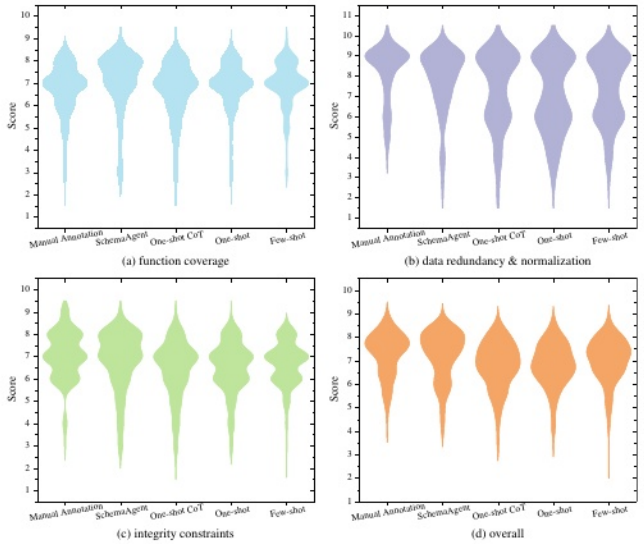

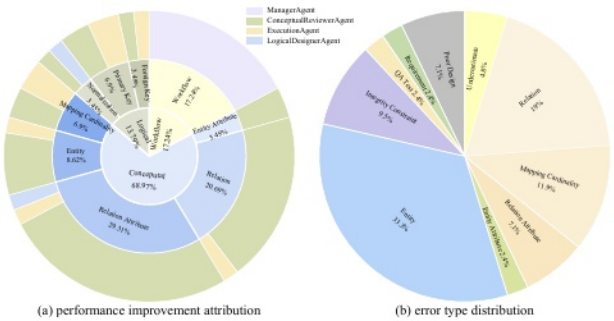

分别为 30%，这反映了用户修改的便捷性。我们的分析表明：（1）通过手动注释实现的可扩展性优化仍有提升空间，这在手动过程中有时会被忽视（例如，设置属性而非表进行扩展）。（2）总体而言，手动注释表现强劲，这验证了我们数据集的良好质量。特别是，手动注释在数据冗余和规范化方面的得分显著更高，这反映了我们对一致性和消除冗余的设计偏好。（3）在所有基准测试中，SchemaAgent 在所有指标上均取得最佳结果。其在数据冗余和规范化方面的出色表现进一步证实了我们方法在范式控制方面的有效性。（4）存在一些来自 LLM 的模式得分高于手动注释的模式，但它们所占比例非常小，我们预计未来会引入可扩展性优化来解决这一问题。

6.6 性能提升与错误分析

为了评估 SchemaAgent 如何提升性能，我们分析了 31 个高分案例（基于自动化指标和 LLM 作为评判标准）。性能提升主要分为三个领域：工作流（无错误反馈）、概念模型和逻辑模型。概念模型的改进包括对实体、关系和映射基数的增强识别。逻辑模型的改进涵盖主键/外键和规范化优化。对于每个类别，我们追溯改进到提供反馈的具体代理。图 10(a) 显示 68.9% 的改进来自概念模型的改进，其中 CMR 代理提供了 85% 的反馈，而 LMD 和 TE 代理提供了其余部分。在 13.79% 的案例中，概念模型正确的情况下，LMD、TE 和 CMR（以及 CMD）协同优化了

为了进一步评估 SchemaAgent，我们还分析了 31 个综合得分较低的案例，识别出十种错误类型。如图 10(b) 所示，作为数据库设计关键阶段的概念设计仍有很大的改进空间。（1）实体错误包括过度识别实体（例如，在用户表中已包含收件人的情况下仍单独创建收件人表）和实体识别不足，尤其是当实体描述简短时。（2）关系错误涉及实体间关系的缺失。（3）关系属性错误主要表现为在关系中添加冗余的自动递增 ID 属性。（4）映射基数错误常常将多对多关系（例如供应商与产品之间的关系）误判为一对多关系，尤其是在需求描述模糊的情况下，这给模型理解现实场景带来了挑战。（5）由于自动评估的固有局限性，在 4.8% 的案例中出现了低估的情况。（6）7.1% 的功能完备的模式缺乏可扩展性，导致设计不佳。此外，我们注意到该模型在处理关联实体（通过外键与其它实体相连接且具有独立主键的实体）和弱实体（其存在依赖于另一个实体的实体）时更容易出错。这表明准确识别和表示复杂的实体关系，尤其是那些涉及非平凡主键结构和存在性依赖的关系，需要更高级的上下文理解和外部知识。

6.7 模式生成案例研究

为了展示我们所提出的 SchemaA-gent 框架的实际有效性，我们以仓库管理领域的一个案例研究为例。如图 11 所示，在 CoT 和少样本提示设置下生成的模式直接将“货物名称”包含在“任务”表中，这会带来数据不一致的风险（例如“货物名称”中的拼写错误）。而 SchemaAgent 的输出遵循 3NF 的原则。在“任务”表中，SchemaAgent 正确识别了函数依赖关系。

表 3：与 Text2SQL 模型的比较。“可执行”表示使用 SQLite 以 DDL 格式执行。

方法	模式		属性		主键	外键	数据类型	可执行文件
	F1	据.....	F1	据.....	据.....	据.....	据.....	据.....
北极文本到 SQL 第一版 70 亿参数模型	4.54	42.26	59.32	33.33	21.00	38.06	61.68	91.86
奥米尼SQL-14B	84.25	56.17	69.55	40.16	63.78	69.55	61.61	95.28
模式代理	90.29	65.09	79.53	49.87	73.23	81.63	83.76	97.14

属性之间的依赖关系，并据此确定一个组合主键。这种规范化结构和组合键的设计显著增强了数据的完整性。

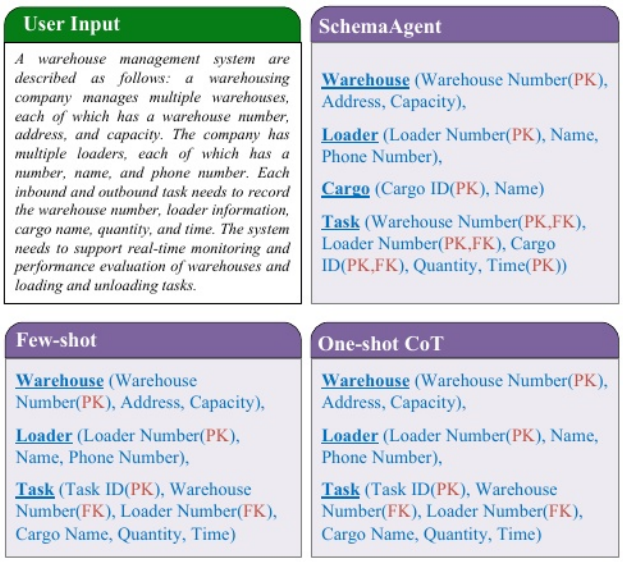

图 11：在仓库管理案例中，SchemaAgent、少样本学习和带 CoT 的单样本学习的模式输出。

我们的方法在生成规范化模式方面表现出色。未来的工作可以整合用户查询模式，以促进策略性反规范化，从而在查询效率和数据一致性之间实现平衡。

6.8 DDL生成案例研究

我们证明了将模式转换为数据定义语言（DDL）是物理设计的一个基本部分，且这一过程十分简单。不同的数据库管理系统（DBMS）采用不同的方法来实现物理设计。为了验证模式的有效性，我们采用基于规则的方法生成特定于 SQLite 的 DDL，然后执行以经验性地验证我们的方法。这种处理方式类似于 Text2SQL 任务。因此，我们选择了几项突出的 Text2SQL 工作进行评估，包括 Arctic-Text2SQL-R1 [51]：一种基于强化学习的 Text2SQL 模型，通过执行反馈来优化生成可执行的 SQL 查询。OmniSQL [21]：一个在大型合成数据集上训练的 Text2SQL 模型。

如表 3 所示，现有的 Text2SQL 模型在关系模式识别方面尤其困难。这凸显了开展专门的 Text2DDL 研究至关重要，因为这涉及推断复杂的结构设计和关系，而现有的 Text2SQL 方法无法应对这些情况。

7 结论与未来工作

数据库模式的自动化仍然是一个重大挑战，这主要是由于其固有的对专业知识和大量积累的实践经验的依赖。在本文中，我们首次提出了一种基于大型语言模型的多智能体框架，用于逻辑模式生成。我们将此框架命名为 SchemaAgent，在其中设计了六个角色，并在工作流程中引入了可控的交互机制以实现传输错误检测与恢复。此外，我们构建了 RSchema，这是一个跨领域数据集，包含超过 381 对用户需求文本及其对应的逻辑模式。我们系统地评估了主流大型语言模型和 SchemaAgent 在 RSchema 上的性能。实验结果表明，SchemaAgent 在所有指标上均达到了最先进的水平，并具有很强的竞争力。

未来工作。本研究是首次尝试利用大型语言模型实现自动化模式生成，这可能会为未来的研究开辟几个有前景的方向，例如：

- 物理设计集成。将物理设计纳入我们的框架会是一项有趣的未来工作，这能使我们的工作更加全面。物理设计可能涉及索引生成、磁盘分配等。
- 函数依赖发现。识别函数依赖是规范化的一个前提条件。传统的函数依赖发现工作基于对数据集的统计分析。然而，借助于大型语言模型丰富的世界知识，目前可以直接从自然语言描述中发现函数依赖。探索基于大型语言模型的函数依赖发现为更智能且无需数据的模式规范化开辟了一条新的研究途径。
- 查询效率与规范化平衡。在本文中，我们优先考虑数据一致性，要求生成的模式满足第三范式。然而，在实际应用中，严格的规范化可能会导致查询性能不佳。一个有前景的方向是将用户的查询需求纳入模式生成过程，以实现数据一致性和查询效率之间的最佳平衡。