多模型数据的范畴演算与代数

Jiaheng Lu赫尔辛基
大学，芬兰
jiaheng.lu@helsinki.fi

多模型数据库旨在同时存储、管理和查询采用多种模型的数据，例如关系型数据、层次型数据和图数据。本文在范畴范式下，为多模型数据库的查询提供理论基础。我们提出两种形式化查询语言：范畴演算和范畴代数，分别将关系演算和关系代数扩展至多模型数据库。我们证明这两种查询语言等价。为便于查询优化，我们提出一系列范畴代数的变换规则。最后，我们分析所提出查询语言的表达能力和计算复杂度。

1 引言

数据的“多样性”是现代数据管理系统中的重要议题之一 [17, 11]。在各种应用中，数据源天然呈现出多样的组织结构和格式，例如关系、图、XML、JSON 等。为提供统一的视图和查询接口，研究者提出了范畴模型（例如 [2, 15, 28]），以定义一种形式化的统一数据模型，从而有效容纳各种不同的数据模型。

本文在范畴范式下提出了一个全面的查询框架。通过引入范畴代数和范畴演算，我们为多模型查询语言建立了理论基础。回顾一下，关系演算和关系代数是用于在关系数据库上表达查询的两种理论语言。它们提供了一种构造查询的方法，用于检索和操作存储在关系数据库系统中的数据。类似地，本文为范畴数据库开发了范畴演算和范畴代数。范畴演算是一种声明式语言，用于描述范畴中所需对象和态射的属性；而范畴代数是一种过程式语言，提供用于操作范畴并从中检索特定对象和态射的运算。

本文的贡献和组织结构如下：

(i) 我们引入了用于在范畴数据库上构造查询的范畴演算。该演算通过纳入多模型谓词扩展了关系定义域演算。这些谓词支持构造诸如图数据中的可达性查询，以及 XML 数据中针对父子关系和祖先-后代关系的枝模式匹配等查询（第 3 节）。

（ii）我们提出了同时包含集合运算和范畴运算的范畴代数。集合运算涉及对单个或多个集合进行操作，而范畴运算则支持集合、函数和范畴之间的转换。范畴代数能够处理范围广泛的查询，包括关系查询、XML枝模式匹配、图模式匹配，以及图上的可达性查询。我们建立了一个定理，证明范畴演算与范畴代数之间的等价性。具体而言，每个范畴演算都可以使用一组范畴代数算子来表示，反之亦然（第4节）。

A. Hadzihasanovic& J.-S. P. Lemay：应用范畴论 2025（ACT 2025），EPTCS 442，2026 年，第 75–90 页，
doi:10.4204/EPTCS.442.6

© Jiaheng Lu 本作品采用知识共享署名许可协议授权。

（iii）我们定义了一系列代数表达式的变换规则，将范畴代数表达式重写为更高效的形式，以进行查询优化。我们分析了所提出查询语言的表达能力和计算复杂度（第5节）。

相关工作。本研究与两类既有研究相关：（1）数据库查询语言的应用范畴论；（2）面向各种数据类型的查询演算与代数。

尽管范畴论最初是作为一种抽象的数学语言发展起来，旨在统一代数与拓扑学，但研究者已探索了它在多个领域的应用，其中包括数据库查询语言。Tannen（1994） [26] 指出，范畴论中的概念可以催生一系列适用于集合类型的实用语言，例如集合、多重集、列表、复杂对象、嵌套关系、数组以及某些类型的树。Libkin（1995） [13] 运用泛性质研究带有近似的查询语言语法。Libkin 和 Wong（1997） [14] 展示了集合上的数据库操作与单子的范畴论概念之间的联系。近来，Schultz 和 Spivak（2016） [24] 提出了一种范畴查询语言，可用作数据集成脚本语言。Gibbons 等人（2018） [6] 运用单子和伴随等范畴论概念，为数据库查询语言设计提供了优雅的数学基础。这些研究展现了范畴论与数据库查询语言之间引人入胜的联系，揭示了范畴论如何为数据库查询语言的设计提供启发并加以改进。

自 Codd（1972） [3] 证明关系代数与关系演算的安全公式能够表达同一类数据库查询以来，查询代数与演算便成为数据库研究的核心课题。关于不同类型数据的查询代数与演算，已有大量相关研究，涵盖关系数据 [3, 29, 19, 20], 嵌套关系数据 [21, 8], 图数据[5, 9], 面向对象数据 [12], 不完整数据和概率数据 [25, 7], 等。这些先前的论文为数据库系统处理各种类型的数据奠定了基础。本文进一步推进这一领域的发展，提出适用于范畴数据的查询演算与代数，并将其应用于多模型数据库。

2 预备知识

数据库可以视为一个范畴，具体而言，是一个集合范畴。在此语境下，范畴中的每个对象代表一个集合，而每个态射对应于两个集合之间的一个函数。态射的复合与函数的复合相对应。在本文余下部分中，我们将“集合”与“对象”交替使用，也将“函数”与“态射”交替使用。此范畴中有三种基本类型的对象：实体对象、属性对象和关系对象。例如，图1展示了电子商务中三种数据（关系数据、XML 和图数据）的统一范畴表示，其中包含客户、社交网络和订单信息，采用了三种不同的数据模型。客户信息存储在关系表中，包括其标识符、名称和信用额度。图数据包含客户之间相互关系的信息，即谁认识谁。在 XML 文档中，每个订单都有一个标识符和一系列有序商品，其中每件商品都包含产品名称。这三种类型的数据可以整合到图1（d）中的统一范畴模式中，其中“认识”是一个关系对象，“客户”“订单”和“Product”是实体对象，而“标识符”“客户名称”和“产品名称”是属性对象。此外，本文讨论的范畴并非任意集合范畴，而是一个薄范畴，定义如下：

定义 1。(薄范畴或偏序范畴) [22] 给定范畴$\mathscr{C}$中的一对对象X 和Y，以及任意两个态射f、g: $X \to Y$，当且仅当态射f 和g 相等时，我们称 $\mathscr{C}$为薄范畴（或偏序范畴）。

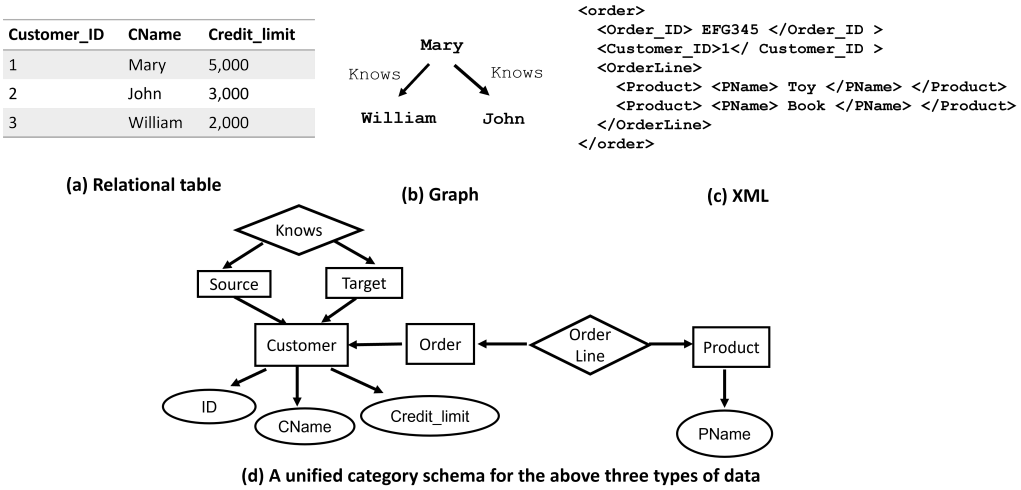

图1：这个简单示例展示了三种数据类型，包括关系、XML 和图。它们具有统一的范畴表示。

在范畴论中，交换图是用图形表示对象与态射之间关系的一种方式。图表是交换的，意为沿图表中不同路径得到的结果相同。薄集范畴中的所有图表都是交换的 [22]。

3 范畴演算

范畴上的查询是从一个范畴 $\mathscr{C}$ 到另一个范畴 $\mathscr{D}$ 的映射。在接下来的两节中，我们将定义两种形式化查询语言：(1) 范畴演算，这是一种用于描述 $\mathscr{D}$ 中结果的声明式语言；(2) 范畴代数，这是一种用于列出对 $\mathscr{C}$ 执行的操作的过程式语言。

范畴演算的字母表列于表1。给定一个范畴 $\mathscr{C}$，令 S_i 表示 $\mathscr{C}$ 中的一个对象，令 $f: S_i \to S_j$ 表示 S_i 与 S_j 之间的一个函数。给定任意元素 $x \in S_i$，有两种方式表示 x 在 f 作用下的像 y：(i) $y = f(x)$；(ii) $y = x \cdot S_j$。第二种表达方式没有歧义，因为在薄范畴中，任意两个对象之间都只有一个函数（参见定义1）。此外，为了表示函数的复合，考虑两个函数 $f: S_i \to S_j$ 和 $g: S_j \to S_k$。给定任意元素 $x \in S_i$，x 在 f 和 g 的复合作用下的像 y 可写作 $y = (g \circ f)(x)$，或 $y = x \cdot S_j \cdot S_k$。例如，考虑图1。假设 x 是 OrderLine 的一个元素，那么 x 的产品名称 y 表示为 $y = x \cdot \text{Product} \cdot$ 名称。

然后，我们定义三类项：值域项、函数项和谓词项 ms.

值域项：范围项 $x \in S$ 表明元素 x 属于集合 S。

函数项：给定两个变量 $x_i \in S$ 和 $x_j \in S'$，函数项 $(f: x_i \to x_j) = g_n \circ \cdots g_2 \circ g_1$ 表明 f 是从 S 到 S' 的函数，且 f 是函数序列 g_1 , g_2, ...和 g_n 的复合。

谓词项：设 x_i 和 x_j 为变量， α 为个体常量， θ 为谓词符号。表达式 $x_i \theta x_j$ 和 $x_i \theta \alpha$ 称为谓词项。谓词可分为三类：

1. **经典谓词**（θ_M）：包括标准的数学比较符号，例如=、≠、<、>、≤、≥，用于比较数值或字符数据。

表1：范畴演算字母表

记号	示例
对象常量	$S_1, S_2 \cdots$
态射常量	$f_1: \ S_1 \to S_2 = g_1 \circ g_2 \cdots$
对象中的元素变量	$x_1 \in S_1, x_2 \in S_2 \cup S_3 \cdots$
逻辑符号	$\exists, \forall, \wedge, \vee, \neg$

2. **树数据谓词**（θ_T）：这些谓词专为树数据结构上的操作而设计。考虑一个描述树数据 $\mathscr{T}$ 的范畴，其中每个树节点都分配有一个杜威码[27, 18]，用于表示其在 $\mathscr{T}$ 中的位置。在杜威编码中，每个节点都由一个向量表示：（i）根节点标记为空字符串 ε；（ii）对于非根节点 u，label(u) = label(s).x，其中 u 是 s 的第 x 个子节点。杜威编码有助于高效计算元素之间的结构关系。例如，如果节点 u 的标签为“1.2.3.4”，那么其父节点的标签为“1.2.3”，祖父节点的标签为“1.2”。

谓词 θ_T 可以包括 isParent、isChild、isAncestor、isDescendant、isFollowing、isSibling、isFollowing-sibling、isPreceding 和 isPreceding-sibling。所有这些 XPath 轴 [4] 都可以通过比较这两个杜威编码来确定。

3. **图数据谓词 (θ_G)**：这些谓词专为图数据结构上的操作而设计。考虑一个表示图 $\mathscr{G}$的范畴，其中所有边都存储在集合 E 中。在此上下文中，若 a 和 b 是 $\mathscr{G}$中的两个节点变量，则记号 $a \rightsquigarrow^E b$ 表示节点 b 可通过 E中的边从节点 a 到达。此外，$a \rightsquigarrow_n^E b$ 表示节点 b 可在 n 跳数内从节点 a 到达。

范畴演算中的（良构）公式递归定义如下 :

1. 任何项（包括范围项、函数项和谓词项）都是公式；2. 如果 α 是公式，那么 $\neg\alpha$也是公式；3. 如果 α_1, α_2 是公式，那么 $(\alpha_1 \vee \alpha_2)$ 和 $(\alpha_1 \wedge \alpha_2)$ 也是公式；4. 如果 α 是一个公式，且 x 在其中作为自由变量出现，那么 $\exists x(\alpha)$ 和 $\forall x(\alpha)$ 都是公式。

表 2 给出了若干公式示例。此处采用自由变量和约束变量出现的标准定义。公式中的自由变量是指未被公式内任何量词量化的变量；它不受诸如 $\forall$ 或 $\exists$ 这样的逻辑运算符约束。相反，约束变量是指被公式内某个量词量化的变量。

范畴演算中的表达式通常具有以下形式：

$$Z = \{X, \mathscr{R} \mid W\}$$

其中 (1) $X = x1$，……，x_m 是 m 个互不相同的变量，表示实体对象或属性对象 ；
(2) $\mathscr{R} = R1$，……，R_n，其中每个 $R_i = (x_{i_1}$，……，$x_{i_j})$ 表示一个具有 j 个分量的关系对象。X 或 $\mathscr{R}$ 可以为空，但不能同时为空；
(3) $W = U_1 \wedge U_2 \wedge \ldots \wedge U_p \wedge F_1 \wedge F_2 \wedge \ldots \wedge F_k \wedge V$，其中 (3.1) U_1 到U_p 是关于X 和 $\mathscr{R}$ 中变量的范围项；(3.2) F_1 到F_k 是关于 X 和 $\mathscr{R}$ 中对象的函数项；

表2：范畴演算的公式

带值域项的公式	安全变量
$x_1 \in O_1$	x_1
$x_1 \in O_1 \wedge \neg(x_1 \in O_2)$	x_1
含函数项和值域项的公式	安全变量
$((f_1 : x_1 \to x_2) = f_2 \circ g_1) \wedge (x_1 \in S_1) \wedge (x_2 \in S_2)$	x_1, x_2
$(\pi_1 : (x_1, x_2) \to x_1) \wedge (x_1 \in S_1) \wedge (x_2 \in S_2)$	x_1, x_2
含有谓词、范围和函数项的公式	数据模型
$(x_1 \in S_1) \wedge (x_2 \in S_2) \wedge (x_1 \rightsquigarrow^E x_2) \wedge (x_1 \cdot \text{名称} = \text{"John"})$	图
$(x_1 \in D_1) \wedge (x_2 \in D_2) \wedge (x_1 \text{ 是祖先 } x_2)$	Tree
含有不安全项的公式	不安全变量
$x_2 \in S_2, x_3 \in S_3, \exists x_1 (x_1 > x_3 \wedge x_2 = 6)$	x_1
$\forall x_1 \exists x_2 \in S_2 (x_1 > x_2)$	x_1
$(x_1 \in S_1) \vee f(x_1) = a_1$	x_1

(3.3) V 可以为空，也可以是满足以下性质的公式：(3.3.1) V 中的每个约束变量都有明确定义的范围，从而确保它们是安全表达式，后文将对此进行解释。(3.3.2) V 中的每个自由变量都属于X 和 $\mathscr{R}$ 中的变量。

每当我们在范畴演算表达式中使用全称量词、存在量词或谓词的否定时，都必须确保所得表达式是有意义的。安全表达式是指能够保证其结果在任何对象中都只包含有限个元素的表达式；否则，该表达式称为不安全表达式。也就是说，在安全表达式中，约束变量和自由变量的范围都必须明确定义。

例2。 考虑图2(a)所示的范畴模式，其中存储了有关学生、他们的地址、性别以及他们所修课程（位于SC对象中）的信息。来看一个查询场景：找出所有男学生及其地址，且这些学生修读了至少一名女学生所修读的每一门课程。图2(b)和(c)分别展示了该查询的结构和返回的范畴。范畴演算表述如下：

$$\{(x_1, x_2) \mid x_1 \in \text{Student}, x_2 \in \text{Address}, f(x_1) = x_2,\ x_1 \cdot \text{Gender} = \text{'Male'},$$
$$\forall y_1 \in \text{Student}, y_1 \cdot \text{Gender} = \text{'Female'} \to \exists y_2, y_3 \in \text{SC}, \exists y_4 \in \text{Course},$$
$$(y_2 \cdot \text{Student} = x_1) \wedge (y_3 \cdot \text{Student} = y_1) \wedge (y_2 \cdot \text{Course} = y_4) \wedge (y_3 \cdot \text{Course} = y_4)\}$$

例3. 设想一棵家谱树。图3(a)描绘了一个个体的范畴模式，其中使用杜威编码来表示他们在树中的位置。考虑一个特定的查询：找出*"John"*的所有祖先的名称。该查询的范畴演算表述如下：

{ $x | x \in$ 名称$\wedge \exists y_1 \in$人物, $\exists y_2 \in$人物, (y_1·杜威编码 是祖先 y_2·杜威编码) $\wedge$ (y_1·名称 =x) $\wedge$(y_2·名称="John")}

例4. 图3(b)展示了图结构数据的范畴模式。"边"对象表示一种关系对象，它投影到每条边的源和目标。此例用于

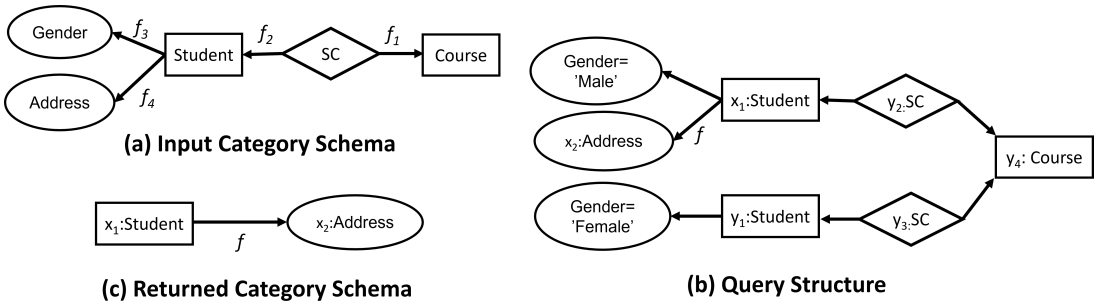

图 2：学生课程类别的示例

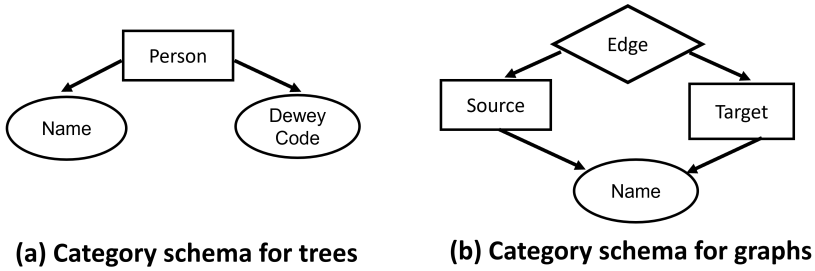

图 3：树和图的范畴模式

演示可达谓词。考虑以下查询：查找所有可由*"John"*到达的人物的名称。

$$\{x \mid x \in \text{名称} \wedge \exists y_1 \in \text{源}, \ \exists y_2 \in \text{目标}, \ (y_1 \rightsquigarrow^{Edge} y_2) \wedge (y_2 \cdot \text{名称} = x) \wedge (y_1 \cdot \text{名称} = \text{“John”})\}$$

4 范畴代数

范畴代数是一种形式系统，用于操作和查询范畴中的对象和态射。它提供一组运算，以一个或多个对象和态射作为输入，并生成一个新对象或范畴作为输出。这些运算分为两类：集合运算和范畴运算。集合运算对单个集合或多个集合（对象）执行操作，而范畴运算则涉及将对象和态射转换为范畴（范畴化），以及将范畴转换为对象（通过极限）。这些运算能够在范畴中精确且有条理地查询和操作数据。

4.1 集合算子

集合算子可以是一元、二元或三元算子。

4.1.1 一元集合算子

下面介绍三个算子：映射、投影和选择。1. 映射（f 或 $\cdot$）

映射算子使用一个函数将一个集合转换为另一个集合。如第 3 节所述，本文介绍函数（态射）的两种表示方式。给定对象 S_0, 和函数$f_i : S_{i-1} \to S_i, 1 \leq i \leq n$，映射算子可以写成以下任一形式:

$$f_n(...(f_1(S_0))) \stackrel{\text{def}}{=} \{y|x \in S_0 \wedge y = f_n(...(f_1(x)))\}, \ or$$

$$S_0 \cdot S_1 ... \cdot S_n \stackrel{\text{def}}{=} \{y|x \in S_0 \wedge y = x \cdot S_1 ... \cdot S_n\}$$

2. 投影（π）

范畴代数中的投影算子与关系代数中应用于关系对象的投影算子相似。设$A = (A_1, A2, ..., A_k)$为关系对象R中的投影分量列表。给定任意$r \in R$，对于$i = 1, 2, ..., n(n \geq k)$，记号$r[A_i]$表示$i$-$th$分量$A_i$在$r$中的值。

$$r[A] \stackrel{\text{def}}{=} (r[A_1], r[A_2], ..., r[A_k])$$

关系R 在A 上的投影定义为

$$\pi_A R \stackrel{\text{def}}{=} \{r|r[A] \in R\}$$

3. 选择（σ）

选择算子是一种一元算子，根据给定条件筛选元素。选择表达式的形式为$\sigma_{A\theta_M B}(S)$，其中 A 和B 表示两个函数表达式，而 θ_M 是一个数学谓词，包含 $=, \neq, <, >, \leq, \geq$。

$$\sigma_{A\theta_M B}(S) \stackrel{\text{def}}{=} \{x|x \in S \wedge (A(x)\theta_M B(x)),\}$$

其中，$A(x)$ 是函数表达式 A 作用于 $x \in S$ 的求值结果，使得

$$A(x) = \begin{cases} f_n(\cdots f_1(x)), & \text{if } A = f_n \circ \cdots \circ f_1 \\ x \cdot S_1 \cdot ... \cdot S_n. & \text{if } A = S_1 \cdot ... \cdot S_n \end{cases}$$

而 $B(x)$ 的定义方式类似。

示例 5。 回顾图2(a)所示的范畴模式。考虑以下查询：选择至少被一名女学生和一名男学生选修的课程。

$S_1 = \sigma_{student.gender='女学生'}(SC)$（选择算子）$S_2 = \sigma_{student.gender='男学生'}(SC)$（选择算子）$S_3 = S_1 \cap S_2$ （二元算子）$S_4 = S_3 \cdot$课程 （映射算子）将它们组合在一起：$(\sigma_{student.gender='女学生'}(SC) \cap \sigma_{student.gender='男学生'}(SC)) \cdot$课程

4.1.2 二元集合算子

集合之间的并（$\cup$）、交（$\cap$）、差（–）和笛卡尔积（$\times$）二元集合算子均按通常方式定义。下面，我们将介绍另外两个关于除法和树算子的二元集合算子。

1. 除法（$\div$）

范畴代数中的除法运算符与关系代数中的类似。具体而言，考虑将度为m 的关系对象R 除以度为n 的对象S ，并分别针对

A 在R 上的分量和B 在S 上的分量，记作$R[A] \div S[B]$。除法对象的度为$m - n$。

$$R[A] \div S[B] \stackrel{\text{def}}{=} \{(x_1,\ldots,x_n) | (x_1,\ldots,x_n) \in \pi_{\overline{A}} R \wedge \forall (y_1,\ldots,y_m) \in \pi_B S, (x_1,\ldots,x_n,y_1,\ldots,y_m) \in R\},$$

其中，$\overline{A}$ 和B 分别是R 和S 的分量列表，并且R 和S 分别与A 和B 互补。如果$n = 1$，则除法表达式中可以省略B 。

除法运算符可以用其他算子实现：

$R[A] \div S[B] = \pi_{\overline{A}} R - \pi_{\overline{A}}$<style 标识符='6'>$((\pi_{\overline{A}} R \times \pi_B$S<style 标识符='10'>$) - R$<style 标识符='13'>$)$

示例 6. 再次参考图2(a)。考虑以下查询： 查找参加所有女性学生所参加课程的男学生的地址。

$S_1 = \sigma_{student \cdot gender='Female'}(SC) S_2 = \sigma_{student \cdot gender='Male'}(SC) S_3 = S_2$[课程
] $\div (S_1 \cdot$ 课程$) S_4 = S_3 \cdot$地址将它们组合起来：$((\sigma_{student \cdot gender='Male'} SC)$[课程
] $\div (\sigma_{student \cdot gender='Female'} SC) \cdot$ 课程$) \cdot$地址

2. 树数据的二元运算符

考虑一个具有树结构的范畴 $\mathscr{T}$。给定$\mathscr{T}$中的两个杜威编码集合 D_1 和 D_2 ，函数 *getParent*(D_1, D_2) 可识别所有具有父子关系的元素对。

$$getParent(D_1,D_2) \stackrel{\text{def}}{=} \{(x,y) | x \in D_1 \wedge y \in D_2 \wedge (x \textit{ is a } \text{prefix } \textit{of } y) \wedge \text{level}(x) = \text{level}(y) - 1\}$$

类似地，我们可以定义算子 *getAncestor* 来查找具有祖先—后代关系的元素对。

$$getAncestor(D_1,D_2) \stackrel{\text{def}}{=} \{(x,y) | x \in D_1 \wedge y \in D_2 \wedge (x \text{ is a prefix of } y)\}$$

其他树数据算子，例如 *getSibling*、*getPreceding* 和 *getFollowing*等，可参见本文完整版 [16]。

4.1.3 图数据的三元运算符

考虑一个具有图结构的范畴 $\mathscr{G}$，其结构如图3(b)所示。给定两个节点集 源 和目标，以及 $\mathscr{G}$中的一个边集 边 。算子 *getReach*(*S*,*T*,*E*) 返回 *S*和 *T* 中的元素对，这些元素可通过*E*中的边相互到达。

$$getReach(S,T,E) \stackrel{\text{def}}{=} \{(x_1,x_2) | x_1 \in S \wedge x_2 \in T \wedge x_1 \rightsquigarrow^E x_2\}$$

同样，我们可以如下定义*n*-跳转算子：

$$getnHop(S,T,E) \stackrel{\text{def}}{=} \{(x_1,x_2) | x_1 \in S \wedge x_2 \in T \wedge x_1 \rightsquigarrow_n^E x_2\}$$

例 7. 考虑一个查询，用于查找图 3(b) 中“John”的所有递归好友。

$S_1 = \sigma_{Name=\text{“}John\text{”}}$源 $S_2 = getReach(S_1,$目标, 边$) S_3 = ($
$\pi_{Target} S_2) \cdot$名称组合在一起：$(\pi_{Target}(getReach($
$\sigma_{Name=\text{“}John\text{”}}$源, 目标, 边$)))\cdot$名称

getReach 的计算与**传递闭包**的计算类似，后者是一个经过充分研究的问题（例如 [23, 10, 1]）。关键区别在于，传递闭包会计算每个节点的所有可达节点对，而*getReach* 则专门返回集合S 与T之间的节点对。本文完整版 [16] 详细介绍了用于计算*getReach* 的算法

4.2 范畴算子

范畴算子在集合、函数和范畴之间执行转换 s.

4.2.1 集合与函数到范畴

给定一个集合族 $S1, \cdots, S_n$ 以及定义在这些集合上的函数，$f_1 : S_{i_1} \to S_{j_1}, \cdots, f_m : S_{i_m} \to S_{j_m}$, 我们有

$$Cat(S_1, ..., S_n, f_1 : S_{i_1} \to S_{j_1}, ..., f_m : S_{i_m} \to S_{j_m})$$

这个算子称为**范畴化**，它利用给定的对象和态射集合构造一个范畴。请注意，此构造仅适用于定义良好的函数$f : S_i \to S_j$，也就是说，对于每个元素$x \in S_i$，集合S_j中必须恰好存在一个对应的像$f(x)$。

4.2.2 范畴到集合

我引 入一个名为**极限** 的新算子，它可将范畴转换为关系对象（集合） .

$$Lim(Cat(S_1, ..., S_n, f_1 : S_{i_1} \to S_{j_1}, ..., f_m : S_{i_m} \to S_{j_m}))$$

$$\stackrel{\text{def}}{=} \{(x_1, \cdots, x_n) | x_i \in S_i, 1 \le i \le n \wedge f_k : x_{i_k} \to x_{j_k}, f(x_{i_k}) = x_{j_k}, 1 \le k \le m\}$$

极限算子定义在一组对象 $\mathscr{S} = \{x1, \cdots, x_n\}$ 及态射 $\mathscr{F} = \{f1, \cdots, f_m\}$上，会生成一个关系对象，其中包含来自 $\mathscr{S}$ 中各个对象的元素，且这些元素满足相应的函数映射 $\mathscr{F}$。在范畴论中，极限在分析和定义范畴中的结构方面发挥着关键作用。极限算子的用途类似于关系数据库中的连接算子：它将多个对象（关系）连接成一个对象（关系），以满足相应的函数映射（连接条件）。

We 如下所示，范畴代数与范畴演算等价：

定理 8. 范畴演算与范畴代数等价。

证明：该证明可通过以下两个步骤完成：
(i) 代数查询 ⊑演算查询；以及 (ii) 演算
查询 ⊑代数查询。

对于 (i)，只需证明每一种代数运算都可以用演算表达式来模拟。用演算表达式模拟代数运算的方式如下：

1. Map $f(S_1), f : S_1 \to S_2$:

$$\{y | y \in S_2 \wedge \exists x \in S_1, y = f(x)\};$$

2. 选择$\sigma_{A\theta_M B}(S)$:

$$\{x | x \in S \wedge A(x) \theta_M B(x)\}$$

3. 投影 $\pi_A R$，其中 $A = A1, ..., A_k$是关系对象 R 的分量子列表。

$$\pi_A R = \{(x_1, ..., x_k) | x_i \in A_i (1 \le i \le k) \wedge \exists (x_1, ..., x_k, x_{k+1}, ..., x_n) \in R\}$$

4. $S_1[A] \div S_2[B]$:

$$\{(x_1,\ldots,x_n)|(x_1,\ldots,x_n) \in \pi_{\overline{A}} S_1 \wedge (\forall (y_1,\ldots,y_m) \in \pi_B S_2, (x_1,\ldots,x_n,y_1,\ldots,y_m) \in S_1)\}$$

5. $getParent(D_1, D_2)$，其中D_1和D_2是两个杜威编码集。

$$\{(x_1,x_2)|x_1 \in D_1 \wedge x_2 \in D_2 \wedge x_1 \ isParent \ x_2\}$$

6. 获取 可达(S,T,E)，其中S和T分别表示源节点集和目标节点集，而E表示边集 t.

$$\{(x_1,x_2)|x_1 \in S \wedge x_2 \in T \wedge x_1 \rightsquigarrow^E x_2\};$$

7. 获取n跳(S, T, E):

$$\{(x_1,x_2)|x_1 \in S \wedge x_2 \in T \wedge x_1 \rightsquigarrow_n^E x_2\};$$

8. $Cat(S_1,\cdots, S_n, f_1: S_{i_1} \to S_{j_1},\cdots, f_m: S_{i_m} \to S_{j_m})$:

$$\{x_1,\ldots,x_n|x_i \in S_i(1 \le i \le n) \wedge f_k : S_{i_k} \to S_{j_k}(1 \le k \le m)\}$$

9. $Lim(Cat(S_1,\cdots, S_n, f_1: S_{i_1} \to S_{j_1},\cdots, f_m: S_{i_m} \to S_{j_m}))$:

$$\{(x_1,\cdots,x_n)|x_i \in S_i(1 \le i \le n) \wedge f_k(x_{i_k}) = x_{j_k}(1 \le k \le m)\}$$

然后我们着手证明 (ii) 演算查询 ⊑代数查询。

我们将给出一种算法，用于将演算表达式转换为语义等价的代数表达式。归约算法可概述如下：首先，将范畴演算表达式转换为前束范式，其中无量词部分（矩阵）为析取范式。其次，针对每个合取子句，构造一个范畴，以包含所有变量及其相关态射。第三，计算每个范畴的极限，并自然地移除不满足该极限交换图的元素。第四，通过选择和除法进一步精炼极限中剩余的元素。最后，投影目标对象和态射以构造所需结果。详细说明如下。

步骤 1:

(1.1）将V转换为前束范式，而不展开范围耦合限定符 s;

(1.2) 将V中的矩阵（公式中所有量词之后剩余的部分）转换为析取范式;

(1.3) 对 (1.2) 得到的变量进行变换，使变量按照其在限定条件中首次出现的顺序，变为x_1,...,x_n。

将上述变换得到的演算表达式记为

$$Z' = \{X', \mathscr{R}' | U' \wedge F' \wedge V'\}$$

步骤 2：生成以下三种类型的对象和态射;

(2.1) 为X'和 $\mathscr{R}'$ 中的每个变量生成实体对象或属性对象。对于每个变量x_i，为x_i的范围项构造一个定义方程。该方程右侧的代数表达式通过对x_i应用以下重写规则得到:

(i) $x_i \in O \Rightarrow S_i = O$ (ii) $\vee \Rightarrow \cup$（集合并）(iii) $\wedge \Rightarrow \cap$（集合交）(iv) $\wedge \neg \Rightarrow -$

例如，如果 $x_i \in O_1 \wedge \neg x_i \in O_2,$，那么 S_i 的定义方程为，

$$S_i = O_1 - O_2$$

在这种情况下，O_1 和 O_2 必须是并集兼容的。

(2.2) 为 $\mathscr{R}'$中每一组变量R_i 生成关系对象。具体而言，对于每个$R_i = (x_{i1}, \ldots\ldots, x_{im})$，创建对象$R_i$，并定义相关的投影态射 π_j：对于每个满足$x_{ij} \in S_{ij}$的j（$1 \leq j \leq m$），有$R_i \to S_{ij}$。

(2.3) 为V'中的所有树谓词和图谓词生成关系对象，包括可达（$\rightsquigarrow^E$）、n跳（$\rightsquigarrow_n^E$）、是父节点以及其他树算子。

(i) $x_1 \in S, x_2 \in T, x_1 \rightsquigarrow^E x_2 \Rightarrow$创建 $R_i = getReach(S, T, E)$，π_1：$R_i \to S$ 和 π_2：$R_i \to T$；

(ii) $x_1 \in S, x_2 \in T, x_1 \rightsquigarrow_n^E x_2 \Rightarrow$创建 $R_i = getnHop(S, T, E)$，π_1：$R_i \to S$ 和 π_2：$R_i \to T$；

(iii) $x_1 \in D_1, x_2 \in D_2,$是父节点$(D_1, D_2) \Rightarrow$创建 $R_i = getParent(D_1, D_2)$, $\pi_1 : R_i \to D_1$, $\pi_2 : R_i \to D_2$。步骤 3：

(3.1) 对于每个子句 C_k（位于V'的析取范式中），生成态射，使得 ($f(x_i) = x_j$或 $x_i \cdot S_j = x_j$) $\Rightarrow f$: $S_i \to S_j$，其中 x_i与 x_j出现在子句 C_k中。

(3.2) 为每个子句构造一个范畴 $\mathscr{C}_k$，其中涉及来自步骤 (2) 和 (3.1) 的相关对象与态射，$\mathscr{C}_k = Cat(S1, \cdots, S_n, f_1 : S_{i_1} \to S_{j_1}, \cdots, f_m : S_{i_m} \to S_{j_m})$;

(3.3) 计　算一个泛极限对象 $L_k = Lim(Cat(\mathscr{C}_k))$，其中 $\mathscr{C}_k$ 是步骤 (3.2) 中的范畴。

步骤4：对于V'的析取范式中的每个子句C_k，继续对步骤(3)生成的每个极限对象L_k 应用选择操作。

(i) $A(x_i)\theta_M B(x_j) \Rightarrow \sigma_{A(x_i)\theta_M B(x_j)}(L_k)$ (ii) $A(x_i)\theta_M \alpha \Rightarrow \sigma_{A(x_i)\theta_M \alpha}(L_k)$

其中a是一个个体常量，θ_M是 $=, \neq, >, <, \leq, \geq$之一。

步骤5：对上一步得到的极限对象计算除法，以纳入与V'的全称量词 $\forall$ 相对应的$S_j (1 \leq j \leq n)$。除法运算定义为$L[S1, \cdots, S_n] \div (S_1 \times \cdots \times S_n)[S1, \cdots, S_n]$，其中$L$表示步骤(4)中的极限对象，而$S_1 \times \cdots \times S_n$表示集合的笛卡尔积。对每个子句执行除法后，使用并运算合并各个子句的结果。

步骤 6：形成投影对象，以考虑 X' 和 $\mathscr{R}'$ 中的目标对象。最后，构造所需的目标范畴。

归约算法的有效性依赖于以下引理。由于篇幅所限，证明可见于本文的完整版本 [16]。

引理 9. 考虑一个范畴 $\mathscr{C}$，其对象集合为 $S_1, S_2, \cdots, S_n$，函数为 $f_1 : S_1 \to S_2, \cdots, f_n : S_n \to S_{n+1}$。设关系对象 $R = Lim(\mathscr{C})$。我们有 $\forall x_i \in \pi_{S_i}(R)$，$f(x_i) \in S_{i+1}$，其中 $1 \leq i \leq n$。

引理 10. 本引理论述多个 $\forall$量词的除法计算的有效性。考虑以下演算表达式：

$$\{x_1 | x_1 \in S_1 \wedge \forall y_1 \in S_2, \cdots, \forall y_n \in S_{n+1}, \exists z_1 \in R_1, \cdots, \exists z_n \in R_n, f_1(z_1) = x_1 \wedge g_1(z_1) = y_1, \cdots \wedge f_n(z_n) = x_n \wedge g_n(z_n) = y_n\} \quad (1)$$

该表达式等价于以下代数运算符：

$$\pi_{S_1}(L[S_2, S_3, \cdots, S_{n+1}] \div (S_2 \times S_3 \times \cdots S_{n+1})[S_2, S_3 \cdots, S_{n+1}])$$

其中 $L = Lim(Cat(S_1, \cdots, S_{n+1}, R_1, \cdots, R_n, f_1 : S_1 \to S_2, \cdots, f_n : S_1 \to S_{n+1}))$。

引理 *11*。此引理阐述了可达性算子对图数据的有效性。考虑以下演算表达式：$\{x_1, x_2 | x_1 \in S_1 \wedge x_2 \in S_2 \wedge x_1 \rightsquigarrow^E x_2 \wedge \exists x_3 \in S_3, f_1(x_3) = x_1 \wedge f_2(x_3) = x_2\}$。此演算可通过以下三个代数运算符实现：$S_4 = getReach(S_1, S_2, E)$; $S_5 = Lim(Cat(S_1, S_2, S_3, S_4, f_1 : S_3 \to S_1, f_2 : S_3 \to S2, \pi_1 : S_4 \to S1, \pi_2 : S_4 \to S_2))$ and $S_6 = \pi_{S_1, S_2 S_5}$。

例12。 回想例2中的演算表达式。我们用此示例来说明上述转换算法。

步骤 1：将其转换为前束范式，并将矩阵转换为析取范式。

$\{x_1, x_2 \mid x_1 \in$ 学生 $\wedge x_2 \in$ 地址$\wedge (f: x_1 \to x_2 = f_2) \wedge \forall y_1 \in$ 学生 $\exists y_2 \in SC\ \exists y_3 \in SC\ \exists y_4 \in$课程, $((x_1 \cdot$性别 $=$"男性" $\wedge y_1 \cdot$性别 $\neq$"女性")$\vee((x_1 \cdot$性别 $=$"男性") $\wedge (y_2 \cdot$学生 $= x_1) \wedge (y_3 \cdot$学生 $= y_1) \wedge (y_2 \cdot$课程 $= y_4) \wedge (y_3 \cdot$课程 $= y_4)))\}$

步骤 2：为每个变量生成对象x_1, x_2, y_1 到 y_4：

$S_1 =$ 学生 对于 x_1；$S_2 =$地址 对于 x_2；$S_3 =$ 学生 对于 y_1；$S_4 = SC$ 对于 y_2；$S_5 = SC$ 对于 y_3；以及$S_6 =$课程 对于 y_4。

步骤 3：为析取范式中的每个子句构造一个范畴，其中包含相关对象和态射，然后计算相应的极限：

第一个极限：$S_7 = \mathrm{Lim}(S_1, S_3)$，它表示$S_1$ 与S的笛卡尔积 3.

第二个极限：$S_8 = \mathrm{Lim}(S_1, S_3, S_4, S_5, S_6, f_1: S_4 \to S_1, f_2: S_5 \to S_3, f_3: S_4 \to S_6, f_4: S_5 \to S_6)$。

步骤 4：对极限对象应用选择条件：

$S_9 = \sigma_{S_1.\text{性别}='\text{男性}'}(S_7) \cap \sigma_{S_3.\text{性别}\neq'\text{女性}'}$

(S_7) $S_{10} = \sigma_{S_1.\text{性别}='\text{男性}'}(S_8)$

步骤 5：针对变量 $\forall y_1$ （即 S_3），计算每个子句的除法，然后将结果取并。

$S_{11} = S_9[\text{学生}] \div S_3$,，结果为空；$S_{12} = S_{10}[\text{学生}] \div S_3$；$S_{13} = S_{11} \cup S_{12} = S_{12}$。步骤 6：返回最终结果 $S_{14} = \pi_{S_1} S_{13}$；$S_{15} = \pi_{S_2} S_{13}$；$S_{16} = \mathrm{Cat}(S_{14}, S_{15}, f_3: S_{14} \to S_{15})$。返回 S_{16}。

5 代数变换规则

关系数据库中的查询优化会用到各种代数变换规则。这些规则用于将原始查询转换为等价的查询，以便更高效地执行。同样，在多模型数据库中，范畴代数变换规则也可用于多模型查询优化。我们提出了一系列重写规则，多模型查询优化器可通过代数变换使用这些规则。

(1) **级联**f：一系列函数可以串联为各个函数操作的复合。

$$f_n(\ldots(f_1(S)) \equiv (f_n \circ \ldots \circ f_1)(S)$$

(2) **Lim 和** π：投影算子（π）的作用是关系对象的极限对象算子的逆。它从关系对象中提取特定分量，这与极限算子相反，后者综合来自多个对象的元素。

$$\pi_{S_1}(Lim(Cat(S_1, S_2, f: S_1 \to S_2))) \equiv S_1$$

$$\pi_{S_2}(Lim(Cat(S_1, S_2, f: S_1 \to S_2))) \subseteq S_2$$

特别是，如果S_1 与S_2,之间不存在态射，则$Lim(S_1, S_2)$=$S_1 \times S_2$。在这种情况下，两个等式都成立。

$$\pi_{S_1}(Lim(S_1,S_2)) \equiv S_1 \text{ and } \pi_{S_2}(Lim(S_1,S_2)) \equiv S_2$$

（3）**将 σ 推送到*lim*中的一个或多个对象**:

$$\sigma_C(Lim(S_1,S_2)) \equiv Lim(\sigma_C(S_1),S_2)$$

或者，如果选择条件C可以写成$C_1 \wedge C_2$,，其中条件C_1仅涉及S_1的值，而条件C_2仅涉及S_2,的值，则该运算可交换如下:

$$\sigma_C(Lim(S_1,S_2)) \equiv Lim(\sigma_{C_1}(S_1),\sigma_{C_2}(S_2)$$

(4) **将 σ 下推至*getReach*中的一个或多个对象**:
假设C_1是集合S上的过滤选择，且C_2作用于T，则以下下推算子有效:

$$getReach(\sigma_{C_1}(S),\sigma_{C_2}(T),E) \equiv \sigma_{C_1 \wedge C_2}(getReach(S,T,E))$$

(5) **将 σ 下推至*getParent* 中的一个或两个对象以及其他树结构算子**。

$$\sigma_C(getParent(D_1,D_2)) \equiv getParent(\sigma_{C_1}(D_1),\sigma_{C_2}(D_2))$$

(6)**将函数映射与积算子交换**。
考虑一个范畴 $\mathscr{C}$。若$f: A \to B$ 和 $g: C \to D$ 是 $\mathscr{C}$ 中的两个态射，则它们的 Product$f \otimes g: A \times C \to B \times D$ 也是 $\mathscr{C}$ 中的一个态射。给定两个集合 S_1 和 S_2,，我们有

$$(f \otimes g)(S_1 \times S_2) \equiv f(S_1) \times g(S_2)$$

(7) 与 Lim 操作 π 交换。给定两个关系对象 R_1 和 R_2、一个函数 $f: R_1 \to R_2$,，以及一个投影列表 L={A1,..., A$_n$, B1,..., B$_m$}，其中 L_1 = {A1,..., A$_n$} 是 R_1 的分量，L_2 = {B1, ..., B$_m$} 是 R_2 的分量。若存在另一个函数 $f_2: R'_1 \to R'_2$，其中 $R'_1 = \pi_{A_1,...,A_n}(R_1)$ 且 $R'_2 = \pi_{B_1,...,B_m}(R_2)$，使得下图可交换:

$$\begin{array}{ccc} R_1 & \xrightarrow{f_1} & R_2 \\ {\scriptstyle \pi_{L_1}}\downarrow & & \downarrow{\scriptstyle \pi_{L_2}} \\ R'_1 & \xrightarrow[f_2]{} & R'_2 \end{array}$$

那么这两个算子 π 和 *Lim* 可以按如下方式交换:

$$\pi_L(Lim(Cat(R_1,R_2,f_1: R_1 \to R_2))) \equiv Lim(Cat(\pi_{L_1}(R_1),\pi_{L_2}(R_2),f_2: \pi_{L_1}(R_1) \to \pi_{L_2}(R_2)))$$

（8）使 g 与 Lim 运算可交换。考虑一个范畴 $\mathscr{C}$，其中有两个对象 S_1 和 S_2,，以及一个态射 $f_1: S_1 \to S_2$。设 g 是一个定义域为 Lim($\mathscr{C}$) 的函数，它可以分解为两个函数 g $= g_1 \otimes g_2$,，其中 g_1 和 g_2 的定义域分别为 S_1 和 S_2,。若存在另一个态射 $f_2: g_1(S_1) \to g_2(S_2)$，使得下图可交换:

$$\begin{array}{ccc} S_1 & \xrightarrow{f_1} & S_2 \\ {\scriptstyle g_1}\downarrow & & \downarrow{\scriptstyle g_2} \\ S'_1 & \xrightarrow[f_2]{} & S'_2 \end{array}$$

那么这两个算子 g 和 极限 可以按如下方式交换:

$$g(Lim(Cat(S_1,S_2,f_1:S_1\to S_2)))\equiv Lim(Cat(g_1(S_1),g_2(S_2),f_2:g_1(S_1)\to g_2(S_2)))$$

(9) 可交换的*Lim* 与 *getReach* 操作。考虑这样一种情形：使用 极限 和 *getReach* 算子对单个对象的内容进行筛选。以下规则表明，可以交换应用这两个筛选条件的顺序。

$$\pi_{S_1}(getReach(\pi_{S_1}\sigma_C(lim(S_1,S_2))\equiv\pi_{S_1}\sigma_C(Lim(\pi_{S_1}getReach(S_1,T,E),S_2))$$

定理 13. 范畴演算和范畴代数可以表达以下所有内容:

- 关系演算和代数查询;
- 图模式匹配和图可达性查询;
- *XML*小枝模式查询。

定理14。考虑一个具有 p 个对象和 q 个态射的范畴 $\mathscr{C}$ 。设 $\mathscr{C}$ 中任一对象所含元素数量的最大值为 n。

1. 在 $\mathscr{C}$中，范畴演算和范畴代数计算的数据时间复杂度上界为 $O(q\cdot n^p)$。*2.* 这些计算的空间复杂度上界为 $NSPACE[\log n]$。

6 结论与未来工作

在这项工作中，我们为范畴数据库提出了两种查询机制：范畴演算和范畴代数。这两种机制旨在提取对象中满足指定查询条件的元素子集。范畴论传统上强调对象之间的抽象关系和结构，往往忽略对象内部的元素，而我们采取了不同的方向。我们着重研究如何提取对象中的元素子集，以应对多模型数据查询的挑战。这种调整为范畴论在数据库领域的应用提供了新颖视角，并带来了现有文献尚未充分探讨的新研究挑战。未来，鉴于范畴代数中算子的统一性和简洁性，我们计划为多模型数据开发整体性的查询优化算法。

参考文献

[1] Rakesh Agrawal、Shaul Dar 和 HV Jagadish（1990）：直接传递闭包算法：设计与性能评估。《ACM 数据库系统汇刊》15(3)，第 427–458 页。可访问：https://doi.org/10.1145/88636.88888。

[2] Kristopher Brown、David I. Spivak 和 Ryan Wisnesky（2019）：面向计算科学的范畴数据集成。《ArXiv》，doi:10.48550/arXiv.1903.10579。[3] E. F. Codd（1970）：大型共享数据库的数据关系模型。《ACM 通讯》13(6)，第 377–387 页，doi:10.1145/362384.362685。[4] Diego Figueira（2013）：关于带传递轴和数据测试的 *XPath*。载于：第*32*届 *ACM SIGMOD-SIGACT-SIGAI* 数据库系统原理研讨会论文集，PODS '13，计算机协会，纽约，美国纽约州，第 249–260 页。可访问：https://doi.org/10.1145/2463664. 2463675。

[5] Nadime Francis、Amélie Gheerbrant、Paolo Guagliardo、Leonid Libkin、Victor Marsault、Wim Martens、Filip Murlak、Liat Peterfreund、Alexandra Rogova 和 Domagoj Vrgoc（2023）：GPC：属性图的模式演算。载于 Floris Geerts、Hung Q. Ngo 和 Stavros Sintos 编：《第42届 ACM SIGMOD-SIGACT-SIGAI 数据库系统原理研讨会论文集》，PODS 2023，美国华盛顿州西雅图，2023年6月18—23日，ACM，第 241–250 页。可访问：https://doi.org/10.1145/3584372.3588662。[6] Jeremy Gibbons、Fritz Henglein、Ralf Hinze 和 Nicolas Wu（2018）：通过伴随实现关系代数。《Proc. ACM Program. Lang.》2(ICFP)，第 86:1–86:28 页。可访问：https://doi.org/10.1145/3236781。[7] Todd J. Green、Grigoris Karvounarakis 和 Val Tannen（2007）：来源半环。载于：《第二十六届 ACM SIGMOD-SIGACT-SIGART 数据库系统原理研讨会论文集》，PODS '07，计算机协会，纽约，美国纽约州，第 31–40 页。可访问：https://doi. org/10.1145/1265530.1265535。[8] Dirk Van Gucht（1987）：非规范化关系模型中扩展关系代数的表达能力。载于 Moshe Y. Vardi 编：《第六届 ACM SIGACT-SIGMOD-SIGART 数据库系统原理研讨会论文集》，1987年3月23—25日，美国加利福尼亚州圣迭戈，ACM，第 302–312 页，doi:10.1145/28659.28692。[9] Jelle Hellings、Catherine L Pilachowski、Dirk Van Gucht、Marc Gyssens 和 Yuqing Wu（2020）：从关系代数到半连接代数：一种图查询优化方法。《计算机杂志》64(5)，第 789–811 页，doi:10.1093/comjnl/bxaa031。arXiv:https://academic.oup.com/comjnl/article- pdf/64/5/789/38335010/bxaa031.pdf。[10] Yannis E Ioannidis、Raghu Ramakrishnan 等（1988）：高效传递闭包算法。载于：VLDB，88，第 382–394 页。可访问：https://doi.org/10.14778/3554821.3554891。[11] Felix Kiehn 、 Mareike Schmidt、Daniel Glake、Fabian Panse、Wolfram Wingerath、Benjamin Wollmer、Martin Poppinga 和 Norbert Ritter（2022）：多语言数据管理：现状与开放挑战。《VLDB基金会会刊》15(12)，第 3750–3753 页，doi:10.14778/3554821.3554891。[12] Gabriel M. Kuper 和 Moshe Y. Vardi（1984）：一种新的数据库逻辑方法。载于：《第3届 ACM SIGACT-SIGMOD 数据库系统原理研讨会论文集》，PODS '84，计算机协会，纽约，美国纽约州，第 86–96 页。可访问：https://doi.org/10.1145/588011. 588026。[13] Leonid Libkin（1995）：数据库中的近似。载于：《数据库理论——ICDT'95，第五届国际会议，捷克共和国布拉格，1995年1月11—13日，论文集》，《计算机科学讲义》893，施普林格，第 411–424 页，doi:10.1007/3-540-58907-4_31。[14] Leonid Libkin 和 Limsoon Wong（1997）：多重集与聚合函数的查询语言。《计算机与系统科学杂志》55(2)，第 241–272 页。可访问：https://doi.org/10.1006/jcss.1997.1523。[15] Jiaheng Lu（2025）：面向多模型数据的范畴统一：第一部分 范畴模型与范式。arXiv:2502.19131。[16] Jiaheng Lu（2025）：面向多模型数据的范畴统一：第二部分 范畴代数与演算。arXiv:2504.09515。[17] Jiaheng Lu 和 Irena Holubová（2019）：多模型数据库：处理数据多样性的新旅程。《ACM计算调查》52(3)，第 55:1–55:38 页，doi:10.1145/3323214。[18] Jiaheng Lu、Tok Wang Ling、Chee-Yong Chan 和 Ting Chen（2005）：从区域编码到扩展 Dewey 编码：高效处理 XML 枝模式匹配。《第31届超大型数据库国际会议论文集》，VLDB '05，VLDB基金会，第 193–204 页，doi:10.5555/1083592.1083618。[19] Ryohei Nakano（1990）：从关系演算到含聚合函数的关系代数的优化转换。《ACM数据库系统汇刊》15(4)，第 518–557 页。可访问：https://doi.org/10.1145/99935.99943。[20] G. Özsoyoˇglu、Z. M. Özsoyoˇglu 和 V. Matos（1987）：扩展关系代数和关系演算，使其支持集合值属性和聚合函数。《ACM数据库系统汇刊》12(4)，第 566–592 页。可访问：https://doi.org/10.1145/32204.32219。

[21] Jan Paredaens & Dirk Van Gucht (1988)：在嵌套代数表达式中使用平坦算子的可能性与局限。载于 Chris Edmondson-Yurkanan & Mihalis Yannakakis（编）：第七届 *ACM SIGACT-SIGMOD-SIGART* 数据库系统原理研讨会论文集，*1988*年*3*月*21*日至*23*日，美国德克萨斯州奥斯汀，ACM，第29–38页，doi:10.1145/308386.308402。[22] Steven Roman 等（2017）：范畴论语言导论。第6版，施普林格。[23] Claus-Peter Schnorr（1978）：一种具有线性期望时间的传递闭包算法。《*SIAM*计算杂志》7(2)，第127–133页，doi:10.1137/0207011。[24] Patrick Schultz、David I. Spivak、Christina Vasilakopoulou & Ryan Wisnesky（2016）：代数数据库。arXiv:1602.03501。[25] Dan Suciu（2024）：半环中的不同差异。载于 Antoine Amarilli & Alin Deutsch（编）：计算优雅性的源流——献给 *Val Tannen* 的论文，开放获取信息学丛书（*OASIcs*）119，Dagstuhl城堡—莱布尼茨信息学中心，德国达格施图尔，第10:1–10:20页，doi:10.4230/OASIcs.Tannen.10。参见 https://drops.dagstuhl.de/entities/document/10. 4230/OASIcs.Tannen.10。[26] Val Tannen（1994）：教程：集合类型语言。载于 Victor Vianu（编）：第十三届 *ACM SIGACT-SIGMOD-SIGART* 数据库系统原理研讨会论文集，*1994*年*5*月*24*日至*26*日，美国明尼苏达州明尼阿波利斯，ACM出版社，第150–154页。参见 https://doi.org/10.1145/ 182591.182608。[27] Igor Tatarinov、Stratis Viglas、Kevin S. Beyer、Jayavel Shanmugasundaram、Eugene J. Shekita & Chun Zhang（2002）：使用关系数据库系统存储和查询有序 *XML*。载于 Michael J. Franklin、Bongki Moon & Anastassia Ailamaki（编）：*2002*年 *ACM SIGMOD* 数据管理国际会议论文集，美国威斯康星州麦迪逊，*2002*年*6*月*3*日至*6*日，ACM，第204–215页。参见 https://doi.org/10.1145/564691.564715。[28] Valter Uotila & Jiaheng Lu（2022）：多模型数据转换的形式化范畴论框架。*CoRR*abs/2201.04905。arXiv:2201.04905。[29] Allen Van Gelder & Rodney W. Topor（1991）：关系演算的安全性与转换。《*ACM*数据库系统汇刊》16(2)，第235–278页。参见 https://doi.org/10.1145/114325.103712。