arXiv:2603.07278v1 [cs.DB] 7 Mar 2026

LLM-FK：用于大规模复杂数据库外键检测的多智能体大语言模型推理

Zijian Tang[1], Ying Zhang[1]*, Sibo Cai[2], Ruoxuan Wang[1]

[1]北京大学， [2]国家开放大学 {zjtang25, wangrx24}@stu.pku.edu.cn, zhang.ying@pku.edu.cn caisb@ouchn.edu.cn

摘要

检测缺失的外键需要准确建模跨数据库模式的语义依赖关系，而传统的基于启发式的方法在捕捉这些关系方面存在根本局限。我们提出**LLM-FK**，这是首个用于外键检测的全自动多智能体框架，旨在应对阻碍朴素的基于大语言模型的解决方案应用于大规模复杂数据库的三项核心挑战：组合式搜索空间爆炸、有限上下文下的模糊推断，以及由孤立局部预测引发的全局不一致性。LLM-FK协调四个专门化智能体：分析器将外键检测问题分解为验证外键候选列对的任务，并通过唯一键驱动的模式分解策略剪枝搜索空间；解释器注入自增强的领域知识；优化器构建紧凑的结构表示并执行多视角思维链推理；验证器则通过整体冲突解决策略强制实现模式级一致性。在五个基准数据集上的实验表明，LLM-FK的F1分数持续高于93%，在大规模MusicBrainz数据库上超越现有基线方法15%，同时将候选搜索空间缩小两到三个数量级且不丢失真实外键，并在缺失数据等挑战性条件下保持鲁棒性。这些结果证明了LLM-FK在真实世界数据库中的有效性与可扩展性。

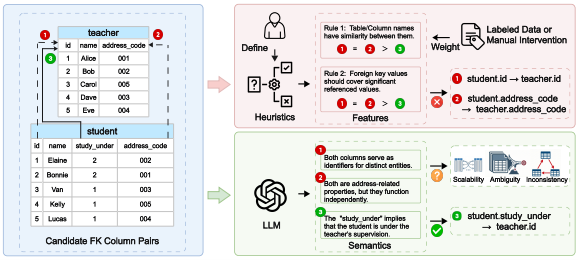

图 1：外键检测对比：启发式方法的局限性与大语言模型语义推理。

1 引言

外键（FK）约束定义了一个表中的引用列（或列集）与另一个表中的被引用列之间的有向依赖关系，从而在关系数据库中确保参照完整性（Melton 和 Simon，2001年）。这些约束在支持以下任务方面发挥着基础作用：包括模式逆向工程、数据集成和语义查询在内的关键下游任务（Wu 等人，2019；Li 等人，2023年）。然而，在真实世界数据库中，由于模式演化、遗留系统迁移，或出于性能考虑而有意省略，外键约束经常缺失，这可能会损害数据可靠性和语义一致性（Johnson 等人，2003年）。虽然对于小型数据库模式，手动识别外键或许可行，但对于具有大量异构表和复杂数据分布的大型数据库，这种做法很快便会变得不切实际（Halevy 等人，2006年；Hai 等人，2023年）。

现有的自动外键检测方法主要依赖启发式方法，利用语法或统计信号，例如列名相似度和值包含率（Rostin 等人，2009年；Chen 等人，2014年；Jiang 和 Naumann，2020年）。尽管这些启发式方法在结构良好或范围较窄的数据库模式中效果显著，却很难捕捉构成外键基础的深层语义依赖（Lausen，2007年；Kent，2012年）。因此，其性能极易受到稀疏数据和非常规命名模式等异常情况的影响，且要保持可靠性，就需要人工调优或高质量的标注数据（Zhang 等人，2010年），这显著限制了它们在大型、异构或不断演变的数据库中的可扩展性和泛化能力。

* 通讯作者。

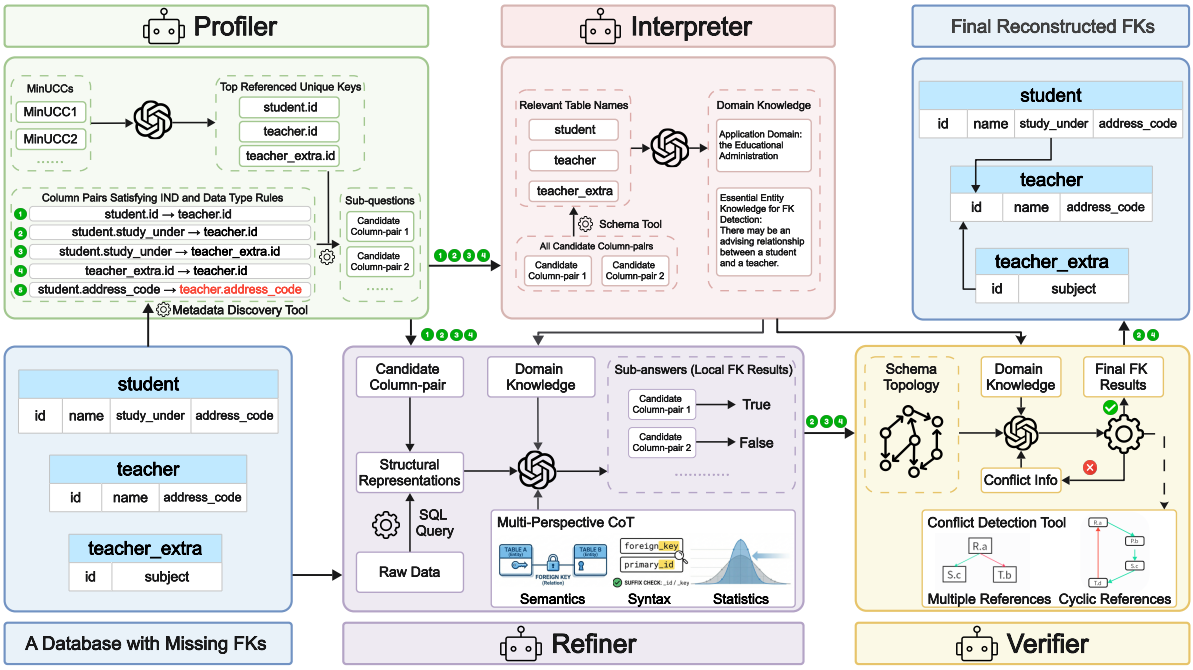

图 2：LLM-FK 框架概览。该流水线协调四个专门化智能体，实现全自动外键检测：分析器执行唯一键驱动的模式分解策略；解释器执行自增强领域知识注入；优化器采用多视角思维链推理；验证器则应用整体冲突解决策略。

大语言模型（LLM）提供了一种颇具前景的替代方案，它能够隐式捕捉超越表层启发式方法的语义关系；然而，将大语言模型直接用于大规模外键检测，会带来若干根本性挑战（见图1）。首先，**组合搜索空间爆炸**严重限制了可扩展性：随着表和列的数量增加，候选列对的数量会迅速增长（例如，仅八张表就约有 4,000 对（PoessandFloyd，2000年）），使穷举评估在计算上难以实现。其次，**有限上下文下的模糊推断**使准确推理更加困难：有限的上下文窗口（Ye 等人，2023年；Chen 等人，2024年）以及处理原始表格数据时的性能下降（Cheng 等人，2022；Chen，2023年；Sui 等人，2024b）使大语言模型无法同时访问完整的模式结构和数据分布，迫使其依赖可能遗漏关键信号的压缩或不完整表示。第三，**孤立局部预测导致的全局不一致**也随之出现，因为大语言模型通常在局部上下文中独立评估外键，例如孤立地评估单个列对或表对，从而导致全局决策相互冲突，例如产生违反模式级一致性的循环引用。

为克服这些局限，我们提出**LLM-FK**，这是一个结构化的多智能体框架，可在大型复杂数据库中全自动进行外键检测，无需人工干预或标注数据。LLM-FK 将全模式外键发现分解为相互协同的推理阶段，每个阶段都明确针对上述挑战之一。为缓解**组合搜索空间爆炸**，Profiler 采用**唯一键驱动的模式分解策略**，利用大语言模型为每个可能被引用的表识别最可能被引用的唯一键，从而将全局检测任务分解为若干聚焦于精简列对候选集的验证问题。这种考虑约束的分解大幅降低了判定复杂度和搜索空间，同时保留了外键的召回率。为应对**有限上下文下的模糊推断**，Interpreter 和 Refiner 协同执行**自增强领域知识注入**与**多视角思维链（CoT）推理**。具体而言，Interpreter 从模式元数据和推断出的领域知识中提取隐含的语义线索，而 Refiner 将这些上下文与精简的结构化表示相结合，使模型即使在上下文有限的情况下，也能从句法、语义和统计视角进行稳健推理。

最后，为确保**局部预测的全局一致性**，Verifier 通过构建模式级依赖图、检测循环引用等结构违规，并进行有针对性的推理以消解不一致，来实施**整体冲突解决策略**，从而生成符合关系语义和整体模式拓扑的协调外键集合。

我们在涵盖不同领域和规模的五个公开基准数据库上进行了大量实验，以评估 LLM-FK。总体而言，LLM-FK 在有效性、效率和鲁棒性方面均表现出持续的优势。它在所有基准上的 F1 分数均超过 93%，在 MusicBrainz（Swartz，2002年）这一包含 300 多张表的复杂真实世界数据库上，性能比基于启发式和基于大语言模型的基线方法高出 15 % 以上。在效率方面，LLM-FK 将候选搜索空间缩小了两到三个数量级，同时未丢失任何真实外键。此外，即使在缺失数据和语义关系晦涩等具有挑战性的条件下，它仍保持较强的鲁棒性。这些结果彰显了 LLM-FK 在大规模复杂数据库外键检测方面的可扩展性和泛化能力。

我们的贡献总结如下：

- 我们提出了 LLM-FK，这是首个用于大规模复杂数据库全自动外键检测的多智能体框架。该框架克服了现有方法在语义方面的局限，无需人工监督或标注数据。

- 我们设计了四个专门化智能体，系统地应对将大语言模型应用于外键检测时的核心挑战：分析器通过唯一键驱动的模式分解策略缩小搜索空间；解释器和优化器通过自增强领域知识注入与多视角思维链推理，实现深入且稳健的推断；验证器则通过整体冲突解决策略确保整个模式的一致性。

- 大量实验证明，LLM-FK 在准确性方面达到最先进水平（SOTA），同时具备高效率，并且针对复杂的大规模数据库模式表现出很强的鲁棒性。

2 相关工作

2.1 基于启发式方法的外键检测方法

基于启发式的方法构成了自动化外键检测的主流研究方向，这类方法依赖从数据库模式和数据实例中提取的预定义语法与统计信号。这些方法大致可分为规则加权方法和基于机器学习的方法。规则加权方法（Zhang 等人，2010年；Chen 等人，2014年；Jiang 和 Naumann，2020年）设计一组人工构建的启发式规则，例如列名相似度和值包含率，并通过加权评分函数对这些规则进行汇总，以估计外键候选项的合理性。尽管这类方法在受限环境中有效，但需要耗费大量人力进行人工调优，以确定适当的启发式权重和决策阈值，从而限制了其在不同领域中的适应性。

基于机器学习的方法尝试通过从标注示例中学习判定函数，减少人工调优，从而自动完成候选项排序或分类（Rostin 等人，2009年；Yan 和 He，2020年）。然而，这种范式高度依赖高质量的标注训练数据，而此类数据获取成本高昂，并且对于大型真实数据库模式而言往往难以获得。更根本的是，由于外键编码的是语义关系，而非纯粹的句法和统计规律，因此僵化的启发式信号在大规模复杂数据库中必然会失效。稀疏数据、非标准命名约定以及异构模式设计等不规则情况，常常会产生违背启发式假设的反例（Zhang 等人，2010年），导致准确率下降且泛化能力有限。相比之下，我们提出的 LLM-FK 利用大语言模型的语义推理能力，直接推断外键关系，从而突破固定启发式规则的表达能力限制，实现自动且稳健的检测。

2.2 大语言模型用于数据库模式分析

近年来，大语言模型在多种数据库模式分析任务中展现出巨大潜力，包括表注释、实体匹配和连接预测（Narayan 等人，2022；Fernandez 等人，2023年；Jaimovitch-López et al.，2023）。这些研究表明，大语言模型能够有效捕捉模式元数据和自然语言描述中的语义信息。尽管取得了这些进展

，对基于大语言模型的数据库级外键检测仍缺乏系统性研究。与之密切相关的工作主要关注成对连接预测。例如，Chorus（Kayali 等人，2023年）利用 Python 代码补全提示来预测表之间的可连接性，但其主要面向临时分析性连接，而非正式的外键约束。因此，这些方法没有明确建模语义方向性、唯一性要求或参照完整性保证，而这些特征正是外键区别于一般连接的关键。

此外，MMQA 初步探索了直接提示大语言模型识别表对之间的外键，将其作为多表问答的前提（Wu 等人，2025年）；然而，这类通用的基于大语言模型的推理范式，包括少样本提示和思维链推理，难以直接应用于大规模复杂数据库（Brown 等人，2020年；Wei 等人，2022）。这些方法存在三项根本局限：枚举列对时会引发组合搜索空间爆炸；在有限上下文下推断结果模糊不清；孤立的局部预测会导致整体不一致。若缺乏结构分解或协调机制，直接应用大语言模型便难以扩展，同时又保持模式级一致性。为应对这些挑战，我们提出 LLM-FK，这是一种多智能体框架，其中每个智能体都经过专门设计，用以解决将大语言模型应用于外键检测时面临的一项局限，从而实现可扩展、一致且具有语义依据的推断。

3 LLM-FK 方法

3.1 问题定义

外键检测旨在从给定的关系数据库模式及其数据实例中恢复隐式引用约束。形式上，给定一个关系数据库 $\mathcal{D} = \{T_1, \ldots, T_n\}$，每个表 T_i 定义为一个元组$(\mathcal{C}_i,\ \mathcal{R}_i)$，其中 $\mathcal{C}_i$ 表示列（模式）的集合，而 $\mathcal{R}_i$ 表示有限的数据元组（实例）集合。外键检测的目标是识别一组基本引用$\Phi = \{(c_f,\ c_p) \mid c_f \in \mathcal{C}_i,\ c_p \in \mathcal{C}_j\}$其中每一对$(c_f,\ c_p)$ 代表一种单列外键关系，其中引用列c_f 指向被引用列 c_p。该表述可推广至复合外键（Harrington，2016年），复合外键被建模为同一源表与目标表之间、位于不同列上的多个基本引用的组合。通过将外键检测简化为对基本列级引用的识别，该定义实现了细粒度推理，同时仍与高阶关系约束兼容。

3.2 预备知识：UCC 和包含依赖

唯一列组合（UCC）和最小 UCC（MinUCC）。UCC 刻画了表中的类键列集。具体而言，对于表 T_i，列的子集 $X \subseteq \mathcal{C}_i$ 构成*UCC*，当且仅当将 T_i 投影到 X 后能够唯一标识每个元组。形式上，对于任意两个不同的元组 $t_u,\ t_v \in \mathcal{R}_i$，其投影值必须满足 $t_u[X] \neq t_v[X]$。*MinUCC* 进一步要求满足最小性，即任何真子集 $X' \subset X$ 都不能保留唯一性。MinUCC 在外键检测中尤为重要，因为它们代表候选键，因此是外键引用唯一有效的目标。

包含依赖（IND）。当列集 $X \subseteq \mathcal{C}_i$ 中出现的值集合包含于 $Y \subseteq \mathcal{C}_j$ 的值集合中时，包含依赖*IND*（记作 $T_i[X] \subseteq T_j[Y]$）成立。包含依赖为引用约束提供必要的结构条件：从 X 指向 Y 的任何外键都必须满足相应的包含依赖。在本研究中，我们仅关注单列包含依赖，这与我们的基本引用形式相符，并为外键候选剪枝提供可靠依据。

3.3 LLM-FK 概述

我们提出了**LLM-FK**，这是一种多智能体框架，旨在大型复杂数据库中实现全自动外键检测。如图2所示，LLM-FK 由四个专门化智能体组成——**分析器**、**解释器**、**优化器**和**验证器**——每个智能体负责一个独立的推理阶段，并配备专门的策略。这些智能体协同工作，将全模式外键发现分解为协调一致、易于处理的子任务，从而共同克服朴素LLM推理的局限性。

分析器通过唯一键驱动的模式分解策略启动工作流，将全局检测任务分解为列对验证问题，并大幅缩小候选项空间。**解释器**随后执行自增强领域知识注入，以揭示嵌入该模式中的隐式语义信号。在这一丰富上下文的基础上，**优化器**运用多视角思维链推理开展深度推断，以评估

每个外键候选项。最后，**验证器** 汇集成对预测，并应用整体冲突解决策略以消除不一致，确保全局模式的一致性。我们将在以下各小节中详细介绍每个组件与策略，更多实现细节见附录B。

3.4 唯一键驱动分解

分析器是大语言模型外键流水线的入口，其主要目标是优化大语言模型的输入，以提升可扩展性和可靠性。它明确应对两项根本挑战：（1）由于大语言模型上下文窗口有限且存在幻觉风险，直接进行全模式外键检测并不可行；（2）随着模式规模和复杂度增加，候选外键会发生组合爆炸。为此，我们提出一种**唯一键驱动的模式分解策略**，系统地将全局外键检测问题转化为一组可处理的列级验证任务，同时积极剪除结构上无效的候选项。

模式分解。 我们将外键检测分解为针对有向列对的独立二元分类问题。具体而言，每个候选项（c_f，c_p）都单独进行评估，由大语言模型判定 c_f 是否在函数上引用 c_p。与表对推理等更粗粒度的分解方式相比，列级分解能够将外键语义的原子单元独立出来。这种设计使子问题结构不受整体模式规模影响，让大语言模型一次专注于一个有向假设，并显著降低判定复杂度和幻觉风险。

候选剪枝。 尽管原子分解简化了各项判定，但它会朴素地引发列对数量上的组合爆炸。为确保计算可行性，我们采用剪枝策略来积极精炼候选集。首先，我们应用基于包含依赖的验证和基于规则的过滤器：包含依赖强制执行必要的值包含约束，而基于规则的检查则剔除数据类型不兼容或语义上不适宜的类型（例如布尔列或浮点列）所构成的列对。其次，我们利用外键必须引用唯一键这一结构性质。我们并不依赖主键（PKs）——它们常常缺失、不完整或过于复合——而是识别最小唯一列组合作为候选被引用键。对于每个可能被引用的表，大语言模型从其最小唯一列组合集合中推断最合理的被引用唯一键。如图 2所示，我们随后丢弃任何被引用列不属于所选唯一键的候选列对。该过程得到一个紧凑而全面的候选集，在保留真实外键召回率的同时消除结构上无效的外键假设。

3.5 知识增强推理

为在大语言模型上下文窗口有限的约束下准确评估经过剪枝的候选对，LLM-FK 协调两个智能体——解释器和优化器——进行结构化的知识增强推理。解释器专注于自增强领域知识注入，以消除语义歧义；优化器则采用多视角思维链推理，开展深入的基于证据的推断。

自增强领域知识注入。解释器通过分析候选集中涉及的所有表的名称，为框架赋予对数据库的全局语义理解。我们有意将分析范围限制在表名上，因为表名既能体现核心实体语义，又足够简洁，可以纳入大语言模型的上下文预算。通过推断总体应用领域和显著的实体概念，解释器弥合了抽象模式符号与真实世界语义之间的差距。至关重要的是，这一过程实现了自我增强 (Sui 等人，2024a)：将全模式领域洞见注入每项推理任务，从而减少在孤立的局部上下文中评估候选列对时产生的歧义。解释器对每个数据库仅执行一次此项分析，从而确保后续所有验证都具有一致的语义锚定，且无需重复计算。

多视角思维链推理。 基于注入的领域知识，优化器实现了一个多视角思维链框架，整合互补信号以进行稳健推理。为此，优化器发出 SQL 查询，以提取并序列化相关信息的紧凑、高密度表示，包括：(1) 模式定义（表名和列名）；(2) 统计元数据（序号位置、数据类型、取值范围、基数、行数和平均文本长度）；(3) 列间依赖关系（取值覆盖率和表大小比率）；以及 (4)

样本数据实例（前 5 行）。这种简洁的序列化方式避免了全表输入所带来的噪声和成本，同时保留了足够且具有判别力的证据，以防止原始数据处理中的误解，例如对数据类型的歧义理解。随后，优化器从三个互补视角进行推理：基于命名约定的句法视角、以分布规律为基础的统计视角，以及聚焦于实体关系的语义视角。通过模拟人类专家的推理过程，这种多视角处理能够实现跨视角补偿，使可靠信号在其他信号受损时纠正判断，例如在混淆模式或数据稀疏表中。

3.6 整体冲突消解

为确保全模式结构一致性，验证器采用一种整体冲突消解策略，对聚合后的成对预测进行处理。它构建一个编码推断出的外键关系的模式拓扑图，随后识别由局部判定引发的结构违规。具体而言，我们关注两种常见的冲突类型：多重引用，即单个列引用多个不同目标；以及循环引用，即外键关系形成闭环。

冲突消解分为两个阶段。首先，对于多重引用冲突，验证器使用第 3.5 节所述的同一知识增强推理框架重新评估竞争候选项，仅保留语义和结构上最合理的外键。其次，对于循环引用，验证器采用迭代消除策略，优先处理最短环，因为它们代表了对关系语义最直接的违反。在每个环中，移除最不合理的引用，并重复这一过程，直至图收敛为无环且不存在歧义的状态。至关重要的是，优化器基于语义的推理通常会将候选项约束在紧密聚集、语义相关的子图中。利用这种局部性，我们的方法具有两项主要优势。首先，不同于依赖孤立置信度分数的方法，在这些特定结构中对冲突候选项进行整体评估，有助于直接进行语义比较，从而准确捕捉内在的语义层级关系。其次，这显著降低了计算复杂度；将冲突消解限制在这些高度局部化的直接上下文中（例如，长度为 2 的循环引用），我们的方法从根本上避免了通常与全局图遍历相关联的级联复杂性。通过这一流程，验证器确保最终外键集合在全局范围内符合关系数据库原则。

4 实验

4.1 实验设置

数据集与评估指标。 我们在五个广泛采用的关系数据库基准上评估 LLM-FK：TPC-H (Poess andFloyd,2000年), Northwind(**NWind**) (微软,2023b),TPC-E (Chen 等人，2011), AdventureWorks(**AdvWorks**) (微软,2023a), 和 MusicBrainz (**MusicBz**) (Swartz,2002年)。这些数据集涵盖多种应用领域，包括商业、零售、金融、制造业和娱乐，并且在模式规模、命名约定和数据特征方面存在显著差异。总的来说，它们涵盖了外键检测面临的广泛现实挑战，从整洁的基准模式到大规模异构数据库。详细的数据集统计信息见附录 C。

为评估外键检测的质量，我们采用标准的精确率（P）、召回率（R）和 F1 分数（F）（Goutte 和 Gaussier，2005年）。所有指标均基于分析器生成的候选集计算，该候选集定义了后续推理的有效搜索空间。此评估协议将外键推断的质量与早期剪枝过程中剔除的平凡候选项区分开来。

基线方法。 我们将 LLM-FK 与涵盖三种方法范式的七种代表性基线方法进行比较。*(1)* 规则加权方法。MC-FK（Zhang 等人，2010年）完全依赖数据统计，并采用均匀引用假设。Fast-FK（Chen 等人，2014年）结合语法相似性与基数信号。HOPF（Jiang 和 Naumann，2020年）进一步利用主键约束来提升检测性能。*(2)* 机器学习方法。ML-FK（Rostin 等人，2009年）基于基本的语法和统计特征训练分类器。AutoSuggest（Yan 和 He，2020年）通过纳入包括序数位置在内的额外元数据，扩展了这一方法。*(3)* 基于大语言模型的方法。 Chorus（Kayali 等人，2023年）通过 Python 代码补全提示预测可连接性。端到端方法（Wu 等人，2025年）直接查询大语言模型，在单次处理中识别每个表对的外键。少样本方法（Brown 等人，2020年）

类别	方法	TPC-H			Northwind			TPC-E			AdventureWorks			MusicBrainz		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
规则加权	MC-FK	0.56	**1.00**	0.72	0.71	0.91	0.80	0.32	0.78	0.46	0.46	0.59	0.52	0.51	0.81	0.62
	Fast-FK	0.50	0.78	0.61	0.67	0.91	0.77	0.82	0.91	0.86	0.54	0.84	0.65	0.62	0.81	0.70
	HOPF	**1.00**	0.10	0.20	0.67	0.19	0.29	0.78	0.16	0.26	0.57	0.45	0.46	0.75	0.86	0.80
基于机器学习	ML-FK	0.67	0.67	0.67	0.00	0.00	0.00	0.79	0.58	0.67	0.77	0.22	0.34	0.35	0.49	0.41
	AutoSuggest	0.50	0.22	0.31	0.48	0.91	0.62	0.45	0.64	0.53	0.24	0.54	0.33	0.30	0.23	0.26
基于大语言模型	Chorus	0.50	0.44	0.47	0.46	0.55	0.50	0.32	0.47	0.38	0.13	0.29	0.18	0.26	0.32	0.29
	端到端	0.82	**1.00**	0.90	**1.00**	**1.00**	**1.00**	0.57	0.87	0.69	0.72	0.75	0.73	0.70	0.81	0.75
	少样本	0.80	0.89	0.84	**1.00**	0.91	0.95	0.46	0.71	0.56	0.63	0.68	0.66	0.70	0.70	0.70
	CoT	0.82	**1.00**	0.90	**1.00**	**1.00**	**1.00**	0.60	0.87	0.71	0.71	0.74	0.72	0.68	0.81	0.74
Ours	**LLM-FK**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.91**	**0.96**	**0.93**	**0.93**	**0.96**	**0.94**	**0.91**	**1.00**	**0.95**

表 1：外键 键检测在五个数据集上的性能比较。**粗体**表示最佳性能，d
带下划线的表示次佳性能。

在此设置中加入明确的外键示例。思维链（Wei 等人，2022）采用思维链提示，引导模型进行逐步推理。这些基线方法共同代表了当前启发式方法和大语言模型驱动范式的最先进水平。

实现细节。所有基于大语言模型的方法均使用 **DeepSeek-R1**（Guo 等人，2025年）实现，温度设为 0，以确保输出具有确定性和可复现性。为防止数据泄漏（Balloccu 等人，2024年），推断期间会屏蔽数据库名称。此外，为确保与数据驱动基线进行公平比较，分析器在基于规则的剪枝过程中会排除空列。更多实现细节及可复现性方面的考量见附录 D。补充实验——包括对其他大语言模型的评估、详细的计算成本分析及其他额外结果——以及定性案例研究，详见附录E。

4.2 主要结果

表1 总结了外键检测的整体表现。结果揭示了几项关键发现：**首先**，LLM-FK 在全部五个数据集上均始终达到最先进水平，在复杂模式上大幅领先于表现最强的基线方法（例如，在 Adventure Works 上的 F1 分数为 +21%，在 MusicBrainz 上为 +15%）。**其次**，与基于启发式的方法相比，LLM-FK 展现出明显更高的准确性和泛化能力。表现最强的基于启发式的基线方法 Fast-FK，其 F1 分数介于 0.61 至 0.86 之间，在不同数据集上的表现差异很大；而 LLM-FK 则始终保持在 0.93 至 1.00 之间的稳定表现。这一提升源于 LLM-FK 能够推理外键设计的核心要素——语义意图，而非仅仅依赖容易失效、且易受不规则情况影响的句法或统计信号。这一点是外键设计的基础——而非仅仅依赖容易失效、且易受不规则情况影响的句法或统计信号。**第**三，尽管通用的基于大语言模型的基线方法在 AdventureWorks 和 MusicBrainz 等复杂模式上的表现急剧下降，LLM-FK 仍展现出更强的稳定性。具体而言，我们的多智能体协作框架将外键检测分解为不变的二元有向列对验证任务，以降低判定复杂度；即使上下文有限，也能进行深入推理，并确保整个模式范围内的一致性，从而有效克服原生大语言模型范式在大规模复杂数据库上面临的挑战。此外，Chorus 表现不佳，证实仅预测连接无法捕捉外键固有的语义方向性和完整性约束。**最后**，LLM-FK 在语义不透明的情况下也表现出很强的鲁棒性，TPC-E 就是一个例子，其中包含许多含义模糊的英文缩写。通用的基于大语言模型的方法在这种情况下表现吃力，而 LLM-FK 得益于自增强领域知识注入和多视角思维链推理，能够结合全局和结构线索，共同消解局部列对的歧义，最终取得 0.93 的 F1 分数。总体而言，这些结果表明，LLM-FK 能够为大规模复杂数据库提供准确、可泛化且鲁棒的外键检测。

4.3 候选项剪枝分析

我们从两个角度分析剪枝策略：搜索空间缩减与剪枝安全性。图3 展示了每个剪枝阶段后剩余候选的数量，并与一种常用的表级基线进行比较，该基线由所有表的两两组合定义。

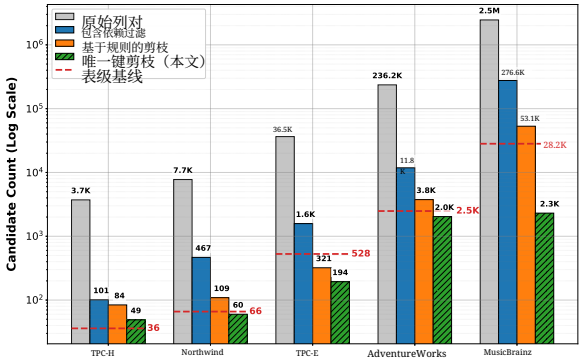

图3：与表级基线相比，经过各阶段剪枝后剩余的候选数量。

配置	TPC-H	NWind	TPC-E	AdvWorks	MusicBz
分析A：精炼器智能体					
使用注入（基线）	0.78	0.92	0.70	0.83	0.87
未使用注入	0.78	0.92	0.61	0.74	0.76
性能下降（Δ）	–	–	**-9%**	**-9%**	**-11%**
分析B：验证器智能体					
使用注入（完整）	1.00	1.00	0.93	0.94	0.95
不含注入	0.78	1.00	0.89	0.94	0.95
性能下降（Δ）	**-22%**	–	**-4%**	–	–

表2：自增强领域知识注入影响的消融实验。

搜索空间缩减。 所提出的多阶段剪枝策略实现了对外键候选空间的大幅压缩。在大规模MusicBrainz数据集上，候选对数量从超过 2.46×10^6 减少到仅2,306，降幅接近三个数量级。在大多数数据集上，所得候选集甚至小于表级枚举产生的候选集。这一缩减直接缓解了大型模式固有的组合爆炸，并大幅减少所需的大语言模型调用次数，从而确保下游推理在计算上仍然可行。

剪枝安全性。 重要的是，这种激进的剪枝并未牺牲召回率。在全部五个数据集上，被剪除的真实外键数量恰好为**零**，这证实该策略保留了 100% 的真实标注外键。我们进一步观察到，声明的主键通常是实际被引用唯一键的超集，这由索引选择或领域特定的建模决策所致。相比之下，真实外键通常引用最小唯一子集。通过以最小唯一列组合而非可能缺失或过度复合的主键为目标，LLM-FK 实现了更有效的剪枝，同时严格保持正确性。

场景	模型	TPC-H	NWind	TPC-E	AdvWorks	MusicBz
缺失 Data	不使用 MP-CoT	0.89	**1.00**	0.88	0.90	0.93
	完整 MP-CoT	**1.00**	**1.00**	**0.92**	**0.91**	**0.96**
	改进（Δ）	*+11%*	–	*+4%*	*+1%*	*+3%*
晦涩语义	不使用 MP-CoT	0.74	**1.00**	0.90	0.82	0.86
	完整 MP-CoT	**0.78**	**1.00**	**0.92**	**0.83**	**0.89**
	改进（Δ）	*+4%*	–	*+2%*	*+1%*	*+3%*
匿名模式	不使用 MP-CoT	0.31	0.83	0.56	0.28	0.38
	完整 MP-CoT	**0.67**	**0.92**	**0.71**	**0.53**	**0.76**
	改进幅度（Δ）	*+36%*	*+9%*	*+15%*	*+25%*	*+38%*

表3：多视角思维链推理在不同挑战性场景下的鲁棒性分析。

4.4 领域知识注入分析

为量化自增强领域知识注入的贡献，我们在优化器和验证器中选择性地禁用该功能。表 2 报告了各智能体输出的 F1 分数变化。

结果证实，注入领域知识对两个智能体都有益。对于优化器，移除注入的知识会导致其在 AdventureWorks 和 MusicBrainz 上的性能显著下降（分别下降 9% 和 11%）。这是因为在大型模式中，孤立的列对缺乏足够的语义上下文，导致大语言模型误解实体关系。对于验证器，领域知识在 TPC-H 等较小的数据集上尤为关键；移除后，F1 分数会下降 22%。在紧凑的模式中，全局语义线索有助于在冲突消解过程中明确区分竞争候选项。在以缩写标识符为主的 TPC-E 上，注入领域知识始终能够缓解语义歧义，避免两个智能体的性能下降。这些发现凸显了全局语义锚定在局部推理和全局一致性约束中的互补作用。

4.5 多视角思维链分析

我们评估了**多视角思维链（MP-CoT）推理**在三种具有挑战性的真实场景下的鲁棒性：**(1)缺失数据**（空表），**(2)模糊语义**（汉语拼音缩写），以及 **(3)匿名模式**（模式名称不可用）。这些场景模拟了部分或退化的数据库模式中常见的不规则情况。我们在表格中比较了完整 MP-CoT 框架与不含 MP-CoT 的变体的 F1 分数3。

结果带来了几点启示。**首先**，移除 MP-CoT 会导致所有数据集上的性能持续下降，

数据集	优化器（CR前）			验证器（CR后）			提升	
	P	R	F	P	R	F	ΔP	ΔF
TPC-H	0.64	1.00	0.78	1.00	1.00	1.00	**+36%**	+22%
NWind	0.85	1.00	0.92	1.00	1.00	1.00	**+15%**	+8%
TPC-E	0.56	0.96	0.70	0.91	0.96	0.93	**+35%**	+23%
AdvWorks	0.71	0.99	0.83	0.93	0.96	0.94	**+22%**	+11%
MusicBz	0.77	1.00	0.87	0.91	1.00	0.95	**+14%**	+8%

表4：整体冲突消解的影响消融实验。

这验证了其必要性。在匿名模式条件下，这种影响最为显著：MP-CoT 在 MusicBrainz 上将 F1 分数提升了至多 38%。**其次**，MP-CoT 通过让其他视角在某种信号不可用时进行补偿，表现出强大的鲁棒性。即使存在缺失数据（此时基于启发式的方法通常会失效），所有数据集上的 F1分数仍保持在 90% 以上。**最后**，分析进一步证实，这些推理视角彼此互补且不冗余，并通过相互弥补局限性，确保系统能够适应模式退化：句法和语义线索可应对数据稀疏性，句法和统计证据可缓解语义歧义，而当模式名称缺失时，统计信号则至关重要。

4.6 整体冲突消解分析

我们通过比较冲突消解前优化器的输出（CR前）与验证器生成的最终预测结果（CR后），评估整体冲突消解（CR）策略的有效性。如表4所示，优化器经过有意设计，采取较为激进的策略，优先保证较高的召回率，但代价是精确率较低。

验证器充当关键的全局过滤器，用于消解结构不一致。在所有数据集上，冲突消解显著提升精确率，提升幅度从 +14% 到 +36%，同时基本保持召回率。在五个数据集中的四个里，优化器识别出的所有真实外键在冲突消解后均得以保留。值得注意的是，所遇到的所有循环引用严格为长度 2，从而确保消解过程避免了与全局遍历相关的级联复杂性。这些结果表明，整体性的模式级推理对于调和局部预测并确保全局一致性至关重要，使 LLM-FK 能够同时实现高准确率与结构正确性。

5 结论

我们提出**LLM-FK**，这是首个面向大规模复杂数据库外键检测的多智能体框架。通过编排四个专门化智能体，LLM-FK 分解检测任务，执行基于语义的推理，并强制保证全局一致性。具体而言，分析器通过唯一键驱动的模式分解策略缩小搜索空间；解释器与优化器利用自增强领域知识注入和多视角思维链推理进行深入而稳健的推断；验证器则通过整体冲突解决策略调和局部预测。大量实验表明，LLM-FK 持续达到最先进性能，为外键发现提供准确、高效且稳健的解决方案，并为下游数据库管理与分析任务奠定实用基础。

局限性

尽管我们的实验涵盖了五个广泛采用的基准数据集，涉及商业、零售、金融、制造业和娱乐领域，并展示了 LLM-FK 强大的泛化能力与稳健性，但仍有一些数据库领域尚未探索，这些领域具有极大规模或高度异构的数据库模式。重要的是，LLM-FK 旨在实现领域无关，仅依赖模式本身固有的信号，而非特定领域知识，从而确保其性能不受特定惯例或语言的影响。此外，其模块化的多智能体设计可无缝扩展至新的数据库，无需额外训练或监督。结合候选剪枝和结构化推理策略，即使面对复杂模式，LLM-FK 也能保持计算上的可行性和稳健的泛化能力，因此适用范围广泛，同时也为未来在专业领域开展评估留出了空间。

致谢

本研究得到了国家数据空间技术与系统重点实验室的支持。

参考文献

Ziawasch Abedjan、Lukasz Golab 和 Felix Naumann。2015年。关系数据画像：一项综述。*TheVLDBJournal*，24(4):557–581。

ArchitaAgarwal、Marilyn George、Aaron Jeyaraj 和 Malte Schwarzkopf。2021年。为遗留数据库改造通用数据保护条例合规性。*VLDB*基金会会议录，15(4)。

阿里巴巴。2022。 Alibaba-Java-Coding-Guidelines：GitHub 仓库。GitHub 仓库，访问于 2026-02-05。

Michael Armbrust、Ali Ghodsi、Reynold Xin、Matei Za- haria 等 1 位作者。2021年。湖仓：统一数据仓库与高级分析的新一代开放平台。载于*CIDR* 论文集，第 8 卷，第 28 页。sn。

Simone Balloccu、Patrícia Schmidtová、Mateusz Lango 和 Ondˇrej Dušek。2024年。泄露、作弊、重蹈覆辙：闭源大语言模型中的数据污染与评估不端行为。载于第 *18* 届欧洲计算语言学协会会议论文集（第 *1* 卷：长篇论文），第 67–93 页。

Jana Bauckmann、Ulf Leser、Felix Naumann 和 Véronique Tietz。2006 年。高效检测包含依赖关系。载于*2007*年 *IEEE* 第 *23* 届国际数据工程会议，第 1448–1450 页。IEEE。

Philip A Bernstein 和 Eric Newcomer。2009年。事务处理原理。Morgan Kaufmann。Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell 和其他 1 位作者。2020年。语言模型是少样本学习器。神经信息处理系统进展，33:1877–1901。Shimin Chen、Anastasia Ailamaki、Manos Athanassoulis、Phillip B Gibbons、Ryan Johnson、Ippokratis Pandis 和 Radu Stoica。2011。TPC-E 与 TPC-C：通过 I/O 对比研究表征新的 TPC-E 基准。*ACM SIGMOD Record*，39(3):5–10。Si-An Chen、Lesly Miculicich、Julian M Eisenschlos、Zifeng Wang、Zilong Wang、Yanfei Chen、Yasuhisa Fujii、Hsuan-Tien Lin、Chen-Yu Lee 和 Tomas Pfister。2024年。TableRAG：利用语言模型实现百万 token 的表格理解。神经信息处理系统进展，37:74899–74921。Wenhu Chen。2023年。大语言模型是少样本（1）轮表格推理器。载于计算语言学协会论文集：*EACL 2023* 年论文集，第 1120–1130 页。Zhimin Chen、Vivek Narasayya 和 Surajit Chaudhuri。2014年。在 Microsoft SQL Server PowerPivot for Excel 中快速检测外键。*VLDB*基金会会议录，7(13):1417–1428。Zhoujun Cheng、Tianbao Xie、Peng Shi、Chengzu Li、Rahul Nadkarni、Yushi Hu、Caiming Xiong、Dragomir Radev、Mari Ostendorf、Luke Zettlemoyer 和其他 1 位作者。2022。将语言模型绑定到符号语言中。*arXiv* 预印本 *arXiv:2210.02875*。

Cody James Christopher、Kristen Moore 和 David Liebowitz。2021年。Schemadb：关系数据集中的结构。*arXiv* 预印本 *arXiv:2111.12835*。

Fabien De Marchi、Stéphane Lopes 和 Jean-Marc Petit。2002年。用于挖掘包含依赖的高效算法。载于国际数据库技术扩展会议，第464–476页。施普林格。

Till Döhmen、Radu Geacu、Madelon Hulsebos 和 Sebastian Schelter。2024年。Schemapile：大型关系数据库模式集合。*ACM*数据管理会议录，2(3)：1–25。

Falco Dürsch、Axel Stebner、Fabian Windheuser、Maxi Fischer、Tim Friedrich、Nils Strelow、Tobias Bleifuß、Hazar Harmouch、Lan Jiang、Thorsten Papenbrock 及另外1位作者。2019。包含依赖发现：对十三种算法的实验评估。载于第*28*届*ACM*国际信息与知识管理会议，第219–228页。

Raul Castro Fernandez、Ziawasch Abedjan、Famien Koko、Gina Yuan、Samuel Madden 和 Michael Stonebraker。2018年。Aurum：一个数据发现系统。输入*2018*年 *IEEE* 第*34*届国际数据工程会议 *(ICDE)*，第1001–1012页。IEEE。

Raul Castro Fernandez、Aaron J Elmore、Michael J Franklin、Sanjay Krishnan 和 Chenhao Tan。2023年。大语言模型将如何颠覆数据管理。*VLDB*基金会会议录，16(11):3302–3309。Dawei Gao、Haibin Wang、Yaliang Li、Xiuyu Sun、Yichen Qian、Bolin Ding 和 Jingren Zhou。2023年。大语言模型赋能的文本到SQL：一项基准评估。*arXiv*预印本 *arXiv:2308.15363*。Cyril Goutte 和 Eric Gaussier。2005年。精确率、召回率和F分数的概率解释及其对评估的影响。载于欧洲信息检索会议，第345–359页。施普林格。Aaron Grattafiori、Abhimanyu Dubey、Abhinav Jauhri、Abhinav Pandey、Abhishek Kadian、Ahmad Al-Dahle、Aiesha Letman、Akhil Mathur、Alan Schelten、Alex Vaughan 等。2024年。Llama 3 模型家族。*arXiv*预印本 *arXiv:2407.21783*。Shuhao Guan 和 Derek Greene。2024年。推进OCR后处理校正：合成数据的比较研究。载于计算语言学协会 *ACL 2024* 论文集：*Findings*，第6036–6047页。Daya Guo、Dejian Yang、Haowei Zhang、Junxiao Song、Peiyi Wang、Qihao Zhu、Runxin Xu、Ruoyu Zhang、Shirong Ma、Xiao Bi 等。2025年。DeepSeek-R1：通过强化学习激发大语言模型的推理能力。*arXiv*预印本 *arXiv:2501.12948*。

Taicheng Guo、Xiuying Chen、Yaqi Wang、Ruidi Chang、Shichao Pei、Nitesh V Chawla、Olaf Wiest 和 Xiangliang Zhang。2024年。基于大语言模型的多智能体：进展与挑战综述。*arXiv*预印本 *arXiv:2402.01680*。

Rihan Hai、Christos Koutras、Christoph Quix 和 Matthias Jarke。2023年。数据湖：功能与系统综述。*IEEE*知识与数据工程汇刊，35(12):12571–12590。

Alon Halevy、Michael Franklin 和 David Maier。2006年。数据空间系统原理。输入*ACM SIGMOD-SIGACT-SIGART*数据库系统原理研讨会第二十五届会议论文集，第1–9页。

Jan L Harrington。2016年。关系数据库设计与实现。Morgan Kaufmann。

Gonzalo Jaimovitch-López、Cèsar Ferri、José Hernández-Orallo、Fernando Martínez-Plumed 和 María José Ramírez-Quintana。2023年。语言模型能否自动完成数据整理？机器学习，112(6):2053–2082。

Lan Jiang 和 Felix Naumann。2020年。全面检测主键和外键。《智能信息系统杂志》，54(3):439–461。

Theodore Johnson、Amit Marathe 和 Tamraparni Dasu。2003年。数据库探索与 Bellman。*IEEE DataEng. Bull.*，26(3):34–39。Youri Kaminsky、Eduardo HM Pena 和 Felix Nau- mann。2023年。发现相似性包含依赖。*ACM*数据管理会议录，1(1):1–24。Moe Kayali、Anton Lykov、Ilias Fountalis、Nikolaos Vasiloglou、Dan Olteanu 和 Dan Suciu。2023年。Chorus：用于统一数据发现与探索的基础模型。*arXiv* 预印本 *arXiv:2306.09610*。Guolin Ke、Qi Meng、Thomas Finley、Taifeng Wang、Wei Chen、Weidong Ma、Qiwei Ye 和 Tie-Yan Liu。2017。LightGBM：一种高效的梯度提升决策树。神经信息处理系统进展，30。Robert E Kent。2012年。数据库语义。*arXiv* 预印本 *arXiv:1209.3054*。Georg Lausen。2007年。RDF 中的关系数据库：键与外键。输入数据库与信息系统本体技术研讨会，第 43–56 页。施普林格。Jinyang Li、Binyuan Hui、Ge Qu、Jiaxi Yang、Binhua Li、Bowen Li、Bailin Wang、Bowen Qin、Ruiying Geng、Nan Huo 等。2023年。大语言模型能否已用作数据库接口？一个面向大规模数据库落地文本到SQL任务的大型基准测试。神经信息处理系统进展，36:42330–42357。

Yiming Lin、Yeye He 和 Surajit Chaudhuri。2023年。Auto-bi：利用局部连接预测和全局模式图自动构建双模型。*arXiv* 预印本 *arXiv:2306.12515*。

Jim Melton 和 Alan R Simon。2001年。*SQL:1999*：理解关系语言组件。Elsevier。

微软。2023a。AdventureWorks 示例数据库：GitHub 仓库。GitHub 仓库，访问于2025年3月24日。

微软。2023b。Northwind 示例数据库：GitHub 仓库。GitHub仓库，访问于2025年3月24日。

Avanika Narayan、Ines Chami、Laurel Orr、Simran Arora 和 Christopher Ré。2022年。基础模型能否处理您的数据？*arXiv*预印本 *arXiv:2205.09911*。

Sam Newman。2021年。构建微服务：设计细粒度系统。“O’Reilly Media, Inc.”。

Thorsten Papenbrock、Tanja Bergmann、Moritz Finke、Jakob Zwiener 和 Felix Naumann。2015a。使用 Metanome 进行数据剖析。*VLDB*基金会会议录，8(12)：1860–1863。

Thorsten Papenbrock、Sebastian Kruse、Jorge-Arnulfo Quiané-Ruiz 和 Felix Naumann。2015b。基于分而治之的包含依赖发现。*VLDB*基金会会议录，8(7):774–785。Meikel Poess 和 Chris Floyd。2000年。面向决策支持和网络商务的新 TPC 基准。*ACM SIGMOD Record*，29(4):64–71。Alexandra Rostin、Oliver Albrecht、Jana Bauckmann、Felix Naumann 和 Ulf Leser。2009年。一种外键发现的机器学习方法。载于 *WebDB*。Aaditya Singh、Adam Fry、Adam Perelman、Adam Tart、Adi Ganesh、Ahmed El-Kishky、Aidan McLaughlin、Aiden Low、AJ Ostrow、Akhila Ananthram 及其他 1 位作者。2025年。OpenAI GPT-5 系统卡。*arXiv* 预印本 *arXiv:2601.03267*。Songyuan Sui、Hongyi Liu、Serena Liu、Li Li、Soo-Hyun Choi、Rui Chen 和 Xia Hu。2025年。Chain-of-query：通过多代理协作，释放大语言模型在 SQL 辅助表格理解中的潜力。载于第*14*届国际自然语言处理联合会议暨第*4*届亚太计算语言学协会分会会议论文集，第 957–986 页。Yuan Sui、Mengyu Zhou、Mingjie Zhou、Shi Han 和 Dongmei Zhang。2024a。表格遇见大语言模型：大语言模型能理解结构化表格数据吗？一项基准与实证研究。载于第*17*届*ACM*国际网络搜索与数据挖掘会议，第 645–654 页。

Yuan Sui、Jiaru Zou、Mengyu Zhou、Xinyi He、Lun Du、Shi Han 和 Dongmei Zhang。2024b。Tap4llm：面向大语言模型推理的半结构化数据采样、增强与打包表格提供工具。载于计算语言学协会论文集：*EMNLP 2024* 精选论文，第 10306–10323 页。

Aaron Swartz。2002年。Musicbrainz：一种语义网服务。*IEEEIntelligentSystems*，17(1):76–77。

Fabian Tschirschnitz、Thorsten Papenbrock 和 Felix Naumann。2017。检测数量极多的表中的包含依赖。*ACM*数据库系统汇刊（*TODS*），42(3):1–29。

Bing Wang、Changyu Ren、Jian Yang、Xinnian Liang、Ji-aqi Bai、Linzheng Chai、Zhao Yan、Qian-Wen Zhang、Di Yin、Xing Sun 及另外1位作者。2025a。Mac-sql：一种用于文本到SQL的多智能体协作框架。载于第*31*届计算语言学国际会议论文集，第540–557页。

Xixi Wang、Miguel Costa、Jordanka Kovaceva、Shuai Wang 和 Francisco C Pereira。2025b。将模式图接入多表问答：一种减少对大语言模型依赖的人工引导框架。载于计算语言学协会会议论文集：*EMNLP2025*，第5829–5842页。

Jason Wei、Xuezhi Wang、Dale Schuurmans、Maarten Bosma、Fei Xia、Ed Chi、Quoc V Le、Denny Zhou及另外1位作者。2022年。思维链提示能够激发大语言模型的推理能力。神经信息处理系统进展，35：24824–24837。

Jian Wu、Linyi Yang、Dongyuan Li、Yuliang Ji、Manabu Okumura和Yue Zhang。2025年。MMQA：使用多表多跳复杂问题评估大语言模型。输入第十三届国际学习表示会议。

Xiaoyu Wu、Ning Wang和Huaxi Liu。2019年。借助众包发现网页表格中的外键。*Computing and Informatics*，38(3)：621–646。

Wenxuan Xie、Gaochen Wu和Bowen Zhou。2024年。Mag-sql：采用软模式链接和迭代式子SQL细化的文本到SQL多智能体生成方法。*arXiv*预印本 *arXiv:2408.07930*。

Cong Yan 和 Yeye He。2020年。Auto-suggest：使用数据科学笔记本学习推荐数据准备步骤。载于*2020*年*ACM SIGMOD*数据管理国际会议论文集，第 1539–1554 页。

An Yang、Anfeng Li、Baosong Yang、Beichen Zhang、Binyuan Hui、Bo Zheng、Bowen Yu、Chang Gao、Chengen Huang、Chenxu Lv 等。2025年。Qwen3 技术报告。*arXiv* 预印本 *arXiv:2505.09388*。

Yunhu Ye、Binyuan Hui、Min Yang、Binhua Li、Fei Huang 和 Yongbin Li。2023年。大型语言模型是多才多艺的分解器：为基于表格的推理分解证据和问题。载于第 *46* 届 *ACM SIGIR* 信息检索研究与发展国际会议论文集，第 174–184 页。

Peiying Yu、Guoxin Chen 和 Jingjing Wang。2025年。Table-critic：一种用于表格推理中的协作式批评与改进的多智能体框架。载于第*63*届计算语言学协会年会论文集（第*1*卷：长篇论文），第17432–17451 页。

Bin Zhang、Hangyu Mao、Lijuan Li、Zhiwei Xu、Dapeng Li、Rui Zhao 和 Guoliang Fan。2024年。多智能体系统中的序贯异步动作协调：一种 Stackelberg决策变换器方法。载于第四十一届国际机器学习会议。

Meihui Zhang、MariosHadjieleftheriou、Beng Chin Ooi、Cecilia M Procopiuc 和 Divesh Srivastava。2010年。多列外键发现。*VLDB*基金会会议录，3(1-2)：805–814。

Wei Zhou、Mohsen Mesgar、Annemarie Friedrich 和 Heike Adel。2025年。借助工具使用，在复杂表格问答中实现在线规划的高效多智能体协作。载于计算语言学协会：*NAACL 2025*年论文集，第945–968 页。

Jun-Peng Zhu、Peng Cai、Kai Xu、Li Li、Yishen Sun、Shuai Zhou、Haihuang Su、Liu Tang 和 Qi Liu。2024年。AutoTQA：借助多智能体大语言模型实现自主表格问答。*VLDB*基金会会议录，17(12):3920–3933。

A 更多相关工作

A.1 元数据发现

大量研究致力于探索高效策略，以发现唯一列组合（UCC）和包含依赖（IND），使得即使在大型数据库中，De Marchi 等人，2002年；Abedjan 等人，2015；Dürsch 等人，2019）也能以计算上可行的方式提取此类元数据。具有代表性的算法包括**Spider**（Bauckmann 等人，2006年），它使用基于磁盘的排序合并连接，高效识别单列包含依赖；以及**Binder**（Papenbrock 等人，2015b），它通过基于哈希的分治策略将该方法推广到多列包含依赖。

为应对脏或不完整数据等实际挑战，近似包含依赖发现技术也被提出（Tschirschnitz等人，2017；Kaminsky等人，2023年）。元数据

LLM-FK中的发现过程与这些成熟算法完全兼容，使从业者能够根据特定目标定制该框架——例如优先考虑计算效率，或最大限度地提高对数据异常的鲁棒性——而无需修改整体推理流水线。

A.2 多智能体系统

近年来，多智能体系统通过促进专门化智能体之间的协作，在解决复杂推理任务方面展现出巨大潜力（Guan 和 Greene，2024年；Guo 等人，2024年；Zhang 等人，2024年）。在表格领域，现有的多智能体框架主要面向文本到SQL和表格问答等下游应用（Xie 等人，2024年；Sui 等人，2025年；Wang 等人，2025a；Yu 等人，2025年；Zhou 等人，2025年）。这些任务通常假设模式已知，并依赖内容匹配和局部模式链接来满足特定查询。

相比之下，外键检测面临着根本不同的挑战：它要求对整个数据库模式进行全局结构推理。据我们所知，LLM-FK 是首个专门利用多代理协作进行外键检测的框架，展示了专门化智能体如何协同应对大型复杂数据库中固有的组合性挑战和不一致性挑战。这一设计展现了多智能体推理在内容导向任务之外的潜力，并将其拓展至结构推断和模式级约束发现。

A.3 现代数据生态系统中的外键检测

尽管外键是关系数据库理论的基础，但在实际应用中往往未对其进行显式定义，因此外键检测成为下游数据操作的一项关键任务。在当今环境下，这一问题的紧迫性和实际意义可从以下两个关键方面概括：

缺失外键的普遍性。在现代数据环境中，显式外键的缺失已成为一种普遍现象，这主要是由不断演进的工程实践和架构转变所推动的。为避免高并发操作导致性能下降，开发人员经常省略物理外键，将约束管理完全转移至应用层（阿里巴巴，2022），使这些隐式约束在数据库迁移或分析提取过程中极易丢失（Jiang 和 Naumann，2020年）。此外，分布式架构——例如微服务中的“每个服务对应一个数据库”模式（Newman，2021年）以及读时模式数据湖仓（Armbrust 等，2021年）——从根本上限制了严格的跨数据库外键约束。实证研究证实了这种稀缺性，表明绝大多数真实数据中的开源Web应用和关系数据库完全缺乏显式外键定义（Agarwal 等，2021年；Christopher 等，2021年；Döhmen 等，2024年）。

在高级数据应用中的关键作用。除了数据集成和模式逆向工程等传统数据管理任务外，外键检测在现代数据生态系统中也日益发挥着关键作用。在复杂、异构的环境中，识别外键关系对于简化数据发现和提高治理效率极具价值（Fernandez 等人，2018年；Hai 等人，2023年）。此外，准确推断这些结构依赖关系，为实现自动化商业智能（BI）建模提供了必要基础（Lin 等人，2023年）；外键检测与BI模型构建的基础步骤密切相关，并支持自动生成模式图，以供下游分析流水线使用。在当今LLM驱动的数据库接口时代，显式外键能够提供确定性连接路径，实证研究已证明这能显著提高文本到SQL解析的准确性（Gao 等人，2023年）。更广泛地说，多表问答（QA）近来已成为LLM驱动的表格理解领域的焦点（Zhu 等人，2024年；Wang 等人，2025b），其中外键检测被明确视为解决跨多个不同表的多跳复杂问题的先决条件（Wu 等人，2025年）。因此，可靠的外键检测是将结构化关系语义无缝融入新兴基于LLM的表格推理系统的关键助力。

B 更多实现细节

B.1 多代理协作的详细阐述

为实现外键检测所需的推理自动化，LLM-FK 通过协调四个专门化智能体——**分析器**、**解释器**、**优化器**和**验证器**——协作执行数据库范围的外键检测任务。每个智能体通过特定的大语言模型指令执行不同的

认知操作，这些指令表示为 π。各智能体的角色和操作形式化定义如下：

分析器（$\mathcal{A}_p$）。分析器作为代理工作流的入口，实现**唯一键驱动的模式分解**，以缓解搜索空间爆炸。给定数据库$\mathcal{D}$、已识别的包含依赖 $\mathcal{J}$（经过数据类型规则预筛选），以及从 $\mathcal{J}$ 中的被引用表推导出的最小唯一列组合$\mathcal{U}$，分析器遍历每个被引用表 T。利用其最小唯一列组合子集 $\mathcal{U}_T \subseteq \mathcal{U}$，在 $\mathcal{I}_{\mathcal{A}_p}$的指导下选择最合适的被引用唯一键 k：

$$k = \pi(T, \mathcal{U}_T, \mathcal{I}_{\mathcal{A}_p}), \tag{1}$$

这些键被汇总为集合 $\mathcal{K}$，从而确定性地剪枝搜索空间。结构上有效的候选外键对随后为：

$$\Psi_{cand} = \{(c_f, c_p) \in \mathcal{J} \mid c_p \in \mathcal{K}\}, \tag{2}$$

解释器（$\mathcal{A}_i$）。解释器执行**自增强领域知识注入**，以减少模式歧义。通过分析 Ψ_{cand}，它从表名集合 $\mathcal{N}_{\Psi_{cand}}$中推导出全局领域知识 $\mathcal{K}_{dom}$：

$$\mathcal{K}_{dom} = \pi(\mathcal{N}_{\Psi_{cand}}, \mathcal{I}_{\mathcal{A}_i}), \tag{3}$$

该知识会在下游共享，以确保各局部推理任务之间的语义一致性。

优化器（$\mathcal{A}_r$）。优化器使用**多视角思维链推理**验证每个候选对 $\psi = (c_f, c_p) \in \Psi_{cand}$，并结合领域知识 $\mathcal{K}_{dom}$ 以及从 SQL 查询中提取的序列化结构表示 S_ψ：

$$p_\psi = \pi(\psi, S_\psi, \mathcal{K}_{dom}, \mathcal{I}_{\mathcal{A}_r}), \tag{4}$$

其中，$p_\psi \in \{0, 1\}$ 表示 ψ 是否构成真实外键。我们将通过验证的正向候选集合表示为：

$$\Psi_{pos} = \{\psi \in \Psi_{cand} \mid p_\psi = 1\}, \tag{5}$$

验证器（$\mathcal{A}_v$）。验证器通过**整体冲突消解策略**确保全局一致性。它以 Ψ_{pos}初始化，并识别汇总模式拓扑中的结构违规（例如多个引用或循环引用）。对于识别出的任何冲突集 $\Psi_{conflict}$，它会确定有效子集 $\Psi_{resolved}$ 利用领域知识 $\mathcal{K}_{dom}$和结构表示 $S_{\Psi_{conflict}}$：

$$\Psi_{resolved} = \pi(\Psi_{conflict}, S_{\Psi_{conflict}}, \mathcal{K}_{dom}, \mathcal{I}_{\mathcal{A}_v}). \tag{6}$$

该消解机制通过剔除无效候选外键来优化全局模式拓扑，确保收敛至无冲突状态，并得到最终的一致外键集 Φ。

多代理协作。整体工作流程在算法2中进行了形式化定义。各智能体的详细提示词见附录F。

B.2 分析器的详细阐述

B.2.1 基于包含依赖和规则的剪枝

为有效缩小搜索空间，分析器强制执行两条基本剪枝规则：

- **类型兼容性：** 候选列必须严格具有相同的数据类型。
- **排除不适用的类型：** 表示连续测量值（例如 float、decimal）或二元状态（例如 bool、bit）的列会被自动过滤，因为从语义上来说，它们不具备标识属性。

除了这些基本约束外，我们基于包含依赖和基于规则的机制还具备很强的灵活性和可扩展性。基于包含依赖的验证支持自适应算法选择（见附录 A.1），以优化执行效率与鲁棒性之间的平衡。此外，过滤空表或空列以及应用特定基数阈值等附加规则，都可以无缝集成到基于规则的剪枝中。

B.2.2 唯一键选择

最优唯一键选择会借助形式化判断标准引导的大语言模型提示（见表5），这些标准包括唯一性有效性、列顺序、命名约定、类型适用性、值简洁性、最小构成和语义对齐。通过将选择过程建立在这些形式化规则之上，大语言模型能够有效地从最小唯一列组合中辨别出最合适的被引用唯一键。这种方法利用模型的推理能力，优先选择不仅满足结构约束、也符合数据库模式逻辑惯例的候选项，确保所选键在语义上有意义。

标准	描述
唯一性有效性	候选项必须能够在表中唯一标识特定元组。
列序	表定义中位置靠前的列优先。
命名约定	优先选择包含典型标识符（如“编号”、“键”、“代码”）的列名。
类型适用性	数据类型通常为整数或字符串，符合标准键定义 .
值简洁性	数据值应简洁且便于人类阅读（文本长度较短）。
最小构成	优先选择组成列较少的候选（原子键），而非复杂的复合键。
语义对齐	根据语义上下文，键必须在逻辑上代表该实体表的 .

表5：唯一键选择的判断标准。

Type	信息
模式定义	表名 列名
统计元数据	序号位置 数据类型 平均值文本长度 不同值数量 表中的行数 基数比（不同值 / 表行数） 最小值 最大值
列间依赖关系	覆盖率 表大小比率 超范围率
样本数据	表中的示例数据（前 5 行）

表 6：用于多视角思维链推理的序列化结构表示。

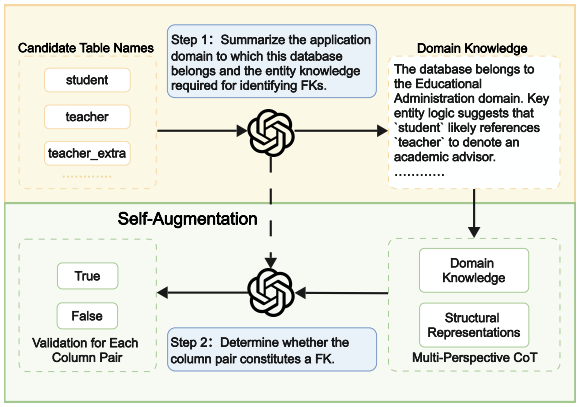

图 4：自增强领域知识注入示意图。

B.3 解释器的详细阐述

解释器采用先总结再推理的策略（见图4），这与人类在做出具体判断前先整体理解数据库领域的认知过程相仿。在这一工作流中，大语言模型首先从各表的语义中综合出全局领域概况，为自身构建统一的上下文。关键的是，这些自行生成的见解会被注入局部推理任务，使该框架能够利用具体领域知识补充抽象模式符号。这种从全局到局部的范式通过将孤立的推断建立在共同基础之上，确保语义一致性作为基础，同时实现全面自动化——自主推导宏观见解，无需依赖外部人工标注。

B.4 优化器的详细阐述

结构化表示提取。 优化器通过对原始表数据构造 SQL 查询，生成序列化的结构表示（见表6），其中包括模式定义、统计元数据、列间依赖关系和样本数据。这确保了多视角推理以精确的模式和数据证据为依据。

多视角思维链推理。 优化器整合了三种推理视角（见图 5）：

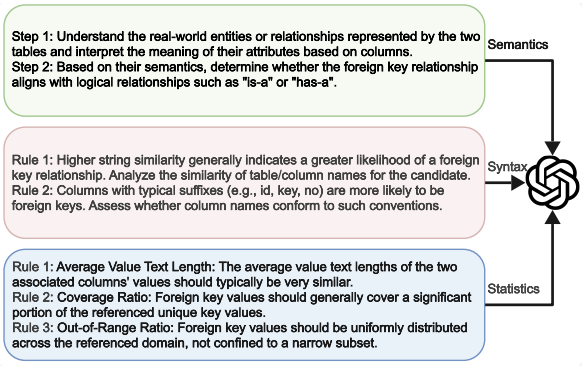

图 5：多视角思维链推理示意图。

- **语义：** 将模式概念映射到真实数据世界中的实体。
- **句法：** 评估命名约定和字符串相似度。
- **统计：** 分析统计分布和包含关系。

B.5 验证器的详细说明

验证器通过**整体冲突消解**算法（参见算法1），解决多重引用和循环引用，从而生成最终结构一致的外键集合 Φ。

算法 1整体冲突消解

要求：候选外键集 Ψ_{pos}确保：一致外键集 Φ 1：初始化 $\Phi \leftarrow \Psi_{pos}$2：// 阶段 1：解决多重引用 3：// 确保每个源列最多引用一个目标列 4：对于 c_{src} 引用不同目标列 $\mathcal{C}_{targets}$ 的每个源列 Φ 执行 5：如果 $|\mathcal{C}_{targets}| > 1$则 6：// 通过推理选择最合理的目标列 7：$c_{best} \leftarrow$ 推理选择$(c_{src}, \mathcal{C}_{targets})$ 8：更新 Φ ，仅保留c_{src}、c_{best}对 c_{src}的引用 9：结束如果 10：结束循环 11：// 阶段 2：解决循环引用 12：// 迭代消除环，以确保拓扑一致性 13：当模式拓扑图 $G(\Phi)$中包含环时 14：// 优先处理可明确解决的直接违规 15：$Cycle \leftarrow$查找最短环(G) 16：// 找出环中最不合理的链接（边） 17：$e_{weak} \leftarrow$ 推理判定最弱项$(Cycle)$ 18：$\Phi \leftarrow \Phi \setminus \{e_{weak}\}$ {打破该环} 19：结束循环 20：返回 Φ

C 数据集详情

表7详细展示了我们实验所用数据集的统计概况。这些数据集涵盖不同领域、模式复杂度和规模，能够全面评估外键检测方法。

- **TPC-H**：数据库领域广泛采用的标准决策支持基准。它模拟了一个模式相对简单（8 张表）、外键关系稀疏的商业环境，可作为正确性验证的基线。
- **Northwind**：一个经典的合成数据集，代表一家小型零售企业。该数据集包含 13 张表，提供了一个易于管理且具有代表性的模式，可用于评估典型库存和销售场景中的外键检测。
- **TPC-E**：一种在线事务处理（OLTP）（Bernstein 和 Newcomer，2009年）基准，用于模拟一家经纪公司。其其模式比 TPC-H 更复杂（33 张表），并包含更丰富的完整性约束。表名和列名中经常使用英语缩写，带来了额外的语义挑战。
- **AdventureWorks**：一个模拟自行车制造公司的合成数据库。它包含 71 张表和大量外键，呈现出复杂的模式结构，包括复合外键，这给外键检测算法带来了相当大的挑战。
- **MusicBrainz**：一个大规模、真实世界的音乐百科数据库。它包含 382 张表和超过 240 万个列对，是我们评估中最复杂的数据集，可作为测试平台，用于在嘈杂的异构环境中评估可扩展性和鲁棒性。

数据集	Type	表	FKs	列对	领域
TPC-H	基准	8	9	3,721	商业
Northwind	合成	13	11	7,744	零售
TPC-E	基准	33	46	36,481	金融
AdventureWorks	合成	71	91	236,196	制造业
MusicBrainz	Real	382	354	2,458,624	娱乐

表 7：数据集的详细统计信息和领域。

D 可复现性声明

D.1 LLM-FK 和基于大语言模型的基线

所有实验均使用**DeepSeek-R1**作为骨干大语言模型。为确保结果具有确定性且可复现，我们将温度参数设为 0，其他超参数则保持默认值。

对于元数据提取，我们采用**Metanome**分析工具（Papenbrock 等人，2015a）来识别结构约束；具体而言，我们关注最小唯一列组合，并利用*Spider* 算法（Bauckmann 等人，2006年）来检测单列包含依赖。

对于基于大语言模型的基线方法：**Chorus**使用其官方开源代码实现。其他基于大语言模型的基线方法以及**LLM-FK**的详细提示设计见附录F和附录G。

D.2 基于启发式方法的基线

所有基于启发式方法的基线均严格按照各自论文中描述的方法重新实现，并仔细处理超参数和评估协议：

- **MC-FK**：由于未提供默认参数，我们采用报告的外键检测置信度分数，并通过经验调整决策阈值，以最大化每个数据集上的 F1 分数。

- **Fast-FK**：由于缺少明确的默认参数，我们根据先前文献和初步调优，为其组成规则分配合理的权重：列名相似度（0.2）、列表名相似度（0.5）、基数比（0.3），以及引用列是否为唯一键（-0.5）。

- **HOPF**：严格按照原始研究中的默认配置实现。

- **ML-FK** 和 **AutoSuggest**：我们采用**ML-FK**提出的留一数据库训练方案。具体而言，对于每个目标数据库，其余四个数据集中的带标签外键候选项用作训练数据。**LightGBM** (Ke 等人，2017) 被用作分类器，因为它效率高且性能出色；所有实验均统一使用默认超参数。

E 更多实验结果与分析

E.1 对不同大语言模型骨干的泛化能力

为证明 LLM-FK 的有效性并不依赖于某个特定的大语言模型骨干，我们使用 DeepSeek-R1（Guo 等人，2025年）、Qwen3-Max（Yang 等人，2025年）、GPT-5-mini（Singh 等人，2025年）和 LLaMA3.3-70B-Instruct（Grattafiori 等人，2024年）评估了我们的框架，且未针对任何模型进行微调。

候选剪枝的泛化能力。如表 8所示，分析器在所有大语言模型上均能持续、大幅且可靠地缩减候选项搜索空间（例如，在 MusicBrainz 上从超过 240 万个缩减至 ∼2,355 对）。至关重要的是，这种缩减始终保留了真正例匹配；所有试验中仅有一个真实外键被错误剪枝，具体发生在 LLaMA3.3 对 AdventureWorks 的处理中。这凸显了我们剪枝方法的高可靠性及其与模型无关的泛化能力。

外键检测的泛化能力。表9报告了外键检测的 F1 分数。与整体表现最佳的端到端基线相比，LLM-FK 在 TPC-E、AdventureWorks 和 MusicBrainz 等复杂模式上显著优于它们，展现了我们的多智能体协作方法在不同底层大语言模型上的泛化能力。在所评估的

模型中，LLaMA 3.3 的表现相对较低，这可能归因于其在处理复杂逻辑约束时固有推理能力的差异。尽管如此，这些结果仍持续优于绝大多数传统的基于启发式的方法（见表1）和端到端基线，进一步证明了 LLM-FK 的有效性。最后，在 TPC-H 等较小数据集上观察到的 F1 分数略低，是受限模式规模所造成的结果；实际上，只有少数外键被误分类，这会使指标受到不成比例的影响，并不表明方法存在根本性局限。

E.2 计算成本分析

继第 4.3 节的初步搜索空间分析之后，我们详细介绍与 LLM 调用相关的运行时开销和计算支出。我们以 Qwen3-Max 作为后端大语言模型，对 API 调用、token用量和货币成本进行全面分析。定价依据 Qwen3-Max 的收费标准：输入不超过 32K 词元时，每 32K 词元收费 2.5 人民币；输出不超过 32K 词元时，每 32K 词元收费 10 人民币。

总体比较。 为评估大语言模型的调用频率及相关费用，我们将方法（LLM-FK）与端到端基线进行比较。该基线通过单次处理识别一对表之间的外键，是本研究中效果最好、效率最高的基线。总体比较结果汇总于表 11。对于 TPC-H 等较小的数据库模式，LLM-FK 的 F1 分数略低（0.78，而基线为 0.90），成本也略高。不过，这一性能差距实际上仅对应少数几个外键分类错误，而数据集规模较小放大了这一差距。对于 TPC-E 和 AdventureWorks 等较大的数据集，LLM-FK 需要更高的 API 成本和更多的token用量。尽管如此，这种额外支出仍是值得的权衡，因为它带来了准确率的大幅提升，并在 API 调用方面表现更佳。尤其是在 MusicBrainz 这类庞大而复杂的数据库模式上，LLM-FK 不仅实现了最高准确率（0.96），还大幅降低了货币成本，展现出在大规模环境下更优的可扩展性和效率。

内部分解。为进一步阐明计算开销的来源，我们提供了各内部代理的详细分解在我们的多智能体框架中：分析器、解释器、优化器和验证器。如表12，分析清楚地表明，主要的时间和成本支出集中于优化器模块。相反，验证器产生的开销极小，因为其输入已被优化器大幅剪枝，例如将2,355个候选减少至仅92个冲突，在MusicBrainz 数据集中。此外，这些剩余冲突自然形成紧密聚集的局部语义子图；值得注意的是，我们在实验中遇到的所有循环引用长度严格为2（见附录E.7 的典型示例）。通过迭代优先解决这些最短环，验证器处理的是高度孤立的直接邻域冲突，而非复杂的级联链，从而只需显著更少的计算开销即可全局解决。

E.3 缺少 UCC 元数据时的鲁棒性

为评估我们的方法在空表条件下（无法获取 UCC）的剪枝效果，我们开展了一项实验来验证其鲁棒性。如表 10 所示，结果表明，该方法在全部五个基准数据集上均展现出稳定出色的剪枝效果。保留候选项的数量与原始数据充足环境下的数量非常接近，且仅有一个真实外键（位于 TPC-E 中）被错误剪枝。这种高度一致性源于，如表 5 所示，我们的方法借助大语言模型，从多个维度进行全面推理，例如列序、命名约定和语义对齐。这种多维推理使我们的框架能够直接根据模式推断出合理的被引用唯一键，有效弥补 UCC 的缺失，确保其即使在完全空表的环境中也具备鲁棒性。此外，值得注意的是，在实际数据库操作中，完全为空的表通常是临时表或暂存表，很少需要复杂的结构分析。

E.4 唯一键驱动的模式分解示例分析

列级处理与表级处理。 以往的方法将整个表对输入大语言模型，要求一次性识别所有外键。这往往会导致注意力分散和幻觉，

数据集	原始配对	DeepSeek-R1		Qwen3-Max		LLaMA3.3		GPT-5-mini	
		候选项	Err.	候选项	Err.	候选项	Err.	候选项	Err.
TPC-H	3,721	49	0	49	0	47	0	49	0
Northwind	7,744	60	0	60	0	60	0	60	0
TPC-E	36,481	194	0	194	0	195	0	192	0
AdventureWorks	236,196	2,034	0	2,057	0	1,987	1	2,024	0
MusicBrainz	2,458,624	2,306	0	2,355	0	2,355	0	2,355	0

表 8：不同大语言模型下分析器的候选剪枝结果。每个条目显示候选项（候选）数量和错误剪枝（错误）数量，其中错误剪枝是指在剪枝过程中被错误移除的外键。

LLM	方法	TPC-H	Northwind	TPC-E	AdventureWorks	MusicBrainz
DeepSeek-R1	端到端	0.90	1.00	0.69	0.73	0.75
	LLM-FK（我们的方法）	**1.00**	**1.00**	**0.93**	**0.94**	**0.95**
Qwen3-Max	端到端	**0.90**	0.92	0.58	0.52	0.75
	LLM-FK（我们的方法）	0.78	**1.00**	**0.92**	**0.92**	**0.96**
LLaMA3.3	端到端	**0.80**	0.95	0.61	0.66	0.74
	LLM-FK（本文方法）	0.70	**0.96**	**0.85**	**0.79**	**0.86**
GPT-5-mini	端到端	**0.90**	0.95	0.67	0.79	0.80
	LLM-FK（本文方法）	0.78	**1.00**	**0.93**	**0.95**	**0.93**

Tab表 9：不同大语言模型的外键检测 F1 分数对比。每个大语言模型的最佳结果以粗体显示 .

数据集	配对	原始		空	
		候选	错误	候选	错误
TPC-H	3,721	49	0	49	0
Northwind	7,744	60	0	69	0
TPC-E	36,481	194	0	156	1
AdventureWorks	236,196	2,034	0	2,127	0
MusicBrainz	2,458,624	2,306	0	2,138	0

表10：原始表与空表中的剪枝结果。候选项表示保留候选项；错误表示被误剪枝的真实外键。

尤其是在同一组表之间存在多个外键时。

例如，在 TPC-E 中，holding_history 表通过两个不同的外键引用 trade：hh_h_t_id → t_id 和 hh_t_id →t_id。在我们的实验中，表级方法通常只能检测到一个外键。相比之下，即使不使用自增强领域知识注入或多视角思维链推理，LLM-FK 也能成功识别这两个外键，这要归功于我们的列对分解。通过将数据库模式分解为原子单元，我们降低了任务复杂度，使其不受数据库模式规模影响，从而能够针对每个候选对进行聚焦推理，并确保在大型数据库模式中依然具有可处理性。

基于唯一键的剪枝与基于主键的剪枝。我们基于唯一键的剪枝

比单纯依赖已声明的主键更稳健。在复杂模式中，主键往往是最小唯一列组合的超集，会引入不必要的冗余。例如，在 production_ productproductphoto（Adven- tureWorks）中，主键是复合键（ProductID+ ProductPhotoID），而 ProductID 单独就构成一个最小唯一列组合，并且是所有引用外键的目标。如果剪枝依赖这个过度复合的主键，那么涉及 ProductPhotoID 的无关候选对就会保留下来，从而增加计算成本。相比之下，基于唯一键的剪枝能在维护外键完整性的同时，高效缩小搜索空间。

E.5 自增强领域知识注入示例分析

我们分析来自 TPC-E 的 holding_summary.hs_s_ symb 和 daily_market.dm_s_symb 这一候选列对。这两列都带有后缀 s_symb（证券代码），但承担着不同的作用：daily_market 记录每日股票数据，而 holding_summary 汇总客户持仓。

由于缺乏领域知识，大语言模型无法解读这些缩写的语义，因此错误地依赖语法相似性，并误判存在外键。借助我们的解释器，全局领域上下文将该数据库识别为“金融交易系统”，并区分

“市场数据”和“账户持仓”。这些知识引导优化器拒绝该候选项，说明自增强领域知识能够有效消除语义歧义。

E.6 多视角思维链推理示例分析

除了确保 LLM-FK 在缺失数据或语义模糊的场景下具有鲁棒性外，多视角思维链推理还利用结构化表示中蕴含的充足信息，并进行深入的语义推理，以应对模糊或不一致的信号。例如，MusicBrainz 中的外键系列_annotation.annotation→ annotation.id涉及数值格式各不相同的列（科学计数法与整数）；由于前几行中的数值较大，前者被表示为科学计数法，而后者仍保持整数格式。如果不进行多视角推理，大语言模型会将科学计数法误解为字符串值，并因前几行中的数据类型和取值范围均不匹配而拒绝该外键。LLM-FK 通过以下方式解决这一问题：

- 注入统计元数据（数据类型、取值范围）以提供充足信息，防止误解。
- 交叉验证语法、语义和统计视角，确保推理稳健，不局限于仅依据格式差异或取值范围不匹配作出的表面判断。

E.7 整体冲突消解示例分析

在 MusicBrainz 中，优化器的输出里，l_artist_recording、artist 和 artist_meta 之间存在复杂冲突。具体而言，l_artist_recording.entity0 被模糊地识别为一个外键，既引用 artist.id，也引用 artist_meta.id。更棘手的是，artist 与 artist_meta 之间存在直接循环引用（即彼此引用），这掩盖了真实的层级关系。这类结构性冲突十分常见，并非随机发生。至关重要的是，优化器基于语义的推理会将这些高度相关的候选项约束在紧密聚集、语义相关的子图中。

通过构建全局模式拓扑，并综合分析冲突子图中的候选项，我们的整体冲突解决策略判定，artist 是语义核心实体，而 artist_meta 则充当辅助元数据。因此，我们通过保留指向 artist.id 的规范引用，并剔除指向 artist_meta.id 的虚假候选项来解决冲突。这一判定优于依赖为列对分配置信度分数的局部验证方法。此类分数往往因孤立推理而不准确，而我们构建全局模式拓扑，能够直接进行语义比较，从而捕捉层级关系。这一整体推理过程消除了相互冲突的引用，确保最终对齐在全局模式层级上兼具语义和结构一致性。

F LLM-FK 的提示词与案例研究

在本附录中，我们提供 LLM-FK 各核心智能体所用提示词的详细示例：分析器、解释器、优化器和验证器。这些示例展示了多智能体框架如何实现外键检测。

我们以图2中介绍的示例数据库作为贯穿全文的案例研究，该案例也展示了整体多智能体工作流。图1展示了基于启发式方法的局限性，以及大语言模型在外键检测语义推理方面的有效性，同时也突显了相应的挑战。图4 具体说明了解释器执行的自增强领域知识注入。

图6–10展示了流水线各阶段的具体提示词及其对应输出。具体而言，这些示例提示词直接填入结构表示，这些表示体现了 SQL 查询的执行结果。

G 基于大语言模型的基线提示词

为确保完整性，我们还提供了基于大语言模型的基线方法所用的提示词设计：端到端、少样本和思维链。图11–13 展示了相应的提示词。

数据集	方法	F1 分数	API 调用	词元		成本（¥）		
				In	Out	In	Out	总计
TPC-H	端到端	**0.90**	**36**	**0.0351**	**0.0006**	**0.088**	**0.006**	**0.094**
	LLM-FK	0.78	60	0.1221	0.0011	0.305	0.011	0.316
Northwind	端到端	0.92	**66**	**0.0617**	**0.0010**	**0.154**	**0.010**	**0.164**
	LLM-FK	**1.00**	71	0.1312	0.0012	0.328	0.012	0.339
TPC-E	端到端	0.58	528	**0.4496**	0.0066	**1.124**	0.066	**1.190**
	LLM-FK	**0.92**	**245**	0.5342	**0.0031**	1.335	**0.032**	1.367
AdventureWorks	端到端	0.52	2485	**2.6313**	0.0346	**6.578**	0.346	**6.924**
	LLM-FK	**0.92**	**2142**	5.5458	**0.0224**	13.864	**0.223**	14.087
MusicBrainz	端到端	0.75	28203	25.9330	0.3653	64.832	3.653	68.485
	LLM-FK	**0.96**	**2588**	**6.3064**	**0.0264**	**15.766**	**0.264**	**16.029**

表11：不同数据集上端到端基线与 LLM-FK 的总体计算成本对比。

数据集	智能体	API 调用	词元		成本（¥）		
			In	Out	In	Out	总计
TPC-H	分析器	8	0.0149	0.0001	0.037	0.001	0.038
	解释器	1	0.0003	0.0005	0.001	0.005	0.006
	优化器	**49**	0.1004	0.0005	0.251	0.005	**0.256**
	验证器	2	0.0064	0.0000	0.016	0.000	0.017
Northwind	分析器	9	0.0067	0.0001	0.017	0.001	0.018
	解释器	1	0.0004	0.0005	0.001	0.005	0.005
	优化器	**60**	0.1211	0.0006	0.303	0.006	**0.309**
	验证器	1	0.0030	0.0000	0.007	0.000	0.008
TPC-E	分析器	20	0.0140	0.0002	0.035	0.002	0.037
	解释器	1	0.0004	0.0006	0.001	0.006	0.007
	优化器	**194**	0.4165	0.0019	1.041	0.019	**1.061**
	验证器	30	0.1032	0.0004	0.258	0.004	0.262
AdventureWorks	分析器	60	0.0433	0.0006	0.108	0.006	0.114
	解释器	1	0.0007	0.0009	0.002	0.009	0.011
	优化器	**2057**	5.3956	0.0205	13.489	0.205	**13.694**
	验证器	24	0.1061	0.0004	0.265	0.004	0.269
MusicBrainz	分析器	140	0.0967	0.0014	0.242	0.014	0.256
	解释器	1	0.0014	0.0011	0.004	0.011	0.015
	优化器	**2355**	5.8799	0.0228	14.700	0.228	**14.927**
	验证器	92	0.3284	0.0011	0.821	0.011	0.832

表 12：LLM-FK 框架中各内部代理计算开销的详细 breakdown rk.

算法 2 LLM-FK 的总体流程 要求：数据库 $\mathcal{D}$，指令集 $\mathcal{I} = \{\mathcal{I}_{\mathcal{A}_p}, \mathcal{I}_{\mathcal{A}_i}, \mathcal{I}_{\mathcal{A}_r}, \mathcal{I}_{\mathcal{A}_v}\}$确保：外键集合 Φ1: // 阶段 1：分析器（$\mathcal{A}_p$）——分析与唯一键驱动的模式分解 2: $\mathcal{J}_{raw} \leftarrow$ ExtractINDs($\mathcal{D}$) {分析器调用工具提取原始包含依赖} 3: $\mathcal{U} \leftarrow$ ExtractMinUCCs($\mathcal{D}$) {分析器调用工具提取最小唯一列组合} 4: $\mathcal{J} \leftarrow$ FilterByType($\mathcal{J}_{raw}$) {分析器调用工具，根据数据类型筛选包含依赖} 5: $\mathcal{K} \leftarrow \emptyset$ {初始化已选唯一键集合} 6: $\mathcal{T}_{ref} \leftarrow$ GetReferencedTables($\mathcal{J}$) 7: 对每个表 $T \in \mathcal{T}_{ref}$ 执行 8: $k \leftarrow \pi(T, \mathcal{U}_T, \mathcal{I}_{\mathcal{A}_p})$ {分析器选择最合理的被引用唯一键} 9: $\mathcal{K} \leftarrow \mathcal{K} \cup \{k\}$10: 结束循环 11: $\Psi_{cand} \leftarrow \{(c_f, c_p) \in \mathcal{J} \mid c_p \in \mathcal{K}\}$ {分析器使用已选唯一键剪枝 $\mathcal{J}$} 12: // 阶段 2：解释器（$\mathcal{A}_i$）——自增强领域知识注入 13: $\mathcal{N}_{\Psi_{cand}} \leftarrow$ ExtractTableNames(Ψ_{cand}) {解释器调用工具提取表名} 14: $\mathcal{K}_{dom} \leftarrow \pi(\mathcal{N}_{\Psi_{cand}}, \mathcal{I}_{\mathcal{A}_i})$ {解释器构建自增强领域知识} 15: // 阶段 3：优化器（$\mathcal{A}_r$）——多视角思维链推理 16: $\Psi_{pos} \leftarrow \emptyset$ {初始化局部外键结果集} 17: 并行处理每个候选项 $\psi \in \Psi_{cand}$ 18: $S_\psi \leftarrow$ GetStructuralRepresentative($\psi, \mathcal{D}$) {优化器在多视角思维链的指导下构造 SQL 查询，以推导结构表示} 19: $p_\psi \leftarrow \pi(\psi, S_\psi, \mathcal{K}_{dom}, \mathcal{I}_{\mathcal{A}_r})$ {优化器执行二元验证（$p_\psi \in \{0, 1\}$）} 20: 如果 $p_\psi = 1$则 21: $\Psi_{pos} \leftarrow \Psi_{pos} \cup \{\psi\}$22: 结束条件判断 23: 结束循环 24: // 阶段 4：验证器（$\mathcal{A}_v$）——整体冲突消解 25: $\mathcal{G} \leftarrow$ ConstructGraph(Ψ_{pos}) {验证器构建模式拓扑图} 26: // 阶段 4.1：优先解决多重引用 27: $\mathbf{C}_{multi} \leftarrow$ DetectMultiRefConflicts($\mathcal{G}$) {验证器调用工具检测多重引用} 28: 对每个冲突集 $\Psi_{conflict} \in \mathbf{C}_{multi}$ 执行 29: $S_{\Psi_{conflict}} \leftarrow$ GetStructuralRepresentative($\Psi_{conflict}, \mathcal{D}$) {验证器在多视角思维链的指导下构造 SQL 查询，以推导结构表示} 30: $\Psi_{resolved} \leftarrow \pi(\Psi_{conflict}, S_{\Psi_{conflict}}, \mathcal{K}_{dom}, \mathcal{I}_{\mathcal{A}_v})$ {验证器通过保留最合理的候选外键来消解冲突} 31: $\Psi_{pos} \leftarrow (\Psi_{pos} \setminus \Psi_{conflict}) \cup \Psi_{resolved}$32: 结束循环 33: // 阶段 4.2：迭代解决循环引用 34: 当 TRUE 时执行 35: $\mathcal{G} \leftarrow$ ConstructGraph(Ψ_{pos}) {验证器使用更新后的边构建图} 36: $\mathbf{C}_{cycle} \leftarrow$ DetectCyclicConflicts($\mathcal{G}$) {验证器调用工具检测循环引用} 37: 如果 $\mathbf{C}_{cycle} = \emptyset$则 38: break {如果不再存在环，则终止} 39: 结束条件判断 40: 选择最短冲突集 $\Psi_{conflict} \in \mathbf{C}^{cycle}$ {验证器优先处理最短环以消解冲突} 41: $S_{\Psi_{conflict}} \leftarrow$ GetStructuralRepresentative($\Psi_{conflict}, \mathcal{D}$) 42: $\Psi_{resolved} \leftarrow \pi(\Psi_{conflict}, S_{\Psi_{conflict}}, \mathcal{K}_{dom}, \mathcal{I}_{\mathcal{A}_v})$ {验证器移除最不合理的候选外键以打破该环} 43: $\Psi_{pos} \leftarrow (\Psi_{pos} \setminus \Psi_{conflict}) \cup \Psi_{resolved}$44: 结束循环 45: $\Phi \leftarrow \Psi_{pos}$46: 返回 Φ

[Role-play]
你是一名专门从事关系数据库设计、数据血缘推理和语义完整性分析的数据库架构分析师。
[Task Description]
你的任务是对候选最小唯一列组合（最小唯一列组合）进行推理，以识别最可能的被引用唯一键
针对给定的表。表`教师`包含若干候选唯一键。根据给定的选择标准，选出唯一最
合适的最小唯一列组合，以作为外键的被引用唯一键。
[Sample Data]
表的前五行如下：
编号	姓名	地址_编码
1	Alice	001
2	Bob	002
3	Carol	005
4	Dave	003
5	Eve	004
[Selection Criterion]
1. 唯一性有效性：候选项必须能够唯一标识一个元组（所有给定候选项均满足此条件）。
2. 列序：在表定义中位置较靠前的列优先。
3. 命名约定：优先选择包含典型标识符（例如“id”“key”“code”）的列名。
4. 类型适用性：数据类型通常为整数或字符串，且符合标准键定义。
5. 值简洁性：数据值应简洁且人类可读（文本长度较短）。
6. 最小构成：与复杂的复合键相比，优先选择组成列（原子键）较少的候选项。
7. 语义对齐：键必须根据表的语义上下文，在逻辑上表示该实体。
[Candidate Solutions]
1. 编号
2. 姓名
3. 地址_代码
[Response Format]
你的回复应采用以下格式：选择：[被引用唯一键]

[Conclusion]
选择：编号

图6：用于唯一键选择的档案提示。

[Role-play]
你是一名数据库架构分析师，专攻关系数据库设计、数据血缘推理和语义完整性分析。
[Task Description]
该任务涉及处理来自关系数据库的表名列表，并通过语义分析重建外键关系。作为
一项前提，第一步需要理解应用领域，并掌握外键所必需的实体知识
识别，为后续流程奠定基础。
[Database Tables]
需要分析以下数据库表：`教师`, `学生`, `教师_扩展`。
[Analysis Steps]
请对上述表进行深入的语义分析并输出摘要。请遵循以下步骤：
1. 领域推断：根据表名，推断该数据库可能所属的应用领域（例如：电子商务、人力资源、企业资源计划等）。
2. 实体定义：对于列表中的每张表，确定它所代表的“实体”，并总结为了识别所需的实体知识
外键的语义。
[Response Format]
你的回复应采用以下格式：
领域摘要：[你的分析]
外键识别的实体知识摘要：[你的摘要]

[Conclusion]
领域摘要：教育行政 / 学术管理系统。
外键识别的实体知识摘要：
教师：核心教师实体。作为主要实体，teacher_extra 必须引用该实体。
学生：核心学习者实体。预计与教师之间存在潜在的语义关联（表示顾问/导师关系）。
teacher_extra：扩展表（垂直分区）。存储补充元数据。从语义上看，它意味着对
教师表，具有相同的逻辑粒度。

图7：P　用于领域知识生成的解释器提示　成。

[Role-play]
你是一名数据库架构分析师，专长于关系数据库设计、数据血缘推理和语义完整性分析。
[Task Description]
外键（FK）表示一种引用约束，其中一张表（引用表）中的值对应于
另一张表（被引用表）中的主键值或唯一键值，从而确保数据一致性。你的任务是根据提供的信息，判断列 A 是否为引用
列 B 的外键。
[Domain Knowledge]
领域摘要：教育行政 / 学术管理系统。
用于外键识别的实体知识摘要：
教师：核心教师实体。作为主要实体，teacher_扩展必须引用该实体。
学生：核心学习者实体。预计与教师存在潜在语义关联（表示顾问/导师关系）。
teacher_扩展：扩展表（垂直分区）。存储补充元数据。从语义上看，它意味着对
教师表，具有相同的逻辑粒度。
[Column A Information]
- 表名：`学生` - 列名：`学习_在` - 序号位置：2 - 数据类型：整数 - 平均长度：1
- 不同值（基数）：2 - 总行数：5 - 基数比（基数 / 总行数）：0.4 - 最小值：1 - 最大值：2
- 样本数据：
该表的前五行如下：
编号	姓名	学习_在	地址_编码
1	Elaine	2	002
2	Bonnie	2	001
3	Van	1	003
4	Kelly	1	005
5	Lucas	1	004
[Column B Information]
- 表名：`教师` - 列名：`编号` - 序号位置：0 - 数据类型：整数 - 平均长度：1
- 不同值（基数）：5 - 总行数：5 - 基数比（基数 / 总行数）：1 - 最小值：1 - 最大值：5
- 样本数据：
表的前五行如下：
编号	姓名	地址_代码
1	Alice	001
2	Bob	002
3	Carol	005
4	Dave	003
5	Eve	004
[Chain-of-Thought]
1. 语义分析：
  - 步骤 1：解读表所表示的现实世界实体。`student.study_under` 在语义上是否意味着引用了`教师`？
  - 步骤 2：确定该关系对于外键而言是否遵循逻辑上的“是一种”“拥有”或“分配给”语义。
2. 句法分析：
  - 分析列名/表名之间的相似度。相似度越高，表示存在关系的可能性越大。
  - 检查标准命名约定（例如 `_id`、 `_key`、 `_no` 等后缀）。
3. 统计分析（包含依赖已预先验证）：
  - 覆盖率：外键应引用被引用键值中的足够部分；在此情况下，覆盖率为 0.4。
  - 长度相似度：两列的平均存储长度应相同或高度相似（1 与 1）。
  - 范围逻辑：外键引用应均匀分布在被引用键域中，而不应局限于一个很小的
子集；在这种情况下，超范围率为 0.6。
  - 表大小比率：外键通常存在于等于或大于被引用表的表中（一对多关系）。此处，行
数比率（`学生` / `教师`）为 1.0。
[Response Format]
你的回答应采用此格式：判定：[真 | 假]

[Conclusion]
判定：是

图 8：用于验证每个外键候选对的优化器提示。

[Role-play]
你是一名数据库架构分析师，专长于关系数据库设计、数据血缘推理和语义完整性分析。
[Task Description]
解决“多重引用冲突”，即源表中的单个列与多个
目标表中的候选列存在潜在外键关系。有效的外键设计规定，源列应恰好引用一个目标列。
根据所提供的信息，选择唯一最合适的外键关系。
[Domain Knowledge]
领域摘要：教育行政 / 学术管理系统。
用于外键识别的实体知识摘要：
教师：核心教师实体。作为主要实体，teacher_extra 必须引用该实体。
学生：核心学习者实体。预期与教师存在潜在语义关联（表示顾问/导师关系）。
teacher_extra：扩展表（垂直分区）。存储补充元数据。从语义上看，它意味着对该实体存在严格的一对一依赖
逻辑粒度相同的教师表。
[Referencing Column Information]
- 表名：`学生` - 列名：`学习_于` - 序号位置：2 - 数据类型：整数 - 平均长度：1
- 不同值（基数）：2 - 总行数：5 - 基数比（基数 / 总行数）：0.4 - 最小值：1 - 最大值：2
- 样本数据：
该表的前五行如下：
编号	姓名	研究_在.....之下	地址_代码
1	Elaine	2	002
2	Bonnie	2	001
3	Van	1	003
4	Kelly	1	005
5	Lucas	1	004
[Referenced Columns Information]
- 表名：`教师` - 列名：`编号` - 序号位置：0 - 数据类型：整数 - 平均长度：1
- 不同值（基数）：5 - 总行数：5 - 基数比（基数 / 总行数）：1 - 最小值：1 - 最大值：5
- 样本数据：
该表的前五行如下：
编号	姓名	地址_代码
1	Alice	001
2	Bob	002
3	Carol	005
4	Dave	003
5	Eve	004
- 表名：`teacher_extra` - 列名：`id` - 序号位置：0 - 数据类型：整数 - 平均长度：1
- 不同值（基数）：5 - 总行数：5 - 基数比（基数 / 总行数）：1 - 最小值：1 - 最大值：5
- 样本数据：
该表的前五行如下：
编号	学科
1	数学
2	计算机科学
3	文学
4	物理学
5	世界历史
[Chain-of-Thought]
1. 语义分析：
  - 步骤 1：解读表所表示的实体。
  - 步骤 2：确定该关系对于外键而言是否符合逻辑上的“是一种”、“拥有”或“分配给”语义。
2. 句法分析：
  - 分析列名/表名之间的相似性。相似性越高，表示存在关系的可能性越大。
  - 检查是否符合标准命名约定（例如后缀 `_id`、 `_key`、 `_no`）。
3. 统计分析（已预先验证包含依赖）：
  - 覆盖率：外键应引用足够比例的被引用键值；覆盖率分别为 0.4、0.4。
  - 长度相似度：两列的平均存储长度应相同或高度相似（1 vs 1 vs 1）。
  - 范围逻辑：外键引用应均匀分布在被引用键域中，而不是局限于一个很小的
子集；越界比例分别为 0.6、0.6。
  - 表大小比率：外键通常存在于与被引用表大小相等或更大的表中（一对多关系）；行数
比率为 1.0、1.0。
[Response Format]
请按以下格式作答：选择：[最合适的被引用列]

[Conclusion]
选择：教师.id

图 9：用于多重引用解析的验证器提示。

[Role-play]
你是一名数据库架构分析师，专长于关系数据库设计、数据血缘推理和语义完整性分析。
[Task Description]
解决“循环引用冲突”，即潜在外键构成闭环的情况。根据提供的信息，选出
该循环中最不合理（最弱）的引用关系，并将其移除以打破循环。
[Domain Knowledge]
领域摘要：教育行政 / 学术管理系统。
用于外键识别的实体知识摘要：
教师：核心教师实体。作为教师_扩展必须引用的主实体。
学生：核心学习者实体。预期与教师保持潜在的语义关联（表示顾问/导师）。
教师_扩展：扩展表（垂直分区）。存储补充元数据。从语义上看，它表示对.....存在严格的一对一依赖。
教师表，共享相同的逻辑粒度。
[Conflict Description]
1. 教师.id → 教师.extra.id
2. 教师.extra.id → 教师.id
[Columns Information]
- 表名：`教师` - 列名：`id` - 序号位置：0 - 数据类型：整数 - 平均长度：1
- 不同值（基数）：5 - 总行数：5 - 基数比（基数 / 总行数）：1 - 最小值：1 - 最大值：5
- 样本数据：
该表的前五行如下：
编号	姓名	地址_代码
1	Alice	001
2	Bob	002
3	Carol	005
4	Dave	003
5	Eve	004
- 表名：`教师_扩展` - 列名：`编号` - 序号位置：0 - 数据类型：整数 - 平均长度：1
- 不同值（基数）：5 - 总行数：5 - 基数比（基数 / 总行数）：1 - 最小值：1 - 最大值：5
- 样本数据：
该表的前五行如下：
编号	学科
1	数学
2	计算机科学
3	文学
4	物理学
5	世界历史
[Chain-of-Thought]
1. 语义分析：
  - 步骤 1：解释表所表示的实体。
  - 步骤 2：确定该关系对于外键而言是否遵循逻辑上的“是一种”、“拥有”或“分配给”语义。
2. 句法分析：
  - 分析列名/表名之间的相似度。相似度越高，存在关联的可能性就越大。
  - 检查是否符合标准命名惯例（例如使用 `_id`、 `_key`、 `_no` 等后缀）。
3. 统计分析（已预先验证包含依赖）：
  - 覆盖率：外键应引用被引用键值中的足够比例；覆盖率分别为 1.0 和 1.0。
  - 长度相似度：两列的平均存储长度应相同或高度相似（1 与 1）、（1 与 1）。
  - 范围逻辑：外键应均匀分布在被引用键域中，而不是局限于一小部分
；越界比例为 0、0。
  - 表大小比率：外键通常存在于大小等于或大于被引用表的表中（一对多关系）；行数
比率为 1.0、1.0。
[Response Format]
你的回复应采用以下格式：选择：[最不合理的引用关系]

[Conclusion]
选择：教师.id → 教师.扩展.id

图 10：用于循环引用消解的验证器提示。

[Role-play]
你是一名数据库架构分析师，专攻关系数据库设计、数据血缘推理和语义完整性分析。
[Task Description]
严格依据所提供的信息，识别以下两个表之间的外键关系。对于识别出的每个外键，
输出一条列表项。以表 A 引用表 B 的情形为例，请使用以下确切格式：
[<表 A 名称>, <表 A 中的外键列名>, <表 B 名称>, <表 B 中被引用列的名称>]。
[Table Information]
- 表名：student
该表的前五行如下：
编号	姓名	学习_下方	地址_编码
1	Elaine	2	002
2	Bonnie	2	001
3	Van	1	003
4	Kelly	1	005
5	Lucas	1	004

- 表名：教师
该表的前五行如下：
编号	姓名	地址_代码
1	Alice	001
2	Bob	002
3	Carol	005
4	Dave	003
5	Eve	004
[Response Format]
你的回答应采用以下格式：外键：[已识别的外键]
输出示例：
1. [['订单'、'客户_编号'、'客户'、'编号']]
2. [[]]

图 11：端到端基线的提示。

[End-to-End Prompt]

[Examples]
示例 1：单外键关系
输入：
- 订单（前五行）：
订单_编号	日期	用户_编号	金额
101	2023-01-01	5	100.0
102	2023-01-02	8	50.5
103	2023-01-03	5	120.0
104	2023-01-04	6	200.0
105	2023-01-05	7	80.0
- 用户（前五行）：
用户标识符	用户_姓名	电子邮件
5	Alice	alice@example.com
6	Bob	bob@test.com
7	Charlie	charlie@corp.com
8	David	david@mail.com
9	Eve	eve@web.com
输出：[['订单', '用户_编号', '用户', '用户标识符']]
示例 2：无外键关系
输入：
- 日志（前五行）：
日志_编号	时间戳	消息
1	12:00:01	错误 404
2	12:00:05	正常 200
3	12:01:10	正常 200
4	12:02:45	错误 500
5	12:05:00	正常 200

- 员工（前五行）：
员工_编号	姓名	部门
99	Frank	人力资源
100	Grace	信息技术
101	Heidi	销售
102	Ivan	市场营销
103	Judy	信息技术
输出：[[]]
示例 3：多个外键关系
输入：
- 航班（前五行）：
航班_编号	出发_代码	到达_代码	时间
AA123	JFK	LHR	10：00
BA001	LHR	JFK	14:00
深度学习456	LAX	HND	08:30
JL002	HND	SFO	16:45
UA789	SFO	LAX	12:15
- 机场（前五行）：
国际航空运输协会_代码	城市	国家
JFK	纽约	美国
LHR	伦敦	英国
LAX	洛杉矶	美国
HND	东京	日本
SFO	旧金山	美国
输出:[['航班', 'dept_code', '机场', '国际航空运输协会_code'],
['航班', 'arr_code', '机场', 'iata_code']]

示例结束。

图 12：少样本基线的提示。

[End-to-End Prompt]

[Examples]
示例 1：单外键关系
输入：
- 订单（前五行）：
订单_编号	日期	用户_编号	金额
101	2023-01-01	5	100.0
102	2023-01-02	8	50.5
103	2023-01-03	5	120.0
104	2023-01-04	6	200.0
105	2023-01-05	7	80.0
- 用户（前五行）：
用户标识符	用户_姓名	电子邮件
5	Alice	alice@example.com
6	Bob	bob@test.com
7	Charlie	charlie@corp.com
8	David	david@mail.com
9	Eve	eve@web.com
输出：[['订单'、'用户_编号'、'用户'、'用户标识符']]
解释：从语义上看，'订单'表记录了交易由特定实体完成。'用户_编号'这一列明确指出了它所指向的实体（“用户”）。将其与“用户”表进行比较，“用户标识符”充当用户实体的主标识符。由于“用户_编号”在订单与用户中“用户标识符”的语义相符，它是一个外键，描述“哪位用户下达了此订单”。
示例 2：无外键关系
输入：
- 日志（前五行）：
日志_编号	时间戳	消息
1	12:00:01	错误 404
2	12:00:05	正常 200
3	12:01:10	正常 200
4	12:02:45	错误 500
5	12:05:00	正常 200
- 员工（前五行）：
员工_编号	姓名	部门
99	Frank	人力资源
100	Grace	信息技术
101	Heidi	销售
102	Ivan	市场营销
103	Judy	信息技术
输出：[[]]
解释：从语义上讲，'server_日志'用于跟踪系统事件，而“employees”用于记录人力资源信息。二者之间不存在合乎语义逻辑的系统日志与员工记录之间不存在逻辑上的关联关系。尽管“log_id”和“emp_id”都是唯一数值标识符，但它们的列名表明，它们代表的是完全不同的现实世界实体（日志条目与人员）。因此，不存在外键。

示例 3：多个外键关系
输入：
- 航班（前五行）：
航班_编号	出发_代码	到达_代码	时间
AA123	JFK	LHR	10:00
BA001	LHR	JFK	14:00
DL456	LAX	HND	08:30
JL002	HND	SFO	16:45
UA789	SFO	LAX	12:15
- 机场（前五行）：
iata_代码	城市	国家
JFK	纽约	美国
LHR	伦敦	英国
LAX	洛杉矶	美国
HND	东京	日本
SFO	旧金山	美国
输出：[['航班', 'dept_code', '机场', 'iata_code'], ['航班', 'arr_code', '机场', 'iata_code']]
解释：从语义上讲，一次航班涉及两个地点：一个出发地点和一个到达地点。这些列'dept_code'（出发）和 'arr_code'（到达）在 '航班' 表中都描述了机场实体。'机场'表包含这些实体的主列表，由 'iata_code' 标识。由于'航班' 中的两列均使用标准的国际航空运输协会位置代码可在 '机场' 表中找到，它们都引用'机场.iata_code'。

示例结束。
[Chain-of-Thought]
现在，请分析所提供的具体信息。请逐步思考，先给出你的解释，然后提供 Python 列表代码块。

图13：思维链基线的提示词。