使用几何数据结构的更快关系算法*

ARYAN ESMAILPOUR，计算机科学系，伊利诺伊大学芝加哥分校，美国
STAVROS SINTOS，计算机科学系，伊利诺伊大学芝加哥分校，美国

关系数据上的优化任务（例如聚类）通常会受到连接操作高昂成本的制约，因为访问完整数据集需要执行这些操作。尽管 BBD 树等几何数据结构在标准计算环境中能够产生快速的近似算法，但由于连接结果的规模，其在关系数据上的应用仍不明确。本文提出了一个框架，利用几何学洞见，在数据以关系数据库中的连接查询结果形式存储时设计更快的算法。我们的核心贡献是开发了 RBBD 树，这是一种针对关系型场景定制的 BBD 树随机化变体。我们不再完全构建 RBBD 树，而是通过在关系连接上利用高效的采样和计数技术，实现 RBBD 树的动态高效扩展，仅维护必要的部分。这样，我们便能够在不具体化连接结果的情况下模拟几何查询过程。作为应用，我们提出的算法在保持相同近似保证的同时，将关系型 k-中心/均值/中位数聚类的最先进水平在运行时间方面提升了 k 倍。我们的方法具有通用性，可应用于关系型场景中的各种优化问题。

CCS 概念：• **计算理论 → 用于数据管理的数据结构和算法**。
附加关键词和短语：关系数据、聚类、k-中心、k-中位数、k-均值、BBD 树

arXiv:2603.11402v1 [cs.DB] 12 Mar 2026

1 引言

聚类等优化问题传统上在计算机科学领域进行研究时，通常假设可以完整访问输入数据。然而，在现实世界的应用中，数据通常存储在关系数据库中，其中的信息分布于多个相互关联的表中。每个表都包含一组元组，来自不同表的元组如果在共享属性上具有相同的值，就可以进行连接。这种关系结构带来了一个根本性挑战：只有在对相关表执行连接操作后，才能完整访问数据。这类连接的计算代价很高，因为连接的输出规模可能比输入表的规模大出多项式倍 [56, 57]。尽管如此，关系数据在实践中无处不在。根据 [1, 3, 61], ，现代数据库系统和数据科学任务中的绝大多数都涉及关系数据。

面向关系数据的机器学习应用通常包含两个阶段：（1）数据准备，即将多张表进行连接以构建特征数据集；（2）建模，即对所得数据应用机器学习算法（例如无监督学习中的聚类算法）。然而，这一两步流程在实践中可能代价过高，主要原因是计算大型连接的成本很高。

•考虑一种机器学习工作流：先对关系数据进行特征工程，再通过聚类进行无监督学习，该工作流基于 Yelp 评论数据集 [2]。该数据集已在先前工作 [22, 26] 中用于 k-均值聚类。该数据库由五个关系组成：(1) 评论(user_id, business_id, stars, date)，描述单条评论；(2) 用户(user_id, review_count, fans, yelping_since)，提供评论者属性；(3) 商家(business_id, city, state, avg_rating)，描述商家元数据；(4) 类别(business_id, category)，列出分配给每个商家的类别（例如，“餐厅”、“咖啡店”）；以及 (5) 属性(business_id, attribute)，记录

*本工作部分得到美国国家科学基金会资助项目 IIS-2348919 的支持。

作者联系信息：Aryan Esmailpour，伊利诺伊大学芝加哥分校计算机科学系，芝加哥，美国，aesmai2@uic.edu；Stavros Sintos，伊利诺伊大学芝加哥分校计算机科学系，芝加哥，美国，stavros@uic.edu。

其他商家特征（例如，“营业至深夜”、“提供无线网络”）。一个典型的特征工程查询按如下方式连接这些关系： Q = 评论 $\bowtie$ 用户 $\bowtie$ 商家 $\bowtie$ 类别 $\bowtie$ 属性。 Q 中的每个元组对应于一条经过丰富的单条评论，其中同时包含数值属性（例如，review_count、avg_rating）和类别属性（例如，category、attribute），这些属性在分析之前会进行独热编码。随后直接在这些连接元组上执行聚类，这意味着各条评论事件会根据其联合的用户属性、商家特征和语义标签进行分组。在这种表示中，如果两条评论由活动概况相似的用户撰写，且针对评分、类别和属性相似的商家，则认为它们相似。因此，每个簇都捕捉到评论者行为与商家语义的一种连贯模式，例如高度活跃的用户评论带有无线网络的深夜餐厅，或不常评论的用户评论面向家庭的餐饮场所。至关重要的是，由于商家通常具有多个类别和属性标签，连接结果 Q 远大于任何基关系。如 [26], 所报告，基础 Yelp 表总共包含大约800万个元组，而完全物化连接包含约2200万个元组。这种显著的膨胀凸显了在学习之前物化关系连接的计算成本，并促使我们需要能够直接在底层表上运行、而无需显式构造 Q的算法。

- 考虑一个名为 Purchase 的表，其中包含 price、T、item 和 category 属性。price 是实数值属性，T 用于标识交易，item 是商品标识符，category 是分类属性。目标是分析经常被共同购买的商品的定价模式。例如，对自连接 $\text{Purchase}(\text{T}, \text{price}_0, \text{item}_0, \text{category}_0) \bowtie \text{Purchase}(\text{T}, \text{price}_1, \text{item}_1, \text{category}_1)$ 的结果进行 k均值聚类，可以根据共同购买商品对的联合价格特征对其进行分组。每个连接元组表示在同一笔交易中购买的一对商品，这些商品对嵌入二维价格空间 $(\text{price}_0, \text{price}_1)$ 中，还可以加入分类信息（例如对应商品的类别）。由于一次购买可能包含多个不同商品，每笔交易都会在自连接中生成许多不同的商品对，这可能导致连接结果的规模大幅膨胀（相对于 Purchase 表的大小呈二次增长）。即使随后仅将连接结果投影到价格属性上，保留重复元组也至关重要：同一价格对的重复出现，意味着相应商品对经常被共同购买，因此在聚类目标中具有更大的统计权重。在这个空间中进行聚类，可以揭示典型的基于价格的共同购买模式，例如低价—低价、低价—高价或高价—高价商品对。此类对常购商品的分析是购物篮分析和零售数据挖掘中的常见做法 [9, 40, 50]。

近期的研究旨在通过设计关系算法 来避免物化完整的连接输出，以解决聚类等优化任务 [22, 26, 54]。然而，这些方法往往存在近似比很大的问题，或运行时间开销高昂。此外，每种已知方法通常都依赖于针对特定问题量身定制的不同专用技术。

本文提出了一种通用框架，可为关系数据上的广泛一类优化问题提供更快的算法。给定一个连接查询和一个数据库实例，我们的方法可用于高效解决聚类、多样性最大化和公平性感知选择等问题。在标准计算环境中，这些问题中的许多都已有成熟的近线性时间近似算法；在该环境下，输入是一张扁平表。然而，由于连接输出的规模，直接将这些技术应用于关系型场景效率低下。我们的核心思路是在不将整个连接结果具体化的情况下，模拟几何数据结构的行为，尤其是基于层次树结构的数据结构，例如*BBD* 树 [13], 而不将整个连接结果具体化。我们通过实现 BBD 树的关系型版本来介绍这一方法，因为它最适合我们要解决的聚类问题。不过，同样的技术也可用于构造其他常用几何数据结构的关系型版本，例如四叉树 [35], kd 树 [27], 或划分树 [21]。我们提出了一种新的随机化数据结构，即

RBBD 树，它保留了 BBD 树的理想性质，同时针对关系数据进行了定制。关键的是，我们证明，只需给定 RBBD 树的一个节点（该树定义在某个无环连接查询的连接结果上），便可在大约 $O(N)$ 时间内计算出其子节点，其中 N是输入数据库的大小。这使得我们能够高效地模拟针对连接结果的几何查询，包括范围计数、范围报告和范围采样查询，因为这类查询通常只需访问对数数量的树节点。

我们的方法具有通用性：任何通过对 BBD 树进行 σ次查询来与输入数据交互的几何算法，都可以转化为针对无环连接查询的随机化关系算法，其总运行时间为 $O(\sigma \cdot N)$（忽略对数因子）。许多几何算法都属于这一类，因此我们的框架为各种任务提供了高效的关系对应算法。作为一项应用，我们证明，所有关系型k-中心、 k-中位数和 k-均值聚类的最先进算法，其运行时间都可以提升 k 倍。

尽管在许多应用中， k被认为很小，但面向较大 k的 k-聚类在现代应用中变得越来越重要，例如向量数据库中的最近邻搜索所用的乘积量化 [45]。这推动了人们在各种计算模型 [16, 25, 30, 32, 34]下，对大规模k-聚类展开广泛的算法研究。此外，在关系型场景中， k 甚至可能大于 N，因为连接结果的数量可能比 N大几个数量级。在关系聚类 [7, 26, 31, 54], 的相关文献中，现有算法通常假设 k 可能很大，后续研究则着重于改善对 k的依赖。对于无环查询，我们提出了首批对 k 和 N均呈线性依赖的关系聚类算法。

1.1 符号与问题定义

连接查询。给定一个数据库模式 R，其定义在属性集合 $\mathbf{A} = \{A_1, \ldots, A_d\}$ 上，其中包含 m 个关系 $R_1, \ldots, R_m$。令 $\mathbf{A}_j \subseteq \mathbf{A}$ 为与关系 $R_j \in \mathbf{R}$ 相关联的属性集合。对于属性 $A \in \mathbf{A}$，令 $\mathrm{dom}(A)$ 表示属性 A 的域。令 $\mathrm{dom}(X) = \prod_{A \in X} \mathrm{dom}(A)$ 表示属性集合 $X \subseteq \mathbf{A}$ 的域。设 D 是数据库模式 R 上的一个数据库实例。为简单起见，我们假设 D 中每个关系 $R_j \in \mathbf{R}$ 都包含N 个元组。全文中，我们考虑数据复杂度，即 m和 d是常数，而N是一个较大的整数。我们用 R_j同时表示关系本身以及 D 中存储在该关系里的元组集合。对于属性子集 $B \subseteq \mathbf{A}$ 和集合 $Y \subset \mathbb{R}^d$，令 $\pi_B(Y)$ 表示由 Y 中元组在属性 B 上的投影所构成的集合。注意， Y 中的两个不同元组可能在 B 上有相同的投影；然而， $\pi_B(\cdot)$定义为一个集合，因此投影后的元组只存储一次。

继关系聚类的相关工作之后， [22, 26, 54], 给定一个连接查询$\boldsymbol{q} := R_1 \bowtie \ldots \bowtie R_m$。数据库实例 D 上的连接查询 $\boldsymbol{q}$的结果集合定义为 $\boldsymbol{q}(\mathbf{D}) = \{t \in \mathrm{dom}(\mathbf{A}) \mid \forall j \in [1, m] : \pi_{\mathbf{A}_j}(t) \in R_j\}$。如果存在一棵称为连接树的树，使得树的节点是$\mathbf{R}$ 中的关系，并且对于每个属性$A \in \mathbf{A}$，包含该属性的节点/关系集合构成一个连通分量，则称连接查询 $\boldsymbol{q}$是无环的。我们给出的所有算法都假设 $\boldsymbol{q}$ 是无环连接查询，但最后会将结果推广到任意一般连接查询。令 $\rho^\star(\boldsymbol{q})$ 为查询 $\boldsymbol{q}$ 的分数边覆盖，这是一个用于界定任意数据库实例上的连接结果数量的参数：对于每个具有 $O(N)$ 个元组的数据库实例$\mathbf{D}'$，都有 $|\boldsymbol{q}(\mathbf{D}')| = O(N^{\rho^\star(\boldsymbol{q})}) = O(N^m)$，如 [15] 所示。

我们使用记号 $\mathrm{fhw}(\boldsymbol{q})$ 表示查询 $\boldsymbol{q}$ 的分数超树宽度 [38] 。分数超树宽度大致衡量了 $\boldsymbol{q}$ 接近无环的程度。对于每个无环连接查询 $\boldsymbol{q}$，我们有 $\mathrm{fhw}(\boldsymbol{q}) = 1$。给定一个有环连接查询 $\boldsymbol{q}$，我们将其转换为一个等价的无环查询，使得每个关系都是一个（可能有环的）连接查询的结果，且其分数边覆盖至多为 $\mathrm{fhw}(\boldsymbol{q})$。因此，对于每个关系包含 $O(N)$ 个元组的数据库实例上的有环连接查询，可以在 $O(N_{\mathrm{fhw}(\boldsymbol{q})})$ 时间内，将其转换为一个等价的无环连接查询，该查询运行在一个数据库实例上

其中每个关系包含 $O(N_{\text{fhw}(q)})$ 个元组 [15]。附录 F 给出了 fhw 的更正式定义。如果根据上下文可以明确 q，我们就用 fhw 代替 fhw(q)。

对于元组 $p,\ q \in \text{dom(A)}$，令 $\phi(p,\ q) = ||p-q||_2 = \left(\sum_{j=1}^{d}(\pi_{A_j}(p) - \pi_{A_j}(q))^2\right)^{1/2}$ 表示它们的欧几里得距离。如果 $A_j \in \mathbf{A}$ 是分类属性，我们应用独热编码 （见附录 D），并定义 $(\pi_{A_j}(p) - \pi_{A_j}(q))^2 = 1$ 当 $\pi_{A_j}(p) \neq \pi_{A_j}(q)$ 时成立，否则定义为 $(\pi_{A_j}(p) - \pi_{A_j}(q))^2 = 0$。为清晰起见并与先前工作保持一致，我们聚焦于欧几里得距离，但我们的算法可扩展到任意 ℓ_p 度量。

在全文中，当我们考虑输入数据存储在单个扁平表中时，使用术语**标准计算环境**。相较之下，当数据是上述连接查询的结果时，我们使用术语**关系型场景**。

聚类。 本文聚焦于三种经典的聚类目标：k-中心、k-中位数和 k-均值聚类。这些形式刻画了衡量聚类质量的不同方式，是理论和实践中应用最广泛的模型之一 [14, 37, 42, 51, 58]。k-中心旨在最小化最坏情况下的距离，k-中位数旨在最小化总距离（平均距离），而 k-均值旨在最小化总平方距离。在需要确保每个点到最近中心的距离都不会太远的应用场景中（例如设施选址或覆盖问题），通常优先采用k-中心。当目标是最小化平均出行成本或总分配距离时（例如按邻近程度对客户进行聚类），k-中位数更为适合。最后，当距离代表误差或方差，且目标是最小化整体离散程度时（例如统计数据分析或机器学习应用），通常采用 k-均值。

我们先介绍一些有用的一般定义。设 P 为 dom(A) 中的一个元组集合，并设 $C \subset \text{dom(A)}$ 为一个 k元组集合。对于一个元组 $p \in \text{dom(A)}$，令 $\phi(p, C) = \min_{c \in C} \phi(p, c)$。我们定义

$$\mathbf{u}_C(P) = \max_{p \in P} \phi(p, C), \quad \mathbf{v}_C(P) = \sum_{p \in P} \phi(p, C), \quad \text{and} \quad \mu_C(P) = \sum_{p \in P} \phi^2(p, C).$$

k-中心聚类：给定一个元组集合 P，其中元素属于 dom(A)，以及参数 k，目标是找到一个集合$C \subseteq P$，满足 $|C| = k$，使得 $\mathbf{u}_C(P)$ 最小化。令 $\text{OPT}(P) = \arg\min_{S \subseteq P, |S|=k} \mathbf{u}_S(P)$ 为一个由k个中心组成的集合，使 $\mathbf{u}^{\text{OPT}(P)}(P)$ 最小。令 kCenterAlg$_\gamma$为一个（已知的） γ-近似算法，用于标准计算环境中的 k-中心聚类问题，其运行时间为 $T_{\text{cen}_\gamma}(|P|)$，其中 γ 为一个常数。

k-中位数聚类：给定一个元组集合 P，其中元素属于 dom(A)，以及参数 k，目标是找到一个集合 $C \subseteq P$，满足 $|C| = k$，使得 $\mathbf{v}_C(P)$ 最小化。令 $\text{OPT}(P) = \arg\min_{S \subseteq P, |S|=k} \mathbf{v}_S(P)$ 为一个由 k个中心组成的集合，使 $\mathbf{v}^{\text{OPT}(P)}(P)$ 最小。令 kMedianAlg$_\gamma$为一个（已知的） γ-近似
在标准计算环境下，求解 k-中位数问题的算法运行时间为 $T_{\text{med}_\gamma}(|P|)$，其中 γ 是一个常数。

k-均值聚类：给定一组元组 P，它们属于 dom(A)，以及一个参数 k，目标是找到一个集合$C \subseteq P$，满足 $|C| = k$，使得 $\mu_C(P)$ 最小化。令 $\text{OPT}(P) = \arg\min_{S \subseteq P, |S|=k} \mu_S(P)$ 为一组k中心，其 $\mu_{\text{OPT}(P)}(P)$ 最小。设 kMeansAlg$_\gamma$是一个已知的 γ-近似算法，用于求解标准计算环境下的 k-均值问题，运行时间为 $T_{\text{mean}_\gamma}(|P|)$，其中$\gamma$ 是一个常数。

对于 k-中心/中位数/均值聚类的最优解，我们使用相同的记号 OPT(·)。根据上下文，总能明确我们指的是 k-中心、k-中位数还是 k-均值聚类。有时，我们将$\mathbf{u}_C(P)$（或$\mathbf{v}_C(P)$ $\mu_C(P)$）的值称为 C在 P中的聚类代价。

相对近似。 如果对于元组集合 $P \subset \text{dom(A)}$ 上的 k-中心（分别为 k-中位数、k-均值）聚类，集合 $C \subset \text{dom(A)}$ 满足 $\mathbf{u}_C(P) \leq \alpha \cdot \mathbf{u}^{\text{OPT}(P)}(P)$（分别为

$\mathbf{v}_C(P) \leq \alpha \cdot \mathbf{v}_{\mathrm{OPT}(P)}(P)$, $\mu_C(P) \leq \alpha \cdot \mu_{\mathrm{OPT}(P)}(P)$），则称其为 -近似。$P$ 上的 α-近似算法始终返回一个中心集合，该集合是 P 上的 α-近似。

核心集。集合 $\mathbf{C} \subseteq \mathrm{dom}(\mathbf{A})$（满足 $|\mathbf{C}| \ll |P|$）是 P 上 k-中心/中位数/均值聚类的核心集，如果在 C 上执行的每个 α-近似算法都返回一个中心集合，且该集合是 P 的 $(1+\varepsilon)\alpha$-近似。在全文中，我们假设 $\varepsilon \in (0, 1)$。

在本文中，我们研究连接查询结果上的 k-中心、k-中位数和 k-均值聚类。定义 *1.1.* [关系型 k-中心/中位数/均值聚类] 给定数据库实例 $\mathbf{D}$、连接查询 $\boldsymbol{q}$ 以及正整数参数 k，目标是计算一个大小为$|\mathbf{S}| \leq k$ 的集合 $\mathbf{S} \subseteq \boldsymbol{q}(\mathbf{D})$，使得 $\mathbf{u}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D}))$（相应地，$\mathbf{v}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D}))$、$\mu_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D}))$）最小化。

备注。 虽然我们关注完整连接结果上的 k-聚类，但我们的所有算法都可以直接扩展到投影-连接查询结果上的 k-聚类，并采用包语义，即保留重复项（另见附录 D）。这种设置在某些应用中特别有用：目标是仅根据一部分属性对连接结果中的所有元组进行聚类（例如，在前文的 Purchase 表中仅使用价格属性）。如附录 D 所述，在包语义下对投影-连接查询进行聚类，也是处理同时涉及数值属性和分类属性的连接聚类问题的关键要素。

1.2 相关工作

对于仅包含数值属性的关系型 k-中心问题，Agarwal 等人 [7] 提出了一种在 $\tilde{O}(\min\{k^2 N_{\mathrm{fhw}} k N_{\mathrm{fhw}} + k^{d/2}\})$ 时间内运行的 $(2+\varepsilon)$-近似算法。

对于关系型 k-均值问题，Khamis 等人 [46], 给出了 Lloyd k-均值启发式算法的一种高效实现。已知的算法还包括相对近似算法。Curtin 等人 [26] 构造了一个加权元组集（网格核心集），使得针对网格核心集上的加权k-均值聚类的近似算法，能够返回关系型 k-均值聚类的近似解。如果所有属性均为数值属性，则其算法的运行时间为 $\tilde{O}(k^m N^{\mathrm{fhw}} + T_{\mathrm{mean}^\gamma}(k^m))$，近似因子为 $(\gamma^2 + 4\gamma\sqrt{\gamma} + 4\gamma)$。在同时包含数值属性和分类属性的一般情况下，运行时间为 $\tilde{O}(k^d N^{\mathrm{fhw}} + T_{\mathrm{mean}^\gamma}(k^d))$。所有后续工作都只关注所有属性均为数值属性的情形。Moseley 等人 [54] 设计了一种关系型 k-均值++ 算法 [8, 11]。对于常数 $\varepsilon \in (0, 1)$，其算法的期望运行时间为 $\tilde{O}(k^4 N^{\mathrm{fhw}} + k^2 N^{\mathrm{fhw}} + T_{\mathrm{mean}^\gamma}(k))$，近似因子为 $(320 + 644(1+\varepsilon)\gamma)$。最后，Esmailpour 和 Sintos [31] 采用层次聚合方法 [22]，推导出一种在 $\tilde{O}(k^2 N^{\mathrm{fhw}})$ 时间内运行的 $O(1)$-近似随机化算法。随后，他们在 $\tilde{O}(k^2 N^{\mathrm{fhw}} + k^4)$ 时间内构造几何核心集，从而得到一种以高概率在 $\tilde{O}(k^2 N^{\mathrm{fhw}} + k^4 + T_{\mathrm{mean}^\gamma}(k^2))$ 时间内运行的 $(4+\varepsilon)\gamma$-近似算法。文献 [31] 中的算法可扩展至关系型k-中位数聚类，从而得到一种在 $\tilde{O}(k^2 N^{\mathrm{fhw}} + k^4 + T_{\mathrm{med}^\gamma}(k^2))$ 时间内运行的 $(2+\varepsilon)\gamma$-近似算法。关系聚类问题也已有加性近似算法 [22]。最近，Surianarayanan 等人 [65] 研究了带离群点的关系型 k-中心聚类问题。

最后，近期有大量研究探讨用于学习问题的关系算法，例如线性回归和因子分解 [47, 48, 60, 64], 支持向量机 [4, 5, 70],独立高斯混合模型 [23, 24]。数据库研究探讨了定义在关系数据上的各种组合问题，例如排序枚举 [28, 29, 67], 分位数 [68], 直接访问 [20], 多样性 [7,10, 53], 以及 top-k [67]。Schleich 等人 [63]对此进行了综述。

1.3 我们的结果

除非另有说明，我们假设对于每个$\mathbf{A}$，$\mathrm{dom}(A) = \mathbb{R}$（或 $\mathrm{dom}(A) = \mathbb{Z}$），并且查询$\boldsymbol{q}$ 是无环的。在附录 D 中，我们展示了当属性集$\mathbf{A}$ 同时包含数值域和分类域时，如何扩展我们的算法；而在第7节中，我们将结果扩展到循环查询。

问题	方法	近似	运行时间
k-中心	[7]	$(1+\varepsilon)\gamma$	$\tilde{O}(k^2 N_{\text{fhw}} + T^{\text{cen}}_{\gamma}(k^2))$
	[7]	$(1+\varepsilon)\gamma$	$\tilde{O}(kN_{\text{fhw}} + k^{\lceil d/2\rceil+1} + T^{\text{cen}}_{\gamma}(k))$
	定理 5.2	$(1+\varepsilon)\gamma$	$\tilde{O}(kN_{\text{fhw}} + T^{\text{cen}}_{\gamma}(k))$
k-中位数	[31]	$(2+\varepsilon)\gamma$	$\tilde{O}(k^2 N_{\text{fhw}} + k^4 + T^{\text{med}}_{\gamma}(k^2))$
	定理 6.6	$(2+\varepsilon)\gamma$	$\tilde{O}(kN_{\text{fhw}} + T^{\text{med}}_{\gamma}(k))$
k-均值	[26]	$\gamma^2 + 4\gamma\sqrt{\gamma} + 4\gamma$	$\tilde{O}(k^m N_{\text{fhw}} + T^{\text{均值}}_{\gamma}(k^m))$
	[54]	$320 + 644(1+\varepsilon)\gamma$	$\tilde{O}(k^4 N_{\text{fhw}} + T^{\text{均值}}_{\gamma}(k))$
	[31]	$(4+\varepsilon)\gamma$	$\tilde{O}(k^2 N_{\text{fhw}} + k^4 + T^{\text{均值}}_{\gamma}(k^2))$
	定理 6.6	$(4+\varepsilon)\gamma$	$\tilde{O}(kN_{\text{fhw}} + T^{\text{均值}}_{\gamma}(k))$

表 1. 对于所有属性均为数值（实数或整数）的数据集，我们的聚类算法与最先进的相对近似方法的比较。运行时间以数据复杂度表示。我们假设 $\varepsilon \in (0,1)$ 是任意常数。使用记号 $\tilde{O}$ 隐去运行时间中的 $\log^{O(1)} N$ 因子。$T^{\text{cen}}_{\gamma}(y)$（分别为 $T^{\text{med}}_{\gamma}(y)$、$T^{\text{mean}}_{\gamma}(y)$）表示在标准计算环境下，对 y 个点进行 k-中心（分别为 k-中位数、k-均值）聚类的已知 γ-近似算法的运行时间。fhw 是 $\boldsymbol{q}$ 的分数超树宽度。请回忆，d 是查询 $\boldsymbol{q}$ 中属性的数量。

首先，在第 3 节中，我们介绍新方法的高层思路。我们着重说明了我们的方法与先前已知算法的不同之处，并提出了一种新的几何洞见，说明如何改进运行时间。

• 在第4节中，我们介绍了 BBD 树的一种变体，称为随机化 BBD 树（或简称 RBBD 树），它在标准计算环境下具有与 BBD 树相似的性质。然而，我们证明，在关系型场景中，给定 RBBD 树的一个节点，可以根据输入数据库的大小，以近乎线性的时间构造其子节点。这是为关系型场景中的各种优化问题设计更快算法的关键思路。

• 在第 5 节中，我们展示了用于关系型 k-中心聚类的高效算法。首先，我们提出˜一种 $O(1)$-近似算法，该算法以 $O(kN)$时间运行，且成功概率很高。随后，我们将新算法的结果作为输入，交由 [7] 中的核心集构造方法，得到一个$(1+\varepsilon)\gamma$-近似˜算法，其运行时间为 $O(kN + T_{\text{中心}\gamma}(k))$。

• 在第 6 节中，我们为关系型 k-中位数/均值聚类设计了高效算法。我们的思路源于 [42]，并提出一种几何方法：在关系型场景中使用 RBBD 树，为关系型 k-中位数/均值聚类设计 $O(1)$-近似算法，该算法返回 $O(k\log^3 N)$个中心。我们将返回的集合用作 [31] 中核心集构造方法（在关系型场景下）的输入，以推导出最终结果。有趣的是，我们在这项工作中的新发现还可用于加速 [31]的核心集构造。最终，我们以高概率在 $\tilde{O}(kN + T_{\text{中位数}\gamma}(k))$时间内，为关系型 k-中位数聚类获得$(2+\varepsilon)\gamma$-近似（相应地，为关系型 k-均值聚类获得$(4+\varepsilon)\gamma$-近似），其运行时间为 $\tilde{O}(kN + T_{\text{均值}\gamma}(k))$。

• 在附录 D 中，我们将结果与 [26] 的通用框架相结合，以扩展适用于混合属性数据的算法，其中属性可以是类别型或数值型。

• 得益于我们方法的通用性，在第7节和附录 E 中，我们展示了如何将新方法用于其他优化问题，从而获得更快的算法，例如关系型场景中的多样性和公平性问题。

我们新提出的关系型聚类算法的运行时间和近似因子，以及与最先进水平的比较，见表 1。

2 预备知识

我们重新介绍了下一节将用到的各种已知工具、算法和数据结构。对于点 $p \in \mathbb{R}^d$和 $r \in \mathbb{R}$，令 $\mathrm{B}(p,r) = \{x \in \mathbb{R}^d \mid \phi(p,x) \leq r\}$ 表示以 p为中心、半径为 r的球。

盒。 盒 ρ 是一个轴平行的 d维超矩形，定义为各属性上d个区间的乘积，即 $\rho = I_1 \times \ldots \times I_d$，其中 $I_i = [a_i, b_i]$对于 $a_i,\ b_i \in \mathbb{R}$。

ε-采样。 设（P，R）为一个集合系统，其中 P 是 $\mathbb{R}^d$ 中 n 个点的集合，且 R 是 $\mathbb{R}^d$中所有（无穷多个）可能盒的集合。对于参数 $\varepsilon \in$（0，1），集合 $\mathrm{N} \subseteq P$是一个 ε-采样，如果对每个 $\rho \in$R，$\left|\frac{|P \cap \rho|}{|P|} - \frac{|\mathrm{N} \cap \rho|}{|\mathrm{N}|}\right| \leq \varepsilon$。已知[18, 41]，从 P中抽取的 O（$\varepsilon^{-2}\log\psi^{-1}$）个均匀随机样本是（$P$，R）的一个 ε-采样，概率至少为 $1-\psi$。

用于聚合查询的预言机。 我们展示一些已知的预言机，它们可用于访问连接结果中的某些元组，或在不显式计算连接查询的情况下，计算连接结果的某些统计量。在接下来的章节中，我们旨在对 $\boldsymbol{q}(\mathbf{D})$ 中位于一个盒内的连接结果进行计数或采样。更正式地说，设 ρ 是 $\mathbb{R}^d$ 中的一个盒。目标是：i) 统计 $|\boldsymbol{q}(\mathbf{D}) \cap \rho|$ 中元组的数量；ii) 从 $\boldsymbol{q}(\mathbf{D}) \cap \rho$ 中均匀随机采样；iii) 报告 $\boldsymbol{q}(\mathbf{D}) \cap \rho$ 中的所有元组；以及 iv) 从 $\boldsymbol{q}(\mathbf{D}) \cap \rho$ 中任意选取一个元组。下一个引理由 [31] 得出。

引理 2.1 ([31])。 设 $\mathbf{D}$ 是由 N 个元组（具有数值属性）构成的数据库实例，且 $\boldsymbol{q}$ 是关于 $\mathbf{D}$ 的无环查询。设 ρ 是 $\mathbb{R}^d$ 中的一个盒。存在一种算法 *(1) CountRect($\boldsymbol{q}$, $\mathbf{D}$ ρ)*，在 $O(N)$ 时间内统计 $|\boldsymbol{q}(\mathbf{D}) \cap \rho|$；*(2) SampleRect($\boldsymbol{q}$, $\mathbf{D}$ ρ, z)*，在 $O(N + z \log N)$ 时间内从 $\boldsymbol{q}(\mathbf{D}) \cap \rho$ 中采样 z 个点；*(3) ReportRect($\boldsymbol{q}$, $\mathbf{D}$ ρ)*，在 $O(N + |\rho \cap \boldsymbol{q}(\mathbf{D})|)$ 时间内报告 $\boldsymbol{q}(\mathbf{D}) \cap \rho$ 中的所有点；*(4) ReprRect($\boldsymbol{q}$, $\mathbf{D}$ ρ)*，在 $O(N)$ 时间内从 $\boldsymbol{q}(\mathbf{D}) \cap \rho$ 中找到一个代表性元组。
预言机的运行时间以完美哈希为前提。[1]

BBD 树。我们采用的主要几何数据结构是 BBD 树 [12, 13], ，它是四叉树 [35] 的一种变体。集合 P 中的 n 个点在 $\mathbb{R}^d$ 中构成的 BBD 树 T 是一棵高度为 O（$\log n$）的二叉树，具有 n 个叶节点，其中 P 中的每个点恰好存储在一个叶节点中。令 $\square$ 表示包含 P 的最小体积轴对齐盒。T 的每个节点 u 都关联一个区域 $\square_u \subseteq \mathbb{R}^d$，该区域或者是一个盒，或者是两个盒之差（即带孔盒）。该区域恰好包含 P 中存储在以 u 为根的子树的叶节点中的点。如果 root 是 T 的根节点，则 $\square_{\mathsf{root}} = \square$。令 $P_u = P \cap \square_u$ 表示包含在以 u 为根的子树中的点。如果 $|P_u| = 1$，则 u 是一个叶节点。否则， u 有两个子节点，分别记为 w 和 z，使得 $\square_w$ 和 $\square_z$ 构成 $\square_u$ 的划分。因此，T 的节点所关联的区域诱导出 $\mathbb{R}^d$ 的层次划分。作为示例，考虑图 1 所示的 23 个点的集合。图 2 展示了该集合的一棵部分构造的（为简单起见略作简化的）BBD 树。图 1 中的蓝色线段表示由所构造的 BBD 树诱导的层次划分。对于图 2 中的节点 v，区域 $\square_v$ 是由角点 C 和 J 定义的盒；而对于节点 w，区域$\square_w$ 是由角点 C 和 K 定义的盒，去除了由角点 L 和 M 定义的内盒。对于图 2 中 BBD 树的每个节点 u，我们还标出了 P_u 中的点数。例如， $|P_w| = 4$，因为由 $[C,K]$ 定义的盒包含 8 个点，而孔盒 $[L,M]$ 包含 4 个点。BBD 树占用 O（n）的空间，并可在 O（$n\log n$）时间内构造完成。

BBD 树需要 $O(n)$ 的空间，并可在 $O(n\log n)$ 时间内构建完成。给定参数$\varepsilon \in (0,1)$ 以及 $\mathbb{R}^d$ 中的球 $\mathrm{B}(x, r)$，BBD 树支持一个查询过程，记为 $\mathrm{T}(x,r)$，该过程在 $O(\log n + \varepsilon^{-d+1})$ 时间内返回一个规范节点集合 $\mathrm{U}(x,r) = u_1, \ldots, u_\kappa$，其中包含 $\kappa = O(\log n + \varepsilon^{-d+1})$ 个节点

[1]为简化起见，我们假设采用完美哈希。标准的随机哈希表以高概率达到相同的渐近界；完美哈希假设仅用于避免在分析过程中处理额外的（常规）随机性。

。这些节点满足：对于所有互不相同的 i、 j，以及 $\mathrm{B}(x,r) \subseteq \bigcup_{1\le i\le \kappa} \square_{u_i} \subseteq \mathrm{B}(x,(1+\varepsilon)r)$，均有 $\square_{u_i} \cap \square_{u_j} = \emptyset$。换言之，BBD 树返回一个盒集合（可能带孔，也可能不带孔），其并集完全覆盖球 $\mathrm{B}(x,r)$，并且可能覆盖 $\mathrm{B}(x,(1+\varepsilon)r) \setminus \mathrm{B}(x,r)$ 的一部分。构建树后，查询过程十分直接。从 T 的根节点开始，并以一个初始为空的集合 $\mathrm{U}(x,r)$ 开始，算法递归遍历该树。访问节点 u 时，如果 $\square_u \subseteq \mathrm{B}(x,(1+\varepsilon)r)$，则将该节点 u 加入 $\mathrm{U}(x,r)$。如果 $\square_u \cap \mathrm{B}(x,r) = \emptyset$，则终止对该分支的遍历。否则，该过程继续递归处理 u 的两个子节点。例如，在图 1 中，令 $\mathrm{B}(x,r)$ 表示内部红色圆，令 $\mathrm{B}(x,(1+\varepsilon)r)$ 表示外部虚线圆。相应的规范节点 $\mathrm{U}(x,r)$ 在图 2 中以红色节点标出。我们注意到，$\mathrm{U}(x,r)$ 中的盒覆盖了内部红色圆中的所有点，此外还覆盖了环带 $\mathrm{B}(x,(1+\varepsilon)r) \setminus \mathrm{B}(x,${v46））中的一个点 P，因为具有 N和 S 这两个对角顶点的盒完全位于外部虚线圆内。BBD 树可用于不同类型的查询，例如报告查询、计数查询或采样查询。

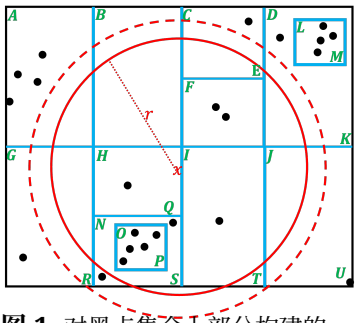


图 1. 对黑点集合上部分构建的 BBD 树的划分。红色圆表示查询球 $\mathrm{B}(x,r)$，而较大的虚线圆表示 $\mathrm{B}(x,(1+\varepsilon)r)$。

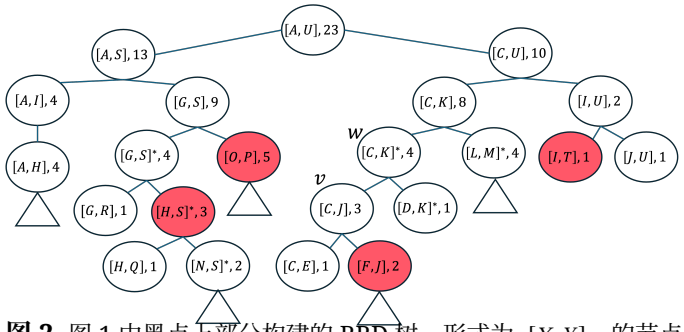


图 2. 图 1 中黑点上部分构建的 BBD 树。形式为 $[X,Y],c$ 的节点 u 具有由对角顶点 X,Y 和 $|P_u| = |\square_u \cap P| = c$ 定义的区域 $\square_u$。带有 $[X,Y]^*,c$ 的节点对应于区域为带孔盒的节点。某些节点下方的三角形表示为简洁清晰而省略的子树。红色节点表示与图 1 中所示球 $\mathrm{B}(x,r)$ 对应的规范节点集合 $\mathrm{U}(x,r)$。

3 概述

在本节中，我们描述本方法的直觉与高层思想。在后续各节中，我们将正式证明本节所提出的论断。

以往的方法。此前已知的大多数关系聚类方法 [7, 22, 26, 31, 54]都遵循相同的总体思路。它们首先设计一种拥有超过 k 个中心的常数近似算法。设 Z 为该集合。然后使用 Z 为关系聚类问题构造一个规模较小的核心集 C。最后，在标准计算环境下运行一种已知的聚类算法，以 C 为输入得出最终解。有趣的是，核心集构造相对较快。在大多数情况下，运行时间主要由第一步决定，即构造一个可能包含超过 k 个点的常数近似解。例如，考虑当前最先进的关系型 k-中心 [7] 聚类算法和关系型 k-中位数/均值聚类算法 [31]。在这些论文中，作者使用一种层次聚合方法，以 $\tilde{O}(k^2N)$的时间构造了具有 $\tilde{O}(k^2)$个中心的 $O(1)$-近似算法。他们的总体思路是将R 中的所有关系放置为一棵平衡二叉树的叶节点。对于一个叶节点 u_1，在标准计算环境下运行标准聚类算法，仅考虑存储在 u_1中的关系的元组。设 S_1 为返回的 k 个中心的集合。同理，他们从兄弟节点 u_2获得 S_2。然后，他们证明 $\mathrm{S}_{1\times 2} = \mathrm{S}_1 \times \mathrm{S}_2$ 是针对连接叶节点中表所得结果的聚类问题的 $O(1)$-近似解

。然后，他们基于 $S_{1\times 2}$ 构造核心集，耗时 $O(|S_{1\times 2}|\cdot N)$，并在树的更高层级继续递归。尽管这种方法有助于设计 $O(1)$-近似算法，但其缺点是会构造规模为 $\Theta(k^2)$ 的中间集合（例如 $S_{1\times 2}$），从而导致算法运行时间达到 $\Omega(k^2\cdot N)$。

我们的方法。我们以 k-中心聚类为例，介绍我们方法的总体思路。设 P 为 $\mathbb{R}^d$ 中的 n 个点的集合，k 为标准计算环境中的一个正整数参数。在计算几何中，已有多种用于 k-中心聚类的高效常数近似算法。虽然这些算法在标准计算环境中速度很快，但如何将其高效扩展到关系型场景中尚不明确。考虑以下针对 P 上 k-中心问题的简单算法。令 $r=\mathbf{u}^{\mathrm{OPT}(P)}(P)$ 为最优代价，并假设预先已知 r。在第一次迭代中，我们将任意一个点 $p_1\in P$ 加入 S。我们从 P 中移除 $\mathrm{B}(p_1,2r)\cap P$ 中的所有点，也就是说，移除 P 中距 p_1 距离为 $2r$ 以内的所有点，然后继续下一次迭代。在第 i 次迭代中，我们从 $\bigcup_{j<i}\mathrm{B}(p_j,2r)$ 中任意选择一个点 $p_i\in P$，移除 $\mathrm{B}(p_i,2r)\cap P$，并继续执行，总共进行 k 次迭代。容易验证，该算法能为 P 上的 k-中心问题返回一个 2-近似解。假设已知最优 r，该算法的朴素实现需要 $O(nk)$ 时间。如果未知 r，我们就需要对 P 的距离进行二分搜索（使用良分离对分解 [19, 43]），从而使运行时间达到 $O(nk\log n)$。不过，在标准计算环境中，这个算法有一种更快的实现方式。我们在 P 上构建一棵 BBD 树 T。对于每个节点

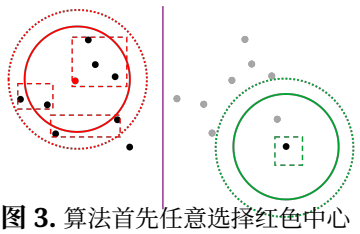


图 3. 算法首先任意选择红色中心（左）。利用 BBD 树，它找出规范节点（三个红色虚线矩形），这些节点包含距红色中心距离为 $2r$ 以内的所有点，以及距离为 $(1+\varepsilon)2r$ 以内的三个点。随后，这些规范节点被标记为非活动。接着，算法选择绿色点（右），因为它位于活动区域内，并找出距离为 $2r$ 以内的规范节点（绿色虚线矩形），然后将其标记为非活动。最后，所有点都位于非活动区域中，因此算法终止，并返回一个 $(2+\varepsilon)$-近似解。

u 对于 T，我们从以 u 为根的子树的叶节点中存储的点里存储一个代表点 $u.rep$，即 $u.rep\in P_u$（见第4节）。在第一次迭代中，$p_1=\mathrm{root}.rep$ 是根节点中的代表点。然后我们执行查询 $\mathrm{T}(p_1,\ 2r)$，得到规范节点集合 $\mathrm{U}(p_1,\ 2r)$。我们将 $\mathrm{U}(p_1,\ 2r)$ 中的每个节点标记为“非活动”。接着从 $\mathrm{U}(p_1,\ 2r)$ 中最深的节点开始，更新活跃节点的代表点，以确保没有代表点位于以非活动节点为根的子树的叶节点中。在接下来的 $k-1$ 次迭代中，我们以类似方式继续，只访问活跃节点。该实现返回一个集合 $S\subseteq P$，使得 $\mathbf{u}_S(P)\le 2(1+\varepsilon)\mathbf{u}^{\mathrm{OPT}(P)}(P)$，其运行时间为 $O(n\log n+k(\varepsilon^{-d+1}+\log n))=O(n\log n)$，且处于标准计算环境中。标准计算环境下更快算法的示例如图3所示。

在关系型场景中，直接实现基于 BBD 树的 k-中心聚类算法，代价会很高。仅构建覆盖所有连接结果的 BBD 树，就需要 $\Omega(|q(\mathbf{D})|)$ 时间。不过，我们有一个简单但关键的观察。**我们其实不必预先构建整棵树。如果即时构建这棵树会怎样？**再来考虑标准计算环境。最初，假设我们只构建了树 T 的根节点 root。目标是计算 $\mathrm{U}(p_i,\ 2r)$。设 v 是查询过程遍历到的 T 已构建部分中的一个节点。如果尚未构建 v 的子节点，我们就构建左子节点 v_1 和右子节点 v_2，然后继续执行常规查询过程。对于 BBD 树中的一个节点 v，如果我们能在 $O(t)$ 时间内构建它的两个子节点 v_1，v_2，那么

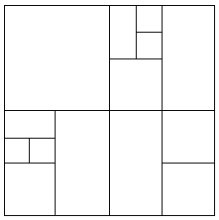
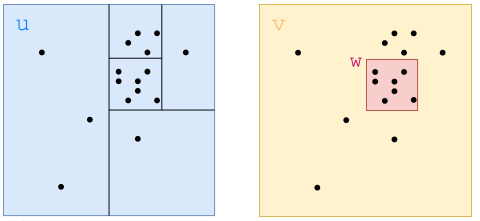


图 4. 所有单元都是中点盒。



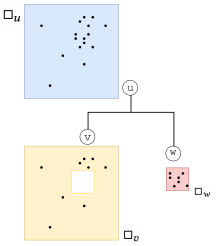


图 5. 通过质心收缩过程查找节点 u的子节点 v, w的示意图。区域 $\square_v$ 是一个带孔盒，而 $\square_w$ 是一个盒。

执行时间上界为 $O(k \cdot t \cdot (\log n + \varepsilon^{-d+1}))$。虽然这是标准计算环境中的一个宽松上界，但我们用它来界定关系型场景中的查询过程。

我们的核心思路是在关系型场景中即时构造一种新型 BBD 树。这使得我们能够设计出若干新的关系型预言机，进而为各种优化问题带来目前已知最快的近似算法。虽然这一思路在概念上很简单，但在关系型场景中构造 BBD 树需要多个步骤，结合计算几何（BBD 树）、数据库理论（关系型计数与采样预言机）和概率分析（ε-样本）中的技术。我们方法的一大优势是通用性。借助所提出的数据结构，标准场景中的许多经典算法都可以轻松直接地转换到关系型场景中。这与先前关于关系聚类的研究（例如 [7, 31, 54]）形成鲜明对比；在那些研究中，算法是针对特定目标专门设计的，并且往往与其经典版本大相径庭。

在下一节中，我们将提出一种 BBD 树的新变体，称为随机化 BBD 树（简称 RBBD 树）。对于关系型场景，如果给定 $\boldsymbol{q}(\mathbf{D})$ 上 RBBD 树的一个节点 v，则可以在 $O(N + \varepsilon^{-2} \log N)$ 时间内计算出其子节点v_1, v_2。

4 RBBD 树

在介绍 RBBD 树之前，我们先概述一下 [13] 中所述的标准 BBD 树的构造方法。接着，我们将介绍 BBD 树的一种随机化变体，称为 RBBD 树。尽管在标准计算环境中，RBBD 树并不比 BBD 树有优势，但我们将证明，在关系型场景中它很有用，因为可以高效地构造 $\boldsymbol{q}(\mathbf{D})$ 上 RBBD 树任意节点的子节点。

4.1 关于 BBD 树构造的更多细节

设 P 是 $\mathbb{R}^d$ 中由 n 个点构成的集合。回想一下，BBD 树的每个节点 u 都关联着一个区域$\square_u$，该区域要么是一个盒，要么是一个带孔盒；还要回想一下 $P_u = \square_u \cap P$。在后一种情况下，设 $\square_u^+$ 为外盒，$\square_u^-$ 为内盒（孔）。中点盒 定义为通过递归应用以下规则所能得到的任意盒，递归从初始包围盒$\square_{\text{root}}$开始：设 ρ 为一个中点盒，令 ξ 为 ρ的最长边。用一个垂直于第 ξ 个坐标轴、经过 ρ中心的超平面，将 ρ 分成两个完全相同的盒。中点盒的一个示例见图 4。树的构造方式保证：如果 $\square_u$ 是一个盒，那么 $\square_u$ 是一个中点盒；如果 $\square_u$ 是一个带孔盒，那么 $\square_u^+$ 和 $\square_u^-$ 都是中点盒。以上性质对树中的任意节点 u均成立。 u的子节点通过以下两种方法之一构造：公平划分 和质心收缩。在树的根节点处，采用公平划分来构造其子节点。其他情况下，如果 u 是通过公平划分（分别为质心收缩）生成的，则其子节点通过质心收缩（分别为公平划分）生成。

公平划分。 他们计算一个经过 $\square_u$ 中心、且与 $\square_u$ 最长边正交的超平面（如果 $\square_u$ 是带孔盒，则与 $\square_u^+$ 正交）。请注意，即使 $\square_u$ 含有一个孔，将 $\square_u$ 分成两个盒的超平面也不会与 $\square_u^-$ 相交，因为 $\square_u^-$ 和 $\square_u^+$ 都是

中点盒。在这种情况下，如果 w， v 是 u的子节点，那么 $\square_w$ 将是一个盒，而 $\square_v$ 将是一个带孔盒，其中有 $\square_v^- = \square_u^-$。公平划分在 $O(1)$时间内完成。

质心收缩。在较高层次上，目标是找到一个中点盒 ρ^*，使其位于 $\square_u$ 内，并满足 $|\rho^* \cap P| \leq \frac{2}{3}|P_u|$ 和 $|(\square_u \setminus \rho^*) \cap P| \leq \frac{2}{3}|P_u|$。为了计算 ρ^*，他们的算法通过迭代来运行。我们描述构造过程时，假设 $\square_u$ 是一个无洞盒子。在附录 A.1 中，我们讨论 $\square_u$ 是带孔盒时的构造过程。首先，令 $\rho = \square_u$。他们计算包含 $P_u \cap \rho$ 中所有点的最小中点盒。令 $\rho' \subseteq \rho$ 为这个盒。他们在 ρ' 上应用 $O(d)$ 次公平划分，直到找到一个非平凡公平划分，即将 $P_u \cap \rho'$ 中的点划分为两个非空子集的公平划分。令 $\hat{\rho} \subset \rho'$ 为非平凡公平划分所产生的两个盒之一，且满足 $|\hat{\rho} \cap P_u| \geq \frac{|\rho' \cap P_u|}{2}$。我们称 ~~$\hat{\rho}$ 为多~~数中点盒。关键观察是 $|\hat{\rho} \cap P_u| \leq |\rho \cap P_u| - 1$。如果 $|\hat{\rho} \cap P_u| > \frac{2}{3}|P_u|$，他们就令 $\rho = \hat{\rho}$，并继续递归。如果 $|\hat{\rho} \cap P_u| \leq \frac{2}{3}|P_u|$，那么 $\rho^* = \hat{\rho}$，随后他们构造 u 的两个子节点 w、v，使得 $\square_w = \rho^*$ 是一个盒（无孔），而 $\square_v$ 是一个带孔盒，且具有 $\square_v^+ = \square_u$ 和 $\square_v^- = \rho^* = \square_w$，如图 5 所示。如 [13], 所示，包含 $P_u \cap \rho$ 中所有点的最小中点盒 $\rho' \subseteq \rho$ 可以在 O（1）时间内找到。由于在每次递归迭代中 $|\hat{\rho} \cap P_u| \leq |\rho' \cap P_u| - 1$，质心收缩过程的执行时间为 $O(|P_u|)$。

4.2 RBBD 树的构造

我们在标准计算环境中，为点集 n points $P \subset \mathbb{R}^d$ 定义了一种称为 RBBD 树的新型 BBD 树变体。RBBD 树的构造与 BBD 树共享大部分步骤和性质。例如，所有构造出的盒（包括内盒和外盒）都是中点盒。公平划分与 BBD 树的公平划分完全相同。唯一的区别在于质心收缩过程的定义和执行方式，具体取决于节点 u。

我们定义随机质心收缩。设 u 为一个节点。我们从 $\square_u \cap P_u$ 中获取一个包含 $\Theta(\varepsilon^{-2}\log n)$ 个均匀随机样本的集合 $\mathbf{N}_u$。如果 $\square_u$ 是一个无洞盒子，我们只需在盒 $\square_u$ 中采样。如果 $\square_u$ 是一个带孔盒，我们就在 $\square_u^+$ 和 $\square_u^-$ 之间的区域中采样。接着，该过程与 BBD 树中的质心收缩类似，只是使用 $\mathbf{N}_u$ 代替 P_u。我们递归地计算包含 $\mathbf{N}_u$ 的最小中点盒 ρ'，并应用 $O(d)$ 公平划分，直到找到一个非平凡公平划分。然后，我们在非平凡公平划分过程生成的、包含 $\mathbf{N}_u$ 中最多点的盒 $\hat{\rho}$（多数中点盒）上递归地继续构造。我们重复这一递归方法，直到找到一个多数中点盒 $\hat{\rho}$（经非平凡公平划分后），其中至多包含来自 $\mathbf{N}_u$ 的 $\frac{2}{3}|\mathbf{N}_u|$ 个样本。如果 $\square_u$ 是一个带孔盒，该过程见附录 A.1。在附录 A.2 中，我们利用 BBD 树和 RBBD 树的性质证明下一个定理。

定理4.1. 给定由 n 个点组成的集合 P，参数 $\varepsilon \in \in (0, 1)$，以及常数参数$\zeta > 1$，*RBBD* 树的高度为 $O(\log n)$，其构造可在 $O(n \log n + n \cdot (f(n, \varepsilon) + \varepsilon^{-2} \log n))$ 时间内完成，所需空间为 $O(n)$；其中 $f(n, \varepsilon)$ 表示从一个盒（或带孔盒）中采样 $\Theta(\varepsilon^{-2} \log n^{\zeta})$ 个点所需的运行时间。给定一个针对点 $p \in \mathbb{R}^d$、半径为r的查询球 $B(p, r)$，可以在 $O(\log n + \varepsilon^{-d+1})$ 时间内找到一个包含 $O(\log n + \varepsilon^{-d+1})$ 个规范节点的集合 U，使得 $B(p, r) \subseteq P \cap \bigcup_{u \in U} \square_u \subseteq B(p, (1+\varepsilon)r)$。该树的高度、空间、构造时间、$U$ 的大小以及查询时间均以至少 $1 - \frac{1}{n^{\zeta-1}}$ 的概率成立。

4.3 关系型场景中的 RBBD 树

设 q 是数据库实例D上的一个无环连接查询。由于 $q(\mathbf{D})$ 的规模，我们无法按照上一小节中的描述构造完整的 RBBD 树。相反，我们将展示：在已构造 u，且它是 $q(\mathbf{D})$ 上某棵 RBBD 树的节点的情况下，如何构造节点 u 的子节点。更具体地说，我们将展示如何在关系型场景中高效地实现 RBBD 树的公平划分和随机质心收缩。此外，对于

RBBD 树的每个新节点 v，我们将展示如何计算计数 $v.c = |\square_v \cap \boldsymbol{q}(\mathbf{D})|$ 和代表元 $v.rep \in \square_v \cap \boldsymbol{q}(\mathbf{D})$。这些信息对于设计我们的算法非常有用。

根节点。我们首先展示如何构造 RBBD 树的根节点 root，它始终是一个无洞盒子。对于每个属性 $A_i \in \mathbf{A}$，我们计算 $\pi_{A_i}(\mathbf{D})$，并保留最大值M_i和最小值 m_i。我们将 $\square_{\mathsf{root}}$ 定义为一个无洞盒子，其对角顶点为 $(m_1, \ldots, m_d)$ 和 $(M_1, \ldots, M_d)$。该过程可直接在 $O(N)$ 时间内执行。

公平划分。关系型场景下公平划分的实现与标准计算环境下的公平划分完全相同。我们只需在 $O(1)$ 时间内确定盒的最长边及其中心。

随机质心收缩。我们只需说明如何实现采样过程，因为所有其他步骤都可以用与标准计算环境类似的方式执行。我们需要从 $\square_u \cap \boldsymbol{q}(\mathbf{D})$ 中均匀随机采样 $\Lambda = \Theta(\varepsilon^{-2}\log(N^{m+4})) = \Theta(\varepsilon^{-2}\log N)$ 个点。如果$\square_u$是一个盒（无孔），那么我们只需调用引理 2.1 中的 $\mathsf{SampleRect}(\boldsymbol{q}, \mathbf{D}, \square_u, \Lambda)$，即可在 $O(N + \varepsilon^{-2}\log^2 N)$ 时间内得到集合 $\mathbf{N}_u$。在附录 A.3 中，我们展示了当 $\square_u$ 是带孔盒时，如何在相同的运行时间内从 $\square_u \cap \boldsymbol{q}(\mathbf{D})$ 中进行采样。

新构造节点中的计数和代表元组。如果 $\square_u$ 是一个无洞盒子，我们调用引理 2.1 中的 $u.c = \mathsf{CountRect}(\boldsymbol{q}, \mathbf{D}, \square_u)$和 $u.rep = \mathsf{ReprRect}(\boldsymbol{q}, \mathbf{D}, \square_u)$。这两个过程的运行时间均为 $O(N)$。在附录 A.4 中，当 $\square_u$ 是一个带孔盒时，我们也能在相同的运行时间内计算计数和代表点。

引理 4.2。设 $\boldsymbol{q}$ 是数据库实例$\mathbf{D}$ 上的一个无环连接查询，数据库实例的大小为 N。给定 $\boldsymbol{q}(\mathbf{D})$上的 *RBBD* 树的一个节点 u，公平划分过程的执行时间为 $O(N)$，而随机质心收缩过程的执行时间为 $O(N + \varepsilon^{-2}\log^2 N)$。

4.4 使用部分构造的 RBBD 树构建关系型预言机

利用引理 4.2 和定理4.1，我们推导出新的预言机，从而能够高效地在关系型场景中求解优化问题。设 T 是$\boldsymbol{q}(\mathbf{D})$上的一棵部分构造的 RBBD 树，即 T 由 $\boldsymbol{q}(\mathbf{D})$上的 RBBD 树中一个连通的节点子集（包含根节点，记为 root）构成。最初，T 仅包含根节点，该节点如第 4.3 节所述构造。对于 T 中每个已构造节点 $u \in$，我们存储 i) $\square_u$与节点 u对应的区域，该区域要么是一个盒，要么是一个带孔盒；以及 ii) $u.a$ 一个布尔变量：若 u 是活跃节点，则其值为 1；若 u 是非活动节点，则其值为 0。如果存在 T 中的节点 $u \in$ 使得$p \in \square_u$ 且 $u.a = 0$，则称点 $p \in \boldsymbol{q}(\mathbf{D})$为非活跃点。不属于非活跃点的点称为活跃点。虽然我们不会显式计算它，为方便起见，我们用 $Q' \subseteq \boldsymbol{q}(\mathbf{D})$表示 $\boldsymbol{q}(\mathbf{D})$中当前活跃点的集合。此外，对于 T 中的每个节点 $u \in$，我们存储 $u.rep$，它是 $\square_u \cap Q'$中的一个任意代表点。最后，对于 T 中的每个节点 $u \in$，我们存储区域 $\square_u$中的活跃点数量，即 $u.c = |\square_u \cap Q'|$。

给定这样一棵部分构造的树 T，我们展示如何在关系型场景中实现五个预言机，同时维护 T（可能需要构造新的节点）。本节中每个预言机的运行时间以至少为 $1 - \frac{1}{N^{m+3}}$ 的概率成立。由于篇幅所限，所有引理的证明见附录 A.5。

将球中的点设为非活动状态。给定一个球 $\mathbf{B}(x, r)$，其中 $x \in \mathbb{R}^d$和 r是非负实数，目标是（隐式地）将 $\mathbf{B}(x, r)$ 中的所有点设为 $\cap Q'$非活动状态。令 $\mathbf{T}.\mathsf{inactive}(x, r)$ 为此预言机，其工作方式如下。首先，我们执行一个类似于 BBD 树上的查询过程。令 $\mathbf{U}(x, r)$ 为我们将要计算的规范节点集合。令 u为查询过程处理的 T 中的一个节点。初始时，$u =$为根节点，且 $\mathbf{U}(x, r) = \emptyset$。如果 $u.a = 0$，我们就停止沿着这条树分支进行查询。以下我们假设 $u.a = 1$。

如果尚未在 T 中构造 u 的子节点，我们就使用引理 4.2 构造其子节点。接下来，我们继续检查 $\square_u$是否与 $\mathrm{B}(x,r)$ 相交。如果 $\square_u \subseteq \mathrm{B}(x,(1+\varepsilon)r)$，则将 u添加到 $\mathrm{U}(x,r)$ 中，并停止沿着这条树分支执行。如果 $\square_u \cap \mathrm{B}(x,r) = \emptyset$，我们就停止沿着这条树分支进行查询。其他情况下，我们递归地在 u的两个子节点上继续查询过程。对于每个添加到 $\mathrm{U}(x,r)$ 中的节点 $u \in$，我们设置 $u.a = 0$。在结束该过程之前，我们需要更新 $\mathrm{U}(x,r)$ 的祖先节点的代表点和计数。从 $\mathrm{U}(x,r)$ 中最深节点的父节点 v开始，我们递归执行以下操作：令 u,w 为 v的两个子节点。如果 $u.a = w.a = 0$，则设置 $v.a = 0$。如果 $u.a = 1$和 $w.a = 0$（分别为 $u.a = 0$和 $w.a = 1$），则设置 $v.c = u.c$和 $v.rep = u.rep$（分别为 $v.c = w.c$ 和 $v.rep = w.rep$）。如果 $u.a = w.a = 1$，则设置 $v.c = u.c + w.c$和 $v.rep = u.rep$。处理完树同一层中所有规范节点的祖先节点后，我们继续更新上一层的节点。

引理 4.3。$\mathcal{T}$.非活动(x,r)使 $\mathcal{B}(x,r) \cap Q'$ 中的所有点变为非活动状态，并且可能使 $\mathcal{B}(x,(1+\varepsilon)r) \cap Q'$ 中的某些点变为非活动状态。此外，在预言机执行结束时，对于每个节点$u \in \mathcal{T}$，值 $u.rep, u.c$ 都是正确的。运行时间为 $O((\varepsilon^{-d+1} + \log N)(N + \varepsilon^{-2}\log^2 N))$。

近似统计球内的活跃点。给定一个球 $\mathrm{B}(x,r)$，目标是近似计算 $|\mathrm{B}(x,r) \cap Q'|$。我们定义预言机 $\mathrm{T}.\mathrm{count}(x,r)$，并运行与 $\mathrm{T}.\mathrm{inactive}(x,r)$ 相同的查询过程，以计算 $\mathrm{U}(x,r)$。我们返回 $\sum_{u \in \mathcal{U}(x,r)} u.c$。

引理 4.4。$\mathcal{T}.\mathrm{count}(x,r)$返回一个数 c，使得 $|\mathcal{B}(x,r) \cap Q'| \le c \le |\mathcal{B}(x,(1+\varepsilon)r) \cap Q'|$，且运行时间为 $O((\varepsilon^{-d+1} + \log N)(N + \varepsilon^{-2}\log^2 N))$。

从 Q' 中获取一个代表元。接下来，目标是返回 Q' 中任意一个点。我们定义预言机 T。rep()。如果 root.$a = 0$，那么我们返回 $\emptyset$，即 $Q' = \emptyset$。如果 root.$a = 1$，那么我们返回 root.rep。

引理 4.5。$\mathcal{T}$.rep() 在 $Q' \neq \emptyset$ 时返回 Q' 中的一个点，否则返回 $\emptyset$ 中的一个点，耗时 $O(1)$。

报告 Q' 中的所有点。假设我们要报告 Q' 中的所有点。我们定义预言机 T 。report()。从根节点 root 开始，我们对 T 执行深度优先搜索。如果到达节点 u，且 $u.a = 0$，则停止遍历树的这一分支。如果到达节点 u，且尚未构造其子节点，或者 u 是 RBBD 树的叶节点，则运行 ReportRect(q，D $\square_u$)。我们继续进行深度优先搜索，直到访问完 T 中所有已构造节点。

引理 4.6. $\mathcal{T}$.report() 在 $O(|\mathcal{T}|N + |Q'|)$时间内返回 Q'，其中 $|\mathcal{T}|$是$\mathcal{T}$中的节点数。

从 z 中采样点 Q'。目标是从 Q' 中均匀随机地采样（放回） z 个点。我们描述预言机 T.sample(z)。我们使用 T 逐个获取每个样本。如果 root.$a = 0$，则不返回样本，因为 $Q' = \emptyset$。否则，我们令 $u =$ 为 root，并按如下方式操作。如果 u 有两个活跃子节点 v，w，那么我们以 $\frac{v.c}{v.c+w.c}$ 的概率移动到节点 v，并以 $\frac{w.c}{v.c+w.c}$ 的概率移动到节点 w。如果 $v.a = 1$ 和 $\overline{w.a = 0}$（分别为 $v.a = 0$ 和 $w.a = 1$），则移动到节点 v（分别为 w）。我们递归地重复此过程，直到到达 RBBD 树的叶节点，或到达其子节点尚未在 T 中构建的活跃节点。如果到达叶节点，则直接将该叶节点中存储的点加入样本集。如果到达节点 u，且其子节点尚未在 T 中构建，则按照第 4.3 节所述，从 $\square_u \cap q$（D）中采样一个点。

Lemma 4.7. $\mathcal{T}$.采样(z)在 $O(z \cdot N)$时间内从 Q'均匀随机返回 z个样本。

5 k-中心聚类

在第 3 节中，我们已经讨论了高效求解关系型 k-中心聚类的高层思路。在此，我们描述全部细节及形式化证明。

5.1 一种常数近似算法

我们首先介绍算法的主要部分：给定距离 r，如果 r 足够大，则返回一个集合 S，使得 $\mathbf{u}_S(q(\mathrm{D})) \le (2+\varepsilon)r$。之后，我们对 $q(\mathrm{D})$ 中的距离进行二分搜索，以获得关系型 k-中心聚类的常数近似。

主算法。令 T 由 $\boldsymbol{q}(\mathbf{D})$ 上的 RBBD 树的根节点 root 构成，如第 4.3 小节所示。初始时，我们令 S $=\emptyset$。我们至多执行以下步骤 k 次迭代。在第i次迭代中，我们令 $p_i=$T .rep()，并将 p_i 加入 S。然后调用 T .inactive($p_i, 2r$)。如果经过 k 次迭代后，T.rep() $\neq\emptyset$，则返回空集 S $=\emptyset$。另一方面，如果 T.rep() 在 k 次迭代之内 $=\emptyset$，则返回 S。算法的伪代码见算法 1。

分析。若 $r\geq \mathrm{u}^{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$，我们证明主算法会返回一个非空集合 S。根据引理 4.3，对于每个新点 p_i，$\boldsymbol{q}(\mathbf{D})$ 中距离不超过 $2r$ 的所有（活动）点都会通过 T 变为非活动$p_i, 2r$（同时还有一些距离不超过 $2(1+\varepsilon)r$ 的点）。因此，正如第 3 节所讨论的，在每次迭代中，至少有一个最优簇中的所有点都会变为非活动。因此，经过 k次迭代后，$\boldsymbol{q}(\mathbf{D})$ 中的所有点都变为非活动（即 T $.rep()=\emptyset$），且 S $\neq\emptyset$。因此，$\mathrm{u}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D}))\leq 2(1+\varepsilon)r$。通过初始设定 $\varepsilon\leftarrow\varepsilon/2$，我们得到：若 $r\geq \mathrm{u}^{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\mathbf{D})$，则 $\mathrm{u}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D}))\leq(2+\varepsilon)r$。

我们运行 T.非活动$(\cdot,\cdot)$ 和 T.rep() 预言机 $O(k)$ 次。每个预言机的运行时间为 $O((\varepsilon^{-d+1}+\log N)(N+\varepsilon^{-2}\log^2 N))$，且成功概率至少为 $1-\frac{1}{N^{m+3}}$，如第 4.4 节所示。总体而言，主算法的运行时间为 $O\left(k\left(\varepsilon^{-d+1}+\log N\right)\left(N+\varepsilon^{-2}\log^2 N\right)\right)=O(k\cdot N\cdot\log N+k\cdot\log^3 N)=\tilde{O}(k\cdot N)$，假设 $\varepsilon\in(0,1)$ 是任意常数，且成功概率至少为 $1-\frac{1}{N^2}$。

完整算法。假设我们对 $\boldsymbol{q}(\mathbf{D})$ 中的两两距离进行二分搜索。对于二分搜索选出的每个距离 r，我们运行主算法。如果它返回一个非空集合 S，则继续对更小的 r 值进行二分搜索；否则，继续搜索更大的 r 值。最后，返回二分搜索过程中最后一次计算得到的非空集合 S，作为最终的中心集合。根据主算法的分析以及第 3 节中的讨论，此算法返回一个集合 S，使得 $\mathbf{u}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D}))\leq(2+\varepsilon)\mathbf{u}^{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$。遗憾的是，我们无法对 $\boldsymbol{q}(\mathbf{D})$ 中的所有两两距离排序后再对它们进行二分搜索，因为这需要 $\Omega(|\boldsymbol{q}(\mathbf{D})|^2)$ 的时间。虽然对 ℓ_2 两两距离进行二分搜索并不简单，但关键思路是对 $\boldsymbol{q}(\mathbf{D})$ 中各点的 ℓ_∞ 两两距离进行二分搜索。我们在附录 B 中展示二分搜索的细节。利用这样一个事实：对于任意一对点t，$p\in\mathbb{R}^d$，$||t-p||_\infty\leq||t-p||_2\leq\sqrt{d}||t-p||_\infty$，我们在附录 B 中证明下面的定理。

定理 *5.1*。给定一个包含 d 个属性的无环连接查询 $\boldsymbol{q}$、一个大小为 N 的数据库实例 D、一个参数 k，以及一个任意小的常数参数 $\varepsilon\in(0,1)$，存在一种算法，能够计算出一个包含 k 个点的集合 $S\subseteq\boldsymbol{q}(D)$ 和一个实数 $r_{\mathcal{S}}$，使得 $u_{\mathcal{S}}(\boldsymbol{q}(D))\leq r_{\mathcal{S}}\leq(2\sqrt{d}+\varepsilon)u^{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(D))$。该算法的运行时间为 $O(kN\log^2 N+k\log^4 N)$，且成功概率至少为 $1-\frac{1}{N}$。

请注意，定理 5.1 中的算法返回了 $\mathbf{u}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D}))$ 的估计值 $r_{\mathcal{S}}$。这有助于在第 5.2 节中推导出关系型 k-中心聚类的更优近似算法，并如第 6 节所示，得到关系型 k-中值/均值聚类的常数近似算法。

5.2 更优近似算法

利用定理 5.1 的结果以及来自 [7], 的核心集构造，我们得到了目前已知最快的关系型 k-中心聚类问题的 $(2+\varepsilon)$-近似算法。首先，我们执行定理 5.1 中的算法，得到 S 和 $r_{\mathcal{S}}$。随后将 S 和 $r_{\mathcal{S}}$ 一并作为 [7] 中核心集构造的输入，从而得到本节的主要结果。

定理 *5.2*。给定一个定义在 d 个属性上的无环连接查询 $\boldsymbol{q}$、一个大小为 N 的数据库实例 D、一个参数 k，以及一个任意小的常数参数 $\varepsilon\in(0,1)$，存在一种算法，能够计算集合 $S\subseteq\boldsymbol{q}(D)$，其中包含 k 个点，并满足 $u_{\mathcal{S}}(\boldsymbol{q}(D))\leq(1+\varepsilon)\gamma u^{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(D))$。该算法的运行时间为 $O(kN\log^2 N+k\log^4 N+T_{cen_\gamma}(k))$，且成功概率至少为 $1-\frac{1}{N}$。

例如，如果在由 [7] 构造的核心集上执行 Feder 和 Greene 算法 [33]，那么我们可以在额外的 $O(k\varepsilon^{-d}\log k)$ 时间内得到一个 $2(1+\varepsilon)$-近似解。令 $\varepsilon\leftarrow\varepsilon/2$ 后，

近似因子变为 $2+\varepsilon$。对于任意小的常数$\varepsilon \in (0,1)$，总体渐近复杂度为 $O(kN\log^2 N + k\log^4 N)$。在附录 B 中，我们给出一种替代的 $(2+\varepsilon)$- 近似算法，其运行时间相同，且无需从 [7] 构造核心集。

6 k-中位数与 k-均值聚类

我们重点讨论关系型 k-均值聚类，而 k-中位数算法几乎可以原样照搬。在本节中，我们首先介绍一种常数近似算法，可返回$O(k)$个中心。接着，将返回的集合用作 [31] 中的核心集构造的输入，从而推导出最终的近似算法，并恰好获得 k 个中心（第 6.2 小节）。我们的新工具还可以改进 [31], 中的核心集构造，进而得到更快的算法。

6.1 使用 O（k）个中心的常数近似算法

用于 $\boldsymbol{q}(\mathrm{D})$ 上关系型 k-中位数/均值问题的 $\mathrm{Im}_{\mathrm{ortho}}(k$ 时间内运行，并返回一个包含 $O(k\log^3 N)$ 个中心的集合。我们提出在关系型场景下对 [42] 中算法的一种修改。然而，[42] 的算法严重依赖于标准计算环境的性质，如何将其扩展到关系型场景并非易事。我们使用新工具即时构建 RBBD 树，从而为关系型 k-均值/中位数问题实现高效的常数近似。

我们首先在 $\boldsymbol{q}(\mathrm{D})$ 上执行关系型 k-中心算法，以获得最优 k-均值/中位数代价的多项式上界和下界。起初， $\boldsymbol{q}(\mathrm{D})$ 中的所有点都被视为活跃。随后，该算法以迭代方式运行。在第 i 次迭代中，我们从活跃点中得到一个由 $O(k)$ 个样本组成的小集合 X_i。接下来， $\boldsymbol{q}(\mathrm{D})$ 中与 X_i 足够接近的元组被视为已分配给 X_i 中的最近点，并被隐式地从 $\boldsymbol{q}(\mathrm{D})$ 中移除（变为非活动）。重复此过程，直到 $\boldsymbol{q}(\mathrm{D})$ 中的每个点都被移除。我们注意到，某些被分配给样本的点所付出的 k-中位数/均值代价，可能远大于它们在最优解 $\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))$ 中所付出的代价。这些点称为坏点，而其余点称为 好点。与 [42], 类似，我们证明每次迭代中好点的数量远大于坏点的数量，从而可以将坏点的代价分摊到好点上。与 [42], 不同的是，作者在标准计算环境中将每个点分配给一个中心，而我们使用在关系型场景中实现 RBBD 树的机制，将 $\boldsymbol{q}(\mathrm{D})$ 中的点组隐式分配给近似最近中心。

算法。首先，我们通过运行 Yannakakis 算法的第一阶段来计算 $|\boldsymbol{q}(\mathrm{D})|$。按照第 4.3 节中的描述，我们在 $\boldsymbol{q}(\mathrm{D})$ 上构造 RBBD 树 T 的根节点 root。令 S 为我们将作为 k-均值聚类解返回的点集合，并令 $r_{\mathcal{S}}$ 为代价 $\mu_{\mathcal{S}}(\boldsymbol{q}(\mathrm{D}))$ 的估计。初始时，我们设定 $r_{\mathcal{S}} = 0$，且 $\mathrm{S} = \emptyset$。我们的算法通过迭代运行。令$Q'_i \subseteq \boldsymbol{q}(\mathrm{D})$ 表示第 i次迭代开始时 $\boldsymbol{q}(\mathrm{D})$ 中活跃点的集合。我们不会显式构造这个点集合，而只使用 Q'_i 来使算法更直观。初始时，$Q'_1 = \boldsymbol{q}(\mathrm{D})$。此外，令 $\tau_i = |Q'_i|$，因此初始时 $\tau_1 = |\boldsymbol{q}(\mathrm{D})|$。与 Q'_i 不同， τ_i 将在算法的每一步中显式存储。我们的方法的伪代码见附录C。

- 计算 $\mu\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))$ $(\boldsymbol{q}(\mathrm{D}))$ 的估计值。我们运行定理 5.2 中针对关系型 k-中心问题的算法。令 V 表示点集， L 表示关系型 k-中心聚类的 $(2+\varepsilon)$-近似算法返回的估计成本。我们可以将 ε 设为任意常数值，例如 0.1。根据定义，很容易验证 $\left(\frac{L}{2+\varepsilon}\right)^2 \leq \mu\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))$ $(\boldsymbol{q}(\mathrm{D}))$ 和 $\mu\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))$ $(\boldsymbol{q}(\mathrm{D})) \leq |\boldsymbol{q}(\mathrm{D})| \cdot L^2$。因此，有 $\frac{L^2}{9} \leq \mu\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))$ $(\boldsymbol{q}(\mathrm{D})) \leq |\boldsymbol{q}(\mathrm{D})| \cdot L^2$.

计算完 L 后，算法按迭代运行。在第 i 次迭代中，我们首先检查 $\tau_i \leq 480 \cdot k\log^2(|\boldsymbol{q}(\mathrm{D})|)$ 是否成立。如果成立，则将所有剩余的活跃点加入 S，即 $\mathrm{S} = \mathrm{S} \cup \mathrm{T}$ .report()。

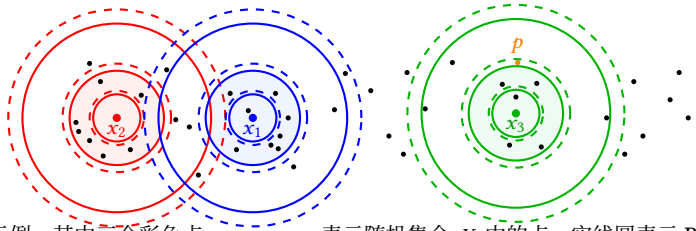


图 6. k-均值算法示例，其中三个彩色点 x_1、x_2、x_3 表示随机集合 X_i 中的点。实线圆表示 $\mathrm{B}(x_\ell, b_j)$，虚线圆表示 $\mathrm{B}(x_\ell, (1+\varepsilon)b_j)$，其中 $\ell \in [3]$、$j \in \{0, 1, 2\}$。如果 $\hat{}\beta_i = 1$，阴影区域中的点属于 Q_i，即它们被视为已覆盖，并变为非活动状态。橙色点 p 可能会变为非活动状态，也可能不会，因为它位于 $\mathrm{B}(x_3, (1+\varepsilon)b_1) \setminus \mathrm{B}(x_3, b_1)$ 中。

• **获取一个包含** $\Theta(k \log^2 |\boldsymbol{q}(\mathrm{D})|)$ 个样本的集合，样本取自 Q_{ni}，**且均匀随机抽取**。在第 i 次迭代中，我们从 Q_{ni} 中均匀随机抽取样本，得到一个包含 X_i 个样本的集合，其中样本数为 $240 \cdot k \log^2 |\boldsymbol{q}(\mathrm{D})|$。具体来说，我们令 $X_i = \mathrm{T}.\mathrm{sample}(240 k \log^2 |\boldsymbol{q}(\mathrm{D})|)$，并更新 $\mathrm{S} = \mathrm{S} \cup X_i$。

• 根据它们到 X_i 的距离对划分 Q'_i。我们隐式地根据点到集合 X_i 的距离，将 Q'_i 中的点划分为若干类别。令 $a_0 = 0, b_0 = {}_{4|\boldsymbol{q}}^{L}(\mathrm{D})|, a_\infty = 2L|\boldsymbol{q}(\mathrm{D})|, b_\infty = \infty$，并且对每个 $j = 1, \ldots, M$，令 $a_j = 2^{j-1}{}_{4|\boldsymbol{q}}^{L}(\mathrm{D})|, b_j = 2^j{}_{4|\boldsymbol{q}}^{L}(\mathrm{D})|$，其中 $M = 2\log(|\boldsymbol{q}(\mathrm{D})|) + 3$。我们构造 T 的一个副本 T′，并对每个 $j = 0, 1, \ldots, M, \infty$ 重复以下步骤。我们定义 $c_{i,j}$，并将 $c_{i,j} = 0$ 初始化。我们逐个遍历点 $x \in X_i$，设置 $c_{i,j} = c_{i,j} + \mathrm{T}'.\mathrm{count}(x, b_j)$，然后调用 $\mathrm{T}'.\mathrm{inactive}(x, b_j)$。

令 $Q'_{i,j} \subseteq Q'_i$ 为在定义 $c_{i,j}$ 之前处于活动状态、并在对所有 $x \in X_i$ 应用 $\mathrm{T}'.\mathrm{inactive}(x, b_j)$ 后变为非活动状态的点集合。虽然算法中没有显式计算 $Q'_{i,j}$，但在分析中，我们证明对于每次迭代 i：

(1) 对于每一对不同的 $j_1, j_2 \in [M] \cup \{0, \infty\}$，$Q'_{i,j_1} \cap Q'_{i,j_2} = \emptyset$，而 $\bigcup_{j \in [M] \cup \{0,\infty\}} Q'_{i,j} = Q'_i$；(2) 对于每个 $j \in [M] \cup \{0, \infty\}$，集合 $Q'_{i,j} \subseteq Q'_i$ 包含 Q'_i 中所有与 X_i 的距离大于 $(1+\varepsilon)a_j$ 且至多为 b_j 的点，可能还包含一些与 X_i 的距离至多为 $(1+\varepsilon)b_j$ 的点，以及一些与 X_i 的距离至少为 a_j 的点；(3) 对于每个 $j \in [M] \cup \{0, \infty\}$，$c_{i,j} = |Q'_{i,j}|$。

• 决定应由 X_i 覆盖 Q'_i 的哪个子集。令 β_i 为满足 $c_{i,\beta_i} > {}_{10}^{\tau_i}\log |\boldsymbol{q}(\mathrm{D})|$ 的最大索引（我们将证明 $\beta_i \le M$ 以高概率存在）。在分析中，我们证明以高概率，将前 $\beta_i + 1$ 个集合中的所有点分配给中心 X_i 所付出的 k-means 成本，$\hat{}Q_i = \bigcup_{\ell \in [\beta_i] \cup \{0\}} Q'_{i,\ell}$，约为最优解分配 Q_i 所付成本的上界，即 $\mu_{X_i}(Q_i) \le O(1) \cdot \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(Q_i)$。我们删除 T′，并使用树 T′ 重复前一条中的过程，直到 $j = \beta_i$。换言之，T 获得 T′ 在定义 c_{i,β_i+1} 之前那一刻的实例。在此版本的 T 中，$\hat{}Q_i$ 中的每个点均为非活动状态。我们在图6中给出了一个示例。我们观察到，在下一次迭代中，$Q'_{i+1} \leftarrow Q'_i \setminus Q_i$。我们更新 $\tau_{i+1} = \tau_i - \sum_{\ell \in [\beta_i] \cup \{0\}} c_{i,\ell}$ 和 $r\mathrm{S} = r\mathrm{S} + \sum_{\ell \in [\beta_i] \cup \{0\}} c_{i,\ell}((1+\varepsilon)b_\ell)^2$。

分析。接下来这些引理的证明见附录C。

引理 6.1。对于每次迭代 i，集合族 $Q'_{i,0}, \ldots, Q'_{i,1}, \ldots, Q'_{i,M}, Q'_{i,\infty}$ 满足第三个项目中的三个性质。

接下来，我们证明：1）算法的迭代次数为 $O(\log N)$，因此 $|\mathrm{S}| = O(k \log^3 N)$；2）对于每次迭代 i，$\mu_{X_i}(\hat{}Q_i) = O\left(\mu_{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\hat{}Q_i)\right)$。对于每次迭代 i，令 $Q_i^{\mathrm{bad}} = \{p \in Q'_i \mid \phi(p, X_i) > 2\phi(p, \mathrm{OPT}(\boldsymbol{q}(\mathrm{D})))\}$。我们证明，$Q_i^{\mathrm{bad}}$ 的大小以高概率很小。

引理 6.2。 对于每次迭代 i, $|Q_i^{\text{bad}}| \le 20 \frac{\tau_i}{\log |\boldsymbol{q}(\mathbf{D})|}$，其概率至少为 $1 - \frac{1}{N^3}$ —.

由于 $|Q_i'| > 480 \cdot k \log^2 |\boldsymbol{q}(\mathbf{D})|$, $M + 2 = 2\log |\boldsymbol{q}(\mathbf{D})| + 5$，并且根据引理 6.1，至少存在一个集合 $Q'_{i,\ j^*}$ 使得 $c_{i,\ j^*} > 10^{\tau_i}$, 这是由鸽巢原理得出的。因此，索引 β_i 定义良好。在下一个引理的证明中，我们还将证明 $\beta_i \neq \infty$。

引理 6.3。 对于每次迭代 i, $|\hat{Q}_i| \ge \frac{\tau_i}{2}$，因此算法执行 $O(\log |\boldsymbol{q}(\mathbf{D})|) = O(\log N)$次迭代，并且 $|S| = O(k \log^3 N)$，其概率至少为 $1 - \frac{1}{N^2}$。 —

下一个引理给出了 S 的 k-均值代价以及算法的运行时间上界。

引理 6.4. 对于任意小的常数 $\varepsilon \in (0, 1)$，有 $\mu_S(\boldsymbol{q}(\mathbf{D})) \le r_S \le (84 + \varepsilon) \cdot \mu_{\text{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$ 成立，概率至少为 $1 - \frac{1}{N}$。该算法以至少 $1 - \frac{1}{N}$ 的概率在 $O(kN \log^5 N + k \log^7 N)$时间内运行。

我们认为，相同的界对关系型 k-中位数聚类同样成立。事实上，关系型 k-中位数问题的常数近似因子更小，因为我们不会将距离平方。我们以下一定理作为结论。

定理 6.5. 给定一个涉及 d 个属性的无环连接查询 $\boldsymbol{q}$、一个大小为 N 的数据库实例 D、一个参数 k，以及一个任意小的常数参数 $\varepsilon \in (0, 1)$，存在一个算法，可计算出一个由 $O(k \log^3 N)$ 个点组成的集合 $S \subseteq \boldsymbol{q}(D)$ 和一个数 r_S，使得 $\mu_S(\boldsymbol{q}(D)) \le r_S \le (84 + \varepsilon) \cdot \mu_{\text{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(D))$，概率至少为 $1 - \frac{1}{N}$。该算法的运行时间为 $O(kN \log^5 N + k \log^7 N)$，概率至少为 $1 - \frac{1}{N}$。对于关系型 k-中位数聚类，存在具有相同理论保证和运行时间的算法。

6.2 更好的近似算法

利用我们在定理 6.5 中的结果，以及来自 [31] 的核心集构造，我们得到了关系型 k-均值聚类问题已知最快的 $(4+\varepsilon)\gamma$-近似，以及关系型 k-中位数聚类问题已知最快的 $(2+\varepsilon)\gamma$-近似。关系型场景中的 RBBD 构造还可以加速 [31] 中的核心集构造。

首先，我们执行定理 6.5 中的算法，得到 S 和 r_S。给定 S r_S，可在$O(kN \log^5 N + k^2 \log^{11} N \cdot \min\{\log^{d+1}(k)\ k \log^5 N\})$ 时间 [31] 内为关系型 k-均值聚类构造一个核心集 C $\subseteq \boldsymbol{q}(\mathbf{D})$，且满足 $|\mathrm{C}| = O(k\varepsilon^{-d} \log^4 N)$。最后，在核心集 C 上执行 γ-近似算法 kMeansAlg，以额外的 $O(T_{\text{均值}\gamma}(k \log^4 N))$ 时间得到一个 $(4+\varepsilon)\gamma$-近似。在附录 C.1 中，我们说明如何利用关系型 RBBD 树构造，将 [31] 中核心集构造的运行时间改进为 $O(kN \log^5 N + k \log^7 N)$ 时间。

定理 6.6. 给定一个无环连接查询 $\boldsymbol{q}$、一个大小为 N 的数据库实例 D、一个参数 k，以及一个任意小的常数参数 $\varepsilon \in (0, 1)$，存在一个算法，可计算出一个由 k 个点组成的集合 $S \subseteq \boldsymbol{q}(D)$，使得 $\mu_S(\boldsymbol{q}(D)) \le (4 + \varepsilon)\gamma \mu_{\text{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(D))$，其概率至少为 $1 - \frac{1}{N}$。该算法的运行时间为 $O(kN \log^5 N + k \log^7 N + T_{\text{mean}\gamma}(k \log^4 N))$，概率至少为$1 - \frac{1}{N}$。对于关系型 k-中位数聚类问题，存在一个近似因子为 $(2 + \varepsilon)\gamma$、且运行时间相同的算法。

7 扩展G

我们的方法相当通用：在标准计算环境中，任何通过 σ 对 BBD 树的查询（利用每个节点中存储的计数与代表点信息）访问输入数据的算法，都可以转换为关系型场景中的随机化算法，并具有相同的近似保证和运行时间 $O(\sigma \cdot N)$（忽略对数因子）。由于许多几何近似算法都属于这一类别，我们的

框架为各类任务产生了高效的关系对应算法。例如，在附录 E 中，我们表明新方法可用于实现 Gonzalez 算法的一个近似版本 [37], ，从而得到针对关系多样性与公平性问题的高效算法。

循环查询。 我们采用分数超树宽度的概念 [38] ，并使用一种标准流程，将我们的算法扩展到每个连接查询 $\boldsymbol{q}$。对于循环查询 $\boldsymbol{q}$，我们将其转换为一个等价的无环查询，使得每个关系都是某个（可能为循环的）连接查询的结果，且其分数边覆盖至多为 $\mathbf{fhw}(\boldsymbol{q})$。我们对这些（可能为循环的）查询求值，以获得新的关系，然后将我们的算法应用于该无环查询。除运行时间外，所有保证均与定理 5.2、6.6、D.1 相同；对于运行时间，我们只需将任何 N 因子替换为N^{fhw}。由于这是数据库理论中的典型方法，我们在附录 F 中给出详细说明。

8 未来工作

这项工作为未来研究开辟了多个方向。一个核心的开放问题是，我们用于 k-中心/中位数/均值聚类的 $\mathcal{O}(kN)$ 时间算法是否最优。尽管我们的方法优于当前最先进水平，但研究其他几何技术˜能否加以改造，从而在关系型场景中实现 $O(N)$ 时间算法，也很有意义。我们还希望探究，标准计算环境中针对 k-中心聚类的已知下界 [17, 33]能否推广到关系型场景，从而有望确立近乎最优的结果。

参考文献

[1] https://db-engines.com/en/ranking_类别。[2] Yelp 数据集挑战赛。https://www.yelp.com/dataset/challenge，2017。访问日期：2025-02-07。[3] Kaggle 机器学习与数据科学调查。https://www.kaggle.com/datasets/kaggle/kaggle-survey-2018，2018。[在线；访问日期：14-May-2024]。[4] Mahmoud Abo-Khamis、Sungjin Im、Benjamin Moseley、Kirk Pruhs 和 Alireza Samadian。用于支持向量机训练的关系梯度下降算法。载于计算机系统算法原理研讨会（*APOCS*），第 100–113 页。SIAM，2021。[5] Mahmoud Abo Khamis、Hung Q Ngo、XuanLong Nguyen、Dan Olteanu 和 Maximilian Schleich。利用稀疏张量进行数据库内学习。载于第 *37* 届 *ACM SIGMOD-SIGACT-SIGAI* 数据库系统原理研讨会论文集，第 325–340 页，2018。[6] Raghavendra Addanki、Andrew McGregor、Alexandra Meliou 和 Zafeiria Moumoulidou。公平最大-最小多样化的近似性与可扩展性改进。载于第 *25* 届国际数据库理论会议，2022。[7] Pankaj K Agarwal、Aryan Esmailpour、Xiao Hu、Stavros Sintos 和 Jun Yang。计算合取查询结果的良好代表性摘要。*ACM* 数据管理论文集，2(5):1–27，2024。[8] Ankit Aggarwal、Amit Deshpande 和 Ravi Kannan。用于 k-均值聚类的自适应采样。载于组合优化近似算法国际研讨会，第 15–28 页。Springer，2009。[9] Rakesh Agrawal、Tomasz Imieliński 和 Arun Swami。在大型数据库中挖掘商品集合之间的关联规则。*ACM SIGMOD* 记录，22(2):207–216，1993。[10] Marcelo Arenas、Timo Camillo Merkl、Reinhard Pichler 和 Cristian Riveros。迈向查询答案多样性的可处理性：超度量来助力。*ACM* 数据管理论文集，2(5):1–26，2024。[11] David Arthur 和 Sergei Vassilvitskii。k-均值：++ 精心选择初始中心的优势。载于第十八届 *ACM-SIAM* 离散算法年度研讨会论文集，第 1027–1035 页，2007。[12] Sunil Arya 和 David M Mount。近似范围搜索。计算几何，17(3-4):135–152，2000。[13] Sunil Arya、David M Mount、Nathan S Netanyahu、Ruth Silverman 和 Angela Y Wu。用于固定维度近似最近邻搜索的最优算法。*ACM*学报（*JACM*），45(6):891–923，1998。[14] Vijay Arya、Naveen Garg、Rohit Khandekar、Adam Meyerson、Kamesh Munagala 和 Vinayaka Pandit。用于 *k*-中位数和设施选址问题的局部搜索启发式方法。*SIAM*计算学报，33(3):544–562，2004。[15] Albert Atserias、Martin Grohe 和 Dániel Marx。关系连接的规模界与查询计划。*SIAM*计算学报，42(4):1737–1767，2013。[16] MohammadH. Bateni 等。极端 *k*-中心聚类。载于*AAAI* 人工智能会议论文集，第 35 卷，第 3941–3949 页。AAAI Press，2021。[17] MohammadHossein Bateni、Hossein Esfandiari、Hendrik Fichtenberger、Monika Henzinger、Rajesh Jayaram、Vahab Mirrokni 和 Andreas Wiese。面向自适应对手和无知对手的最优全动态 k-中心聚类。载于*2023* 年 *ACM-SIAM* 离散算法年度研讨会论文集（*SODA*），第 2677–2727 页。SIAM，2023。

[18] Chazelle Bernard。《差异方法：随机性与复杂性》，2001。[19] Paul B Callahan 和 S Rao Kosaraju。多维点集的一种分解及其在 k-最近邻和 n 体势场中的应用。ACM学报（JACM），42(1):67–90，1995。[20] Nofar Carmeli、Nikolaos Tziavelis、Wolfgang Gatterbauer、Benny Kimelfeld 和 Mirek Riedewald。用于直接访问合取查询排序答案的可处理排序。ACM数据库系统汇刊，48(1):1–45，2023。[21] Timothy M Chan。最优划分树。载于第二十六届计算几何年度研讨会论文集，第 1–10 页，2010。[22] Jiaxiang Chen、Qingyuan Yang、Ruomin Huang 和 Hu Ding。关系数据的核心集及其应用。载于《神经信息处理系统进展》，35:434–448，2022。[23] Zhaoyue Cheng 和 Nick Koudas。规范化数据上的非线性模型。载于 2019 年 IEEE 第 35 届国际数据工程会议（ICDE），第 1574–1577 页。IEEE，2019。[24] Zhaoyue Cheng、Nick Koudas、Zhe Zhang 和 Xiaohui Yu。在规范化数据上高效构造非线性模型。载于 2021 年 IEEE 第 37 届国际数据工程会议（ICDE），第 1140–1151 页。IEEE，2021。[25] Sam Coy、Artur Czumaj 和 Gopinath Mishra。论并行 k-中心聚类。载于第 35 届 ACM 算法与架构并行性研讨会论文集，第 65–75 页，2023。[26] Ryan Curtin、Benjamin Moseley、Hung Ngo、XuanLong Nguyen、Dan Olteanu 和 Maximilian Schleich。Rk-means：关系数据的快速聚类。载于国际人工智能与统计会议，第 2742–2752 页。PMLR，2020。[27] Mark de Berg、Otfried Cheong、Marc J. van Kreveld 和 Mark H. Overmars。《计算几何：算法与应用》。Springer Berlin Heidelberg，德国柏林，第 3 版，2008。doi:10.1007/978-3-540-77974-2。[28] Shaleen Deep、Xiao Hu 和 Paraschos Koutris。带投影的连接查询排序枚举。超大型数据库基金会论文集，15(5):1024–1037，2022。[29] Shaleen Deep 和 Paraschos Koutris。合取查询结果的排序枚举。载于第 24 届国际数据库理论会议，2021。[30] Alina Ene、Sungjin Im 和 Benjamin Moseley。使用 MapReduce 进行快速聚类。载于第 17 届 ACM SIGKDD 国际知识发现与数据挖掘会议论文集，第 681–689 页，2011。[31] Aryan Esmailpour 和 Stavros Sintos。关系聚类的改进近似算法。ACM 数据管理论文集，2(5):1–27，2024。[32] Artur Czumaj 等。用于高维聚类的完全可扩展 MPC 算法。载于第 51 届国际自动机、语言与程序设计研讨会（ICALP）论文集，第 50:1–50:20 页。莱布尼茨信息学国际论文集（LIPIcs），2024。[33] Tomás Feder 和 Daniel Greene。近似聚类的最优算法。载于第二十届 ACM 计算理论研讨会论文集，第 434–444 页，1988。[34] Arnold Filtser、Shaofeng H.-C. Jiang、Yi Li、Anurag Murty Naredla、Ioannis Psarros、Qiaoyuan Yang 和 Qin Zhang。通过数据缩减实现更快的 *k*-中心近似算法。载于 2025 年国际机器学习会议（ICML）论文集。PMLR，2025。[35] Raphael A Finkel 和 Jon Louis Bentley。四叉树：一种用于复合键检索的数据结构。Acta informatica，4:1–9，1974。[36] Rudolf Fleischer、Mordecai J Golin 和 Yan Zhang。在线维护直线上的 k-中位数和 k-覆盖。Algorithmica，45(4):549–567，2006。[37] Teofilo F Gonzalez。使簇间最大距离最小化的聚类。Theoretical computer science，38 306，1985 293: – 。[38] G. Gottlob、G. Greco 和 F. Scarcello。树宽与超树宽。《可处理性：解决难题的实用方法》，1，2014。[39] Allan Grønlund、Kasper Green Larsen、Alexander Mathiasen、Jesper Sindahl Nielsen、Stefan Schneider 和 Mingzhou Song。一维快速精确 k-均值、k-中位数和 Bregman 散度聚类。arXiv 预印本 arXiv:1701.07204，2017。[40] Jiawei Han 和 Micheline Kamber。《数据挖掘：概念与技术》。Morgan Kaufmann，2000。[41] Sariel Har-Peled。《几何近似算法》。第 173 号。美国数学学会，2011。[42] Sariel Har-Peled 和 Soham Mazumdar。论 k-均值和 k-中位数聚类的核心集。载于第三十六届 ACM 计算理论研讨会论文集，第 291–300 页，2004。[43] Sariel Har-Peled 和 Manor Mendel。低维度量空间中网的快速构造及其应用。载于第二十一届计算几何年度研讨会论文集，第 150–158 页，2005。[44] Piotr Indyk、Sepideh Mahabadi、Mohammad Mahdian 和 Vahab S Mirrokni。用于多样性最大化和覆盖性最大化的可组合核心集。载于第 33 届 ACM SIGMOD-SIGACT-SIGART 数据库系统原理研讨会论文集，第 100–108 页，2014。[45] Herve Jegou、Matthijs Douze 和 Cordelia Schmid。用于最近邻搜索的乘积量化。IEEE 模式分析与机器智能汇刊，33(1):117–128，2010。

[46] Mahmoud Abo Khamis、Ryan R Curtin、Benjamin Moseley、Hung Q Ngo、XuanLong Nguyen、Dan Olteanu 和 Maximilian Schleich。带加性不等式的函数聚合查询。ACM数据库系统汇刊（TODS），45(4)：1–41，2020。[47] Mahmoud Abo Khamis、Hung Q Ngo、XuanLong Nguyen、Dan Olteanu 和 Maximilian Schleich。Ac/dc：数据库内学习惊雷。在第二届端到端机器学习数据管理研讨会论文集，第 1–10 页，2018。[48] Arun Kumar、Jeffrey Naughton 和 Jignesh M Patel。在规范化数据上学习广义线性模型。在 2015 年 ACM SIGMOD数据管理国际会议论文集，第 1969–1984 页，2015。[49] Yash Kurkure、Miles Shamo、Joseph Wiseman、Sainyam Galhotra 和 Stavros Sintos。用于公平最大-最小多样化的更快算法。ACM 数据管理论文集，2(3)：1–26，2024。[50] Jure Leskovec、Anand Rajaraman 和 Jeffrey Ullman。海量数据集挖掘。剑桥大学出版社，2014。[51] Stuart Lloyd。脉冲编码调制中的最小二乘量化。IEEE 信息论汇刊，28(2)：129–137，1982。[52] Sepideh Mahabadi 和 Stojan Trajanovski。公平多样的数据摘要的核心集。神经信息处理系统进展，36：78987–79011，2023。[53] Timo Camillo Merkl、Reinhard Pichler 和 Sebastian Skritek。合取查询答案的多样性。在第 26 届国际数据库理论会议（ICDT 2023）。Schloss Dagstuhl-Leibniz 信息学中心，2023。[54] Benjamin Moseley、Kirk Pruhs、Alireza Samadian 和 Yuyan Wang。用于 k-均值聚类的关系算法。在第 48 届自动机、语言与编程国际学术会议（ICALP），第 198 卷，第 97:1–97:21 页，2021。[55] Zafeiria Moumoulidou、Andrew McGregor 和 Alexandra Meliou。公平性约束下的多样数据选择。在第 24 届国际数据库理论会议，2021。[56] Hung Q Ngo、Ely Porat、Christopher Ré 和 Atri Rudra。最坏情况最优连接算法。ACM学报（JACM），65(3)：1–40，2018。[57] Hung Q Ngo、Christopher Ré 和 Atri Rudra。倾斜卷土重来：连接算法理论的新进展。ACM SIGMOD 记录，42(4)：5–16，2014。[58] Rafail Ostrovsky、Yuval Rabani、Leonard J. Schulman 和 Chaitanya Swamy。Lloyd型方法对 *k*-均值问题的有效性。ACM学报，59(6)：1–22，2012。[59] Sekharipuram S Ravi、Daniel J Rosenkrantz 和 Giri Kumar Tayi。离散化问题的启发式算法和特殊情形算法。《运筹学》，42(2)：299–310，1994。[60] Steffen Rendle。将因子分解机扩展到关系数据。超大型数据库论文集，6 348，2013 5 337( )：–。[61] 报告。关系数据库管理系统市场：行业分析与 2021–2027 年预测：按类型、部署方式、终端用户和地区划分，2022。[62] Jeffrey S Salowe。通过参数搜索进行 L-无穷范数距离选择。《信息处理快报》，30 14，1989 1 9( )：–。[63] Maximilian Schleich、Dan Olteanu、Mahmoud Abo-Khamis、Hung Q Ngo 和 XuanLong Nguyen。关系数据上的模型学习：简明教程。在《可扩展不确定性管理：第 13 届国际会议，SUM 2019，法国贡比涅，2019 年 12 月 16–18 日，论文集 13 》中，第 423–432 页。Springer，2019。[64] Maximilian Schleich 、 Dan Olteanu 和 Radu Ciucanu。在因子化连接上学习线性回归模型。在 2016 年国际数据管理会议论文集，第 3–18 页，2016。[65] Vaishali Surianarayanan、Neeraj Kumar 和 Stavros Sintos。带集合离群点的聚类及其在关系聚类中的应用。ACM 数据管理论文集，3(5)，2025。doi:10.1145/3767712。[66] Arie Tamir。图上的厌恶型设施选址。SIAM 离散数学杂志，4(4)：550–567，1991。[67] Nikolaos Tziavelis、Deepak Ajwani、Wolfgang Gatterbauer、Mirek Riedewald 和 Xiaofeng Yang。完整合取查询答案排序枚举的最优算法。在超大型数据库论文集。国际超大型数据库会议，第 13 卷，第 1582 页。NIH 公共访问，2020。[68] Nikolaos Tziavelis、Nofar Carmeli、Wolfgang Gatterbauer、Benny Kimelfeld 和 Mirek Riedewald。连接上的分位数高效计算。在第 42 届 ACM SIGMOD-SIGACT-SIGAI 数据库系统原理研讨会论文集，第 303–315 页，2023。[69] Xiaolin Wu。通过矩阵搜索实现最优量化。算法杂志，12(4)：663–673，1991。[70] Keyu Yang、Yunjun Gao、Lei Liang、Bin Yao、Shiting Wen 和 Gang Chen。在规范化数据上使用高斯核实现因子化 SVM。在 2020 年 IEEE 第 36 届数据工程国际会议（ICDE）上，第 1453–1464 页。IEEE，2020。[71] Mihalis Yannakakis。无环数据库模式的算法。在超大型数据库，第 81 卷，第 82–94 页，1981。

A 第 4 节缺失的细节

A.1 当 $\square_u$ 是带孔盒时执行质心收缩

首先，我们描述标准 BBD 树中的过程。最初 $\rho = \square_u^+$。他们计算出包含 $\rho \cap P_u$ 中所有点以及内盒 $\square_u^-$ 的最小中点盒。令 $\rho' \subseteq \rho$ 为该盒。他们对 ρ' 应用 $O(d)$ 公平划分，直到找到非平凡公平划分，即将 $\rho' \cap P_u$ 中的点划分为两个非空子集的公平划分。令 $\hat{\rho} \subset \rho$ 为非平凡公平划分所创建的两个盒之一，使得 $|\hat{\rho} \cap P_u| \geq \frac{|\rho' \cap P_u|}{2}$。关键观察是，$|\hat{\rho} \cap P_u| \leq |\rho \cap P_u| - 1$。如果 $\hat{\rho}$ 包含内盒 $\square_u^-$，则检查以下条件。如果 $|\hat{\rho} \cap P_u| > \frac{2}{3}|P_u|$，则将 $\rho = \hat{\rho}$，并继续递归。如果 $|\hat{\rho} \cap P_u| \leq \frac{2}{3}|P_u|$，则构造 u 的两个子节点 w，v，使得 $\square_w = \rho' \setminus \hat{\rho}$ 是一个盒（不带孔），而 $\square_v$ 是一个带孔盒，其中 $\square_v^+ = \hat{\rho}$ 和 $\square_v^- = \square_u^-$。

否则，令 $\hat{\rho}$ 为递归过程中找到的第一个不包含内盒 $\square_u^-$ 的盒。接着，它们创建 u 的两个子节点 w、v，使得 $\square_w$ 是一个带孔盒，其包含 $\square_w^+ = \rho$ 和 $\square_w^- = \rho'$；而 $\square_v$ 是一个带孔盒，其包含 $\square_v^+ = \rho'$ 和 $\square_v^- = \square_u^-$。此外，它们还创建 v 的两个子节点 v_1、v_2。$\square_{v_2}$ 是一个带孔盒，包含 $\square_{v_2}^+ = \rho' \setminus \hat{\rho}$ 和 $\square_{v_2}^- = \square_u^-$；而 $\square_{v_1} = \hat{\rho}$ 是一个（不带孔的）盒。递归过程继续进行，并设置 $\rho = \hat{\rho}$。注意，$\rho = \hat{\rho}$ 不包含任何孔。它们继续采用递归方法，找到包含 $P_u \cap \rho$ 的最小中点盒 ρ'，并对 ρ' 进行公平划分，直到找到满足 $\frac{1}{3}|P_u| \leq |\hat{\rho} \cap P_u| \leq \frac{2}{3}|P_u|$ 的 $\hat{\rho}$。最后，它们创建 v_1 的两个子节点 v_{1a}、v_{1b}，使得 $\square_{v_{1b}}$ 是一个带孔盒，包含 $\square_{v_{1b}}^+ = \rho'$ 和 $\square_{v_{1b}}^- = \hat{\rho}$；而 $\square_{1a} = \hat{\rho}$ 是一个盒。总的来说，质心收缩可能会创建两个新节点 v、w，或六个新节点 w、v、v_1、v_{1a}、v_{1b}、v_2。如 [13], 所示，如果我们知道 ρ 的角点和 $\square_u^-$ 的角点，就可以在 $O(1)$ 时间内找到包含 $P_u \cap \rho$ 中所有点的最小中点盒 $\rho' \subseteq \rho$。由于在每次递归迭代中 $|\hat{\rho} \cap P_u| \leq |\rho' \cap P_u| - 1$，质心过程会在 $O(|P_u|)$ 时间内执行。

对于 RBBD 树，我们在 $\mathbf{N}_u$ 上执行相同的操作，而不是在 P_u 上。第一次找到一个不包含 $\square_u^-$ 的盒 $\hat{\rho}$（在非平凡公平划分之后）时，我们按照上述说明，用六个新节点构造子树。

A.2 定理4.1的证明

我们通过证明以下引理来证明定理4.1。

引理A.1。 在一个节点 u 上执行随机化质心收缩过程，可计算出一个中点盒 $\hat{\rho}$，使得 $\frac{1}{3} - \varepsilon \leq \frac{|\hat{\rho} \cap P_u|}{|P_u|} \leq \frac{2}{3} + \varepsilon$，其概率至少为 $1 - \Omega\left(\frac{1}{n^{\zeta}}\right)$。

证明。随机化质心收缩中的递归方法与标准质心收缩相似，但它不是在 P_u 上执行，而是在 $\mathbf{N}_u$ 中执行。因此，随机化质心收缩过程将在 $O(\varepsilon^{-2}\log n^{\zeta}) = O(\varepsilon^{-2}\log n^{\zeta})$ 次递归调用后结束。算法在首次找到满足 $|\hat{\rho} \cap \mathbf{N}_u| \leq \frac{2}{3}|\mathbf{N}_u|$ 的多数中点盒 $\hat{\rho}$ 时停止。根据定义，集合 $\mathbf{N}_u$ 是一个 ε-采样，其概率至少为 $1 - \Omega\left(\frac{1}{n^{\zeta}}\right)$。因此，$\frac{|\hat{\rho} \cap P_u|}{|P_u|} \leq \frac{|\hat{\rho} \cap \mathbf{N}_u|}{|\mathbf{N}_u|} + \varepsilon \leq \frac{2}{3} + \varepsilon$。还需证明不等式的另一方向。令 ρ' 为我们执行最后一次公平划分以找到 $\hat{\rho}$ 时的中点盒。根据定义，有 $|\mathbf{N}_{u \cap \rho'}| \geq \frac{2}{3}|\mathbf{N}_u|$。注意，根据 $\hat{\rho}$ 的定义，有 $|\mathbf{N}_u \cap \hat{\rho}| \geq \frac{1}{2}|\mathbf{N}_{u \cap \rho'}|$。因此，$|\mathbf{N}_u \cap \rho| \geq \frac{1}{2} \cdot |\mathbf{N}_{u \cap \rho'}| \geq \frac{1}{3}|\mathbf{N}_u|$。由于 $\mathbf{N}_u$ 是一个 ε-采样，其概率至少为 $1 - \Omega\left(\frac{1}{n^{\zeta}}\right)$，我们得出 $\frac{|\hat{\rho} \cap P_u|}{|P_u|} \geq \frac{|\hat{\rho} \cap \mathbf{N}_u|}{|\mathbf{N}_u|} - \varepsilon \geq \frac{1}{3} - \varepsilon$。引理得证。 □

引理 A.2. 高概率下，*RBBD* 树与 *BBD* 树具有相同的性质和渐近查询复杂度。*RBBD* 树的高度为 $O(\log n)$，节点数为$O(n)$，查询复杂度为 $O(\log n + \varepsilon^{-d+1})$，其概率至少为 $1 - \frac{1}{n^{\zeta-1}}$。

证明。RBBD 树的构造与 BBD 树类似，区别在于质心收缩。采用随机质心收缩，对于节点 u，我们找到一个盒 $\hat{\rho}$，使得 $\frac{1}{3} - \varepsilon \leq \frac{|\hat{\rho} \cap P_u|}{|P_u|} \leq \frac{2}{3} + \varepsilon$，其概率至少为 $1 - \Omega\left(\frac{1}{n^{\zeta}}\right)$。

暂且假设随机质心收缩的每次执行都满足引理A.1中的不等式。接下来，我们将证明，在此假设下，对于足够小的常数 ε，BBD 树与 RBBD 树在渐近意义上没有区别。在 [13] 中，作者证明了：在 BBD 树中，每向下遍历 4 层，与节点关联的点数至少减少到原来的$\frac{2}{3}$倍，因此树的高度为 $O(\log n)$，且 BBD 树有$O(n)$个节点。类似地，在我们的情形中，注意到每次随机质心收缩都会在树中引入 3 个新层，并且我们将其与公平划分交替进行，因此每经过 4 层，点数至少会减少到原来的 $\frac{2}{3} - \varepsilon$倍。对于较小的 ε，例如 $\varepsilon < 0.1$，根据相同的论证，树的高度为 $O(\log n)$。等价地，我们也可以这样论证：在 RBBD 树中，每向下遍历 8 层，与节点关联的点数至少减少到原来的$\frac{2}{3}$倍，树的高度为 $O(\log n)$，节点数为 $O(n)$。接下来，[13]中的作者论证了：在 BBD 树中，每向下遍历 $4d$ 层，关联单元的大小至少减少到原来的$\frac{2}{3}$倍。类似地，在我们的情形中，注意到要缩小单元的大小，就必须沿 d 个维度逐一缩小其大小。由于树中至少每隔 4 层就会进行一次公平划分，而且每次划分都会使最长边至少缩短到原来的 2/3，因此至多经过 d 次划分（即树的至多 $4d$ 层）后，大小就会缩小到这一倍数。只要假设所有随机质心收缩算子都正确执行，也就是说，假设引理A.1以概率 1 成立，那么 BBD 树的所有证明 [13] 也都适用于 RBBD 树。鉴于引理A.1中的不等式成立的概率至少为$1 - \Omega\left(\frac{1}{n^{\zeta}}\right)$，并对 $O(n)$个节点应用联合界，可知所有随机质心收缩算子都正确执行的概率至少为 $1 - \frac{1}{n^{\zeta-1}}$。因此，我们得出结论：BBD 树的所有证明和性质在渐近意义上也都以至少 $1 - \frac{1}{n^{\zeta-1}}$的概率适用于 RBBD 树。

最后，BBD 树与 RBBD 树之间的查询过程完全相同。给定查询球 $\mathrm{B}(p, r)$，我们从树的根节点开始。设 u 为查询过程中处理的 RBBD 树节点。我们检查 $\mathrm{B}(p, r) \cap \square_u = \emptyset$。如果是，则停止沿着树的该分支继续执行。如果 $\square_u \subseteq \mathrm{B}(p, (1+\varepsilon)r)$，我们就将 u 添加到规范节点集合 $\mathrm{U}(p, r)$ 中。在其他情况下，我们继续搜索 u 的两个子节点。最后，RBBD 树总能计算出一个规范节点集合 U，使得 $\mathrm{B}(p, r) \cap P \subseteq P \cap \bigcup_{u \in \mathrm{U}} \square_u \subseteq \mathrm{B}(p, (1+\varepsilon)r) \cap P$。由于 RBBD 树的高度以高概率为 $O(\log n)$，根据上述讨论，来自 [12, 13] 的查询分析也适用于 RBBD 树，从而证明 $\mathrm{U}(p, r)$ 可在 $O(\varepsilon^{-d+1} + \log n)$ 时间内计算，并且 $|\mathrm{U}(p, r)| = O(\varepsilon^{-d+1} + \log n)$，其概率至少为 $1 - \frac{1}{n^{\zeta-1}}$。

□

引理 A.3。 假设我们能够在$f(n, \varepsilon)$ 时间内，从盒或带孔盒中获取随机样本集$\mathcal{N}_u$，其中 f是依赖于 n 和 ε的正实值函数，那么以至少 $1 - \frac{1}{n^{\zeta-1}}$的概率，*RBBD* 树可以在 $O\left(n \log(n) + n \cdot \left(f(n, \varepsilon) + \varepsilon^{-2} \log^2 n\right)\right)$时间内构建完成。

证明。由于公平划分过程与标准 BBD 树中的过程相同，利用标准 BBD 树中的方法，所有公平划分过程均可在 $O(n \log n)$ 时间内执行。接下来，我们关注随机质心收缩过程。对于一个节点 u，我们在 $f(n, \varepsilon)$ 时间内得到集合 N_u。由于我们明确知道 N_u 中的所有点以及 $\square_u^+$的角点，因此可以按照 [13]中的描述，使用每个坐标上的最大值和最小值，在 $O(\varepsilon^{-2} \log n)$ 时间内找到包含 N_u和内盒 $\square_u^-$的最小中点盒 ρ'。计算出 ρ'后，我们执行 $O(1)$ 次公平划分，以找到第一个非平凡公平划分。每次公平划分中的超平面都可以在 $O(1)$ 时间内直接计算。利用点的每个坐标上的简单二叉搜索树

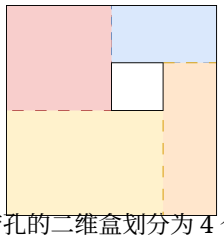


图 7. 将一个带孔的二维盒划分为 4 个盒。

对于 N_u，我们可以在 $O(\log(\varepsilon^{-2}\log n))$ 时间内检查公平划分是否平凡。找到第一个非平凡公平划分后，我们通过检查 $|\hat{\rho}\cap N_u| > |(\rho'\setminus\hat{\rho})\cap N_u|$，使用与最后一个非平凡公平划分对应的搜索二叉树，在 $O(\log(\varepsilon^{-2}\log n)) = O(\log n)$ 时间内计算 $\hat{\rho}$。每次非平凡划分之后，盒 $\hat{\rho}$ 至多包含来自 N_u 的 $|\rho'\cap N_u|-1$ 个样本。因此，这种递归方法最多执行 $O(|N_u|) = O(\varepsilon^{-2}\log n)$ 次。由此，每次随机质心收缩的运行时间为 $O(f(n,\varepsilon) + \varepsilon^{-2}\log^2 n)$。根据引理 A.2，RBBD 树的高度为 $O(\log n)$，且以至少 $1-\frac{1}{n^{\zeta-1}}$ 的概率有 $O(n)$ 个节点，因此所有随机质心收缩过程的运行时间为 $O(n(f(n,\varepsilon) + \varepsilon^{-2}\log^2 n))$，概率至少为 $1-\frac{1}{n^{\zeta-1}}$。□

A.3 从带孔盒中采样

我们考虑更复杂的情况：u 是一个节点，且 $\square_u$ 是一个带孔盒。目标是从 $\square_u\cap \boldsymbol{q}(D)$ 中均匀随机采样 $\Lambda = \Theta(\varepsilon^{-2}\log(|\boldsymbol{q}(D)|^2)) = \Theta(\varepsilon^{-2}\log N)$ 个点。使用 Yannakakis 算法 [71]，可以在我们的方法开始时以 $O(N)$ 时间计算 $|\boldsymbol{q}(D)|$。根据引理 2.1，我们知道如何使用过程 SampleRect 高效地从一个盒中采样 z 个元组。然而，$\square_u$ 是一个带孔盒，因此我们应从两个盒的差集中采样，即 $\square_u^+\setminus\square_u^-$。我们利用 $\square_u^-$ 的角点将 $\square_u$ 划分为 $2d$ 个互不相交的盒。二维示例见图 7。令 $R = \{\rho_1,\ldots,\rho_{2d}\}$ 为将 $\square_u$ 划分为 $2d$ 个互不相交盒的集合。首先，我们决定从 R 中的每个盒抽取多少样本（总共 Λ 个样本），然后调用函数 SampleRect，从每个盒中获取相应数量的样本。我们对每个 $i\in[2d]$ 计算 $C_i = \text{CountRect}(\boldsymbol{q}, D, \rho_i)$。根据这些值 $C_1,\ldots,C_{2d}$，对于每个 $j\in[\Lambda]$，我们以概率 $\frac{C_i}{\sum_{\ell\in[2d]} C_\ell}$ 从 R 中选择一个盒，作为抽取样本点的来源，即选择一个盒 $\rho_i\in R$。令 s_i 为我们应从 ρ_i 获取的样本数。然后，对于每个 $i\in[2d]$，我们运行 $S_i = \text{SampleRect}(\boldsymbol{q}, D, \rho_i, s_i)$。令 $N_u = \bigcup_{i\in[2d]} S_i$。我们声称，$N_u$ 是针对 $\boldsymbol{q}(D)\cap\square_u$ 中的点的一个 ε-样本。只需证明 N_u 是从 $\square_u\cap\boldsymbol{q}(D)$ 中有放回均匀随机采样得到的 Λ 集合即可。等价地，我们证明抽取一个样本时，选中特定元组 $t\in\square_u\cap\boldsymbol{q}(D)$ 的概率为 $\frac{1}{|\boldsymbol{q}(D)\cap\square_u|}$。令 t 为任意一个（不失一般性地）属于 ρ_i 的元组。从 $\square_u\cap\boldsymbol{q}(D)$ 中抽取一个样本时选中 t 的概率为 $\frac{C_i}{\sum_{\ell\in[2d]C_\ell}}\cdot\frac{1}{C_i} = \frac{1}{\sum_{\ell\in[2d]}C_\ell} = \frac{1}{|\boldsymbol{q}(D)\cap\square_u|}$。因此，$N_u$ 是从 $\square_u\cap\boldsymbol{q}(D)$ 中有放回均匀采样得到的 Λ 集合，并且 N_u 以高概率是一个 ε-样本。计算出 N_u 后，我们运行递归方法，寻找包含 N_u 和内盒 $\square_u^-$ 的最小中点盒，并应用 $O(d)$ 次公平划分，直至找到第一个非平凡公平划分。该方法只需使用 N_u、$\square_u^+$ 和 $\square_u^-$ 即可实现，因此其实现方式与标准计算环境中的相同。最后，我们分析运行时间。CountRect 过程的运行时间为 $O(N)$，因此所有数值 $C_1,\ldots,C_{2d}$ 均可在 $O(2dN) = O(N)$ 时间内计算。然后，每次运行 $\text{SampleRect}(\boldsymbol{q}, D, \rho_i, s_i)$ 所需时间为 $O(N + C_i\log N)$。因此，集合 N_u 可在 $O(N+\varepsilon^{-2}\log^2 N)$ 时间内计算。根据引理 4.2，计算出 N_u 后，随机质心收缩的后续步骤可在 $O(\varepsilon^{-2}\log^2 N)$ 时间内完成。总之，关系型场景中的随机质心收缩运行时间为 $O(N+\varepsilon^{-2}\log^2 N)$。

A.4 带孔盒中的计数与代表点

我们假设 $\square_u$ 是一个带孔盒。按照随机质心收缩中的做法，我们构造划分 $\mathrm{R} = \{\rho_1, \ldots\ldots, \rho_{2d}\}$。对于每个 $\rho_i \in \mathrm{R}$，我们计算 $C_i = \mathrm{CountRect}(\boldsymbol{q}, \mathrm{D}, \rho_i)$，并令 $u.c = \sum_{\ell \in [2d]} C_\ell$。类似地，我们通过执行 $\mathrm{ReprRect}(\boldsymbol{q}, \mathrm{D}, \rho_i)$来计算代表元 $u.rep$，直到在集合 R 中找到一个非空盒（若存在）。根据引理 2.1以及 $2d = O(1)$这一事实，节点 u 的 $u.c$ 和 $u.rep$ 均可在$O(N)$时间内计算。

A.5 第 4.4 节中引理的证明

引理 4.3 的证明。若 T 中所有节点均为活跃节点，那么我们将得到一组规范节点 $\mathrm{U}'(x, r)$，使得 $B(x, r) \cap \boldsymbol{q}(\mathrm{D}) \subseteq \bigcup_{u \in \mathrm{U}'(x,r)} (\square_u \cap \boldsymbol{q}(\mathrm{D})) \subseteq B(x, (1+\varepsilon)r) \cap \boldsymbol{q}(\mathrm{D})$。由于存在非活动节点，T 上的查询过程会跳过树中已经处于非活动状态的分支。因此 $\mathrm{U}(x, r) \subseteq \mathrm{U}'(x, r)$。设 p为一个非活跃点，即 $p \in \boldsymbol{q}(\mathrm{D}) \setminus Q'$。点 p将保持非活动状态，因为我们的过程只会使新节点变为非活动状态（不会使任何节点变为活动状态）。接下来，假设存在一个活跃点 $p \in Q'$，使得 $\phi(p, x) > (1+\varepsilon)r$。根据定义，不存在节点$u \in \mathrm{U}(x, r)$ 使得 $p \in \square_u$，因此 p将保持活跃状态。最后，设 $p \in Q'$为一个活跃点，使得 $\phi(p, x) < r$。那么将存在一个节点 $u' \in \mathrm{U}'(x, r)$，使得 $p \in \square_{u'}$。由于 p处于活跃状态，因此不存在节点 $v \in \mathrm{T}$ 使得 $v.a = 0$ 且 $p \in \square_v$。节点 u' 变为非活动状态，因此 p 将处于非活动状态。由此可得 $B(x, r) \cap Q' \subseteq \bigcup_{u \in \mathrm{U}(x,r)} (\square_u \cap Q') \subseteq B(x, (1+\varepsilon)r) \cap Q'$。

接下来，我们证明更新操作后，$u.rep$和 $u.c$的值是正确的。证明采用归纳法。设 u是一个节点，v，w是它的子节点。如果 $v.a = w.a$，那么我们的算法会正确地设置 $u.a = 0$，因为 $\square_u = \square_v \cup \square_w$。如果 $v.a = 1$，$w.a = 0$ 且 $v.rep$，$v.c$是正确的，那么我们的算法会正确地设置 $u.rep = v.rep$和 $u.c = v.c$，因为 $\square_u \cap Q' = \square_v \cap Q'$。如果 $v.a = 0$ 且 $w.a = 1$，则可用同样的方式论证。最后，如果 $v.a = w.a = 1$，那么若 v，w中的计数和代表点正确，则 $v.rep \in \square_u \cap Q'$确实成立；根据定义，$|\square_u \cap Q'| = |\square_v \cap Q'| + |\square_w \cap Q'| = v.c + w.c$。

如引理 4.2 所示，每个新节点都能在 $O(N + \varepsilon^{-2}\log^2 N)$ 时间内构造完成。预言机在 RBBD 树上执行查询，以计算规范节点 $\mathrm{U}(x, r)$。由于实现随机质心收缩所需的样本数为 $O(\varepsilon^{-2}\log N^{m+4})$，因此 RBBD 树中的查询过程以至少 $1 - \frac{1}{N^{m+3}}$ 的概率具有与 BBD 树中的查询过程相同的性质（如引理 A.2 所示）。因此，$|\mathrm{U}(x, r)| = O(\varepsilon^{-d+1} + \log N)$，且 $\mathrm{U}(x, r)$ 以至少 $1 - \frac{1}{N^{m+3}}$ 的概率在 $O(\varepsilon^{-d+1} + \log N)$ 时间内计算完成。因此，运行时间以至少 $1 - \frac{1}{N^{m+3}}$ 的概率为 $O((\varepsilon^{-d+1} + \log N)((N + \varepsilon^{-2}\log^2 N)))$。本节后续引理的证明也同样如此。□

引理 4.4 的证明。根据引理 4.3 的证明，我们知道，查询过程会计算出一个节点集合 $\mathrm{U}(x, r)$，满足 $B(x, r) \cap Q' \subseteq \bigcup_{u \in \mathcal{U}(x,r)} (\square_u \cap Q') \subseteq B(x, (1+\varepsilon)r) \cap Q'$。由于活跃（已构造）节点上的计数器正确，因此引理得证。□

引理 4.5 的证明。如果 $Q' = \emptyset$，那么 $\mathrm{root}.a = 0$ 确实必然成立。否则，就会存在一条从 T 的根节点到节点 u （没有已构造的子节点）的活动节点路径，其中 $\square_u \cap \boldsymbol{q}(\mathrm{D}) \neq \emptyset$。如果 $Q' \neq \emptyset$，并且已知每个节点中的代表点均正确（引理 4.3），则返回的点 $\mathrm{root}.rep$ 是来自 Q'的一个点。为了访问 $\mathrm{root}.a$ 和 $\mathrm{root}.rep$，我们花费 $O(1)$ 时间。□

引理 4.6 的证明。令 U 为 T 中按如下方式构造的节点的子集：当且仅当满足以下条件时，节点 u 属于 U：i) $u \in \mathrm{T}$，ii) 从根节点到 u （包括根节点和 u）之间不存在非活动节点，且 iii) u 的子节点尚未在 T 中构造，或者 u 是完全构造的 RBBD 树的叶节点。注意，$\mathrm{ReportRect}(\boldsymbol{q}, \mathrm{D}, \square_u)$ 过程恰好对每个 $u \in U$ 执行一次。根据活跃点 Q' 的定义，我们有 $Q' = \bigcup_{u \in U} (\square_u \cap \boldsymbol{q}(\mathrm{D}))$。事实上，

算法 1： 主算法($\boldsymbol{q}$，D k，ε，r)

在 $\boldsymbol{q}$(D) 上构建 RBBD 树 T 的根节点；
$\mathcal{S} = \emptyset$;
当 $\mathcal{T}$.rep() $\neq \emptyset$ **执行**
  如果 $|\mathcal{S}| = k$ **则**
    返回 $\emptyset$;
  p = T。代表();
  S =S $\cup \{p\}$;
  T .非活动($p, 2r$)
返回$\mathcal{S}$;

对于每个点 $p \in Q'$，不存在满足 $p \in \square_w$ 的非活动节点 $w \in$ T。因此，对每个 $u \in U$ 执行的过程 ReportRect($\boldsymbol{q}$, D $\square_u$) 恰好返回 Q'。

对于运行时间，我们注意到 $|U| \leq |$T $|$，并且每个过程 ReportRect($\boldsymbol{q}$,D $\square_u$) 的运行时间为$O($ $N + |\square_u \cap \boldsymbol{q}(D)|)$。因此，总运行时间为 $O(|$T $| \cdot N + |Q'|)$。□

引理 4.7 的证明。我们的目标是证明，在每次迭代中，点 $p \in Q'$ 被选中的概率为 $\frac{1}{|Q'|}$。设 $p \in Q'$ 是完整~~-~~RBBD 树的叶节点 u中存储的一个点（这样的节点可能尚未在 T 中构造）。回想一下，当且仅当从根节点到 u的路径上，T 中不存在非活动节点时，点 p才属于 Q'。假设 v是从根节点到 u的路径上，在 T 中构造的最后一个节点。令根节点为 $\rightarrow v_1 \rightarrow \ldots \rightarrow v_\kappa \rightarrow v$。设 w_1、……、w_κ分别是节点 v_1、……、v_κ的兄弟节点。根据第 4.3 节，我们知道，单元 $\square_v$中的点被采样的概率为 $\frac{1}{|\square_v \cap \boldsymbol{q}(\mathrm{D})|}$。选中 p的概率为 $\prod_{j \in [\kappa]} \frac{v_j.c}{v_j.c + w_j.c} \cdot \frac{1}{|\square_v \cap \boldsymbol{q}(\mathrm{D})|}$。注意，$v$是活跃节点，且树中尚未构造 v的子节点，因此 $\square_v \cap \boldsymbol{q}$(D) $= \square_v \cap Q'$。此外，根据计数的正确性，$v_1.c + w_1.c = |Q'|$和 $v.c = |\square_v \cap Q'|$。而且，对于每个 $j \in [\kappa]$，都有$v_j.c = v_{j+1}.c + w_{j+1}.c$。由此可得，选中 $p \in Q'$的概率为 $\frac{1}{|Q'|}$。

对于每个样本，我们花费 $O(\log N)$ 的时间，找到 T 中的一个节点 v；该节点要么是叶节点，要么其子节点尚未构建，因为 T 的高度以高概率为 $O(\log N)$。找到 v后，按照引理 2.1 所示，在 $\square_v \cap Q' = \square_v \cap \boldsymbol{q}$(D) 中的采样过程需要 $O(N)$ 的时间。由于我们获得了 z个样本，总运行时间为 $O(zN)$。

□

B 第 5 节中遗漏的细节

定理 5.1 的证明。注意，对于任意一对 $t, p \in \boldsymbol{q}$(D)，$||t - p||_\infty = \max_{j \in [d]} |\pi_{A_j}(t) - \pi_{A_j}(p)|$。因此，两个点之间的距离可以表示为两个输入元组的坐标之差的绝对值。利用这一观察，我们构造一个由 $O(N)$ 个点组成的集合 V，使得 $\boldsymbol{q}$(D) 中的所有两两距离都包含在 V 中各点的两两距离中。初始时，$V = \emptyset$。对于每个属性 $A_j \in$ A，以及每个满足 $A_j \in$ A$_i$ 的关系 $R_i \in$ R，我们将 R 中元组相对于 A_j 的投影加入 V_i，即 $V = V \cup \{\pi_{A_j}(R_i)\}$。考察所有属性和关系后，注意到对于 D 中任意两个元组之间的两两距离，V 中都存在两个距离相同的点。构造出 V 后，我们对 V 中各点的两两距离进行二分搜索。为此，我们使用 [62] 的结果，在 O 时间内返回 $\mathbb{R}^1$ 中由 V 个点组成的集合的第 z 小两两距离，其中 $= O(N \log N)$。因此，我们进行二分搜索，针对范围 $[0, 1, \ldots, |V \times V|]$ 内 z 的所选 O 个（$\log N$）值，查询第 z 小的距离。设 r' 为 [62] 返回的第 z 小距离。我们设置 $r = \sqrt{d} \cdot r'$

然后运行第 5 节中描述的主算法。最后，我们返回二分搜索返回的最后一个非空集合 S。

引理 B.1. 使用二分搜索，关系型 k-中心问题的主算法会返回一个包含 k 个点的集合 $\mathcal{S}$，使得 $\boldsymbol{u}_{\mathcal{S}}(\boldsymbol{q}(\mathrm{D})) \leq 2\sqrt{d}(1+\varepsilon)\boldsymbol{u}^{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\boldsymbol{q}(\mathrm{D}))$。

证明。对于任意一对点 $t, p \in \mathbb{R}^d$, $||t-p||_\infty \leq ||t-p||_2 \leq \sqrt{d}||t-p||_\infty$ 令 $r^* = \mathrm{u}^{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\boldsymbol{q}(\mathrm{D}))$，并设 $\boldsymbol{q}(\mathrm{D})$ 中的两个元组为 t^*, p^*，使得 $||t^*-p^*||_2 = r^*$。令 $\hat{r} = ||t^*-p^*||_\infty$。根据 V 的定义，V 中将有两个点，它们之间的距离恰为 $\hat{r}$。根据定义，我们有 $\hat{r} \leq r^* \leq \sqrt{d}\hat{r}$。对于每个 $r' \geq \hat{r}$，都有 $r = \sqrt{d}r' \geq \sqrt{d}\hat{r} \geq r^*$。因此，根据第 5 节的分析，主算法会返回一个非空集合 S'，其中至多有 k 个点，且满足 $\mathrm{u}_{\mathrm{S}'}(\boldsymbol{q}(\mathrm{D})) \leq 2(1+\varepsilon)r$。令 $r' \leq \hat{r}$ 为二分搜索中最后一个有效迭代时的距离，该次迭代返回了集合 $\mathrm{S} \neq \emptyset$。我们有 $r = \sqrt{d}r'$。在这种情况下，经过至多 k 次迭代后，$\mathrm{T}.\mathrm{rep}() = \emptyset$（等价地，$\boldsymbol{q}(\mathrm{D})$ 中的所有点均为非活动）。由于每次迭代中，我们都会将每个 $p_i \in \mathrm{S}$ 周围距离不超过 $2r$（以及一些距离不超过 $2(1+\varepsilon)r$）的所有点设为非活动，这意味着 $\boldsymbol{q}(\mathrm{D})$ 中的所有点到 S 的距离均不超过 $2(1+\varepsilon)r$。因此，我们有 $\mathrm{u}_{\mathrm{S}}(\boldsymbol{q}(\mathrm{D})) \leq 2(1+\varepsilon)r = 2\sqrt{d}(1+\varepsilon)r' \leq 2\sqrt{d}(1+\varepsilon)\hat{r} \leq 2\sqrt{d}(1+\varepsilon)r^*$。引理得证。 □

最后，作为对 $\mathbf{u}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D}))$ 的估计，我们令 $r_{\mathcal{S}} = 2(1+\varepsilon)r$，其中 r 是返回最终集合 S 的主算法的输入值。注意，$\mathbf{u}_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D})) \leq r_{\mathcal{S}} \leq 2\sqrt{d}(1+\varepsilon)\mathbf{u}^{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\boldsymbol{q}(\mathbf{D}))$。

使用二分搜索遍历 ℓ_∞ 距离的算法，其运行时间仅比主算法的运行时间多一个 $\log N$ 因子。事实上，二分搜索会探索 $O(\log N)$ 个距离，并使用 $O(N \log N)$ 时间，借助 [62] 中的算法计算每个被探索的距离。因此，总运行时间为

$$O\left(N\log^2 N + k\left(\varepsilon^{-d+1} + \log N\right)\left(N + \varepsilon^{-2}\log^2 N\right)\log N\right) = O(k \cdot N \cdot \log^2 N + k \cdot \log^4 N) = \tilde{O}\left(k \cdot N\right),$$

概率至少为 $1 - \frac{1}{N}$。 □

B.1 另一种 $(2+\varepsilon)$-近似

无需计算 [7] 中的核心集，即可直接从我们在第 5 节中的主算法推导出一个 $(2+\varepsilon)$-近似算法。令 $r_{\mathcal{S}}$ 为定理 5.1 返回的值。显然，$\frac{r_{\mathcal{S}}}{2\sqrt{d}+\varepsilon} \leq \mathrm{u}^{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\boldsymbol{q}(\mathrm{D})) \leq r_{\mathcal{S}}$。我们按乘法步长 $(1+\varepsilon)$ 离散化范围 $[\frac{r_{\mathcal{S}}}{2\sqrt{d}+\varepsilon}, r_{\mathcal{S}}]$，得到集合 $\Psi = \{\frac{r_{\mathcal{S}}}{2\sqrt{d}+\varepsilon}(1+\varepsilon)^0, \frac{r_{\mathcal{S}}}{2\sqrt{d}+\varepsilon}(1+\varepsilon)^1, \ldots, (1+\varepsilon)^\psi \frac{r_{\mathcal{S}}}{2\sqrt{d}+\varepsilon}\}$，其中 $\psi = O(\frac{1}{\varepsilon}\log(2\sqrt{d}+\varepsilon))$。我们对 Ψ 执行二分搜索。对于二分搜索选出的每个 $r \in \Psi$，我们都执行第 5 节中的主算法。按照描述，我们在 Ψ 中继续尝试 r 的更大或更小取值。最后，很容易看出，在二分搜索的最后一步中，对于距离 $r \in \Psi$，返回了一个集合 S'，且 $r \leq (1+\varepsilon)\mathrm{u}^{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\boldsymbol{q}(\mathrm{D}))$，因此 $\mathrm{u}_{\mathrm{S}'}(\boldsymbol{q}(\mathrm{D})) \leq 2(1+\varepsilon)^2\mathrm{u}^{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\boldsymbol{q}(\mathrm{D}))$。当 $\varepsilon \leftarrow \varepsilon/5$ 时，我们有 $(1+\varepsilon/5)^2 \leq 1+\varepsilon$，因此 $\mathrm{u}_{\mathcal{S}'}(\boldsymbol{q}(\mathrm{D})) \leq r_{\mathcal{S}'} \leq (2+\varepsilon)\mathrm{u}^{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\boldsymbol{q}(\mathrm{D}))$。注意，无需显式构造集合 Ψ（即，可以在 $O(1)$ 时间内计算 Ψ 中第 j 小的距离）。相反，我们只需为我们的 k-中心算法额外运行 $O(\log\log_{(2\sqrt{d}+\varepsilon)}\varepsilon)$ 步。对于任意小的常数 $\varepsilon \in (0,1)$，总运行时间仍为 $O(kN\log^2 + k\log^4 N)$，概率至少为 $1 - \frac{1}{N}$。

C 第 6 节中缺失的证明

引理 6.1 的证明。由引理 4.7，我们注意到集合 X_i 是从 Q_i' 均匀随机选取的 $O(k)$ 个样本的集合。对于每个 $j \in \{0, 1, 2, \ldots\ldots, M, \infty\}$，我们执行 T 。非活动(x, b_j)

算法 2： Rel-k-Means-Approx($\boldsymbol{q}$, D k, ε)

使用 Yannakakis 算法计算 $|\boldsymbol{q}(\mathbf{D})|$；
在 $\boldsymbol{q}(\mathbf{D})$ 上构建 RBBD 树 T 的根节点；
$(V, L) \leftarrow$ Rel-k-Center($\boldsymbol{q}$, D k, ε)；$M \leftarrow 2\log(|\boldsymbol{q}(\mathbf{D})|) + 3$；
$a_0 \leftarrow 0$；$b_0 \leftarrow \frac{L}{4|\boldsymbol{q}(\mathbf{D})|}$；$a_\infty \leftarrow 2L|\boldsymbol{q}(\mathbf{D})|$；$b_\infty \leftarrow \infty$；
$a_j \leftarrow 2^{j-1}\frac{L}{4|\boldsymbol{q}(\mathbf{D})|}$，$b_j \leftarrow 2^j \frac{L}{4|\boldsymbol{q}(\mathbf{D})|}$ $\forall j \in [M]$；
S $\leftarrow \emptyset$；$r_S \leftarrow 0$；$\tau_1 \leftarrow |\boldsymbol{q}(\mathbf{D})|$；$i \leftarrow 1$；
当 $\tau_i > 480 \cdot k \log^2 |\boldsymbol{q}(\mathbf{D})|$ **执行**
  $X_i \leftarrow$ T.采样($240k \log^2 |\boldsymbol{q}(\mathbf{D})|$)；
  S $\leftarrow$ S $\cup X_i$；
  T′ $\leftarrow$ 副本(T)；
  对于 $j \in \{0, 1, \ldots, M, \infty\}$ **执行**
    $c_{i,j} \leftarrow 0$；
    对于每个 $x \in X_i$ **执行**
      $c_{i,j} \leftarrow c_{i,j} +$ T′.count(x, b_j)；
      T′.非活动(x, b_j)；
  $\beta_i \leftarrow \max\{j : c_{i,j} > \frac{\tau_i}{10^{\text{对数}|\boldsymbol{q}(\mathbf{D})|}}\}$；
  将 T 重置为定义 c_{i,β_i+1} 之前的状态′；
  $\tau_{i+1} \leftarrow \tau_i - \sum_{\ell \in [\beta_i]\setminus\{0\}} c_{i,\ell}$；
  $r_S \leftarrow r_S + \sum_{\ell \in [\beta_i]\setminus\{0\}} c_{i,\ell} \cdot ((1+\varepsilon)b_\ell)^2$；
  $i \leftarrow i + 1$；
S $\leftarrow$ S $\cup$ T.report()；
返回 S；

对每个 $x \in X_i$。设 $p \in Q'_i$ 是一个点，它在 $c_{i,j}$ 被定义的时刻处于活跃状态，并在对每个 $x \in X_i$ 应用 T′.非活动(x, b_j) 之后变为非活动。根据 T′.非活动(x, b_j) 预言机的正确性（引理 4.3），$p \notin \bigcup_{\ell<j} Q'_{i,\ell}$ 以及 $p \in Q'_{i,j}$。此外，$p \notin \bigcup_{\ell>j} Q'_{i,\ell}$，因为一个点只会变为非活动一次。因此，对于每一对 $j_1, j_2 \in [M] \cup \{0, \infty\}$，$Q'_{i,j_1} \cap Q'_{i,j_2} = \emptyset$。此外，当 $j = \infty$ 时，所有剩余的活跃点根据定义都将变为非活动。因此 $\bigcup_{j \in [M]\cup\{0,\infty\}} Q'_{i,j} = Q'_i$。所以性质 (1) 成立。

接下来，如果我们固定一个索引 $j \in [M] \cup \{0, \infty\}$，对于每个 $x \in X_i$，我们执行 $c_{i,j} = c_{i,j} +$ T′.count(x, b_j)，然后执行 T′.inactive(x, b_j)。根据引理 4.3 和引理 4.4 的证明，T′.count(x, b_j) 预言机返回的恰好是执行 T′.inactive(x, b_j) 时变为非活动的 Q'_i 点的数量。因此，同样有 $c_{i,j} = |Q'_{i,j}|$，性质 (3) 成立。

最后，我们证明性质（2）。令 $j = 0$，并令 $x_1 \in X_i$ 为我们处理的第一个采样。我们执行 T′.inactive(x_1, b_0)。根据引理 4.3，该预言机会将 Q'_i 中距离 x_1 不超过 b_0 的所有点设为非活动（它也可能将距离 $(1+\varepsilon)b_0$ 内的一些点设为非活动）。类似地，对于任意其他 $x_s \in X_i$，预言机 T′.inactive(x_s, b_0) 会将当前活跃的、距离 x_s 不超过 b_0 的所有点设为非活动（它也可能将距离 $(1+\varepsilon)b_0$ 内的一些点设为非活动）。因此，在处理 $j = 0$ 的过程结束时，$Q'_{i,0}$ 中距离 X_i 中某个点不超过 b_0 的所有点都会相对于 T′ 变为非活动。距离 $(1+\varepsilon)b_0$ 内 X_i 的一些点也可能已经变为非活动。因此，$Q'_{i,0}$ 满足性质（2）。根据强归纳法，假设对于每个 $j < j^*$，每个集合 $Q'_{i,j}$ 都满足性质（2），其中 $j^* \in [M] \cup \{\infty\}$。对于每个 $x \in X_i$，预言机 T′.inactive(x, b_{j^*})

会将 Q'_i 中当前活跃且距离 x 不超过 b_{j^*} 的所有点设为非活动（并且可能将距离 $(1+\varepsilon)b_{j^*}$ 内的一些点设为非活动）。令 $Q'_{i,j^*,x}$ 为这一点集，因此有 $Q'_{i,j^*} = \bigcup_{x\in X} Q'_{i,j^*,x}$。根据性质（1），我们有 $Q'_{i,j^*} \cap (\bigcup_{j<j^*} Q'_{i,j}) = \emptyset$。根据归纳假设，集合 $\bigcup_{j<j^*} Q'_{i,j}$ 包含 Q'_i 中距离 X_i 不超过 b_{j^*-1} 的所有点，并且可能包含距离 $(1+\varepsilon)b_{j^*-1}$ 内 X_i 的一些点。因此，Q'_{i,j^*} 包含所有与 X_i 的距离大于 $(1+\varepsilon)b_{j^*-1} = (1+\varepsilon)a_{j^*}$ 且不超过 b_j 的点，可能包含距离 X_i 不超过 $(1+\varepsilon)b_{j^*}$ 的一些点，并且可能包含与 X_i 的距离大于 $b_{j^*-1} = a_{j^*}$ 的一些点。因此，Q'_{i,j^*} 满足性质（2）。

□

引理 6.2 的证明。固定算法的第 i 次迭代。对于每个 $p_s \in \mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))$，我们放置最小球 B_s，其中心为 p_s，并包含来自 Q'_i 的 $20k^{\tau_i} \cdot \log |\boldsymbol{q}(\mathrm{D})|$ 个点。由于 $X_i \subset Q'_i$ 和 $|X_i| = \Theta(k\log^2 |\boldsymbol{q}(\mathrm{D})|)$，因此以至少 $1 - \frac{1}{N^3}$ [41, 42] 的概率，每个球 B_s 内至少有一个 X_i 点。也就是说，对每个 $s \in [k]$，都有 $X_i \cap B_s \neq \emptyset$。令 $x_s \in X_i$ 表示对每个 $s \in [k]$ 位于球 B_s 内的点。令 $p \in Q'_i \setminus ((\bigcup_{\ell\in[k]} B_\ell) \cap Q'_i)$ 为球 $\bigcup_{\ell\in[k]} B_\ell$ 之外的一个点。令 p_s 为 $\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))$ 中距离 p 最近的中心，即 $\phi(p, \mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))) = \phi(p, p_s)$。由于 $x_s \in B_s \cap Q'_i$，注意到 $\phi(p, x_s) \leq 2\phi(p, p_s)$，所以 $p \in Q'_i \setminus Q_i^{\mathrm{bad}}$，即 p 是一个好点。因此，以至少 $1 - \frac{1}{N^3}$ 的概率，Q'_i 中所有坏点都位于球 B_1、……、B_k 内。然而，这些球中的每一个至多包含 Q'_i 的 $20k^{\tau_i} \log |\boldsymbol{q}(\mathrm{D})|$ 个点。因此，以至少 $1 - \frac{1}{N^3}$ 的概率，有 $|Q_i^{\mathrm{bad}}| \leq k \cdot 20k^{\tau_i} \log |\boldsymbol{q}(\mathrm{D})| = 20^{\tau_i} \log |\boldsymbol{q}(\mathrm{D})|$。□

引理 6.3 的证明。显然，$|\hat{Q}_i| \geq \tau_i - c_{i,\infty} - M \cdot 10^{\tau_i} \log |\boldsymbol{q}(\mathrm{D})|$，因为 β_i 是满足 $Q'_{i,\beta_i} > 10^{\tau_i} \log |\boldsymbol{q}(\mathrm{D})|$ 的最大索引。注意，对于每个点 $p \in Q'_{i,\infty}$，都有 $\phi(p, X_i) \geq 2L|\boldsymbol{q}(\mathrm{D})|$。然而 $\mu_{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\boldsymbol{q}(\mathrm{D})) \leq |\boldsymbol{q}(\mathrm{D})|L^2$，因此 $\phi(p, \mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))) \leq \sqrt{|\boldsymbol{q}(\mathrm{D})|L}$，从而 $p \in Q_i^{\mathrm{bad}}$。因此，$Q'_{i,\infty} \subseteq Q_i^{\mathrm{bad}}$。此外，由引理 6.2 可知，$|Q_i^{\mathrm{bad}}| \leq 20^{\tau_i} \log |\boldsymbol{q}(\mathrm{D})|$ 的概率至少为 $1 - \frac{1}{N^3}$。因此，$|\hat{Q}_i| \geq \tau_i - 20^{\tau_i} \log |\boldsymbol{q}(\mathrm{D})| - (2\log |\boldsymbol{q}(\mathrm{D})| + 3) 10^{\tau_i} \log |\boldsymbol{q}(\mathrm{D})| \geq \frac{\tau_i}{2}$。所以，每次迭代中，至少一半剩余的点会变为非活动状态。由此可知，该算法以至少 $1 - \frac{1}{N^2}$ 的概率运行 $O(\log |\boldsymbol{q}(\mathrm{D})|) = O(\log N)$ 次迭代。每次迭代中，我们将集合 X_i 加入 S，且 $|X_i| = \Theta(k\log^2 N)$，因此以至少 $1 - \frac{1}{N^2}$ 的概率有 $|\mathrm{S}| = O(k\log^3 N)$。□

引理 6.4 的证明。我们首先证明近似因子。固定算法的第 i 次迭代。如果 $\beta_i > 0$，则论证如下。令 $¶ = Q'_{i,\beta_i} \setminus Q_i^{\mathrm{bad}}$。对于任意点 $p \in \hat{Q}_i \cap Q_i^{\mathrm{bad}}$ 和 $q \in ¶$，有 $\phi(p, X_i) \leq 2(1+\varepsilon)\phi(q, X_i)$。该不等式由集合 $Q'_{i,j}$ 的定义以及 T′.inactive(·, ·) 预言机得出。此外，以至少 $1 - \frac{1}{N^2}$ 的概率，$|¶| > 10^{\tau_i} \log |\boldsymbol{q}(\mathrm{D})| - 20^{\tau_i} \log |\boldsymbol{q}(\mathrm{D})| = 20^{\tau_i} \log |\boldsymbol{q}(\mathrm{D})| \geq |\hat{Q}_i \cap Q_i^{\mathrm{bad}}|$。根据定义，¶ 中的每个点相对于 X_i 都是一个 good 点，因此 $\mu_{X_i}(¶) \leq 4\mu_{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(¶)$。此外，由于 $¶ \subseteq Q_i$，我们有 $\mu_{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(¶) \leq \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\hat{Q}_i)$。因此，

$$\mu_{X_i}(\hat{Q}_i \cap Q_i^{\mathrm{bad}}) \leq 4(1+\varepsilon)^2 \mu_{X_i}(¶) \leq 16(1+\varepsilon)^2 \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(¶) \leq 16(1+\varepsilon)^2 \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\hat{Q}_i).$$

So,

$$\begin{aligned}\mu_{X_i}(\hat{Q}_i) &= \mu_{X_i}(\hat{Q}_i \cap Q_i^{\mathrm{bad}}) + \mu_{X_i}(\hat{Q}_i \setminus Q_i^{\mathrm{bad}}) \leq 16(1+\varepsilon)^2 \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\hat{Q}_i) + 4\mu_{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\hat{Q}_i) \\ &\leq 20(1+\varepsilon)^2 \cdot \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\hat{Q}_i).\end{aligned}$$

If $\beta_i = 0$，那么对于每个点 $q \in \hat{Q}_i$，都有 $\phi(q, X_i) \le (1+\varepsilon)\, 4|\boldsymbol{q}^{L}(\mathbf{D})|_{\circ}$ 因此，对于 $\varepsilon \le 0.1$，

$$\mu_{X_i}(\hat{Q}_i) = \sum_{q \in \hat{Q}_i} \phi^2(q, X_i) \le (1+\varepsilon)^2 \frac{L^2}{16|\boldsymbol{q}(\mathbf{D})|} < \frac{L^2}{9|\boldsymbol{q}(\mathbf{D})|} \le \frac{1}{|\boldsymbol{q}(\mathbf{D})|} \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D})).$$

无论如何，我们有 $\mu_{X_i}(\hat{Q}_i) \le \max\{20(1+\varepsilon)^2 \cdot \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\hat{Q}_i)\, |\boldsymbol{q}^{1}(\mathbf{D})|, \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))\}$。

根据定义，回顾一下，对于每个 $i_1 \ne i_2$，都有 $\hat{Q}_{i_1} \cap \hat{Q}_{i_2} = \emptyset$。总体而言，$\mu_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D})) \le 21(1+\varepsilon)^2 \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$，其概率至少为 $1 - \frac{1}{N}$。

最后，对于每个点 $p \in \hat{Q}_i$，满足 $p \in Q'_i$，对于索引 $j' \le \beta_i$，我们将 $((1+\varepsilon)b_{j'})^2$ 计入 $r_{\mathcal{S}}$。根据定义，如果 $j' > 0$，则 $\phi(p, X_i) \le (1+\varepsilon)b_{j'} \le 2(1+\varepsilon)\phi(p, X_i)$。因此，相对于 X_i，p 的 k-均值成本至多计入 $4(1+\varepsilon)^2\phi^2(p, X_i)$，且至少为 $\phi^2(p, X_i)\hat{}$，计入 $r_{\mathcal{S}}$。如果 $j' = 0$，那么根据 $\beta_i = 0$ 时 $\mu_{X_i}(Q_i)$ 的界，我们计入一个至多为 $|\boldsymbol{q}^{1}(\mathbf{D})|\, \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$、且至少为 $\mu_{\mathcal{S}}(\hat{Q})$ 的值，计入 $r_{\mathcal{S}}$。一般而言，对于 $\hat{Q}_i$ 中的所有点，我们计入一个至多为 $\sum_{p \in \hat{Q}_i} 4(1+\varepsilon)^2\phi^2(p, X_i) \le 84(1+\varepsilon)^4 \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\hat{Q}_i)$、且至少为 $\sum_{p \in \hat{Q}_i} \phi(p, X_i) \ge \mu_{\mathcal{S}}(\hat{Q}_i)$ 的值，计入 $r_{\mathcal{S}}$。总体而言，$\mu_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D})) \le r_{\mathcal{S}} \le 84(1+\varepsilon)^4 \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$。令 $\varepsilon \leftarrow \varepsilon/400$，可得 $\mu_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D})) \le r_{\mathcal{S}} \le (84+\varepsilon)\mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$，其概率至少为 $1 - \frac{1}{N}$。

接下来，我们分析运行时间。首先，我们以至少 $1 - \frac{1}{10N}$ 的概率，在 $O\left(kN\log^2 N + k\log^4 N\right)$ 时间内运行定理 5.2 中的 k-中心算法。随后，该算法以至少 $\frac{1}{N^2}$ 的概率运行 $O(\log N)$ 次迭代。对于每次迭代 i，利用引理 4.7 的结果，我们以至少 $1 - \frac{1}{N^{m+3}}$ 的概率，在 $O(kN\log^2 N)$ 时间内得到集合 X_i。$\ell \in [M] \cup \{\text{有}\}$ $O(\log N)$ 个不同取值；对于每个 ℓ，我们以至少 $1 - \frac{O(\log^2 N)}{N^3}$ 的概率，在 $O(kN\log^3 N + k\log^5 N)$ 时间内计算 c_i、ℓ，并执行 T′.inactive() 预言机 $O(k\log^2 N)$ 次（回顾引理 4.4，T′.inactive() 的运行时间为 $O(N\log N + \log^3 N)$）。固定一次迭代 i 后，我们以至少 $1 - \frac{O(\log^3 N)}{N^3} \ge 1 - \frac{1}{N^2}$ 的概率，在 $O(kN\log^4 N + k\log^6 N)$ 时间内计算所有取值 c_i、ℓ。总体而言，在 $O(\log N)$ 次迭代期间（概率至少为 $1 - \frac{1}{N^2}$），计算所有计数所需的运行时间为 $O(kN\log^5 N + k\log^7 N)$，其概率至少为 $1 - \frac{2}{N^2}$。类似地，在最后一次迭代（记为 i'）中，注意到 $|\mathrm{T}| = O(k\log^5 N)$，其概率至少为 $1 - \frac{2}{N^2}$。根据引理 4.6，以及 $|\tau_{i'}| = O(k\log^2 N)$ 这一事实，预言机 T.report() 的运行时间为 $O(kN\log^5 N)$，其概率至少为 $1 - \frac{3}{N^2}$。总体而言，我们的算法以至少 $1 - \frac{1}{N}$ 的概率，在 $O(kN\log^5 N + k\log^7 N)$ 时间内运行。□

C.1 更快的核心集构造

在描述来自 [31], 的更快核心集构造之前，我们先简要说明，第 4.4 节中定义的预言机也可以针对盒而非球进行定义。

在标准计算环境中，当查询形状不是球而是盒 ρ in $\mathbb{R}^d$ 时，也可以使用 BBD 树执行报告、计数和采样查询。查询过程 T (ρ) 返回一个包含 $O(\log n + \varepsilon^{-d+1})$ 个规范节点的集合 U (ρ)，使得 $\rho \subseteq \bigcup_{u \in \mathcal{U}(\rho)} \square_u \subseteq \rho \oplus \varepsilon_\rho$，其中 $\varepsilon_\rho = \varepsilon \cdot \mathrm{diam}(\rho)$，$\mathrm{diam}(\rho)$ 是 ρ 的直径，而 $\rho \oplus \varepsilon_\rho$ 是 ρ 与 ε_ρ 的闵可夫斯基和，即，$\rho \oplus \varepsilon_\rho$ 是 ρ 的一个超集，使得 $\rho \oplus \varepsilon_\rho$ 中的每个点到其在 ρ 中最近点的距离都不超过 $\varepsilon \cdot \mathrm{diam}(\rho)$。

在关系型场景中，给定一个部分构造的RBBD树和一个盒 □，预言机 T .inactive(□) 会使 $\bigcup_{u \in \mathcal{U}(\square)} (\square_u \cap \boldsymbol{q}(\mathbf{D}))$ 中的所有点变为非活动点，这与第 4.4 节中的停用预言机类似。同样，T .count(□) 会对 $\bigcup_{u \in \mathcal{U}(\square)} |\square_u \cap \boldsymbol{q}(\mathbf{D}))|$ 中的所有活跃点进行计数。此外，我们定义预言机 T.rep(□)，它从 $\square \cap \boldsymbol{q}(\mathbf{D})$ 中返回一个活跃点。我们运行 T (□) 查询，并在必要时构造新节点（如第 4.4 节所示），跳过已经

非活动的节点。我们计算 U (□)，并针对节点 $u \in$ U (□) 返回 $u.rep$。根据RBBD树（以及BBD树）的定义，如果 Q' 是活跃点的集合，则 T.inactive(□) 会使点集 $\bigcup_{u \in \mathcal{U}(\square)} \square_u \cap Q'$ 变为非活动点，其中 $\square \cap Q' \subseteq \bigcup_{u \in \mathcal{U}(\square)} (\square_u \cap Q') \subseteq (\square \oplus \varepsilon_\square \cap Q')$。对于 count，我们注意到预言机会返回一个数值 $\sum_{u \in \mathcal{U}(\square)} |\square_u \cap Q'|$，使得 $|\square \cap Q'| \leq \sum_{u \in \mathcal{U}(\square)} |\square_u \cap Q'| \leq |\square \oplus \varepsilon_\square \cap Q'|$。类似地，T.rep(□) 会返回一个活跃点 $p \in Q'$，使得 $p \in \square \oplus \varepsilon_\square \cap Q'$。每个新预言机的运行时间为 $O((\varepsilon^{-d+1} + \log N)(N + \varepsilon^{-2} \log^2 N))$，其概率至少为 $1 - \frac{1}{N^{m+3}}$。

第 [31] 节中的核心集构造需要执行以下操作。设 $\square_1$ 是 $\mathbb{R}^d$中的一个正方体盒（所有边长都相等）。对于 $\square_1$，我们应从 $\square_1 \cap \boldsymbol{q}$(D)中获取一个代表点，并计算计数 $\square_1.c = |\square_1 \cap \boldsymbol{q}$(D)|。对于每个新的正方体盒 $\square_i \in \mathbb{R}^d$，重复从$(\square_i \setminus (\bigcup_{j<i} \square_j)) \cap \boldsymbol{q}$(D)中获取一个代表点，并计算计数 $\square_i.c = |(\square_i \setminus (\bigcup_{j<i} \square_j)) \cap \boldsymbol{q}$(D)|。换言之，每当遇到一个新的正方体盒 $\square_i$，目标就是统计属于 $\square_i$ 但不属于任何先前定义的、针对 $j < i$的正方体盒 $\square_j$ 的 $\boldsymbol{q}$(D)中的点数（并获取一个代表点）。虽然我们的目标并非完整描述他们的核心集构造，但直观而言，他们考虑的每个正方体盒 $\square_i$ 都会在核心集 C 中生成一个新点（代表点），其权重为 $\square_i.c$。若 S 是第一阶段返回的点集，他们会得到 $O(|\mathbb{S}|\varepsilon^{-d} \log N)$个正方体盒，因此其核心集大小为 $O(|\mathbb{S}|\varepsilon^{-d} \log N)$。在 [31],中，作者通过在 $\square_i$中进行均匀随机采样来计算$\square_i.c$的估计值。然而，为了对每个正方体盒˜进行准确估计，所需的样本数大约为 $O(|\mathbb{S}|) = \Omega(k)$，因此运行时间会额外增加 k^2 项，此处略去额外的 $\log N$ 因子。借助我们在关系型场景中的新 RBBD 构造，计算 $\square_i.c$ 的操作变得更简单。首先，令 T 由 $\boldsymbol{q}$(D)上的 RBBD 树的根节点构成，具体如第 4.3 节所述。对于我们考虑的每个新正方体盒 $\square_i$，执行 T .count($\square_i$) 来获取计数，执行 T .rep($\square_i$) 来获取一个代表点，并执行 T .inactive($\square_i$)，将$\bigcup_{u \in$U (□$_i$) $\square_u \cap \boldsymbol{q}$(D)中的所有点设为非活动。这样的算法能够以 $O((\varepsilon^{-d+1} + \log N)(N + \varepsilon^{-2} \log^2 N)|\mathbb{S}|\varepsilon^{-d} \log N)$的运行时间构造核心集，且具有高概率。在我们的情形中，$|\mathbb{S}| = O(k \log^3 N)$，因此对于任意小的常数 ε，运行时间为 $O(kN \log^5 N + k \log^7 N)$。最后，我们注意到，在 [31] 中，作者只计算了 $\square_i.c$的一个估计值。使用 RBBD 树时，我们同样不会精确计算计数。相反，根据 RBBD 树的保证，我们计算每个正方体盒中的点数，并假设每个盒子的边界额外偏移了 $\varepsilon \cdot$ 直径。回顾上文的讨论，每个正方体盒 $\square_i$都由 $\square_i \oplus \varepsilon_{\square_i}$近似。不过，这只会使总体代价增加一个 $(1 + \varepsilon)^2$ 因子（由于这些盒子都是正方体盒，其直径决定了每个正方体盒中两点间的最大距离，而这一距离用于 [31] 中界定总体误差）。令 $\varepsilon \leftarrow \varepsilon/4$，即可得到与 [31]完全相同的近似。

D 基于类别属性和数值属性的关系聚类

在本节中，我们展示如何将我们的树数据结构与 [26] 的框架相结合，从而获得在同时存在数值属性和类别属性时关系聚类的已知最快算法。令$\mathbf{A}_{(c)} \subset \mathbf{A}$ 表示类别属性的集合，并令 $\mathbf{A}^{(n)} = \mathbf{A} \setminus \mathbf{A}^{(c)}$表示数值属性的集合，其中 $d_c = |\mathbf{A}_{(c)}|$ 且 $d_n = |\mathbf{A}_{(n)}|$。为简单起见，我们假设 $\mathbf{A}^{(c)}$ 中的每个分类属性最多可取 $z = O(N)$个不同的值。在接下来的小节中，我们首先说明我们的聚类算法如何扩展到包语义下的连接-投影查询（在聚类目标中忽略类别属性），然后将我们的结果与 [26] 的框架相结合，从而推导出一个更快的算法（比 [26]更快），并在目标中同时考虑类别属性和数值属性。

D.1 将类别属性投影掉：包语义下的连接-投影查询考虑这样一种情形：数据库模式包含类别属性$\mathbf{A}^{(c)}$，但目标是仅根据其数值属性对连接结果进行聚类。形式上，对于 $X \subseteq \mathbf{A}$ 和$Y \subseteq \boldsymbol{q}(\mathbf{D})$，令 $\bar{\pi}_X(Y)$为多重集 $\bar{\pi}_X(Y)=\{\pi_X(t) \mid t \in Y\}$。与标准投影算子的唯一区别在于，$\bar{\pi}$ 会保留重复项（这等价于包语义中的投影算子）。给定数据库实例$\mathbf{D}$上的连接查询 $\boldsymbol{q}$，目标是对 $\bar{\boldsymbol{q}}(\mathbf{D}) = \bar{\pi}_{\mathbf{A}^{(n)}}(\boldsymbol{q}(\mathbf{D}))$进行聚类，即在包语义下将 $\boldsymbol{q}(\mathbf{D})$投影到数值属性$\mathbf{A}^{(n)}$ 上。

定理 5.2 和 6.5 中的所有算法都可以自然地扩展到这一设定，并保持相同的近似和时间保证。现在，RBBD 树仅定义在 d_n 数值属性 $\mathbf{A}^{(n)}$上；对于每个盒（或带孔盒） ρ，引理 2.1 中的所有操作都可以在 $\bar{\boldsymbol{q}}(\mathbf{D}) \cap \rho$上于 $O(N)$ 时间内完成。例如，从$\bar{\boldsymbol{q}}(\mathbf{D}) \cap \rho$ 中均匀随机采样一个元组，等价于从 $\{t \in \boldsymbol{q}(\mathbf{D}) \mid \pi_{\mathbf{A}^{(n)}}(t) \in \rho\}$ 中均匀随机采样一个元组。更一般地，我们的算法所用的所有 RBBD 树操作，例如对盒内的元组进行计数、报告或采样，都可以直接应用于采用集合语义的相应连接查询 $\boldsymbol{q}$，以处理采用包语义的投影连接查询 $\bar{\boldsymbol{q}}$ 的结果。因此，在采用包语义的连接-投影查询上执行我们的所有算法，都能保持相同的近似和时间保证。

备注。 虽然我们的算法通常不能扩展到集合语义下的连接-投影查询，但在元组重数不会影响目标值的情况下仍然适用。值得注意的是，这包括关系型 k-中心聚类问题（第 5 节）和关系型远边问题（附录 E.3）；对于这两个问题，集合语义和包语义下的连接-投影查询会产生等价的实例。

D.2 同时具有数值和类别属性的聚类

我们关注 [26] 中的 k-均值聚类问题；不过，我们的所有结果也都可以扩展到关系型 k-中心聚类和 k-中位数聚类。在机器学习中，处理类别变量的一种常见方法是采用独热编码，将每个类别变量转换为二进制表示。例如，考虑一个可以取 z 个不同值的分类属性 $A \in \mathbf{A}$。假设这些值有一个任意但固定的顺序。那么， A 的第 j个值可以用一个二进制向量表示，其中第 j位为 1，其余各位为 0。

代价高昂的算法。在关系型场景中，我们可以将每个分类属性 $A \in \mathbf{A}^{(c)}$替换为 z 个二元属性 $A_1^{(\text{bin})}$，……，$A_z^{(\text{bin})}$。关系 R中包含属性 A的每个元组 t都会被替换为一个新元组 t'，使得如果 π_A（t）对应于 A的第 j个值，那么$\pi_{A_j^{(\text{bin})}}$（t'）为 $= 1$，且对于每个 $h \in [z] \setminus \{j\}$，$\pi_{A_h^{(\text{bin})}}$（$t'$）为$= 0$。经过此变换后，连接查询 $\boldsymbol{q}$涉及 m个关系，总共包含 $d_n + d_c \cdot z$ 个数值属性。因此，距离函数 $\phi(\cdot,\cdot)$在数值属性上仍是欧几里得度量。根据定理 6.5，对于涉及数值属性和分类属性的关系型 k-均值问题，我们可以得到一个 5γ-近似算法（针对 $\varepsilon = 1$），其数据复杂度为 $O(2^z kN + T$均值$\gamma(2^z k))$。当 $z = O(1)$时，这与定理 6.5 给出的运行时间相符。然而在实践中， z 可能很大。例如，分类属性可能表示国家、颜色或产品类别，因此由于对 z的依赖呈指数级增长，上述方法并不实用。接下来，我们将介绍一种用于关系型 k-均值聚类的替代算法，该算法无需显式进行独热编码，且其运行时间不会随 z呈指数级增长。

已知算法 *[26]*。首先，从数据库中删除所有悬空元组。对于每个属性$A \in \mathbf{A}$，令 S_A 表示数据库实例 $\mathbf{D}$ 中出现的所有值的集合；即$S_A = \text{adom}_{\mathbf{D}}(A)$。对于每个值 $x \in S_A$，他们计算权重 $w(x) = \sum_{t \in \boldsymbol{q}(\mathbf{D})} 1[\pi_A(t) = x]$,

表示其投影到属性 A 后等于 x 的连接结果数量。利用 Yannakakis 算法，可以在 $O(N)$ 时间内计算所有值 $x \in \text{adom}(A)$ 和属性 $A \in \mathbf{A}$ 对应的权重 $w(x)$。然后，对于每个属性 $A \in \mathbf{A}$，分两种情况处理。如果 $A \in \mathbf{A}^{(c)}$，他们会采用一种简单高效的过程，在 $O(N)$ 时间内计算加权类别集合 S_A 的最优 k-均值聚类。如果 $A \in \mathbf{A}^{(n)}$，他们会调用 [36, 39, 69]，在 $O(kN)$ 时间内计算加权数值集合 S_A 的最优 k-~均值聚类。无论哪种情况，都令 C_A 表示属性 A 对应的 k 个中心所组成的集合。计算完所有属性 $A \in \mathbf{A}$ 对应的 C_A 后，他们构造网格 $C = \bigtimes_{A \in \mathbf{A}} C_A$，其中包含各属性中心的所有可能组合。注意，$|C| = O(k^d)$，其中 $d = |\mathbf{A}|$ 是属性总数。对于每个中心 $p \in C$，他们计算权重 $\omega(p)$，其定义为关系 $\boldsymbol{q}(\mathbf{D})$中以 p 为最近中心的元组数量。~利用引理 2.1 中的技术，可以在 $O(N)$ 时间内计算每个权重 $\omega(p)$。可以证明，加权集合 C（权重函数为 ω）构成关系型 k-均值聚类的核心集。在加权集合 C 上运行用于 k-均值聚类的 γ-近似算法，可得到一个包含 k 个中心的集合 S，使得 $\mu_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D})) \leq (2\sqrt{\gamma}+1)^2 \mu_{\text{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$。该算法的运行时间为 $\tilde{O}(k^d N + T\text{mean}_{\gamma}(k^d))$。

我们的算法。我们利用新结果，更高效地处理数值属性，从而设计出一种更快的算法。首先，我们按照 [26] 的方法处理类别属性：对于每个 $A \in \mathbf{A}^{(c)}$，我们构造 S_A，并以与 [26] 相同的方式计算每个 $x \in S_A$ 的权重 $w(x)$。然后，我们在加权集合 $\text{adom}(A)$ 上应用求解最优 k-均值聚类的直接算法，并令 C_A 表示最优中心集合。接下来，我们处理数值属性 $\mathbf{A}^{(n)}$。与 [26], 不同，我们同时处理所有数值属性。具体而言，我们运行定理 6.5 中的关系型 k-均值算法，同时跳过类别属性；也就是说，我们按照附录 D.1 中的描述，在 $\bar{q}(\mathbf{D}) = \bar{\pi}_{\mathbf{A}^{(n)}}(\boldsymbol{q}(\mathbf{D}))$ 上运行该算法。令 C' 为所得的 k 个中心组成的集合。值得注意的是，我们不会删除 $\bar{q}(\mathbf{D})$ 中的重复项，因为我们的算法会自动对元组进行正确加权。特别地，直接在 $\bar{q}(\mathbf{D})$ 上运行 k-均值聚类，等价于在加权集合 $\pi_{\mathbf{A}^{(n)}}(\boldsymbol{q}(\mathbf{D}))$ 上运行 k-均值聚类，其权重为 $w(p) = \sum_{t \in \boldsymbol{q}(\mathbf{D})} 1[\pi_{\mathbf{A}^{(n)}}(t) = p]$。

然后，我们将候选中心集合构造为 $C = C' \times \left(\bigtimes_{A \in \mathbf{A}^{(c)}} C_A\right)$。遗憾的是，与 [26], 不同，我们无法直接计算对于每个 $p \in C$，在 $\boldsymbol{q}(\mathbf{D})$中有多少个元组恰好以 p 作为其在 C中的最近中心。相反，我们借鉴了在关系型场景中构造 RBBD 树的思路，近似计算以每个中心 $p \in C$为最近中心的元组数量。对于每个 $p \in C$，我们计算一个近似权重 $\bar{\omega}(p) = |\{t \in \boldsymbol{q}(\mathbf{D}) \mid \iota(t) = p\}|$，其中 $\iota: \boldsymbol{q}(\mathbf{D}) \to C$ 是一个函数，它将 $t \in \boldsymbol{q}(\mathbf{D})$中的每个元组映射到 C 中的某个中心，并满足$\sum_{t \in \boldsymbol{q}(\mathbf{D})} \phi^2(t, \iota(t)) \leq 2\,\mu_{\text{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$。这等价于先根据定理 6.5，以 $2 \cdot 5\gamma$（而非 5γ）作为近似因子，对 $\bar{\boldsymbol{q}}(\mathbf{D})$中的元组进行聚类，然后计算真实权重 $\omega(p)$（回想一下，$\omega(p)$是指在$\boldsymbol{q}(\mathbf{D})$中以 p 为最近中心的元组数量）。我们在附录D.3中展示了计算权重 $\bar{\omega}(\cdot)$的详细过程。最后，我们对加权集合 C运行 k-均值聚类算法，得到一个包含 k 个最终中心的集合 S。

定理D.1。 给定一个包含 d_n 数值属性和 d_c 分类属性的无环连接查询 $\boldsymbol{q}$、一个大小为 N 的数据库实例 $\mathbf{D}$ 以及一个参数 k，存在一种算法，能够计算出一个包含 k个点的集合$\mathcal{S} \subseteq \boldsymbol{q}(\mathbf{D})$，使得 $\mu_{\mathcal{S}}(\boldsymbol{q}(\mathbf{D})) \leq \left((\sqrt{10}+1)\sqrt{\gamma} + \sqrt{10\gamma}\right)^2 \mu_{\text{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$在 $\tilde{O}(k^{d_c+1}N + T_{\gamma}^{\text{mean}}(k^{d_c+1}))$ 的时间内完成，其概率至少为 $1 - \frac{1}{N}$。

证明。由附录D.3可知，$\sum_{t \in \boldsymbol{q}(\mathbf{D})} \phi^2(t, \iota(t)) \leq 2\,\mu_{\text{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$。这等价于先按照定理 6.5 对 $\bar{\boldsymbol{q}}(\mathbf{D})$ 进行聚类，其近似

误差为 $2\cdot 5\gamma$（而非 5γ），然后计算精确权重 $\omega(p)$。按照 [26]，中的分析，总体近似因子变为 $((\sqrt{10}+1)\sqrt{\gamma}+\sqrt{10\gamma})^2$。

对于每个分类属性 $A\in\mathbf{A}_{(c)}$，中心集合 C_A 在 $O(N)$时间内计算，如前文所述。将定理 6.5 中的算法应用于 $\bar{\pi}_{\mathbf{A}_{(n)}}(\boldsymbol{q}(\mathbf{D}))$ 时，以高概率在 $O(kN+T_{\text{均值}\,\gamma}(k))$ 时间内运行。由于 $|C|=O(k^{d_c+1})$，$p\in C$的所有权重 $\bar{\omega}(p)$在$\tilde{O}(k^{d_c+1})$时间内计算（见附录D.3）。最后，加权集合 C 上的 k-均值聚类在 $\tilde{O}(T_{\text{均值}\gamma}(k^{d_c+1}))$时间内执行。总体而言，我们算法的总运行时间以高概率为 $\tilde{O}(k^{d_c+1}N+T_{\gamma}^{\text{mean}}(k^{d_c+1}))$。□

D.3 近似最近中心点

回顾一下，$\bar{\boldsymbol{q}}(\mathbf{D})=\bar{\pi}_{\mathbf{A}_{(n)}}(\boldsymbol{q}(\mathbf{D}))$，并注意 $|\bar{\boldsymbol{q}}(\mathbf{D})|=|\boldsymbol{q}(\mathbf{D})|$。

我们设计了一种算法，可高效地为所有$p\in C$计算权重 $\omega(p)$ 的估计值 $\bar{\omega}(p)$。我们使用第 4.4 节中引入的预言机，并做了几处细微修改。回顾一下，对于部分构造的 RBBD 树 T 中的每个节点 u，我们都会存储对应的盒（或带孔盒）$\square_u$、二元变量 $u.a$、代表点 $u.\text{rep}$，以及计数器 $u.c$。在修改后的版本中，我们还会在 T 的每个已构造节点处额外存储并维护 k^{d_c} 个辅助计数器。具体而言，对于 T 中的每个节点 $u\in$，我们维护如下形式的计数器：$u(i_1,i_2,\ldots,i_{d_c})$，其中 $1\le i_j\le k$对每个 $j\in[d_c]$都成立。每个计数器定义为

$$u(i_1,i_2,\ldots,i_d)=|\{t\in\boldsymbol{q}(\mathbf{D})\mid\pi_{\mathbf{A}_{(n)}}(t)\in\square_u,\forall A^{(j)}\in\mathbf{A}_{(c)},\phi(\pi_{A^{(j)}}(t),C_{A^{(j)}})=\phi(\pi_{A^{(j)}}(t),C_{A^{(j)}}[i_j])\}|,$$

其中，$C_{A^{(j)}}[i_j]$表示 $C_{A^{(j)}}$中的第 i_j个点（按照任意但固定的顺序排列）。直观地说，每个计数器 $u(i_1,i_2,\ldots,i_{d_c})$存储 $\boldsymbol{q}(\mathbf{D})$中满足以下条件的元组数量：其在数值属性上的投影位于 $\square_u$ 内，且在每个分类属性 $A^{(j)}$ 上的投影都最接近 $C_{A^{(j)}}$中的第i_j个中心。

我们首先描述如何为新构造的节点 $u\in$T计算 $u(i_1,i_2,\ldots,i_{d_c})$。假设 $\square_u$ 是一个无洞盒子；如果它包含一个孔，我们只需按照附录 A.3 中的说明，对每个互不相交的盒子重复该过程。遵循引理 2.1 的方法（最初在 [7]中提出），对于每个关系 $R\in$ R，我们找出所有满足以下条件的元组 $t\in R$：对于每个 $A\in\mathbf{A}(R)$，$\cap\mathbf{A}^{(n)}$，$\pi_A(t)\in\pi_A(\square_u)$；并且对于每个 $A^{(j)}$，$\in\mathbf{A}(R)\cap\mathbf{A}^{(c)}$，$\pi_{A^{(j)}}(t)$的最近中心是 $C_{A^{(j)}}$中的第i_j个中心。其余元组均予以丢弃。令 $R'\subseteq R$表示关系 R中剩余的元组。然后，我们在连接 $\bowtie_{R\in\mathbf{R}}R'$ 上执行 Yannakakis 算法的计数版本，并在 $O(N)$时间内设置 $u(i_1,i_2,\ldots,i_{d_c})=|\bowtie_{R\in\mathbf{R}}R'|$。正确性直接源自 R'的定义。由于 Yannakakis 算法执行 k^{d_c} 次，在 T 中构造一个新节点需要 $O(k^{d_c}N)$时间。标记新的非活动节点后，我们以类似于 $u.c$的方式更新 $u(i_1,i_2,\ldots,i_{d_c})$的值。例如，考虑一个具有子节点$\hat{v}$ 和 $\bar{v}$的节点 u。根据定义，有 $u(i_1,i_2,\ldots,i_{d_c})=\hat{v}(i_1,i_2,\ldots,i_{d_c})+\bar{v}(i_1,i_2,\ldots,i_{d_c})$。如果$\hat{v}$变为非活动节点，则 $u(i_1,i_2,\ldots,i_{d_c})=\bar{v}(i_1,i_2,\ldots,i_{d_c})$。

对于球 $\mathbf{B}(x,r)$，令 $\mathrm{T}.\overline{\text{inactive}}(x,r)$ 的定义类似于 $\mathrm{T}.\text{inactive}(x,r)$，只不过它还会按上文所述更新所有值 $u(i_1,i_2,\ldots,i_{d_c})$。此外，令 $\mathrm{T}.\text{count}(x,r)$ 表示一个类似于 $\mathrm{T}.\overline{\text{count}}(x,r)$ 的预言机，唯一区别在于它返回 k^{d_c}个计数器。具体而言，对于每一个 $(i_1,\ldots,i_{d_c})\in[k]^{d_c}$，它返回一个计数器 $\tau(i_1,\ldots,i_{d_c})=\sum_{u\in\mathrm{U}(x,r)}u(i_1,\ldots,i_{d_c})$。

算法。在定义新的预言机 $\mathrm{T}.\text{inactive}(x,r)$ 和$\mathrm{T}.\overline{\text{count}}(x,r)$ 之后，我们继续进行用于设置权重的主算法。最初，我们对每个 $c\in C$ 设置 $\bar{\omega}(c)=0$。

与第 6 节算法的第一步一样，我们计算一个值 L，使得 $\frac{L^2}{9}\le\mu_{\mathrm{OPT}(\bar{\boldsymbol{q}}(\mathbf{D}))}(\bar{\boldsymbol{q}}(\mathbf{D}))\le\mu_{C}(\bar{\boldsymbol{q}}(\mathbf{D}))\le|\boldsymbol{q}(\mathbf{D})|^3\cdot L^2$。

对于每个 $i \in \{0, 1, \ldots, O(\frac{1}{\varepsilon}\log N)\}$，我们设置 $r = (1+\varepsilon)^i$，并对每个 $c' \in C'$ 重复以下步骤。我们调用函数 T .count(c', r)，对于每个 $(i_1, \ldots, i_{d_c}) \in [k]^{d_c}$，我们更新 $\bar{\omega}(p) = \bar{\omega}(p) + \tau(i_1, \ldots, i_{d_c})$；对于每个满足 $\pi_{\mathbf{A}_{(n)}}(p) = c'$ 的 $p \in C$，以及每个 $A^{(j)} \in \mathbf{A}(c)$，$\pi_{A^{(j)}}(p)$ 是 $C_{A^{(j)}}$ 中的第 i_j 个中心。然后我们调用 T .inactive(c', r)。

分析。令 ι: $\boldsymbol{q}(\mathbf{D}) \to C$ 为赋值函数，它根据我们的算法将 $\boldsymbol{q}(\mathbf{D})$ 中的每个元组映射到 C 中的一个中心。特别地，当元组 $t \in \boldsymbol{q}(\mathbf{D})$ 对某个唯一的 $p \in C$ 的 $\bar{\omega}(p)$ 作出贡献时，我们令 $\iota(t) = p$。根据定义，$\bar{\omega}(p) = |\{t \in \boldsymbol{q}(\mathbf{D}) \mid \iota(t) = p\}|$。

由于 $\mu_{C'}(\bar{\boldsymbol{q}}(\mathbf{D})) \le |\boldsymbol{q}(\mathbf{D})|^3 L^2$，因此有 $\phi(\pi_{\mathbf{A}_{(n)}}(t)\ C') \le |\boldsymbol{q}(\mathbf{D})|^{3/2} L$，且针对 r 考虑的最大半径大于 $|\boldsymbol{q}(\mathbf{D})|^{3/2} L$。因此，每个元组 $t \in \boldsymbol{q}(\mathbf{D})$ 至少被分配到 C 中的一个中心。

Lemma D.2. $\sum_{t \in \boldsymbol{q}(\mathbf{D})} \phi^2(t, \iota(t)) \le (1+\varepsilon)\mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$.

证明。令 $t \in \boldsymbol{q}(\mathbf{D})$ 为一个被分配到 C 中某个中心的元组，其半径为 $r > \frac{1}{9|\boldsymbol{q}(\mathbf{D})|^L}$，并令 $\boldsymbol{q}_1(\mathbf{D}) \subseteq \boldsymbol{q}(\mathbf{D})$ 为此类元组的集合。根据 RBBD 树的正确性，$\pi_{\mathbf{A}_{(n)}}(t) \notin \bigcup_{c' \in C'} \mathrm{B}(c', r/(1+\varepsilon))$，这意味着 $\phi(\pi_{\mathbf{A}_{(n)}}(t)\ \pi_{\mathbf{A}_{(n)}}(\iota(t))) \le (1+\varepsilon)^2 \phi(\pi_{\mathbf{A}_{(n)}}(t)\ C')$。此外，根据构造，对于每个 $A \in \mathbf{A}(c)$，$\pi_A(\iota(t)) \in C_A$ 是 $\pi_A(t)$ 在 C_A 中的最近中心。因此，$\phi(t, \iota(t)) \le (1+\varepsilon)^2 \phi(t, C)$。

现在令 $t \in \boldsymbol{q}(\mathbf{D})$ 为分配给 C for $r = \frac{1}{9|\boldsymbol{q}(\mathbf{D})|^L}$ 中某个中心的元组，并令 $\boldsymbol{q}_2(\mathbf{D}) \subseteq \boldsymbol{q}(\mathbf{D})$ 表示这些元组。于是，我们有 $\phi(t, \iota(t)) \le (1+\varepsilon)\frac{1}{9|\boldsymbol{q}(\mathbf{D})|^L}$。

综合这两种情况，我们有 $\sum_{t \in \boldsymbol{q}(\mathbf{D})} \phi^2(t, \iota(t)) = \sum_{t \in \boldsymbol{q}_1(\mathbf{D})} \phi^2(t, \iota(t)) + \sum_{t \in \boldsymbol{q}_2(\mathbf{D})} \phi^2(t, \iota(t)) \le (1+\varepsilon)^4 \sum_{t \in \boldsymbol{q}(\mathbf{D})} \phi^2(t, C) + (1+\varepsilon)^2 \frac{1}{81|\boldsymbol{q}(\mathbf{D})|^{L^2}} \le (1+2\varepsilon)^4 \mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$。令 $\varepsilon \leftarrow \varepsilon/12$，即得所需的界 $\sum_{t \in \boldsymbol{q}(\mathbf{D})} \phi^2(t, \iota(t)) \le (1+\varepsilon)\mu_{\mathrm{OPT}(\boldsymbol{q}(\mathbf{D}))}(\boldsymbol{q}(\mathbf{D}))$。 □

该算法的总运行时间为 $\tilde{O}(k^{d_c+1}N)$，因为它执行了 $\tilde{O}(k)$ 次预言机调用，每次调用均需 $\tilde{O}(k^{d_c}N)$ 时间来计算并更新计数器 $u(i_1, \ldots, i_{d_c})$。

E 对其他优化问题的扩展

利用前几节中的工具，我们展示如何为多样性和公平性中的各种优化问题实现高效算法。首先，我们展示如何实现一个预言机，用于计算给定点集 S 到 $\boldsymbol{q}(\mathbf{D})$ 中最远点的近似值。然后，˜我们展示一种 $O(kN)$ 时间的近似实现，用于关系型场景中广为人知的 Gonzalez 算法 [37]。Gonzalez 算法用于解决多样性和公平性中的各种问题。

我们先从一些定义开始。

Gonzalez 算法。Gonzalez 算法 [37] 是一种著名且强大的算法，可用于为多个组合优化问题（例如聚类和多样性问题）[59, 66]推导出常数近似算法。在标准计算环境中，该算法非常简单：设 P 为 n 个点在 $\mathbb{R}^d$ 中组成的集合，并设 $C = \emptyset$ 为一个初始为空、待返回的集合。Gonzalez 算法运行 k 次迭代。在每次迭代中，我们找到点 $\arg\max_{p \in P \setminus C} \phi(p, C)$ 并将其加入集合 C。Gonzalez 算法的朴素实现需要 $O(nk)$ 时间。在欧几里得空间中，运行时间可改进为 $O(n \log k)$。

定义 *E.1*（*Gonzalez* 算法的关系型实现）。给定数据库实例 D、一个连接查询 $\boldsymbol{q}$ 和一个正参数 k，目标是为 k 次迭代实现 Gonzalez 算法。也就是说，给定一个集合 S，其中 $|S| < k$，计算一个点 $p \in \boldsymbol{q}(\mathbf{D})$，使得 $\phi(p, S) = \max_{q \in \boldsymbol{q}(\mathbf{D}) \setminus S} \phi(q, S)$。

我们还定义了 Gonzalez 算法的 ε-近似实现。在 Gonzalez 算法的关系型 ε-近似实现的每次迭代中，都会将一个点 $p \in \boldsymbol{q}(\mathrm{D}) \setminus \mathrm{S}$ 加入 S，使得 $\phi(p, \mathrm{S}) \geq (1-\varepsilon) \max_{q \in \boldsymbol{q}(\mathrm{D}) \setminus S} \phi(q, \mathrm{S})$。

E.1 计算最远点

设 $S \subseteq \boldsymbol{q}(\mathrm{D})$ 是 $\boldsymbol{q}(\mathrm{D})$ 的一个子集，且满足 $|S| < k$。目标是计算一个点 $p \in \boldsymbol{q}(\mathrm{D}) \setminus \tilde{S}$，使得 $\phi(p, S) = \max_{t \in \boldsymbol{q}(\mathrm{D}) \setminus S} \phi(t, S)$。我们提出一种算法，可在 $O(kN)$ 时间内近似解决该问题；也就是说，我们计算一个点 $p \in \boldsymbol{q}(\mathrm{D}) \setminus S$，使得 $\phi(p, S) \geq (1-\varepsilon) \max_{t \in \boldsymbol{q}(\mathrm{D}) \setminus S} \phi(t, S)$，其中常数 ε 可以任意小。

高层思路。 高层思路是通过二分搜索来猜测 $\max_{t \in \boldsymbol{q}(\mathrm{D})} \phi(t, S)$，类似于第 5 节中的二分搜索。对于猜测的每个距离 r，我们使用 RBBD 数据结构检查球 $\bigcup_{s \in S} B(s, r)$ 是否覆盖了 $\boldsymbol{q}(\mathrm{D})$ 中的所有点。如果是，则继续二分搜索更小的 r。如果不是，也就是说，RBBD 树中仍有活跃节点，则继续二分搜索更大的 r。从最后一次 $\bigcup_{s \in S} B(s, r)$ 未覆盖 $\boldsymbol{q}(\mathrm{D})$ 中所有点的迭代开始，我们在 S 中添加 $\boldsymbol{q}(\mathrm{D})$ 中任意一个在 RBBD 树中没有非活跃祖先的点。该点到 S 的距离，将是 $\boldsymbol{q}(\mathrm{D})$ 中任意点到 S 的最大距离的一个良好近似。

算法。 设 $S \subseteq \boldsymbol{q}(\mathrm{D})$。我们按照第 4.3 节中的描述，构造 $\boldsymbol{q}(\mathrm{D})$上 RBBD 树 T 的根节点 root。

我们按照第 5 节中的方法执行二分搜索。唯一的区别是，如果 $\hat{r}$ 是由 [62], 返回的第 z 小距离，我们会在范围 $[\hat{r}, \sqrt{d}\hat{r}]$ 内再次执行二分搜索。注意，对于任意一对点 $t, q \in \mathbb{R}^d$, $||t-q||_\infty \leq ||t-q||_2 \leq \sqrt{d}||t-q||_\infty$。因此，如果我们以相对步长 $(1+\varepsilon)$ 对 $[\hat{r}, \sqrt{d}\hat{r}]$ 进行离散化，即 $Y_{\hat{r}} = \{\hat{r}(1+\varepsilon)^0, \hat{r}(1+\varepsilon)^1, \hat{r}(1+\varepsilon)^2, \ldots, \hat{r}(1+\varepsilon)^{\log(\sqrt{d})/\varepsilon}\}$，并最终在 $Y_{\hat{r}}$ 上执行二分搜索，则二分搜索最终会得到某个距离 r^*：算法在距离 r^* 处满足某一性质，但在距离 $(1+\varepsilon)r^*$ 处不满足同一性质。这与我们在附录 B.1 中执行的二分搜索有关。

令 r 为二分搜索返回的距离之一。对于每个点 $s \in S$，我们运行 T .inactive(s, r)。对每个 $s \in S$ 重复上述过程后，我们调用 T .rep()。如果它返回$\emptyset$，我们继续二分搜索 r的较小值；否则，我们继续二分搜索 r的较大值。在二叉树的下一次迭代中继续之前，我们修改 T，使其所有节点重新变为活动状态，并在其节点上设置初始代表点。在此二分搜索结束时，令 r' 为我们找到的、使 $p = \mathrm{T}.\mathrm{rep}() \neq \emptyset$ 的最后一个（最大）距离。我们返回 p。

正确性。

引理 E.2。 $\phi(p, S) \geq \frac{1}{(1+\varepsilon)^2} \max_{t \in \boldsymbol{q}(\mathrm{D}) \setminus S} \phi(t, S)$。

证明。该算法执行来自第 4.4 节的 $O(k)$ 个预言机。如果 r' 是二分搜索找到的、使 T .rep $\neq \emptyset$ 的最大距离，则有 $\phi(p, S) > r'$。

回忆一下，二分搜索（如附录 B 中所定义）是在 ℓ_∞ $\boldsymbol{q}(\mathrm{D})$ 的两两距离上执行的，然后以因子 $(1+\varepsilon)$ 对范围 $[r, \sqrt{d}r]$ 进行离散化。根据定义，在二分搜索的执行过程中，对于 $r'' = (1+\varepsilon)r'$，我们有 T .rep() $= \emptyset$；否则，r' 就不会是我们用 T .rep() $\neq \emptyset$ 找到的最大距离。根据第4节中 RBBD 树的性质，我们知道每个点 $q \in \boldsymbol{q}(\mathrm{D})$ 都是非活动的，即存在 $s \in S$，使得 $q \in \bigcup_{u \in \mathrm{U}(s, r'')} (\square_u \cap \boldsymbol{q}(\mathrm{D}))$。由于 RBBD 树对球的近似因子为 $(1+\varepsilon)$，因此对于每个 $q \in \boldsymbol{q}(\mathrm{D})$，都有 $\phi(q, S) \leq (1+\varepsilon)r''$。因此，$\max_{t \in \boldsymbol{q}(\mathrm{D}) \setminus S} \phi(t, S) \leq (1+\varepsilon)r''$。由此可得，$\overline{\phi(p, S)} > r' = \frac{r''}{1+\varepsilon} \geq \frac{1}{(1+\varepsilon)^2} \max_{t \in \boldsymbol{q}(\mathrm{D}) \setminus S} \phi(t, S)$。 □

注意到[1] $(1+\varepsilon/2)_2 \geq 1-\varepsilon$，因此如果我们设为 $\varepsilon \leftarrow \varepsilon/2$，就得到 $\phi(p, S) \geq (1-\varepsilon)\max_{t\in \boldsymbol{q}(\mathrm{D})\setminus S}\phi(t, S)$。

引理 E.3。该算法的运行时间为 $O\left(k(\varepsilon^{-d+1}+\log N)(N+\varepsilon^{-2}\log^2 N)\log N\right)$，概率至少为 $1-\frac{1}{N}$。

证明。二分搜索运行 $O(\log N)$ 次迭代。为了在二分搜索的每一步中计算距离，我们需要花费 $O(N\log N)$ 时间，如附录 B 所示。因此，二分搜索所需的所有值均可在 $O(N\log^2 N)$ 时间内计算得出。对于二分搜索的每一步，T 。非活动(s, r) 预言机以至少 $1-\frac{1}{N^{m+2}}$ 的概率运行 $O((\varepsilon^{-d+1}+\log N)(N+\varepsilon^{-2}\log^2 N))$，而 T 。rep() 的运行时间为 $O(1)$。因此，我们的算法的总体运行时间为$O(N\log^2 N+k(\varepsilon^{-d+1}+\log N)(N+\varepsilon^{-2}\log^2 N)\log N)$ $= O(k(\varepsilon^{-d+1}+\log N)(N+\varepsilon^{-2}\log^2 N)\log N)$，且概率至少为 $1-\frac{1}{N}$。□

定理 E.4. 给定一个无环查询 $\boldsymbol{q}$、大小为 N 的数据库实例 D 、任意小的常数参数 ε，以及一个集合 $S\subseteq \boldsymbol{q}(\mathbf{D})$，其中 $|S|<k$，则可以在 $O(kN\log^2 N+k\log^4 N)$ 时间内计算出一个点 $p\in \boldsymbol{q}(\mathbf{D})\setminus S$，使得 $\phi(p,S)\geq(1-\varepsilon)\max_{t\in \boldsymbol{q}(\mathrm{D})\setminus S}\phi(t,S)$，其成立概率至少为$1-\frac{1}{N}$。

从 $S=\emptyset$ 开始并重复执行定理 E.4 k 中的算法，我们可以在 $O(k^2N\log^2 N+k^2\log^4 N)$ 时间内，得到一个 ε- 近似的关系型场景下 Gonzalez 算法实现。接下来，我们展示一种更快的 Gonzalez 算法实现，该实现使用了定理 5.2 中的关系型 k-中心聚类算法。

E.2 关系型场景下 Gonzalez 算法的实现

˜我们提出了一种$O(kN)$执行方式，用于实现Gonzalez 算法的 ε近似。该实现将使我们能够在关系型场景中为多个多样性和公平性问题推导出更快的算法。

在标准计算环境中，下面的论证是已知的 [49]。设 $P\subset\mathbb{R}^d$ 和 $S\subseteq P$为针对 k'-中心聚类问题的（$2+\varepsilon$）近似算法返回的 $k'=O(k\varepsilon^{-d})$个中心的集合。在 S 上执行 k 步Gonzalez 算法，将返回在 P上的ε近似Gonzalez 算法实现。

在关系型场景中，给定参数 k，我们使用参数 $k'=O(k\varepsilon^{-d})$（而非 k）运行定理 5.2中的算法。令 $S\subseteq \boldsymbol{q}(\mathrm{D})$表示我们的算法在 $|S|=O(k\varepsilon^{-d})$下返回的中心集合，使得$\mathrm{u}_S(\boldsymbol{q}(\mathrm{D}))\leq(1+\varepsilon)\mathrm{u}^{\mathrm{OPT}(\boldsymbol{q}(\mathrm{D}))}(\boldsymbol{q}(\mathrm{D}))$，其概率至少为 $1-\frac{1}{N}$。最后，我们在 S 上运行Gonzalez 算法的几何实现 [33]，迭代 k次，并令S为返回的集合。根据 [33] 以及定理 5.2的正确性，我们得到S是在 $\boldsymbol{q}(\mathrm{D})$上对Gonzalez 算法的 ε实现的结果。获得S的运行时间为 $O(k\varepsilon^{-d}n\log^2 N+k\varepsilon^{-d}\log^4 N)$，其概率至少为 $1-\frac{1}{N}$（定理 5.2使用 $k'=O(k\varepsilon^{-d})$而非 k）。此外，Gonzalez 算法在 S 上的几何实现 [33] 的运行时间为 $O(k\varepsilon^{-d}\log k)$。总的来说，Gonzalez 算法的 ε实现的运行时间为 $O(k\varepsilon^{-d}n\log^2 N+k\varepsilon^{-d}\log^4 N)$，其概率至少为 $1-\frac{1}{N}$。

定理 E.5. 给定一个无环查询 $\boldsymbol{q}$、一个大小为 N 的数据库实例 D 以及任意小的常数参数 $\varepsilon\in(0,1)$，存在一个 ε-近似版本的 *Gonzalez* 算法，在$\boldsymbol{q}(\mathbf{D})$上运行 k步，耗时 $O(kN\log^2 N+k\log^4 N)$。该运行时间以至少$1-\frac{1}{N}$的概率成立。

E.3 多样性与公平性问题

多样性。 众所周知，Gonzalez 算法可用于为多样性和公平性领域的多个问题构造小型核心集 [44]。更具体地说，给定一个数据库实例

D 大小为 $|\mathbf{D}| = N$、一个连接查询 $\boldsymbol{q}$ 和一个正整数参数 k，目标是计算一个集合 $\mathrm{S} \subseteq \boldsymbol{q}(\mathbf{D})$，其大小为 $|\mathrm{S}| = k$，使得，

- $\min_{p,q \in \mathrm{S}} \phi(p, q)$ 被最大化（关系远程边 – RRE）。
- S 上最小生成树的权重（相对于欧几里得距离）被最大化（关系远程树 – RRT）。
- S 上旅行商问题的最小成本（相对于欧几里得距离）被最大化（关系远程环 – RRC）。
- $\sum_{p \in \mathrm{S}} \min_{q \in \mathrm{S} \setminus \{p\}} \phi(p, q)$ 被最大化（关系远程伪森林 – RRP）。
- S 的完美匹配的最小成本（相对于欧几里得距离）被最大化（关系远程匹配–RRM）。

对于所定义的每个问题，我们都运行定理 E.5 中的算法。然后，利用 [44], 的结果，我们得出下一个定理。

定理 E.6. 给定一个无环连接查询 $\boldsymbol{q}$、一个大小为 N 的数据库实例 D 以及一个参数k, 存在一种针对 $\boldsymbol{q}(\mathbf{D})$ 上的 *RRE*、*RRT* 和 *RRC* 问题的 $O(1)$-近似算法，其运行时间为$O(kN \log^2 N + k \log^4 N)$。近似因子和运行时间均以至少 $1 - \frac{1}{N}$ 的概率成立。此外，存在一种针对 $\boldsymbol{q}(\mathbf{D})$ 上的 *RRP* 和 *RRM* 问题的 $O(\log k)$-近似算法，其运行时间为 $O(kN \log^2 N + k \log^4 N)$。运行时间以至少$1 - \frac{1}{N}$ 的概率成立。

我们指出，RRE 问题已在 [7] 中得到研究。Agarwal 等人 [7] 为 RRE 问题设计了一种 $O(1)$-近似算法，该算法适用于 $\boldsymbol{q}(\mathbf{D})$（对任意无环连接查询 $\boldsymbol{q}$），运行时间为$\tilde{O}(\min\{k^2 N, kN + k^{d/2}\})$，其中 d 是属性总数，$d = |\mathbf{A}|$。我们的算法将运行时间缩短了乘法因子 k，或减少了加法项 $k^{d/2}$。

公平性。 有趣的是，Gonzalez 算法的 ε-近似实现可用于公平性问题。我们定义两个问题：关系公平远程边（RFRE）问题和关系公平远程团（RFRC）问题。不失一般性，假设存在一个关系，例如 R_1，除数值属性$\mathbf{A}_1$, 外，还具有一个分类属性 A'，其 $\mathrm{dom}(A')$ 为 $= \{c_1, c_2, \ldots\ldots, c_g\}$，其中 g 是不同取值的数量。令 $Q_c = \{t \in \boldsymbol{q}(\mathbf{D}) | \pi_{A'}(t) = c\}$，即属于组 $c \in \mathrm{dom}(A')$的 $\boldsymbol{q}(\mathbf{D})$中的元组。给定 g个正整数参数 $k_1, \ldots\ldots, k_g$，目标是计算集合 $\mathrm{S} \subseteq \boldsymbol{q}(\mathbf{D})$，使得对于每个 $c \in \mathrm{dom}(A')$，都有 $|\mathrm{S} \cap Q_c| \geq k_c$，并且

- $\min_{p,\ q \in \mathrm{S}} \phi(p,\ q)$ 最大化（RFRE）。
- $\frac{1}{2} \sum_{p,q \in \mathrm{S}} \phi(p, q)$ 被最大化（RFRC）。

公平远程边问题（文献中亦称公平最大最小问题）已在标准计算环境 [6, 49, 55] 中得到广泛研究。已知，对每个组的元素运行 Gonzalez 算法 $O(k\varepsilon^{-d})$ 次迭代（其中 $k = k_1 + \ldots k_g$），可为公平远程边问题构造一个大小为 $O(gk\varepsilon^{-d})$ 的核心集。随后，在该核心集上执行一种代价高昂的 $\frac{1}{2}$-近似算法 [6, 49, 55]。若目标是得到 $O(1)$-近似算法，则核心集的大小为 $O(gk)$。类似地，我们可以对每个不同的 $c \in \mathrm{dom}(A')$，在 Q_c 上运行定理 E.5 中的 ε-近似实现来构造核心集。结合定理 E.5 与 [6, 49, 55], 中的算法，我们得到如下结果。

定理 E.7。 给定一个无环连接查询 $\boldsymbol{q}$、一个大小为 N 的数据库实例 D 以及参数k_1、……、k_g，满足 $k = k_1 + \ldots + k_g$，则存在一种 $O(1)$-近似算法，可在 $\boldsymbol{q}(\mathbf{D})$ 上解决 *RFRE* 问题，运行时间为 $O(kN \log^2 N + k \log^4 N + T_{FRE}(gk^2))$，其中 T_{FRE} 是标准计算环境中针对 gk^2 个点上的公平远程边问题的 $O(1)$-近似算法的运行时间。该运行时间以至少 $1 - \frac{1}{N}$ 的概率成立。

此外，公平远程团（或公平最大和）问题也已在标准计算环境中得到研究 [52]。Mahabadi 和 Trajanovski [52] 构造了一个大小为 $O(\sum_{i\in[g]} k_i^2) = O(gk^2)$ 的核心集，方法如下。他们在第 i 组的点上运行 Gonzalez 算法 k_i 次迭代。随后，对于 Gonzalez 算法返回的每个点，他们将第 i 组点中最近的 k_i 个邻居加入核心集。

在关系型场景中，对于每个 $c \in \text{dom}(A')$，我们首先在 Q_c上运行定理 E.5 中 Gonzalez 算法的 ε-近似实现。然后，对于定理 E.5 返回的每个点 p，我们使用 [7] 中介绍的基于欧几里得距离的预言机，在 Q_i 中查找最近的 k_i 个邻居，运行时间为 $O(N + k_i \log N)$。综合运用定理 E.5, [52], 和 [7],，我们得到以下结果。

定理 E.8. 给定一个无环连接查询 $\boldsymbol{q}$、大小为 N的数据库实例$\mathbf{D}$ 以及参数k_1、...、 k_g，满足 $k = k_1 + \ldots + k_g$，则存在一个 $O(1)$-近似算法，可在 $O(kN\log^2 N + k\log^4 N + T_{FRC}(gk^2))$ 时间内解决 $\boldsymbol{q}(\mathbf{D})$ 上的 *RFRC* 问题。其中， T_{FRC}是标准计算环境下针对 gk^2 个点上的公平远程团问题的 $O(1)$-近似算法的运行时间。该运行时间以至少 $1 - \frac{1}{N}$ 的概率成立。

F 循环查询的扩展

连接查询 $\boldsymbol{q}$ 的分数边覆盖是一个点 $x = \{x_R \mid R \in \mathbf{R}\} \in \mathbb{R}^m$，满足对于任意属性 $A \in \mathbf{A}$，$\sum_{R\in\mathbf{R}_A} x_R \geq 1$，其中 $\mathbf{R}_A$ 是 $\mathbf{R}$ 中所有包含该属性 A 的关系。正如所示，连接查询 $\boldsymbol{q}$ 的最大输出规模为 $O(N^{\|x\|_1})$。由于上述界限对任意分数边覆盖都成立，我们将 $\rho^\star = \rho^\star(\boldsymbol{q})$ 定义为具有最小 ℓ_1-范数的分数覆盖。等价地， $\rho^\star(\boldsymbol{q})$ 是线性规划（LP）最优解的目标函数值：$\min \sum_{R\in\mathbf{R}} x_R$， s.t. $\forall R \in \mathbf{R}$： $x_R \geq 0$ 且 $\forall A \in \mathbf{A}$： $\sum_{R\in\mathbf{A}_R} x_R \geq 1$。

接下来，我们给出广义超树分解（GHD）的定义。 $\boldsymbol{q}$ 的一个GHD是一个二元组$(\mathrm{H}\ \lambda)$，其中H是一棵作为有序节点集的树，且 $\lambda : \mathrm{H} \to 2^{\mathbf{A}}$ 是一个标记函数，它将每个顶点 $u \in \mathrm{H}$关联到$\mathbf{A}$中的一个属性子集，称为 λ_u，并满足以下条件：i)（覆盖性）对于每个 $R \in \mathbf{R}$，存在一个节点 $u \in \mathrm{H}$，使得$\mathbf{A}_R \subseteq \lambda_u$，其中$\mathbf{A}_R$ 是 R中所包含的属性集合；ii)（连通性）对于每个 $A \in \mathbf{A}$，节点集$\{u \in \mathrm{H} : A \in \lambda_u\}$构成H的一棵连通子树。

给定一个连接查询 $\boldsymbol{q}$、它的一个GHD $(\mathrm{H}\ \lambda)$以及一个节点 $u \in \mathrm{H}$， u的宽度定义为其导出超图$(\lambda_u, \mathrm{E}_u)$的最优分数边覆盖数，其中$\mathrm{E}_u = \{\mathbf{A}_R \cap \lambda_u : R \in \mathbf{R}\}$。给定一个连接查询和一个GHD $(\mathrm{H}\ \lambda)$，$(\mathrm{H}\ \lambda)$的宽度定义为H中所有节点上的最大宽度。于是，连接查询的分数超树宽度如下：连接查询 $\boldsymbol{q}$的分数超树宽度，记为$\mathrm{fhw}(\boldsymbol{q})$，为$\mathrm{fhw}(\boldsymbol{q}) = \min_{(\mathcal{H},\lambda)} \max_{u\in\mathcal{H}} \rho^\star(\lambda_u, \mathrm{E}_u)$，即所有GHD上的最小宽度。

用于 k中心/中位数/均值聚类的关系算法。 总体而言， $O(N_{\mathrm{fhw}})$ 是 H 中每个节点所物化的连接结果数量的上界，也是计算 H 中每个节点的连接结果所需的时间复杂度 [15]。因此，我们将原始有环查询转换为一个无环连接查询（其连接树为 H），其中每个关系包含 $O(N_{\mathrm{fhw}})$ 个元组。我们对新的无环连接查询执行定理 5.2 和定理 6.6 中的算法（以及定理 E.6、定理 E.7 和定理 E.8 中的算法）。近似因子保持不变。运行时间则通过将 N 因子替换为 N^{fhw} 因子进行调整。