DIN-SQL：文本到 SQL 的分解上下文学习与自我修正

摘要

在诸如 Spider 的数据集上，使用大型语言模型（LLMs）进行微调的模型与提示方法之间的性能差距目前仍然很大。为了提高 LLM 在推理过程中的性能，我们研究了如何通过分解任务为更小的子任务来有效实现这一目标。具体来说，我们将生成问题分解为子问题，并将这些子问题的解决方案馈送到 LLM 中，证明这是一种有效的方法，可以显著提高它们的性能。我们在三个 LLM 上的实验表明，这种方法始终可以将它们的简单少量样本性能提高约 10％，将 LLM 的准确性推向或超过 SOTA。在 Spider 的保留测试集上，SOTA 的执行准确率是 79.9，而使用我们的方法的新 SOTA 到目前为止是 85.3。我们的基于上下文的学习方法比许多经过大量微调的模型至少高出 5%。此外，在 BIRD 基准上评估时，我们的方法实现了 55.9%的执行准确率，并在它的保留测试集上创造了新的 SOTA。

1. 简介

自然语言接口的目标是使终端用户更容易访问关系数据库中的数据。例如，给定语句“查找工资高于其经理的员工”，以及员工和经理表的模式，我们可能希望在 SQL 中生成一个查询以从数据库中检索这些员工。在过去二十年里，该领域的研究已经通过几个阶段取得进展，早期系统具有特定领域性，支持受控自然语言（Popescu 等人，2003 年、2004 年；Li 等人，2007 年；Li 和 Jagadish，2014 年）或基于规则的方法（Stratica 等人，2005 年），而最近的系统则使用监督模型，在各种领域和数据集上进行训练，从而提供更大的领域独立性（钟等，2017 年；余等，2018 年）。最近，人们开始利用大型文本和代码存储库对深度神经网络进行训练（董和 Lapata，2016 年；Devlin 等人，2018 年）。

这一进展的最新发展是在零样本和少量样本提示下使用大型语言模型 (LLMs)[Rajkumar 等人，2022 年；刘等人，2023 年]。已证明 LLM 在仅使用很少示例且无需微调的情况下提供了强大的基线[陈等人，2021 年；Brown 等人，2020 年；刘等人，2023 年]。然而，与精心设计和微调的模型相比，这些模型在常用的基准测试（例如 Spider）上表现不佳。表 1 显示了两个最新的 LLM，CodeX 和 GPT-4，在 Spider 数据集的开发集上的性能。尽管表现出色，但 LLM 相对于现有方法[Scholak 等人，2021 年；李等人，2023 年]尤其落后于中等和复杂查询。本文所研究的问题

这是这些语言模型失败的地方，如果它们面临的一些问题可以得到缓解，以推动性能达到或超过微调最先进的模型。

提示比使用预训练或微调等传统方法具有多个优势。主要好处是，语言模型可以在不需要大量特定任务训练数据的情况下执行预测任务。从头开始训练模型或对其进行微调是一项资源密集型过程，通常需要大量的训练样本和计算资源，这些资源可能无法获得。此外，研究表明，少量提示可以优于以前的最先进的方法，并且即使在有限的训练示例中也可以实现高精度（Brown 等人，2020 年；Wei 等人，2022 年 b）。

最近有证据表明，通过使用诸如链式思维、最少到最多和分解等方法，可以提高 LLM 在更复杂任务（如数学单词问题、复合导航步骤）上的性能。在这些方法中，一个任务被分解为多个步骤，并且中间结果用于生成最终答案。与包含明确步骤或操作的代数表达式不

同，由于语言的声明性结构以及查询子句之间的复杂关系，分解复杂的 SQL 查询可能是一项更具挑战性的任务。

在本文中，我们提出了一种基于少量提示的新方法，将自然语言文本到 SQL（称为文本到 SQL）的任务分解为多个子任务。过去使用 LLM 进行文本到 SQL 提示的工作仅在零样本设置下进行了评估[Rajkumar et al.，2022，Liu et al.，2023a]。然而，零样提示只为大多数任务提供了 LLM 的潜在能力的下限[Zhang et al.，2022，Kojima et al.，2022，Wei et al.，2022b，2021，Brown et al.，2020]。我们证明了我们的方法比少量提示的方法优越得多。我们也

微调方法

方法 执行精度

RED-SQL 3B+ NatSQL 84.5

李等人，2023 年

在接下来的内容中，

将出现大量的文本。

在这个文本中，将有

一个非常长的段落。

这段文字的长度是

这个段落非常长。

这个文本很短。

在下面的文本中，

笔记：

这个文本没有标点符号。

在这个文本中

有一个很长的句子。

在下面的文本文本中，

有

一个包含很多逗号的句子。

笔记：

在下面的文本中，

在这段文字中，有一个

包含大量句号的句子。

笔记：

在下面的文本中，

在这段文字中，有一个

笔记：

在下面的文本中，
仅推理的方法

Method	Execution accuracy
Zero-shot GPT-4 (Ours)	64.9
Few-shot GPT-4 (Ours)	67.4
Zero-shot CodeX [Rajkumar et al., 2022]	55.1
Few-shot CodeX (Ours)	61.5

Table 1: Zero-shot and few-shot prompting compared to fine-tuned approaches on the dev set of Spider

表 1：与在 Spider 开发集上微调的方法相比，零样本和少样本提示的效果
在两个跨域挑战基准测试集 Spider 和 BIRD 上比较我们的方法与先前的方法。对于 Spider 数据集，我们使用 Zhong 等人（2020 年）的两种官方评估指标：执行准确率和精确集合匹配准确率。我们使用 Chen 等人（2021 年）的 CodeX 系列的两个变体 Davinci 和 Cushman，以及 GPT-4 模型来提示。在 Spider 的留出测试集上，我们的方法分别使用 GPT-4 和 CodeX Davinci 模型取得了 85.3% 和 78.2% 的执行准确率，以及 60% 和 57% 的精确集合匹配准确率。执行准确率和精确匹配准确率之间的巨大差距是由于我们方法的 Few-Shot in-context 性质。预训练和微调方法更有可能生成具有更高精确集合匹配准确率的 SQL 查询，因为这些模型在训练过程中已经看到了许多遵循测试集中查询组成风格（两组中的查询通常都是由同一人编写的）的例子。在我们的工作之前，测试集上的最优结果是 Li 等人（2023a）的 79.9% 执行准确率和 Li 等人（2023b）的 74% 精确集合匹配准确率，而我们的方法在执行准确率方面创造了新的基准。在 BIRD 基准上，当使用 GPT-4 时，我们的方法在留出测试集上实现了新的最优结果，达到了 55.9% 的执行准确率，在开发集上达到了 50.72%，此外，通过引入本基准中介绍的有效性得分，我们的方法优于 GPT-4 基线，在开发集上显示出 9% 的改进。这突显了我们方法的有效性。
我们的贡献可以总结如下：（1）通过任务分解提高基于 LLM 的文本到 SQL 模型的性能，（2）引入针对任务复杂度量身定制的自适应提示策略，（3）在提示上下文中解决模

式链接挑战，以及（4）使用 LLM 进行自我纠正。要复制报告的结果，请访问我们的 GitHub 存储库 1 以获取提示、结果和代码。

2 相关工作

序列到序列模型 [Sutskever et al.，2014 年] 在包括文本到 SQL 的代码生成任务中表现出巨大的潜力。其核心思想是联合编码给定的自然语言查询和数据库模式，并利用解码器来预测目标 SQL。

在编码器端，使用双向 LSTM 在 IRNet [Graves 和 Graves, 2012] 中为问题和数据库模式学习表示，使用卷积神经网络在 RYANSQL [Choi 等人，2021] 中为问题和数据库模式学习表示，使用诸如 BERT 在 SQLova [Hwang 等人，2019] 中预训练的语言模型以及在 RATSQL [Wang 等人，2019]、SADGA [Cai 等人，2021] 和 LGESQL [Cao 等人，2021] 中使用的图神经网络。Gan 等人。[2021] 提出一种中间表示来弥合自然语言查询和 SQL 查询之间的差距。还有针对表格语言模型的工作，这些模型对表和文本进行编码，例如 TaBERT [Yin 等人，2020]、TaPas [Herzig 等人，2020] 和 Grappa [Yu 等人，2020]。

解码器端的方法可以分为基于草图的插槽填充方法和生成式方法。[Qin 等人，2022 年] 基于草图的方法将问题分解为多个子问题插槽预测，并对要生成的 SQL 查询中的插槽进行聚合预测。[Hwang 等人，2019 年；Xu 等人，2017 年；Hui 等人，2021 年] 这些方法的一个缺点是它们不能推广到不遵循预定义模板的查询。[Guo 等人，2019 年；Wang 等人，2019 年；Cao 等人，2021 年；Huang 等人，2021 年] 生成式方法将 SQL 查询解释为抽象语法树。

与预训练和微调模型不同，Rajkumar 等人。 [2022] 和刘等人。 [2023a] 在 Spider 数据集上使用不同的提示对 LLM 进行了零样本提示能力评估。提示技术还用于表理解、表格推理和表格到文本生成等任务[郭等，2023，陈，2022]，并报告了一些显著的结果，只需在提示中提供一小部分示例即可使用 LLM。

3 少样本错误分析

为了更好地理解少样本条件下的 LLM 失败情况，我们在 Spider 数据集的训练集中随机选择了来自不同数据库的 500 个查询，排除了我们提示中使用的所有数据库。我们搜索产生与黄金查询不同的结果的查询，因此导致执行精度失败。我们手动检查这些失败，并根据图 1 和接下来讨论的内容将其分为六类。

链接方案 这个类别包含最多的查询失败，包括模型无法识别的问题中提到的列名、表名或实体的情况。在某些情况下，查询需要聚合函数，但选择了匹配的列名。例如，“所有体育场的平均容量和最大容量是多少？”问题数据库架构包括一个名为“average”的列，该模型选择的是该列而不是容量列的平均值。

JOIN 这是第二大类，包括需要 JOIN 但模型无法识别所有所需的表或正确外键以连接表的查询。

GROUP BY 本类包括 SQL 语句需要使用 GROUP BY 子句，但模型要么没有识别到需要分组，要么错误地使用了用于分组结果的列。

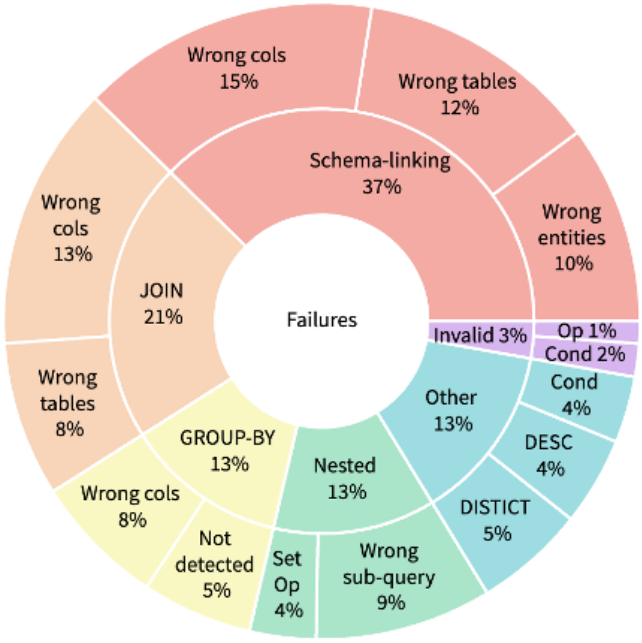


Figure 1: Statistics of simple few-shot failures using CodeX Davinci (Op refers to operators, Cond refers to conditions, and cols refers to columns)

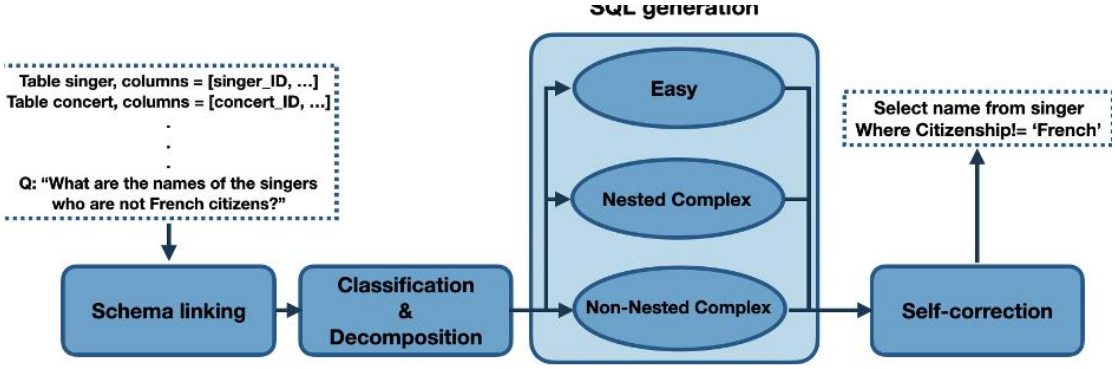


Figure 2: An overview of the proposed methodology including all four modules

包含嵌套查询和集合运算符的查询

对于这一类，黄金查询使用了嵌套或集合操作，但模型没有识别出嵌套结构，或者无法检测到正确的嵌套或集合操作。

无效的 SQL 生成的一小部分 SQL 语句存在语法错误，无法执行。

杂项 这个类别包括了所有不适用于上述任何一个类别的案例。例子包括包含多余谓词、缺少谓词，或者有缺失或多余的 DISTINCT 或 DESC 关键字的 SQL 查询。这个类别也包括了那些 WHERE 子句中没有条件，或者是查询中有冗余聚合函数的案例。

4 方法

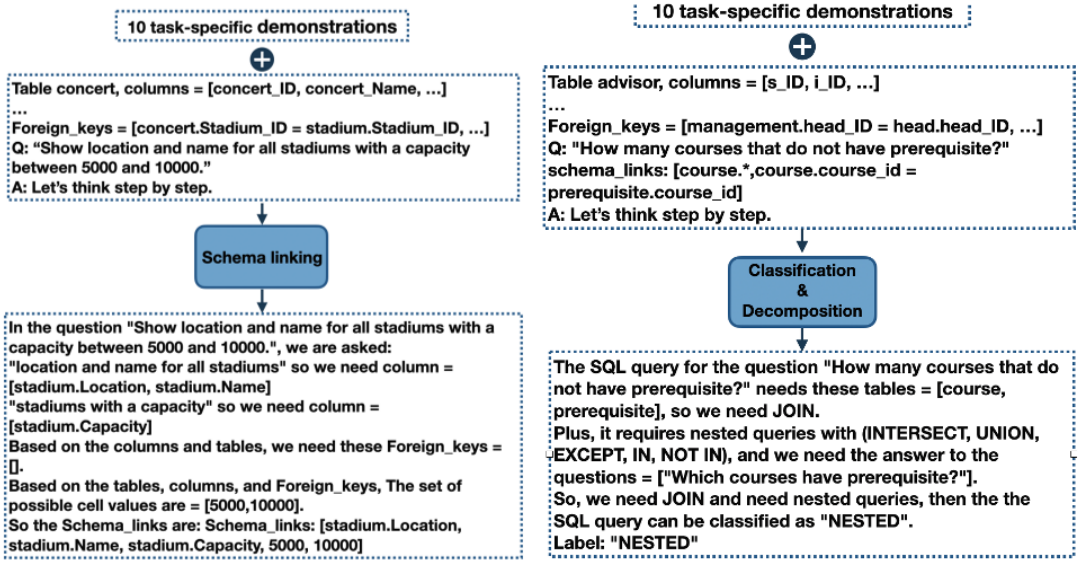
尽管在零样本设置中有所改进，但少量样本模型在更复杂的查询上表现不佳，包括那些不太简单的模式连接以及使用多个连接或嵌套结构的查询，如第 3 节所述。

我们解决这些挑战的方法是将问题分解为较小的子问题，然后逐个解决每个子问题，并使用这些解决方案来构建原始问题的解决方案。类似的方法（例如思维链提示 [Wei 等人，2022 年] 和从少到多提示 [Zhou 等人，2022 年]）已经被用来提高在可以分解为多个步骤的任务上的 LLM 性能，例如数学单词问题和复合泛化 [Cobbe 等人，2021 年；Lake 和 Baroni，2018 年]。与这些领域不同的是，任务具有过程结构，其中一个步骤直接连接到下一个步骤，SQL 查询中的大多数部分都是声明性的，可能的步骤及其边界不那么明显。然而，编写 SQL 查询的思想过程可以分为（1）检测与查询相关的数据库表和列，（2）识别更复杂的查询的一般查询结构（例如分组、嵌套、多个连接、集合运算等），（3）如果可以识别的话，制定任何程序性子组件，以及（4）根据子问题的解决方案编写最终查询。

基于这种思维方式，我们提出的文本到 SQL 任务分解方法包括四个模块（如图 2 所示）：(1)模式关联，(2)查询分类与分解，(3)SQL 生成，以及(4)自我纠正，在下面的小节中将对其进行详细说明。虽然这些模块可以使用文献中的技术来实现，但我们将它们全部通过提示技术来实现，以表明如果问题被简单地分解为正确的粒度级别，那么 LLMs 能够解决所有这些问题。提示中使用的少量示例来自各自的基准训练集。

4.1 调用链接模块

模式链接负责识别自然语言查询中的数据库模式引用和条件值。 它被证明有助于跨域的一般化以及复杂查询的合成，使其成为几乎所有问题的关键初步步骤。 [雷等人，2020 年]



(a) schema linking module (b) classification and decomposition module

Figure 3: Examples showing the input and output of schema linking (left) and classification and decomposition (right)

(a) 链接模式模块

现有的 文本到 SQL 方法 [曹等人，2021 年，王等人，2019 年，郭等人，2019 年，宣等人，2021 年]。 在我们的案例中，这也是一类 LLM 失败最多的类别（图 2）。

我们设计了一个基于提示的模块来连接模式。提示包括从蜘蛛数据集的训练集中随机选择的十个样本。按照链式思维模板[Wei 等人，2022 年]，提示以 Kojima 等人建议的“让我们

一步一步思考”开头[2022]。对于问题中提到的每一列名称，都会从给定的数据库模式中选择相应的列及其表。还可以从问题中提取可能的实体和单元格值。图 3a 显示了一个示例，完整的提示可以在附录 A.3 中找到。

4.2 分类与分解模块

对于每个连接，都有一定的机会无法检测到正确的表或连接条件。随着查询中连接的数量增加，至少一个连接无法正确生成的概率也会增加。缓解这个问题的一种方法是引入一个模块来检测要连接的表。此外，一些查询具有过程组件，例如不相关的子查询，这些可能独立生成，并与主查询合并。

为了解决这些问题，我们引入了一个查询分类和分解模块。该模块将每个查询分为三类之一：简单、非嵌套复杂和嵌套复杂。简单类别包括可以不进行连接或嵌套就可回答的单表查询。非嵌套类别包括需要连接但不需要子查询的查询，而嵌套类别中的查询可能包含连接、子查询和集合操作。这些类别的标签对我们的查询生成模块很重要，它使用每个查询类别的不同提示。除了类别标签外，查询分类和分解还会检测到用于非嵌套查询和嵌套查询的要连接的表集以及任何可能检测到的嵌套查询的子查询。图 3(b) 显示了提供给模型的示例输入及其生成的输出。

4.3 SQL 生成模块

随着查询变得越来越复杂，需要添加额外的中间步骤来弥合自然语言问题与 SQL 查询之间的差距。在文献中被称为“不匹配问题”（Guo et al.，2019），这给 SQL 生成带来了重大挑战，因为 SQL 主要是为了查询关系数据库而设计的，而不是……

自然语言中表达意义[凯特，2008 年]。虽然更复杂的查询可以受益于以链式思维的方式列出中间步骤，但这样的列表可能会降低处理较简单任务的性能[韦等人，2022 年 b]。基于同样的原因，我们的查询生成由三个模块组成，每个模块针对不同的类别。

对于简单课程中的问题，一个简单的几次提示就足够了，不需要中间步骤。该类示例 Ej 遵循格式 <Qj, Sj, Aj>，其中 Qj 和 Aj 分别给出英文查询文本和 SQL 语句，Sj 表示模式链接。我们的非嵌套复杂类包括需要连接查询。我们的错误分析（第 3 节）表明，在简单的少量提示下，找到正确列和外键来连接两张表对于 LLM 来说是具有挑战性的，尤其是当查询需要连接多张表时。为了解决这个问题，我们求助于一个中间表示来弥合查询和 SQL 语句之间的差距。文献中引入了各种中间表示。特别是，SemQL [Guo 等人，2019 年] 去掉了没有明确对应自然语言查询的操作符 JOIN ON、FROM 和 GROUP BY，并合并了 HAVING 和 WHERE 子句。NatSQL [Gan 等人，2021 年] 建立在 SemQL 的基础上并去除了集合操作符。自然语言查询中的表达式可能不会明确映射到唯一的 SQL 分支或它们可以映射到多个分支，因此删除操作符使从自然语言到 SQL 的转换变得更容易。作为我们的中间表示，我们使用 NatSQL，它被证明与其他模型结合使用时具有最先进的性能[Li 等人，2023 年 a]。非嵌套复杂类示例 Ej 的演示遵循格式 <Qj, Sj, Ij, Aj>，其中 Sj 和 Ij 分别表示 j 个示例的模式链接和中间表示。

最后，嵌套复杂类是最复杂的类型，并且在生成最终答案之前需要几个中间步骤。该类可以包含查询，这些查询不仅需要使用嵌套和集合操作（如 EXCEPT、UNION 和

INTERSECT）来创建子查询，还可以进行多个表连接，就像前一个类一样。为了进一步分解问题为多个步骤，我们为这个类设计了一个提示，以便 LLM 首先解决来自上一个模块的子查询，然后使用它们来生成最终答案。此类的提示采用以下格式：<Qj, Sj,<Qj1, Aj1,..., Qjk, Ajk>, Ij, Aj>，其中 k 表示子问题的数量，Qjai 和 Aj 分别表示第 i 个子问题和第 i 个子查询。与之前一样，Qj 和 Aj 分别表示英语查询和 SQL 查询，Sj 提供模式链接，Ij 是一个 NatSQL 中间表示。

所有三个查询类别的完整提示在附录 A.4 中提供，而所有三个类别的示例都来自用于分类提示的相同训练集数据库。

4.4 自我纠错模块

有时，生成的 SQL 查询语句可能缺少或包含多余的关键词，例如 DESC、DISTINCT 和聚合函数。我们在多个 LLM 上的经验表明，这些问题在较大的 LLM（例如 GPT-4 生成的查询比 CodeX 更少）中并不常见，但仍然存在。为了解决这个问题，我们提出了一种自我纠正模块，其中模型被指示纠正这些小错误。这是通过零样本设置实现的，在此设置下，仅向模型提供有缺陷的代码，并要求其修复错误。我们为自我纠正模块提出了两种不同的提示：通用和温和。使用通用提示时，我们要求模型识别并更正“BUGGY SQL”中的错误。另一方面，温和的提示不假设 SQL 查询有缺陷，而是要求模型检查任何潜在问题，并对要检查的子句提供一些提示。我们的评估表明，对于 CodeX 模型，通用提示可以产生更好的结果，而温和提示对于 GPT-4 模型更有效。除非明确说明否则，DIN-SQL 中的默认自我纠正提示针对 GPT-4 设置为温和，针对 CodeX 设置为通用。可以在附录 A.6 中找到这两种通用和温和自我纠正提示的示例。

5. 实验

5.1 模型

我们使用两个 Codex 家族变体（Davinci 和 Cushman 变体）以及 GPT-4 模型评估了所提出的方法。在撰写本文时，这些是最先进的开源语言模型

论文。较小的模型适用性较低，因为人们认为提示是参数量达到十亿级的 LLMs 的新兴能力（魏等，2022 年 a）。

5.2 超参数

所有模型都是通过 OpenAI API 访问的。通过将温度设置为零，使用贪心解码来生成输出。自我纠正模块的最大标记数设置为 350，而所有其他模块都设置为 600。自我纠正模块的停止标记序列设置为“

”，而对于所有其他模块，则设置为“Q:”。

我们在两个跨域挑战数据集上进行了评估，即 Spider 和 BIRD。Spider 包含 10,181 个问题和 5,693 个独特的复杂 SQL 查询，跨越 200 个数据库，涵盖 138 个领域，每个领域包含多个表。该数据集的标准协议将其分为 8,659 个训练示例（跨越 146 个数据库）、1,034 个开发示例（跨越 20 个数据库）以及一个包含 2,147 个测试示例的留出集（跨越 34 个数据库）。这些集合中使用的数据库互不重叠。SQL 查询根据使用的 SQL 关键字数量、嵌套子查询的存在以及列选择和聚合的使用被分为四个难度级别。BIRD 是一个广泛

的数据集，包含 12,751 个唯一的问句-SQL 对，包括 95 个大型数据库，总大小为 33.4GB。它涵盖了超过 37 个专业领域，包括区块链、曲棍球、医疗保健和教育。BIRD 还引入了外部知识作为额外资源来帮助模型生成准确的 SQL 查询。具体来说，引入了四种类型的外部知识：数值推理知识、领域知识、同义词知识和值说明。值得注意的是，SQL 查询在

Model	EX	EM
DIN-SQL + GPT-4 (Ours)	**85.3**	60
RESDSQL-3B + NatSQL (DB content used) [Li et al., 2023a]	79.9	72
DIN-SQL + CodeX davinci (Ours)	78.2	57
Graphix-3B+PICARD (DB content used) [Li et al., 2023b]	77.6	**74**
SHiP+PICARD (DB content used) [Zhao et al., 2022]	76.6	73.1
N-best Rerankers + PICARD (DB content used) [Zeng et al., 2022]	75.9	72.2
RASAT+PICARD (DB content used) [Qi et al., 2022]	75.5	70.9
T5-3B+PICARD (DB content used) [Scholak et al., 2021]	75.1	71.9
RATSQL+GAP+NatSQL (DB content used) [Gan et al., 2021]	73.3	68.7
RYANSQL v2 + BERT [Choi et al., 2021]	-	60.6
SmBoP + BART [Rubin and Berant, 2020]	-	60.5

Table 2: Execution accuracy (EX) and exact set match accuracy (EM) on the holdout test set of Spider

BIRD 数据集往往比 SPIDER 数据集更复杂。没有访问数据库内容的语言模型经常在架构连接方面遇到挑战。因此，我们在 BIRD 数据集中的提示包括来自每个表的示例行，以帮助模型进行架构连接。此外，我们还将每个问题提供的外部知识作为提示串联起来，并将其放在每个问题之后。然而，由于上下文窗口大小有限、存在外部知识以及包含样本行等限制，我们不得不减少 BIRD 数据集中提示的数量。

5.4 指标

我们使用每个数据集的官方指标来评估模型性能：对于 Spider 使用精确集匹配准确率 (EM) 和执行准确率 (EX)，对于 BIRD 使用有效效率得分 (VES) 和执行准确率 (EX)。精确集匹配准确率（EM）将每个子句视为一个集合，并将对每个子句的预测与参考查询中对应的子句进行比较。如果预测的 SQL 查询的所有组件都与真实值相匹配，则认为其正确。该指标不考虑值。执行准确率（EX）将预测 SQL 查询在某些数据库实例上的执行输出与真实 SQL 查询进行比较。由于可能存在多个有效的 SQL 查询，因此执行准确率提供了模型性能的更准确估计。

对于给定的问题，精确集匹配精度只评估预测的 SQL 与其中一个的匹配程度。有效效率得分（Ves）是一种度量生成的 SQL 查询运行效率的指标。如果生成的查询正确，即它们的结果与参考查询相同，则此度量是有意义的。因此，Ves 度量同时考虑了生成的查询的准确性和执行时间方面的效率。

5.5 结果

5.5.1 测试集结果

如表 2 所示，使用 GPT-4 在 Spider 的留出测试集上实现了最高的执行准确率，在此撰写时所有公开的结果中排名第三。这是在没有利用数据库内容的情况下实现的。就精确匹配准确性而言，我们的方法与不使用数据库内容的先前工作取得了可比的结果。如表 3a 所示，在鸟类数据集 (BIRD) 中，我们使用 GPT-4 的方法在测试集上达到了 55.9% 的执行准确率，创造了新的最先进的技术 (SOTA)。

5.5.2 开发集结果

我们在开发过程中对蜘蛛进行评估的主要方式是在可轻松访问的开发集中，而不是只能通过 Yu 等人提供的[2018 年]评估服务器访问的测试集。表 4a 显示了我们的方法在使用不同语言模型（LLM）时的表现，与 Rajkumar 等人的零样本提示[2022 年]、Liu 等人的少量提示[2023 年]以及我们自己的少量提示进行了比较。为了确保对于少量提示公平比较，我们将用于我们三个类别的所有示例（易类、非嵌套复杂类和嵌套复杂类）包含在提示中。鉴于 CodeX Cushman 模型具有比 CodeX Davinci 和 GPT-4 模型更小的输入上下文大小，我们只使用每个类别的两个示例（总共六个示例）。

我们的方法在精确集匹配和执行能力方面都显著优于简单的少样本提示和零样本提示。

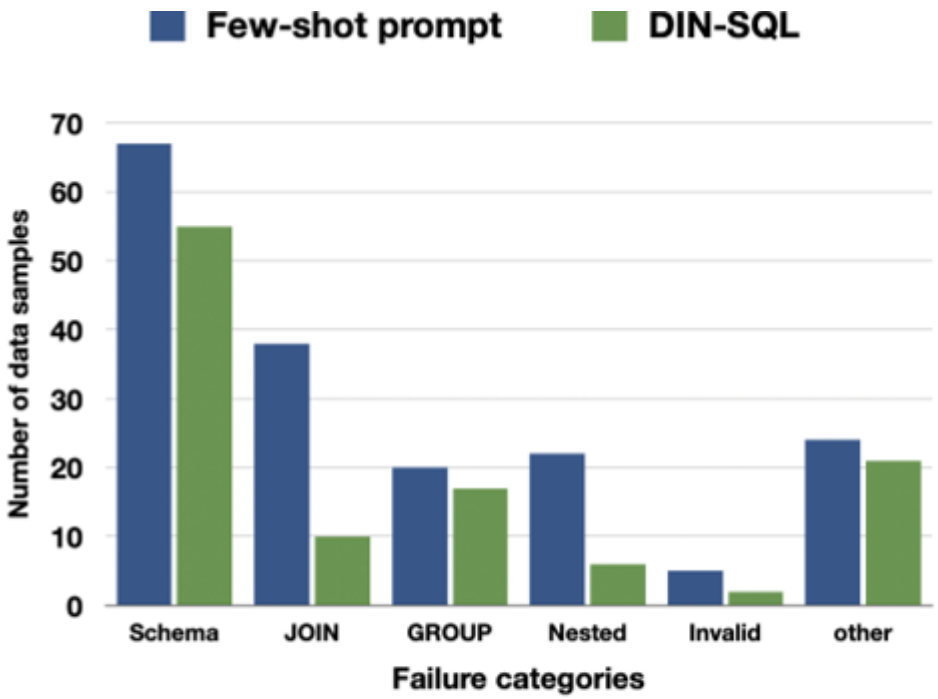


Figure 4: The break-down of failure cases for DIN-SQL (green) and the basic few-shot prompting (blue) across different categories

所有模型的执行精度都得到了提高，尽管它们的规模不同。例如，与仅使用几个提示相比，我们的方法可以将所有模型的执行精度至少提高 10%。

在 BIRD 开发集上，我们的方法显示出显著的改进，在执行精度方面实现了与 GPT-4 基准 Li 等人[2023c]相比高达 4%的提升，并在有效效率评分方面取得了显着的 9%的提升，从而建立了一个新的 SOTA。这些和其他结果都在表 3b 中报告。

在测试集上的表现（如表 2 和表 3a 所示）比开发集要好，对于 Spider 和 BIRD 都是如此。当测试集被隐藏时，很难准确地找出原因，但 我们推测测试集中较少的问题可能需要数据库内容的知识，因此我们的方法更容易预测正确的 SQL 查询。此外，开发集中存在模式模糊性（例如，一个查询实体可以映射到多个数据库实体，但只有一个被认为是正确的），因此测试集中可能具有更少的模糊性。

我们进一步分析了我们的方法在不同难度级别的查询上的性能。表 4b 显示了我们在蜘蛛数据集上进行开发设置时，与基本的少量提示相比，我们的方法的表现。在所有难度级别上，我们的方法都优于基本的少量提示，在额外困难和困难类中，少量提示表现不佳。我们的改进

(a) 在 BIRD 的保留测试集上的执行精度 (EX) 和有效效率得分 (VES)

(b) 在鸟类数据集上开发集上的执行精度(EX) 和有效效率评分(VES)

在容易类（与基本少数派相比）中，由于在提示中包含方案链接，因此存在差异，这突出了我们方案链接模块的重要性。

执行精度（EX）

(a) 在 Spider 开发集上使用不同的 LLM 进行零样本和少样本提示的性能比较。

（b）与我们在不同查询难度级别上的基本少量提示相比，性能比较

表 4：DIN-SQL 在其他上下文学习方法中的性能

5.5.3 错误修复

在第 3 节中，我们在训练集中的 500 个查询中对基本的少样本提示进行了错误分析。为了了解这些错误的程度，我们在相同的 500 个查询上运行了 DIN-SQL。如图 4 所示，我们的方法提高了所有类别的性能，JOIN 和嵌套类别取得了最大的改进。尽管有一个明确的模块用于链接模式，但大部分失败案例仍然属于该类别。

5.6 脱壳研究

在消融研究中，我们评估了使用或不使用每个模块的方法。如表 5 所示，对于 CodeX Davinci 模型，排除任何一种模块都会导致执行准确率的整体下降。

随着我们研究不同查询类别的每个模块的有效性，出现了更多细节。模式链接帮助所有查询类别，其中最难的类别改进最少。我们对失败案例样本的检查显示，由于问题或模式中的歧义，有时模式连接会发现冗余链接，这可能会引入冗余连接或输出列。

如果没有分类，我们不得不为所有查询使用简单的少量提示或分解后的思维链提示。据报道，在没有分类的情况下，性能不佳

Prompting	Model	Easy	Medium	Hard	Extra	All
DIN-SQL (generic self-corr)	CodeX Davinci	**89.1**	75.6	**58**	**38.6**	**69.9**
DIN-SQL (gentle self-corr)	CodeX Davinci	87.5	**76.9**	51.7	36.1	68.7
DIN-SQL w/o self-corr	CodeX Davinci	83.9	75.4	52.3	36.1	67.3
DIN-SQL w/o schema-linking	CodeX Davinci	87.3	70.6	57.6	27.1	65.9
DIN-SQL w/o classification (simple few-shot prompting)	CodeX Davinci	87.9	68.2	51.7	27.1	63.1
DIN-SQL w/o classification (decomposed COT prompting)	CodeX Davinci	84.2	71.2	54.3	38.6	68.2
DIN-SQL (gentle self-corr)	GPT-4	**91.1**	**79.8**	**64.9**	**43.4**	**74.2**
DIN-SQL (generic self-corr)	GPT-4	89.9	76.5	59.2	34.3	70.0
DIN-SQL w/o self-correc	GPT-4	**91.1**	79.1	63.2	41.6	73.3

Table 5: Performance of our method, in terms of execution accuracy, on the dev set with and without each module

表 5 中的 模块 是我们包含所有组件但不包括分类的综合框架。这意味着该方法不仅包含 COT 提示，还包括 Schema Linking、Self-Correction 和 NatSQL 中间表示，这些都是我们工作的重大贡献。在本表中呈现的 分解后的思维链 结果是指使用针对嵌套复杂类开发的最复杂的提示来回答所有问题，而不是根据问题的难度级别采用基于分类的方法来确定提示的复杂性。相比之下，DIN-SQL 的简单小样本结果指的是在不同级别的难度下为所有问题使用最简单的提示类别，即“容易”类别。不出所料，分解后的思维链提示对于困难和超难查询表现得更好，而简单的小样本则更适用于“容易”类别。

为了自我纠正，我们使用了 CodeX Davinci 和 GPT-4 进行研究。对于 CodeX Davinci，一个通用的自我纠正提示有助于模型在所有查询类别中工作。一个温和的自我纠正提示也很有帮助，但对 CodeX Davinci 的收益比通用提示要小。然而，GPT-4 生成有缺陷代码的可

能性较小，并且提供一个通用提示“有缺陷的 SQL：...修复后的 SQL：...”可能会损害性能。GPT-4 对于温和的提示效果更好，并提高了除简单类以外的所有类别的性能。

6 结论

提示使大型语言模型在各种领域的众多 NLP 任务上取得了显著的性能，而无需使用大量的训练数据。在我们的研究之前，利用 LLM 进行文本到 SQL 任务的提示方法的有效性不如专门针对该任务进行微调的模型。为了弥合这一差距，我们设计了一种分解技术来解决导致这种差异的一些挑战。我们在两个具有挑战性的数据集 Spider 和 BIRD 上进行了广泛的实验，结果表明，我们的方法显着提高了所有查询类别的提示性能，并产生了与最先进的微调方法相媲美甚至更好的结果。

7. 限制

这项工作存在一些局限性或改进空间。 我们手工构建的演示对于每个查询类都是固定的。未来的研究可能会探索自适应和自动化的技术，以生成更细粒度的演示，从而进一步提高我们的方法性能。此外，截至撰写本文时，我们提出的方法具有分解和分步结构，在使用 GPT-4 回答来自 Spider 数据集中的自然语言问题时，成本约为 0.5 美元，并且有大约 60 秒的延迟。随着 LLM 的不断发展，预计这些成本和延迟会降低，但降低成本也是另一个可能的方向。

致谢

这项研究由 加拿大自然与工程科学研究委员会资助。我们感谢陶宇和洪金素在 Spider 的保留测试集上运行我们的代码，以及 金阳李、宾源惠、雷诺德陈、葛曲 和其他 BIRD 作者在 BIRD 的保留测试集上运行我们的代码。我们还要感谢 Csaba Czepesvari、Dale Schuurmans 和 NeurIPS 的匿名审稿人对本工作的建设性意见。

参考文献

汤姆·布朗，本杰明·曼恩，尼克·莱德，梅勒妮·苏比亚，贾里德·卡普兰，普拉福尔·达尔瓦尔，阿林德·尼拉坎塔，潘纳夫·希亚姆，吉里许·萨斯特里，阿曼达·阿斯克尔 等人。语言模型是少量样本学习者。神经信息处理系统进展，33：1877–1901，2020 年。

芦驰，贾洁源，博彦徐，智丰郝。Sadga：结构感知的双图聚合网络用于文本到 SQL。神经信息处理系统进展，34：7664–7676，2021 年。

曹睿生、陈露、陈志宾、赵延斌、朱苏珠、余开宇。Lgesql：一种带有混合局部和非局部关系的增强文本到 SQL 模型。arXiv 预印本，2021 年 6 月 1 日：arXiv:2106.01093 [cs.CL]。

陈马克，杰瑞·特沃雷克，Heewoo Jun，钱启明，Henrique Ponde de Oliveira Pinto，贾里德·卡普兰，哈里·爱德华兹，尤里·伯尔达，尼古拉斯·约瑟夫，格雷格·布洛克曼等。使用代码训练的大语言模型的评估。arXiv 预印本 arXiv：2107.03374，2021 年。

陈文虎。 大型语言模型是一类少样本（1）表格推理器。 arXiv 预印本，arXiv:2210.06710，2022 年。

东允真·崔、明哲新、永云金、东润理。RyanSQL：在跨域数据库中递归应用基于草图的槽填充以实现复杂的文本到 SQL。计算语言学，47（2）：309–332，2021 年。

卡尔·科贝，维内特·科萨拉贾，穆罕默德·巴瓦尔安，马克·陈，何武俊，卢卡什·凯泽，马提亚斯·普拉珀特，杰瑞·托雷克，雅各布·希尔顿，黑尾直彦 等。 训练验证器解决数学问题。 预印本 arXiv:2110.14168 ，2021 年。
雅各布·德夫林，明纬昌，肯顿·李，克里斯蒂娜·图塔纳。Bert：用于语言理解的深度双向转换器的预训练。arXiv 预印本 arXiv:1810.04805，2018 年。
李东和 Mirella Lapata。神经注意力语言到逻辑形式。在第 54 届计算语言学协会年会论文集中（卷 1：长篇论文），2016 年，第 33-43 页。
干宇健，陈新云，谢金霞，马修·珀弗，约翰·R·伍德沃德，约翰·德雷克，张巧夫。自然 SQL：从自然语言规范中推断出 SQL 更容易。arXiv 预印本 arXiv:2109.05153，2021 年。
亚历克斯·格雷夫斯 和 亚历克斯·格雷夫斯。 长短期记忆，监督序列标记与递归神经网络，2012 年，第 37-45 页。
郭家琦，战泽诚，高岩，肖炎，刘健光，刘婷，张冬梅。《在跨域数据库中使用中间表示实现复杂的文本到 SQL》。arXiv eprint arXiv:1905.08205，2019 年。
祝信国，颜敏萱，齐洁馨，周建平，何紫微，林周翰，郑冠杰，王欣冰。《提示规划与知识记忆相结合的少量表到文生成》。arXiv 预印本 arXiv：2302.04415，2023 年。
乔纳森·赫兹格，Paweł Krzysztof Nowak，托马斯·穆勒，弗朗切斯科·皮钦诺，朱利安·马丁·艾伊斯恩霍夫。小吃：通过预训练进行弱监督表格解析。arXiv 预印本，arXiv:2004.02349，2020 年。

黄俊阳，王永波，王永良，杨东，肖扬华。nl2sql 的关系感知半自回归语义分析。arXiv 预印本，卷：2108.00804，2021 年。
宾元慧、项实、耿睿颖、李斌华、李永彬、孙健、朱晓丹。通过模式依赖学习提高文本到 SQL 转换。arXiv 预印本，2021 年 3 月 4 日：arXiv:2103.04399。
黄完兴、尹晋永、朴承炫、Seo Minjoon。使用表格意识词上下文对维基百科 SQL 进行综合探索。arXiv eprint arXiv:1902.01069，2019 年。
Rohit Kate。使用意义表示法语法来改进语义分析。在 CoNLL2008:计算自然语言学习第十二届会议论文集，2008 年，第 33-40 页。
图沙尔·科赫、哈什·特里维迪、马修·芬莱森、姚福、凯尔·理查德森、彼得·克拉克和阿希什·萨巴尔瓦尔。《解构提示：一种模块化方法来解决复杂任务》。预印本，arXiv:2210.02406，2022 年。
高岛健，张祥善，雷德·马歇尔，松尾裕太，岩澤雄介。大型语言模型是零样本推理者。arXiv 预印本，arXiv:2205.11916，2022 年。
布伦登·莱克 和 马可·巴罗尼。无系统性的一般化：序列到序列循环网络的复合技能。 在国际机器学习会议上，第 2873 - 2882 页。 PMLR，2018 年。
雷文强、王微信、马志新、甘天健、陆伟、Kan Min-Yen 和 Chua Tat-Seng。在《2020 年自然语言处理方法会议论文集》中重新审视了文本到 SQL 中的模式连接的作用，第 6943-6954 页，2020 年。

费利和 Hosagrahar V. Jagadish。构建关系数据库的交互式自然语言接口。《VLDB 终身成就奖论文集》，第 8 卷，第 1 期：73-84 页，2014 年。
李浩阳，张静，李翠萍，陈红。 解耦文本到 SQL 中的骨架解析和模式链接。arXiv 预印本，arXiv：2302.05965，2023 年 a。
金阳李、宾源惠、雷诺陈、鲍文秦、陈浩马、南火、费黄、文宇杜、罗思、永彬李。 图形-t5：使用带有图形感知层的预训练转换器进行文本到 SQL 解析。 arXiv 预印本 arXiv:2301.07507，2023 年 b。
金阳李，宾源惠，葛趣，斌华李，家喜杨，博恩李，白琳王，博恩秦，荣玉曹，瑞英耿，南火，陈浩马，凯文·C·C·张，飞黄，雷诺德·程，永彬李。Can llm 已经可以作为数据库接口了吗？大型数据库基于文本到 SQL 的大长椅，2023 年 c。
李云遥，杨华海，HV Jagadish。Nalix：一种针对 XML 数据的通用自然语言搜索环境。ACM 数据库系统期刊（TODS），32（4）：30-ES，2007 年。
刘爱伟，胡旭明，文立杰，于品石。ChatGPT 的零样本文本到 SQL 转换能力全面评估。arXiv 预印本，arXiv:2303.13547，2023a。
刘鹏飞，袁伟哲，付金兰，蒋正宝，林晃彦，格雷厄姆·纽比。预训练、提示和预测：自然语言处理中提示方法的系统调查。计算机协会计算调查，55（9）：1–35，2023 年 b。
安娜·马里亚·波佩斯库，奥伦·埃齐兹和亨利·考茨。自然语言界面数据库理论。在《第 8 届国际智能用户界面会议论文集》中，第 149 - 157 页，2003 年。
安娜·马里亚·波佩斯库，亚历克斯·阿曼纳苏，奥伦·埃茨齐，大卫·科，亚历山大·耶茨。数据库中的现代自然语言接口：使用统计分析结合语义可处理性。在 2004 年计算语言学国际会议 (COLING 2004): 计算语言学国际会议论文集中，第 141-147 页，2004 年。
解行奇、景耀堂、子为贺、向鹏玩、城虎周、信冰王、全实张、周寒林。Rasat：在预训练的 seq2seq 模型中集成关系结构以实现文本到 SQL。arXiv 预印本，卷：arXiv:2205.06983，2022 年。

秦 Bowen，惠斌源，王利汉，杨敏阳，李金阳，李彬华，耿瑞英，曹荣玉，孙健，司罗茜等。 文本到 SQL 解析调查：概念、方法和未来方向。 arXiv 预印本 arXiv:2208.13629，2022 年。
Nitish Rajkumar、Raymond Li 和 Dzmitry Bahdanau。《大型语言模型的文本到 SQL 能力评估》。arXiv eprint arXiv:2204.00498，2022 年。
Ohad Rubin 和 Jonathan Berant。Smbop：半自回归底部向上语义分析。arXiv 预印本，来自：arXiv:2010.12412 ，2020 年。
Torsten Scholak、Nathan Schucher 和 Dzmitry Bahdanau。Picard：从语言模型中逐步解析，用于受约束的自回归解码。arXiv 预印本 arXiv:2109.05093，2021 年。
Niculae Stratică、Leila Kosseim 和 Bipin C Desai。使用语义模板构建 Cindy 虚拟图书馆的自然语言接口。数据与知识工程，第 55 卷，第 1 期：4-19 页，2005 年。
伊利亚·苏茨凯韦，奥利奥尔·维尼亚尔斯和乔克·维·勒。神经网络的序列到序列学习。《神经信息处理系统进展》，第 27 卷，2014 年。

白林王、理查德辛、肖东刘、Oleksandr Polozov 和马修·理查森。 Rat-sql：文本到 SQL 解析器中的关系感知模式编码和链接。 arXiv e-print arXiv:1911.04942，2019 年。
Jason Wei，Maarten Bosma，Vincent Y Zhao，Kelvin Guu，Adams Wei Yu，Brian Lester，Nan Du，Andrew M Dai 和 Quoc V Le。精调语言模型是零样本学习者。arXiv 预印本 arXiv：2109.01652，2021 年。
魏杰森、易太、Rishi Bommasani、科林·拉夫尔、Barret Zoph、Sebastian Borgeaud、Dani Yogatama、Maarten Bosma、Denny Zhou、Donald Metzler 等。大型语言模型的涌现能力。arXiv 预印本，arXiv：2206.07682 ，2022 年 a。
魏杰森、王学志、达尔·舒尔曼斯、马特恩·博斯玛、埃德·奇、乔克·勒和邓尼·周。思维链提示在大型语言模型中激发推理。arXiv 预印本 arXiv:2201.11903，2022 年 b。
夏俊徐，常刘，黎明之歌。Sqlnet：从自然语言生成结构化查询而无需强化学习。arXiv 预印本，卷：1711.04436，2017 年。
关璇，王永波，王勇良，文祖杰，董阳。Sead：一种端到端的带表结构噪声消除的文本到 SQL 生成方法。arXiv 预印本 arXiv:2105.07911，2021 年。
彭城阴，格雷厄姆·纽比格，文涛伊和塞巴斯蒂安·里德尔。Tabert：联合理解文本和表格数据的预训练。arXiv 预印本，arXiv:2005.08314，2020 年。
陶宇，张睿，杨凯，Yasunaga Michihiro，王东旭，李子帆，James Ma，Irene Li，姚青宁，Roman Shanelle 等。蜘蛛：用于复杂跨域语义分析和文本到 SQL 任务的大规模人工标记数据集。arXiv 预印本，arXiv:1809.08887，2018 年。
陶宇，吴健生，林诗薇琪，王爱琳，谭一晨，杨心怡，Dragomir Radev，Richard Socher 和 熊彩明。Grappa：用于表格语义分析的基于语法增强的预训练。arXiv eprint arXiv:2009.13845，2020 年。
陆曾，Sree Hari Krishnan Parthasarathi 和 Dilek Hakkani-Tur。用于文本到 SQL 系统的 n 最佳假设重排序。arXiv 预印本 arXiv:2210.10668，2022 年。
张卓生、张阿斯顿、穆丽和亚历克斯·斯摩拉。在大型语言模型中自动触发思维链。arXiv 预印本，arXiv:2210.03493，2022 年。
赵一云，蒋家荣，胡奕群，兰伟伟，朱亨利，查阿努吉，亚历山大·李，潘琳，王军，黄宗伟。高质量数据合成对文本到 SQL 解析的重要性。arXiv 预印本，arXiv：2212.08785，2022 年。

朱睿琪、陶宇和 Dan Klein。使用蒸馏测试套件对文本到 SQL 进行语义评估。在《2020 年计算语言学协会会议：自然语言处理中的经验方法》（The 2020 Conference on Empirical Methods in Natural Language Processing）上。2020 年。
钟维克，熊彩明，理查德·索彻。Seq2sql：使用强化学习从自然语言生成结构化查询。arXiv 预印本 arXiv:1709.00103 ，2017 年。
邓尼·周、纳撒尼尔·舍勒、乐浩、杰森·韦伊、内森·斯凯尔斯、薛志伟、戴尔·舒尔曼斯、奥利维尔·布斯克特、罗伯特·李、埃德·奇。从少到多提示允许大型语言模型进行复杂的推理。arXiv 预印本，arXiv:2205.10625，2022 年。

第 15 页

提示

本节提供了在我们提出的四模块方法中使用的所有提示的全面列表，这些方法同时适用于 GPT-4 和 CodeX 模型。为了便于复制和理解该方法，每个模块使用的提示都进行了详细说明。此外，我们还包含了用于少量样本和零样本实现我们的方法的提示。

对于用于非嵌套复杂查询类和嵌套复杂查询类的少数示例，我们使用了来自 NatSQL GitHub 存储库 2 中的 NatSQL 中间表示。该存储库为 Spider 训练集中的所有查询提供了中间表示。

第 16 页

A.1 无目标提示

用于零样本提示场景的提示灵感来自刘等人[2023a]为 ChatGPT 提出的工作。在图 5 中，我们展示了本工作中使用的零样本提示的一个示例。

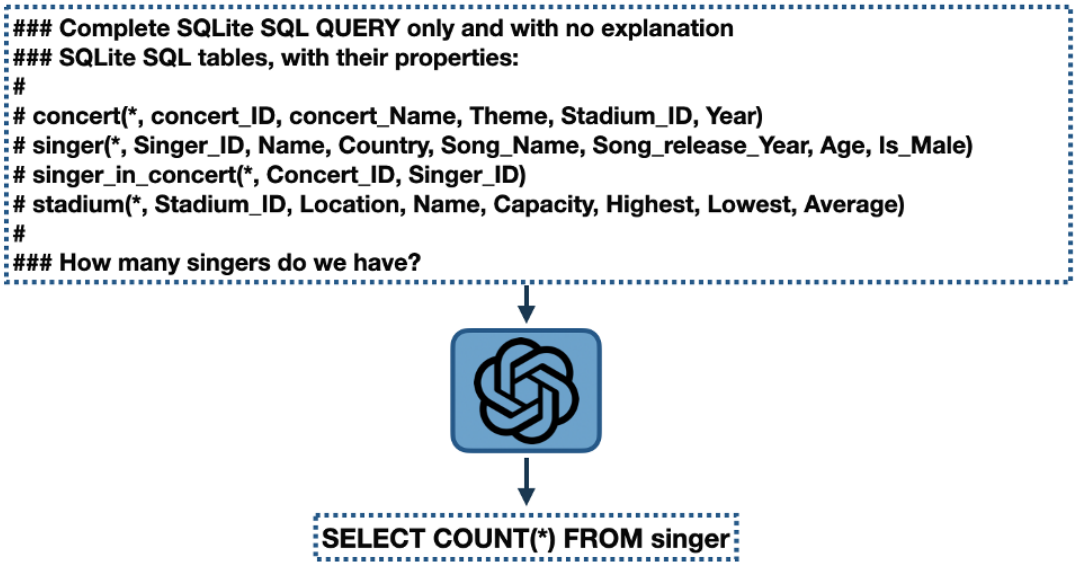


Figure 5: An example of Zero-shot prompting.

A.2 少样本提示

根据给定的问题创建 SQL 查询。

表顾问，列=[*，s_ID，i_ID]

表格教室，列= [* ，建筑物，房间号，容量]

表课程，列= [*， 课程编号， 课程名称， 部门名称， 学分]

表部门，列= [*, dept_name, building, budget]

表导师，列=【*，ID，name，dept_name，salary】

表预需求，列=【*，course_id，prereq_id】

表格部分，列数=[*，course_id，sec_id，semester，year，building，room_number，time_slot_id]

表 student，列[*，ID，name，dept_name，tot_cred]

表采用，列=[*，ID，course_id，sec_id，semester，year，grade]
表教学，列=[*，ID，course_id，sec_id，semester，year]
表 time_slot，列：[*，time_slot_id，day，start_hr，start_min，end_hr，end_min]
问：“找出拥有客房容量超过 50 间的建筑物。”
SQL：SELECT DISTINCT building FROM classroom WHERE capacity > 50

译文：从 classroom 表中选择不同（distinct）的 building，其中 capacity 大于 50。
问：“找出可容纳 50 至 100 名学生的房间号，以及它们的建筑类型——”
“在那个地点，有一些事物。”
SQL：SELECT building, room_number FROM classroom WHERE capacity BETWEEN 50 AND 100
问：“历史系学分最高的学生叫什么名字？”
SQL：从 student 中选择名字，条件是 dept_name = 'History'，按 tot_cred 降序排列，只返回第一条。
问：“找出营销或财务部门的总预算。”
SQL：从部门表中选择预算列，条件为 部门名称 为“市场营销”或 部门名称 为空。

'金融'
Q: "查询名字包含'Soisalon'的教师的部门名。" SQL: SELECT dept_name FROM instructor WHERE name LIKE '%Soisalon%'
问：“哪个系的学分最多？”
SQL：从 course 表中选择 department_name，按 department_name 分组，按 credits 的总和降序排序，限制只返回第一条记录。
问：“2010 年春季有多少教师教一门课程？”
SQL：从 teaches 表中选择不同 ID 的数量，其中 semester = '春季' AND YEAR = 2010。
问题: "找到学生姓名及其部门名称，按总学分升序排列。"
SQL：SELECT name,dept_name FROM student ORDER BY tot_cred
问：“找出课程数量最多的年份。”
SQL：SELECT YEAR FROM SECTION GROUP BY YEAR ORDER BY count(*) DESC LIMIT 1

译文：从 SECTION 表中选择 YEAR 字段，按 YEAR 分组，按数量降序排列，限制结果只显示前 1 条记录。
问题：“平均工资高于 42000 美元的部门的名称和平均工资是多少？”
SQL：SELECT dept_name, AVG(salary) FROM instructor GROUP BY dept_name HAVING AVG(salary)> 42000

以上内容的翻译是：

选择部门名称，教师薪资的平均值 从教师表中 按照部门名称分组 带有 平均薪资大于 42000 的筛选条件
问：“有多少个房间的容量超过 50？”

SQL：SELECT count(*) ,building FROM classroom WHERE capacity > 50 GROUP BY building
问："找出提供最多课程的前三大系？" SQL：SELECT dept_name FROM course GROUP BY dept_name ORDER BY count(*) DESC LIMIT 3

A:

A:
问："找出每栋楼中房间的最大容量和平均容量。" SQL：SELECT MAX（capacity），AVG（capacity），building FROM classroom GROUP BY building
问："找出由不止一个系提供的课程的标题。"
SQL：SELECT title FROM course GROUP BY title HAVING count(*)> 1

翻译结果:

——

——
问："找出营销或财务部门的总预算。" SQL：SELECT SUM（budget）FROM department WHERE dept_name = '营销'或 dept_name ='财务'
问："找出预算最高的部门的名称和大楼。"
SQL：SELECT dept_name,building FROM department ORDER BY budget DESC LIMIT 1

译文： 选择部门名称、建筑物。 从部门表中查询。 按预算降序排列。 返回第 1 条记录。
问："哪些部门的预算超过了平均水平？这些部门的名字是什么，它们位于什么大楼里？"
SQL：从部门表中选择部门名称和建筑物，其中预算大于所有部门的平均预算。
问题："找出每个系的总学生数和教师总数。" SQL：SELECT COUNT(DISTINCT T2.id) , COUNT(DISTINCT T3.id) , T3.dept_name FROM department AS T1 JOIN student AS T2 ON T1.dept_name = T2.dept_name JOIN instructor AS T3 ON T1.dept_name = T3.dept_name GROUP BY T3.dept_name
问题："找出具有两个先修课程的课程标题？"
SQL：SELECT T1.title FROM course AS T1 JOIN prereq AS T2 ON T1.course_id = T2.course_id GROUP BY T2.course_id HAVING count(*) = 2

问题："找出在 2009 年和 2010 年参加过任何课程的学生的名字。" SQL：SELECT DISTINCT T1.name FROM student AS T1 JOIN takes AS T2 ON T1.id= T2.id WHERE T2.YEAR= 2009 OR T2.YEAR= 2010
问："按字母顺序列出所有课程名称及其教师姓名，年份为 2008 年。"
SQL：SELECT T1.title, T3.name FROM course AS T1 JOIN teaches AS T2 ON T1.course_id = T2.course_id JOIN instructor AS T3 ON T2.id = T3.id WHERE T2.YEAR = 2008 ORDER BY T1.title

"+

table of contents

搜索所有课程
问题：“找出具有两个先修课程的课程标题？”
SQL：SELECT T1.title FROM course AS T1 JOIN prereq AS T2 ON T1.course_id = T2.course_id GROUP BY T2.course_id HAVING count(*) = 2
问：“找出预算最高的部门的名称和大楼。”SQL：SELECT dept_name,building FROM department ORDER BY budget DESC LIMIT 1
问：“找到有多个先修课程的课程的标题、学分和部门名称？”
SQL：SELECT T1.title，T1.credits，T1.dept_name FROM course AS T1 JOIN prereq AS T2 ON T1.course_id = T2.course_id GROUP BY T2.course_id HAVING count(*) > 1
问：“给出预算高于平均水平的部门的名字和大楼。”SQL：选择部门名，大楼。在部门中，当预算>（从部门选择的平均预算）时。
问：“找出在 2009 年秋季授课但在 2010 年春季没有授课的教师的 id。”SQL：SELECT id FROM teaches WHERE semester = 'Fall' AND YEAR = 2009 EXCEPT SELECT id FROM teaches WHERE semester = 'Spring' AND YEAR = 2010
问：“找出没有先修课程的课程名称？” SQL：SELECT title FROM course WHERE course_id NOT IN (SELECT course_id FROM prereq)
问：“找出所有工资低于最大工资的唯一教师。”SQL：选择 不同 的工资 从 教师 WHERE 工资 <（选择最大（工资） 从 教师）
问题：“找出 2003 年秋季学期上过课的学生的名字。”
SQL：从 student 中选择名字，id 在 takes 中（选择学期为“秋季”且年份为 2003）的学生。
问题：“找出所有部门中平均工资高于所有教师平均工资的最低工资。”
SQL：选择部门名称、最低工资 from instructor group by 部门名称 having 平均工资> (select avg(salary) from instructor)
问：“移动计算”课程先修课的课程名称是什么？

答：计算机科学导论。
SQL：从 course 表中选择 title，其中 course_id 在 prereq 表中的子查询中返回，该子查询为 T1.prereq_id 连接 course 表为 T2.course_id，且 T2.title = '移动计算'。
问：“给出教室内容量最大的课程的标题和学分。”
SQL：选择 T3.tittle，T3.credits。来自 classroom 作为 T1 连接 SECTION 作为 T2，在 T1.building = T2.building 和 T1.room_number = T2.room_number 中使用 T2.course_id = T3.course_id 连接 course as T3 在 T1.capacity（选择 max（capacity）from classroom）。

A.3 链接提示方案

根据数据库模式和外键，为每个问题生成 SQL 查询 的 schema_links。

表城市，列= [* ， City_ID，官方名称，状态，面积（km ^ 2），人口，人口普查排名]
表竞赛记录，列 = [*，竞赛 ID，农场 ID，排名]
表农场，列 = [* ， Farm_ID ， Year ， Total_Horses ， Working_Horses ，
总牛，母牛，公牛，母牛，猪，羊和山羊表 farm_competition, 列[*,竞赛 ID,年份,主题,主办城市 ID,主办方] 外键[farm_competition. Host_city_ID = 城市. City_ID,
competition_record. Farm_ID = farm.农场 ID]
农场。Farm_ID, competition_record.Competition_ID = farm_competition.Competition_ID]
问："显示举办过最多比赛的城市的状态。"
A：让我们一步一步来。在问题"显示举办过最多比赛的城市的状态"中，我们被问到：
"城市的状态"，所以我们需要列= [城市的状态]
"最多的比赛"所以我们需要列=[farm_competition.*]
根据列和表，我们需要这些 外键 = [farm_competition . Host_city_ID = city . City_ID] 。
根据表格、列和外键，可能的单元格值集合为=。因此，模式链接是：
模式链接：[城市状态，农场竞赛.主办城市 ID = 城市.城市 ID，农场竞赛*]
表部门，列=[，*Department_id，名称，创建，排名，预算十亿，员工人数] 表头，列=[*，
头部 ID，姓名，出生州，年龄]
表管理，列=[*，部门编号，主管编号，临时代理]
外键：[management.head_ID = head.head_ID, management.department_ID = department.Department_ID]
问："有多少部门主管年龄超过 56 岁？"
A：让我们一步一步来。在"有多少部门主管年龄超过 56 岁？"这个问题中，我们被问到：
"有多少部门主管" 所以我们需要列=[头*]
"年长"的意思，所以我们需要 column=[head.age]
根据列和表，我们需要这些外键= []。
根据表格、列和外键，可能的单元格值集合为[56]。因此，模式链接是：
[头.*，头.年龄，56]
表格部门，列出了= [* ， Department_ID， Name， Creation， Ranking，
Budget_in_Billions， Num_Employees]。
表头，列数=[*，头 ID，姓名，出生州，年龄]
表管理，列=[*，部门编号，主管编号，临时代理]
外键：[management.head_ID = head.head_ID, management.department_ID = department.Department_ID]
问："在阿拉巴马州出生的秘书管理的部门有哪些独特的成立年份？"
A：让我们一步一步来。在问题"出生于阿拉巴马州的秘书管理的部门的独特创建年份是什么？"中，我们被问到：
"部门的独特创建年份"，所以我们需要列=[部门。创建]
"由...管理"的部门，所以我们需要列= [management.department_ID]
因此我们需要 列= [head . born_state]

根据列和表，我们需要
Foreign_keys=[department.Department_ID=management.department_ID,
management.head_ID= head.head_ID]。
根据表格、列和外键，可能的单元格值集合为 ['阿拉巴马州']。因此，架构链接是：
模式链接：[部门.创建，部门.部门 ID = 管理.部门 ID，主管.头衔 ID = 管理.头衔 ID，主管.
出生状态，"阿拉巴马州"]
表名 地址，列 = [*, address_id, line_1, line_2, city, zip_postcode, state_province_county,
country] 申请者评估，列 = [*, candidate_id, qualification, assessment_date,
asessment_outcome_code]

候选人表，列 = [* , candidate_id, candidate_details]
表课程，列=[*，course_id，course_name，course_description，其他详细信息]
表 人，列 = [* , person_id, first_name, middle_name,
姓氏，手机号码，电子邮件地址，登录名，密码]
表 People_Addresses，列 = [*, person_address_id, person_id, address_id, date_from,
date_to]
表 Student_Course_Attendance，列[*，student_id,course_id,date_of_attendance]
表 Student_Course_Registrations，列[*，student_id，course_id，registration_date]
表 Students，列：[*，student_id，student_details]
外键： Students.student_id = People.person_id,People_Addresses.address_id =
Addresses.address_id,People_Addresses.person_id =
人.人编号，课程注册.课程编号=
课程.课程编号,学生注册.学生编号=
学生.学生 ID，Student_Course_Attendance.学生 ID=
Student_Course_Registrations.学生编号,Student_Course_Attendance.课程编号 =
Student_Course_Registrations.课程编号,Candidates.候选人编号=
人.人_id,候选人评估.候选人_id = 候选人.候选人_id]
Q: "列出从未上课的学生的 id？"
A：让我们一步一步来。在问题"列出从未参加课程的学生 ID？"中，我们被问到：
"学生 id" 所以我们需要 列= [Students.student_id]
从未参加课程，所以我们需要列= [学生课程出席记录. 学生 ID]
根据列和表，我们需要这些 外键 = [Students . student_id = Student_Course_Attendance .
student_id] 。
根据表格、列和外键，可能的单元格值集合为=。因此，模式链接是：
模式链接：[Student_Student_id = Student_Course_Attendance_Student_id]
表 Country，列 = [* ， id，name]
表格联赛，列数=[*，id，country_id，name]

表 玩家，列 = [* ， id， player_api_id， player_name， player_fifa_api_id，
birthday， height， weight] 表 玩家 属性，列 = [*， id， player_fifa_api_id，
player_api_id， date， overall_rating， potential， preferred_foot，
attacking_work_rate， defensive_work_rate， crossing， finishing
, 头球精度, 短传, 传球, 控球, 抢点射门, 脚法, 罚球精度
长传，控球，加速，冲刺速度，敏捷性，反应能力，平衡感
射门力量，跳跃，耐力，体力，远射，侵略性，拦截
定位、视野、惩罚、标记、站立扑救、滑铲、守门员潜水
守门员能力：控球、射门、站位、反应。
表团队，列=[*，id，team_api_id，team_fifa_api_id，team_long_name，team_short_name]
表 Team_Attributes，列 = [* ， id， team_fifa_api_id， team_api_id， date，
buildUpPlaySpeed]
,buildUpPlaySpeedClass,buildUpPlayDribbling,buildUpPlayDribblingClass
buildUpPlayPassing,buildUpPlayPassingClass,buildUpPlayPositioningClass,chanceCreationP
assing
,机会创造传球类,机会创造带球突破,机会创造带球突破类
, 投篮机会创造,投篮机会创造类,位置创造类
防御压力 防御压力等级 防御侵略性 防御侵略性等级
[防御队列宽度，防守队形，防守阵型]
表 sqlite_sequence，列[*，name，seq]
外键 = [Player_Attributes. 玩家 API ID = Player.玩家 API ID,
Player_Attributes.player_fifa_api_id = Player.player_fifa_api_id,
League.country_id = Country.id, Team_Attributes.team_api_id = Team.team_api_id,
Team_Attributes.team_fifa_api_id = Team.team_fifa_api_id]
问："列出所有左脚球员的名字，他们的综合评分为 85 到 90。"
A：让我们一步一步来。在问题"列出所有左脚球员的姓名，他们的综合评分在 85 到 90 之间"中，我们被问及：
"所有左脚球员的名字" 所以我们需要列 = [Player . player_name , Player_Attributes . preferred_foot]
"有总体评分的球员"，所以我们需要列= [Player_Attributes.overall_rating]
根据列和表，我们需要这些 外键 = [Player_Attributes.player_api_id = Player.player_api_id]。
根据表格、列和外键，可能的单元格值集合为左 [85, 90]。所以

Schema_links 是：
链接：[玩家.球员名字，Player_Attributes.首选脚，Player_Attributes.综合评级，
Player_Attributes.球员 API ID = 球员.球员 API ID，左，85，90]
表顾问，列=[*，s_ID，i_ID]

表格教室，列= [*，建筑物，房间号，容量]
表课程，列= [*， 课程编号， 课程名称， 部门名称， 学分]
表部门，列= [*, dept_name, building, budget]
表导师，列=【*，ID，name，dept_name，salary】
表预需求，列=【*，课程 id，预需求 ID】
表格部分，列数=[*，course_id，sec_id，semester，year，building，room_number，time_slot_id]
表 student，列[*，ID，name，dept_name，tot_cred]
表采用，列=[*，ID，course_id，sec_id，semester，year，grade]
表教学，列=[*，ID，course_id，sec_id，semester，year]
表 time_slot，列：[*，time_slot_id，day，start_hr，start_min，end_hr，end_min]
外键： [course.dept_name = department.dept_name, instructor.dept_name = department.dept_name, section.building = classroom.building]
房间号= classroom.room_number
section.course_id = course.course_id, teaches.ID = instructor.ID, teaches.course_id = section.course_id, teaches.sec_id = section.sec_id,
教.学期=部分.学期，教.年=部分.年，学生.系名=部门.系名，
取 ID= 学生 ID，取课程 ID= 节次课程 ID，
取 sec_id 为 section 的 sec_id， 取 semester 为 section 的 semester，
取值.year=section.year,导师.s_ID=学生.ID,
顾问.i_ID = 教授. ID，先修课程.prereq_id = 课程.course_id，先修课程.course_id = 课程.课程 id]
问：“在 2010 年秋季，钱德勒学院提供什么课程？”
A：让我们一步一步来。在问题“Chandler 在 2010 年秋季提供的课程名称是什么？”中，我们被问到：
课程标题，所以我们需要一列= [课程.标题]
“Chandler 提供的课程” 所以我们需要 列 = [SECTION.building]
“秋季”所以需要 column = [SECTION.semester]
“2010 年”所以我们需要列=[部分。年份]
根据列和表，我们需要 Foreign_keys=[course.course_id= section.course_id]。
根据表格、列和外键，可能的单元格值集合为 = [Chandler, Fall, 2010]。因此，模式链接是：
模式链接：[课程.标题， 课程.课程编号 = 部分.课程编号，部分.建筑物，部分.年份，部分.学期，钱德勒，秋季，2010]
表顾问，列=[*，s_ID，i_ID]
表格教室，列= [* ，建筑物，房间号，容量]
表课程，列= [*， 课程编号， 课程名称， 部门名称， 学分]
表部门，列= [*, dept_name, building, budget]

表导师，列=【*，ID，name，dept_name，salary】
表预需求，列=【*，课程 id，预需求 ID】
表格部分，列数=[*，course_id，sec_id，semester，year，building，room_number，time_slot_id]
表 student，列[*，ID，name，dept_name，tot_cred]
表采用，列=[*，ID，course_id，sec_id，semester，year，grade]
表教学，列=[*，ID，course_id，sec_id，semester，year]
表 time_slot，列：[*，time_slot_id，day，start_hr，start_min，end_hr，end_min]
外键： [course.dept_name = department.dept_name, instructor.dept_name = department.dept_name]，
section.building= classroom.building, section.room_number= classroom.room_number,
section.course_id = course.course_id,teaches.ID = instructor.ID,teaches.course_id = section.course_id,
teaches.sec_id= 节.节编号,teaches.semester= 节.学期,teaches.year= 节.年份,
学生表：department.dept_name, takes.ID=学生 ID, takes.course_id=section.course_id

课程编号，
取 sec_id 为 section 的 sec_id， 取 semester 为 section 的 semester，
取值.year = section.year， advisor.s_ID = student.ID， advisor.i_ID = instructor.ID，
prereq(prereq_id=course.course_id) 和 prereq(course_id=course.course_id)。
问：“找出在 2009 年秋季上课但不在 2010 年春季上课的教师的 id。”
A：让我们一步一步来。在问题“找到在 2009 年秋季授课的教师的 id”中
但不是在 2010 年春季。"，我们被问到：
“授课教师 ID”，因此我们需要列= [teaches.id]
所以需要 列 = [teaches.semester, teaches.year]
根据列和表，我们需要这些外键= []。
根据表格、列和外键，可能的单元格值集为=
[2009 年秋季，2010 年春季]。因此，链接方案是：
链接：[teaches.id,teaches.semester,teaches.year,'Fall','2009','Spring','2010']
表 账户，列 = [* ，帐户 ID，客户 ID，帐户开户日期，账户名称，其他账户详细信息]
表 客户，列 = [* ，客户 ID，客户名字，客户中间名，
客户姓氏,性别,电子邮件地址,登录名,登录密码,电话号码,
城镇/城市,州/省/县,国家]
表 Financial_Transactions， 列 = [* ， transaction_id， account_id， invoice_number，transaction_type，
交易日期,交易金额,交易备注,其他交易详情]
表发票行明细，列=[*，订单项 ID，发票号码，产品 ID，产品标题，产品数量，产品价格，衍生产品成本，衍生增值税应付，衍生总成本]

表发票，列=[*，发票号码，订单编号，发票日期]
表 Order_Items，列= [*，order_item_id，order_id，product_id，product_quantity，other_order_item_details]
表订单，列=[*，order_id，customer_id，date_order_placed，order_details]
表 产品类别，列 = [* ，生产类型代码，产品类型描述，增值税评级]
表 产品，列= [* , product_id, parent_product_id, production_type_code , 单价, 产品名称, 产品颜色, 产品尺寸]
Foreign_keys = [Orders.customer_id = Customers.customer_id,Invoices.order_id = Orders.order_id, Accounts.customer_id= Customers.customer_id,
产品.生产类型代码= 产品类别.生产类型代码，财务交易.账户 ID = 账户.账户编号,财务交易.发票号码= 发票.发票号码,
Order_Items.order_id = Orders.order_id, Order_Items.product_id = Products.product_id,
发票行项目。产品编号 = 产品编号，发票行项目。发票号码 = 发票号码，
发票行项目.order_item_id = 订单项目.order_item_id]
Q: "显示所有帐户的 ID、开户日期、账户名称和其他账户详细信息。"
A：让我们一步一步来。在问题“显示所有帐户的 id、开户日期、账户名称和其他详细信息”中，我们被问到：
“账户 ID、开户日期、账户名称和其他账户详细信息。”所以我们需要列= [Accounts.account_id,
账户.账号,账户.其他账户详情,账户.开户日期]
根据列和表，我们需要这些外键= []。
根据表格、列和外键，可能的单元格值集合为=。因此，模式链接是：
模式链接：[Accounts.account_id, Accounts.account_name,
账户.其他账户详情 账户.开户日期]
表顾问，列=[*，s_ID，i_ID]
表格教室，列= [* ，建筑物，房间号，容量]
表课程，列= [*， 课程编号， 课程名称， 部门名称， 学分]
表部门，列= [*, dept_name, building, budget]
表导师，列=【*，ID，name，dept_name，salary】
表预需求，列=【*，课程 id，预需求 ID】
表格部分，列数=[*，course_id，sec_id，semester，year，building，room_number，time_slot_id]
表 student，列[*，ID，name，dept_name，tot_cred]
表采用，列=[*，ID，course_id，sec_id，semester，year，grade]
表教学，列=[*，ID，course_id，sec_id，semester，year]

表 time_slot，列：[*，time_slot_id，day，start_hr，start_min，end_hr，end_min]

外键： [course.dept_name = department.dept_name, instructor.dept_name = department.dept_name]，
section.building= classroom.building, section.room_number= classroom.room_number, section.course_id = course.course_id, teaches.ID = instructor.ID, teaches.course_id = section.course_id, teaches.sec_id = section.sec_id,
教.学期=节.学期，教.年=节.年，学生.系名=系.系名，修.学号=学生.学号，修.课程编号=节.课程编号，
取.学号= section.学号， 取.学期= section.学期， 取.年份= section.年份，指导教师.s_ID= 学生.ID，
顾问.i_ID = 指导教师.ID，先修课程.prereq_id = 课程.course_id，先修课程.course_id = 课程.course_id]
问："找出拥有客房容量超过 50 间的建筑物。"
A：让我们一步一步来。在问题"找出有房间容量超过 50 的建筑物"中，我们被问到：
"有房间的建筑" 所以我们需要 列出= [教室容量]
"房间容量"所以我们需要列=[classroom.building]
根据列和表，我们需要这些外键= []。
根据表格、列和外键，可能的单元格值集合为[50]。因此，模式链接是：
链接模式：[教室建筑，教室容量，50]
表城市，列= [* ， City_ID，官方名称，状态，面积（km ^ 2），人口，人口普查排名]
表竞赛记录，列 = [*，竞赛 ID，农场 ID，排名]
表 Farm, 列 = [*Farm_ID,Year,Total_Horses,Working_Horses,Total_Cattle,Oxen,Bulls,Cows,Pigs,Sheep_and_Goats] 表 Farm_competition, 列 = [*,Competition_ID,Year,Theme,Host_city_ID,Hosts]
外键： farm_competition.Host_city_ID = city.City_ID competition_record.Farm_ID = 农场。Farm_ID, competition_record.Competition_ID = farm_competition.Competition_ID]
问："显示人口在 1500 到 500 之间共享的城市的状态。"
A：让我们一步一步来。在问题"显示人口大于 1,000 的城市共享的状态"中
如果一个数在区间(1500,500)内，我们被问到：
"城市共享的状态"，所以我们需要列=[城市.状态]
"人口城市"所以我们需要列= [城市。人口]
根据列和表，我们需要这些外键= []。
根据表格、列和外键，可能的单元格值集合为[1500, 500]。因此，模式链接如下：
城市.状态 城市.人口 1500 500

A.4 分类与分解提示

根据嵌套查询和连接，对给定的问题进行分类，分为 简单、非嵌套或嵌套。

如果需要嵌套查询，请预测 NESTED。

如果需要连接，但不需要嵌套查询：预测非嵌套
如果不需要 JOIN 也不需要嵌套查询，预测难度为 Easy。
表顾问，列=[*，s_ID，i_ID]
表格教室，列= [*，建筑物，房间号，容量]
表课程，列= [*， 课程编号， 课程名称， 部门名称， 学分]
表部门，列= [*, dept_name, building, budget]
表导师，列=【*，ID，name，dept_name，salary】
表预需求，列=【*，课程 id，预需求 ID】
表格部分，列数=[*，course_id，sec_id，semester，year，building，room_number，time_slot_id]
表 student，列[*，ID，name，dept_name，tot_cred]
表采用，列=[*，ID，course_id，sec_id，semester，year，grade]
表教学，列=[*，ID，course_id，sec_id，semester，year]
表 time_slot，列：[*，time_slot_id，day，start_hr，start_min，end_hr，end_min]

外键= [course.dept_name = department.dept_name,instructor.dept_name = department.dept_name,section.building = classroom.building,section.room_number = classroom.room_number,
section.course_id = course.course_id,teaches.ID = instructor.ID,teaches.course_id = section.course_id,teaches.sec_id = section.sec_id,teaches.semester = section.semester,
教.年 = 节.年,学生.系名 = 部门.系名,选修.学号 = 学生.学号,选修.课程编号 = 节.课程编号,选修.节号 = 节.节号,选修.学期 = 节.学期,选修.年 = 节.年,导师.s_ID = 学生.学号,
顾问.i_ID = 指导教师.ID，先修课程.prereq_id = 课程.course_id，先修课程.course_id = 课程.course_id]
问：“找出拥有客房容量超过 50 间的建筑物。”
schema_links: [教室.建筑, 教室容量, 50]
A：让我们一步一步来。对于问题“找到有超过 50 个座位的房间”的 SQL 查询，需要使用这些表[classroom]，所以我们不需要 JOIN。
此外，它不使用 (INTERSECT, UNION, EXCEPT, IN, NOT IN) 的嵌套查询，并且我们需要对问题 [""] 的答案。
所以，我们不需要 JOIN 也不需要嵌套查询，那么 SQL 查询就可以被归类为“简单”。
标签：“简单”
Q: "所有指导数学系学生的导师的名字是什么？"
模式链接： advisor.i_id = instructor.id, advisor.s_id = student.id, instructor.name, student.dept_name, student.tot_cred, math
A：让我们一步一步来。问题“数学系所有指导学生的教师姓名按学生总学分排序”的 SQL 查询需要这些表=[advisor,instructor,student]，所以我们需要 JOIN。

此外，它不需要嵌套查询（INTERSECT, UNION, EXCEPT, IN, NOT IN），并且我们需要对问题 [""] 的答案。
因此，我们需要 JOIN 而不需要嵌套查询，那么 SQL 查询可以被归类为“非嵌套”。
标签：“非嵌套”
问：“有多少门没有先修课程？”
schema_links:[课程.*, 课程.课程编号 = 先修课.课程编号]
A：让我们一步一步来。问题“有多少没有先修课程？”的 SQL 查询需要这些表= [course，prereq]，所以我们需要 JOIN。
此外，它需要嵌套查询（ INTERSECT、UNION、EXCEPT、IN、NOT IN ），而且我们需要回答问题["哪些课程有先修课？"]。
因此，我们需要连接查询和嵌套查询，然后 SQL 查询可以被归类为“嵌套”。 标签：“嵌套”
问：“找出可以容纳 50 到 100 名学生的房间号及其建筑物。”
schema_links: [classroom.building , classroom.room_number , classroom.capacity , 50 , 100]
A：让我们一步一步来。对于问题“找到可以容纳 50 到 100 名学生的房间号及其建筑物”的 SQL 查询，我们需要这些表=[classroom]，所以我们不需要 JOIN。
此外，它不使用（INTERSECT, UNION, EXCEPT, IN, NOT IN）的嵌套查询，并且我们需要对问题 [""] 的答案。
所以，我们不需要 JOIN 也不需要嵌套查询，那么 SQL 查询就可以被归类为“简单”。
标签：“简单”
问题: "找到由统计学系和心理学系提供的课程的标题。" schema_links: [course.title, course.dept_name, Statistics, Psychology]
A：让我们一步一步来。对于问题“找到提供的课程的标题”所对应的 SQL 查询，如下所示：

由统计学系和心理学系。" 需要这些表格 [课程]，所以我们不需要 JOIN。
此外，它需要嵌套查询（ INTERSECT、UNION、EXCEPT、IN、NOT IN），并且我们需要对问题["找到心理学系提供的课程"]的答案。所以我们不需要连接符，而且需要嵌套查询，那么 SQL 查询就可以被归类为“嵌套”。
标签：“嵌套”
问：“找出在 2009 年秋季上课但不在 2010 年春季上课的教师的 id。”
schmea_links: [teaches.id, teaches.semester, teaches.year, Fall, 2009, Spring, 2010]
A：让我们一步一步来。对于问题“找出在 2009 年秋季上课但不在 2010 年春季上课的教师的 id”的 SQL 查询，需要使用这些表=[teaches]，所以我们不需要 JOIN。此外，它要求嵌套查询（ INTERSECT、UNION、EXCEPT、IN、NOT IN），并且我们需要对问题的答案是[找到在 2010 年春季上课的教师的 id]。因此，我们不需要 JOIN，并且需要嵌套查询，然后 SQL 查询可以归类为
“嵌套”。
标签：“嵌套”

问："找出提供最高总学分的系？"
schema_links:[课程.系名，课程.学分]
A：让我们一步一步来。问题"找出提供最多学分的部门"的 SQL 查询需要这些表[课程]，所以我们不需要 JOIN。
此外，它不使用（INTERSECT, UNION, EXCEPT, IN, NOT IN）重嵌套查询，并且我们需要问题的答案 [""]。
所以，我们不需要 JOIN 也不需要嵌套查询，那么 SQL 查询就可以被归类为"简单"。
标签："简单"
问："指导学分最多的顾问叫什么名字？"
模式链接：[advisor.i_id = instructor.id, advisor.s_id = student.id, instructor.name, student.tot_cred] A: 让我们一步一步来。查询"具有最多学分的学生的指导老师的姓名是什么？"需要这些表 [advi-sor, instructor, student]，所以我们需要连接。
此外，它不需要嵌套查询（INTERSECT, UNION, EXCEPT, IN, NOT IN），并且我们需要对问题 [""] 的答案。
因此，我们需要 JOIN 而不需要嵌套查询，那么 SQL 查询可以被归类为"非嵌套"。
标签："非嵌套"
问："找出每个系的总学生人数和教师总数。"
模式链接：department.dept_name = instructor.dept_name，student.id，student.dept_name = department.dept_name，instructor.id。
A：让我们一步一步来。问题"查找每个部门的学生总数和教师总数"的 SQL 查询需要这些表[department, instructor, student]，所以我们需要 JOIN。
此外，它不需要嵌套查询（INTERSECT, UNION, EXCEPT, IN, NOT IN），并且我们需要对问题 [""] 的答案。
因此，我们需要 JOIN 而不需要嵌套查询，那么 SQL 查询可以被分类为"非嵌套"。
标签："非嵌套"
问："给出预算高于平均水平的部门的名字和大楼。"
schema_links: [部门预算，部门名称，建筑物]
A：让我们一步一步来。查询"给出部门预算高于平均水平的名称和建筑物"的 SQL 语句需要这些表= [部门]，所以我们不需要 JOIN。
此外，它需要嵌套查询（ INTERSECT、UNION、EXCEPT、IN、NOT IN），我们需要

问题的答案是["各部门的平均预算是什么"]。
所以，我们不需要 JOIN 语句，而是需要嵌套查询，那么这个 SQL 查询就可以被归类为"嵌套"。
标签："嵌套"

A.5 SQL 生成

A.5.1 课程简介

使用 模型链接 为每个问题生成 SQL 查询。

表顾问，列=[*，s_ID，i_ID]
表格教室，列= [*，建筑物，房间号，容量]
表课程，列= [*， 课程编号， 课程名称， 部门名称， 学分]
表部门，列= [*, dept_name, building, budget]
表导师，列=【*，ID，name，dept_name，salary】
表预需求，列=【*，course_id，prereq_id】
表格部分，列数=[*，course_id，sec_id，semester，year，building，room_number，time_slot_id]
表 student，列[*，ID，name，dept_name，tot_cred]
表采用，列=[*，ID，course_id，sec_id，semester，year，grade]
表教学，列=[*，ID，course_id，sec_id，semester，year]
表 time_slot，列：[*，time_slot_id，day，start_hr，start_min，end_hr，end_min]
问："找出拥有客房容量超过 50 间的建筑物。"
链接模式：[教室建筑，教室容量，50]
SQL：SELECT DISTINCT building FROM classroom WHERE capacity > 50

译文：查询课室表中容量大于 50 的不同教学楼。
问："找出可容纳 50 至 100 名学生的房间号，以及它们的建筑类型——"
"这是对每个人都有意义的事情。"
链接模式：[教室.建筑物名称，教室.房间号，教室.容量，50，100]
SQL：SELECT building, room_number FROM classroom WHERE capacity BETWEEN 50 AND100
问："历史系学分最高的学生叫什么名字？"
模式链接：[学生.姓名，学生.系名，学生.总学分，历史]SQL：从学生中选择名字等于"历史"的系名为"历史"按降序排列限制为 1 的总学分
问："找出营销或财务部门的总预算。"
模式链接：[部门.预算，部门.部门名称，市场营销，财务] SQL：从部门选择预算总和 部门名称= '市场营销' 或者 部门名称='财务'
问："找到教授名字中包含 'Soisalon' 的部门名称。"
模式链接：[教师.系名，教师.姓名，学生沙龙]
SQL：从 instructor 表中选择部门名称，其中包含 "Soisalon" 的名字。
问："学分最多的部门叫什么名字？"
模式链接：[课程.系名, 课程.学分]
SQL：从 course 表中选择 department_name，按 department_name 分组，按 credits 的总和降序排序，限制只返回第一条记录。

问：“2010 年春季有多少教师教一门课程？”
模式链接：[teaches.ID,teaches.semester,teaches.YEAR,'Spring',2010]
SQL：从 teaches 表中选择不同 ID 的数量，其中 semester = '春季' AND YEAR = 2010。

问题: "找到学生姓名及其部门名称，按总学分升序排列。"
模式链接：[学生.姓名，学生.系名，学生.总学分]
SQL：SELECT name,dept_name FROM student ORDER BY tot_cred
问：“找出课程数量最多的年份。”
[部分.年份, 部分.*]
SQL：SELECT YEAR FROM SECTION GROUP BY YEAR ORDER BY count(*) DESC LIMIT 1

译文：从 SECTION 表中选择 YEAR 字段，按 YEAR 分组并按数量降序排序，限制结果为前一条记录。
问题：“平均工资高于 42000 美元的部门的名称和平均工资是多少？”
模式链接：[教师.系名，教师.薪水，42000]
SQL：SELECT dept_name, AVG(salary) FROM instructor GROUP BY dept_name HAVING AVG(salary)> 42000

以上内容的翻译是：

选择部门名称，教授薪资的平均值，来自 instructor 表，按照部门名称分组，条件为教授薪资的平均值大于 42000。
问：“有多少个房间的容量超过 50？”
模式链接：[classroom.*, classroom.building, classroom.capacity, 50]
SQL：SELECT count(*) FROM classroom WHERE capacity > 50 GROUP BY building
问：“找出提供最多课程的前三大系？”
模式链接：[课程.系名，课程.*]
SQL：从课程表中选择系名，按系分组并按行数降序排列，限制为前 3 条记录。
问：“找出每栋楼中房间的最大容量和平均容量。”
链接模式：[classroom.building, classroom.capacity]
SQL：SELECT capacity FROM classroom GROUP BY building ORDER BY capacity DESC LIMIT 1;
问：“找出由不止一个系提供的课程的标题。”
课程标题
SQL：SELECT title FROM course GROUP BY title HAVING count(*)> 1

翻译结果:

课程标题

选课数量 > 1

A.5.2 非嵌套复杂度

使用模式链接和中间表示来生成每个问题的 SQL 查询。

表顾问，列=[*，s_ID，i_ID]
表格教室，列= [*，建筑物，房间号，容量]
表课程，列= [*， 课程编号， 课程名称， 部门名称， 学分]
表部门，列= [*, dept_name, building, budget]
表导师，列=【*，ID，name，dept_name，salary】
表预需求，列=【*，课程 id，预需求 ID】
表格部分，列数=[*，course_id，sec_id，semester，year，building，room_number，time_slot_id]
表 student，列[*，ID，name，dept_name，tot_cred]
表采用，列=[*，ID，course_id，sec_id，semester，year，grade]
表教学，列=[*，ID，course_id，sec_id，semester，year]
表 time_slot，列：[*，time_slot_id，day，start_hr，start_min，end_hr，end_min]
外键： [course.dept_name = department.dept_name, instructor.dept_name = department.dept_name, section.building = classroom.building,]
section.room_number = classroom.room_number, section.course_id = course.course_id, teaches.ID = instructor.ID, teaches.course_id = section.course_id,
teaches.sec_id = section.sec_id, teaches.semester = section.semester, teaches.year = section.year, student.dept_name = department.dept_name, takes.ID = student.ID,
takes.course_id = section.course_id, takes.sec_id = section.sec_id, takes.semester = section.semester,
取年段为 section.year，指导教授为 advisor.s_ID，指导学生为 advisor.i_ID，先修课程为 prereq.prereq_id，课程为 course.course_id，

prereq.course_id= course.course_id]
问："找出营销或财务部门的总预算。"
模式链接：[部门.预算，部门.部门名称，市场营销，财务]
A：让我们一步一步来。为了创建给定问题的 SQL，我们需要连接这些表=[]。
首先，创建一个中间表示，然后使用它来构造 SQL 查询。
中间表示：从部门表中选择 总预算 部门，其中 部门.dept_name = "市场营销" 或者 部门.dept_name = "财务"
SQL：从部门表中选择预算之和，其部门名称为'市场营销'或'财务部'。
问："找出预算最高的部门的名称和大楼。"
模式链接：[部门.预算，部门.部门名称，部门.建筑物]
A：让我们一步一步来。为了创建给定问题的 SQL，我们需要连接这些表=[]。
首先，创建一个中间表示，然后使用它来构造 SQL 查询。

中间表示：select department.dept_name, department.building from department order by department.budget desc limit 1
SQL：SELECT dept_name,building FROM department ORDER BY budget DESC LIMIT 1

译文： 选择部门名称、建筑物。 从部门表中查询。 按预算降序排列。 返回第 1 条记录。
问："哪些部门的预算超过了平均水平？这些部门的名字是什么，它们位于什么大楼里？"
模式链接：[部门.预算，部门.部门名称，部门.建筑物]
A：让我们一步一步来。为了创建给定问题的 SQL，我们需要连接这些表=[]。
首先，创建一个中间表示，然后使用它来构造 SQL 查询。
中间表示：从部门选择部门名称，部门建筑。其中@大于或等于部门预算的平均值。
SQL：从部门表中选择部门名称和建筑物，其中预算大于所有部门的平均预算。
问："找出每个系的总学生数和教师总数。"
模式链接：[department.dept_name= student.dept_name,student.id,department.dept_name=instructor.dept_name,instructor.id]
A：让我们一步一步来。为了创建给定问题的 SQL，我们需要连接这些表=[部门、学生、教师]。
首先，创建一个中间表示，然后使用它来构造 SQL 查询。
中间表示: "选择 不同 的学生.id, 不同的教师.id, 部门.dept_name 从部门组由教师.dept_name"
SQL：SELECT COUNT(DISTINCT T2.id) , COUNT(DISTINCT T3.id) , T3.dept_name FROM department AS T1 JOIN student AS T2 ON T1.dept_name = T2.dept_name JOIN instructor AS T3 ON T1.dept_name = T3.dept_name GROUP BY T3.dept_name
问题："找出具有两个先修课程的课程标题？"
模式链接：[课程标题，课程.课程 ID =先修课程.课程 ID]
A：让我们一步一步来。为了创建给定问题的 SQL，我们需要连接这些表=[课程，先修课]。
首先，创建一个中间表示，然后使用它来构造 SQL 查询。
中间表示：选择课程。标题从课程中，其中 count（prereq.*）= 2 组 by prerequisite。
SQL：SELECT T1.title FROM course AS T1 JOIN prereq AS T2 ON T1.course_id = T2.course_id GROUP BY T2.course_id HAVING count(*) = 2
问："找出在 2009 年和 2010 年参加过任何课程的学生的名字。"
模式链接：[学生.姓名，学生.学号= takes.学号，takes.YEAR，2009，2010]
A：让我们一步一步来。为了创建给定问题的 SQL，我们需要连接这些表=[学生，选修]。

首先，创建一个中间表示，然后使用它来构造 SQL 查询。
中间表示：从学生表中选择不重复的学生名字，条件为 takes.year 等于 2009 或者 2010 。

SQL：SELECT DISTINCT T1.name FROM student AS T1 JOIN takes AS T2 ON T1.id= T2.id WHERE T2.YEAR= 2009 OR T2.YEAR= 2010

翻译结果：选择不同的 T1.name，从学生表（T1）和选课表（T2）连接起来，条件为 T1.id = T2.id，且 T2.YEAR 等于 2009 或 2010。
问："按字母顺序列出所有课程名称及其教师姓名，年份为 2008 年。"
模式链接：[课程标题，课程.课程编号=教授.课程编号，teaches.id=教师.id，教师.姓名，teaches.年份，2008]
A：让我们一步一步来。为了创建给定问题的 SQL，我们需要连接这些表=[课程、教、教师]。
首先，创建一个中间表示，然后使用它来构造 SQL 查询。
中间表示：select course.title, instructor.name from course where teaches.year=2008 order by course.title asc
SQL：SELECT T1.title, T3.name FROM course AS T1 JOIN teaches AS T2 ON T1.course_id = T2.course_id JOIN instructor AS T3 ON T2.id = T3.id WHERE T2.YEAR = 2008 ORDER BY T1.title

"+

table of contents

搜索所有课程

A.5.3 嵌套复杂度

使用 中间表示 和 模板链接 为每个问题生成 SQL 查询。

表顾问，列=[*，s_ID，i_ID]
表格教室，列= [* ，建筑物，房间号，容量]
表课程，列= [*， 课程编号， 课程名称， 部门名称， 学分]
表部门，列= [*, dept_name, building, budget]
表导师，列=【*，ID，name，dept_name，salary】
表预需求，列=【*，课程 id，预需求 ID】
表部分，列=[，*course_id，sec_id，semester，year，building，room_number，time_slot_id]*
表学生，列=[，ID，name，dept_name，tot_cred]
表采用，列=[*，ID，course_id，sec_id，semester，year，grade]
表教学，列=[*，ID，course_id，sec_id，semester，year]
表 time_slot，列：[*，time_slot_id，day，start_hr，start_min，end_hr，end_min]
外键= [course.dept_name = department.dept_name,instructor.dept_name = department.dept_name,section.building = classroom.building,section.room_number = classroom.room_number,

section.course_id= 课程.course_id,teaches.ID= 教授.ID,teaches.course_id= section.course_id,teaches.sec_id= section.sec_id,teaches.semester= section.semester,teaches.year=section.year,student.dept_name= 部门.dept_name,takes.ID= 学生.ID,takes.course_id=section.course_id,
取 sec_id= section.sec_id,takes.semester= section.semester,takes.year= section.year,advisor.s_ID= student.ID,advisor.i_ID= instructor.ID,prereq.prereq_id= course.course_id,prereq/course_id=course.course_id]
问题：“找出具有两个先修课程的课程标题？”
模式链接：[课程标题，课程.课程 ID =先修课程.课程 ID]
A：让我们一步一步来。"找到有两门先修课的课程的标题？"可以解决
通过回答以下子问题"有哪些课程需要两个先修课？"，来了解答案。
子问题“哪些课程有两门先修课？”的 SQL 查询语句为：SELECT T1.title FROM course AS T1 JOIN prereq AS T2 ON T1.course_id= T2.course_id GROUP BY T2.course_id HAVING count(*)= 2。
那么，问题“找到有两个先修课程的课程标题？”的答案是=
中间表示：选择课程。标题从课程中，其中 count（prereq.*）= 2 组 by prerequisite。
SQL：SELECT T1.title FROM course AS T1 JOIN prereq AS T2 ON T1.course_id = T2.course_id GROUP BY T2.course_id HAVING count(*) = 2

问：“找出预算最高的部门的名称和大楼。”
模式链接：[部门.部门名称，部门.建筑物，部门.预算]
A：让我们一步一步来。"找到预算最高的部门的名字和大楼。"可以分解为知道下列子问题的答案："拥有最大预算的部门对应什么名字和大楼？"
子问题“预算最大的部门对应的部门名称和建筑物是什么？”的 SQL 查询语句为：SELECT dept_name,building FROM department ORDER BY budget DESC LIMIT 1。
所以，问题“找到预算最高的部门的名字和大楼。”的答案是=
中间表示：select department.dept_name, department.building from department order by department.budget desc limit 1
SQL：SELECT dept_name,building FROM department ORDER BY budget DESC LIMIT 1

译文： 选择部门名称、建筑物。 从部门表中查询。 按预算降序排列。 返回第 1 条记录。
问：“找到有多个先修课程的课程的标题、学分和部门名称？”
模式链接：[课程标题, 课程学分，系名，课程编号 = 先修课编号]
A：让我们一步一步来。“找出有多个先修课程的课程的标题、学分和部门名称”可以通过回答以下子问题“有多个先修课程的课程的标题、学分和部门名称是什么？”来解决。
子问题“有哪些课程有多于一个先修课？”的 SQL 查询为 SELECT T1.title, T1.credits, T1.dept_name FROM course AS T1 JOIN prereq AS T2 ON T1.course_id= T2.course_id GROUP BY T2.course_id HAVING count(*)> 1

所以，问题“找到预算最高的部门的名字和大楼。”的答案是=
中间表示：从课程表中选择课程标题，学分，部门名称，其中 count（prereq）* 大于 1 分组通过 prereq 课程 id
SQL：SELECT T1.title，T1.credits，T1.dept_name FROM course AS T1 JOIN prereq AS T2 ON T1.course_id = T2.course_id GROUP BY T2.course_id HAVING count(*) > 1
问：“给出预算高于平均水平的部门的名字和大楼。”
模式链接：[部门.部门名称，部门.建筑物，部门.预算]
A：让我们一步一步来。"给出预算高于平均水平的部门的名称和大楼号码"可以通过回答以下子问题来解决："部门的平均预算是多少？"。
子问题“部门平均预算是多少?” 的 SQL 查询为：SELECT avg(budget) FROM department。
所以，问题“给出预算高于平均水平的部门的名称和大楼。”的答案是=
中间表示：从部门选择部门名称，部门建筑。其中@大于或等于部门预算的平均值。
SQL：从部门表中选择部门名称和建筑物，其中预算大于所有部门的平均预算。
问：“找出在 2009 年秋季上课但不在 2010 年春季上课的教师的 id。”
链接：[teaches.id,teaches.semester,teaches.YEAR,"Fall","2009","Spring","2010"]
A：让我们一步一步来。 "找出在 2009 年秋季上课但在 2010 年春季没有上课的教师的 id" 可以通过知道以下子问题的答案来解决："找出在 2010 年春季上课的教师的 id"。
子问题“查找在 2010 年春季授课教师的 id”的 SQL 查询语句为 SELECT id FROM teaches WHERE semester='Spring' AND YEAR=2010。
所以，问题“找到在 2009 年秋季上课但不在 2010 年春季上课的教师 id”的答案是=
中间表示：select teaches.ID from teaches where teaches.semester="Fall" and teaches.year= 2009 and teaches.semester ! = "Spring" and teaches.year= 2010
SQL：从 teaches 中选择 id，其中 semester = '秋季' AND YEAR = 2009 EXCEPT SELECT

教过 的 id ，来自 在 Spring 学期且年份为 2010 。
问：“找出没有先修课程的课程名称？”
模式链接：[课程标题, 课程编号]
A：让我们一步一步来。"找出没有先修课程的课程名称？" 可以通过知道以下子问题的答案来解决："有哪些课程有先修课程？"。
子问题“哪些课程有先修课？”的 SQL 查询语句是 SELECT course_id FROM prereq
所以，问题“找出没有先修课程的课程名称”的答案=中间表示：从 course 表中选择 course.title，其中@ @不在 prereq.course_id 中。
SQL：从课程表中选择标题 WHERE course_id 不在 (SELECT course_id FROM prereq)
Q: "找出所有不重复的工资低于最大工资的教师。"
链接: [教师工资]
A：让我们一步一步来。"找出所有工资低于最大工资的不同讲师的工资"可以通过知道以下子问题的答案来解决："讲师的最大工资是多少？"
子问题“讲师的最大工资是多少”的 SQL 查询为 SELECT max(salary) FROM instructor。

所以，问题“找出所有不重复教师的工资，其工资小于最大工资。”的答案是=
中间表示：选择不同的教师工资 from 教师 where@.@<最大（教师工资）
SQL：从 instructor 表中选择工资不等于 instructor 表的最大工资的工资。
问题：“找出 2003 年秋季学期上过课的学生的名字。”
模式链接：[学生. id，学生. 名称，采取. id，采取学期，秋天，2003]
A：让我们一步一步来。"找出在 2003 年秋季学期修过任何课程的学生的名字。"可以由回答以下子问题得到解答："找出在 2003 年秋季学期修过任何课程的学生。"
子问题“找出在 2003 年秋季学期上过课的学生”的 SQL 查询语句为：SELECT id FROM takes WHERE semester= ’Fall’ AND YEAR= 2003。
所以，问题“找出在 2003 年秋季学期上过课的学生的名字”的答案是=
中间表示：从学生表中选择姓名，条件为修读学期“秋季”且修读年份为 2003。
SQL：从 student 中选择名字，其 id 在（在 takes 中选择 id，其中 semester='秋季'且 year = 2003）
问题：“找出所有部门中平均工资高于所有教师平均工资的最低工资。”
模式链接：[教师.工资,教师.系名]
A：让我们一步一步来。"找出所有部门中平均工资高于所有教师平均工资的最低工资"可以通过知道以下子问题的答案来解决："所有教师的平均工资是多少？"。
子问题“所有教师的平均工资是多少？”的 SQL 查询为 SELECT avg(salary) FROM instructor。
那么，问题“找到平均工资高于所有教师平均工资的部门的最低工资”答案=
中间表示：select min(instructor.salary), instructor.dept_name from instructor where avg (instructor.salary)>avg (instructor.salary) group by instructor.dept_name
SQL：选择部门名称、最低工资 从教师表 GROUP BY 部门名称 HAVING 平均工资 > (选择教师表中的平均工资)
问：“移动计算”课程先修课的课程名称是什么？

答：计算机科学导论。

模式链接：[课程标题，课程编号=先修课编号，先修课编号，课程标题，移动计算]
A：让我们一步一步来。"移动计算课程先修课的课程名称是什么？"可以通过回答下面的问题来解决："移动计算课程先修课的 id 是什么？"。
子问题“移动计算课程先修课的 id 是什么?” 的 SQL 查询语句是 SELECT T1.prereq_id FROM prereq AS T1 JOIN course AS T2 ON T1.course_id= T2.course_id WHERE T2.title= ’Mobile Computing’
那么，课程移动计算的先修课是什么课程？答案是=
中间表示：select course.title from course where@.@ in prereq.* and course.title="Mobile Computing"
SQL：从 course 表中选择 title，其中 course_id 在 prereq 表中的子查询中返回，该子查询为 T1.prereq_id 连接 course 表为 T2.course_id，且 T2.title = '移动计算'。

问：“给出教室内容量最大的课程的标题和学分。”
链接模式： [教室容量,教室楼=SEC-]
[building, classroom, room_number] = [section, room_number], course [title, credits, course_id] = [section, course_id]]
A：让我们一步一步来。"给在课堂上讲授的课程起个标题并署名
“具有最大的容量。”可以解决知道以下子问题的答案：“最大房间的容量是多少？”
子问题“最大房间能容纳多少人？”的 SQL 查询语句为：(SELECT max(capacity) FROM classroom)
所以，对于“给出在教室里有最大容量的课程的标题和学分”这个问题的答案是=
中间表示：选择课程.标题，课程.学分从课堂按教室容量降序排列限制为 1"
SQL：选择 T3.tittle，T3.credits。来自 classroom 作为 T1 连接 SECTION 作为 T2，在 T1.building = T2.building 和 T1.room_number = T2.room_number 中使用 course 作为 T3，在 T2.course_id = T3.course_id 中，其中 T1.capacity 等于（选择最大的 capacity）from classroom

第 33 页

A.6 自我校正提示

A.6.1 普通自我纠正提示

通用自我纠正提示在零样本设置中实现，其中所有查询都被假定为“有缺陷的 SQL”。该提示的一个示例如图 6 所示。

选择 名称 ,容量 FROM 体育场 ORDERBY 平均数 DESCLIMIT 1

第 34 页

A.6.2 温和自我纠正提示

我们以零样本设置实现了温柔的自我纠正提示。对于这个自我纠正提示，我们不假设它是有缺陷的，并且包含了一些修复 SQL 查询的说明。此提示的一个示例如图 7 所示。