连接并集上的采样

Yurong Liu*罗切斯特大学
yliu217@u.rochester.edu

Yunlong Xu*罗切斯特大学
yxu103@u.rochester.edu

Fatemeh Nargesian 罗切斯特大学
fnargesian@rochester.edu

arXiv:2303.00940v2 [cs.DB] 9 Mar 2023

摘要

数据科学家经常利用多个关系数据源进行分析。在学习和近似查询回答中，一个标准假设是，数据是从底层分布中抽取的均匀独立样本。为避免连接和并集的开销，给定一组连接，我们研究如何在不执行全连接和并集的情况下，从这些连接的并集中获取随机样本。我们提出了一个通用框架，用于对链式、无环和循环连接的集合并集进行随机抽样，并保证样本的均匀性和独立性。我们研究了连接并集大小评估这一新问题，并提出了两种近似方法：基于列直方图的方法和基于数据随机游走的方法。我们提出了一种在线并集采样框架，该框架以易于计算的参数近似值初始化，并在抽样过程中动态细化这些近似值。我们在 TPC-H 基准测试的工作负载上评估该框架，并探讨并集近似准确度与采样效率之间的权衡。

PVLDB 引用格式：
Yurong Liu、Yunlong Xu 和 Fatemeh Nargesian。《连接并集上的采样》。PVLDB，14(1)：XXX-XXX，2020。doi:XX.XX/XXX.XX

PVLDB 工件可用性：
源代码、数据和/或其他工件已发布于
https://github.com/DataIntelligenceCrew/sample-union-joins.git。

1 引言

数据科学家通常会利用多个数据源来收集训练数据。由于规范化，大多数关系数据源不会以单个表的形式存储，因此用户通常需要先执行连接，再基于连接结果进行学习 [11]。此外，数据可能来自分布式数据源，每个数据源都可描述为对内部数据库、数据湖或网络数据执行连接 [3, 12]。因此，目标数据是连接结果的并集。

连接的成本很高，而连接后的学习由于存在经规范化可避免的冗余，会导致训练性能不佳。已有一些研究尝试实现连接上的学习，但仅限于某些模型（包括线性回归），无法应用于更通用的模型 [11, 22, 23, 35]。幸运的是，学习理论中的一项重要结果 [36] 表明，学习和近似查询回答 [26] 并不需要完整结果，独立同分布（i.i.d.）样本即可实现有界误差。

*这些作者贡献相同 本作品采用知识共享署名-非商业性使用-禁止演绎 4.0 国际许可协议授权。访问 https://creativecommons.org/licenses/by-nc-nd/4.0/ 查看许可协议副本。如需进行超出许可协议范围的使用，请发送邮件至 info@vldb.org 获取许可。版权归所有者/作者所有。出版权授权给 VLDB 基金会。《VLDB基金会会刊》，第 14 卷，第 1 期 ISSN 2150-8097。
doi:XX.XX/XXX.XX

J_W： **客户_W**$(\cdots)$ $\bowtie$ **零件1_W**$(\cdots)$ $\bowtie$ **供应商1_W**$(\cdots)$ $\bowtie$**零件供应1_W**$(\cdots)$ $\bowtie$ **零件供应2_W**$(\cdots)$ $\bowtie$**行项目1_W**$(PartKey1, \cdots)$ $\bowtie$ **订单1_W**$(OrderKey1, \cdots)$ $\bowtie$**订单2_W**$(OrderKey1, \cdots)$ $\bowtie$ **行项目2_W**$(PartKey1, \cdots)$

$\bigcup J\_E$： **客户_E**$(\cdots)$ $\bowtie$ **零件供应_E1**$(\cdots)$ $\bowtie$**零件供应_E2**$(\cdots)$ $\bowtie$ **DoubleOrders_E**$(OrderKey1, OrderKey2, \cdots)$ $\bowtie$**行项目_E1**$(OrderKey1, \cdots)$ $\bowtie$ **行项目_E2**$(OrderKey2, \cdots)$

$\bigcup J\_MW$： **客户_MW**$(\cdots)$ $\bowtie$ **双重零件供应商_MW**$(\cdots)$ $\bowtie$**双重订单行项目_MW**$(\cdots)$

图 1：在非规范化 TPC-H 上连接的并集示例。

这一结果适用于多个模型。因此，在从多个数据源收集数据时，需要思考的问题是：如何在不执行连接和并集的情况下，从数据源的并集中获取样本。给定一组连接，目标是从连接结果的并集中独立且随机地返回 N 个元组的样本。

示例 1。 假设某在线零售公司的数据科学家想要训练一个模型，以便为客户未来的捆绑订单应用促销。为此，数据科学家需要从底层分布中随机且独立地抽取大小为 k 的客户数据及其捆绑购买样本。假设客户订单数据存储在多个数据库中，每个数据库都有自己的模式。例如，该公司可能在东部、西部和中西部地区分别设有一个供应商数据库。获取客户捆绑数据需要为每个地区的数据库构造一个查询，如图 *1* 所示，然后对查询结果取并集。由于没有一个单一关系包含所有所需特征，这些查询需要连接来自多个关系的数据。注意， J_W 是循环连接，而 J_E和 J_{MW} 是无环的。在 J_W 中，对关系订单 进行自连接，以获取同一订单中商品的信息（捆绑购买）。这三个连接具有相同的输出模式。构建目标数据集时，首先面临的挑战是，尽管其中一些查询是在高度非规范化的关系（或视图）上执行的，例如 J_E中的零件供应商关系，但由于某些基关系（例如行项目和订单）非常庞大，执行全连接的代价会非常高。

自 20 世纪 90 年代以来，针对单个连接进行随机抽样的问题一直受到积极研究 [1]。其目标是在不执行全连接的情况下，从连接中获得一个随机且独立的样本 J，使得样本中每个元组的概率均为 $1/|J|$。一种解决方案是对基关系进行连接样本抽样，以获得连接元组样本 [1]。然而，连接样本会产生数量远少于样本数的连接元组。此外，研究表明，所获得的连接样本并不能保证独立性 [18]。对于近似查询回答，一些技术

例如 RippleJoin [16] 和 WanderJoin [25] 分别设法使用非随机/独立样本和随机/相关样本。其他针对连接进行抽样的技术则采用接受-拒绝抽样范式，以保证独立同分布（i.i.d.）[10, 33]。Zhao 等人最近的工作提出了一个针对单个连接进行抽样的框架，可以处理一般的多路连接 [38]。对连接进行随机抽样的动机与连接大小估计密切相关；由于连接大小估计可应用于查询优化，因此它也一直是数据库领域关注的重点 [4, 10, 34]。

示例 2。接续示例 1，第二个挑战是对连接样本取并集，同时保证均匀性，即每个元组进入最终样本的概率为 $\frac{1}{|J_W \cup J_E \cup J_{MW}|}$。一种简单的解决方案是对以离线方式获得的连接样本取并集。假设我们应用现成的连接采样算法，分别从 J_W、J_E 和 J_{MW} 中获得样本 S_W、S_E 和 S_{MW}。我们有 $P(t \in S_W) = 1/|S_W|$，$P(t \in S_E) = 1/|S_E|$，$P(t \in S_{MW}) = 1/|S_{MW}|$，$U = J_W \cup J_E \cup J_{MW}$，$P(t' \in U) = 1/|U|$，和 $P(t'' \in U) = 1/|U|$。很容易证明，这无法保证均匀性，而且元组出现在 U 中的概率并不相等。考虑反例 $r \in S_E$、$r \notin S_W$、$r \notin S_{MW}$，我们有 $P(r \in U) = 1/|S_E|$；然而，如果 $r \in S_E \cap S_W \cap S_{MW}$，我们得到 $P(r \in U) = (\frac{1}{|J_E|} + \frac{1}{|J_W|} + \frac{1}{|J_{MW}|}) \cdot \frac{|S_E \cap S_W \cap S_{MW}|}{|U|}$，因为我们执行集合并集，并且对重叠元组只保留一个实例。接受/拒绝采样算法可以帮助调整这一概率，使其达到 $1/|U|$；然而，正如我们在 §2 中所示，该算法需要预先知道每个连接及其并集的大小，这就要求知道 J_E、J_W 和 J_{MW}的所有组合的重叠大小。一个想法是根据样本估计重叠和并集。然而，这并不可行，因为与连接样本或关系类似，从连接的重叠区域获得样本的概率很低。

本文提出了一个用于连接并集上随机抽样的通用框架。具体而言，我们考虑有放回的集合并集采样。从不相交并集中抽样是集合并集的一种直接扩展。经典的连接抽样 [10, 34] 以及近期重新审视的框架 [38]，研究的都是对连接进行有放回随机抽样。另一个相关问题是对无环合取查询并集的结果进行随机枚举 [8]。随机查询结果枚举算法的中间结果可视为从并集中无放回抽样得到的随机样本，这与我们的问题不同。此外，本文研究了更广泛类别连接（链式、循环和无环）上的并集采样。在 § 3.2 中，我们对这类研究与我们的框架进行了详尽讨论和分析比较。

要解决连接并集采样问题，面临着若干挑战。首先，对多个连接的随机样本取并集并不能保证均匀性。我们的解决方案是一种接受/拒绝采样算法，它定义了用于选择连接的伯努利和非伯努利概率分布。后者模拟了并集计算的行为。其次，事实证明，要保证均匀性，采样框架需要事先知道每个连接的大小以及连接并集的大小。尽管集合并集大小近似 [2, 7, 21]及其面向数据流的在线扩展 [9, 14, 21, 28] 已在近似计数领域得到广泛研究，但据我们所知，尚无研究探讨如何在不执行全连接、也不计算重叠的情况下，近似计算连接的并集大小。

第三，基于直方图的估计需要获知连接幂集中的指数数量个连接集合之间的重叠情况。我们通过将问题重新表述为使用更小单位的统计量来缩减计算空间，这种统计量称为每个连接的 k-重叠，表示某个连接结果中与恰好（$k-1$）个其他连接共享的子集大小。接下来，我们提出该框架的两种实例化方法，用于估计任意数量关系以及所有连接类型（链式、循环和无环）下的连接重叠：一种基于直方图的方法和一种随机游走方法。基于直方图的技术成本低，只需了解有限的连接统计信息。在某些情况下，它可能产生较宽松的界，从而导致较高的拒绝率。基于直方图的方法非常适用于野外数据，或数据市场等场景；在这些场景中，可用的元数据有限，且无法访问全部数据。随机游走方法能够准确估计参数，并实现较低的延迟。它需要通过抽样对参数进行预热，并提供理论保证。为平衡参数估计成本与采样效率之间的权衡，我们提出一种在线并集采样算法：使用基于直方图的方法初始化参数，并使用随机游走方法更新参数，同时重用随机游走过程中获得的样本并确保均匀性。

在本文中，我们作出以下贡献：

- 我们提出了对连接的并集进行随机抽样的问题。
- 我们设计了一个框架，用于对链式、循环和无环类型的连接的并集进行抽样（第 2 节）。该框架的任何实例化始终能从完整结果中返回均匀独立样本（定理 1），但采样效率各不相同（第 6.2 节）。
- 我们设计了基于直方图（第 4 节）和随机游走（第 6 节）的方法，用于界定任意链式、无环或循环连接集合的重叠大小。
- 我们提出了一种在线并集采样技术，用于平衡延迟与预热成本之间的权衡（第 6.2 节）。
- 我们使用 TPC-H 基准测试进行了广泛的实验评估，以研究参数估计和采样方法的误差与运行时间（第 9 节）。我们还评估了框架相对于关系大小、样本数量和重叠大小的可扩展性。

2 问题定义

设 A 为属性的全集，A_i 为关系 J_i 中的属性。给定一组连接 $S = \{J_1, \ldots\ldots, J_n\}$。连接 J_j定义为 $J_j = R_{j,1} \bowtie_{A_{j,1}} R_{j,2} \bowtie_{A_{j,2}} \cdots \bowtie_{A_{j,n_1-1}} R_{j,n_1}$，其中 $R_{j,1}, \cdots R_{j,n_1-1}$ 为基关系。与关系代数类似，我们假设所有连接在执行后都具有相同的输出模式，即属性的数量和名称相同。请注意，连接的长度和所涉及的关系仍可能不同。我们还假设连接属性已标准化为相同的名称。仅在必要时提及属性名称。在关系代数中，有两种并集：集合并集和不相交并集。前者会从并集结果中消除重复元组，后者则会保留重复元组。可并性的概念 [31]

可应用于基关系，以对齐属性，使连接具有相同的模式。

从连接的并集中进行抽样的问题，是以 $1/|union(J_1, \cdots J_n)|$ 的概率返回每个元组，其中并集可以是集合并集或不相交并集。通常仅返回一个抽样元组是不够的，因此，我们希望持续生成完全独立的抽样元组，直到达到所需的样本大小 N。我们将集合并集和不相交并集的抽样问题表述如下。

定义 1.（连接的不相交并集采样）给定一组连接 $S = J_1 \quad J_n \quad N \quad V = \{, \ldots, \}$，$J_1 \uplus \ldots \uplus J_n$ 从中返回独立样本，使得每个抽样元组被返回的概率为 $\frac{1}{|V|} = \frac{1}{J_1+\ldots+J_n}$。

从不相交并集中抽样很简单。给定不相交并集 $V = J_1 \uplus \ldots \uplus J_n$，我们首先以概率 $P(J_j) = \frac{|J_j|}{|J_1+\ldots+J_n|}$ 选择一个连接 J_j，然后从 J_j 中随机选择一个元组。这意味着每个抽样元组 t 的概率为 $P(t) = \frac{|J_j|}{|V|} \cdot \frac{1}{|J_j|} = \frac{1}{|V|}$。我们重复这一过程，直到获得 N 个抽样元组。由于无论之前的采样迭代如何，返回的样本始终是均匀的，因此该算法总是返回独立样本。从单个连接中抽取元组的方法长期以来一直是一个广受关注的问题 [10, 10, 33, 37, 38]。我们将在 § 3.2 中重新探讨连接上的随机采样。

集合并集操作会从并集结果中删除重复元组。因此，对集合并集执行独立同分布抽样的算法，应以集合并集大小的概率返回元组全集中的每个元组。

定义 2.（连接的集合并集采样）给定一组连接 $S = \{J_1, \ldots, J_n\}$，令 $\mathcal{U}$ 为 $U = J_1 \cup \ldots \cup J_n$ 中唯一元组构成的离散空间。返回 N 个来自 $\mathcal{U}$ 的独立样本，使得每个抽取的元组都以概率 $\frac{1}{|J_1 \cup \cdots \cup J_n|}$ 被返回。

3 AUNION 采样框架

令 U 为连接的集合并集中元组构成的全集。我们假设每个连接中都没有重复项。给定集合并集 $U = \bigcup_{j=1}^{n} J_j$，对于每个值 $u \in U, P(t = u) = \frac{1}{|U|}$，我们都希望如此。

示例 *3.* 考虑具有相同输出模式的连接 J_1 和 J_2。假设 $t_1 = (3, 6, 4) \in J_1$ 和 $t_2 = (3, 6, 4) \in J_2$。每个元组 t 的值，即 $t.val$，都可以按照标准约定将其属性值拼接起来得到。根据集合的定义，t_1 和 t_2 指的是全集 $U = J_1 \cup J_2$ 中的同一个元组，不妨称其为 u。我们希望 $P(u)$，即从 U 中选出值为 u 的元组的概率，等于 $\frac{1}{|U|}$。元组 t_1 和 t_2 分布在不同的连接中。因此，如果从各自对应的连接中抽取 t_1 或 t_2，就会得到 u。也就是说，我们希望 $P(t = u) = P(t_1) + P(t_2) = \frac{1}{|U|}$。请注意，我们可能会进行有放回抽样，也可能在样本中同时得到 t_1 和 t_2。无论我们是否选择移除重复项，我们的框架都能保证 $P(t = u) = P(t_1.val) = P(t_2.val) = \frac{1}{|U|}$。

在每次采样迭代中，该框架执行两个步骤：连接选择和连接随机采样。该框架以 $1/|U|$ 概率持续进行有放回的元组抽样，直到达到所需的样本大小 N。一种直接的方法是基于并集技巧 [15]。在每次迭代中，我们遍历所有连接，并以伯努利概率 $P(J_j) = |J_j|/|U|$ 选择一个连接。这意味着每次迭代中可能会选择多个连接。选择 J_j 后，我们从 J_j 中随机抽样一个元组 t，且采用有放回抽样。回忆一下，$u = t.val$ 表示元组 t 的值。对于具有重复值 u 的元组，仅当它们是从相同的连接中抽样得到时，我们才接受这些元组；否则，我们接受这些元组。这意味着，只有当重复元组 t 是从首次观测到 $u = t.val$ 的连接中抽样得到时，才会保留该元组。根据上述描述，只有先以 $|J_j|/|U|$ 的概率选择一个包含 u 的连接 J_j，再以 $1/|J_j|$ 的概率抽样 J_j，才会返回元组值 $u \in U$。这保证了每个值 $u \in U$ 都以 $\frac{|J_j|}{|U|} \cdot \frac{1}{|J_j|} = \frac{1}{|U|}$ 的概率被返回。尽管该算法简单，但对于高度重叠的连接，其拒绝率很高，并且可能导致较高的延迟。这是因为采用了两阶段框架，而该框架对于确保抽样的均匀性至关重要。接下来，我们将介绍一种连接选择算法，以更加谨慎的方式选择连接。在第§ 7节中，我们提出一种新方法，利用第一阶段执行的计算来降低第二阶段的延迟。

3.1 非伯努利连接选择

上述技术仅当重叠区域中的样本恰好来自一个预先指定的连接时，才会保留这些样本。考虑图 2a 中两个连接 J_1 和 J_2，它们的数据区域 B 存在重叠。我们选择并保留任意样本 $t_1 \in J_1$。之后，在选择 J_2 时，如果我们抽到一个 $t_2 \in B$，就会拒绝 t_2。避免拒绝的诀窍是，只保留 B 中来自 J_1 的部分，并将其作为我们从 J_1 抽样的唯一空间。因此，我们有 $P(J_1) = \frac{|A+B|}{|A+B+C|} = \frac{|J_1|}{|U|}$，以及 $P(J_2) = \frac{|C|}{|A+B+C|} = \frac{|J_2|-|B|}{|U|}$。

我们的连接选择过程如算法1所示。在抽样之前，算法需要确定从哪个连接中限制性地抽取每个重叠区域。我们将连接的这种划分称为<样式 id='1'>覆盖 （算法1第2行）。对连接 $S = \{J_1, \cdots J_n\}$ 的覆盖，即 $C = J_1, \quad J_n \quad S \quad J_i = t \in \{, \cdots, \}$，$J_i | t \notin \bigcup_{j<i} J_j \quad J_i \quad J_i$ 是一种排序，满足 $, \{, \}$。实际上，连接的覆盖 $,$ 是连接 J_i 上的一个选择查询。可以从第一个连接开始，保留或移除重叠部分，从而创建 S 的覆盖。图 2b 展示了三个重叠连接的覆盖示例。给定覆盖 C，要计算 $J_{,i}$ 的大小，只需遵循容斥原理。令 $O_\Delta = \bigcap_{J_j \in \Delta} J_j$ 和 S_i 分别表示在 C 给出的排序中出现在 J_i 之前的连接集合，则有如下结果。

$$|J_i'| = |J_i| + \sum_{m=1}^{i-1} \sum_{\Delta \subset S_i, |\Delta|=m} (-1)^m |O_\Delta \cup \{J_i\}|$$

基于此覆盖，每个 J_i 以 $P(J_i) = \frac{|J_{,i}|}{|U|}$ 被选中。抽样时，我们应始终遵循预先定义的覆盖，即对于任意样本 $t \in J_i$，若 $t \notin J_{,i}$，就应将其丢弃。然而，如果我们事先没有重叠信息，那么在选定 J_i 并对 t 进行抽样后，便无法验证 t 是否属于 $J_{,i}$。因此，在对 $t \in J_i \setminus J_{,i}$ 进行抽样时，我们会遇到一个非平凡情形。如果之后从 J_j 中对 t 进行抽样，且满足 $J_{,j} \cap J_i \neq 0$，也就是说，$J_{,j}$ 覆盖了与 J_i 重叠的部分，那么我们应执行一项关键操作，称为修订。这意味着从样本中移除 $t \in J_j$ 并重新抽取 J_j，同时保留来自 J_i 的 t。

示例 4。 考虑图 *2b* 中的连接 J_1、 J_2 和 J_3。这些连接的覆盖分别以蓝色、红色和绿色突出显示。算法分别以概率 $|J'_1|/|U|$、 $|J'_2|/|U|$ 和 $|J'_3|/|U|$ 选择 J_1、 J_2 和 J_3。假设在某次迭代中，我们已选择 J_2 并从 $t \in J_2 \setminus J'_2$ 中抽取了样本。现在假设我们选择 J'_1 并从同一个 t 中抽取样本。由于覆盖告知我们只能从 J'_2 区域中抽取 J_2，因此我们从目标集中移除 *t's*，接受从 J_1 中抽取的 t，并将其分配给记录中的 J_1。

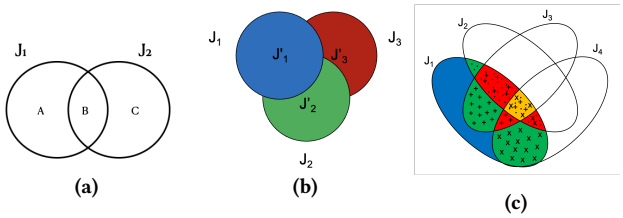


图 2：（a）并集操作，（b）三个连接的覆盖，以及（c）四个连接的 $\frac{k}{j}$。

定理 1。 给定连接 $S = \{J_1, \ldots, J_n\}$，算法*1*以概率 $\frac{1}{|J_1 \cup \ldots \cup J_n|}$ 返回值为 u 的每个结果元组 t。

证明。直观而言，由算法1定义的覆盖决定了从一组连接的重叠部分中抽取某个值时，专门从哪个连接抽取。回顾一下，U 是连接元组集合并集的全集，即 $\{u | u \in \cup_i J_i\}$。算法1使用映射策略函数 f: $U \to S$，它告知我们特定的 u 被分配给哪个 J_i。请注意，u 可能属于多个 J_i，但 f 指的是可以抽取 u 的唯一 J_i。设 S 的覆盖 C 为 U 关于 f 的商空间，并且 g: $S \to C$ 是一个映射函数，使得 $g(J_i) = J'_i$。那么，$g \circ f$ 会将每个 $u \in U$ 映射到覆盖 C 中的一个连接。对于所有 u，我们记 $|g(f(u))|$ 为 $|\{u' | g(f(u')) = g(f(u))\}|$。换言之，抽取一个 $u \in U$ 的概率取决于选取 $g(f(u))$ 的概率，以及随后从 $g(f(u))$ 中抽取 u 的概率。因此，我们得到 $P(t = u)$ 的概率如下。

$$P(t = u) = P(f(u)) \cdot \frac{1}{|g(f(u))|} = \frac{|g(f(u))|}{|U|} \cdot \frac{1}{|g(f(u))|} = \frac{1}{|U|}$$

□

计算算法1第 2 行的概率分布，需要了解 $|J'_i|$ 以及 $|U|$。在第 4 节中，我们将介绍估算 k 连接与 $|J'_i|$ 重叠的方法。

3.2 重新审视 JoinSampling

为对单个连接（算法1第7行）进行抽样，我们参考了 Zhao 等人 [38], 的工作，该工作是一个可从任意类型连接中进行抽样的通用框架。该框架定义了一个连接数据图，其中关系中的每个元组都是一个节点。每个元组 t 都带有一个权重标签，该权重被定义为其所产生的连接结果中元组数量的上界 t。该框架执行接受-拒绝抽样。根据权重以某种概率对关系中的每个元组进行抽样，并按照与权重相关的某个比例予以拒绝，以保证均匀性。我们作出了一些设计选择，以将 Zhao 等人的连接抽样框架作为我们并集抽样框架中的一个子程序。

算法1 并集采样

输入： 连接 $S = \{J_j, 1 \le j \le n\}$，元组数量 N
输出： 元组 $\{t_i, 1 \le i \le N\}$

1: $\{|J_j|, \ 1 \le j \le n\}$ $|U| \leftarrow warmup(S)$ ▷ 基于直方图（第 4 节）或随机游走（第 6 节）
2: $\{|J'_j|\} \leftarrow cover(S)$
3: $T \leftarrow \{\}$ ▷ 目标样本
4: $orig\_join_i \leftarrow \{\}$ ▷ 已见元组的原始连接记录
5: **当** $n < N$ **时执行**
6: 以 J_j 的概率选择 $\frac{|J'_j|}{|U|}$
7: $t \leftarrow$ 来自 J_j的一个随机样本
8: **如果** $t \in orig\_join_i$ 对于任意 $i < j$ **则**拒绝 t
9: **else**
10: **如果** $t \in orig\_join_i$ 对于任意 $i > j$ **则** ▷ 修订
11: 从 $orig\_join_i$ 移除 t 并将 t 添加到 $orig\_join_j$
12: 从 T移除所有 t's
13: **如果** $t \notin orig\_join_i$ **则** 添加 t 到 $orig\_join_j$
14: $T \leftarrow T \cup \{t\}$
15: **返回** T

首先，对于权重实例化，我们采用三种技术：扩展 Olken 算法、精确算法，以及 Zhao 等人提出的 Wander Join [25], [38]。其次，该框架需要基关系上的索引结构，以了解哪些元组可以连接。作为替代，我们使用关系的哈希表来维护元组的可连接性信息。第三，Zhao 等人框架的一个限制是，假设关系之间仅存在键-外键连接。由于在通用连接中，某些元组可能在其他关系中没有可连接的元组，我们通过修改扩展 Olken 算法来解除这一假设：通过在哈希表中进行额外的线性搜索，将这些元组的权重（从而其概率）设为零。

最后，为了获得接受/拒绝比率，该框架允许我们代入任意连接大小上界估计。我们还需要计算算法1中连接的大小上界。为此，接下来我们采用 Ngo 等人提出的算法中的部分内容，并扩展 Olken 算法 [33]，以计算任意数量的关系的连接大小上界。

假设一个连接 $J = R_1 \bowtie_{A_1} R_2 \bowtie_{A_2} \cdots \bowtie_{A_{n-1}} R_n$。令 $M_{A_i}(R_{i+1})$表示关系 R_{i+1}的属性 A_i 中的最大值频率。由于 R_2 中属性 A_i 取值为 v 的每个元组，最多可以在属性 A_i上与 R_{i+1}中的 $M_{A_i}(R_{i+1})$个元组匹配，因此连接 J的大小具有以下上界：$|J| \le |R_1| \cdot \prod_{i=1}^{n-1} M_{A_i}(R_{i+1})$。在我们的框架中，我们在所有算法中都采用上述对 Olken 算法的扩展来进行连接大小估计。

3.3 成本分析

由于算法1中从连接中抽样的子程序基于现有算法，因此在成本分析中，我们将连接上的随机抽样延迟与我们的算法解耦，并将从连接子程序获得的样本总数视为总成本。

定理 2。 给定连接 $S = \{J_1, \ldots, J_n\}$，算法*1*返回 N均匀独立样本的预期总抽样成本为 $N + N\log N$。

证明。给定覆盖 $C = \{J'_j \mid j \in [1,n] \cap \mathbb{Z}\}$，算法1以概率 $|J'_j|/|U|$ 对每个连接 J_j 进行抽样。令 N_j 表示来自 J_j 且包含在最终结果中的元组数量。根据算法1可知，当从 J'_j 获得某个元组时，该元组就会出现在最终样本中。因此，期望情况下有 $N_j = \frac{|J'_j|}{|U|} \cdot N$。令 ψ_j 表示算法1从 J_j 中获得的元组总数。由于不同连接中的元组集合可能相交，一个元组可能是被拒绝的样本、被接受的样本或修订后的样本。根据并集界，算法1的迭代次数不超过从每个连接中抽取的元组数量之和。利用这一原理，可得 $\psi \le \sum_{j=1}^{n} \psi_j$ 的预期总迭代次数。给定 N_j 张优惠券，优惠券收集问题给出了一个界：为了至少抽到每张优惠券一次，我们预期需要进行多少次有放回抽样 [30]。据此可得 $\psi_j = N_j \log N_j$ 的期望值。因此，预期迭代次数如下。

$$\psi \le \sum_{j=1}^{n} N_j \log N_j = \sum_{j=1}^{n} N.\frac{|J'_j|}{|U|} \log\left(N.\frac{|J'_j|}{|U|}\right)$$

令 $\alpha_j = \frac{|J'_j|}{|U|}$。我们有如下结论。

$$\psi \le \sum_{j=1}^{n} \alpha_j.N \log(\alpha_j.N) = N\left(\sum_{j=1}^{n} \alpha_j \log \alpha_j + \sum_{j=1}^{n} \alpha_j \log N\right)$$

根据覆盖的定义，我们知道 $\sum_{j=1}^{n} \frac{|J'_j|}{|U|} = 1$。因此，期望总时间满足以下界。

$$\psi \le N(\log(H(n)) + \log N) \le N + N \log N$$

□

值得注意的是，尽管我们的算法无法对样本间隔提供严格的确定性保证，但其总时间与 Carmeli 等人提出的用于随机枚举合取查询并集结果的算法的 $O(N \log N)$ 时间相当，其中 N 是答案的数量 [8]。

4 连接并集的大小

执行完整连接并计算集合并集的代价很高。我们提出了一种新颖的方法，通过连接的大小和连接重叠的大小来计算集合并集的大小。为此，我们首先将每个连接 J_j 划分为 n 个不相交部分，记为 $J_j = \bigcup_{k=1}^{n} \mathrm{A}_j^k$，其中 A_j^k 是 J_j 中第 k 个重叠所包含的元组集合；也就是说，A_j^k 中的每个元组都属于 J_j，且恰好出现在另外 $k-1$ 个连接中。基本情况 A_j^1 包含 J_j 中属于所有重叠的集合补集的元组。图2c展示了连接 J_1 的 A_j^k 区域。由于对于每个 J_j，各个 A_j^k 彼此不相交，我们可以如下定义集合并集 U 的大小。

$$|U| = \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{1}{k} |\mathcal{A}_j^k| \tag{1}$$

注意，A_j^k 是非平凡信息，需要结合连接的 k-组合的重叠大小才能得到。计算 A_j^k 有两项挑战。首先，以下两者之间没有关系：两两重叠信息与高阶 k 阶重叠 $\mathrm{A}_j^k (k > 2)$。其次，在没有全连接的情况下计算两两重叠大小，比计算单个连接大小更具挑战性。

假设我们有一种能够计算任意连接集合重叠的方法。更正式地说，给定一个连接集合 $\Delta \in S$，O_Δ 表示 Δ 中连接的重叠。在第 4 节和第 7 节中，我们介绍了用于所有连接类型（链式、循环和无环）的重叠估计算法。现在，我们转而关注如何使用 O_Δ 计算 A_j^k。下面通过一个示例介绍我们解决方案的直观思路。

示例 *5*。考虑图*2c*中的连接 $S = \{J_1, \cdots J_4\}$。$k \in [1,4]$ 对应的区域 A_1^k 采用颜色编码。我们希望计算 *A21* 的大小。虚线区域、+ 区域和 × 区域包含了所有两两重叠。假设我们首先计算具有 J_1 的连接的两两重叠大小之和，即 $\sum_{\Delta \in \mathbb{P}_2 \wedge J_1 \in \Delta} |O_\Delta|$，其中 $\mathbb{P}_2$ 是 S 中所有大小为 *2* 的子集的集合。然而，为了确定恰好与 J_1 个连接重叠的区域 *A21* 的大小，我们需要排除所有 *A31* 和 *A41* 区域。事实上，*A31* 中的每个子区域在上述总和中被计算了两次。例如，$J_1 \cap J_2 \cap J_3$ 同时位于 $J_1 \cap J_2$ 和 $J_1 \cap J_3$ 中。同样，*A41* 在 O_Δ 的总和中被计算了三次，因为它包含在 $J_1 \cap J_2 \cap J_3$、$J_1 \cap J_2 \cap J_4$ 和 $J_1 \cap J_3 \cap J_4$ 中。为避免重复计算，公式 *1* 中的 A_j^k 按 *1/k* 加权。

定理 3. 设 $S = \{J_1, J_2, \ldots J_n\}$ 和 $\mathbb{P}_k$ 都是大小为 k 的S的子集，则对于任意连接路径 J_j以及任意 $1 \le k \le n$，都有

$$|\mathcal{A}_j^k| = \sum_{\Delta \in \mathbb{P}_k \wedge J_j \in \Delta} |O_\Delta| - \Big(\sum_{r=k+1}^{n} \binom{r-1}{k-1} \cdot |\mathcal{A}_j^r| \Big).$$

对于 $k = n$，有 $|\mathcal{A}_j^n| = |O_S|$。对于 $k = 1$，有如下结论。

$$|\mathcal{A}_j^1| = \sum_{\Delta \in \mathbb{P}_1 \wedge J_j \in \Delta} |O_\Delta| - \sum_{r=2}^{n} \binom{r-1}{0} |\mathcal{A}_j^k| = |J_j| - \sum_{r=2}^{n} |\mathcal{A}_j^r|$$

证明。当 $k = n, \mathbb{P}_n$ 是表示包含 J_j 的全集 S 的集合。因此，显然 $|\mathrm{A}_j^n| = O_S$ 可以用 $\bigcap_{J_j \in S} J_j$ 计算。接着，对于 $k \in [2, n-1] \cap \mathbb{Z}$，我们动态计算 $|\mathrm{A}_j^k|$。现在，假设我们已知 $|\mathrm{A}_j^{k+1}|$。回想一下，A_j^k 由 J_j 中恰好出现在 $k-1$ 条其他连接路径中的所有元组组成。也就是说，J_j 中的元组属于某个 $\Delta \in \mathbb{P}_k$，但不属于任何更高阶重叠 $\Delta' \in \mathbb{P}_r$，其中 $r \in [k+1, n]$。因此，我们首先累加集合 $\Delta \in \mathbb{P}_k$ 的所有 k 阶重叠，其中 $J_j \in \Delta$。由于 J_j 已确认，这类集合的数量为 $\binom{n-1}{k-1}$ Δ。请注意，元组 $t \in \mathrm{A}_j^k$ 可能出现在多个 $\Delta \in \mathbb{P}_r$、$r \in [k+1, n]$ 中。因此，为了得到 $|\mathrm{A}_j^k|$ 的精确值，对于每个 $r \in [k+1, n]$，我们需要统计满足 $J_j \in \Delta$ 的 $\Delta \in \mathbb{P}_r$ 的数量。从 $r = k+1$ 开始，每个这样的 $\Delta \in \mathbb{P}_{k+1}$ 组合都包含 J_j，因此它会在剩余的 $\binom{k}{k-1}$ 个 $\Delta' \in \mathbb{P}_k$'s 中出现一次。因此，我们需要从总和中减去 $(k-1) \cdot |\mathrm{A}_j^{k+1}|$。现在考虑一般情形 r，其中 $k < r \le n$。在 J_j 确认后，每个 $\Delta \in \mathbb{P}_r$ 组合的其他 $k-1$ 条路径有 $\binom{r-1}{k-1}$ 种 $\Delta' \in \mathbb{P}_k$ 选择方式，因此，对于每个 r，需要从总和中减去总计 $\binom{r-1}{k-1} |\mathrm{A}_j^r|$。因此，我们可以按定理所示整理计算 $|A_j^k|$ 的公式。□

使用该定理计算给定 J_j 的 $|A_j^k|$'s 及所有 $k \in [1, n]$ 时，我们首先按照第 4 节提出的方法，用 $|O_S|$ 初始化 $|A_j^n|$。接着，计算 $|A_j^{n-1}|$ 需要评估已计算出的 $|A_j^n|$，以及 S 的每个大小为 $n-1$ 的子集对应的 $|O_\Delta|$。同样，使用第 4 节计算一个 $|O_\Delta|$。一般而言，从 $n-1$ 迭代到 1，可以根据已计算出的 $|A_j^r|$'s（其中 $r \in (k, n]$）以及可通过我们用于成对连接路径重叠的方法计算出的 $|O_\Delta|$'s，计算每个 $|A_j^k|$。

计算集合并集的大小，需要计算连接的所有 k-子集的重叠，这一过程相对于输入连接的数量呈指数级增长。值得指出的是，在实践中，输入连接的数量较少。然而，当 S 较大时，如果我们按照 S 的幂集格自底向上的遍历顺序计算 $|O_\Delta|$，就可以通过复用部分计算结果来加快计算。

预热阶段： 请注意，对于任意数量的连接和关系，计算 k-重叠及重叠的精确值在计算上代价高昂，甚至不可行。接下来，我们介绍该框架的两种实例化方案，用于近似这些参数。我们考虑两种情况：集中式和去中心化的 [18]。在集中式场景中，可以通过直接访问数据来获取关系，例如数据库中的关系。我们为这种场景提出了随机游走方法。在去中心化场景中，数据是私有的，或抽样代价高昂。例如数据市场中的数据，或数据库中的大型关系。基于直方图的方法适用于这种场景。该框架的不同实例化方案仅在预热阶段计算并集大小界和连接重叠界的方式上有所不同。值得指出的是，两种方法都能保证均匀性。在效率与估算成本之间存在权衡：更紧的上界设置成本更高，但一旦设置完成，便能更高效地生成样本。另一方面，较宽松的上界更容易计算，但会导致采样效率较低（因为拒绝率可能更高）。我们提出了一种基于随机游走方法的并集采样改进版本，无需预热，并能更好地权衡上界计算与采样效率。

5 使用直方图进行实例化

数据库管理系统通常会维护直方图，将其作为一种特殊的列统计量，以便在查询优化期间提供有关表列中数据分布的更详细信息。这些直方图有助于估计基数，尤其是在列中的数据分布不均匀时。本节介绍如何利用这些直方图，以及关系中元组的最大度等更简约的统计量，来估计连接重叠和并集大小。在此，我们提出一种针对链式连接的解决方案，其灵感来自 Olken 关于连接大小估计的开创性工作[34]。在 § 8.2 中，我们将框架扩展到更一般的循环连接和无环连接。

5.1 等长链式连接的重叠

我们首先估算多个链式连接的重叠。假设所有连接包含相同数量的关系，并且每对连接中的关系之间都存在一对一映射，使得映射后的关系具有相同的模式。给定一个连接集合 S 以及一个子集 $\Delta \subseteq S$，令 $O_\Delta = \bigcap_{J_j \in \Delta} J_j$ 为出现在所有 $J_j \in \Delta$ 中的元组集合。显然，重叠的一个宽松上界为 $\min\{|J_j| : J_j \in \Delta\}$。我们首先按照一致的关系顺序对连接进行划分。每一步中，我们根据较小子连接的重叠动态估算各子连接的重叠大小：将较小子连接的重叠大小乘以连接属性值的最大度的最小值。例如，对于包含三个关系的连接，我们首先计算所有连接中第一个关系的重叠。接着，将第一个关系的重叠乘以第一个关系中各值的最大度的最小值，计算所有连接中前两个关系的重叠，以此类推。

更正式地说，令 $K(i)$ 为第 i 次连接后重叠元组数量的上界。因此，$|O_\Delta| \le K(n-1)$。令 $M_{A_l}(R_j, i)$ 表示关系 R_j 中连接属性 A_l 的定义域内各值的最大度，i 表示连接 J_j 中的最大度；令 $d_{A_l}(v, R_j, i)$ 表示值 v 在 A_l 的定义域中的度。注意，连接属性的直方图中包含值度数的统计信息。我们可以动态地得到一个上界：$K(i) = K(i-1) \cdot \min_{J_j \in \Delta}\{M_{A_l}(R_j, i+1)\}$。注意，对于 $K1$，我们根据值来计算上界，即 $K(1) = \sum_{v \in C} \min_{J_j \in \Delta}\{d_{A_1}(v, R_j, 1) \cdot d_{A_1}(v, R_j, 2)\}$。到目前为止，这个上界需要所有连接中第一个关系的完整直方图，以及其余关系中各值的最大度。如果关系中所有连接属性都有直方图，我们还可以进一步收紧上界：将最大度的最小值项 $M_{A_l}(R_j, i+1)$ 替换为连接属性上各值平均度的最小值。

5.2 链式连接的重叠

现在，我们放宽这一假设，以适用于长度任意且关系模式任意的连接。请注意，连接后的模式仍应相同。我们引入拆分方法，旨在将连接重新组织为对大小相同的关系进行连接，从而能够应用 § 5.1 的结果。拆分方法通过将关系拆解为子关系来推导新的连接，每个子关系恰好包含两个属性。派生连接具有相同的模式且无损，也就是说，每个派生连接生成的数据都与原始连接相同，并且它们都包含相同数量的关系。此外，对于一个派生连接中的每个关系，其他连接中都存在与之对应的关系。由于派生连接满足 § 5.1 的要求且生成相同的数据，我们可以直接应用 § 5.1 来估计原始连接的重叠大小。尽管输入连接可能不包含模式相同的关系，但它们肯定具有对应的属性，并且连接后的模式相同。因此，将所有关系拆分为包含两个属性的子关系并重新定义连接，可得到包含相同数量、模式相同的关系的连接。

请注意，我们的拆分方法不同于数据库理论中的规范化。规范化旨在根据函数依赖将关系分解为子关系，以避免异常 [13]。拆分后的关系会保留其原始大小的记录，以供估计步骤使用。我们将从同一个原始关系拆分出的两个关系之间的连接称为伪连接。以下定理描述了一种界定链式连接重叠的通用方法。

定理 4。 给定一组分裂连接 S和一个子集 $\Delta \subset S$，令 $O_\Delta = \bigcap_{J_j \in \Delta} J_j$。令 $M_{A_l}(R_j,\ i)$为关系 $R_j,\ i$的连接属性 A_l定义域中各值的最大度，其中该关系属于连接 J_j；令 $d_{A_l}(v,\ R_j,\ i)$为定义域 A_l 中值 v 的度。我们定义如下。

$$M_{j,i} = \begin{cases} M_{A_i}(R_{j,i+1}) \ \ if\ R_{j,i} \bowtie R_{j,i+1} \\ \\ 1 \ \ if\ R_{j,i} \bowtie' R_{j,i+1} \end{cases}$$

设 K（i）为第 i 次连接后重叠元组数量的上界，设 d_{A_l}（v、R_j、i）为值 v在 A_l定义域中的度。由此，我们可以动态地得到 Δ、 $|O_\Delta|$中连接重叠大小的上界，具体如下。

$$|O_\Delta| \le \mathcal{K}(n-1) = \mathcal{K}(n-2) \cdot \min_{J_j \in \Delta}\{M_{j,n}\}$$
$$\mathcal{K}(1) = \sum_{v \in C} \min_{J_j \in \Delta}\{d_{A_1}(v, R_{j,1}) \cdot d_{A_1}(v, R_{j,2})\}$$
$$\mathcal{K}(i) = \mathcal{K}(i-1) \cdot \min_{J_j \in \Delta}\{M_{j,i}\}$$

证明。该定理的证明基于第 5.1 节和第 5.2 节。

□

我们指出，如果数据存在偏斜，定理 4 可能会成为连接重叠的有偏估计量。在此，我们提出一种仅需最少统计信息的解决方案。通过使用所有连接属性的直方图信息并计算重叠的期望值和上界，我们可以直接扩展该定理，使其成为无偏估计量。

6 使用随机游走进行实例化

第 4 节提出的技术以直接方式执行连接并集大小估计。在本节中，我们考虑一种在线估计连接重叠大小的替代方法，该方法更加准确。其思路是在预热阶段通过从连接路径中获取元组并在主要采样步骤中重用这些元组，动态更新连接大小和重叠大小。

6.1 重新审视连接大小估计

为解决连接上的在线聚合问题，Wander Join 提出了一种算法，通过在底层连接数据图上执行随机游走来实现 [25]。通过计算连接上的 COUNT 操作，此方案可用于连接大小估计。连接数据图将元组之间的连接关系建模为一张图，其中节点是元组，若两个元组可以连接，则它们之间存在一条边。借助连接图，我们可以通过执行随机游走轻松获得成功连接的元组。从连接中抽取的元组的概率可以利用连接图即时计算。给定一个连接 $J = R_1 \bowtie R_2 \bowtie \ldots \bowtie R_m$，结果元组 $t = t_1 \bowtie t_2 \bowtie \ldots \bowtie t_m$ 的概率计算为 $p(t) = \frac{1}{|R_1|} \cdot \frac{1}{|d_2(t_1)|} \cdot \ldots \cdot \frac{1}{|d_m(t_{m-1})|}$，其中 $d_i(t_{i-1})$ 表示 R_i中与 t_{i-1}连接的元组数量。

示例 6。 考虑图 *3d* 中 J 的索引图。选择 a_1 的概率为 15。在与 a_1 可连接的三个元组中，选择 b_2 的概率为 12。同理，选择 c_1 的概率为 13。因此，获得元组 $a_1 \bowtie b_2 \bowtie c_1$ 的概率为 $p(a_1 \bowtie b_2 \bowtie c_1) = \frac{1}{5} \times \frac{1}{2} \times \frac{1}{3}$。

假设我们已从连接路径 J 中获得大小为 m 的样本 S。根据霍维茨-汤普森估计量 [17]，基于样本 S 对 J 的连接大小进行估计，即 $|J|_S$，可按 $|J|_S = \sum_{t \in S} \frac{1}{p(t_k)} \cdot \frac{1}{m}$ [25] 计算。随着获得新的连接样本，我们可以实时更新这一估计。假设向 S 中添加了一个新元组 t_0，我们可以按如下方式更新连接大小估计。

$$|J|_{S \cup t_0} = \frac{\sum_{t \in S} \frac{1}{p(t_k)} + \frac{1}{p(t_0)}}{(m+1)} = \frac{\sum_{t \in S} \frac{1}{p(t_k)}}{m} + \frac{\frac{m}{p(t_0)} - \sum_{t \in S} \frac{1}{p(t_k)}}{(m+1)m}$$
$$= |J|_S + \frac{1}{m+1}\left(\frac{1}{p(t_0)} - |J|_S\right)$$

我们将在后续对随机游走重叠的讨论中重新探讨 $|J|$ 的均值和方差。因此，将以一定的置信水平返回实时近似答案，并且随着样本大小增大，准确度也会提高。在 Wander Join 的基础上，我们有两种方法来估计重叠大小。

首先，我们将 α设为一个参数，它是我们希望达到的置信水平。存在一个与 α对应的置信水平值 z_α。置信区间的半宽度为 $z_\alpha \cdot \frac{\sigma}{\sqrt{n}}$，其中 n是样本大小，而 σ是样本集合的标准差。当半宽度小于我们定义的阈值时，终止抽样。

6.2 连接重叠

我们描述了一种基于随机游走的算法，用于对连接进行抽样并估计连接大小。给定一组连接路径 $\Delta \in S$，我们希望估计 Δ中连接的重叠，即 O_Δ。令$S_j = \{t_1, t_2, \ldots, t_m\}$表示从连接$J_j \in \Delta$中抽取的元组集合。令 $count(t)$表示一个元组 t在集合中的出现次数。我们定义 $S'j$，使得对于 S_j中的每个元组 t，$S'j$恰好包含$p^{-1}(t)$个这样的元组 t，即 $S'j = \{t \in S_j \mid count(t) = p^{-1}(t)\}$。因此，样本 $S'j$保留了 J_j的分布。我们假设连接路径之间的重叠区域和非重叠区域均具有均匀性，也就是说，我们通过执行随机游走来抽取元组并估计连接大小；对于任意 $J_j \in \Delta$，都有 $\frac{|O_\Delta|}{|J_j|} = \frac{|\bigcap_{J_j \in \Delta} S'_j|}{|S'_j|}$。因此，可以即时估计连接重叠大小，如下所示。

$$|O_\Delta| = |\bigcap_{J_i \in \Delta} J_j| = |J_j| \cdot \frac{|\bigcap_{J_i \in \Delta} S'_i|}{|S'_j|} \tag{2}$$

如何得到 $|\bigcap_{J_i \in \Delta} S'_i|$? 我们固定一个 $J_j \in \Delta$，并持续从这个单一来源抽样，形成 S_j。在每一轮中，如果我们接受样本 t，就检查每个$J_i \in \Delta$，其中 $i \ne j$，以确定 t是否包含在 J_i中。由于我们已经为每个 J_i建立了索引（存储在哈希表中），这项操作的成本可能很低，因为它只需要通过键进行 $(N-1) \times (M-1)$ 次查询，其中 $N = |\Delta|$，而 M是连接路径中的表数。如果 t位于每个 J_i中，我们就将其纳入 $\bigcap_{J_i \in \Delta} S'i$。现在，我们可以将此估计代入定理 3，计算 Δ中的连接并集大小。接下来，我们计算 $|O_\Delta|$的置信区间。$|\bigcap_{J_i \in \Delta} S'_i|/|S'_j|$的方差记为 σ_j^2，可通过二项抽样计算，其方差为 $\hat{p}_j(1-\hat{p}_j)$，均值为 $\hat{p}_j$ [24]。Li 等人证明，$|J_j|$的均值和方差（分别记为 ϕ_j^2）为 $T_n^j(u) = \frac{1}{n-1}\sum_{i=1}^n f^j(i))$ 和$T_n^j,2(u) = \frac{1}{n-1}\sum_{i=1}^n (f^j(i) - T_n^j(f))^2$ [25]。假设

这些项相互独立，则 $|O_\Delta|$ 的方差如下。

$$\sigma^2_{|O_\Delta|} = T^j_{n,2}(u)\cdot\hat{p}_j\cdot(1-\hat{p}_j) + T^j_{n,2}(u)\cdot\hat{p}_j + T^j_n(u)\cdot\hat{p}_j\cdot(1-\hat{p}_j)$$

This gives us the following confidence interval for $|O_\Delta|$ of Eq. 2.

$$E = z\cdot\sqrt{\frac{1}{n}\sum_{J_j\in\Delta}(T^j_{n,2}(u)\cdot\hat{p}_j\cdot(1-\hat{p}_j)+ T^j_{n,2}(u)\cdot\hat{p}_j + (T^j_n(u)\cdot\hat{p}_j\cdot(1-\hat{p}_j))} \tag{3}$$

这意味着，若要使重叠估计达到90%置信度，该算法平均需要大小为 $(1.96\cdot_E{}^{z}\cdot\sigma_{|O_\Delta|})_2$的样本。 ——

请注意，我们使用随机游走进行重叠估计的估计量是无偏的。我们首先通过向集合 S_j中添加$p^{1}(t)$个元组 t来保证均匀性。我们知道有如下结果。

$$\lim_{|S_j|\to\infty}\frac{|\bigcap_{J_j\in\Delta}S'_j|}{|S'_j|} = \frac{\lim_{|S_j|\to|\Delta|}|\bigcap_{J_j\in\Delta}S'_j|}{\lim_{|S_j|\to|\Delta|}|S'_j|} = \frac{|\bigcap_{J_j\in\Delta}J_j|}{|J_j|}$$

因此，我们可以证明，当 $|S_j|$变大时，我们的结果会越来越准确，并在 $|S_j| = |\Delta|$时达到精确结果。随着重叠估计的准确度越来越接近真实值，我们也能更好地估计并集大小，这表明对于算法中使用的值，我们的估计量得到了改进。

7 在线并集采样

基于直方图的方法几乎没有预处理成本，但采样效率较低；而随机游走方法在预热阶段需要一定的采样成本，却能带来更好的估计结果和效率。为了设计一种预处理成本最低且采样效率高的采样算法，我们提出了在线并集采样算法，如算法 2所示。从整体上看，在对连接的并集进行采样的同时，以在线方式执行连接和并集大小估计。算法 2在算法1的基础上进行了两项优化：样本复用，以及结合参数更新的回溯。该算法首先使用基于直方图的方法初始化连接参数和并集参数，随后使用随机游走方法选择连接并对连接进行采样。在每次迭代中，获得的样本都会用于根据第 § 6.1 节提出的连接和并集估计方法进一步改进估计。

样本复用 （算法 2 的第 8—10 行）这弥补了随机游走的开销。回顾一下，随机游走抽取的元组并不均匀；不过，通过额外的接受/拒绝步骤，我们可以在主采样阶段复用这些元组。对于每个连接，我们都会记录每个元组 t及其概率 $p(t)$，该概率是在连接采样过程中按照第 § 6.1 节所述方法计算的。假设我们已经从 J_j中抽取了 $S = \{t_1, t_2, \ldots\ldots, t_l\}$ $t_i \in J_j$。回顾一下， S可能包含重复项，即存在 i、 j使得 $t_i = t_j$。那么，如果我们选择 J_{j_1} 可以先从 t_1、 t_2、……、 t_l中随机选择一个元组 t，但仅以 $p\ t\ ()\cdot|J_j|$的概率接受它。通过这种方式，算法保证复用的 t 以概率$p(t)\cdot\frac{1}{l}\cdot p\ t\ ()\cdot|J_j| = \frac{1}{|J_j|}$从 J_j 中抽取，从而确保并集上的抽样具有均匀性。注意，如果我们接受了 t，就不会将 t放回池中，即这是一个无放回抽样过程，并且 l会发生变化。当存储的所有元组都被用完后，下次选中 J_j时，我们只需使用第 § 3.2 节中的技术对连接进行抽样。

注意，接受率，即 R，可以等于或大于 1。这意味着算法在某一轮中可能返回多个 t实例，同时仍然确保均匀化条件。我们将 r_i定义为在某一轮中接受 i个 t实例的概率。也就是说， $\sum_i^n r_i\cdot i = R$，其中 $\sum_i^n r_i = 1$、 $0 \le r_i \le 1$。然后，我们必须通过从该系统的多个有效解中选择一个，来确定实例数量 $n \in N^+$。

算法 2 带重用和回溯的集合并采样

输入： 连接路径 $\{J_j$、 $1 \le j \le m\}$，元组数量 N，回溯参数 ϕ，目标置信水平 γ
输出： 元组 $\{t_i$、 $1 \le i \le N\}$

```
1: {|J_j|、 1 ≤ j ≤ m} |U|, ← warmup(S), {|J'_j|} ← cover(S)
2:  conf_level ← 0、 T ← {} ▷ 结果样本
3:  P ← [j][] ▷ 各连接路径中选定元组的记录概率
4:  orig_join_i ←{}                    ▷ 已见元组的原始连接记录
5: </style>当 n < N 时
6:      以概率 |J_j|/|U|, 选择 J_j      ——
7:      如果 S_j ≠ ∅, 则
8:          样本 t ∈ S_j, 接受条件为 ——p t^I()·|J_j|, 从 S_j中移除 t
9:      如果 S_j = ∅ 或 t 从 S_j 中被拒绝 那么
10:         t ← 来自 J_j的随机样本
11:     如果 t ∈ orig_join_i 对于任意 i < j 那么 拒绝 t
12:     else
13:         如果 t ∈ orig_join_i 对于任意 i > j 那么      ▷ 修订
14:             从 orig_join_i 移除 t，并将 t 添加到 orig_join_j
15:             从 T移除所有 t，并删除 P[i][t]
16:         如果 t ∉ orig_join_i，则添加 t 到 orig_join_j
17:         T ← T ∪ {t} 并更新 P[j][t]
18:     如果 ∑_{j∈[m]} |P[j]|% φ == 0 和 conf_level < γ 则
19:         {|J_j|, 1 ≤ j ≤ m} |U|, T
20:         conf_level ← Update(T,  P)          ▷ 回溯（第 7 节）
21: 返回 T
```

使用参数更新进行回溯 在第 4 节中，我们表明，尽管基于直方图的方法开销较小且实用，但它可能无法对我们的采样参数给出无偏估计量。此外，在第 6 节中，我们证明随机游走是一种无偏估计量，其参数估计在样本数量趋于无穷时会收敛到真实值。算法 2 使用基于直方图的方法的估计值初始化框架，并通过应用随机游走来细化参数。需要注意的是，采用这种细化策略时，尽管每一轮中被抽取元组的概率均为 $1/|U|$，但无法保证各轮抽取的元组具有均匀性，因为随着随机游走次数增加， $|U|$的估计值会在不同轮次之间发生变化。为缓解这种不均匀性，我们引入一种回溯技巧，对之前轮次中已抽取的所有元组采用接受/拒绝策略。

算法 2 将 T 初始化为结果样本集合，并初始化一个列表 P，用于存储从连接中获得的元组 $p(t)$的概率，无论其被接受、拒绝，还是随机游走失败时的情况。我们还指定参数 ϕ，表示回溯的频率。在抽样过程中，我们记录所有$p(t)$，无论 t 是被拒绝或重复使用的元组，还是

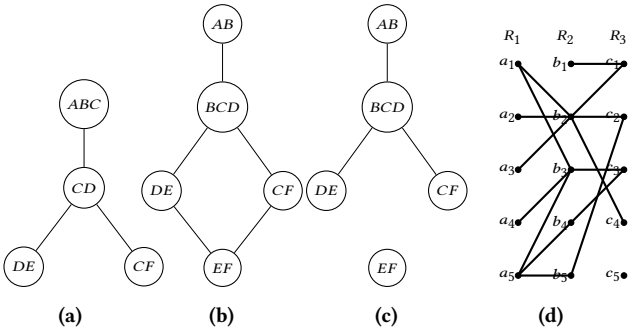


图3：(a) 无环连接，(b) 循环连接，(c) 用于估计 (b) 中重叠的树结构，(d) 用于随机抽样的骨架连接和残余连接 (d') J 的连接数据图

随机游走失败的结果（在这种情况下，$p(t) = 0$）。每经过 ϕ 次迭代，即记录 ϕ 个 $p(t)$后，我们按照随机游走方法更新连接、重叠和并集的估计值，然后按照算法 **??** 进行回溯，根据 $|U|$的新估计值调整先前抽样元组的概率。

在回溯过程中，我们遍历结果中所有先前抽样的元组，并通过以 $\frac{|J_{(t.val)}|'/|U|'}{|J_{t.val}|/|U|}$ 的概率拒绝元组 t，来调整它们的概率。其中，$|J(t.val)|$ 和 $|U|$ 是原始值，而 $|J(t.val)|'$ 和 $|U|$ 是更新后的值。不难看出，回溯算法保证结果中的每个元组都以 $\frac{1}{|U|'}$ 被抽样。我们还会跟踪估计大小的置信水平 γ，并在准确度超过预定义阈值时停止回溯。

8 其他类型的连接

本节将说明如何将我们的抽样框架推广到无环连接。我们算法中的连接子程序依赖于一种现有算法。Zhao 等人提出了一种适用于所有类型连接（链式、无环和无环 [38]）的连接随机抽样方法。我们框架中讨论的两种实例化方案提出了估计连接重叠大小参数的不同方法。随机游走方法依赖于从连接中获得的样本来进行估计，并在连接采样子程序中处理无环连接和循环连接。为简洁起见，我们不再重复 Zhao 等人的算法，而是介绍如何将基于直方图的方法扩展到无环连接和循环连接。

8.1 无环连接

我们将关系组织成一棵连接树，其中每个节点对应一个关系，每条边表示一次连接。图3c展示了连接树的一个示例。将我们的采样算法扩展到无环连接的基本思路是：把并集中的所有无环连接和链式连接转换为等长链式连接这一基本情形，并使用§ 5.1中的结果估算连接重叠。我们的解决方案首先构建一个标准连接模板。模板是一种连接树结构，每个连接的结构都可以转换为这种结构。我们将标准模板形式化为一种包含二属性关系的链式连接。之所以需要模板，是因为§ 5.1中进行大小估算所必需的基于度的比较，只有在关系具有完全相同的样本结构时才能应用。

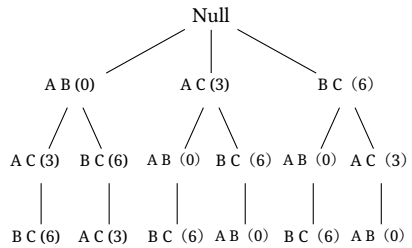
要将无环连接改写为基链连接，我们首先构造一棵等价的连接树，使得从每一层最左侧的节点开始进行广度优先遍历时，所有树都能得到具有相同模式的连接。链式连接实际上是一种只有一个分支的连接树。连接可能形成不同的树结构。因此，在将它们分解为基链连接之前，我们还需要先选定一种标准树结构（模板）。在估算过程中，选择合适的模板至关重要。糟糕的模板可能会导致我们得到最差的界 $\min_{j\in[n]} |J_j|$。

示例 7。考虑图 3a 中的连接。假设我们选择模板 $(A, D) \bowtie (A, C) \bowtie (B, C) \bowtie (B, E) \bowtie (E, F)$。要得到 (B, E)，我们需要预估 $(A, B, C) \bowtie (C, D) \bowtie (D, E)$ 的大小；要得到 (E, F)，我们需要预估 $(D, E) \bowtie (C, D) \bowtie (C, F)$ 的大小。由于我们还需要预估虚假连接大小，这两次对关系之间的预估会丢失大量信息。然而，模板 $(A, B) \bowtie (B, C) \bowtie (C, D) \bowtie (D, E) \bowtie (E, F)$ 给出了更好的界，因为我们只对关系进行一次预估，就能得到 (E, F)。

不难发现，如果我们希望保留原始关系的大部分结构，就应优先选择将属性放在其原始关系中的模板。我们将为一组链式和循环连接寻找标准模板的问题，表述为将连接拆分为二属性关系的问题，使模板树中同一关系内属性的成对距离总和最小。

8.1.1 属性成对评分。假设 S 中所有 J 都会生成包含属性 D 的表。对于任意一对属性 A，$A' \in$ D，令 $Dist_j(A, A')$ 表示在 J_j 的连接树中，A 和 A' 对应的节点（关系）之间的距离。注意，两个属性 A 和 A' 之间的距离，等于在模板中获得 (A, A') 所需执行的连接次数。于是，我们将 A 和 A' 之间的评分定义为 $score(A, A') = \sum_{j\in[n]} Dist_j(A, A')$。

再次考虑图 3a。$score(A, B) = 0 + 0 + 0 = 0$，在为标准选择表时具有最高优先级。此外，$score(A, F) = 2 + 3 + 2 = 7$ 表示 A和 F 彼此相距较远，在原始表中同时出现的可能性较小。因此，得分较低的配对更有可能原本位于同一张表中。得分越低，优先级越高。我们将所有配对构成一棵树，其中根节点为空节点；从根节点到叶节点的每条路径，在去除空根节点后，都是一条符合条件的路径。例如，如果生成的表的模式为 D $= \{A, B, C\}$，且 $(A, B) = 0$，$(A, C) = 3$，$(B, C) = 6$，那么这棵树将如图 8.1.1 所示。



我们希望标准模板的得分最低，因此可以将问题转化为寻找最小代价路径，并通过递归求解。

8.1.2 交替得分。另一个值得注意之处是，拆分关系以及未估算子连接大小的连接保留了大部分信息，因此我们可以为满足 $Dist_j(A, \ A_\prime)_=$ 0 的情况赋予权重。我们可以将这种情况的得分视为一个超参数，通过调整它来找到最紧界。

给定一个标准模板，我们现在介绍如何转换无环连接和循环连接，同时保留伪连接的信息。考虑树结构的无环连接。假设节点 R_i 有 k 个子节点，R_{i_1}，R_{i_2}，……，R_{i_k}，并且考虑模板的一种极端情况：每个表 R_{i_j} 都有一个属性，该属性与 R_i 中的某个属性配对。在这种情况下，我们对每个 $R_{i,\ j} = R_i \bowtie, R_{i_j} \bowtie, Childs(R_{i_j})$ 执行伪连接，并使用 § ??中的方法估算 $|R_{i,\ j}|$。在这一步中，我们还记录 R_{ij} 中每个属性 A 的估计最大度，如下所示：

$$M_A(R'_{i_j}) = \begin{cases} M_A(R_i) \cdot M_A(R_{i_j}) & \text{if } A \text{ is join attribute} \\ \max\{M_A(R_i), M_A(R_{i_j})\} & \text{otherwise} \end{cases}$$

通过这种方式，我们可以根据标准模板拆分 R_{ij}，并利用基数和最大度的信息。此外，我们也能够相应地估算重叠大小。请注意，转换时不一定需要对所有子节点与其父节点执行伪连接；在实际场景中，我们会根据标准模板中关系的模式选择子节点。

8.2 循环连接

本节将我们的采样算法扩展到有环查询。按照 [38],提出的方法，我们通过移除一组关系来打破连接超图中的所有环，使连接成为连通的无环连接。残余连接，即 S_R，是被移除的关系集合；骨架连接，即 S_M，是主无环连接中的关系集合。图 3c 展示了图 3b 中循环连接对应的等价骨架连接树和残余连接。设 S_R 中的属性为 $Attr(S_R)$，S_M 中的属性为 $Attr(S_M)$。我们将 S_R 视为新无环连接中的单个关系。我们甚至可以通过对 S_R 执行连接来物化 S_R。注意，残余 S_R 中的某些属性 $Attr(S_R)$ 可能会与 $Attr(S_M)$ 连接。这意味着我们有一个无环连接（骨架连接），以及一个可与骨架中的两个或更多关系连接的残余。现在，S_R 中任意属性的最大度 $M(S_R)$ 定义如下。

$$M(\mathcal{S}_R) = \max_{v_i \in A_i} |t : t \in \mathcal{S}_R, \pi_{A_i}(t) = v_i, \forall A_i \in Attr(\mathcal{S}_M) \cap Attr(\mathcal{S}_R)|$$

由于我们将残余部分视为一个关系，因此结合度数信息，可以按照 § 8.1 所述，将 S_R 拆分为基本链式连接结构，从而估算连接大小和重叠大小。请注意，选择要移除的关系集合会对性能产生显著影响。实践中，我们遵循 Zhao 等人 [38] 使用的方法来决定在哪里打断循环。

8.3 选择谓词

我们的采样算法可以通过两种方式支持选择谓词。第一种方式是在预处理期间将谓词下推到关系中，也就是说，使用谓词筛选每个关系，并在抽样过程中处理筛选后的关系。这种范式适用于基于直方图的方法和随机游走。另一种方式是在抽样过程中强制执行选择谓词。这种范式仅适用于随机游走，除非基于直方图的方法能够获知该谓词的选择度并调整度统计。由于这种范式会引入额外的拒绝因子，因此最适合选择性不太强的选择谓词。

9 评估

数据集： 我们使用三个数据集，其中包含根据 TPC-H 基准测试定制的不同类型的连接。每个查询工作负载都是从相应数据集的连接并集中抽样得到的。UQ1 包含五个链式连接，每个连接都包含五个关系：国家、供应商、客户、订单和行项目；UQ2 包含三个链式连接，涉及以下关系：地区、国家、供应商、零件供应和零件，此外，我们还按照 $Q_2^N \cup Q_2^P \cup Q_2^S$ 在 [8] 中所述添加了选择谓词；UQ3 则包含一个无环连接和两个链式连接。UQ3 由以下关系构成：供应商、客户和订单。我们对这些关系进行垂直和水平拆分，以得到具有不同模式的关系。因此，处理 UQ3 时需要应用拆分方法。

为了实验数据规模的影响，我们使用 TPCH-DBGen 生成不同规模的关系。例如，对于 TPC-H 比例因子 N-gb 和 K% 的比例比率，UQ3 是一个大小为 $K\% \cdot N \cdot 3$ 的数据集。对于 UQ2，三个连接使用相同的数据，但选择谓词的约束各不相同。因此，UQ2 的重叠尺度较大。我们还改变 UQ1 中连接之间的重叠尺度 P%。生成不同查询时，我们保持原始对应关系中 P% 的数据相同。这样一来，尽管由于关系之间的信息未知，我们无法确保查询中的重叠比率恰好为 P%，但可以保证查询之间的重叠比率与重叠尺度成正比。请注意，我们没有对循环连接查询进行实验，特别是因为将循环连接转换为无环连接以及从循环连接进行在线采样，都是基于一项已有工作 [38] 完成的。

算法 我们分别将第 4 节和第 6 节中的技术代入定理 3，以评估基于直方图和随机游走的实例化方案。基于直方图的连接估计可通过基线方法 EW（精确权重） [38], 或 EO（扩展Olken算法） [38], 实例化：前者通过计算连接数据图中每个元组的精确权重，得到权重的真实值；后者则是我们在第 3.2 节中介绍的方法。在线技术的连接估计采用了第 6 节中的随机游走。我们还将 FullJoinUnion 视为连接大小和并集大小估计的真实值。该算法执行全连接并计算并集。请注意，FullJoinUnion 在大型数据集上的开销极高。对于数据大小超过 5GB（每个关系）的情况，我们的实验超时。我们不评估 DisjoinUnion，因为它与对单条连接路径进行抽样一致，不会产生额外延迟。我们不评估伯努利集合并集采样，因为它与非伯努利方法略有不同，且理论上效率较低。

实现： Theframework 使用 Python 实现。连接中的关系存储在哈希关系中，并采用线性搜索。无环连接采用树结构实现，而无环连接

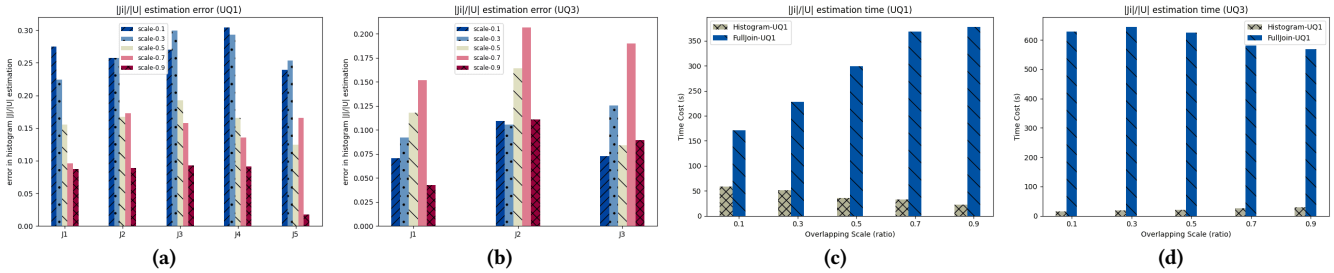


图4：使用基于直方图的 +EO 估计(a) UQ1和(b) UQ3上的连接与并集大小比率时的误差；使用基于直方图的 方法和FullJoin估计(c) UQ1和(d) UQ3上的并集大小时的运行时间。

通过递归处理。所有实验均在一台配备 2 个 Intel® Xeon Gold 5218 @ 2.30GHz（64 个核心）、512 GB DDR4 内存、Samsung® SSD 983 DCT M.2（2 TB）以及 4 个 GPU——TU102（GeForce RTX 2080 Ti）的机器上进行。

9.1 连接与并集大小近似

9.1.1 误差。 我们评估查询中每个连接的比率 $|J_i|/|U|$的估计误差，因为我们的算法依赖该比率来定义连接上的概率分布。在这些实验中，我们使用原始数据规模为3GB的UQ1和UQ3。预处理后，UQ1为9GB，UQ3为5.4GB。重叠尺度设为0.2。图4a和图4b展示了使用基于直方图的方法时，UQ1和UQ3的比率估计误差随重叠尺度的变化。

在较大的重叠尺度下，误差往往较小且稳定；在较小的尺度下，性能则不稳定。 这是因为重叠尺度较小时，少量样本就会对估计性能产生很大影响。然而，当重叠尺度较大时——这正是我们的应用场景——随机性会被消除。此外，我们观察到，图4b中UQ3的平均误差优于图4a中的UQ1。由于我们对每个连接都采用上界，我们的基于直方图的方法在长度较短的连接上具有更高的准确度。鉴于UQ3在长度和数量上都更小，这解释了为什么对UQ3的估计相对更准确。

9.1.2 运行时间。 我们在图 4c 和 4d 中报告了参数估计方法的运行时间。首先，基于直方图的方法明显快于暴力全连接。其次，对于 UQ1，我们观察到，随着重叠尺度增大，全连接的开销随之增加，而基于直方图的方法所需时间却会减少。这是因为，当重叠尺度较大且重叠结构复杂时，全连接更难扫描数据；但对于我们的方法，较大的重叠尺度反而有助于更快地找到最大度元组。

与基于直方图的技术不同，我们的随机游走技术会在预热阶段收集采样统计量。在评估重叠大小的置信水平时，我们实际上评估的是重叠部分在连接中所占的比例，即 $\frac{|\bigcap_{J_i \in \Delta} S'_i|}{|S'_j|}$。在公~~式 2 中~~，我们将 $|J_j|$ 视为精确值，以满足独立性假设。实际上，这等同于将 $|J_j|$ 的置信水平设为 1、置信区间设为 0，这近似于 Wande Join [25] 给出的情况。当置信水平达到 90% 或获得 1,000 个样本时，我们便终止在线抽样。

图 5a 比较了基于直方图的方法与采用 EO [33, 38], 作为连接大小实例化方法的随机游走，在 UQ1 上估计连接与并集大小比率时的误差。我们对每个查询使用了 3GB 的数据规模。首先，随机游走表现优于基于直方图的方法；事实上，随机游走准确度极高且稳定，对所有连接的误差都接近于零。 这是因为索引的特性会为我们提供有关重叠的额外信息。需要指出的是，这种准确的估计是以预热阶段进行抽样为代价的。我们很快将讨论复用这些样本的采样技术的实证评估。此外，尽管各连接上的估计误差相当稳定，但重叠越大，基于直方图的方法就越准确。由于重叠大小估计的准确度很大程度上取决于我们收集的样本中的重叠大小，实际重叠越大，我们就越容易在样本中找到重叠。由于我们在每一步都取最小值作为重叠大小的上界，当重叠大小接近数据规模时，上界会收紧，从而使重叠大小估计更加准确。尽管随机游走的表现更好，基于直方图的方法速度相对更快，而且可用于没有索引结构的数据库。

9.2 SetUnion 采样

9.2.1 随样本数量扩展。 对于基于直方图的方法，在从单个连接中抽样时，我们使用 EW 和 EO 两种权重初始化方法，而在随机游走中只使用 EW。首先，图 5c、5d 和 5e 展示了 SetUnion 如何随样本数量扩展。总体而言，我们可以看到，使用 EW 初始化时，基于直方图的方法和随机游走的性能几乎没有差异。换言之，估计界限的准确度对采样效率几乎没有影响。 然而，对于基于直方图的方法，在所有三个查询上，使用 EW 的速度都远慢于使用 EO，因为采用精确权重计算后，拒绝率为零。

9.2.2 运行时间分解。 图 5f、5g 和 5h 展示了在参数估计（连接大小、重叠和权重）、生成接受的结果以及生成拒绝的结果上所花时间的比较。性能下降的原因来自 EO，以及我们需要拒绝从与其指定连接不同的连接中抽取的重复元组。

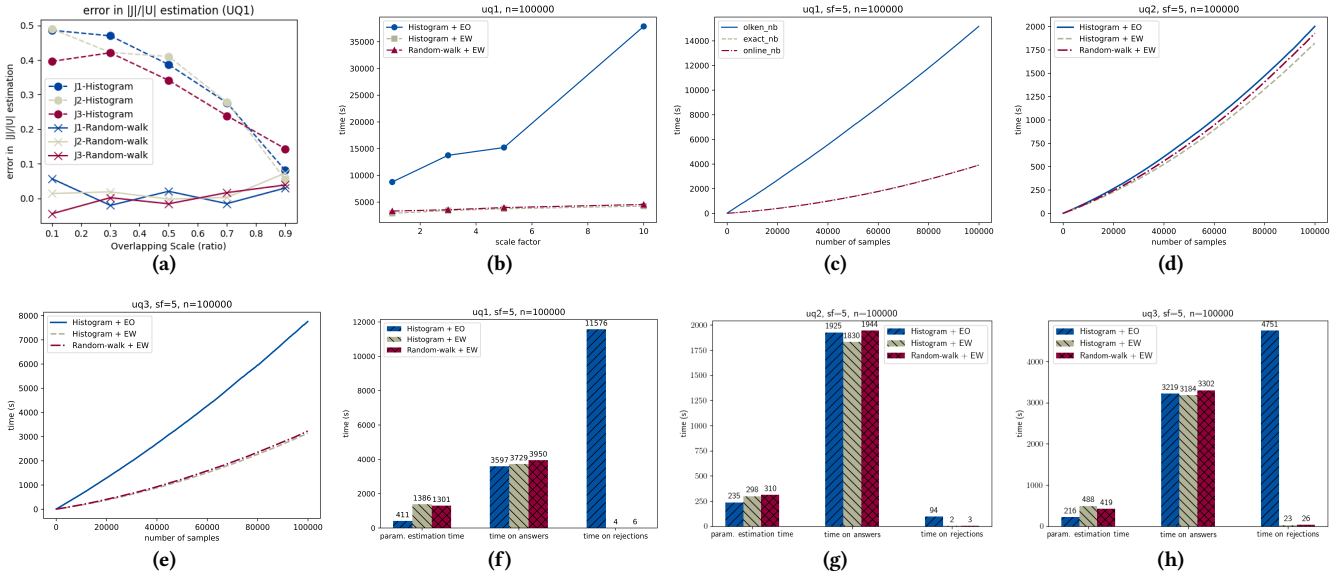


图5：（a）连接与并集大小比率的误差；（b）在SetUnion 中，UQ1上的运行时间随数据规模的变化；（c）UQ1和（d）UQ2上的运行时间随样本大小的变化；（e）UQ3；（f）UQ1 、（g）UQ2 和（h）UQ3的时间分解。

从这些图中，最显著的发现是，尽管使用 EO 的效率远低于使用 EW，但它在预热阶段的表现更好。此外，由于它使用权重上界从连接中抽样，因此多了一个拒绝阶段，而且在拒绝的结果上花费的时间远多于使用 EW 时。除此之外，对于所有查询，三种实例化方案组合在接受的结果上花费的时间相近。此外，我们的 SetUnion算法花费很少时间来拒绝重复元组，并且在使用 EW 进行连接采样时效率很高。

9.2.3 随关系大小扩展。 尽管我们对这三个查询都使用 $scale factor = 5$，但由于关系数量和重叠程度不同，执行完整连接后得到的并集大小也会不同。根据我们的设置，三个查询的并集大小从大到小依次为 UQ1、UQ3、UQ2。从两组图中可以看出，采样时间与最终并集大小成正比。此外，当期望并集大小较小时（如 UQ2），EO 在采样期间与 EW 的差距相对较小，并且在预热阶段优势更明显。

此外，图5b报告了 UQ1 在不同数据规模下的采样时间。首先，使用 EO 估算连接大小会使两种算法都比使用 EW 更慢；总体而言，随着数据规模增大，EW 的扩展性更好，因为使用精确权重时，从单个连接路径进行抽样的拒绝率为 0。其次，尽管两种算法的采样时间都会随数据规模增大而增加，但数据规模对 EO 的影响远大于对 EW 的影响。随着每个关系的规模增大，由于待连接关系中的元组数量增加，元组的拒绝率也会更高。最后，在并集大小估算中使用基于直方图或随机游走的初始化，对效率影响都很小，这与我们此前得出的结论一致。

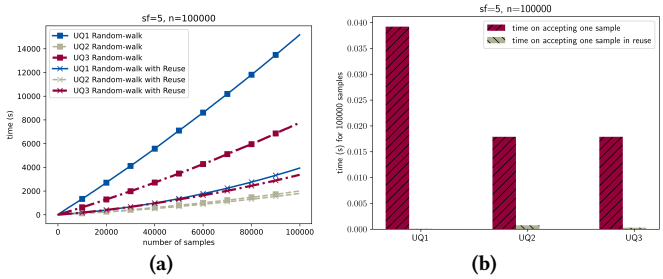


图6：（a）时间随样本大小的变化，有无复用的对比；（b）常规阶段与复用阶段的单位样本耗时。

9.3 在线并集采样与样本复用

在下一组实验中，我们评估利用预热期间收集的样本进行复用时，随机游走采样的运行时间。我们比较了三种查询中复用与不复用样本的随机游走采样。图 6a 展示了采样时间随样本大小的变化。首先可以清楚地看到，复用样本时效率高得多。在预热阶段从预采样并连接的元组池中抽样时，我们只需快速检查是否拒绝或接受，无需对每个关系进行抽样。此外，复用样本时的曲线斜率略有变化。当预采样样本全部用完后，SetUnion 的性能会逐渐回归到原有水平。另一个有趣的现象是，期望并集大小越大，复用样本带来的性能提升越明显。对于 UQ1，复用与不复用之间差距很大；但对于 UQ2，差距则小得多。图 6b 比较了常规采样阶段和复用采样阶段成功接受一个元组所花费的时间。我们使用各阶段采样所花费的总时间与成功采样的元组数量之比进行比较

，可以看到，复用预采样元组时效率高得多。这表明在线并集采样带来了巨大的效率提升。

10 相关工作

查询结果的随机访问 与我们最接近的问题是对合取查询结果进行随机访问。Bagan 等人表明，自由连通无环合取查询可以通过枚举算法求值，相邻答案之间的延迟为常数，代价是需要一个线性时间的预处理阶段 [5]。然而，由于这项工作不能保证中间答案的随机性，生成的结果可能存在极端偏差，因此不适用于学习任务。最近，Carmeli 等人研究了如何以均匀随机的顺序枚举无环合取查询的并集的答案 [8]。所提出的算法要求能够完全访问数据库，即计算完整连接，并且需要与数据库大小成线性关系的预处理时间。因此，该算法不适用于开放数据、数据市场、专有数据库或网络数据库上的随机抽样，因为这些场景的访问模型是逐元组访问。与 Carmeli 等人需要计算精确的连接大小和重叠大小的方法不同，我们的框架提出了抽样策略，并介绍了如何使用简单的统计量来近似这些参数：在直接方法中使用度，在在线方法中则使用随机样本的子集。

连接上的随机采样在 20 世纪 90 年代，人们提出了单个连接路径上的随机抽样问题 [1]。Acharya 等人提出了一种解决方案，仅使用来自基关系的随机样本即可获得较好的近似答案，但准确度仍有待提高 [1]。对连接的随机样本进行连接，得到的样本大小远小于原有样本的大小。此外，研究表明，得到的连接样本不满足独立性要求 [18]。为解决这一问题，Olken 提出了针对两表连接以特定概率拒绝两个样本连接的思路 [33]；Chaudhuri 等人提出了适用于线性连接、但不适用于任意连接的技术 [10]。这两种方法都需要表的完整信息以及索引结构。Chaudhuri 等人又提出了另一种策略——分组采样算法，该算法仅依赖部分统计信息，从而显著提高了效率 [10]。然而，以上三种方法都只适用于两表连接。Ripple join 返回的是相关且均匀的样本 [16]。Wander join [25] 将 Ripple join 扩展为从连接中返回独立但非均匀的样本。最近，Zhao 等人提出了一种处理一般多路连接并保证独立同分布的框架 [38]。该算法可以嵌入我们的框架，用于对单个连接路径进行随机抽样。

集合与查询的并集集合并集问题已在近似计数文献中得到研究 [21]。目标是设计一种随机化算法，能够高效地输出集合并集大小的近似值。Karp 等人提出了一种 $(1+\epsilon)$-随机化近似算法，可在线性运行时间内近似计算集合并集的大小。该算法需要知道每个集合的精确大小，并从每个集合中获取一个均匀随机样本。 [21]。随后，Bringmann 和 Friedrich 将该算法应用于使用均匀随机抽样为高维几何对象设计算法。他们还证明，对于高维长方体，该问题是 #P难的 [7]。集合并集的计算也与零阶频率矩估计有关 [2]。这一领域的一条研究路线关注 DNF计数问题 [20], , 包括设计基于哈希的算法 [9, 14, 21, 28]。另一条热门研究路线是估计以流式方式到达的各集合的并集 [6, 19, 27, 29]。

11 结论

本文研究了两个新问题：连接并集采样和连接并集大小近似。本文提出了一种通用的并集抽样框架，在（1）数据库管理系统中提供数据统计信息，且（2）能够访问关系中的数据时，对连接重叠和并集参数进行估计。该框架还可扩展到任意多路无环和循环连接的并集大小。值得探讨的未来研究方向包括分析数据倾斜对近似结果的影响，以及将并集采样算子集成到数据库引擎中。

参考文献

[1] Swarup Acharya、Phillip B. Gibbons、Viswanath Poosala 和 Sridhar Ramaswamy。1999年。用于近似查询回答的连接摘要。《SIGMOD Rec.》28卷2期（1999年6月），–。https://doi.org/ 。/ 。Noga Alon、Yossi Matias 和 Mario Szegedy。 。近似频率矩的空间复杂度。载于 STOC，Gary L. Miller（编）。ACM，20–29。[3] Abolfazl Asudeh 和 Fatemeh Nargesian。2022年。迈向数据市场中的分布感知查询应答。《PVLDB》15卷11期（2022年），3137–3144。[4] Albert Atserias、Martin Grohe 和 Dániel Marx。2008年。关系连接的大小界限与查询计划。载于《2008年第49届IEEE计算机科学基础年会论文集》（FOCS '08）。IEEE计算机学会，美国， 739 748 10 1109 2008 43 –。https://doi.org/ 。/ [5] FOCS. 。Guillaume Bagan、Arnaud Durand 和 Etienne Grandjean。2007年。关于无环合取查询与常数延迟枚举。载于 CSL，Jacques Duparc 和 Thomas A. Henzinger（编），第4646卷。208–222。[6] Ziv Bar-Yossef、T. S. Jayram、Ravi Kumar、D. Sivakumar 和 Luca Trevisan。2002年。对数据流中的不同元素计数。载于 RANDOM（《计算机科学讲义》），José D. P. Rolim 和 Salil P. Vadhan（编），第2483卷。Springer，1–10。[7] Karl Bringmann 和 Tobias Friedrich。2008年。近似计算高维几何对象并集与交集的体积。载于 ISAAC（《计算机科学讲义》），Seok-Hee Hong、Hiroshi Nagamochi 和 Takuro Fukunaga（编），第5369卷。Springer，436–447。[8] Nofar Carmeli、Shai Zeevi、Christoph Berkholz、Benny Kimelfeld 和 Nicole Schweikardt。2020年。使用随机访问和随机顺序枚举回答（并集）合取查询。载于 PODS。393–409。[9] Supratik Chakraborty、Kuldeep S. Meel 和 Moshe Y. Vardi。2016年。概率推理中近似计数的算法改进：从线性 SAT 调用到对数 SAT 调用。载于 IJCAI，Subbarao Kambhampati（编）。3569–3576。[10] Surajit Chaudhuri、Rajeev Motwani 和 Vivek Narasayya。1999年。关于连接上的随机抽样。《SIGMOD Rec.》28卷2期（1999年6月），263–274。https://doi.org/10. 1145/304181.304206[11] Lingjiao Chen、Arun Kumar、Jeffrey F. Naughton 和 Jignesh M. Patel。2017年。迈向规范化数据上的线性代数。《PVLDB》10卷11期（2017年），1214–1225。[12] Nadiia Chepurko、Ryan Marcus、Emanuel Zgraggen、Raul Castro Fernandez、Tim Kraska 和 David R. Karger。2020年。ARDA：面向机器学习的自动关系数据增强。《PVLDB》13卷9期（2020年），1373–1387。[13] E. F. Codd。1971年。数据库关系模型的进一步规范化。研究报告 / RJ / IBM / 加利福尼亚州圣何塞 RJ909（1971年）。[14] Paul Dagum、Richard M. Karp、Michael Luby 和 Sheldon M. Ross。2000年。一种用于蒙特卡洛估计的最优算法。《SIAM》29卷5期（2000年），1484–1496。[15] Arnaud Durand 和 Yann Strozecki。2011年。含二阶变量的逻辑查询问题的枚举复杂度。载于 CSL（LIPIcs），Marc Bezem（编），第12卷。189–202。[16] Peter J. Haas 和 Joseph M. Hellerstein。1999年。用于在线聚合的涟漪连接。载于《1999年ACM SIGMOD数据管理国际会议论文集》（美国宾夕法尼亚州费城）（SIGMOD '99）。计算机协会，美国纽约州纽约市，287–298。https: //doi.org/ 10 1145 304182 304208 。/ 。[17] D.G. Horvitz 和 DJ Thompson。1952年。有限总体无放回抽样的推广。《J. Amer. Statist. Assoc.》47卷260期（1952年），663–685。[18] Dawei Huang、Dong Young Yoon、Seth Pettie 和 Barzan Mozafari。2019年。样本连接：面向实践者的理论指南。《PVLDB》13卷4期（2019年），547–560。[19] Daniel M. Kane、Jelani Nelson 和 David P. Woodruff。2010年。一种用于不同元素问题的最优算法。载于 PODS，Jan Paredaens 和 Dirk Van Gucht（编）。ACM，41–52。[20] Richard M. Karp 和 Michael Luby。1983年。用于枚举和可靠性问题的蒙特卡洛算法。载于 FOCS。56–64。[21] Richard M. Karp、Michael Luby 和 Neal Madras。1989年。用于枚举问题的蒙特卡洛近似算法。《J. Algorithms》10卷3期（1989年），429–448。[22] Arun Kumar、Jeffrey F. Naughton 和 Jignesh M. Patel。2015年。在规范化数据上学习广义线性模型。载于 SIGMOD，Timos K. Sellis、Susan B. Davidson 和 Zachary G. Ives（编）。ACM，1969–1984。[23] Arun Kumar、Jeffrey F. Naughton、Jignesh M. Patel 和 Xiaojin Zhu。2016年。是否连接？特征选择前应三思连接问题。载于 SIGMOD，Fatma Özcan、Georgia Koutrika 和 Sam Madden（编）。ACM，19–34。[24] Yong-Gu Lee 和 Sam-Yong Kim。2008年。统计学导论。Yulgokbooks，韩国（2008年），342–351。[25] Feifei Li、Bin Wu、Ke Yi 和 Zhuoyue Zhao。2016年。Wander Join：通过随机游走进行在线聚合。载于《2016年数据管理国际会议论文集》（美国加利福尼亚州旧金山）（SIGMOD '16）。计算机协会，美国纽约州纽约市，615–629。https: //doi.org/ 10 1145 2882903 2915235 。/ 。Kaiyu Li 和 Guoliang Li。 。近似查询处理：有哪些新进展，又该何去何从？[26] 2018 ——近似查询处理综述。《Data Sci. Eng.》3卷4期（2018年），379–397。

[27] Kuldeep S. Meel、Sourav Chakraborty 和 N. V. Vinodchandran。2022年。Delphic 集合并集大小的估计：实现对流大小的独立性。载于*PODS*。ACM，41–52。[28] Kuldeep S. Meel、Aditya A. Shrotri 和 Moshe Y. Vardi。2017年。基于哈希的方法用于近似 DNF 计数。载于*FSTTCS (LIPIcs)*，Satya V. Lokam 和 R. Ramanujam（编），第 93 卷。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，41:1–41:14。[29] Kuldeep S. Meel、N. V. Vinodchandran 和 Sourav Chakraborty。2021年。流式模型中集合并集大小的估计。载于*PODS*。ACM，126–137。[30] Rajeev Motwani 和 Prabhakar Raghavan。1995年。随机化算法。剑桥大学出版社。[31] Fatemeh Nargesian、Erkang Zhu、Ken Q. Pu 和 Renée J. Miller。2018 年。开放数据上的表联合搜索。*VLDB* 基金会会刊 11，7（2018年），813–825。[32] Hung Q Ngo、Ely Porat、Christopher Ré 和 Atri Rudra。2018年。最坏情况最优连接算法。美国计算机协会会刊（*JACM*）65，3（2018年），1–40。[33] Frank Olken。1993年。数据库随机抽样。博士论文。加利福尼亚大学伯克利分校。[34] Frank Olken 和 Doron Rotem。1995年。数据库随机抽样：综述。统计与计算5，1（1995年），25–42。[35] Maximilian Schleich、Dan Olteanu 和 Radu Ciucanu。2016年。因子化连接上的线性回归模型学习。载于*SIGMOD*，Fatma Özcan、Georgia Koutrika 和 Sam Madden（编）。ACM，3–18。[36] V. N. Vapnik 和 A. Y. Chervonenkis。1971年。事件相对频率向其概率的一致收敛。概率论及其应用XVI，2（1971年），264–280。[37] Swarup Acharya、Phillip B Gibbons、Viswanath 和 Poosala Sridhar Ramaswamy。1998年。用于近似查询回答的连接摘要。（1998年）。[38] Zhuoyue Zhao、Robert Christensen、Feifei Li、Xiao Hu 和 Ke Yi。2018年。重新审视连接上的随机采样。载于*2018*年国际数据管理会议论文集（美国德克萨斯州休斯敦）*(SIGMOD '18)*。计算机协会，美国纽约州纽约市，1525–1539。https://doi.org/10.1145/ 3183713.3183739