迈向维护合取查询的参数化困难性

QICHEN WANG, 新加坡南洋理工大学

我们研究在单元组更新下，动态维护固定的无自连接合取查询结果的细粒度复杂性。先前研究表明，自由连通查询可以在更新时间为 $O(|D|^{\delta})$ for some $\delta \in [0.5, 1]$的情况下得到维护，其中 $|D|$ 是当前数据库的大小。然而，最佳已知上界 $O(|D|)$ 与针对任意 $\epsilon \geqslant 0$的下界 $\Omega(|D|^{0.5-\epsilon})$ 之间仍存在差距。

我们通过引入两个结构参数——高度 k和维度 d——来量化合取查询的动态复杂性，从而缩小这一差距。我们建立了新的细粒度下界，表明除非广受认可的猜想不成立，否则任何维护具有这些参数的查询的算法都必须承担 $\Omega(|D|^{1-1/\max(k,d)-\epsilon})$, 的更新时间。这些结果给出了维护自由连通查询的首个 super-$\sqrt{|D|}$ 下界，并表明在考虑任意大的 k和 d时，现有算法可能已达到紧确界。

作为对我们下界结果的补充，我们确定了一个依赖于数据的参数，即广义 H索引 $h(D)$，其上界为 $|D|^{1/d}$,，并设计了一种用于维护星形查询的高效算法。星形查询是高度为 2 的自由连通查询中的常见类别。该算法在线性空间 $O(|D|)$下实现了实例特定的更新时间 $O(h(D)^{d-1})$。这与我们的参数化下界相匹配，并在有利情况下提供了实例特定的性能。

CCS 概念 pts: • **计算理论 → 数据库查询处理与优化（理论）** .

附加关键词和短语：合取查询、细粒度复杂性、查询维护、动态算法

arXiv:2603.14754v1 [cs.DB] 16 Mar 2026

1 引言

更新下的合取查询应答是数据库理论与系统中的一项基础任务，应用于增量视图维护 [13, 24, 27, 33, 35, 36], 、复杂事件处理[34], 以及流处理 [10]。在完全动态场景中，给定一个固定的无自连接合取查询[1] Q 以及一个初始为空的数据库 D。随后，数据库经历一系列元组插入与删除操作。目标是维护一个数据结构，使得当前查询结果 $Q(D)$ 能够随时被检索。具体而言，令 $\omega(Q)$ 表示 最优静态指数，即对于每个数据库实例 D，所有能够在 D 上评估查询Q 的算法都需要运行时间 $\Omega(|D|^{\omega(Q)-\epsilon} + |Q(D)|)$，其中某个常数为 $\epsilon \geqslant 0$。我们关注这样的动态算法：在每次更新后，所维护的数据结构都支持检索 $Q(D)$，且其开销严格小于从头重新评估 Q，即在时间 $O(|D|^{\omega(Q)-\epsilon} + |Q(D)|)$ 内完成，其中某个常数为 $\epsilon > 0$。这里，$|D|$ 表示当前数据库的大小，$|Q(D)|$表示当前查询结果的大小。我们通过处理一次插入或删除操作的摊销更新时间 t_u 以及枚举延迟来衡量效率。理想情况下，枚举延迟应为常数，即与 $|D|$无关。

[1]在本文余下部分，我们只考虑“无自连接”的合取查询，并在文中省略“无自连接”这一限定。含自连接的合取查询可能比与之对应的无自连接合取查询更简单；对应的无自连接查询是通过为所有关系 [12]指定不同的关系符号而得到的。虽然针对含自连接的查询已有一些下界结果（例如，[9] 中关于维护含自连接的布尔查询的定理 1.2），但对于静态和动态求值，含自连接查询的难度结果目前尚未在一般情况下得到确定。

作者的贡　联系信息：Qichen Wang，南洋理工大学，新加坡，qichen.wang@ntu.edu.sg .

本作品依据知识共享署名-非商业性使用-禁止演绎 4.0 国际许可协议授权。

尽管存在这一限制，动态问题仍然具有挑战性。近期研究得出了非平凡的上界和下界。最优静态指数 $\omega(Q)$ 通过仅插入归约，为摊销更新时间 [5, 23] 提供了一个直接的下界 $\Omega(|D|^{\omega(Q)-1-\epsilon})$。具体来说，如果我们通过逐个插入元组来构造数据库实例 D，那么一个摊销更新时间为 t_u、枚举时间为 $O(|D|^{\omega(Q)-\epsilon}+|Q(D)|)$ 的动态算法，就可以在 $O(|D|\cdot t_u+|D|^{\omega(Q)-\epsilon}+|Q(D)|)$内计算 Q。因此，对于任意 $\gamma>0$，即使仅进行插入更新，我们也必须有$t_u=\Omega(|D|^{\omega(Q)-1-\gamma})$；否则，$\omega(Q)$ 的最优性就会被推翻。对于自由连通查询（即满足 $\omega(Q)=1$的无环查询），这一紧确下界 [5, 23]适用于仅插入更新序列；但在考虑删除操作时，它并非最优下界。Berkholz 等人 [9] 提出了 q-层次查询的概念，这类查询是自由连通查询的一个子集，并证明了一个紧确的二分分类：假设在线矩阵向量乘法（OMv）猜想成立，那么任何非 q-层次化的合取查询都无法在支持常数延迟枚举的同时，实现常数更新时间。特别是，如果一个合取查询不是 q-层次化的，那么任何动态算法若要实现常数延迟枚举，就必须承担摊销 $\Omega(|D|^{1/2-\epsilon})$，无论 $\epsilon>0$ 更新时间如何。在上界方面，人们已开展多项研究，以高效维护自由连通合取查询 [24, 33, 35]。这些研究表明，每次更新可在线性时间 $O(|D|)$ 内维护自由连通查询，同时仍支持常数延迟枚举。Kara 等人 [28] 研究了层次查询（另一类与自由连通查询相交的无环查询子集）的摊销更新时间 t_u 与枚举延迟 t_d之间的权衡。对于自由连通层次查询，他们证明了一个乘积权衡 $t_d\times t_u=O(|D|)$。另一方面，由于 $\omega(Q)=1$ 适用于所有自由连通查询，任何非常数枚举延迟都会导致比在 $|Q(|D|)|=\Omega(|D|)$时重新计算查询更大的开销。综上，这些结果表明，目前自由连通查询可达到的最佳指数 δ 介于 0.5 和 1 之间。能否缩小这一差距，仍是该领域的一个开放问题。

1.1 我们的贡献

在这项工作中，我们通过引入参数化下界来细化已知的二分分类，从而推进了对这一问题的研究。我们为合取查询 Q定义了两个新的结构参数：高度（记为 k）和维度（记为 d）。直观而言，Q 的高度衡量其关系超图中最长无弦路径的长度；随着我们通过增加关系来“扩展”查询，高度也会增加。对于自由连通查询 Q，Q 的高度等于所有自由连通连接树中的最小高度。另一方面，Q 的维度衡量查询如何通过增加每个关系中的属性来“扩展规模”。[2] 这些参数在合取查询中形成了一个自然的层级。图 1 展示了本文提出的、针对带有 $\omega(Q)=1$的自由连通查询的层级。

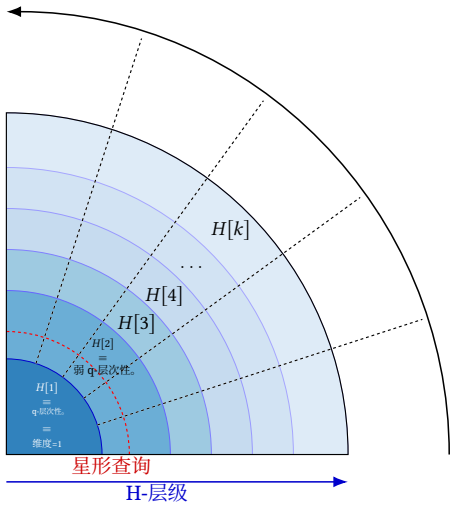


图 1. 带有 $\omega(Q)=1$的查询层级。

接着，我们证明依赖于 (k,d) 的细粒度下界。具体而言，我们证明：

定理 1.1（**基于高度的下界**）。设 Q 是一个高度为 k 的查询，且满足 $k\geqslant 2$。假设组合 k-团猜想成立，则不存在动态组合算法能够以 $O(|D|^{(k-1)/k-\epsilon})$ 的摊销更新时间维护 Q，并支持任意常数 $\epsilon>0$ 下的常数延迟枚举。

[2]高度参数和维数参数的形式化定义分别见第3节和第4节。

定理 1.2（**基于维度的下界**）。设 Q 是一个维度为$d \geqslant 2$的合取查询。假设 $OuMv_k$ 猜想成立，则不存在动态算法能够以 $O(|D|^{(d-1)/d-\epsilon})$ 的摊销更新时间维护 Q，并支持任意常数 $\epsilon > 0$ 下的常数延迟枚举。

定理 1.1 表明，更高高度的查询本质上需要更大的更新成本：对于高度为 k的查询，无法将指数降至低于 $(k-1)/k$。定理 1.2 表明，维度为 d的查询同样无法以低于 $(d-1)/d$ 的指数进行维护。此外，他们还重新证明了 Berkholz 等人提出的先前二分分类 [9]:

命题 1.3。 对于任意合取查询 Q，以下陈述等价：*(1)* Q 是 q-层次化的；*(2)* Q 的高度为 $k = 1$；*(3)* Q 的维度为 $d = 1$。

这些结果也补充了在 $\omega(Q) \leqslant 2$时的 $\omega(Q) - 1$ 下界。当 $\omega(Q) \leqslant 1.5$时，由高度和维度导出的下界占主导地位，任何维护算法都必须承担摊销更新时间 $\Omega\left(|D|^{1-1/\max(k,\ d)-\epsilon}\right)$；而当 $1.5 < \omega(Q) \leqslant 2$时，任何维护算法都必须承担摊销更新时间 $\Omega\left(|D|^{\delta-\epsilon}\right)$，其中 $\delta = 1 - 1/\max\left(k,\ d,\ \frac{1}{2-\omega(Q)}\right)$，且为任意常数 $\epsilon > 0$。尤其是，这些结果是**首次为维护自由连通查询提供超平方根下界**，这意味着，对所有自由连通查询采用通用的 $O(|D|^{1-\epsilon})$-时间维护算法将违反这些猜想；除非在所依赖的、被猜想为困难的问题上找到突破性算法，否则这一结论将从否定角度解决该开放问题，并由此**证明当前上界算法的最优性**。

在算法方面，我们通过针对星形查询提出一个新的参数化上界来补充我们的下界；星形查询是高度为 2 的查询子集，详细信息见定义 4.5。具体而言：

定理 1.4（**星形查询算法**）。设 Q 是一个维度为 $d \geqslant 1$ 的星形查询，$j \in [0、\ d]$ 是 $\bar{x}_1$ 中的输出属性，D 是当前数据库实例，$h(D)$ 是其广义 H- 索引（一种数据库密度度量，见第 *5* 节）。存在一种使用线性空间 $O(|D|)$ 的动态算法，可维护 Q，其摊销更新时间为 $O(h(D)^{d-1})$，并支持 $O(h(D)^{d-j})$ 延迟枚举。

定理 1.4 表明，对于分析型工作负载中广泛使用的星形查询，通过利用依赖于实例的 H-索引参数，有时可以优于一般的 $(d-1)/d$ 上界。与此同时，$h(D) \leqslant |D|^{1/d}$ 始终成立；任何能够在时间 $O(h(D)^{d-1-\epsilon})$ 内维护此类查询、且对任意 $\epsilon > 0$ 都成立的算法，都会与定理 1.2 和 OuMv $_k$ 猜想相矛盾。

1.2 文章结构

本文其余部分安排如下。第 2 节介绍符号约定，并回顾本文所需的背景知识。我们还将正式定义查询的高度参数和维数参数。第 3 节研究高度参数，形式化定义高度 k 查询，并通过从奇偶环检测问题归约，证明定理 1.1。第 4 节研究维数参数，并通过从 OuMv$_k$ 问题归约，证明定理 1.2。第 5 节介绍适用于星形查询的动态算法，并证明定理 1.4。最后，第 6 节讨论相关工作和未来方向，证明及其他示例则见附录。

2 预备知识

2.1 合取查询及其结构性质

2.1.1 无环合取查询。

合取查询（*CQ*） *[4]*。设数据库模式由属性集合 $\mathbf{V}$ ={x_1、……、 x_m} 以及关系集合 $\mathbf{E}$ = {R_1、……、 R_n}定义。每个关系 $R_i \in \mathbf{E}$ 都有一个模式 $\bar{x}_i \subseteq \mathbf{V}$。合取查询（CQ）表示为:

$$Q(\bar{y}) \leftarrow R_1(\bar{x}_1), \cdots, R_n(\bar{x}_n),$$

其中， $\bar{y} \subseteq \mathbf{V}$ 是输出属性的集合。如果 $\bar{y} = \mathbf{V}$，则该查询是完整合取查询，我们可以省略头部中的 $\bar{y}$，并将其记作 $Q(*)$。如果 $\bar{y} = \emptyset$，则该查询是布尔合取查询，记作 $Q()$。我们也用 $(\mathbf{V}, \mathbf{E}\ \bar{y})$ 表示一个合取查询。

对于任意属性 $x \in \mathbf{V}$，我们令 $\mathbf{E}[x] = \{R(\bar{x}) \in \mathbf{E} : x \in \bar{x}\}$ 为包含 x 的关系集合。我们还令 $\mathbf{V}_\bullet = \{x \in \mathbf{V} : |\mathbf{E}[x]| = 1\}$ 为唯一属性的集合，即不是连接属性的属性。

超图与无环性。 关联关系超图的无环性定义了合取查询的无环性 Q。

定义 *2.1*（关系超图）。与合取查询 $Q = (\mathbf{V}, \mathbf{E}\ \bar{y})$ 相关联的关系超图为 $\mathbf{H} = (\mathbf{V}, \mathbf{E})$，其中 $\mathbf{E}$ 是超边（关系）的集合，$\mathbf{V}$ 是顶点（属性）的集合。每条超边 $R_i \in \mathbf{E}$ 都包含顶点集合 $\bar{x}_i$。

$\mathbf{H}$ 的原图 $G = (V, E)$ 是一个顶点集为 $V = \mathbf{V}$ 的图；当且仅当存在 $R(\bar{x}) \in \mathbf{E}$，使得 $v, u \in \bar{x}$ 时，其中存在一条边 $(v, u) \in E$。定义在 $V \subseteq \mathbf{V}$ 上的原图是一个图 $G(V) = (V, E)$；当且仅当存在 $R(\bar{x}) \in \mathbf{E}$，使得 $v, u \in V \cap \bar{x}$ 时，其中存在一条边 $(v, u) \in E$。

无环性有多种定义；在本文中，我们考虑标准的α-无环性。CQ Q 是 α-无环的，当且仅当它存在一个广义连接树（GJT）$\mathbf{T}$ [35]。

广义连接树（GJT）是一棵树 $\mathbf{T} = (\mathbf{V}(\mathbf{T}), \mathbf{E}(\mathbf{T}))$，其中每个节点 $v \in \mathbf{V}(\mathbf{T})$ 对应一个输入关系 $R_i(\bar{x}_i) \in \mathbf{E}$，或一个广义关系 $[\bar{x}]$，其中 $[\bar{x}] \subseteq \bar{x}_i$适用于某个 $R_i(\bar{x}_i)$。GJT $\mathbf{T}$ 满足:

- **覆盖：** 对于每个关系 $R_i(\bar{x}_i) \in \mathbf{E}$，都存在一个节点 $v \in \mathbf{V}(\mathbf{T})$ 使得 $v = R_i$。此外，$\mathbf{T}$ 的所有叶节点都必须是 $\mathbf{E}$ 中的输入关系。
- **连接：** 对于任意属性 $x \in \mathbf{V}$，节点集合 $\{v \in \mathbf{V}(\mathbf{T}) : x \in v\}$ 在 $\mathbf{T}$ 中诱导出一棵连通子树。

这个定义推广了传统的连接树；传统连接树是其中所有节点 $\mathbf{V}(\mathbf{T})$都对应于 $\mathbf{E}$ 中输入关系的特殊情况。

2.1.2 更新、枚举与计算模型。我们假设数据库初始为空。我们考虑一种动态场景，其中数据库经历一系列单元组更新。每次更新包括向关系 $R(\bar{x})$中插入或删除一个元组 $t \in \mathrm{dom}(\bar{x})$。我们遵循集合语义：插入一个已存在于 R中的元组 t，或删除一个不存在于 R中的元组 t，都不会改变数据库实例。尽管某个特定元组值可能在整个序列中被多次插入或删除，但我们将每次操作视为一个独立事件。

更新按顺序处理；针对单次更新的维护过程必须在接收下一次更新之前结束。每次更新后，系统都可以进行查询结果的枚举。我们将枚举延迟 定义为以下各阶段所经过的最长时间：枚举开始到输出第一个元组、输出任意两个连续元组之间，以及输出最后一个元组到过程终止。我们关注数据复杂度，并将查询大小视为常数。

我们的复杂度分析基于统一成本度量下标准的字 RAM 计算模型。对于大小为 N 的输入，我们假设机器字长为 $w = \Omega(\log N)$ 位。在该模型中，基本操作（包括算术运算、位运算、拼接和直接内存访问）均可在 $O(1)$ 时间内执行。

2.1.3 自由连通合取查询。合取查询的无环性定义并未考虑输出属性的影响。不过，如果我们也将输出属性 $\bar{y}$ 纳入考量，就可以细化无环性的概念，定义出自由连通查询这一类。具体而言，若将输出属性集合 $\bar{y}$ 视为一个额外关系后得到的派生查询 $Q' = (\mathrm{V}, \mathrm{E} \cup \{\bar{y}\}, \bar{y})$ 也是无环的，则无环合取查询 $Q = (\mathrm{V}, \mathrm{E}\ \bar{y})$ 就是自由连通的。等价地，无环合取查询 Q 若存在自由连通连接树，则它是自由连通的。

自由连通连接树。广义连接树 T 若满足标准的**覆盖** 和**连接** 性质，以及以下三个附加性质，则称为自由连通连接树：
• **上方：** 没有输入关系节点 $v \in \mathrm{V}(\mathrm{T})$ 是任何广义关系节点的祖先。• **守卫：** 对于任意广义关系节点 $v \in \mathrm{V}(\mathrm{T})$，其所有子节点 $v_c \in \mathrm{V}(\mathrm{T})$ 均满足 $v \subseteq v_c$。• **连通：** 存在一个连通节点集 $\mathrm{V}_{\mathrm{con}}(\mathrm{T}) \subseteq \mathrm{V}(\mathrm{T})$，使得：**(i)** $\mathrm{V}_{\mathrm{con}}(\mathrm{T})$ 在 T 中诱导出一棵包含 T 的根节点的连通子树；**(ii)** 对于每个父节点为 v_p 的节点 $v \in \mathrm{V}_{\mathrm{con}}(\mathrm{T})$，它们共享的属性均为输出属性，即 $(v \cap v_p) \subseteq \bar{y}$；**(iii)** 连通集中的属性并集覆盖所有输出属性，即 $\bar{y} \subseteq \bigcup_{v \in \mathcal{V}_{\mathrm{con}}(\mathcal{T})} v$。

自由连通查询类受到广泛关注，因为在布尔矩阵乘法猜想和超团猜想成立的假设下，只有自由连通查询才能在线性时间内求值，或在经过线性预处理后以常数延迟枚举。

猜想 2.2（布尔矩阵乘法猜想（BMM））。两个 $n \times n$ 布尔矩阵的乘法无法在 $O(n^2)$时间内完成。

猜想 2.3（超团猜想（HyperCliqe））。对于任意整数 $k \geqslant 3$，无法在时间 $O(m)$内，确定具有 m条超边的 $(k-1)$-均匀超图中是否存在 k 超团。

定理 2.4（[8]）。假设 *BMM* 和 *HyperClique* 成立，当且仅当 Q 是自由连通的，合取查询 Q 才能在线性时间 $O(|D| + |Q(D)|)$ 内求值。

定理 2.5（[8]）。假设 *BMM* 和 *HyperClique* 成立，当且仅当 Q 是自由连通的，合取查询 Q 才能在经过线性时间 $O(|D|)$ 预处理后，以 $O(1)$ 延迟枚举。

这些定理还表明，假设 BMM 成立，当且仅当 Q 是自由连通的 CQ Q 才 $\omega(Q) = 1$。这种二分定理也适用于仅插入更新序列的动态场景。

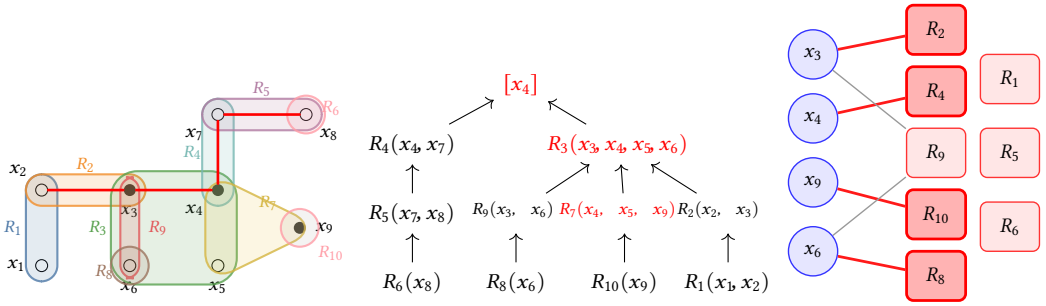
定理 2.6（[23, 24, 35]）。对于任意仅插入更新序列，若且唯若 Q 是自由连通的，在假设 *BMM* 和 *HyperClique* 成立的条件下，*CQ* Q 可在均摊 $O(1)$ 时间内维护，同时支持 $O(1)$延迟枚举。

例*2.7*。考虑自由连通查询

$$Q_1(x_3, x_4, x_9) \leftarrow R_1(x_1, x_2), R_2(x_2, x_3), R_3(x_3, x_4, x_5, x_6), R_4(x_4, x_7), \\ R_5(x_7, x_8), R_6(x_8), R_7(x_4, x_5, x_9), R_8(x_6), R_9(x_3, x_6), R_{10}(x_9).$$

图2a展示了其关系超图，其中输出属性以实心点标出；图2b展示了一棵可能的广义连接树，其中所有红色节点构成连通子树。

2.1.4 自由连通查询的细粒度分类。在静态场景中，每个自由连通查询都可以高效地求值和枚举。然而，Berkholz 等人 [9] 指出，在动态场景中，要理解维护查询的难度，需要进行更细粒度的分类。具体而言，他们识别出*q-层次*查询这一类查询，它是自由连通查询的子集，可以高效地维护。



(a) 关系超图 (b) 广义连接树 (c) N 和 $V_k(N)$ 对于 $\{x_5\}$

图2. 例2.7示意图

算法1： 归约$(Q = (\mathcal{V}, \mathcal{E}\ \bar{y}))$

$_1$ $E_e \leftarrow$ {E}中的所有耳；$_2$ **当** 存在 $R(\bar{x})$，$R'(\bar{x}') \in E$ 使得 $R \in E_e$ 且 R' 是其锚点 **执行** $_3$ $E \leftarrow E - \{R\}$

$_4$ **返回**$\mathcal{E}$;

q-层次化合取查询。 若合取查询 $Q = (V, E\ \bar{y})$ 满足以下两个条件，则称其为 q-层次化：（1）对于任意一对属性 x_1，$x_2 \in V$，$E[x_1] \subseteq E[x_2]$ 或 $E[x_2] \subseteq E[x_1]$ 或 $E[x_1] \cap E[x_2] = \varnothing$；（2）如果 $x_1 \in \bar{y}$ 且 $E[x_1] \subsetneq E[x_2]$，则 $x_2 \in \bar{y}$。

定理 2.8（[9],定理 1.1）。给定任意更新序列，若合取查询 Q 是 q-层次化的，则在 *OMv* 猜想成立的前提下，可以在$O(1)$ 时间内维护它，同时支持 $O(1)$延迟枚举；反之亦然。

弱 q-层次化合取查询。 q-层次化类的限制相当严格；例如，即使是简单的查询 $Q(x_1) \leftarrow R_1(x_1, x_2), R_2(x_2)$ 也不是 q-层次化的。Wang 等人 [23, 35] 通过定义弱 q-层次化查询，对该类进行了小幅扩展；在先进先出（FIFO）更新序列等某些实际条件下，这类查询可以得到高效维护。为了定义弱 q-层次化查询，我们需要引入耳和骨架的概念。

耳这一概念最早由 GYO 算法 [20, 38], 引入：如果存在另一个关系 R' 包含关系 R 的所有非唯一属性 R，则该关系是一个耳。该算法通过反复移除所有唯一属性，以及属性包含于另一个关系中的关系，来检验 CQ 的α-无环性；等价地说，就是反复从查询中移除耳，然后检查查询是否变为空。如果查询归约为空集，则可以将每个被移除的耳 R 连接到其锚点 R'，从而为该查询构造一个有效的连接树。

Hu 和 Wang [23] 扩展了这一概念，以涵盖输出属性，并将此类关系称为 可约关系。在这一扩展下，广义 GYO 归约过程如下：反复移除唯一属性，前提是该属性不是输出属性，或其所属关系 R 的所有非唯一属性都是输出属性。然后移除属性包含于另一个关系中的每个关系。基于这一广义归约，我们现在给出自由连通查询中耳的形式化定义。

定义 *2.9*（耳）。在自由连通查询 $Q = (V, E\ \bar{y})$ 中，如果存在另一个关系 $R'(\bar{x}') \in E$，使得满足以下任一条件，则称关系 $R(\bar{x}) \in E$ 为该查询的一只耳：（i）$\bar{x} \backslash V_\bullet \subseteq \bar{x}'$ 和$\bar{x} \cap V_\bullet \subseteq V \backslash \bar{y}$；或（ii）$\bar{x} \backslash V_\bullet \subseteq \bar{x}' \cap \bar{y}$。我们将 R' 称为 R 的一个锚点。

查询 Q的骨架记为 E_*，是指在从 E 中递归移除一个耳、直至无法继续移除后所剩关系的子集。这一归约总结如下：

算法1。该算法与 GYO 算法类似，不同之处在于：耳的集合在开始时计算一次，并在从 E 中移除耳的过程中保持不变。因此，骨架保证非空，因为移除一个耳需要剩余查询中存在一个锚点。对于给定的自由连通查询，我们进一步定义残余查询的概念：

定义 *2.10*（残余查询）。给定一个自由连通查询 Q 及其骨架 $\mathrm{E}*$，其中 $\mathrm{V}_* = \{x \mid \exists R(\bar{x}) \in \mathrm{E}*, x \in \bar{x}\}$ 是 $\mathrm{E}*$中的属性集合，而 $\bar{y}_* = \{x \mid x \in \bar{y} \wedge \exists R(\bar{x}) \in \mathrm{E}*, \ x \in \bar{x}\}$ 是 $\mathrm{E}*$中的输出属性集合，则对于 Q，残余查询 Q_* 是由 $\mathrm{E}*$诱导的查询 $Q_* \leftarrow (\mathrm{V}*, \ \mathrm{E}*, \ \bar{y}_*)$。

一对关系 R_i， R_j 可能既是耳关系，又是另一个关系的锚点，从而使骨架以及残余查询不具有唯一性。然而，不同残余查询所对应的底层超图应当是同构的，并且这些残余查询的难度保持不变：

引理2.11。 设 $Q_1 = (\mathcal{V}_1, \mathcal{E}_1$ 和 $Q_2 = (\mathcal{V}_2, \mathcal{E}_2\, \bar{y}_2)$ 是给定自由连通查询 Q 的两个残余查询，那么对于每个 $R(\bar{x}) \in \mathcal{E}_1,$，要么 *(1)* $R(\bar{x}) \in \mathcal{E}_2,$，要么 *(2)* 存在 $R'(\bar{x}') \in \mathcal{E}_2,$，使得 $R'(\bar{x}') \notin \mathcal{E}_1$，且 $\bar{x} \backslash \mathcal{V}_\bullet = \bar{x}' \backslash \mathcal{V}_\bullet$。

证明。根据定义，任何不是耳的关系都必须保留在骨架中。因此，任何具有非耳锚点的耳都必须被移除。现在考虑两个残余查询 Q_1 和 Q_2。如果$R(\bar{x}) \in \mathrm{E}_1$，但 $R(\bar{x}) \notin \mathrm{E}_2,$，那么 R 必定是原查询 Q 的一个耳，其锚点也都是 Q 的耳。由于 R 在 Q_2 中被移除，却仍保留在 Q_1中，因此必定存在 R 的一个锚点 $R' \in \mathrm{E}_2$，使得 $R' \notin \mathrm{E}_1$。同理，由于 $R' \notin \mathrm{E}_1,$，它必定有一个锚点 $R'' \in \mathrm{E}_1$。锚点关系具有传递性：若 R_i 是 R_j 的锚点，且 R_k 是 R_i的锚点，那么 R_k 也是R_j的锚点。因此，我们必须有 $R'' = R$；否则，根据传递性， R 也会在 E_1 中有一个不同于其自身的锚点，因此本应从 E_1，中移除，这就产生了矛盾。由此可知， R和 R' 互为锚点，因此 $\bar{x} \backslash \mathrm{V}_\bullet^1 = \bar{x}' \backslash \mathrm{V}_\bullet^2$。□

例*2.12*。 考虑例2.7中的 Q_1。关系 R_1、 R_6、 R_8、 R_9， R_{10} 是耳，其中 R_2 是 R_1的锚点， R_5 是 R_6的锚点， R_4 是 R_8 和 R_9的锚点， R_7 是 R_{10}的锚点。其残余查询为

$$Q_1'(x_3, x_4, x_9) \leftarrow R_2(x_2, x_3), R_3(x_3, x_4, x_5, x_6), R_4(x_4, x_7), R_5(x_7, x_8), R_7(x_4, x_5, x_9).$$

定义*2.13*。 如果自由连通查询 Q 不是 q-层次化的，且其残余查询 Q_* 是 q-层次化的，则称其为弱 q-层次化。

自由连通连接树高度。 自由连通查询的分类与其自由连通连接树 [23, 35]的高度密切相关。

定义*2.14*。 广义连接树 T 的高度 是从任意叶节点到根节点的路径上输入关系的最大数量，不计广义关系。

引理2.15（[23], 引理2.2和引理 3.7）。 设 Q 是一个自由连通合取查询。 Q 是 q-层次化的，当且仅当它存在一棵高度为1 的自由连通连接树； Q 是弱 q-层次化的，当且仅当它存在一棵高度为2的自由连通连接树。

在这项工作中，我们通过扩展高度的概念，细化了合取查询的分类。对于自由连通查询，我们证明，当且仅当查询存在一棵高度为 k 的自由连通连接树，但不存在高度为 $k-1$ 的自由连通连接树时，该查询的高度为 k。从这一视角来看，q-层次化类和弱 q-层次化类分别恰好对应高度为 1 和高度为 2 的查询。

2.2 k-环检测问题

在图中寻找环是算法图论中的一个基本问题。给定一个有向图 $G = (V, E)$ 其中有 n 个顶点和 m 条边， k-环是由 k 个互不相同的顶点组成的序列$(v_1, v_2, \ldots, v_k)$ 满足：对于每个 $1 \leqslant i < k$, 都有 $(v_i, v_{i+1}) \in E$；并且 $(v_k, v_1) \in E$。 k-环检测问题要判断给定图中是否包含 k-环。一般而言，这个问题是 NP 完全的。例如，当 $k = n$时，它可归约为哈密顿环问题。另一方面，以 k为参数时，该问题是固定参数可解（FPT）的，这意味着对每个固定的 $k = O(1)$，都能在多项式时间内求解。

The k-环检测问题也是细粒度复杂性中的核心问题。特别是，特殊情形 $k = 3$ （三角形检测）已被广泛用于建立数据库查询处理的条件下界 [22, 31, 37]。然而，除三角形以外，一般的 k-环问题在数据库查询的语境中鲜少被讨论。Lincoln 和 Vyas[29]基于组合 k-团猜想，证明了在稀疏图上检测奇环和偶环的下界。

猜想 2.16（组合 k-团猜想 [1]）。不存在一种组合算法，能够在 $O(n^{k-\epsilon})$ 时间内检测具有 n 个顶点和 $O(n^2)$ 条边的图中是否存在 k-团，其中 $\epsilon > 0$ 为任意常数， $k \geqslant 3$为任意整数

定理 2.17（奇环 [29], 定理 15）。给定一个有向图 $G = (V, E)$ 具有 $|V| = n$和 $|E| = m$, 并且 $m = n^{1+2/(k-1)}$, 假设组合 k-团猜想成立，则不存在组合算法能够在 $O(m^{2k/(k+1)-\epsilon})$ 时间内检测是否存在奇数k-环，其中 $k \geqslant 3$ 为任意奇整数， G，且 $\epsilon > 0$为任意常数。

定理 2.18（偶环 [29], 推论 16）。给定一个有向图 $G = (V, E)$ 满足 $|V| = n$和 $|E| = m$，且 $m = n^{k/(k-2)}$, 不存在任何组合算法，能够在时间 $O(m^{(2k-2)/k-\epsilon})$ 内检测是否存在偶数长度为k的环，其中任意偶整数 $k \geqslant 4$ 满足 G，对于任意常数 $\epsilon > 0$，假设组合 k-团猜想成立。

请注意，奇环和偶环的下界都假设图是稀疏图。要证明数据库查询的下界，这一限制是必要的，因为我们按输入规模 $O(|D|)$ 而非域大小来衡量复杂度。在上界方面，Alon 等人 [7] 提出了一种算法，其检测 k-环的运行时间为 $O(m^{2-(1+\lceil k/2 \rceil^{-1})/(k+1)})$。随后，Dahlgaard 等人 [15] 将无向图中寻找偶环的运行时间改进至$O(m^{2k/(k+2)})$。此外，Lincoln 等人 [30] 给出了一种归约，将检测无向偶环的复杂度与检测有向奇环的复杂度联系起来。

我们将不 简要回顾这些工作中的相关技术，它们对我们的证明很有用 .

彩色编码。 检测 k-环的主要挑战在于确保环中的所有 k 顶点互不相同。彩色编码是 Alon 等人提出的一种随机化技术， [6] 用于应对这一挑战。其核心思想是从一个包含k 种颜色的集合中，为每个顶点 $v \in V$ 随机分配一种颜色。根据定义，一个彩色 k-环（其中所有 k 顶点的颜色都互不相同）是一个简单的k-环。如果图中包含一个 k-环，那么在随机颜色分配下，该环的顶点有一定概率会呈现彩色。通过将这一随机过程重复足够多次，就能以较高概率找到任何现存的 k-环。

这种随机化过程可以使用一个 (n, k)-完美哈希族 [32] 进行去随机化。

定义 *2.19*（(n, k)-完美哈希族）。A (n, k)-完美哈希族是一个哈希函数集合H，它将包含 n 个元素的全集映射到包含 k 个值的范围，使得对于任意包含 S 个元素且满足$|S| = k$的子集，至少存在一个哈希函数 $f \in H$ 在 S 上是单射（即， f将 S 中的 k 个互不相同的元素映射到 k 个互不相同的值）。

给定这样的哈希函数族 H，可以遍历其中的每个 $f \in H$，为 G的顶点着色，然后在着色图上运行环检测算法。该族 H保证，对于 G中的每个 k-环，至少有一个哈希函数 $f \in H$使该环具有彩色性。存在一种针对 H的构造，使得 $|H| = k^{O(1)} \log(n)$ [32]，从而确保去随机化算法仍然是固定参数可解的。在本文中，我们将彩色编码和完美哈希视为可在需要时使用的黑盒。

2.3 （广义）在线矩阵向量乘法

在线矩阵向量乘法（OMv） [3] 是理解在线问题难度的一项基础问题。形式化地，给定一个 $n \times n$ 布尔矩阵 M，以及按顺序给出的一个由n维向量组成的序列 $\{v_1, \cdots, v_n\}$。任务是在收到下一个向量 v_{i+1}之前，针对每个 v_i回答矩阵向量乘法 Mv_i。这便引出了著名的在线矩阵向量猜想。

猜想 2.20（OMv 猜想）。对于任意常数 $\epsilon > 0$，不存在一种算法能够在时间 $O(n^{3-\epsilon})$ 内解决 OMv 问题，其中输入包含一个由 n 个向量组成的序列。

猜想中的难度结果为各种动态问题给出了下界，这也为回答合取查询提供了细粒度复杂性结果 [9, 23, 27]。

Jin 和 Xu [26] 将该问题推广到了张量版本，并提出了参数化的 OuMv_k。给定一个张量 $M \in [n]^k$，对于每一轮 i，都会给出一串 kn 维向量$\{u^{(1)}, u^{(2)}, \cdots, u^{(k)}\}$，任务是在收到下一轮向量之前，回答 $u^{(1)} \times u^{(2)} \times \cdots \times u^{(k)}$ 与 M是否有非空交集。

猜想 2.21（OuMv_k 猜想）。对任意 $\gamma, \epsilon > 0$，不存在一种算法能够解决 $OuMv_k$ 问题，同时满足查询数为 n^γ、预处理时间为线性且总查询时间为 $O(n^{\gamma+k-\epsilon})$。

本文将广义 OuMv_k 猜想应用于合取查询在更新下的维护，从而给出更紧的下界。

3 扩展时的维护困难性

从 q-层次化查询和弱 q-层次化查询的例子中，我们观察到一种规律：通过添加额外关系来扩展查询时，维护该查询的难度会增加。本节将探讨随着查询扩展，维护困难性如何变化。

3.1 环检测与合取查询

我们首先说明结合 k 环检测回答合取查询之间的联系；环检测可识别出三类困难性查询：Q_{2k-1}、Q_{2k}和Q'_k。

引理 3.1。考虑具有任意输出属性 $\bar{y}$的查询：

$$Q_{2k-1}(\bar{y}) \leftarrow R_1(x_1), R_2(x_1, x_2), \cdots, R_{2k-2}(x_{2k-3}, x_{2k-2}), R_{2k-1}(x_{2k-2}).$$

如果对于任意常数 $\epsilon > 0$，Q_{2k-1}都能以均摊 $O(|D|^{(k-1)/k-\epsilon})$ 时间进行维护，同时支持 $O(|D|^{1-\epsilon})$延迟枚举，那么组合 k-团猜想将不成立。

证明。给定一个具有 m条边和 n个顶点的图 $G = (V, E)$，其中 $n = m^{(k-1)/k}$，我们首先找到一个 $(n, 2k-1)$-完美哈希族 H；对于每个 $f \in H$，我们使用哈希函数 f为每个顶点着色，从而将图划分为 $2k-1$个互不相交的子图 $V = V^1 \cup V^2 \cup \cdots \cup V^{2k-1}$，其中 $V^x \leftarrow \{v \in V | f(v) = x\}$。令 P为所有 $(2k-1, 2k-1)$-排列的集合，并令 p为其中任意一个排列。对于任意 f，p，我们按如下方式创建更新序列：

(1) 我们为 $R_i = \{(u, v) \in E | f(u) = p(i-1), f(v) = p(i)\}$ 赋值，并将所有此类边插入 R_i，适用于所有 $i \in [2, 2k-2]$

(2) 对于每个满足 $f(v) = p(2k-1)$ 的 $v \in V$，我们将所有 $\{(u)|(v,u) \in E, f(u) = p(1)\}$ 插入 R_1，并将所有 $\{(u)|(u,v) \in E, f(u) = p(2k-2)\}$ 插入 R_{2k-1}。(3) 我们尝试在当前数据库实例上枚举 Q_{2k-1}。如果存在任何结果，那么就存在一条 $2k-2$ 路径 $x_1, x_2, \cdots, x_{2k-2}$，因为颜色编码保证，对于任意一对 $x_i, x_j, x_i \neq x_j$。此外，我们知道存在两条边 (v, x_1) 和 (x_{2k-2}, v)，其中 v 的颜色与任何 x_i 都不同。因此，$(v, x_1, \cdots, x_{2k-2})$ 构成一个长度为 $2k-1$ 的环，我们可以通过报告检测到 $2k-1$ 环来终止。(4) 否则，我们从 R_1 和 R_{2k-1} 中删除所有元组，并重复 (2) - (4)，直到不存在更多 v。若情况如此，我们可以断定给定的 f, p 不存在 $2k-1$ 环。

上述更新序列将每条边插入数据库一次，并从数据库中删除每条边至多一次，总计进行 $O(m)$ 次更新；因此，如果此类查询可以在 $O(|D|^{(k-1)/k-\epsilon})$ 时间内维护，那么维护整个更新序列的总运行时间为 $O(m^{(k-1)/k-\epsilon} \times m) = O(m^{(2k-1)/k-\epsilon})$ 时间。令 $k' = 2k-1$、$O(m^{(2k-1)/k-\epsilon}) = O(m^{2k'/(k'+1)-\epsilon})$。同时，在上述过程中总共会执行 $O(n)$ 次枚举操作。如果枚举延迟为 $O(m^{1-\epsilon})$，那么枚举过程需要花费 $O(m^{1-\epsilon} \times m^{(k-1)/k}) = O(m^{2k'/(k'+1)-\epsilon})$ 时间；将这两部分相加，上述过程的总运行时间为 $O(m^{2k'/(k'+1)-\epsilon})$。

请注意，我们还需要验证所有可能的颜色编码和排列，以确保不会遗漏图中的任何环。维护所有 $f \in H$ 和 $p \in P$ 的总运行时间为

$$O(\underbrace{\log m}_{\text{size of } H} \times \underbrace{(k')!}_{\text{Number of permutations}} \times \underbrace{m^{2k'/(k'+1)-\epsilon}}_{\text{Maintenance cost for each } p,f}) = \underbrace{O(m^{2k'/(k'+1)-\epsilon'})}_{k'=O(1), \log m \text{ suppressed in } \epsilon'}.$$

完成后，我们可以确定给定的图 G 是否包含 $k' = 2k-1$ 环。由于上述过程是组合性的，因此它提供了一种用于解决奇环检测问题的组合算法。然而，定理 2.17 表明，除非组合 k-团猜想不成立，否则不存在能在 $O(m^{2k'/(k'+1)-\epsilon})$ 时间内检测此类环的组合算法。因此，这种维护方法的存在也意味着该猜想不成立。□

引理 3.2。考虑具有任意输出属性 $\bar{y}$ 的查询：

$$Q_{2k}(\bar{y}) \leftarrow R_1(x_1), R_2(x_1, x_2), \cdots, R_{2k-1}(x_{2k-2}, x_{2k-1}), R_{2k}(x_{2k-1}).$$

如果 Q_{2k} 可以在均摊 $O(|D|^{(k-1)/k-\epsilon})$ 时间内维护，同时支持 $O(|D|^{1-\epsilon})$ 延迟枚举，且适用于任意常数 $\epsilon > 0$，那么组合 k-团猜想不成立。

证明。给定一个图 $G = (V, E)$，它有 m 条边，$n = m^{(k-1)/k}$ 且 $k' = 2k$，我们构造与奇环情形相同的更新序列。此时，总更新时间为 $O(m^{(2k-1)/k-\epsilon}) = O(m^{(2k'-2)/k'-\epsilon})$，对于任意颜色编码 $f \in H$ 和排列 p，枚举时间也为 $O(m^{(2k-1)/k-\epsilon}) = O(m^{(2k'-2)/k'-\epsilon})$，这将得到一个能在 $O(m^{(2k'-2)/k'-\epsilon})$ 时间内检测偶环的组合算法。结合定理 2.18，维护算法的存在意味着组合 k-团猜想不成立。□

引理 3.3. 考虑查询

$$Q_k(x_k) \leftarrow R_1(x_1), R_2(x_1, x_2), \cdots, R_k(x_{k-1}, x_k).$$

如果对于任意常数 $\epsilon > 0$，Q_k 可以在摊销 $O(|D|^{(k-1)/k-\epsilon})$ 时间内维护，同时支持 $O(|D|^{1/k-\epsilon})$ 延迟枚举，那么组合 k-团猜想就不成立。

证明。给定一个含有 m 条边的图 $G = (V, E)$，我们维护两个查询 $Q_k(x_k)$，如下所示

$$Q_k(x_k) \leftarrow R_1(x_1), R_2(x_1, x_2), \cdots, R_k(x_{k-1}, x_k)$$
$$Q'_k(x_k) \leftarrow R_{2k-1}(x_{2k-2}), R_{2k-2}(x_{2k-3}, x_{2k-2}), \cdots, R_k(x_{k-1}, x_k).$$

更新序列与引理 3.1 的证明相同；唯一的区别在于枚举过程。完成每一轮更新后，我们枚举 Q_k 和 Q'_k，并检查 $Q_k \cap Q'_k$ 是否为空。如果不为空，则检测到一个 $k' = 2k - 1$ 环。输出规模至多为 $n = m^{(k-1)/k}$。因此，为了确保开销严格小于重新求值，延迟应为 $O(|D|^{1/k-\epsilon})$，其中 $\epsilon > 0$。当摊销更新时间为 $O(|D|^{(k-1)/k-\epsilon})$, 时，总运行时间变为 $O(m^{(2k-1)/k-\epsilon}) = O(m^{2k'/(k'+1)-\epsilon})$,，上述过程为奇数 k' 环检测问题提供了一种组合算法。结合定理 2.17，这一维护算法的存在表明组合 k-团猜想不成立。□

3.2 无弦路径与高度

第 3.1 节中给出的查询表明，涉及的关系越多，难度就越高。为了量化这种影响，我们基于关系超图上的无弦路径，为任意合取查询 Q引入一个关系超图上的结构参数：高度。

给定一个无向图 $G = (V, E)$, 图 G上的无弦路径 [21] 是由互不相同的顶点$P = \{v_1, \cdots, v_{n+1}\}$和边序列 $P_e = \{e_1, \cdots, e_n\}$组成的序列，使得对于每个$i \in [1, n]$，都有 $e_i = (v_i, v_{i+1})$，并且由 P诱导的子图 $G_P \leftarrow (V_p = P, E_p = \{(v_i, v_j) \in E, v_i, v_j \in P\})$ 本身就是这条路径。换句话说，对于任意满足 $|i - j| > 1, (v_i, v_j) \notin E$的 $i, j \in [1, n + 1]$，都有.....

图中路径长度自然由边的数量定义（n）；但将其推广到超图时需要谨慎，因为单条超边可能包含任意数量的顶点，而且给定序列 P，边序列 P_e可能并不唯一。我们提出如下推广定义：

定义 *3.4*（超图上的无弦路径）。给定超图 $\mathrm{H} = (\mathrm{V}, \mathrm{E})$, 我们将超图 H 上的无弦路径定义为由互不相同的顶点构成的序列 P，而无弦路径 P的边序列P_e则定义为由超边构成的序列。无弦路径 P是

• 长度为 1 的无弦路径，当且仅当 $P \leftarrow \{\varnothing\}$；而 $P_e \leftarrow \{e_1\}$ 作为 P的边序列，若包含任意一条超边 e_1•，则构成长度为 2 的无弦路径，当且仅当 $P \leftarrow \{v_1\}$ 且存在两条超边 e_1, e_2 满足 $v_1 \in e_1 \cap e_2$。所有此类超边对构成 P• 长度为 k的无弦路径 $P_e \leftarrow \{e_1, e_2\}$ 的边序列，当且仅当 $P \leftarrow \{v_1, \cdots, v_{k-1}\}$ 满足以下条件：（1）（邻接）对于每个 $i \in [1, k-2]$, 存在一条超边 $e_{i+1} \in E$同时包含 $v_i, v_{i+1} \in e$；（2）（无捷径）对于任意 $i, j \in [1, k-1]$ 满足 $|i - j| > 1$, 不存在一条超边 $e \in E$同时包含 $v_i, v_j \in e$；（3）（端点）存在 $e_1, e_k \in E$，使得 $v_1 \in e_1$且 $v_2 \notin e_1$，并且 $v_{k-1} \in e_k$且 $v_{k-2} \notin e_k$。任何满足上述条件的 $P_e \leftarrow \{e_1, \cdots, e_k\}$都是 P的边序列。注意，由于“无捷径”要求，边序列中的所有超边也互不相同；对于给定的超图，这样的边序列可能并不唯一。

如果不存在严格更长的无弦路径 P' 且满足 $P \subsetneq P'$，则无弦路径 P是极大的。当所有超边都是标准边时，该定义与标准图中的定义一致，并且序列 P被定义为该路径的内部顶点 v_2、.....、 v_n。“邻接”和“无捷径”要求都可以从标准图自然地推广而来。然而，由于可能存在一元超边，两个“端点”超边中只保留了一个顶点。

此外，对于非全且非布尔查询，我们进一步定义 q-无弦路径的概念：它是仅包含 P 中非输出属性的无弦路径，其中一个“端点”超边包含输出属性。这推广了 [9, 23] 中 q-核和钩核的概念。

长度为 1 的无弦路径也是 q-无弦路径，其边序列 $\{e_1\}$ 满足 $e_1 \cap \bar{y} \neq \emptyset$ 和 $e_1 \backslash \bar{y} \neq \emptyset$，即包含输出属性和非输出属性的 e_1。长度为 k（$k \geqslant 2$）的 q-无弦路径是 H 上的一条无弦路径 $P \leftarrow \{v_1, \cdots, v_{k-1}\}$，满足

(5)（**非输出顶点**）对于所有 $x_i \in P$、$x_i \notin \bar{y}$，也就是说，P 中的所有属性都不是输出属性；
(6)（**输出端点**）存在一个超边 $R(\bar{x})$，满足 $x_1 = \bar{x} \cap P$，且 $\bar{x} \cap \bar{y} \neq \emptyset$，也就是说，$\bar{x}$ 同时包含输出属性和 P 的一个端点 x_1。q-无弦路径的边序列必须包含这样一个超边 R。

例*3.5*。考虑例2.7中的 Q_1。我们可以在其关系超图上找到一条长度为6的无弦路径 $P \leftarrow \{x_2, x_3, x_4, x_7, x_8\}$（在图2a中以红色标出），其中 $P_e \leftarrow\{R_1, R_2, R_3, R_4, R_5, R_6\}$。注意，$x_1$ 不能作为一个端点，因为不存在只包含 x_1 而不包含其他 $x \in P$ 的关系。此外，我们不能将 x_6 和 x_5 纳入 x_3 与 x_4 之间的无弦路径，因为 R_3 可以作为捷径。

此外，我们可以找到一条长度为3的 q-无弦路径 $P \leftarrow \{x_7, x_8\}$，其边序列为 $P_3 \leftarrow\{R_4, R_5, R_6\}$，使得 R_4 是具有输出属性 x_4 的输出端点。

有了无弦路径后，我们现在可以定义查询的高度：

定义*3.6*。给定一个合取查询 Q，令 $\mathrm{H} = (\mathrm{V}, \mathrm{E})$ 为 Q 的关系超图。如果 H 不包含长度大于 $2k$ 的无弦路径，不包含长度大于 k 的 q-无弦路径，并且

- H 包含一条长度为 $2k$ 或长度为 $2k-1$ 的无弦路径；并且/或者
- H 包含一条长度为 k 的 q-无弦路径。

引理 3.7。当且仅当一个合取查询 Q 是 q-层次化的时，它的高度为1。

将查询限制为自由连通后，我们可以将查询高度与自由连通连接树高度联系起来。其核心思想是，从 Q 中移除所有耳节点会生成一个高度降低一层的残余查询 Q'。此外，通过重新附加被移除的耳，可以将 Q' 的自由连通连接树提升为 Q 的自由连通连接树，并使树高度增加一层。由于 q-层次化（高度为1）的查询存在高度为1的自由连通连接树，对高度进行归纳可知，每个高度为 k 的查询都存在高度为 k 的自由连通连接树。

另一方面，Q 的关系超图中的每条无弦路径都必须在任意自由连通连接树中实现，这排除了高度为 $k-1$ 的自由连通连接树存在的可能性，因此高度为 k 的查询不可能拥有这样的连接树。结合这两个方向，即得出以下引理，详细证明见附录 B。

引理 3.8。给定一个高度为 k 的自由连通查询 Q，Q 存在一棵高度为 k 的自由连通连接树，但不存在高度为 $k-1$ 的自由连通连接树。

3.3 定理 1.1 的证明

在高度为 k 的查询中找到长度为 $2k$ 或 $2k-1$ 的无弦路径，或在非全且非布尔的高度为 k 的查询中找到长度为 k 的 q-无弦路径后，我们现在可以利用无弦路径及其诱导查询，建立高度为 k 的查询与环检测问题之间的联系，并证明定理 1.1 关于维护高度为 k 的查询难度的结论。

给定一个高度为 k 的查询 $Q = (\mathrm{V}, \mathrm{E}\ \bar{y})$，我们随后可以模拟 Q_{2k-1}、Q_{2k} 或 Q'_k。不失一般性地，我们以 Q_{2k-1} 为例。给定针对 Q_{2k-1} 的更新序列 S，我们首先从 Q 中识别出一条长度为 $2k-1$ 的无弦路径 $P = \{x_1, \cdots, x_{k-2}\}$，它对应于 Q_{2k-1} 中的所有属性。对于满足 $v \notin P$ 的每个 $v \in \mathrm{V}$，我们将其定义域设为一个特殊值 $\perp$。随后，我们可以为 Q 创建一个等价的更新序列 S'，使得对于每个 $R(\bar{x}) \in \mathrm{E}$，我们按照以下规则为 R 创建更新：

• $\bar{x} \cap P = \varnothing$。我们在所有更新的开头插入元组 $(\perp, \cdots, \perp)$，用于 R；• $\bar{x} \cap P = \{x_i\}$，其中 x_i 不是 P 的端点。我们通过为所有 $u \in \mathrm{dom}(x_i)$ 赋值 $x_i = u$，并将所有其他属性赋值为 $\perp$，在所有更新的开头插入元组 $(u, \perp, \cdots, \perp)$；• $\bar{x} \cap P = \{x_i\}$，其中 x_i 是 P 的端点。当存在针对 Q_{2k-1} 的更新 (u) 将 R_1（或 R_{k-1}）更新为 $(u, \perp, \cdots, \perp)$ 时，我们使用 $x_1 = u$（或 $x_{k-2} = u$）以及将所有其他属性设为 $\perp$，从 R 更新 $(u, \perp, \cdots, \perp)$；• $\bar{x} \cap P = \{x_i, x_{i+1}\}$。当存在将 R_i 更新为 v）的更新 $(u$ 时，我们使用 $x_i = u$ 和 $x_{i+1} = v$，并将所有其他属性设为 $\perp$，从 R 插入或删除 $(u, v, \perp, \cdots, \perp)$。

由于无弦路径的性质， P中不存在包含超过两个属性的 R，也不存在带有 $|j - i| > 1$的 $\bar{x} \cap P = \{x_i, x_j\}$。不难看出，查询结果 Q 与 Q_{2k-1}之间存在一一映射，并且该结果可以轻易推广至 Q_{2k}和 Q'_k。因此，给定任意针对 Q_{2k-1}、 Q_{2k}或 Q'_k的更新序列 S，只要 Q具有长度为 $2k-1$的无弦路径，或长度为 k的q-无弦路径，我们就可以在 Q 上构造一个等价的更新序列S'，从而维护一个高度为 k的查询应当与维护 Q_{2k-1}、 Q_{2k}或 Q'_k一样困难。

4 扩展时的维护困难性

在第 3 节中，我们分析了通过添加额外关系来扩展查询的参数化困难性。很自然会问，两个高度相同的查询是否仍会因其连接属性或独有输出属性的结构不同，而在维护难度上有所差异[3]。为刻画这一影响，我们引入查询的第二个参数，即维度 。为定义维度，我们首先形式化查询上连通子集 的概念。

定义*4.1*（连通子集）。给定合取查询 $Q = (\mathrm{V}, \mathrm{E}\ \bar{y})$，令 $\mathrm{V}_c \subseteq \mathrm{V} \backslash \mathrm{V}_\bullet$ 为连接属性的一个子集。我们定义诱导活动关系 $\mathrm{E}_c = \{R(\bar{x}) \in \mathrm{E} \mid \bar{x} \cap \mathrm{V}_c \neq \varnothing\}$。为进一步完善定义，当 $\mathrm{V}_c = \{\varnothing\}$时，每个 $R_i(\bar{x}_i) \in \mathrm{E}$ 都可以作为活动关系 $\mathrm{E}_c = \{R_i\}$。

如果原图 $G(\mathrm{V}_c)$ 是连通的，我们称 V_c为连通子集 。令 $\mathrm{V}(\mathrm{E}_c) = \cup_{R(\bar{x}) \in \mathcal{E}_c} \bar{x}$表示出现在 E_c中的属性集合。我们将活跃属性 集合定义为 $\mathrm{V}_\star = \mathrm{V}(\mathrm{E}_c) \backslash \mathrm{V}_c$。

定义 *4.2*（维度）。令 $Q = (\mathrm{V}, \mathrm{E}\ \bar{y})$ 为一个合取查询。令 $\mathrm{V}_c \subseteq \mathrm{V}$ 为一个连通子集，其诱导关系为 E_c。剩余关系为 $\mathrm{E}_r = \mathrm{E} \backslash \mathrm{E}_c$。

若存在一个单射映射π: $\mathrm{N} \to \mathrm{V}_\star$，使得对于每个 $R(\bar{x}) \in \mathrm{N}$， $\pi(R) \in \bar{x} \cap \mathrm{V}_\star$，则称关系子集 $\mathrm{N} \subseteq \mathrm{E}_r$ 为一个不同邻居集。我们将该映射的像 $\mathrm{V}_k(\mathrm{N}) = \{\pi(R) \mid R \in \mathrm{N}\}$称为键集。注意，对于固定的 N，可能存在多个键集。

V_c的局部维度记作 $\dim(\mathrm{V}_c)$，定义如下：

$$\dim(\mathcal{V}_c) = \max_{\mathcal{N} \subseteq \mathcal{E}_r,\ \mathcal{V}_k(\mathcal{N})} \begin{cases} |\mathcal{V}_k(\mathcal{N})| & \text{if } \mathcal{V}_c \cap \bar{y} \neq \varnothing \text{ or } \mathcal{V}_k(\mathcal{N}) \subseteq \bar{y}, \\ |\mathcal{V}_k(\mathcal{N}) \cup (\bar{y} \cap \mathcal{V}_\star)| & \text{otherwise.} \end{cases}$$

直观而言，如果 V_c包含输出属性，或者 $\mathrm{V}_k(\mathrm{N})$仅包含输出属性，我们只计算 $\mathrm{V}_k(\mathrm{N})$的贡献。否则，我们还会计算其他任意活跃输出属性。

The 查询维度 是所有连通子集的最大局部维度 :

$$\dim(Q) = \max_{\mathcal{V}_c \subseteq \mathcal{V}} \dim(\mathcal{V}_c).$$

现在我们 建立查询维度与层次结构之间的联系。

引理 4.3. 当且仅当对于所有$\mathcal{V}_c \subseteq \mathcal{V}$，$\dim(\mathcal{V}_c) = 1$，$CQ$ 才是 q-层次化的。

结合引理 3.7，我们能够证明维度为 1 的查询、高度为 1 的查询和 q-层次查询之间的等价性，并证明命题 1.3。

[3]对于唯一的非输出属性，我们可以将这些属性投影掉，而不影响查询结果 .

例*4.4*。 考虑例2.7中的查询 Q_1 from。它是一个维度为4的查询，其维度为 dim $(\{x_5\}) = 4$。这里，连通集为 $\mathrm{V}_c = \{x_5\}$ ，诱导关系为 $\mathrm{E}_c = \{R_3, R_7\}$。图 2c 展示了一个使维度最大化的不同邻居集 $\mathrm{N} = \{R_2, R_4, R_8, R_{10}\}$ 。相应的键集 $V_k(\mathrm{N})$ 包含 $\{x_3, x_4, x_6, x_9\}$。

4.1 星形查询

我们通过星形查询这一构造，将维度为d 的合取查询与广义在线矩阵向量乘法（OuMv_k）问题的难度联系起来。星形查询中存在一个中心关系，它与一组卫星关系相连接。所有这些卫星关系之间都不共享任何公共属性，而中心关系则与每个卫星关系共享公共属性。不失一般性，我们考虑以下形式的星形查询：

定义 *4.5*（星形查询）。查询 $Q = (\mathcal{V}, \mathcal{E}\ \bar{y})$ 是一个星形查询，当且仅当其形式如下 ：

$$Q(x_{d-j+1}, \ldots, x_d) \leftarrow R_1(x_1, \cdots, x_d), R_2(x_1), \ldots, R_{k+1}(x_k)$$

对于任意 $0 \leqslant d - j \leqslant k \leqslant d$，其中 R_1是中心关系， $R_2, \ldots, R_{k+1}$是卫星关系， d是查询的维度， j是输出属性的数量。存在两种特殊情况：$j = d, k = d$（完整查询）或 $j = 0, k = d$（不含输出属性的布尔查询）。

星形模式在数据仓库设计中无处不在：中心关系 R_1 是事实表，所有其他卫星关系都是维度表。实践中，这些连接通常发生在主键和外键之间。尽管我们在查询定义中不施加键约束，但我们的下界实例满足这些约束；与此同时，先前的研究表明，在先进先出等受限更新序列下，施加键约束可能会改变维护这些序列的难度 [36]。

任何星形查询都是高度为 2 的查询：通过将R_1 作为根节点，并将所有卫星作为连接到 R_1的叶节点，可以得到一棵高度为 2 的自由连通连接树 T。然而，我们可以证明，它们的下界可能超过定理 1.1 中确立的 $O(\sqrt{|D|})$ 下界：

引理4.6。 考虑星形查询： $Q_d() \leftarrow R_1(x_1, \ldots, x_d), R_2(x_1), \ldots, R_{d+1}(x_d)$. 该查询的维度为 d。除非 *OuMv*$_k$ 猜想不成立，否则无法以摊销 $O(|D|^{(d-1)/d-\epsilon})$ 时间维护 Q_d，并以 $O(1)$ 延迟进行枚举，其中 $\epsilon > 0$为任意常数。

证明。我们将 OuMv_k 问题归约为维护 Q_d。令 M为一个 d 维布尔张量，其大小为 $n \times \cdots \times n$，并令 $u_1, \ldots, u_d$为一系列 n^γ向量集合。由于当 $n = |D|^{1/d}$,时$\gamma \geqslant d - 1$，输入规模总计上界为 $(|D|^{1/d}) \times (|D|^{1/d})^\gamma = |D|^{(\gamma+1)/d}$。给定一个 OuMv_k实例，我们可以通过以下归约解决该问题：

(1) 将张量 M编码为 R_1：当且仅当 $M[t_1, \ldots, t_d] = 1$时，将元组 $(t_1, \ldots, t_d)$ 插入 R_1。(2) 对于每一轮 j的向量更新和查询：• 对于每个 $i \in [1, d]$，通过在 R_{i+1}中插入 (x_i) 来更新它，使其表示向量 $u_i^{(j)}$，条件是 $u_i^{(j)}[x_i] = 1$。• 枚举 Q_d的结果。当且仅当 M 与 $u_1^{(j)} \times \cdots \times u_d^{(j)} = 1$的交集非空时，该结果非空。• 从 $R_2, \ldots, R_{d+1}$中删除元组，为第 $j + 1$ 轮做准备。

如果 Q_d 可以在每次更新耗时 $O(|D|^{(d-1)/d-\epsilon})$ 的时间内维护，那么总更新时间将为 $O(|D|^{(d-1)/d+(\gamma+1)/d-\epsilon}) = O(|D|^{1+\gamma/d-\epsilon}) = O(n^{d+\gamma-d\epsilon})$,，枚举每个查询需要 $O(1)$，从而可以在 $O(n^{d+\gamma-\epsilon})$ 时间内解决 OuMv_k 问题，这与 OuMv_k 猜想相矛盾。□

引理 4.7。 考虑星形查询 $Q'_d(x_{d-j+1}, \cdots, x_d) \leftarrow R_1(x_1, x_2, \cdots, x_d), R_2(x_1), \cdots, R_{k+1}(x_k)$, 对于任意 $0 \leqslant d - j \leqslant k \leqslant d$。除非 *OuMv*$_k$ 猜想不成立，否则无法在$O(|D|^{(d-1)/d-\epsilon})$ 时间内维护 Q'_d，同时支持任意 $\epsilon > 0$ 的 $O(|D|^{(d-j)/d-\epsilon})$-延迟枚举。

证明。我们改编引理 Q_d 证明中的构造。在此情形下，枚举需要输出张量 M 的切片。对于每个输出元组 $(t_{d-j+1}, \ldots\ldots, t_d)$，前 k 个维度都存在有效的连接结果。因此，对于每个候选输出，我们只需验证剩余维度（从 $k+1$ 到 d）的交集是否存在于 $u_{k+1} \times \cdots \times u_d$ 中。我们可以在笛卡尔积 $u_{k+1} \times \cdots \times u_d$ 上构建哈希索引，以便对每个条目在 $O(1)$ 时间内完成检查。计算该索引每轮需要 $O(|D|^{(d-k)/d})$ 时间，而对 Q'_d 执行到剩余维度的去重投影会额外带来 $O(|D|^{(k+j-d)/d})$ 延迟。

如果摊销更新时间为 $O(|D|^{(d-1)/d-\epsilon})$，且枚举 $\pi_{x_{k+1}, \cdots, x_d} Q'_d$的枚举延迟为 $O(|D|^{(d-j)/d-\epsilon} \times |D|^{(k+j-d)/d}) = O(|D|^{k/d} - \epsilon),$，由于每轮总共有$O(|D|^{(d-k)/d})$ 个输出，总运行时间将为 $O(n^{\gamma+d-1-\epsilon'})$，这再次违背了该猜想。 □

4.2 定理 1.2 的证明

我们通过从星形查询（Q_d和 Q'_d）的困难性归约到一个维度为 $\dim(Q) = d$ 的任意查询 Q，来证明该定理。设 V_c 为连通子集，E_c 为诱导活动关系，N 为不同邻居集，$\mathrm{V}_k(\mathrm{N})$ 为达到最大维度的键集（即 $\dim(\mathrm{V}_c) = d$）。我们构造一个 Q 的实例，以如下方式模拟困难查询 Q_d（或 Q'_d）。

令 M 为输入张量，u_1、……、u_d为与 Q_d 相关联的 OuMv_k 问题实例的输入向量。我们通过以下步骤将该实例映射到 Q：

(1) 我们将 d 个维度映射到 Q 中的属性。如果 $\mathrm{V}_c \cap \bar{y} \neq \varnothing$ 或 $\mathrm{V}_k(\mathrm{N}) \subseteq \bar{y}$，我们只需将维度 x_1、……、x_d映射到 $\mathrm{V}_k(\mathrm{N})$ 中的属性。否则，我们将前 x_1、$\cdots$、x_k 个维度映射到 $\mathrm{V}_k(\mathrm{N})$ 中的属性，并将剩余维度 x_{k+1}、……、x_d映射到 $(\bar{y} \cap \mathrm{V}_\star) \backslash \mathrm{V}_k(\mathrm{N})$。此映射涉及的所有属性均赋予定义域 $[n]$。

(2) 连通子集 V_c 中的属性充当“粘合剂”，用于重构张量 M。我们将它们的定义域设为 $[N]$，其中 $N = n^d$。令 V' 为归约中所有活跃属性的集合：$\mathrm{V}' = \mathrm{V}_c \cup \mathrm{V}_k(\mathrm{N})$ 或 $\mathrm{V}' = \mathrm{V}_c \cup \mathrm{V}_k(\mathrm{N}) \cup (\bar{y} \cap \mathrm{V}_\star)$ 取决于 V_c 和 $\mathrm{V}_k(\mathrm{N})$。任意属于 $v \in \mathrm{V} \backslash \mathrm{V}'$ 的属性均固定为常数值 $\perp$。我们为所有 $\bar{x} \cap \mathrm{V}' \neq \varnothing$ 定义 $R(\bar{x})$ 的更新序列如下：

（3）**张量编码：**考虑满足 $|\bar{x} \cap \mathrm{V}'| \geqslant 2$ 的关系。这些关系构成核心结构。我们用唯一标识符 $ID \in [1, N]$ 为张量 M 的非零项建立索引。对于每个满足 $\mathrm{V}_c \cap \bar{x} \neq \varnothing$ 的关系 $R(\bar{x})$，我们将张量M投影到属性 $\bar{x} \backslash \mathrm{V}_c$ 上（不去除重复项），并将 V_c 中的那些变量设为相应的张量项 ID。如果 $R(\bar{x})$ 仅包含来自 V_c 的变量（即 $\bar{x} \cap \mathrm{V}' \subseteq \mathrm{V}_c$），我们只需针对所有活跃标识符插入元组 $(ID, \ldots\ldots, ID)$。这确保了 V_c 的连通性会在整个查询中传播同步 ID。如果 $R(\bar{x})$ 不包含来自 V_c 的变量，我们便为属性 $\bar{x} \cap \mathrm{V}'$ 插入张量M的去重投影，而不约束查询输出。

(4) 中和未使用的关系。 对于任意满足 $R(\bar{x})$ with $\bar{x} \cap \mathrm{V}' = \varnothing$ 的关系，我们在更新流开始时向其中填入单个元组 $(\perp, \ldots, \perp)$。这确保这些关系不会限制连接结果。

(5) 向量更新： 在 (3) 和 (4) 之后，我们开始更新向量。考虑满足 $\bar{x} \cap \mathrm{V}' = \{v\}$ 的关系，其中某个 $v \in \mathrm{V}_k(\mathrm{N})$。根据不同邻居集的定义，这类关系必然存在，并且与特定维度一一对应。我们将向量 u_i 的更新流直接映射到该关系。向该关系中插入值 c，对应于设置 $u_i[c] = 1$。

正确性。该构造确保，当且仅当存在一个元组标识符 ID（表示 M中的一个非零项），且其所有坐标值都出现在相应的卫星关系中（表示向量 u_i 的值为 1）时，才存在连接结果。这恰好重构了

条件 $M(x_1,\ldots,x_d) \wedge \bigwedge u_i(x_i)$。数据库的大小由 M和向量更新主导，从而保留下界 $O(|D|^{(d-1)/d-\epsilon})$。

5 使用参数化复杂性维护星形查询

在第3节和第4节中，我们证明了，对于高度为 k 且维度为 d 的查询 Q，其维护所需的时间无法优于摊销 $O(|D|^{1-1/\max\{k,\ d\}-\epsilon})$ 时间。本节中，我们为星形查询这一类给出与之匹配的上界。我们引入一个依赖于数据的参数——广义 H 指数，并设计了一种摊销更新时间为 $O(h(D)^{\dim(Q)-1})$的算法。由于 $h(D)$ 自然地受 $|D|^{1/\dim(Q)}$, 限制，这就证明了对于这些查询，我们的下界是最优的。

5.1 广义 H 指数

经典 H 指数是一种广泛使用的指标，用于衡量学者出版物的产出和引用影响力。如果一位学者至少有 10 篇出版物，每篇至少获得 10 次引用，且没有 11 篇出版物各自获得至少 11 次引用，那么其 H 指数就是 10。参数化图算法中也采用了同样的思路 [17, 18], : 将图划分为稠密部分（由拥有许多邻居的顶点诱导）和稀疏剩余部分。不过，标准 H 指数是针对图等二元关系定义的。因此，我们引入一个广义版本，用于衡量星形查询中关系的稠密程度。

定义*5.1*（广义 H 指数）。考虑一个星形查询 Q，其中心关系为 $R_1(x_1,\ \ldots\ldots,\ x_d)$, 维度为 $d = \dim(Q)$。对于任意属性 $x_i \in \{x_1,\ \ldots\ldots,\ x_d\}$，令 $\deg_{R_1}(v)$ 表示投影 $\pi_{x_i}R_1$中某个值 v的频数。我们将特定属性 x_i的 H 指数记为 $h_i(D)$，其定义为满足以下条件的最大整数：$\pi_{x_i}R_1$中至少有 $h_i(D)$ 个不同的值，其频数均至少为 $h_i(D)^{d-1}$。形式化地说：

$$h_i(D) = \max\left\{k \in \mathbb{N} \;\middle|\; \left|\left\{v \in \pi_{x_i}R_1 : \deg_{R_1}(v) \geqslant k^{d-1}\right\}\right| \geqslant k\right\}.$$

查询 Q在数据库 D上的广义 H 指数定义为 $h(D) = \max_{i=1}^{d} h_i(D)$。

命题 5.2。对于任意数据库实例 D，广义 H 指数满足 $h(D) \leqslant |D|^{1/\ \ \ \

视图维护。我们将查询 Q 维护为 2^d 个子查询 $Q^{(\mathbf{b})}$ 的并集，对应于$R_1^{(\mathbf{b})}$。对于固定的 $\mathbf{b}$，令 $I_H = \{i \mid \mathbf{b}[i] = 1\}$ 表示重属性的索引。我们维护：
(1) 卫星投影： 对于每个 $R_i^{(\mathbf{b}[i])}$，我们维护其投影：$V_{R_i}^{(\mathbf{b}[i])} = \pi_{x_{i-1}} R_i^{(\mathbf{b}[i])}$。**(2) 轻过滤核心（$V_S$）：** 对于中心关系 $R_1^{(\mathbf{b})}$，我们主动用轻卫星对其进行过滤。我们维护半连接视图：$V^{(\mathbf{b}_S)} = R_1^{(\mathbf{b})} \ltimes \left(\ltimes_{i \notin I_H} V_{R_{i+1}}^{(0)} \right)$. 由于轻键的度较低，因此轻卫星的更新易于传播。**(3) 重核心（V_C）：** 我们进一步将轻过滤核心投影到重属性上，并通过重卫星的笛卡尔积进行过滤：$V^{(\mathbf{b}_C)} = \left(\pi_{\{x_i\}_{i \in I_H}} V_S^{(\mathbf{b})} \right) \ltimes \left(\times_{i \in I_H} V_{R_{i+1}}^{(1)} \right)$. V_C 仅包含完全由重键组成且通过了所有卫星过滤的元组。

枚举。为枚举 Q 的结果，我们遍历所有 $\mathbf{b} \in \{0,\ 1\}^d$。对于每个 $\mathbf{b}$，我们：1. 枚举元组 $t_{core} \in V_C^{(\mathbf{b})}$。2. 使用 t_{core} 查询 $V^{(\mathbf{b}_S)}$ 的索引，查找通过轻筛选的匹配元组$t_{full} \in R_1$。3. 对于每个 t_{full}，将其与卫星连接（使用标准的常数延迟查找），扩展为最终结果。由于该查询是自由连通的 [24, 35]，此过程可保证 $O(1)$延迟枚举。随后，我们对输出属性 $\bar{y}$ 执行去重投影，这会额外增加 $O(|D|^{(d-j)/d})$ 延迟。

重新平衡。到目前为止，我们假设重/轻划分是固定的，且 $h(D)$ 也固定不变。然而，在动态场景中，随着更新到来，某个键被分类为重键还是轻键，以及广义 H 指数 $h(D)$，都可能随时间变化。我们使用一种标准的重新平衡技术来处理这些变化，该技术类似于动态表的教科书方法 [14]。

(1) 键重新分类： 只有当某个键 x_i 的度超过 $2\tau_i$ 时，我们才将其提升为重键；只有当其度降至 $\frac{1}{2}\tau_i$以下时，才将其降级。这确保了某个键每两次状态变化之间，必定发生过涉及该键的 $\Omega(h_i(D)^{d-1})$ 次更新。

为了执行这样的重新平衡，我们首先从 R_1中删除所有相关元组，然后从 R_{i+1}中删除所有相关元组，再将它们以更新后的标签重新插入 R_{i+1}，然后插入 R_1。这些更新为在划分之间移动元组所需的 $O(h_i(D)^{d-1})$ 开销提供了补偿，因为此类键的频数为 $O(h_i(D)^{d-1})$，并使每次更新的摊销成本为 $O(1)$。

(2) 全局参数更新： 如果任一维度中的重键数量超过 $2h(D)$，或者与上次参数更新相比，$h(D)$ 减少了一半，我们就使用一个能够实时跟踪的新 $h(D)$ 重建数据结构。

发生全局再平衡时，重集和轻集之间最多有 $O(h(D))$ 个键需要移动，最坏情况下成本为 $O(|D|) = O(h(D)^d)$。然而，要达到需要重建的状态，必须发生 $\Omega(h(D) \cdot h(D)^{d-1}) = \Omega(h(D)^d)$ 次更新，例如，至少有$h(D)$ 个键从轻集移至重集，且每个键的频数都从上一次全局再平衡时至多为$h(D)^{d-1}$ 变为 $2h(D)^{d-1}$；或者至少有 $0.5h(D)$个重键从重集移至轻集，且其频数从上一次全局再平衡时至少为 $h(D)^{d-1}$ 变为 $0.5h(D)^{d-1}$。将这项成本分摊到这些更新上，得到摊销成本 $O(1)$。

5.3 复杂度分析

不难看出，该算法的空间需求与 $O(|D|)$成线性关系；因此，要完成定理 1.4 的证明，只需分析摊销更新复杂度。更新成本由三部分组成：维护成本、再平衡成本，以及跟踪$h(D)$的成本。在这三部分中，我们还证明了再平衡步骤的每次更新摊销成本仅为常数；现在我们将重点关注其余两部分。

*Main*维护成本。我们分析单个元组更新的成本，假设 $h(D)$ 是常数 .

• **更新中心** R_1**：** 向 R_1 中插入一个元组 t 只会影响一个分区 $R_1^{(\mathbf{b})}$。使用合适的哈希索引，检查 V_S的半连接并更新 V_C，每次更新需要 $O(1)$ 时间。• **更新轻卫星** $R_i^{(0)}$**：** 在此处插入的元组具有连接键 k，其度为$_{R_1}(k) < \tau_i$。此次更新可能会触发向 $V^{(\mathbf{b}_S)}$ 中插入元组。由于度有上界 $\tau_i = O(h_i(D)^{d-1})$，因此 R_1中至多有 $O(h_i(D)^{d-1})$ 个元组会被激活。将这些元组传播至 V_C，每个 R_1 元组还需要常数时间，因此总成本为 $O(h_i(D)^{d-1})$。• **更新重卫星** $R_i^{(1)}$**：** 连接键 k是重值。这不会影响 V_S，但可能会更新笛卡尔积 $\times_{i\in I_H} V_{R_{i+1}^{(1)}}$。由于每个维度至多有 $h_i(D)$ 个重值，笛卡尔积中受影响的元组数量实际上有上界 $O(h(D)^{|I_H|-1})$，因此对 V_C的更新有上界 $O(h(D)^{|I_H|-1}) \leqslant O(h(D)^{d-1})$。

因此，在重/轻分类固定的情况下，更新成本为 $O(h(D)^{d-1})$。

跟踪 $h(D)$。维护算法还需要跟踪所有维度 i上 $h_i(D)$ 的变化，从而 $h(D)$。为此，对于每个维度 i，我们维护两个分组计数查询 $V_{\mathsf{cnt}_i} = \pi_{x_i,\ \text{计数}(*)}$ 作为 $\deg_{R_1}$，以及 $V_{\mathsf{feq}_i} = \pi_{\mathsf{deg},\ \mathsf{count}(*)}$ 作为 $\mathsf{feq}^{V_{\mathsf{cnt}_i}}$。在对 R_1进行单元组更新时，这两个查询可以在 $O(1)$ 时间内维护；通过维护结果 $h_i = \max\left\{\mathsf{cnt} \,\middle|\, \sum_{\text{维度}i=}^{|D|^{1/}} {}^{(Q)}\ \mathsf{cnt}\left(V_{\mathsf{feq}_i}[\mathsf{cnt}].\mathsf{feq}\right) \geqslant \mathsf{cnt}^{\dim(Q)-1}\right\}$，我们可以实时获得 h_i，而该结果也可以通过维护频率直方图在 $O(1)$ 时间内维护。

综合所有部分，每次元组更新的摊销更新时间为 $O\left(h(D)^{\dim(Q)-1}\right)$。由于始终有 $h(D) \leqslant |D|^{1/\dim(Q)},$，这与定理 1.2 对所有星形查询给出的下界相匹配，因此证明了该类查询的界是紧的。

6 其他讨论与未来工作

在这项工作中，我们引入了两个结构参数——高度 和维度，以建立在更新条件下维护合取查询的参数化下界。需要指出的是，高度 依赖于组合 k-团猜想，而该猜想仅排除了组合算法的存在。尽管组合算法 这一概念尚未得到正式定义，但由于组合算法具有实用高效、简洁优雅且易于推广等特点，它已被数据库理论等不同研究领域广泛采用 [16, 19, 31], [2]。这也立即引出了一个值得研究的问题，涉及上界设计和下界设计：我们能否利用快速矩阵乘法加速动态查询求值?

另一个关键问题是，能否设计出匹配这些下界的维护算法。为了补充我们的下界，我们研究了星形查询的维护问题，它是分析型工作负载中的一个核心子类。我们基于广义 H 指数 $H(D)$建立了一个参数化上界。它将先前的权衡算法 [27, 28] 推广到依赖具体实例的情形，表明效率往往取决于数据倾斜，而非最坏情况下的规模。推广此类技术将有益于理论和系统研究。另一方面，如何在枚举延迟和维护成本之间实现权衡仍是一个开放问题，尤其是对于具有 $\omega(Q) > 1$的查询。

此外，弥合一般查询方面的差距仍然是一项重大挑战。具体而言，我们需要设计新的算法来处理类似 *Loomis-Whitney* 的查询 [5, 27], ，这类查询包含嵌套连接键，使星形查询的新维护技术更加复杂。未来的研究方向包括将上界和下界拓展到一般情形，例如比较 [25, 37], 、并集 [11], 以及针对特定更新模式的查询 [5, 23]。

A 引理3.7的证明

若方向：令 $Q = (\mathrm{V}, \mathrm{E}\ \bar{y})$ 为高度为 1 的查询。若 Q 不是 q-层次化的，则 (1) 存在 $x_1, x_2 \in \mathrm{V}$，使得 $\mathrm{E}[x_1] \cap \mathrm{E}[x_2] \neq \varnothing$ 且 $\mathrm{E}[x_1] \backslash \mathrm{E}[x_2] \neq \varnothing, \mathrm{E}[x_2] \backslash \mathrm{E}[x_1] \neq \varnothing$；或 (2)$x_1 \in \bar{y}, x_2 \notin \bar{y}$，而 $\mathrm{E}[x_1] \subsetneq \mathrm{E}[x_2]$。然而，对于 (1)，令 $e_1 \in \mathrm{E}[x_1] \cap \mathrm{E}[x_2], e_2 \in \mathrm{E}[x_1] \backslash \mathrm{E}[x_2], e_3 \in \mathrm{E}[x_2] \backslash \mathrm{E}[x_1], P \leftarrow \{x_1, x_2\}$ 构成一条长度为 3 的无弦路径，其中 $P_e \leftarrow \{e_2, e_1, e_3\}$ 是其边序列，这与 Q 是高度为 1 的查询相矛盾；对于 (2)，令 $e_1 \in \mathrm{E}[x_1] \cap \mathrm{E}[x_2], e_2 \in \mathrm{E}[x_1] \backslash \mathrm{E}[x_2]$，则 $P \leftarrow \{x_2\}$ 是一条长度为 2 的 q-无弦路径，其中 $P_e \leftarrow \{e_1, e_2\}$ 是一条边序列，且 e_1 包含输出属性 x_2，这同样与 Q 是高度为 1 的查询相矛盾。因此，关于 Q 不是 q-层次化的假设不成立。

仅当方向： 类似地，设 Q 是一个 q-层次查询；我们无法找到长度为 $\geqslant 3$ 的无弦路径，否则我们可以找到一对属性 x_1，x_2 使得 $\mathrm{E}[x_1] \cap \mathrm{E}[x_2] \neq \varnothing$ 且 $\mathrm{E}[x_1] \backslash \mathrm{E}[x_2] \neq \varnothing$，$\mathrm{E}[x_2] \backslash \mathrm{E}[x_1] \neq \varnothing$。类似地，我们无法找到长度为 $\geqslant 2$ 的 q-无弦路径；否则，我们可以找到属性 x_1，x_2 使得 $x_1 \in \bar{y}$、$x_2 \notin \bar{y}$，而 $\mathrm{E}[x_1] \subsetneq \mathrm{E}[x_2]$。这两种情况都违背了 Q 是 q-层次化的事实；因此，该查询的高度必为 1。

B 引理 3.8 的证明

To p为证明引理 3.8，我们首先展示高度定义的递归性质 n.

引理 *B.1. </style}*自由连通查询 Q 是高度为 k 的 $k > 1$ 查询，当且仅当其残余查询Q' 是高度为 $k-1$ 的查询。

证明。当 $k = 2$时，即对于弱 q-层次化查询，引理显然成立，因为它们的残余查询是 q-层次化的。现在考虑一般情形 $k > 2$。根据残余查询和耳的定义，给定一棵自由连通连接树 T，每个叶节点都对应一个耳，其父节点是该耳的锚点。因此，我们可以得出两个重要结论：(1) 如果Q 有一棵高度为 k的连接树，那么其残余查询 Q' 有一棵高度至多为 $k-1$的连接树。这是因为从 T 中移除所有叶节点（耳）后，得到的连接树高度为 $k-1$。如有必要，继续从所得树中移除任何剩余的耳，只会进一步降低树的高度。(2) 因此，如果残余查询 Q' 存在一棵高度为 $k-1$的自由连通连接树 T'，那么我们可以将移除的每个耳 R 作为其锚点 R' 在 T'中的子节点，从而为 Q 构造一棵高度至多为 k 的连接树。所得结构仍是一棵有效的自由连通连接树，满足连接、覆盖和连通条件，且不违反守卫或上方条件。由于添加这些耳只会引入新的叶节点，树的高度会增加 1；因此，重构得到的 Q 连接树高度至多为 k。

我们首先证明“若”方向。假设残余查询 Q' 是一个高度为 $k-1$ 的查询。设 Q 具有一棵高度为 $\leqslant k-1$ 的自由连通连接树 T。根据上方的观察（1），从 T 中移除所有耳后，可得到一棵高度至多为 $k-2$ 的 Q' 连接树。这与 Q' 是一个高度为 $k-1$ 的查询这一假设矛盾，因为这样 Q' 就会拥有一棵高度小于 $k-1$ 的连接树。因此，Q 不可能具有高度小于 k 的自由连通连接树。另一方面，根据观察（2），我们知道如何为 Q 构造一棵高度为 k 的连接树。综上，Q 的连接树可能达到的最小高度为 k。换言之，Q 是一个高度为 k 的查询。

对于“仅当”方向，假设 Q 是一个高度为 k 的查询。根据定义，Q 的每棵自由连通连接树的高度都至少为 k，并且存在一棵高度为 k的连接树 T。从这棵 T 中移除所有叶节点，得到一棵高度为 $k-1$的连接树 T'；进一步从 T'中移除耳只会降低高度，因此 Q'的高度至多为 $k-1$。现在，反设 Q' 不是高度为 $k-1$ 的查询。那么，存在一棵高度至多为$k-2$的 Q' 自由连通连接树。利用观察（2）中的构造，将先前移除的耳附加到这棵较矮的连接树上，我们可以得到一棵高度至多为 $k-1$的 Q 连接树。这将与

假设 Q的高度为 k相矛盾。因此， Q' 的连接树高度必定达到最小值 $k-1$，这意味着Q' 是一个高度为 $k-1$ 的查询。□

由于每个高度为 1 的查询都是 q-层次化的，并且具有高度为 1 的连接树（引理 3.7），设 Q' 是从 Q 中移除所有耳后得到的残余查询，且 T' 是 Q' 的高度为 k' 的自由连通连接树，则可以通过将每个耳节点添加为其锚节点下的叶子节点，并将树的高度增加 1，构造出 Q 的高度为 $k'+1$ 的自由连通连接树。因此，我们可以通过以下步骤为高度为 k 的查询 Q 构造高度为 k 的自由连通连接树：(1) 递归地移除所有耳节点，得到残余查询 Q_2，$\cdots$，Q_k，其中 Q_k 的高度为 1；(2) 找到 Q_k 的高度为 1 的自由连通连接树 T_k；(3) 对于 Q_{k-1}，$\cdots$，Q_2，Q，将所有耳添加回 T_k，得到高度为 2 的 T_{k-1}，$\cdots$，T_2，T，$\cdots$，$k-1$，k。

此外，为了证明 Q不存在高度为 $k-1$ 的自由连通连接树，我们证明每条无弦路径都必须在任意自由连通连接树中实现：

引理 B.2。 给定一个 CQ $Q=(\mathcal{V},\mathcal{E}\ \bar{y})$，设 $P\leftarrow\{v_1,\cdots,v_{k-1}\}$ 是Q的关系超图 $\mathcal{H}$上的一条长度为 k 的无弦路径。对于 Q 的任意连接树 $\mathcal{T}$，设 n_1 是一个满足 $v_1\in n_1$ 的节点，且对所有 $i\in[2,k-1]$ 都有 $v_i\notin n_1$；设 n_l是一个满足 $v_{k-1}\in n_l$ 的节点，且对所有 $i\in[1,k-2]$都有 $v_i\notin n_l$；那么连接树上连接 n_1 与 n_l的路径 $P_{\mathcal{T}}$包含 P中的所有属性。此外，这样的路径长度至少为 k。

引理 B.2 的证明。我们先考虑一条长度为 2 的路径 v_1,v_2。根据定义，我们知道存在 e_1,e_2,e_3，使得 $v_1\in e_1$ 且 $v_2\notin e_1,v_1,v_2\in e_2$，并且 $v_2\in e_3$ 和 $v_1\notin e_3$。根据连接树的连接性质，设 T_1 为由 v_1诱导的子树，T_2为由 v_2诱导的子树。由于 e_2同时属于 T_1和 T_2，因此所有满足 v_1或 v_2的节点在 T 上也是连通的。令 e_1为 n_1，令 e_3为 n_2；连接 n_1与 n_2的路径上必定存在一个节点 n'，满足$v_1,v_2\in n'$；否则，该路径将在 v_1和 v_2之间包含另一个节点 v_3，结合连接条件与 e_2便会形成一个环，这与 T 是一棵树的事实相矛盾。

另一方面，假设长度为 $k-1$ 的无弦路径满足该引理，那么对于长度为k 的无弦路径，我们可以采用与长度为 2 的情形类似的思路，证明 $v_1,\cdots,v_{k-1}$-诱导子树的连通性。因此，设 n_1和 n_k为两个端点，则 n_1和 n_k之间的路径必须覆盖 $v_1,\cdots,v_{k-1}$，否则树上就会出现一个环。通过归纳法，我们可以证明该引理。

由于不存在捷径，要覆盖 $v_1,\cdots,v_{k-1}$至少需要 $k-2$ 个节点，而 v_1和 v_{k-1}是两个端点，因此这样的路径长度至少为 k。□

引理 B.3. 给定一个合取查询 $Q=(\mathcal{V},\ \mathcal{E}\ \bar{y})$，设 $P\leftarrow\{v_1、\ \cdots、\ v_{k-1}\}$ 是Q的关系超图 $\mathcal{H}$上的一条长度为 k 的无弦路径。对于 Q的任意连接树 $\mathcal{T}$，$\mathcal{T}$的高度至少为 $\lceil\frac{k}{2}\rceil$。

引理 B.3 的证明。对于无弦路径 P，我们可以在任意连接树 T 上找到一条对应的路径 $P_{\mathcal{T}}$，其长度至少为 k。根据高度的定义，将两个端点设为 T 的叶节点，并要求该路径经过根节点，可以使树的高度最小。假设这两个叶节点到根节点的距离相等，则高度至少为 $\lceil\frac{k}{2}\rceil$。□

因此，对于高度为 k的布尔查询或全无环查询，该查询的所有连接树的高度都至少为 k。对于非全且非布尔的自由连通查询，我们可以进一步证明 q-无弦路径与连接树高度之间的联系。

引理 B.4. 给定一个自由连通的 $Q=(\mathcal{V}、\ \mathcal{E}\ \bar{y})$，设 $P\leftarrow\{v_1、\ \cdots、\ v_{k-1}\}$ 是 Q 的关系超图 $\mathcal{H}$ 上一条长度为 k 的 q-无弦路径。对于 Q 的任意自由连通连接树 $\mathcal{T}$，$\mathcal{T}$的高度至少为 k。

引理 B.4 的证明。Hu 和 Wang [23] 表明，对于一个高度为 2 的查询，若 $\bar{y}$ 非空或非满，则可能不存在长度为 3 或长度为 4 的路径，但存在一条长度为 2 的 q-无弦路径 $P = \{x_1\}$，其中 $x_1 \notin \bar{y}$、$R_1(\bar{x}_1)$，$R_2(\bar{x}_2)$ 是两个满足 $x_1 \in \bar{x}_1 \cap \bar{x}_2$ 的关系，且 $\exists x \in \bar{x}_1 \cap \bar{y}$。在这种情况下，$R_1$ 必须是任意自由连通连接树上 R_2 的父节点，否则就会违反连接树的连通性。另一方面，对于一个高度为 k 的查询 Q，若其残余查询 Q' 满足 $\mathrm{H}[Q']$ 包含这样一条长度为 $k-1$ 的 q-无弦路径，根据 q-无弦路径的定义，设 $R_{k-1}(\bar{x}_{k-1})$ 是残余查询 Q' 中仅包含 x_{k-2} 的关系。由于 R_{k-1} 不是 Q 中的耳，且 R_{k-1} 只包含非输出属性，因此 Q 中存在一个耳 $R_k(\bar{x}_k)$，其锚点为 R_{k-1}，并且 $\bar{x}_k \cap \bar{x}_{k-1} \cap P = \emptyset$ 和 $\bar{x}_k \cap \bar{x}_{k-1} \neq \emptyset$。因此，任取 R_{k-1} 与 R_k 之间的连接属性 $x_k \in \bar{x}_k \cap \bar{x}_{k-1}$，我们都能找到一条满足 $P' \leftarrow P \cup \{x_k\}$ 的长度为 k 的 q-无弦路径，从而完成证明。□

结合这两个引理，我们可以证明，高度为 k 的查询不可能存在高度为 $k-1$ 的自由连通连接树。

C 引理 4.3 的证明

对于一个 q-层次查询 Q，我们可以发现，不存在满足 R_1，$R_2 \in \mathrm{E}_c$ 的 V_c；否则该查询就不是 q-层次化的。

设 $Q = (\mathrm{V}, \mathrm{E}\ \bar{y})$ 是一个合取查询。如果任意 $\mathrm{V}_c \in \mathrm{V}$ 的维度为 $\dim(\mathrm{V}_c) > 1$，考虑使 V_c 的维度最大的 N 和 $\mathrm{V}_k(\mathrm{N})$。如果 (1) $\mathrm{V}_c \cap \bar{y} \neq \emptyset$ 或 $\mathrm{V}_k(\mathrm{N}) \subseteq \bar{y}$，则存在两个 R'，$R'' \in \mathrm{N}$，使得 $x_1 \in \bar{x}'$ 且 $x_2 \in \bar{x}''$。如果存在满足 x_1，$x_2 \in \bar{x}$ 的 $R \in \mathrm{E}_c$，那么 $\mathrm{E}[x_1] \cap \mathrm{E}[x_2] \neq \emptyset$，而 $\mathrm{E}[x_1] \nsubseteq \mathrm{E}[x_2]$ 且 $\mathrm{E}[x_2] \nsubseteq \mathrm{E}[x_1]$，这会使该查询不再是 q-层次化的；否则，设满足 $x_1 \in \bar{x}_1$ 且 $x_2 \in \bar{x}_2$ 的 R_1，$R_2 \in \mathrm{E}_c$，则必有 $x_3 \in \mathrm{V}_c \cap \bar{x}_1 \cap \bar{x}_2$，而 x_1/x_2 和 x_3 将违反 q-层次化条件。

另一方面，如果 (2) V_c 都是非输出属性，且 $\mathrm{V}_k(\mathrm{N}) \backslash \bar{y} \neq \emptyset$，则可能存在一个唯一的输出属性 $x \in \mathrm{V}_\star$ 会对维度作出贡献。在这种情况下，如果 E_c 只包含一个关系，我们可以找到一个非输出属性 $x' \in \mathrm{V}_k(\mathrm{N})$，由此可知 $\mathrm{E}[x] \subsetneq \mathrm{E}[x']$，且 $x \in \bar{y}$，但 $x' \notin \bar{y}$，因此该查询不是 q-层次化的；否则，如果 E_c 包含多个关系，我们可以找到一个非输出属性 $x' \in \mathrm{V}_c$，且 $\mathrm{E}[x] \subsetneq \mathrm{E}[x']$，且 $x \in \bar{y}$，但 $x' \notin \bar{y}$，因此该查询不是 q-层次化的。

对于仅当方向，由于查询仅包含维度为 1 的 V_c，而这对 $\mathrm{V}_c = \emptyset$ 和所有 $\mathrm{E}_c = \{R\}$ 都成立，因此我们可以证明：如果对于所有 x_1、$x_2 \in \mathrm{V}$，$\mathrm{E}[x_1] \cap \mathrm{E}[x_2] = \emptyset$ 始终成立，那么查询中的所有关系都不连通，且该查询是 q-层次化的。如果 $\mathrm{E}[x_1] \cap \mathrm{E}[x_2] \neq \emptyset$，那么 $\mathrm{E}[x_1] \subseteq \mathrm{E}[x_2]$ 或 $\mathrm{E}[x_2] \subseteq \mathrm{E}[x_1]$；否则，设 $R \in \mathrm{E}[x_1] \cap \mathrm{E}[x_2]$，我们可以证明 $\mathrm{E}_c = \{R\}$ 的维度 $\dim(\{\emptyset\}) \geqslant 2$，其中有一个包含 $R' \in$ 的邻居集 $\mathrm{N} \leftarrow \{R'、R''\}$，满足 $\mathrm{E}[x_1] \backslash \mathrm{E}[x_2]$，且 $\mathrm{E}\ R'' \in \mathrm{E}[x_2] \backslash \mathrm{E}[x_1]$，并令 x_1 和 x_2 为它们不同的键。此外，如果存在 $\mathrm{E}[x_1] \subsetneq \mathrm{E}[x_2]$，且 $x_1 \in \bar{y}$，那么它必定有 $x_2 \in \bar{y}$；否则，对于任意同时具有 x_1、$x_2 \in \bar{x}$ 的 R，当选取 $\{R'\}$ 作为 $\mathrm{V}_c = \{\emptyset\}$ 和 $\mathrm{E}_c = \{R\}$ 的不同邻居集 N，且满足 $x_2 \in \bar{x}'$ 时，由于 x_2，$\mathrm{V}_k(\mathrm{N}) \nsubseteq \bar{y}$，而 x_1 是一个活跃输出属性，因此 $\mathrm{V}_c = \{\emptyset\}$ 和 $\mathrm{E}_c = \{R\}$ 的维度至少为 2。因此，如果 Q 的维度为 1，那么该查询就是 q-层次化的。

D 补充运行示例

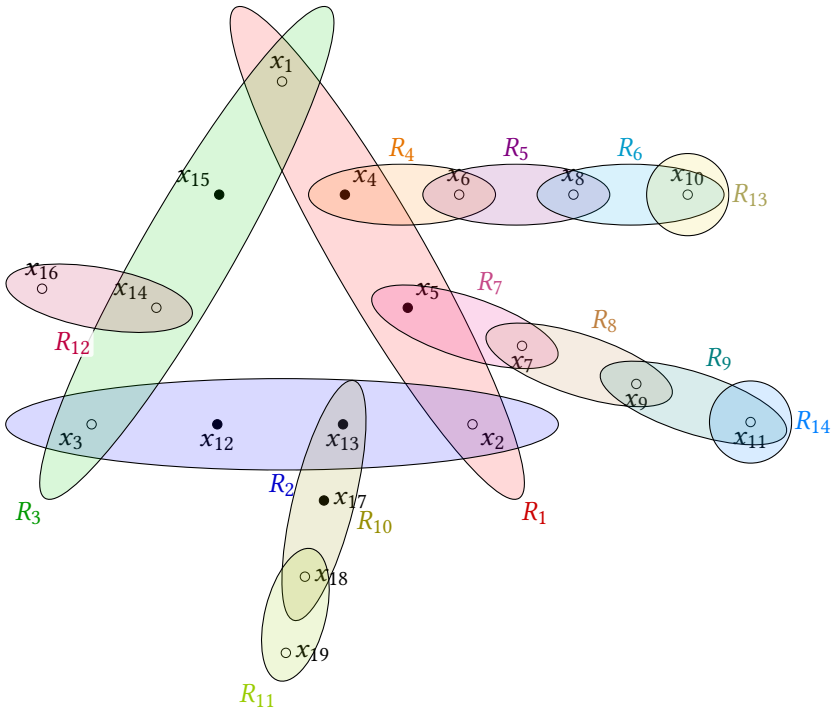


图 3. Q 的超图，其中实心点表示输出属性。

示例 *D.1.* 考虑以下查询：

$$Q(x_4, x_5, x_{12}, x_{13}, x_{15}, x_{17}) \leftarrow R_1(x_1, x_4, x_5, x_2), R_2(x_3, x_{12}, x_{13}, x_2), R_3(x_1, x_{15}, x_{14}, x_3),$$
$$R_4(x_4, x_6), R_5(x_6, x_8), R_6(x_8, x_{10}), R_7(x_5, x_7), R_8(x_7, x_9), R_9(x_9, x_{11}),$$
$$R_{10}(x_{13}, x_{17}, x_{18}), R_{11}(x_{18}, x_{19}), R_{12}(x_{14}, x_{16}), R_{13}(x_{10}), R_{14}(x_{11})$$

图 3 展示了其关系超图。该查询是一个高度为 5 的查询，其中包含长度为 9 的无弦路径 $x_{11}, x_9, x_7, x_5, x_4, x_6, x_8, x_{10}$，或长度为 5 的 q-无弦路径 x_4, x_6, x_8, x_{10}。然而，由于 x_4 和 x_5 之间存在捷径，$x_{11}, x_9, x_7, x_5, x_2, x_3, x_1, x_4, x_6, x_8, x_{10}$ 并不是无弦路径。

考虑维度时，该查询是一个维数为 6 的查询，其中 $\mathrm{V}_c = \{x_1, x_2, x_3\}$、$\mathrm{E}_c = \{R_1, R_2, R_3\}$、$\mathrm{N} = \{R_{12}, R_{10}, R_4, R_7\}$、$\mathrm{V}_k(N) = \{x_4, x_5, x_{13}, x_{14}\}$ 和 $\mathrm{V}_\star = \{x_4, x_5, x_{12}, x_{13}, x_{14}, x_{15}\}$。$\mathrm{V}_\star$ 中的所有顶点都对维度有贡献，因为 $\mathrm{V}_\star \backslash \mathrm{V}_k(N) = \{x_{12}, x_{15}\} \subseteq \bar{y}$ 和 $\mathrm{V}_c \cap \bar{y} = \emptyset$。通过设置 $\mathrm{V}_c = \{x_1, x_2, x_3, x_{13}\}$、$\mathrm{E}_c = \{R_1, R_2, R_3, R_{10}\}$ 和 $\mathrm{V}_\star = \{x_4, x_5, x_{12}, x_{14}, x_{15}, x_{17}, x_{18}\}$，无法得到更大的维度。尽管新的 $\mathrm{V}_\star$ 包含更多属性，但由于 $x_{13} \in \bar{y}$，我们只能选择 $\mathrm{V}_k(\mathrm{N}) = \{x_4, x_5, x_{14}, x_{18}\}$，其中 $\mathrm{N} = \{R_4, R_7, R_{12}, R_{11}\}$，使其计入维度，并且在这种情况下 $\dim(\mathrm{V}_c) = 4$。

考虑静态数据库时，可以通过将 $O(|D|^{1.5} + |Q(D)|)$ 替换为$\{R_1、R_2、R_3\}$和一个包 $B_1(x_1、x_2、x_3、x_4、x_5、x_{12}、x_{13}、x_{14}、x_{15}) \leftarrow R_1、R_2、R_3$，从而将查询转化为自由连通查询。该查询的运行时间表明，其宽度为 $\omega(Q) \leqslant 1.5$，由该宽度得到的动态求值下界为 $\Omega(|D|^{\omega(Q)-1}) = \Omega(|D|^{0.5})$，而这一界被查询维度给出的下界所支配。

例D.2（定理1.2的运行示例）。考虑例2.7中的 Q_1，现在我们用$\mathrm{E}_c = \{R_3$替换$\mathrm{V}_c = \{x_5\}$，以模拟一个OuMv_4问题。我们选择$\mathrm{N} \leftarrow \{R_2, R_4, R_{10}, R_8\}$，其连接键 x_3、x_4、x_9、x_6 互不相同（示意图见图 2c）。给定一个OuMv_4 实例，令$M(a_1, a_2, a_3, a_4, l)$ 为张量的稀疏形式，其中 l是元组标识符。我们按如下方式创建更新序列：**(1)** 将一个特殊元组 $(\perp)$ 插入 R_6，并将 $(\perp, \perp)$ 插入 R_1 和 R_5；**(2)** 将$\pi_{a_1, a_2, l, a_3} M$中的所有元组插入 R_3，其中 $\{a_1 \to x_3, a_2 \to x_4, l \to x_5, a_3 \to x_6\}$；**(3)** 将$\pi_{a_2, l, a_4} M$中的所有元组插入 R_7，其中 $\{a_2 \to x_4, l \to x_5, a_4 \to x_9\}$；**(4)** 将$\pi_{a_1, a_3} M$中的所有元组插入 R_9，其中 $\{a_1 \to x_3, a_3 \to x_6\}$。完成步骤(4)后，我们初始化张量，并开始处理每个向量上的更新：**(5)** 对于 u_3 u_4上的每次更新 (t)，我们分别在 R_8、R_{10}上执行相同的更新；**(6)** 对于 u_2上的每次更新 (t)，我们在 R_4上执行更新 $(t, \perp)$；**(7)** 对于 u_1上的每次更新 (t)，我们在 R_2上执行更新 $(\perp, t)$；**(8)** 更新完向量的第 i轮后，我们枚举 Q_1。如果存在任何查询结果，则 M 与 $u_1^{(i)} \times u_2^{(i)} \times u_3^{(i)} \times u_4^{(i)}$ 的交集非空。

示例 *D.3*（维护星形查询）。考虑维度为4 的星形查询：

$$Q(x_2, x_3, x_4) \leftarrow R_1(x_1, x_2, x_3, x_4), R_2(x_1), R_3(x_2), R_4(x_3).$$

注意，x_4 是 R_1 中的一个输出属性，它不与任何卫星连接；但属性$x_1 \in \mathrm{V}_k(\mathrm{N})$是一个非输出属性；因此，我们可以将 x_4纳入维度的考量。

数据库实例（R_1）。设当前数据库实例 D 的广义 H 指数为 $h = 2$。将键分类为**重**的阈值为 $\tau = (h+1)^{d-1} = 3^3 = 27$。

我们构造 R_1，其中约有 80 个元组，分布如下：

• **属性** x_1 **分布：** 值 A 出现在 **30 个元组** 中（例如，元组 $t_1 \ldots t_{30}$）；值 B 出现在 **30 个元组** 中（例如，元组 $t_{31} \ldots t_{60}$）；值 C 出现在 **10 个元组** 中（例如，元组 $t_{61} \ldots t_{70}$）；值 D-M 各出现在 **1 个元组** 中。有 2 个值（A、B）的频数为 $\geqslant 27$。因此，x_1 支持$h = 2$。由于频数为 $3^3 = 27$ 的值不足 3 个，因此未达到 $h = 3$。

• **属性** x_2 **分布：** 值 X 出现在 **40 个元组** 中（与 x_1的 A 和 B 元组重叠）；值 Y：出现在 **35 个元组**中；其他值各出现 $\leqslant 5$ 次。与 x_1类似，值 X 和 Y 均超过了 27 这一重值阈值。

• **属性** x_3，x_4 **分布：** 所有值均唯一（频数为 1）。因此，没有值的频数超过 27，它们均为轻值。

分区 R_1。给定阈值 $\tau = 27$，算法将 R_1 逻辑上划分为多个桶$R_1^{(b_1 b_2 b_3 b_4)}$，其重键/轻键如下：

属性	**重键 轻键**	
x_1	{A, B}	{C, D ...}
x_2	{X, Y}	{...}
x_3	$\varnothing$	所有值
x_4	$\varnothing$	所有值

• 分区 $(1, 1, 0, 0)$ **[**重 x_1, 重 x_2**]**：此分区存储数据中稠密键的笛卡尔积。其中包含诸如 (A, X ...), (A, Y ...), (B,X ...) 和 (B, Y ...) 这样的元组。• 分区 $(1, 0, 0, 0)$ **[**重 x_1, 轻 x_2**]**：包含满足 $x_1 = \mathrm{A,B}$ 但不满足 $x_2 \neq \mathrm{X,Y}$ 的元组。回想一下，x_2 键的频数为 $\leqslant 5$，但 X 和 Y 除外。• 分区 $(0, 1, 0, 0)$ **[**轻 x_1, 重 x_2**]**：包含满足 $x_1 \neq \mathrm{A, B}$ 但不满足 $x_2 = \mathrm{X,Y}$ 的元组。请记住，C 的频数为 10，而 D-M 中所有键的频数均为 1。• 分区 $(0, 0, 0, 0)$ **[**全轻**]**：包含所有剩余元组，例如 (D, Z ...) 这样的单例。• 其他分区：对于其他分区，如果 x_3 为重键或 x_4 为重键，则这些分区均为空，因为不存在这些键。

维护过程。**场景 A：更新轻键**。假设将一个元组 (C) 插入卫星 $R_2(x_1)$。算法会检查 C 是否 $\in K^H_{x_1}$。在本例中，答案是否定的。因此，算法必须扫描 R_1 的轻分区（其中 $b_1 = 0$），查找与 $x_1 = \mathrm{C}$ 匹配的元组。根据分布可知，C 恰好出现 10 次。由于 $10 < \tau = 27$，扫描成本严格受限于$O((h+1)^{\dim(Q)-1})$。

场景 B：对重键进行更新。假设将元组 (A) 插入卫星 $R_2(x_1)$。算法检查 $\mathrm{A} \in K^H_{x_1}$。在这种情况下，它是**重**的。因此，算法不会扫描 R_1 以查找 A（这需要 30 次操作；如果操作数更高，就可能违反严格的界限）。相反，它会更新重视图 V_C。该视图实际上跟踪重键的笛卡尔积：{A、B} × {X、Y}。它将 $x_1 = \mathrm{A}$ 标记为存活。代价与所有重键的笛卡尔积大小成正比，而该大小上限为 $h^{\dim(Q)-1} \approx 2^3 = 8$ 次操作。

场景 C：对 R_1进行更新。假设将元组 $(a_1、\ a_2、\ a_3、\ a_4)$ 插入中心 R_1。算法检查 $a_1 \in K^H_{x_1}$、$a_2 \in K^H_{x_2}$、$a_3 \in K^H_{x_3}$ 和 $a_4 \in K^H_{x_4}$，并得到向量**b**。之后，算法将 $(a_1、\ a_2、\ a_3、\ a_4)$ 插入 $R_1^{(\mathbf{b})}$；对于 $a_1、\ \cdots、\ a_3$，当 a_i 为轻值时，它会检查 $a_i \in R_i^{(0)}$；如果该元组通过所有这些检查，算法就只将该元组投影到重键，并将其传播到 V_C。此类更新的代价为常数。

场景 D：键重新分类。假设算法已从 R_1 中删除了带有$x_1 = \mathrm{A}$ 的 17 个元组，使键 A 变为轻，而 X 和 Y 的频数在这些删除操作之后仍高于 27。随后，算法从 R_1中删除剩余的 13 个带有 $x_1 = \mathrm{A}$ 的元组，再删除 R_2中所有带有 $x_1 = \mathrm{A}$ 的元组。在 R_1中，每个元组的删除耗时为 $O(1)$；由于 A 已从 R_1中完全移除，因此不会触发传播，R_2中的删除也耗时 $O(1)$。接着，算法将所有元组重新插入 R_2，并赋予轻标签；由于当前 R_1 中不含 $x_1 = \mathrm{A}$，这需要 $O(1)$ 的时间。之后，算法最终将所有元组重新插入 R_1，并赋予新标签，每个元组耗时 $O(1)$。重新平衡步骤的总成本为 $O(13)$，执行该步骤需要 17 次删除操作。将成本分摊到这些删除操作上，可得摊销成本为 $O(1)$。

场景 E：全局参数更新。假设在向 R_1 中插入了 53 个带有 $x_1 = \mathrm{D}$ 的元组和 44 个带有 $x_1 = \mathrm{C}$ 的元组后，C 和 D 都变成了重键，并且这两个键都已完成键重新分类。我们假设这些新元组包含唯一的 $x_2,\ x_3,\ x_4$ 值，因此不会改变其他键的重/轻分类。现在，x_1 的重键数量翻倍了。因此需要进行全局参数更新。完成后，$h_1(D)$ 变为 3，而 $h_2(D),\ h_3(D),\ h_4(D)$保持不变。因此，我们只需对 x_1 中的重键重新分类。x_1 中的所有键都是轻键，因为它们的频数都小于 $4^3 = 64$。算法将先删除所有重键，再将其重新作为轻键插入。重新平衡的总成本可能为 $O(h(D)^{\dim(Q)})$，但这是在经过 $h(D) \times h(D)^{\dim Q-1}$ 次更新之后发生的（h 个键被加入重键集合，同时每个键的频数从至多 $h(D)^{\dim(Q)-1}$ 变为至少 $2h(D)^{\dim(Q)-1}$）。因此，这类更新的摊销成本为 $O(1)$。

致谢

本研究由新加坡教育部学术研究基金第一级（RS32/25）及南洋理工大学创业资助支持。作者谨向 Ce Jin、Haozhe Zhang 和匿名审稿人致谢，感谢他们富有见地的讨论与评论。作者还要感谢易科教授一直以来的指导。

参考文献

[1] Amir Abboud、Arturs Backurs 和 Virginia Vassilevska Williams。2015年。如果当前团算法是最优的，那么Valiant 解析器也是最优的。载于 IEEE 第56届计算机科学基础年会（FOCS 2015），美国加利福尼亚州伯克利，*17 20 2015,* 98 117 10 1109 2015 16 - 10月，[2] Venkatesan Guruswami（编）。IEEE计算机学会，2024。–。doi: . /FOCS.。Amir Abboud、Nick Fischer 和 Yarin Shechter。 . 更快的组合 k 团算法。载于拉丁美洲理论信息学研讨会。Springer，193–206。[3] Amir Abboud 和 Virginia Vassilevska Williams。2014年。常见猜想蕴含动态问题的强下界。载于2014年IEEE第55届计算机科学基础年会。IEEE，434–443。[4] Serge Abiteboul、Richard Hull 和 Victor Vianu。1995年。数据库基础。Addison-Wesley Longman Publishing Co., Inc.[5] Mahmoud Abo Khamis、Ahmet Kara、Dan Olteanu 和 Dan Suciu。2024年。动态查询求值中的仅插入与插入删除。ACM数据管理会议录 2, 5, 219 2024 26 10 1145 3695837 文章（11月），页码。doi: . / Noga Alon、Raphael Yuster 和 Uri Zwick。 . 彩色编码。《ACM学报》（7月），–。[6] 1995 42 4 1995 844 856 doi:10.1145/210332. 210337[7] Noga Alon、Raphael Yuster 和 Uri Zwick。1997年。寻找和计数给定长度的环。Algorithmica 17，3（ 1997 209 223 10 1007 02523189 ），–。doi: . /BF Guillaume Bagan、Arnaud Durand 和 Etienne Grandjean。2007年。关于无环合取查询与常数延迟枚举。载于计算机科学逻辑。Springer Berlin Heidelberg，德国柏林海德堡，208–222。[8] [9] Christoph Berkholz、Jens Keppeler 和 Nicole Schweikardt。2017年。更新下的合取查询应答。载于第36届 ACM SIGMOD-SIGACT-SIGAI数据库系统原理研讨会论文集（美国伊利诺伊州芝加哥）（PODS ’*17* 303 318 10 1145 3034786 3034789 ）。计算机协会，美国纽约州纽约，–。[10] 2015 doi: . / . Paris Carbone、Asterios Katsifodimos、Stephan Ewen、Volker Markl、Seif Haridi 和 Kostas Tzoumas。 . Apache Flink：流处理与批处理的单一引擎。IEEE数据工程简报 38，4（2015），28–38。[11] Nofar Carmeli 和 Markus Kröll。2019年。关于合取查询的并的枚举复杂度。载于第38届 ACM SIGMOD-SIGACT-SIGAI数据库系统原理研讨会论文集。ACM，134–148。[12] Nofar Carmeli 和 Luc Segoufin。2023年。带自连接的合取查询：迈向细粒度枚举复杂性分析。载于第42届 ACM SIGMOD-SIGACT-SIGAI数据库系统原理研讨会论文集（美国华盛顿州西雅图）（PODS ’23）。计算机协会，美国纽约州纽约，277–289。doi:10. 1145/3584372.3588667[13] Rada Chirkova 和 Jun Yang。2012年。物化视图。《数据库基础与趋势》4，4（2012），295–405。[14] T. H. Cormen、C. E. Leiserson、R. L. Rivest 和 C. Stein。2009年。《算法导论》（第3版）。麻省理工学院出版社。[15] Søren Dahlgaard、Mathias Bæk Tejs Knudsen 和 Morten Stöckel。2017年。通过有界k游走更快地寻找偶环。载于第49届 ACM SIGACT计算理论年会论文集（加拿大蒙特利尔）（STOC 2017）。计算机协会，美国纽约州纽约，112 120 10 1145 3055399 3055459 –。doi: . / . Shiyuan Deng、Shangqi Lu 和 Yufei Tao。 . 关于连接采样及组合式输出敏感连接算法的难度。载于第42届 ACM SIGMOD-SIGACT-SIGAI数据库系统原理研讨会论文集（美国华盛顿州西雅图）（PODS ’23）。计算机协会，美国纽约州纽约，99–111。doi:10.1145/3584372.3588666[16] 2023 [17] David Eppstein、Michael T. Goodrich、Darren Strash 和 Lowell Trott。2012年。使用h指数参数化数据结构扩展动态图子图统计。《理论计算机科学》447（2012），44–52。doi:10.1016/j.tcs.2011.11.034 组合算法与应用（COCOA 2010）。[18] David Eppstein 和 Emma S. Spiro。2009年。图的h指数及其在动态图子图统计中的应用。载于《算法与数据结构》，Frank Dehne、Marina Gavrilova、Jörg-Rüdiger Sack 和 Csaba D. Tóth（编）。Springer Berlin Heidelberg，德国柏林海德堡，278–289。[19] Austen Z Fan、Paraschos Koutris 和 Hangdong Zhao。2023年。布尔合取查询与和积问题的细粒度复杂性。载于第50届自动机、语言与编程国际研讨会（ICALP 2023）。Schloss Dagstuhl–Leibniz 信息学中心，127–1。[20] Marc H. Graham。1979年。关于通用关系。技术报告。多伦多大学。[21] Robert Haas 和 Michael Hoffmann。2006年。经过三个顶点的无弦路径。《理论计算机科学》351，3（2006），360–371。doi:10.1016/j.tcs.2005.10.021 参数化与精确计算。[22] Xiao Hu 和 Qichen Wang。2023年。高效计算合取查询之差。ACM数据管理会议录 1, 2, 153 2023 26 10 1145 3589298 文章（6月），页码。doi: . / Xiao Hu 和 Qichen Wang。 . 面向查询维护的依赖更新分析。ACM数据管理会议录 3, 2, 117 2025 25 10 1145 3725254 文章（6月），页码。doi: . / Muhammad Idris、Martin Ugarte 和 Stijn Vansummeren。 . 动态Yannakakis算法：更新下紧凑且高效的查询处理。载于2017年 ACM数据管理国际会议论文集（美国伊利诺伊州芝加哥）（SIGMOD ’17）。计算机协会，美国纽约州纽约，1259–1274。[23] 2025 [24] 2017

doi:10.1145/3035918.3064027[25] Muhammad Idris、Martín Ugarte、Stijn Vansummeren、Hannes Voigt 和 Wolfgang Lehner。2018。更新条件下带不等式的合取查询。超大型数据库国际会议论文集 11, 7 (2018), 733–745. [26] Ce Jin 和 Yinzhan Xu。2022。广义 BMM 和 OMv 带来的紧动态问题下界。载于第*54*届 *ACM SIGACT*计算理论年会论文集（意大利罗马）*(STOC 2022)*。计算机协会，纽约，NY，美国，

1515 1528 10 1145 3519935 3520036 –。doi:。/。Ahmet Kara、[27] Hung Q. Ngo、Milos Nikolic、Dan Olteanu 和 Haozhe Zhang。 。更新条件下的三角形查询维护。*ACM*数据库系统汇刊 2020 45, 3, 11 2020 46 10 1145 3396375 文章（aug），页码。doi:。/。Ahmet Kara、Milos Nikolic、Dan Olteanu 和 Haozhe Zhang。[28] 2020 。层次查询静态和动态求值中的权衡。载于第*39*届 *ACM SIGMOD-SIGACT-SIGAI*数据库系统原理研讨会论文集。375–392。[29] Andrea Lincoln 和 Nikhil Vyas。2020。环和游走的算法与下界：小空间与稀疏图。载于第*11*届理论计算机科学创新会议（*ITCS 2020*）（莱布尼茨国际信息学会议录（*LIPIcs*），第*151*卷），Thomas Vidick（编）。Schloss Dagstuhl – Leibniz信息学中心，德国 Dagstuhl, 11 1 11 17 10 4230 2020 11 : – :。doi:。/LIPIcs.ITCS.。Andrea Lincoln 、[30] Virginia Vassilevska Williams 和 Ryan Williams。2018。稀疏图中最短环和路径的紧难度。载于第二十九届 *ACM-SIAM*离散算法研讨会论文集。工业与应用数学学会，1236–1252。[31] Stefan Mengel。2025。合取查询求值的下界。载于第*44*届数据库系统原理研讨会伴随会议录（德国柏林）*(PODS '25)*。计算机协会，纽约，NY，美国，5。doi:10.1145/3722234.3725824[32] Moni Naor、Leonard J. Schulman 和 Aravind Srinivasan。1995。分割器与近最优去随机化。载于第*36*届计算机科学基础年会，*FOCS 1995*，美国威斯康星州密尔沃基，*1995*年*10*月*23*日至*25*日。IEEE计算机学会, 182 191 10 1109 1995 492475 –。doi:。/SFCS.。Milos Nikolic 和 Dan Olteanu。 。[33] 2018 受益于三重锁因子分解的增量视图维护。载于 *ACM SIGMOD* 数据管理国际会议论文集。ACM, 365–380。[34] Dante Pinto 和 Cristian Riveros。2024。复杂事件识别与层次合取查询相遇。*CoRR* abs/2408 01652 2024 2408 01652 10 48550 2408 01652 。（）。arXiv:。doi:。/ARXIV.。Qichen Wang、Xiao Hu、Binyang Dai 和 Ke Yi。[35] 2023 。无连接的变更传播。*VLDB Endowment*会议录 16, 5（jan 2023 1046 1058 10 14778 3579075 3579080），–。doi:。/。Qichen Wang 和 Ke Yi。[36] 2020 。更新条件下无环外键连接的维护。载于*2020*年 *ACM SIGMOD*数据管理国际会议论文集。1225–1239。[37] Qichen Wang 和 Ke Yi。2025。带比较的合取查询。*ACM*数据库系统汇刊 （2025年9月）。doi:10.1145/3769424[38] C. T. Yu 和 M. Z. Özsoyoğlu。1979。分布式查询的树查询成员资格算法。载于*IEEE*计算机学会第三届国际计算机软件与应用会议，*COMPSAC 1979*，*1979*年*11*月*6*日至*8*日，美国伊利诺伊州芝加哥。IEEE，306–312。doi:10.1109/CMPSAC.1979.762509

收稿日期：2025年12月；录用日期：2025年2月