arXiv:2512.17967v1 [cs.DB] 18 Dec 2025

Memelang：一种用于大语言模型生成的向量关系查询的轴向语法

Bri Holt
bri@memelang.net

摘要

面向大语言模型工具使用的结构化生成，凸显了紧凑型领域特定语言中间表示（IR）的价值：这类表示可直接生成，并能进行确定性解析。本文介绍了轴向语法：通过特定秩的分隔符令牌的位置，从线性令牌序列中还原多维结构。只需从左到右遍历一遍，即可为每个令牌分配其在 n维网格中的坐标，从而无需括号或由大量子句构成的表层语法，便可实现确定性解析。本文将这一语法应用于$Memelang$：这是一种紧凑型查询语言，旨在作为可由大语言模型生成的中间表示，其固定坐标角色可直接映射到表/列/值槽位。Memelang 支持坐标稳定的相对引用、解析时变量绑定，以及隐式上下文延续，以减少大语言模型生成的查询中的重复内容。它还通过值项上的内联标签编码分组、聚合和排序，从而能够基于坐标索引表示，在单次流式遍历中推导出分组执行计划。本文提供了参考词法分析器/解析器，以及一个可生成参数化 PostgreSQL SQL 的编译器（可选使用 pgvector 运算符）。

1 引言

本文提出一种紧凑的中间表示（IR），旨在供大语言模型直接生成，并确定性地编译为混合向量关系查询。重点在于尽量减少生成字符串中的结构自由度，同时保留一条直接明了的参数化 SQL 编译路径。

任务定义。 给定自然语言查询和数据库架构，大语言模型生成一个 IR 字符串，该字符串 (i) 简洁且词元数量较少，(ii) 可确定性地解析 为具有类型正确的 roles 的结构化表示，并且 (iii) 可编译为参数化 执行计划，以支持混合向量关系评估。

设计目标。

- **无需嵌套的确定性解析**。 该语法必须允许唯一解析，且不依赖括号、缩进敏感性或大量子句关键字。
- **适合流式处理的结构**。 解析应能通过单次从左到右的扫描完成，从而支持增量验证和提前检测错误（并可选择在受限解码下实现前缀有效的补全）。
- **低熵表层形式**。 该表示应尽量减少词元数量和句法变化，从而减轻生成器的负担，并限制等价但形式不同的子句排列。
- **受限的结构自由度**。语义角色（例如表/列/值）应与固定位置关联，以约束输出空间并提高生成的可靠性。
- **安全的编译边界**。编译应保留用户提供值的参数化形式，避免通过字符串拼接构造查询，从而降低注入风险，并支持后端规划和缓存。

非目标。 Memelang 并非旨在成为：(i) 通用编程语言；(ii) 对 SQL 的完整重新实现（包括任意嵌套、相关子查询或引擎特定的过程性功能）；或 (iii) 优化器规范。本文聚焦于一种紧凑、可由大语言模型生成的中间表示，其结构确定，支持常见的混合查询模式，并可进行可预测的编译与验证。

贡献。 本文作出以下贡献：

- 定义轴向语法，并为按分隔符定秩的令牌流提供一种确定性的单遍坐标分配方案。
- 介绍*Memelang*，一种轴向语法查询中间表示，具有固定槽位语义（表/列/值）、相对引用、解析时变量绑定，以及用于分组/聚合/排序的内联标签。
- 提供参考词法分析器/解析器、坐标索引中间表示，以及一个可生成参数化 PostgreSQL SQL 的编译器（包括可选的 pgvector 相似度运算符）。

章节2 阐述轴向语法与受限解码、文本到 SQL 中间表示以及混合向量—关系系统之间的关联。

2 相关工作

2.1 语法约束解码与结构化生成中间表示

大量研究将结构化大语言模型输出视为约束生成问题，即将解码限制为语法或模式所接受的字符串。研究表明，针对受限解码的增量解析对 SQL 等结构化目标有效，尤其是在生成过程中通过解析器参与的过滤来实现 [1]。更广泛而言，无需微调的语法约束解码 [2] 以及近期面向受约束推理和正确性的解码框架 [3, 4] 强调，表示选择可以通过缩小语法有效续写的集合来降低搜索复杂度。从实证来看，结构化生成越来越多地采用 JSON Schema 作为约束语言；近期的基准测试工作（JSONSchemaBench）对约束解码框架进行了比较，并指出它们在模式覆盖率、效率和下游质量方面的实际权衡 [5]。Memelang 与这些工作互为补充：它不提出新的解码算法，而是提出一种中间表示表面形式，其结构由分级分隔符和固定坐标角色决定，可减少子句排列，并以极低的分隔符开销实现确定性解析。

2.2 文本到 SQL 中间表示

中间表示（IR）是文本到 SQL 中常用的一种策略，旨在弥合自然语言意图与 SQL 子句机制之间的不匹配。IRNet 会生成 SemQL 中间形式，并将其确定性地映射为 SQL [6],；而 NatSQL 则明确删除或规范化难以与自然语言对齐的部分 SQL 内容（例如某些关键字和嵌套结构），以简化生成过程 [7]。此外，表现出色的文本到 SQL 解析器通常依赖结构化模式编码和显式的 SQL 结构建模，例如关系感知的模式编码与链接 [8]，或结构化程序的自底向上构建 [9]。大语言模型时代近期的文本到 SQL 系统也强调执行反馈和搜索/纠错循环，以及具备改进的模式链接和自我纠错能力的轻量级框架 [10–12]。大多数跨领域文本到 SQL 结果仍在 Spider 等基准测试套件上报告 [13], ，同时也持续开展工作，以更新适用于现代大语言模型的旧版数据集 [14]，并研究多语言求值设置 [15]。Memelang 的总体动机同样是降低生成正确 SQL 的难度，但其机制不同：它依据坐标位置固定表/列/值角色（而非依据 SQL 子句语法），并使用分级分隔符在单遍中还原结构，目标是实现一种熵更低、可由大语言模型生成且便于编译、适用于混合工作负载的中间表示。有关基于大语言模型的文本到 SQL 方法及其趋势的更全面概述，请参阅近期综述 [16–18]。

2.3 神经生成的查询语言和线性化

语义解析和程序合成中的一个常见设计维度，是如何将层级结构线性化以适配序列模型：基于语法的动作序列和 AST 驱动生成会显式呈现语法约束 [19]。在文本到 SQL 任务中，结构化解码器和类 AST 表示也会通过将生成过程与语法产生式或组合子结构绑定，减少无效程序 [8, 9]。其他线性化方式（例如 S 表达式、栈语言或缩进敏感格式）通常保留显式嵌套标记，或依赖空白结构。相比之下，轴向语法无需显式嵌套即可进行线性化：多维结构根据少量分级分隔符的位置推断得出，语义角色则由稳定的坐标索引分配。这是相较于以往线性化方式的主要创新：Memelang 的分隔符

编码轴之间的转换，而坐标角色编码语义槽位，二者共同约束有效字符串的范围，无需使用充斥括号的语法。

2.4 混合向量-关系查询模型与系统

近期系统研究主张，为混合工作负载采用统一的表示和优化器，以处理关系谓词与向量相似性搜索的组合 [20–24]。关于向量数据库管理系统设计以及经过筛选的 ANN（向量+标量）工作负载的综述进一步强调，混合查询如今已成为主要的接口需求，而非边缘情况 [25, 26]。除了查询模型之外，近期研究还强调了生产环境中的相关问题，例如可扩展的嵌入搜索服务 [27] 以及面向向量数据库管理系统部署的软件测试与可靠性路线图 [28]。这些贡献主要针对查询处理、索引和执行计划。Memelang 针对的是另一个瓶颈：面向生成器的查询中间表示，即大语言模型能够可靠生成、同时仍可编译为混合执行计划的查询中间表示。其设计目标是以紧凑且可确定性解析的形式表达混合约束（符号过滤器、连接、聚合和向量谓词），而不是提出新的向量索引或优化器技术。在实际部署中，Memelang 的编译目标可以与提供向量运算符的 SQL 后端对接（例如通过 PostgreSQL 扩展 `pgvector`） [29]。

2.5 面向工具使用型智能体和 API调用的中间表示

大语言模型工具使用研究通常将结构化输出定义为 API调用，其参数必须符合预定义的模式。工具使用可以通过监督数据或自监督数据来学习或改进，这些数据教会模型使用正确的参数调用工具 [30],；也可以通过检索增强方法，将 API 的使用建立在文档基础之上 [31]。ReAct 等智能体提示方法将推理与行动交替进行，以便与外部系统交互 [32]。在实践中，许多工具 API 采用类似 JSON 模式的参数规范，而商业函数/工具调用接口则明确提供模式约束的参数生成 [33]。Memelang 不同于这些模式对象方法，它采用线性坐标类型中间表示，而非嵌套键值结构。其贡献在于提供了一种紧凑的表示：分隔符诱导的坐标和固定槽位角色减少了结构歧义，同时仍兼容基于语法或模式的受限解码技术。

3 轴向语法

轴向语法是一种线性词元语法，其中少量分级分隔符在原本扁平的词元流上诱导出一个坐标系。解析通过一次从左到右扫描完成。每个非分隔符词元都会在一个 n 维格点中获得一个坐标。随后，具体实现将坐标位置解释为语义角色。语法将词元映射到坐标，语义则将坐标映射到角色。

下一页将给出轴向语法的形式化描述。

分词。固定一个元数 $n \in \mathbb{N}_{>0}$。令 Σ 为一个令牌字母表，它被划分为分级分隔符 S 和非分隔符 A：

$$\Sigma = S \dot\cup A.$$

每个分隔符都有一个由全函数给出的秩

$$\rho : S \to \{0, 1, \dots, n-1\}.$$

扫描状态与坐标分配。对于令牌流 $\tau = (t_1, \dots, t_T) \in \Sigma^T$，定义扫描状态 $\mathbf{i}^{(k)} \in \mathbb{N}^n$，其中 $\mathbf{i}^{(0)} = \mathbf{0}$ 和

$$\mathbf{i}^{(k)} \;=\; \begin{cases} U_{\rho(t_k)}\big(\mathbf{i}^{(k-1)}\big) & \text{if } t_k \in S, \\ \mathbf{i}^{(k-1)} & \text{if } t_k \in A, \end{cases}$$

其中对于 $\mathbf{i} = (i_{n-1}, \dots, i_0)$ 和秩 r，

$$\big(U_r(\mathbf{i})\big)_m \;=\; \begin{cases} i_m & m > r, \\ i_r + 1 & m = r, \\ 0 & m < r. \end{cases}$$

每个非分隔符词元的（原始）坐标为

$$\phi_\tau(k) \;=\; \mathbf{i}^{(k)} \qquad \text{for all } k \text{ with } t_k \in A.$$

单元格化（诱导网格）。定义（原始）单元格内容映射 $C_\tau :\ \mathbb{N}^n \to A^*$，如下：

$$C_\tau(\mathbf{x}) \;=\; \langle\, t_k \;:\; t_k \in A \;\wedge\; \phi_\tau(k) = \mathbf{x} \,\rangle,$$

同时保留流顺序。空单元格具有 $C_\tau(\mathbf{x}) = \epsilon$。

可选地，某个实施例可以应用（可能依赖于 τ 的）双射重索引

$$\pi_\tau : \mathbb{N}^n \to \mathbb{N}^n,$$

从而得到有效坐标 $\phi^\pi_\tau(k) = \pi_\tau(\phi_\tau(k))$ 和有效单元格

$$C^\pi_\tau(\mathbf{x}) = C_\tau\big(\pi_\tau^{-1}(\mathbf{x})\big).$$

为方便记法，当 τ 明确时，我们可以写作 π。

切片局部反转（反向索引）。为了支持沿秩 r方向、在每个固定高阶前缀内进行右对齐或反向索引，允许重索引映射依赖于 τ。对于$\mathbf{x} = (x_{n-1}, \dots, x_0)$，令高阶前缀为$\mathbf{x}_{>r} = (x_{n-1}, \dots, x_{r+1})$，并定义切片宽度

$$W^\tau_r(\mathbf{u}) = \begin{cases} 0 & \text{if } \nexists\, \mathbf{y} \text{ with } \mathbf{y}_{>r} = \mathbf{u} \text{ and } C_\tau(\mathbf{y}) \neq \epsilon, \\ 1 + \max\{\, y_r : C_\tau(\mathbf{y}) \neq \epsilon \wedge \mathbf{y}_{>r} = \mathbf{u} \,\} & \text{otherwise.} \end{cases}$$

然后定义秩-r 反转重索引 $\pi^{\mathrm{rev},r}_\tau :\ \mathbb{N}^n \to \mathbb{N}^n$ 如下

$$\big(\pi^{\mathrm{rev},r}_\tau(\mathbf{x})\big)_m = \begin{cases} x_m & m \neq r, \\ W^\tau_r(\mathbf{x}_{>r}) - 1 - x_r & m = r \;\wedge\; x_r < W^\tau_r(\mathbf{x}_{>r}), \\ x_r & m = r \;\wedge\; x_r \geq W^\tau_r(\mathbf{x}_{>r}). \end{cases}$$

此映射是一个对合（因此是双射），所以 $(\pi^{\mathrm{rev},r}_\tau)^{-1} = \pi^{\mathrm{rev},r}_\tau$。

具身解释。 一个轴向语法实例由一个（可能是不完全的）单元格解释器补全

$$g : A^* \rightharpoonup \mathcal{X},$$

生成坐标索引表示（偏映射）

$$E : \mathbb{N}^n \rightharpoonup \mathcal{X}, \qquad E(\mathbf{x}) = \begin{cases} g(C_\tau^\pi(\mathbf{x})) & C_\tau^\pi(\mathbf{x}) \neq \epsilon, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

若有定义，则写作 $E(\mathbf{x})\downarrow$；若未定义，则写作 $E(\mathbf{x})\uparrow$。

向前继承闭包。 **</style}对于秩，令 $\mathbf{e}_r \in \mathbb{N}^n$ 为第 r 个单位向量。定义前向传递（继承）补全 $\mathrm{cf}_r(E)$ 如下:**

$$\mathrm{cf}_r(E)(\mathbf{x}) = \begin{cases} E(\mathbf{x}) & E(\mathbf{x})\downarrow, \\ \mathrm{cf}_r(E)(\mathbf{x} - \mathbf{e}_r) & E(\mathbf{x})\uparrow \ \wedge\, x_r > 0, \\ \bot & \text{otherwise,} \end{cases}$$

其中 $\bot$ 表示继承后缺失。

相对引用。 令 $R \subseteq A$ 为指定的引用原子，其偏移量为

$$\Delta : R \to \mathbb{Z}^n.$$

对于 $\mathbf{x} \in \mathbb{N}^n$ 和 $a \in R$，定义（当 $\mathbf{x} + \Delta(a) \in \mathbb{N}^n$ 时）

$$\mathrm{res}(a, \mathbf{x}; E) \;=\; \mathrm{cf}_r(E)\big(\mathbf{x} + \Delta(a)\big),$$

对于由具体实施方式选定的继承秩 r （或者将 cf_r 替换为任意选定的规范化算子）。

变量绑定（单遍环境）。设 $\mathcal{V}$ 为变量名集合。设 $\prec$ 按首次出现的顺序排列有效坐标:

$$\mathbf{x} \prec \mathbf{y} \iff \min\{k : t_k \in A \wedge \phi_\tau^\pi(k) = \mathbf{x}\} < \min\{k : t_k \in A \wedge \phi_\tau^\pi(k) = \mathbf{y}\}.$$

绑定策略 是以从左到右的方式，在按 $\prec$排序的单元格上构造环境 $\beta : \mathcal{V} \rightharpoonup \mathbb{N}^n$ 的任意方法 :

$$\beta_{\text{new}} \;=\; \beta_{\text{old}} \cup \{\, v \mapsto \mathbf{x} \;:\; v \text{ is bound at } \mathbf{x} \,\}.$$

对 v 的引用表示 $E(\beta(v))$（若 $\beta(v)$ 未定义，则会报错）。

这将轴向语法定义为：(i) 一个分隔符排序流转换器 ϕ_τ，用于生成 n 维坐标格；以及 (ii) 一个具身映射 g （以及可选的规范化处理，例如 cf_r、相对偏移量Δ和绑定 β）。为确保可定型性，Axis:r:i 表示秩为 r 时的坐标索引 $i_r = i$。

4 Memelang 语法与语义

Memelang 以三个秩（矩阵、向量、限制）实例化一种轴向语法，如表 1 所示；并在限制轴上采用固定的语义角色（表、列、值），其索引如表2所示。

表1：Memelang 轴秩映射。

Rank	Name	分隔符
轴：2	矩阵	双分号 (;;)
轴：1	向量	单个分号（;）
轴：0	限制	空白字符（\s+）

对于 Axis:0 分隔，连续一个或多个空白字符会被词法分析为单个分隔符标记。向量内部开头和结尾的空白字符会被规范化。

标记语义角色取决于其恢复出的坐标，因此无需依赖子句关键字、括号或层级作用域标记。在每个向量中，Axis:0 索引均右对齐，使最右侧的限制占据 Axis:0:0，无论向量长度如何。Axis:0:2 是表槽，Axis:0:1 是列槽位，Axis:0:0 是值槽位（表 2）。

表 2：Memelang Axis:0 索引映射。

索引	Role
Axis:0:2	表标识符
轴:0:1	列标识符
轴:0:0	值项 / 谓词

Memelang 的表层语法使用一小组保留标记来表示通配符、相对引用、比较和析取：

- **通配符**。 单独的下划线（ ）表示通配符原子，可匹配任意槽位中的任意值。
- 向量同值。 </style}@ 符号表示当前矩阵中前一个向量的相同限制索引所解析出的值。
- **矩阵同值**。 {@v4} 表示前一个矩阵中最后一个向量的相同限制索引所解析出的值。
- **比较运算符**。 比较运算符标记（=、!=、 >、 >=、 <、 <=、 ~、!~）可选地控制求值期间的约束交集；省略运算符时，默认使用相等运算符（=）。
- **字面量**。 ALNUM(joe)、QUOT ("Joe Smith")、INT (5) 和 DEC (5.1) 等标记表示字面原子。
- **析取**。 逗号（,）在逻辑上对同一表达式中的替代值进行析取（OR）。

Axis:0:1 列槽位中的所有列都会被投影。Memelang 允许 Axis:0 表槽位和列槽位中出现非固定标记，使单个值谓词能够覆盖多个列和/或表。在 Memelang 中，这在语法上有效，但当前数据库引擎普遍不支持。实际上，在列槽位中使用通配符，或在值槽位中使用仅包含通配符的向量，即可投影表中的所有列。

FUNC 注解将分组、聚合和排序附加到值槽位。从语法上看，FUNC 是一个以冒号为前缀的标签链，置于 Axis:0:0 项的开头（例如，:min:asc>=4.2）。聚合标签（:sum、:cnt、:min、:max、:avg、:last）会编译为针对向量列的 SQL 聚合表达式；:grp 标记分组键；:asc 和 :des 指定对（可能经过聚合的）表达式进行排序。

形式为 :$x 的绑定标签会在引入处附加到值槽位；同一矩阵中后续出现的 $x 将被解释为对已绑定坐标表达式的引用。未绑定变量将被视为错误。

4.1 Memelang 查询语法

下面展示了适用于查询模式的简洁扩展巴科斯-诺尔范式风格片段：

列表 1：最简扩展巴科斯-诺尔范式（查询模式）。

```
query := (matrix ";;")+ ;
matrix := vector (";" vector)* ;
vector := limit (WS limit)* ;
limit := left | right | (cmp right) | (left cmp right) ;
left := (term)? (":" func)* ;right := term ("," term)* ;
term := atom | (mod atom) | (atom mod atom) ;
atom := ALNUM | QUOT | INT | DEC | "_" | "@" | "$var"
| EMB ;
cmp := "=" | "!=" | "<" | "<=" | ">" | ">=" | "~" |
"!~" ;
mod := "<->" | "<#>" | "<=>" | "+" | "-" | "*" | "/"
| "%" | "**" ;
```

5 Memelang 示例

5.1 带前向传递的标量过滤器

请看以下 Memelang 查询：

```
movies year <1970; title _;;
```

此矩阵包含两个由 ; 分隔的向量，并以 ;; 终止。在第一个向量中，三个限制槽位（表、列、值）分别实例化为：表为电影，列为年份，值谓词为<1970。在第二个向量中，表槽被省略，并继承自前一个向量；列槽为标题，值槽位为通配符，表示投影标题且不附加任何限制条件。 -

表3：示例1坐标表示为（轴:2，轴:1，轴:0）。

表达式	坐标	Role
电影	(0, 0, 2)	表
year	(0, 0, 1)	列
<1970	(0, 0, 0)	值 / 谓词
(@)	(0, 1, 2)	表（沿用）
标题	(0, 1, 1)	列
-	(0, 1, 0)	值（通配符）

将此 IR 编译为 PostgreSQL SQL 后得到：

```
SELECT t0.year, t0.title FROM movies AS t0 WHERE t0.
year < 1970;
```

5.2 带余弦距离和排序的双列输出

以下查询返回 (i) 针对 movies.description 与文本提示计算的余弦距离表达式，以及 (ii) 电影标题。: asc 标签按距离表达式升序排列结果（距离最近的排在最前）。

```
movies description <=>"robot":asc<0.35;title _;;
```

第一个向量设置表槽=movies、列=descr 并且值槽位编码了一个余弦距离表达式 <=>"robot", 带有内联排序标签 :asc，并按阈值 <0.35 进行筛选。第二个向量投影标题，并继承前一个向量的表槽。

在参考实现中，嵌入钩子会将引号中的文本替换为向量字面量；此处 [...] 表示一个嵌入。

```
SELECT (t0.description <=> '[...]'::VECTOR), t0.title
FROM movies AS t0
WHERE (t0.description <=> '[...]'::VECTOR) < 0.35
ORDER BY (t0.description <=> '[...]'::VECTOR) ASC;
```

5.3 绑定的合作演员自连接

一种常见的共同主演查询可以表示为对 roles.movie 的自连接，并以绑定的演员姓名作为锚点。下面的 Memelang 查询将锚点姓名绑定到 $a，通过 @ @ @ 在电影列上连接 roles 的第二个实例，并通过 !=$a 排除锚点演员。

```
roles actor :$a="Bruce Willis";movie _;
@ @ @;actor !=$a;;
```

这里，第一个向量约束 roles.actor，并将引入的值绑定到 $a。第二个向量投影 roles.movie。第三个向量 @ @ @ 引入前一个表的新实例，并将其电影列与前一个向量的电影值设为相等。表槽中的 @ 会触发自连接。最后一个向量从连接后的实例中投影合作演员的姓名，同时排除 $a。

SQL呈现如下：

```
SELECT t0.actor,t0.movie,
t1.movie,t1.actor
FROM roles AS t0, roles AS t1
WHERE t0.actor = 'Bruce Willis'
AND t1.movie = t0.movie
AND t1.actor != t0.actor;
```

5.4 带向量谓词和元限制的分组连接

以下查询针对 1980 年之前上映的“战争”主题电影，并返回按该电影观测到的最低角色评分排名的前 12 个标题：

```
% War stories before 1980: top 12 movies by minimum
role rating movies year <1980;description <=>"war"<=\$
ption,title :grp;roles movie @;rating :min:des;%m lim 12;;
```

该矩阵包含五个查询模式向量，后接一个 Meta 模式向量：

- **向量 1**（电影 年份 <1980）。引入电影表，并按 movies.year < 1980进行筛选。
- **向量 2**（描述 <=>"战争"<=$sim）。沿用表槽（电影），将列设为描述，并约束行内余弦距离表达式，使其相对于运行时参数 $sim 与嵌入提示"战争"进行比较。
- **向量 3**（标题 :grp）。投影 movies.title，并通过 :grp 将其标记为分组键。

• **Vector 4**（roles movie @）。引入第二个表实例 roles，并通过将 roles.movie 设为 @ 来进行连接，即设为前一个向量的值槽位（此处为分组后的 movies.title）。

• **Vector 5**（评分 :min:des）。保留 roles，并对 roles.rat 应用聚合函数 :min
并通过 :des 按此聚合结果降序排序。

• **Meta**（%m lim 12）。设置编译时限制（LIMIT 12），而不引入额外的查询谓词。

在参考编译器中，分组查询要求每个投影表达式要么是分组键（:grp），要么是聚合函数。未明确标记的表达式会被指定为默认聚合函数（:last），以满足 SQL 的 GROUP BY 要求；此策略不会改变先前的筛选谓词。

编译为 PostgreSQL SQL 后，结果如下：

```
SELECTMAX(t0.year),
MAX(t0.description <=> [...]),t0.title,
MAX(t1.movie),MIN(t1.rating)
FROM movies AS t0, roles AS t1
WHERE t0.year < 1980
AND (t0.description <=> [...]) <= $sim
AND t1.movie = t0.titleGROUP BY t0.title
ORDER BY MIN(t1.rating) DESCLIMIT 12;
```

这里，嵌入钩子为“war”提供向量参数，而 $sim 则被编译为由运行时提供的参数（也可通过元指令覆盖）。

6 结论

Memelang 提出了一种轴向语法，作为面向大语言模型的神经符号中间表示，用于统一的向量关系查询。其核心设计——结合前向传递和绑定的张量形状抽象语法树——减少了冗余，提高了解析的确定性，并为编译到 PostgreSQL 的参数化 SQL 提供了清晰路径。通过将符号约束（表、列、连接、聚合）与稠密向量语义（距离运算符和嵌入钩子）相结合，Memelang 旨在弥合当前大语言模型到数据库工作流中的一个实际空缺：在约束生成条件下兼顾紧凑性与正确性。

网站： https://www.memelang.net/
代码仓库：
https://github.com/memelang-net/memesql9

专利： 美国专利 12,475,098，另有专利申请正在审理中。

参考文献

ing。[3] Torsten Scholak、Nathan Schucher、Dzmitry Bah- danau。《PICARD：用于语言模型受限自回归解码的增量解析》。arXiv 预印本 arXiv:2109.05093，2021年。https://arxiv.org/abs/2109.05093。[2] Saibo Geng、Martin Josifoski、Maxime Peyrard、Robert West。《无需微调的结构化自然语言处理任务语法约束解码》。arXiv 预印本 arXiv:2305.13971，2023年。https: //arxiv.org/abs/2305.13971。[3]Debangshu Banerjee、Tarun Suresh、Shub- ham Ugare、Sasa Misailovic、Gagandeep Singh。《CRANE：通过受限的大语言模型生成进行推理》。arXiv 预印本 arXiv:2502.09061，2025年。https://arxiv.org/abs/2502.09061。[4]Lingxiao Li、Salar Rahili、Yiwei Zhao。《通过受限解码实现正确性有保证的代码生成》。arXiv 预印本 arXiv:2508.15866，2025年。h t t p s : //arxiv.org/abs/2508.15866。[5] Saibo Geng、Hudson Cooper、Micha-l Moskal、Samuel Jenkins、Julian Berman、Nathan Ranchin、Robert West、Eric Horvitz、Harsha Nori。《从语言模型生成结构化输出：基准测试与研究》。arXiv 预印本 arXiv:2501.10868，2025 年。https://arxiv.org/abs/2501.10868。[6] Jiaqi Guo、Zecheng Zhan、Yan Gao、Yan Xiao、Jian-Guang Lou、Ting Liu、Dongmei Zhang。《利用中间表示实现跨领域数据库中的复杂文本到 SQL》。ACL，2019年。https://aclanthology.org/P19-1444/。[7] Yujian Gan、Xinyun Chen、Jinxia Xie、Matthew Purver、John R. Woodward、John Drake、Qiaofu Zhang。《Natural SQL：让从自然语言规范推断 SQL 更容易》。EMNLP 论文集，2021年。https://aclanthology.org / 2021.findings-emnlp.174/。[8] Bailin Wang、Richard Shin、Xiaodong Liu、Oleksandr Polozov、Matthew Richardson。《RAT-SQL：面向文本到 SQL 解析器的关系感知的模式编码与链接》。ACL，2020年。https: // aclanthology.org/2020.acl-main.677/。

[9]Ohad Rubin、Jonathan Berant。*SmBoP*：半自回归自底向上的语义解析。NAACL，2021年。https://aclanthology.org/2021.naacl-main.29/。

[10] ShengminPiao、Jieun Lee、Sanghyun Park。*LitE-SQL*：一种轻量高效的文本到 *SQL* 框架，采用基于向量的模式链接和执行引导的自我纠错。arXiv 预印本 arXiv:2510.09014，2025年。https://arxiv.org/abs/2510.09014。

[11] Shuai Lyu、Haoran Luo、Ripeng Li、Zhonghong Ou、Jiangfeng Sun、Yang Qin、Xiaoran Shang、Meina Song、Yifan Zhu。SQL-o1：一种用于文本到 SQL 的自奖励启发式动态搜索方法。arXiv 预印本 arXiv:2502.11741，2025年。ht tps://arxiv.org/abs/2502.11741。[12] Bohan Zhai、Canwen Xu、Yuxiong He、Zhewei Yao。ExCoT：借助执行反馈优化文本到 SQL 推理。arXiv 预印本 arXiv:2503.19988，2025年。https://arxiv.org/ abs/2503.19988。[13]Tao Yu 等。Spider：用于复杂跨领域语义解析和文本到SQL任务的大规模人工标注数据集。EMNLP，2018年。https://arxiv.org/abs/1809 .08887。[14]Dzmitry Pihulski、Karol Charchut、Viktoria Novogrodskaia、Jan Koco´n。LLMSQL：为大语言模型时代的文本到SQL升级 WikiSQL。arXiv 预印本 arXiv:2510.02350，2025年。https: // arxiv.org/abs/2510.02350。[15] Khanh Trinh Pham、Thu Huong Nguyen、Jun Jo、Quoc Viet Hung Nguyen、Thanh Tam Nguyen。多语言文本到SQL：通过协作语言智能体对语言模型的能力边界进行基准测试。arXiv 预印本 arXiv:2509.24405，2025年。https://arxiv.org/ abs/2509.24405。[16] Zijin Hong、Zheng Yuan、Qinggang Zhang、Hao Chen、Junnan Dong、Feiran Huang、Xiao Huang。下一代数据库接口：基于大语言模型的文本到SQL综述。arXiv 预印本 arXiv:2406.08426，2024年。https://arxiv.org/ abs/2406.08426。[17] Xinyu Liu、Shuyu Shen、Boyan Li、Peixian Ma、Runzhi Jiang、Yuxin Zhang、Ju Fan、Guoliang Li、Nan Tang、Yuyu Luo。大语言模型时代的文本到SQL综述：现状如何，未来将走向何方？arXiv 预印本 arXiv:2408.05109，2024年。https://arxiv.org/abs/2408.05109。

[18]Xiaohu Zhu、Qian Li、Lizhen Cui、Yongkang Liu。《大语言模型增强的文本到 SQL 生成：综述》。arXiv 预印本 arXiv:2410.06011，2024年。https://arxiv.org/abs/2410.06011。[19] Pengcheng Yin、Graham Neubig。《通用代码生成的句法神经模型》。ACL，2017年。https://aclanthology.org/P17-1041/。[20] Zhengren Wang、Dongwen Yao、Bozhou Li、Dongsheng Ma、Bo Li、Zhiyu Li、Feiyu Xiong、Bin Cui、Linpeng Tang、Wentao Zhang。《Text2VectorSQL：迈向向量搜索与 SQL 查询的统一接口》。arXiv 预印本 arXiv:2506.23071，2025年。https://arxiv.org/abs/2506.23071。[21] Rui Ma、Kai Zhang、Zhenying He、Yinan Jing、X. Sean Wang、Zhenqiang Chen。《CHASE：面向结构化与非结构化数据混合查询的原生关系型数据库》。arXiv 预印本 arXiv:2501.05006，2025年。https://arxiv.org/abs/2501.05006。[22]Hyunjoon Kim、Chaerim Lim、Hyeon-jun An、Rathijit Sen、Kwanghyun Park。《Exqutor：面向向量增强分析查询的扩展查询优化器》。arXiv 预印本 arXiv:2512.09695，2025年。https://arxiv.org/abs/2512.09695。[23] Jingyi Yang、Songsong Mo、Jiachen Shi、Zihao Yu、Kunhao Shi、Xuchen Ding、Gao Cong。《ARCADE：跨多种数据模态的混合与连续查询处理实时数据系统》。arXiv 预印本 arXiv:2509.19757，2025年。https://arxiv.org/abs/2509.19757。[24] Joydeep Chandra、Satyam Kumar Navneet、Yong Zhang。《混合多模态图索引（HMGI）：集成关系型搜索与向量搜索的综合框架》。arXiv 预印本 arXiv:2510.10123，2025年。https://arxiv.org/abs/2510.10123。[25]James Jie Pan、Jianguo Wang、Guoliang Li。《向量数据库管理系统综述》。arXiv 预印本 arXiv:2310.14021，2023年。https://arxiv.org/abs/2310.14021。[26] Yanjun Lin、Kai Zhang、Zhenying He、Yinan Jing、X. Sean Wang。《向量-标量混合数据上的过滤式近似最近邻搜索综述》。arXiv 预印本 arXiv:2505.06501，2025年。https://arxiv.org/abs/2505.06501。

[27] Guoyu Hu、Shaofeng Cai、Tien Tuan Anh Dinh、Zhongle Xie、Cong Yue、Gang Chen、Beng Chin Ooi。*HAKES*：用于嵌入搜索服务的可扩展向量数据库。arXiv 预印本 arXiv:2505.12524，2025年。https://arxiv.org/abs/2505.12524。

[28] Shenao Wang、Yanjie Zhao、Yinglin Xie、Zhao Liu、Xinyi Hou、Quanchen Zou、Haoyu Wang。迈向可靠的向量数据库管理系统：面向 *2030* 年的软件测试路线图。arXiv 预印本 arXiv:2502.20812，2025年。https://arxiv.org/abs/2502.20812。[29] pgvector 贡献者。*pgvector*：适用于 *Postgres* 的开源向量相似性搜索。GitHub 代码仓库。https://github.com/pgvector/pgvector。[30] Timo Schick、Jane Dwivedi-Yu、Roberto Dessi、Roberta Raileanu、Maria Lomeli、Eric Hambro、Luke Zettlemoyer、Nicola Cancedda、Thomas Scialom。*Toolformer*：语言模型可以教会自己使用工具。NeurIPS，2023年。https://papers.nips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html。[31]Shishir G. Patil、Tianjun Zhang、Xin Wang、Joseph E. Gonzalez。*Gorilla*：与海量 *API* 相连的大语言模型。NeurIPS，2024年。https://papers.nips.cc/paper_files/paper/2024/hash/e4c61f578ff07830f5c37378dd3ecb0d-Abstract-Conference.html。[32] Shunyu Yao、Jeffrey Zhao、Dian Yu、Nan Du、Izhak Shafran、Karthik Narasimhan、Yuan Cao。*ReAct*：在语言模型中协同推理与行动。ICLR，2023年。https://openreview.net/pdf?id=WE_vluYUL-X。[33] OpenAI。函数调用——*OpenAI API* 文档。在线文档。https://platform.openai.com/docs/guides/function-calling。