 SQL 查询的结构和语义特征。这样，GenAgent 就能依据结构相似性而非仅仅依靠关键词重叠来检索示例。

B.2 智能体提示设计

SQLAgent 的性能依赖于专为 InfoAgent 和 GenAgent 设计的特定提示指令。

InfoAgent上下文扩展。 在部署阶段，InfoAgent 会执行第二步扩展，以推断隐含的依赖关系。系统会提示大语言模型充当数据库专家。例如，如果初始检索结果包含 users.user_name 和 orders.order_amount，提示会指示模型识别必要的连接键（例如 users.user_id 和 orders.customer_id），以连接这些表。该指令的简化版本如下：

"给定用户查询和检索到的模式片段，分析其中的关系。如果需要使用两个表，但不存在连接条件，请识别并添加形成有效 *SQL* 连接所必需的主键/外键。"

GenAgent 合成。GenAgent 接收一个包含以下内容的复合提示：

1. **角色定义：**“你是一名 SQL 数据分析专家。” 2. **模式上下文：** InfoAgent 提供的精炼 JSON 结构。 3. **少样本示例：** 从知识库中检索到的排名前 k 个三元组（S, Q, U）。 4. **用户查询：** 目标自然语言问题。

这种结构化提示可确保模型遵循目标数据库特定的语法和模式约束。

C 数据库结构的定义

如主要方法论部分所述，我们将关系型数据库表示为可遍历的图结构，以便进行自主探索。该模型由不同的节点和关系类型组成，用于体现数据库模式的层次结构和关系特性。本附录对各个组件及其相应属性进行了正式定义，并概述了该图的统计组成。

C.1 图模式概述

该图模型由五种不同的节点类型和五种定义其连接方式的关系类型组成。主要节点类型包括：

- **数据库**：表示整个数据库实例的根节点。
- **模式**：表示数据库中的一个模式或数据集。
- **表**：表示单个表。
- **字段**：表示表中的一列。
- **共享字段组**：我们提出的一种抽象，表示多个表共享的一组可复用字段。

这些节点之间的关系定义了图的结构完整性。表3概述了节点类型之间的有效连接。

Table 3： 定义图模式中节点之间连接的关系类型 a.

起始节点类型	关系类型	结束节点类型
数据库	HAS_模式	模式
模式	HAS_表	表
表	使用_字段_组	共享字段组
表	具有_唯一_字段	字段
共享字段组	具有_字段	字段

C.2 节点属性定义

图中的每个节点都包含一组用于存储其元数据的属性。以下表格详细列出了五种节点类型各自的属性。

表4：数据库 节点。

属性	描述	示例
<id>	节点的内部唯一标识符。	15313
name	数据库实例的名称。	OPEN_目标 –平台 –2
type	节点类型说明符。	数据库

表 5：属性：模式节点。

属性	描述	示例
<id>	节点的内部唯一标识符。	5430
数据库	父数据库的名称。	NOAA_数据
描述	模式的描述（如果有）。	-
name	模式的名称。	NOAA_重大_地震
type	节点类型说明符。	模式

表 6： 属性： 表 节点。

属性	描述	示例
<id>	节点的内部唯一标识符。	185
数据库	父数据库的名称。	COVID19_美国
ddl_摘要	表的 DDL语句摘要。	含有245列的表
全限定名称	表的全限定名称。	CENSUS_BUREAU_ACS_2...
name	表的名称。	SCHOOLDISTRICTSECONDARY...
模式	父模式的名称。	人口普查局_统计局_美国社区调查
type	节点类型说明符。	表

表 7： 共享字段组节点的属性。

属性	描述	示例
<id>	节点的内部唯一标识符。	2079
数据库	父数据库的名称。	NEW_YORK_PLUS
描述	组的自动生成描述。	字段组_3ebfe5f9...
字段_计数	此组封装的字段数量。	24
字段_哈希	已排序字段名称的 MD5 哈希，用于标识唯一组。	3ebfe5f9...
name	为组自动生成的名称。	字段组_3ebfe5f9
模式	父模式的名称。	纽约_出租车_行程_
type	节点类型说明符。	共享_字段_组

C.3 图组成统计

为直观展示规模，表 9 汇总了根据我们的实验数据集构建的图的组成情况。统计数据突显了字段的普遍性：字段占所有节点的 90% 以上。

我们模型的一个关键组件是共享字段组，旨在减少冗余。我们的分析证实了其有效性：

- **组总数：** 共有 **542** 个唯一的 SharedFieldGroup 节点。
- **平均扇出：** 平均而言，每个组关联 **10.6** 张表，表明其被频繁复用。

表 8: 字段节点的属性。

属性	描述	示例
<id>	节点的内部唯一标识符。	18
数据库	父数据库的名称。	NEW_YORK
描述	字段的描述（如有）。	-
name	列/字段的名称。	促成_因素_载体_4
节点_类型	指定该字段是否属于某个组或在表中唯一。到一张表中。	唯一_字段
样本_数据	此列中的示例数据。	-
模式	父模式的名称。	NEW_YORK
表	父表的名称（针对唯一字段）。	纽约州警察局_MV_碰撞事故
type	字段的数据类型。	TEXT

表 9: 构建的图表示的统计概览。

组件	计数	百分比
节点类型		
数据库	151	0.2%
模式	267	0.3%
表	7,848	8.7%
共享字段组	542	0.6%
字段	81,298	90.2%
节点总数	**90,106**	**100.0%**
关系类型		
HAS_模式	534	-
HAS_表	15,696	-
USES_字段 _组	11,478	-
具有_唯一_字段	104,808	-
HAS_字段	57,788	-
关系总数	**190,304**	-

- **最大扇出：** 使用最广泛的组被 **334** 张不同的表共享。

这项分析表明，冗余模式结构在大型数据库中十分普遍，并验证了我们的抽象在创建更精简、高效的表示形式以供探索方面的实用性。

C.4 结构可视化

D 识别共享字段组

识别共享字段组节点是一项关键的预处理步骤，旨在抽象并整合数据库中反复出现的模式结构。此过程确保字段组成相同的表都关联到同一个规范组节点，从而减少图表示中的冗余。该方法分为两个主要阶段：（1）为每组不同的字段生成唯一签名；（2）应用贪心算法，选出最终互不重叠的共享组集合。

阶段 1：字段组签名生成 为了统一识别具有相同字段的表，我们首先为每个表的字段结构计算一个基于内容的签名。该签名根据表中所有列的名称和数据类型生成。

该过程详见算法1：为字段集合创建规范字符串表示。每个字段都格式化为其名称和类型的字符串拼接（例如，“user_id:INTEGER”）。为确保签名不受原始列顺序影响，这些格式化后的字符串会按字母顺序排序，然后连接成一个带分隔符的字符串。最后，将 MD5 哈希函数应用于此规范字符串，以生成紧凑且唯一的 128 位签名。任何生成相同签名的两张表都被视为具有结构相同的模式。

算法 1 字段组签名生成

输入： 一个表的字段集合 $F_{table} = \{(n_1, t_1), (n_2, t_2) \ldots, (n_k, t_k)\}$，其中 n_i 是字段名称，t_i 是其数据类型。
输出： 一个唯一的签名字符串 S_{group}。

函数GenerateSignature(F_{table})
  初始化一个空列表 L_{fields}
  对于 每个字段 (n_i, t_i) 在 F_{table} **执行**
    $s_i \leftarrow$ 拼接(n_i,“:” t_i)　▷ 格式化为“名称:类型"
    追加 s_i 到 L_{fields}
  结束循环
  将 L_{fields} 按字母顺序排序
  $S_{canonical} \leftarrow$ 连接(L_{fields}，“|”)　▷ 创建规范分隔字符串
  $S_{group} \leftarrow$ MD5($S_{canonical}$)　▷ 计算 MD5 哈希
  返回 S_{group}
结束函数

阶段 2：不重叠组的贪心选择 在遍历所有表的初始阶段之后，我们获得了一组潜在的共享组；每个共享组都由其唯一签名标识，并关联着与之匹配的表列表。随后需要进行优化阶段，以生成最终的、互不重叠的共享字段组。这一点至关重要，因为复杂模式可能包含嵌套或重叠的字段结构。

我们采用算法 2中详细介绍的贪心选择算法来解决这一问题。其核心原则是优先处理最重要且影响最大的共享结构。首先，根据两个标准对所有潜在组按降序排序：主要依据其涵盖的表数量，其次依据其包含的字段数量（作为平局时的判断标准）。这种启发式方法会优先选择能够代表最广泛模式结构的组。
随后，算法遍历这个已排序的列表。对于每个候选组，它会检查其成员表中是否有任何表已经被分配到此前选定的组。如果不存在重叠，则将该组加入最终集合，并将其所有成员表标记为已分配。此过程确保每个表最多只能属于一个共享字段组，从而得到明确且高效的最终图结构。

E 详细动作空间

本节详细描述了大语言模型在探索阶段使用的离散动作空间。在树搜索的每一步中，大语言模型都会从以下动作中选择一个，根据当前查询状态和模式上下文逐步构建 SQL 查询。

- **选择未使用列：** 向查询的“SELECT”语句添加一个此前未选择的列。此操作会逐步扩展查询所检索信息的广度。

算法 2 共享字段组的贪心选择

输入： 所有潜在组 G_{all} 的集合，其中每个组 $g \in G_{all}$ 都具有一个签名和一组字段，以及一组匹配表 T_g。
输出： 最终得到的互不重叠共享组集合 G_{final}。

函数SelectGroups(G_{all})
  $G_{filtered} \leftarrow \{g \in G_{all} \mid |T_g| \geq 2\}$ ▷ 仅考虑至少包含两张表的组
  按 $G_{filtered}$ 降序排序，先依据 $|T_g|$，再依据字段数量。
  初始化 $G_{final} \leftarrow \emptyset$
  初始化已分配表集合 $T_{assigned} \leftarrow \emptyset$
  对于 每个组 g，按排序结果 $G_{filtered}$ **执行**
    $T_{current} \leftarrow$组 g 中的表集合。
    如果 $T_{current} \cap T_{assigned} = \emptyset$ **那么** ▷ 检查是否与已分配的表重叠
      $G_{final} \leftarrow G_{final} \cup \{g\}$ ▷ 将组添加到最终集合
      $T_{assigned} \leftarrow T_{assigned} \cup T_{current}$ ▷ 将表标记为已分配
    结束 if
  结束 for
  返回 G_{final}
结束函数

• **添加谓词约束：**对现有列应用过滤条件，通常通过添加“WHERE”子句实现。此操作会缩小查询范围，使其聚焦于符合特定条件的数据子集。

• **引入连接：** 根据外键关系或“共享字段组”链接，将当前表集合与新表连接起来。此操作对于探索不同表之间的关系，以及综合来自多个来源的信息至关重要。

• **应用聚合函数：** 将汇总函数（例如“COUNT”、“SUM”、“AVG”、“MAX”、“MIN”）应用于先前选定的列。此操作将查询的用途从简单的记录检索转变为数据汇总，使模型能够了解数据的规模和分布。

• **添加分组子句：**按一个或多个选定的非聚合列对结果集进行分组。此操作通常与聚合函数结合使用，以执行分类分析，例如计算数据中不同细分的指标（例如，“每个地区的总销售额”）。

• **添加排序子句：**通过“ORDER BY”子句，根据指定列按升序（‘ASC‘）或降序（‘DESC‘）对最终结果集进行排序。这有助于发现极值，例如表现最佳的项目或最近发生的事件。

• **添加HAVING子句：**对‘GROUP BY‘聚合的结果应用筛选条件。不同于在聚合前筛选行的‘WHERE‘子句，‘HAVING‘会在聚合后筛选整个组，从而支持更复杂的分析问题，例如识别达到特定阈值的类别（例如，“订单超过5笔的客户”）。

用于双智能体 SQL 合成的 F 算法

本节详细介绍我们在方法论的部署阶段所描述的 SQL 合成双智能体框架的算法实现。该流程设计为一个迭代循环，其中 **InfoAgent**与 **GenAgent**协作精炼上下文并生成最终 SQL 查询。InfoAgent 负责模式定位和上下文管理，而 GenAgent 则利用获取的知识库来合成查询。算法3 对这一协作流程进行了形式化描述。

算法 3 双智能体 SQL 合成工作流

输入： 用户话语 U_{in}、数据库模式图$\mathcal{G}$、知识库$\mathcal{K}$、最大迭代次数 N_{max}
输出： 最终 SQL 查询 Q_{final} 或失败

1: **函数** 双智能体合成(U_{in}，$\mathcal{G}$，$\mathcal{K}$ N_{max})
2: $i \leftarrow 0$
3: $is_success \leftarrow$ **错误**
4: $feedback_info \leftarrow \emptyset$ ▷ 存储失败尝试的上下文，以便进行剪枝
5: **当** $i < N$ $_{max}$ **且非** is $_{success}$ **执行**
▷ **InfoAgent**：模式定位与扩展
6: $keywords \leftarrow$ LLM.ExtractKeywords(U_{in})
7: $S_{initial} \leftarrow$ 语义搜索($\mathcal{G}$ $keywords$) ▷ 检索初始的 top-k 模式组件
8: $S_{expanded} \leftarrow$ 大语言模型。扩展上下文(U_{in}， $S_{initial}$) ▷ 推理并添加必要的相关组件
9: $S_{ctx} \leftarrow S_{expanded} \setminus$ 未使用项剪枝($feedback_info$) ▷ 根据上次失败优化上下文
▷ **GenAgent**：知识检索与 SQL 合成
10: $Q_{embed} \leftarrow$ 嵌入查询意图(U_{in}， S_{ctx})
11: $\mathcal{K}_{retrieved} \leftarrow$ 相似性搜索($\mathcal{K}$ Q_{embed}) ▷ 检索排名前 k 的三元组
12: $Q_{cand} \leftarrow$ 大语言模型.GenerateSQL(U_{in}, S_{ctx},$\mathcal{K}_{retrieved}$) ▷ 生成候选查询
▷ 执行与验证反馈循环
13: $result$， $error \leftarrow$ 执行SQL($\mathcal{D}$ Q_{cand})
14: **如果** $error \neq \emptyset$ **或** $result$ 为空**，则**
15: $feedback_info \leftarrow$ (Q_{cand}， S_{ctx}，“执行失败”) ▷ 为上下文整理反馈剪枝
16: **else**
17: $is_aligned \leftarrow$ 大语言模型.忠实度检查(U_{in}， $result$) ▷语义忠实度检查
18: **如果** $is_aligned$**那么**
19: $Q_{final} \leftarrow Q_{cand}$
20: $is_success \leftarrow$ **真**
21: **else**
22: $feedback_info \leftarrow$ (Q_{cand}， S_{ctx}， “语义不匹配")
23: **结束 if**
24: **结束 if**
25: $i \leftarrow i + 1$
26: **结束 while**
27: **如果** $_{is}$ $success$ **则**
28: **返回** Q_{final}
29: **else**
30: **返回** 失败
31: **结束 if**
32: **结束函数**

算法细节 该算法详细说明了双智能体框架管理的上下文精炼和查询生成迭代流程。流程从**初始化**阶段开始（第 1–4 行），此阶段会准备迭代计数器、成功标志以及用于存放反馈信息的空结构。feedback_info 变量对于将失败尝试中获得的见解传递至下一轮迭代至关重要。

主循环中的每个周期都以 **InfoAgent** 执行模式定位和扩展（第 6–9 行）开始。它首先使用大语言模型从用户话语中提取语义关键词，并执行语义搜索以检索一组初始模式组件。由于这组组件可能不完整，第二个由大语言模型驱动的步骤会通过推理逻辑必需项（例如连接键）来扩展它。随后会执行一个关键的 **上下文剪枝** 步骤（第 9 行）：InfoAgent 利用此前任何失败迭代的反馈，移除未使用的模式组件，从而缩小搜索空间。

随后，经过精炼的上下文会传递给**GenAgent**，后者利用用户意图从知识库 $\mathcal{K}$中查找最相关的三元组，执行**知识检索** （第 11–12 行）。GenAgent 将这些三元组作为上下文示例，继续进行**SQL 合成** （第 13 行），生成一个候选查询。随后，该查询会经历**执行与验证** （第 15–24 行）。此步骤有三种可能的结果。如果发生**执行失败** （例如语法错误）或**语义不匹配** （即查询输出与用户意图不一致），系统会将失败反馈存储到 feedback_info 中，供下一轮循环使用。当查询执行正确并通过保真度检查时，就会**成功**，此时系统设置成功标志并终止循环。整个过程会持续进行，直到找到有效查询或达到由**终止** 条件（第 27–31 行）规定的最大迭代次数，之后返回最终查询或失败信号。

G 大语言模型的使用

根据 ICLR 的政策，本声明旨在披露我们在准备本稿件时对大语言模型（LLM）的使用情况。大语言模型的使用严格限于改进文本的语言表达和可读性，包括语法纠正和风格润色。大语言模型未参与本研究的任何核心科学工作，例如研究构思、实验设计或数据分析。所有智力贡献均为作者原创，作者对最终内容承担全部责任。

H 与推理时框架的扩展比较

为进一步阐明 SQLAgent 相对于近期最先进的文本到 SQL 框架的定位，我们在本节中进行了详细的概念比较。

H.1 主动知识构建与推理时提示

诸如**Din-SQL**、**C3**和**Chess** 之类的框架，主要致力于优化推理时 的推理过程。它们假设，只要提示或分解策略足够复杂，冻结的大语言模型就能将自然语言正确映射到未见过的数据库模式。

相比之下，SQLAgent 的前提是：对于复杂的企业级数据库，主要瓶颈在于智能体不熟悉独特的模式语义和复杂的连接路径。我们的框架将知识获取与查询生成解耦：

- **离线探索阶段**：SQLAgent 自主与实时数据库交互，构建由三元组组成的可执行知识库（S， Q, U）。在收到任何用户查询之前，该阶段就像一位“自动化数据工程师”，学习数据库的特定逻辑。
- **部署阶段**：智能体将这些经过验证且针对特定数据库的示例作为上下文示例提示进行检索。这有效解决了“冷启动”问题，因为仅依赖推理的方法往往会因含糊的列名或未记录的关系而难以应对。

H.2 摊销成本分析

尽管探索阶段会带来初始化开销，但这项成本会在数据库应用的整个生命周期内**逐步摊销**。在企业环境中，数据库模式相对稳定，而用户查询则频繁且重复。通过一次性投入进行主动探索，SQLAgent 能为后续所有交互提供更稳健、更准确的接口，因此，对于持久化数据平台而言，探索成本是一项值得的权衡。
本节将进行定性分析，展示在探索阶段获取的数据库特定知识如何促进复杂 SQL 合成，随后详细剖析实验中观察到的当前失败模式。

I 案例研究与错误分析

本节将进行定性分析，展示在探索阶段获取的数据库特定知识如何促进复杂 SQL 合成，随后以文字详细剖析实验中观察到的当前失败模式。

I.1 案例研究：解析嵌套结构与分片关系

SQLAgent 的主要优势在于能够主动发现复杂的连接路径和未记录的模式语义，而这些往往是零样本模型无法察觉的。我们通过 GA4_OBFUSCATED_SAMPLE_ECOMMERCE 数据集中的一个典型查询来说明这一点。该数据集具有按时间分片的表和嵌套 VARIANT 字段。

用户问题：“我想了解在 2020 年 12 月购买了‘Google 海军蓝斑点 T 恤’的顾客的偏好。与这件商品一起购买的其他商品中，哪一件的总购买数量最高？”

挑战：数据库将每日事件存储在单独的分片表中（例如 EVENTS_202012*）。SQLAgent 利用 **共享字段组** 将这些表抽象为一个可遍历的统一模板，从而显著降低搜索复杂度。此外，产品信息封装在名为 ITEMS 的 VARIANT 列中。要访问商品_名称，需要执行 UNNEST 操作；尽管模式的 DDL 中并未明确说明这一要求，但它对于成功执行至关重要。

检索到的探索　　三元组：在探索阶段，智能体验证了以下三元组　　：

- **模式子结构**（S）：{events_*,items (VARIANT), UNNEST(items) AS i}
- **SQL 片段**（Q）：SELECT user_pseudo_id, i.item_name FROM ga4.events_*, UNNEST(items) AS i WHERE i.item_name IS NOT NULL
- **描述**（U）：“如何从 GA4 表的嵌套 items 数组中提取各个产品名称。”

成功与失败：若没有此三元组，基线模型会尝试直接访问items.item_name，从而产生语法错误。借助该三元组，**GenAgent** 在公用表表达式（公用表表达式）中正确实现了 UNNEST 模式，并成功执行。

I.2 详细错误分析

对 Hard 类别中 100 个失败样本的人工分析（74.22% 尚未解决）表明，当前的瓶颈主要源于企业级文本到 SQL 的固有复杂性，而非对模式不熟悉。这些挑战主要有两方面：**(i) 推理约束**，极端的逻辑嵌套（例如，≥ 4 子查询）或过长的语句（> 100行）有时会超出骨干大语言模型的多步推理能力和上下文窗口；**(ii) 细粒度语义对齐**，涉及精确的值到单元格对齐（例如，日期格式化），以及在列数超过 800 的密集模式中，对功能不同但语义相似的字段进行消歧。这些发现表明，尽管探索阶段有效弥合了“模式熟悉度”方面的差距，自主数据智能体的下一个突破方向仍是提升大规模企业环境中的多步逻辑综合能力和精确值对齐能力。