重新审视连接点上的随机抽样

赵卓悦[1]、罗伯特-克里斯滕森[1]、李飞飞[1]、肖虎[2]、柯毅[2]
1 犹他大学　　2 香港科技大学
{zyzhao,robertc,lifeifei}@cs.utah.edu　　{xhuam,yike}@cse.ust.hk

摘要

连接的成本很高，尤其是在大型数据和/或多重关系中。降低其高昂成本的一个可行方法是只返回全部连接结果中的一个简单随机样本，这对许多任务来说已经足够。事实上，早在 1999 年，Chaudhuri 等人就提出了连接抽样问题，认为这是大型数据库系统面临的一个基本挑战。他们还指出了这一问题的一个基本障碍，即采样算子*不能*通过连接进行采样，即 sample($R \bowtie S$) ≠ sample(R) $\bowtie$ sample(S)。为了克服这一障碍，他们使用了预先计算的统计量来指导采样过程，但只展示了如何对双关联连接进行采样。

本文针对*非循环和循环多向连接*重新探讨了这一经典问题。我们以 Chaudhuri 等人的想法为基础，但在几个非核心方向上进行了扩展。首先，我们提出了对多向连接进行随机抽样的总体框架，其中包括作为特例的 Chaudhuri 等人的算法。其次，我们探索了几种实例化该框架的方法，这取决于底层数据的先验信息，并在样本生成延迟和吞吐量之间做出了不同的权衡。我们分析了不同实例的特性，并与基准方法进行了对比评估；结果清楚地证明了我们的新技术的优越性。

综合传播战略概念

• **信息系统** → **查询优化**；**连接算法**；*在线分析处理引擎*； - **计算理论** → **草图和采样**；

关键词

随机抽样；多向连接；连接抽样框架

ACM 参考格式：

赵卓悦、罗伯特-克里斯滕森、李飞飞、胡晓和易可 .2018.重新审视连接的随机抽样。In *SIGMOD'18: 2018 International Conference on Management of Data, June 10-15, 2018, Houston, TX, USA*.ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3183713.3183739

1　引言

连接是关系数据库系统中的一个基本操作。许多查询和分析工作负载都依赖联接操作来连接两个或多个关系中的记录。或半结构化数据，例如

许多关于非结构化

SIGMOD'18，2018 年 6 月 10-15 日，美国德克萨斯州休斯顿

https://doi.org/10.1145/3183713.3183739

在图数据库中，连接也可以表述为关系连接。另一方面，连接的成本很高，尤其是在大量数据和/或多重关系（称为多向连接）的情况下。

然而，一个重要的现象是，许多应用并不需要完整的连接结果。相反，对连接结果进行随机抽样往往就足够了[2, 8]。这方面的例子包括估计总数（COUNT）、求和（SUM）、平均值（AVG）等聚合门、中位数和定量数等更复杂的分析任务、核密度估计、统计推断、聚类、回归、分类等。从本质上讲，任何依赖于*整体*数据的分析任务都可以使用样本来代替全部数据（MAX 和 MIN 等少数操作除外）。事实上，统计学的全部文献都是关于如何有效利用样本和量化特定抽样程序的准确性。

在众多抽样方法中，*简单随机抽样*是统计文献中使用和研究最广泛的方法。在规模为 k 的简单随机抽样中，*以相等的概率*抽取基本群体中的每个元素，并*独立重复*该过程 k *次*。我们还可以将抽样区分为*替换*抽样和*不替换*抽样。在后一种情况下，被抽取的元素在抽取后会从总体中剔除，因此样本必须包含 k *个*不同的元素。

当可以直接访问底层群体（例如存储在数组中）时，简单的随机抽样很容易，但当隐含地给出底层群体（例如连接的结果）时，随机抽样就变得非常棘手。事实上，在 SIGMOD'99 大会上，两篇著名论文 [2, 8] 同时提出了对连接进行随机抽样的问题，作为对业界的一个基本挑战，同时提供了一些部分解决方案。Acharya 等人[2]注意到，当连接只由遵循特殊结构的外键连接组成时，连接结果只映射到一个关系，这使得问题变得容易得多（更多细节见第 2.3 节）。

正如文献 [8] 所指出的，该问题的一个基本挑战是采样操作*不能*通过连接运算符向下推送，即 $\text{sample}(R) \bowtie \text{sample}(S) \neq \text{sample}(R \bowtie S)$。$\text{sample}(R) \bowtie \text{sample}(S)$中的每对连接图元都是从完整连接结果中统一选择的，但各对图元之间有很强的相关性。事实上，关于在线聚合的*波纹连接*[11, 13, 14, 16]的一系列工作正是研究如何使用这样一个*非独立样本*来估计简单的聚合（如 COUNT、SUM、AVG），这并非易事，而且可能代价高昂。此外，如何将非独立样本用于上述所有其他不那么简单的分析任务，目前还不清楚。有趣的是，最近的 wander join 算法[22]会从连接中返回*独立但不均匀的样本，*同样，由于*不均匀性，*这些样本只能用于估计集合。

促使我们重新审视这一经典问题的另一个重要原因是，近年来将关系运算与

机器学习并入一个统一的系统，如 Spark [5]。除了避免在不同系统间传输数据的明显优势外，统一系统的更大优势在于关系处理（尤其是连接）和机器学习任务的潜在性能提升。最近关于联接学习的研究[21, 32]朝着这个方向迈出了充满希望的第一步。他们的方法基于因式分解计算，因此只适用于特定的机器学习模型，如线性回归。另一方面，我们认为连接取样为这一问题提供了更通用的解决方案。事实上，统计机器学习中有一个成熟的理论，即训练模型所需的样本大小与模型的内在复杂性和表现力（即 VC 维度）有关。因此，当需要在一个非常大的连接结果上进行训练时，在连接结果上抽取一个样本，然后用这个样本训练模型，可以大大节省连接计算和模型训练的成本，同时在准确性上也会造成很小且有界的损失。但需要注意的是，该理论要求样本是均匀和独立的，因此既不能使用波纹连接，也不能使用徘徊连接。

为了获得均匀且独立的样本，Chaudhuri 等人 [8] 提出了两种方法。第一种归功于 Olken [25]，它以适当的概率剔除来自 sample(R) ⋈ sample(S)的样本。第二种方法是在 S 统计信息的指导下，非均匀地从 R 中抽取样本。

3 个或更多的关系作为未来的工作，但从未完成。**我们的贡献**

这项工作重新审视了这个经典而重要的问题。我们设计了一个通用的连接抽样框架，它将 Chaudhuri 等人的方法和 Olken 的方法作为特例，但以*非难*的方式扩展了他们的方法，以处理*任意多向连接、非循环或循环连接，以及带有选择预设的连接*。

在此过程中，我们利用了有关连接大小上限的新成果 [6、18、22]，从而显著改善了连接大小。在此过程中，我们利用了有关连接大小上界的新成果 [6, 18, 22]，从而显著改进了连接大小上界。

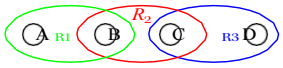
采样效率。我们的采样框架可以根据基础数据的先验信息，以不同的方式进行实例化，并在采样延迟和吞吐量之间做出不同的权衡。具体来说，本文有以下贡献：

- 我们设计了一个通用的连接采样框架（第 3 节），它可以与任何连接大小上限方法相结合。

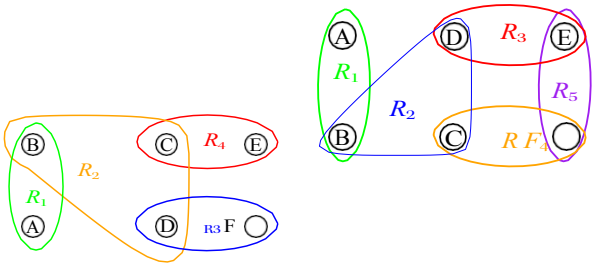
  正如我们将在后面*的定理* 3.1、5.1 和 5.2 中说明的那样，框架的任何实例*总是从完整的连接结果中返回统一且独立的样本*，但在抽样效率上可能有所不同。
- 我们的研究表明，现有的连接采样算法都有一个共同点，那就是 "连接"。

R_i	一个输入关系。
a_i , a, b, c, . .	的属性。
dom(A)	属性 A 的域。
t, t' , t_i	元组。
A	所有属性的集合。
A(R_i)	R 的属性$_i$。
H	连接查询（用超图表示）。
H (I)	实例 I 的实例超图。
J	连接结果集。
d_A (v, R)	v 在 R 中属性 A 上的频率。
M_A (R)	R 中 A 值的最大频率。
r_0	与 R_1 中所有元组连接的根元组。
W (t), W (t, R)	t 与下面所有关系的连接大小。
W_i(t, R)	t 与 R 及所有关系的连接大小
R_i, R_j, . .	R 的子关系$_i$。
H_s	低于 R_s。
H_r	骨架查询（必须是非循环查询）。残
M_r	差查询
	H_r 中能与 H_s 中任意连接结果连接的最大图元个数。

表 1：本文使用的符号。



(a) 链式连接



(b) 明星加盟。 **(c) 循环连接。**

我们的框架的特殊情况，并提出新的实例处理多向连接的框架（第 4 节）。

- 我们展示了如何扩展我们的框架，以处理一般的多向连接，非循环或循环，有选择或无选择谓词（第 5 节）。
- 我们使用 TPC-H 基准和大型社交图谱数据集 进行了广泛的实验评估，以研究

  第 5 节介绍了所提方法的性能，并将我们的方法与其他基线方法进行了比较。第 6 节的结果

图 1：以超图表示的若干连接。

- 我们将在本节中根据算法特性和经验表现，介绍对各种实例的见解。

7，以及一些有用的扩展（如条件连接、分组、更新）和我们方法的局限性。

第 2 节介绍了我们的研究背景，第 8 节回顾了

第 9 节是本文的结尾。

2 预览

2.1 问题定义：作为超级图加入

的效率和可扩展性。

我们的方法，以及与其他方法相比的优越性。

设 A 是数据库中的属性集，A(R_i) 是关系 R_i 中的属性集。多向连接查询 $R1 \triangleright\triangleleft \cdots \triangleright\triangleleft R_n$ 可以表示为一个超图 H = (A, {A(R_i), i = 1, ... n}$)$，其中每个顶点代表一个属性，每个超边代表

包含关系的所有属性。图 1 显示了各种连接的超图的一些

直观示例。

对于每个属性 $A_i \in$ A ，让 dom(A) 成为其域，从中抽取值。我们有时使用 R_i (A, B) 表示 A(R_i) = {A, B} 。为简化表述，我们假设任意两个关系最多连接一个共同属性[1]。

R_i 中的每个元组 t 为每个属性 $A \in$ A(R_i) 赋值 $a \in$ dom(A)因此，连接结果 t 是任何赋值给

[1] 如果情况并非如此，则可引入 "复合 "属性，将多个属性合二为一。

R1

A	B
1	2
2	2
3	6
4	7

R2

B	C
2	18
5	18
6	26
6	31
7	32

R3

C	D
18	101
18	102
26	103
31	104

R1 ⊳ R2 ⊳ ◁ R3

A	B	C		D
1			218	101
1			218	102
2			218	101
2			218	102
3			626	103
3			631	104

图 2：关系 R1、R2、R3 **与连接** R1 ▷ ◁ R2 ⊳ ◁ R3。

A 中的所有属性，这样对于每个 $_{Ri}$，都存在一个与 τ 一致的元组 $_{t\in Ri}$，即 $\pi_{A(Ri)}\tau = t$。图 2 显示了图 1a 中链连接在一个特定实例上的示例。

给定一个特定的数据库实例 I，我们也可以把每个不同的属性值看作一个顶点，把每个元组看作一个超边，超边包含了这个元组中的所有值。我们称这种（超
)图的*实例（超）图*，表示为 H (*I*)。因此，对于图 2 中的 3 关联链连接，连接结果 τ 对应于
是连接图中一条长度为 3 的路径，它从 dom(*A*) 中的某个值出发，到 dom(*D*) 中的某个值结束。

对于一般联接查询 H，联接结果是 H (*I*) 的一个子图，对于所有 $_{Ai}\in$ A，每个 dom($_{Ai}$) 中都恰好包含一个值。*联接抽样的问题*是以 1/| *J* | 的概率返回每个这样的子图，其中 $J = R_1 \triangleright\triangleleft \cdots \triangleright R_n$ 表示所有联接结果。
结果。只返回一个采样结果通常是不够的、
而我们希望连续生成采样结果，直到达到某个所需的样本量 k
。联合采样要求这些样本完全独立。

2.2 两两关联联合取样

我们首先回顾了现有的双相关采样算法。

连接。考虑 $R_1(A, B) \triangleright\triangleleft R_2(B, C)$，其中 B 是两个关系的共同连接点。对于任意值 v∈ dom(*B*)，设 $dB(v, Ri)$ 是 v *在* $_{Ri}$ 中属性 B 上的频率，即 $dB(v, Ri) = |t : t\in Ri, \pi B(t) = v|$。设 $_{MB}(Ri) = _{maxv} dB(v, Ri)$ 。
第一种算法由奥尔肯提出[25]。他的算法首先
从 R_1 中均匀采样一个元组 t_1，然后从 $_{R2}$ 中从所有与 $_{t1}$ 连接的元组中均匀采样一个元组 $_{t2}$，即 $\pi B(t2) = \pi B(t1) = v$。最后，它以 $dB(v, _{R2})/MB(_{R2})$ 的概率输出连接结果 $_{t1} \triangleright\triangleleft _{t2}$，否则拒绝接受。

如果 $_{MB}(R_2)$ 远大于 R_2 中的典型值频率，Olken 算法的拒绝率可能会很高。

Chaudhuri 等人[8]认为，如果频率信息 $dB(v, R_2)$'s 已知，则可以通过以 $\pi B(_{t1}) = v$ *的概率对*元组 t1∈ $_{R1}$ 进行采样，从而大大提高采样效率。
与 $_{dB}(v, _{R2})$ *有关*。在对这样一个元组 $_{t1\in R1}$ 进行采样后，我们就可以
随机选取某个 $_{t2\in R2}$ 与 $_{t1}$ 连接，并返回连接结果，而不拒绝接受。当无法获得完整的频率信息时、
他们使用了一种混合策略，即用他们的方法处理高频值（假定频率可用），用 Olken 的方法处理低频值。如果连接属性 B 上有索引，Olken 算法和 Chaudhuri 等人的算法都可以实现、需要对两种关系进行全面扫描。

2.3 多向外键连接取样

在与 Chaudhuri 等人的算法相同的 SIGMOD 会议上，Acharya 等人[2] 实际上研究了多向连接的采样问题。不过，他们的算法只适用于*非循环的*

R_n 在 [2] 中称为*源关系*。用超图术语来说，这种连接必须满足以下要求（可能在重新标记关系之后）：

(1) $\forall i < j$，且 $A(R_i) \cap A(R_j) \neq \emptyset$，它们的共同属性是 R_i 中的主键和 R_j 中的外键；以及

(2) $\forall i \leq n-1$，存在一个 $j > i$，使得 $A(R_i) \cap A(R_j) \neq \emptyset$。

这两项要求的强烈含义是 R_n 与连接结果之间是一一对应的，因此 R_n 被称为*源*关系。因此，从连接结果中采样就简化为从 R_n 中采样。

例如，图 2 所示的数据库实例没有即使对关系进行了重新标记，也无法满足这些要求。对这种连接的采样不能简化为对任何单一关系的采样，这将是本文要解决的关键难题。

3 联合抽样框架

为了解决多向连接抽样问题，我们首先给出了一个通用的连接抽样框架，该框架将 Olken 算法和 Chaudhuri 等人的算法作为特例。在第 4 节中，我们利用文献中开发的最新连接大小边界，展示了该框架的几个实例。

我们将首先考虑链式连接；第 5 节将扩展到其他类型的连接。为了便于表述，我们引入一个 R_0，它只有一个 "根 "元组 r_0，它与所有的 R_1 中的元组。此外，对于每个 R_i 和每个元组 $t \in R_i$，定义

$$w(t) = |t \triangleright \triangleleft R_{i+1} \triangleright \triangleleft R_{i+2} \triangleright \triangleleft \cdots \triangleright \triangleleft R_n|。$$

因此，$w(r_0)$ 只是全连接大小；对于任意 $t \in R_n$，$w(t) = 1$。

设 $W(t)$ 是 $w(t)$ 的上限。事实上，所有实例该框架的不同之处仅在于如何计算 $W(t)$。这种抽样框架的基本思想是，应使用权重 $w(t)$ 对元组 t 进行抽样。但 $w(t)$ 通常无法获得，因此我们使用 $W(t)$ 作为代理，但随后必须以适当的概率剔除样本。为简化符号，我们使用以下简称 $w(R) = \sum_{t \in R} w(t)$ 和 $W(R) = \sum_{t \in R} W(t)$，其中 R 可以是任何元组集合。我们在 $W(t)$'s 上维护以下不变式：

$$\text{对于任意 } t，W(t) \geq w(t)； \quad (1)$$

$$\text{对于任意 } t \in R_n，W(t) = w(t) = 1；$$

(2) 对于任意 $t \in R_i$，$W(t) \geq W(t \rtimes R_{i+1})$，$0 \leq i \leq n-1$。 (3)

请注意，半连接 $t \rtimes R_{i+1}$ 表示 Rn 中所有图元的集合。R_{i+1} 与 t 相连。事实上，不变式 (2) 和 (3) 意味着 (1)，但是我们为清晰起见，我们倾向于保留它。算法描述如下。下面我们将展示每个连接结果将由 Al-根据不变式 (1-3)，算法 1 的概率相同。因此，多次调用该算法将返回一个重新放置的随机样本。如果需要进行不替换抽样，我们只需拒绝已经抽取过的样本即可。

定理 3.1. *在链式连接中，算法 1 以概率 $1/W(r_0)$ 返回 J 中的每个连接结果，其中 $W(r_0) \geq w(r_o) = |J|$ 是算法运行前连接大小的上限值。*

证明。我们将通过归纳法证明，对于任意 i，每一个局部连接结果（r_0，$t1$，...，t_i）$\in R_0 \triangleright \triangleleft R_1 \triangleright \triangleleft \cdots \triangleright \triangleleft R_i$ 都是以 $W(t_i)/W(r_0)$ 的概率采样的。那么，取 $i = n$ 并应用不变式 (2) 即可得出该定理。

算法 1：对链式连接进行采样

输入：R1, - - - Rn ,W (t) 为 t0 和任意 t∈ Ri , i∈ [1, n] 。
输出：从 R 中采样的元组1 ⊳⊲---⊳⊲Rn，或拒绝

```
1 t ← r0;
2 S ← (r0);
3 对 i = 1, ., n do
4     W′(t) ← W(t);
5     W(t) ← W(t ⋊ Ri);
6     拒绝概率为 1 - W(t ⋊ Ri)/W′(t);
7     t ← 一个随机元组 t′∈ (t ⋊ Ri)，概率为 W(t′)/W(t ⋊ Ri);
8     将 t 加入 S;
9 结束
10 报告 S;
```

基本情况 $i = 0$ 是微不足道的。假设对 i 成立，我们将证明对 $i+1$ 也成立。考虑任意 $(r_0, t_1, \ldots, t_{i-1}, t_i) \in R_0 \bowtie R_1 \bowtie \cdots \bowtie R_i$。根据归纳假设，我们知道（$r_0$，$t_1$，...，$t_{i-1}$）的采样概率为 $W(t_{i-1})/W(r_0)$。$(t_{i-1})/W(r_0)$。
算法的第 i 步不剔除概率为 0.5%的样本。
概率 $W(t_{i-1} \bowtie R_i)/W(t_{i-1})$。以不拒绝为条件，$t_i$ 被采样的概率为 $W(t_i)/W(t_{i-1} \bowtie R_i)$。因此，（$r_0$，$t_1$，...，$t_i$）被采样的概率为

$$\frac{W(t_{i-1})}{W(r_0)} \cdot \frac{W(t_{i-1} \bowtie R_i)}{W(t_{i-1})} \cdot \frac{W(t_i)}{W(t_{i-1} \bowtie R_i)} = \frac{W(t_i)}{W(r_0)} \qquad \square$$

请注意，算法 1 的第 5 行更新了 $W(t)$ 因此，因瓦利蚂蚁 (3) 在 t 时刻收紧。这对当前调用的 算法没有影响，但对今后需要更多样本时的调用有帮助。

根据定理 3.1，采样效率（即成功返回样本的概率）为 $|J|/W(r_0)$，因此我们希望 $W(r_0)$ 尽可能接近 $|J|$。因此，算法自改进的意义在于，每次调用（包括那些拒绝部分抽样）收紧了部分 $W(t)$ 的值。最终，当不变式（3）在所有图元上都很紧密时，我们就会得到 $W(r_0) = |J|$，从而实现 100% 的抽样效率。

备注。需要注意的是，对于 t_0 和任意 $t \in R_i$（$i \in [1, n]$），算法 1 的输入不是所有 $W(t)$ 的值，而是任何能够推导出 $W(t)$ 上界的方法。

对于任何给定的元组 t，*算法 1 都会*立即初始化 $W(t)$，而且只针对采样过程中遇到的 t。无论选择哪种 $W(t)$，算法 1 也总是返回与内相关的样本。原因在于，无论采样历史如何，算法 1 返回的已接受样本总是均匀的。根据贝叶斯定理和归纳

4 框架实例

如前所述，算法 1 的不同实例只在如何设置连接大小上限（即 $W(t)$）上有所不同。在延迟和吞吐量之间需要权衡：设置更严格的上限需要更高的成本，但一旦设置到位，就能更高效地生成样本。另一方面，较松的上限界值更容易计算，但采样效率低（因为拒绝率可能更高）。Olken 算法和 Chaudhuri 等人的算法实际上是这种权衡的两个极端。Olken 算法的设置成本几乎为零，因为它要求但采样效率很低。而 Chaudhuri 等人的算法则使用了最严格的上界，采样效率达到 100%，但需要 R_2 上的全部频率信息。

在本节中，我们首先展示如何将这两种极端方法推广到 n *相关*链式连接，然后探索新的方法，在上界计算和采样效率（同样适用于 n *相关*链式连接）之间取得更好的平衡。我们将在第 5 节讨论如何扩展到其他类型的连接。

4.1 推广奥尔肯算法

在我们的框架中推广 Olken 算法是很简单的。对于 $i = 2, \ldots, n$ 的每个关系 $R_i(A_i, A_{i+1})$，让 $MA_i(R_i) = \max_v dA_i(v, R_i)$，即属性 A_i 上的最大频率。，n，设 $MA_i(R_i) = \max_v dA_i(v, R_i)$ 为属性 A_i 的最大频率。注意
这些基本统计信息通常已由数据库保存。然后，对于 $i = 1, ., n - 1$，我们设置 $W(t)$ 的值如下。

$$W(t) = \prod_{j=i+1}^{n} MA_j(R_j)$$

法，一组样本的联合概率是它们各自概率的乘积，这意味着样本是相互独立的。

当算法 1 退化为 $n = 2$ 的情况时，Olken 算法和 Chaudhuri 等人的算法都是算法 1 的特例：Olken 算法和 Chaudhuri 等人的算法都是算法 1 的特例。
该方法相当于为所有 $t1 \in R_1$ 设置 $W(t_1) = MB(R_2)$、
假设 $R_1 = AB$，$R_2 = BC$（回顾一下，对于所有 $t_2 \in R_2$，$W(t_2)$ 必须为 1）。对于任意 $t_1 \in R_1$，Chaudhuri 等人的算法设置 $W(t_1) = w(t_1) = |t1 \bowtie R2|$。因此，该算法的采样效率为 100%，但需要 R_2 中关于连接属性的全部频率信息。

对于所有 t∈ $_{Ri}$，并设置 W ($_{r0}$) = W ($_{R1}$)。基本上，我们假设 $_{Ri}$ 中的每个元组都与 $_{Ri+1}$ 中的 $_{MAi+1}$ ($_{Ri+1}$) 元组连接。不变式（1-3）很容易验证。我们稍后会提到广义
奥尔肯算法作为扩展奥尔肯算法（EO）。

请注意，在实例化算法 1 时，无需明确存储所有 W (t) 的值。在奥尔肯实例化的情况下，一个关系中的所有图元都共享相同的初始值。只有当算法 1 对某些 W (t) 进行改进时，我们才需要明确地维护它们。

4.2 推广乔杜里等人的算法

Chaudhuri 等人的算法为所有 t 设置 W (t) = w (t)。对于 2 关联连接，w (t) 只是 R_2 中的值频率。对于 n *关联*链连接，每个 w (t) 是一个子连接大小。为了计算所有的 w (t)，我们回顾一下连接的（超）图表示法
和实例图 H (I)。我们的观察结果是，对于图元 $_{t∈Ri}$ *而言*，部分连接 t ▷◁$_{Ri+1}$▷◁----▷◁$_{Rn}$ 只是从边 t 开始一直到 $_{Rn}$ 的路径集合，而 w (t) 则是
此类路径的数量。

虽然枚举所有路径（即计算全连接）的成本可能很高，但计算所有计数可以使用动态编程，线性时间为 O (|R_1 | + - - + |$_{Rn}$|)。这种动态
编程公式是基于从 $_{Rn}$ 到
R_1 使用不变式 (2) 和 (3)，只是在不变式 (3) 中将 ≥ 替换为 = 。因此，其成本与所有关系式的总大小呈线性关系。

我们稍后将把乔杜里等人的广义算法称为精确权重（EW）算法。

4.3 其他连接尺寸上限

数据库理论界对连接大小的上界重新产生了兴趣。我们的抽样框架的通用性允许使用其中的任何一种。

最著名的结果是 AGM 约束 [6]。在一般查询中，它需要求解一个线性程序，详情请读者参阅 [6] 和写得更好的调查论文 [24]。然而
在链式连接的情况下，AGM 约束很简单，我们包括
为完整起见，在此将其删除：
$|R_1 \bowtie \cdots \bowtie R_n| \le$ 最小值 $\prod_{i:\{1,n\}\subseteq I \cap \{i,i+1\}\neq\emptyset, i=2,\ldots,n-2} |R_i|$。

为了将 AGM 约束引入抽样框架，我们首先--对于所有 $t \in R_i$，将每个 $W(t) = \mathrm{AGM}(R_{i+1} \bowtie \cdots \bowtie R_n)$。*查询分解 Lemma* [24] 确保不变式 (3) 成立
使用 AGM 边界来初始化 $W(t)$。

请注意，AGM 边界和 Olken 边界一般不具有可比性。如果最大频率较小，那么 Olken's bound
界限可能更小。然而，在最坏的情况下，最大频率可能和关系大小一样大，即 $M_A(R_i) = |R_i|$，而 Olken 的边界可能比它大得多（几乎是二次方）。
AGM 约束。事实上，当只给出关系大小时，AGM 约束是连接大小的最优约束。

根据上述观察，另一个自然的想法是，如果有部分频率信息，则将频繁值和不频繁值分开处理。例如，数据库系统可能已经收集了热门值。我们以 $R1(A1, A2) \bowtie R2(A2, A3) \bowtie R3(A3, A4)$ 的大小来说明这一想法。对于
如果 $|t' \bowtie R_i| = h$，否则为轻。让 R^H_i 成为 R_i 中的元组 t 是*重型的*。和 R^L_i 是以
R_i 灯图元集。连接 $R1 \bowtie R2 \bowtie R3$ 分解为 4 个子条目：
$r1 \bowtie r2_{x2} \bowtie r3_{x3}$，
其中每个 x_i 可以是 H 或 L。我们对每个子连接的大小设定了如下上限。(1) 对于 $R1^H \bowtie R2^H \bowtie R3^H$（实际上包括
2 of the 4 subjoins），我们使用 AGM 约束，即 $|R1| \cdot |R3|$。(2) 对于 $R1^L \bowtie R2^H$，我们将其约束为 $|R1| \cdot h$。(3) 对于 $R1^H \bowtie R2^L$，我们将其约束为 $|R2| \cdot h$。(注意，我们利用广义
在后两种情况下，采用奥尔肯算法）。

举个具体例子，假设 $|R1| = |R2| = |R3| = 10$。在 R_2 和 R_3 中，有一个频繁值在 100 次出现中
元组，而其余的值每个都出现在不到 10 个元组中。
因此，$h = 10$，$|R1^H| = |R2^H| = 100$。那么奥尔肯边界为 $|R1| \cdot 100 \cdot 100 = 10^8$。AGM 界值为 $|R1| \cdot |R3| = 10^8$。改进后的边界为

$$= |R1| \cdot |R2^H| + |R1| \cdot |R2^H| + |R1| \cdot h = 100 \cdot 100 + 100 \cdot 100 + 10 \cdot 10.$$

为了改进奥尔肯的松散边界，同时也避免像广义乔杜里算法那样运行完整动态编程的高成本，我们首先从 r_0 开始执行一系列随机行走。由于 $w(t)$ 的估计是基于中心极限定理，而中心极限定理需要最小样本量才能成立，因此这些随机行走将使我们只对*具有足够行走次数的*图元子集获得 $w(t)$ 的估计值。

更确切地说，我们设置一个 ϑ 的阈值，对于任何元组 t，如我们计算 $w(t)$ 的置信区间，并将 $W(t)$ 设为 $W(t)$ 的上界。

使用游走连接估计器[22]计算这个置信区间。由于这些上限是以概率方式计算的，因此对于某些 t *而言*，不变式（3）不成立的可能性很小。在这种情况下，我们以自下而上的方式提高 $W(t)$，从而恢复所有 $W(t)$ 的不变式 (3)。对于任何没有经历过足够多随机行走的图元 t，我们设置 $W(t)$ = *未知*。

接下来，我们开始执行算法 1。每当我们找到一个元组 t，使得有一些 $t' \in t \ltimes R_i$ 的 $W(t')$ = *未知数时*，我们就运行动态编程算法，为每个这样的 t' 计算 $W(t') = w(t')$。注意，这里我们将运行动态程序
自上而下，只计算和记忆计算 $w(t')$ 所需的 $W(u)$。事实上，它们正是 $R_{i+1}, \ldots, R_n$ 中参与连接 $t' \bowtie R_{i+1} \bowtie \cdots \bowtie R_n$ 的图元。这种 DP 探索的成本与此类图元的数量成线性关系（通常远小于 $w(t')$）。
另一个观察结果是，算法 1 还能执行随机的
r_0 来寻找样本，尽管它是在 $W(t)$ 的指导下进行*的。尽管*如此，它们仍然形成随机漫步，[22]中提供的估计器仍可用于这些随机漫步，因此，当
算法 1 返回样本时，它们仍可持续贡献样本。
的所有图元的置信区间缩小。
随机游走已通过（这将导致更严格的上界
$W(t)$ 的数量越多，返回的样本就越多）。

5 其他类型的连接

5.1 无循环连接查询

在本节中，我们将展示如何推广我们的抽样算法
非循环连接查询。我们以树状结构组织关系
这样 R_1 就是根。与之前一样，我们还添加了一个 R_0，其中包含只有一个元组 r_0 与 R_1 中的每个元组连接。对于每个非叶关系 R_i，让 R_{i_1}、R_{i_2}、……成为 R_i 的子关系。我们使用 $R_i \prec R_j$ 表示 R_i 是 R_j 的祖先。
树

对于任意 R_i 和任意 t 元组 $t \in R_i$，$w(t)$ 的定义会发生变化至

$$w(t) = |t \ltimes (\ltimes_{i:R_i \prec R_j} R_j)| \cdot \prod_{R_k}$$

如果 R_i 不是叶子关系，我们定义 $w(t,$) 为

4.4 漫游连接作为初始化

最近，Li 等人[22] 提出了一种基于随机漫步的算法

来估计连接大小。为了估算 $|J|$，他们的算法在 H (I) 上执行了一系列从 r_0 到 R_n 的随机行走。这些随机游走会在 J 上生成非均匀但独立的样本，并且

仍可用于估计连接规模。我们的观察结果是，从 r_0 开始的每个随机行走不仅可以用来估计

$|J| = w(r_0)$，而且每个中间连接大小 $w(t) = |t \triangleright\triangleleft R_{i+1} \triangleright\triangleleft \cdots \triangleright\triangleleft R_n|$ 其中 $t \in R_i$ 是该随机行走路径上的一个元组。

将我们的抽样算法从链式连接扩展到非循环查询的基本思想是，每当我们到达一个下面有多个子树的关系（如图 1b 中的 R_2）时，我们就将抽样过程分支到两个子树中。这意味着 R_i 中的一个元组可能在 R_k 中有多个子元组，这也解释了为什么我们还需要引入 $w(t, R_k)$，它是子连接的大小，其中 t 但只限于其 subtrees 中的一个。

同样，我们引入连接大小上界 $W(t)$ 和 $W(t, R_i^k)$ 以帮助取样。不变式 (1-3) 现在变为：

$$W(t) \geq w(t), \forall t; \quad (4)$$

$$W(t, R_i^k) \geq w(t, R_i^k), \forall t \in R_i, R_i \text{ 为非叶}; \quad (5)$$

$$W(t) = 1, \forall t \in R_i, R_i \text{ 是一片叶子}; \quad (6)$$

$$W(t) \geq \sum_k W(t, R_i^k), \forall t \in R_i, R_i \text{ 为非叶}; \quad (7)$$

$$W(t, R_k) \geq W(t \rtimes R_i^k), \forall t \in R_i, R_i \text{ 为非叶}。 \quad (8)$$

而 5 个不变式中的 4 个是从链大值不变量（7）是新的。直观地说，由于不同的分支是独立的，一个分支的任何连接结果都可以通过 t 与另一个分支的任何连接结果相结合，*从而*使不同分支的连接大小相乘。

算法 2：循环采样（t，R_i）

1 $W'(t, R_i) \leftarrow W(t, R_i)$；
2 $W(t, R_i) \leftarrow W(t \rtimes R_i)$；
3 拒绝概率为 $1 - W(t \rtimes R_i)/W'(t, R_i)$；
4 $t \leftarrow$ 一个随机元组 $t \in (t \rtimes R_i)$，概率为 $W(t')/W(t \rtimes R_i)$；
5 $W(t) \leftarrow W(t)$；
6 $W(t) \leftarrow \prod_k W(t, R_i^k)$；
7 拒绝概率为 $1 - \prod_k W(t, R_i^k)/W(t)$；
8 在 S *后面*加上 t；
9 **如果** R_i *是叶子，***则返回**；
10 **对** R_i *的***每个** *子关系* R_k **做**
11 无环样本（t，R_i）；
12 **结束**

算法 1 现在变成了递归算法，如图所示

算法 2，初始调用为 ACYCLIC-Sample(r_0, R_1)。除非样本被拒绝，否则终止时会将 S 报告为样本。与链式连接情况类似，算法 2 中的第 2 行和第 6 行不会影响正确性，但会迭代收紧连接大小上限，从而随着时间的推移提高采样效率。下面的定理证明了算法的正确性，其证明见 附录 B。

定理 5.1 *对于任何非循环连接，算法 2 返回* J *中每个连接结果的概率为* $1/W(r_0)$，*其中* $W(r_0)$ *是算法运行前的值。*

中讨论的所有方法都是为了初始化连接大小上限。第 4 节可以沿用，但需要注意一些问题。特别是，非循环查询的 AGM 约束并不像链式连接的情况那么简洁；详情请参考 [6]。Olken 的上限仍然是子树中每个关系中最大频率的乘积。Chaudhuri 等人的实例化需要精确的连接大小，而这仍然可以通过动态编程来计算。事实上，动态编程递归与链式连接中的后向追踪思想相同，只是现在使用了不变式 (6-8)（在 (7) 和 (8) 中用 = 替换 ≤）。

关系子集，这样查询就变成了一个连接的、无循环的查询。例如，对于图 1c 中的循环查询，我们可以删除 R_3、R_4 或 R_5 中的任意一个。我们将删除的关系称为*残差查询*表示为 H_r，主非循环查询表示为主非循环查询。*骨架查询*，记为 H_s。请注意，对于非常复杂的循环查询，例如，当查询图是一个完整图，且≥ 5 属性和 ≥ 10 个关系时，残差查询可能大于的骨架查询。但对于实践中出现的大多数查询，k 残差查询往往很小。我们定义

$$M_r = \max_{v_i \in \mathrm{dom}(A_i)} |\{t : t \in H_r, \pi_{A_i}(t) = v_i, \text{对于所有 } A_i \in A(H_s) \cap A(H_r)\}|$$

即 H_r *中与* H_s *的*共同属性固定为特定值后的最大图元数。请注意，这等同于 H_r 中能与来自 H_s *的*任何连接结果连接的最大图元个数。在某些情况下（例如下面的三角形查询），M_r 的值可以通过简单的方法获得。否则，我们可以全面评估残差查询 H_r 来计算 M_r。 循环查询的抽样算法如下。

算法 3：循环采样（H_s，H_r）

1 $t \leftarrow$ 使用算法 2 从 H_s 中提取样本；
2 拒绝概率为 $1 - |t \rtimes H_r|/M_r$；
3 从 $t \rtimes H_r$ 返回一个均匀样本；

定理 5.2. *算法 3 返回* J *中每个连接结果的概率为* $\frac{1}{|\text{先生}| \cdot |H_s|}$ *的条件下返回* J *中的每个连接结果从* H_s.

于证明考虑整个查询的任何连接结果。这些图元 H_s 利用条件，以概率 $1/|H_s|$ 进行采样。然后，算法以概率 $|t \rtimes H_r|/M_r$ 决定是否继续采样，如果继续采样，则以概率 $|t \rtimes H_r|$ 对剩余的图元进行采样。因此，总体采样概率为

$$\frac{1}{|\text{先生}|} \cdot \frac{|t \rtimes H_r|}{M_r} = \frac{1}{|\text{先生}| \cdot |H_s|} \cdot \frac{|t \rtimes H_r|}{M_r} \quad \square$$

5.2 循环查询

在本节中，我们将把采样算法扩展到循环查询。我们通过移除查询超图中的一个

举例说明。我们用最简单的循环连接，即*三角形查询* $_{R1}$ $(A, B) \triangleright\triangleleft_{R2} (B, C) \triangleright\triangleleft_{R3} (A, C)$ 来说明算法。我们打破将查询分为 $H_s = {}_{R1} \triangleright\triangleleft_{R2}$ 和 $H_r = {}_{R3}$。请注意，在这种情况下 $M_r = 1$ 是微不足道的，假设关系代数语义是 a 关系不能包含重复的元组。如果 R_3 包含另一个属性，比如 D，或者包的语义是用。如果是这样，我们可以扫描一次 $_{R3}$ 来计算 M_r。

对于骨架查询，如果我们使用 Chaudhuri 等人的实例化方法，算法将首先在 R_1 中采样一个图元 t_1，采样概率与它在 R_2 中加入的图元[illegible]量成正[illegible]然后在 R_2 中均匀随机抽取其中一个图元，即 t_2。然后，在 R 中采样一个图元 t，采样概率与它在 R 中加入的图元数量成正比。算法计算 $(t_1, t_2) \rtimes_{R3}$，并找到唯一的元组 $t_3 \in R3$ 与 t_1 和 t_2 连接（如果存在的话）。由于此处 $M = 1$ 情况下，如果 t_3 存在，算法只需返回 (t_1, t_2, t_3)，否则拒绝。

备注有趣的是，在三角形查询中，我们的算法与*楔形抽样*算法[33]不谋而合，而*楔形抽样*算法是从大型图中抽取三角形的最佳算法之一。同样，在对有 4 个顶点的子图模式进行采样时，我们的算法会退化为*路径采样*算法 [17]。请注意，这种子图模式查询是多向连接问题的一个特例，在这种情况下，共有 4 个属性，而每个关系恰好包含两个属性。从*本质上讲，我们的算法将这些图采样算法推广到了任意超图。*

另一个问题是如何将查询分解为骨架查询和剩余查询。这不会影响算法的正确性，但会决定算法的采样效率。定理 5.2 建议我们分解查询，使 $_{Mr}$ - |Hs | 最小。我们总是可以通过评估残余查询来计算 $_{Mr}$，残余查询通常很小，但计算 $_{Hs}$ 的连接大小可能很昂贵。不过，估计出 $_{Hs}$ 的连接大小足以做出足够好的决策（如查询优化），因此我们
可以使用任何现有的连接大小估算技术，具体取决于是否有索引和/或先验统计数据，例如使用 wander join [22]。

5.3 选择谓词

有两种方法可以在我们的抽样算法中支持选择谓词。首先，我们可以简单地将谓词下推到关系中。更确切地说，我们用谓词（通过扫描或索引探针）过滤每个关系，然后将过滤后的关系输入我们的抽样算法。事实上，对于选择性很强的谓词，这可能仍然是最好的选择。特别是在我们使用动态编程计算 $W(t) = w(t)$ 作为精确子连接大小的算法版本中，由于我们需要花费输入大小的线性时间来建立采样框架，因此在运行动态程序之前首先过滤关系以减小其大小总是有益的。

在其他实例中，我们的目标是降低设置成本，或者当选择谓词的选择性不强时，在采样过程中执行选择谓词会更有效。
过程。更确切地说，在算法 1 和 2 中，我们用 $_{\sigma\varphi}(t \rtimes _{Ri})$ 替换每个 $t \rtimes Ri$，其中 φ 是 $A(_{Ri})$ 上的选择谓词（如果有的话）。请注意，无需更改算法 3，因为
骨架查询已包含所有属性。

6 实验

这项工作的目标是设计可扩展的高效方法，以便从连接中生成简单的随机样本。一旦从数据集（在我们的例子中是连接结果）中获得简单随机样本，就可以利用获得的样本集执行许多不同的有用分析任务。我们可以设计特定的方法，在*不使用简单随机样本的情况下*逼近某些特定的分析功能（例如，用于 SUM 和 COUNT 在线聚合的波纹连接 [14] 或 wander 连接 [22]），但这些方法无法推广到回归和分类等更复杂的任务中。另一方面，如第 1 节所示，简单随机抽样的适用范围更广。因此，我们实验评估的重点是了解不同方法在从各种连接中产生*统一和独立样本时的*性能，而不是如何将这种简单随机样本用于不同的分析目的，这在许多不同领域都是众所周知的课题，而不是我们研究的重点。

6.1 实验装置

我们在论文中提出的几个替代方案中实例化了建议的抽样框架。我们特别探讨了以下方法：

- 完全连接（FJ）：计算完全连接结果的基准方法。由于我们关注的是自然连接（等价连接），因此我们使用

哈希连接作为基础方法（使用 PostgreSQL 返回的优化查询计划）。我们在下面报告的完全连接时间只包括 PostgreSQL 运行连接到完成的时间。我们还在商业数据库系统上测试了 FJ，结果与此类似。

- 扩展的奥尔肯（EO）：混合方法，该方法是在奥尔肯的基础上引入的。
第 4.3 节将 AGM 约束与
是对最初的奥尔肯算法的推广[25]。
- 精确权重 (EW)：动态编程方法，用于计算连接超图实例中每个元组的精确权重 $w(t)$。这是*对原始 Chaudhuri 等人的算法* [8]，第 4.2 节讨论了链式连接，第 5.1 节讨论了任意连接。
- 在线探索（OE）：结合使用在线
的元组进行聚合（例如通过游走连接）。
行走次数，以及通过 EW（仅针对连接超图实例中的元组 t）按需探索连接超图实例中以元组 t 为根的连接子树，如第 4.4 节所述。
- 反向采样 (RS)：这是一种*仅适用于多向外钥连接的方*法，由 [2] 提出，并在 [3] 中进行了回顾。
第 2.3 节，只从 R_n 中采样，并使用 R_n 中的采样记录向后遍历 R_1，以产生连接的样本。

对于所有方法，我们都会报告检索特定数量样本所需的总运行时间。总运行时间不包括建立 B 树等所有索引的时间，这是所有方法的共同成本，因为它们都需要在连接属性上建立索引。

需要注意的是，尽管我们只讨论了多向链式连接的在线探索方法，但由于徘徊连接适用于任何连接拓扑 [22]，因此只要使用本节讨论的更新版 EW，同样的想法也能轻松用于任何连接查询。

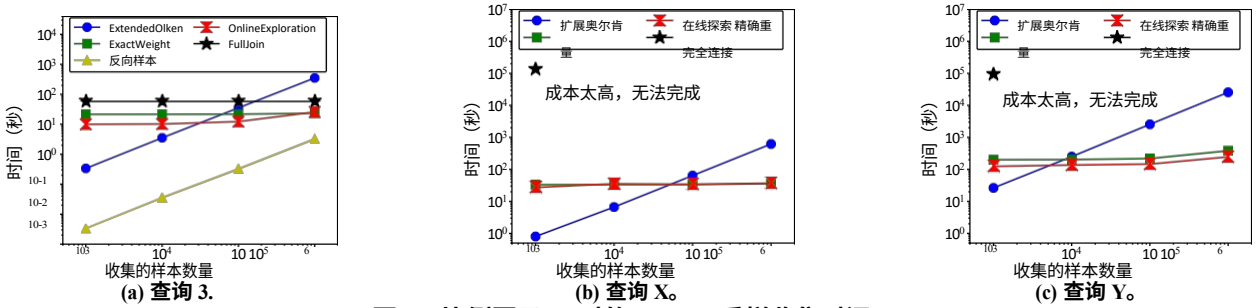
5.1 按需探索连接子树。此外，虽然我们的抽样框架会持续返回随机样本，但生成样本的探索路径可被视为随机漫步，这就增加了该路径上任何元组的随机漫步次数，并提高了以这些元组为根的连接子树的漫步连接估计值。

最后，如第 5.2 节和第 5.3 节所示，我们的连接采样框架实例化的任何方法，即 EO、EW 和 OE，都可以支持循环查询和选择谓词

很容易需要注意的是，通过删除关系并分别生成骨架查询 H_s 和残余查询 H_r 来"打破 "循环的方法有很多种。我们在
在第 5.2 节末尾，优化我们的选择，选出最佳策略
来削减一个循环。当 *EO、EW 和 OE* 应用于循环连接查询时，这种优化*就包含在*它们的*成本中。*

我们用 C++ 实现了所有方法，并在一台运行 Ubuntu 14.04 和 E5-2609 处理器的机器上进行了实验。
频率为 2.4GHz。所有实验都是在数据完全位于内存中的情况下进行的；磁盘 IO 仅在数据加载时发生。

TPC-H 数据集。我们的第一个数据集由使用标准 TPC-H 基准生成的数据组成。我们对生成器进行了修改，使其能够根据 Zip-fian(1)分布生成每个订单的行项目数，从而为数据的连接度添加一些偏度。默认情况下，我们使用的缩放因子为 10（大约 10GB），但在实验中，我们将缩放因子从 1 调整到了 40



(a) 查询 3. **(b) 查询 X。** **(c) 查询 Y。**

图 3：比例因子 10 时的 TPC-H 采样收集时间。

来展示这些算法在处理不同规模数据时的性能和扩展性。我们探讨了 TPC-H 数据的三个连接查询。附录 C 显示了这三个 SQL 查询。

- 查询 3 (Q3)：是*多向外键链式连接*，连接表 *customer*、orders 和 *lineitem* 上的主键和之间的外键。这种连接的卡入度正好是 *lineitem* 表的大小。
- 查询 X (QX)：是*一般的多向链式连接*，它连接*国家*、*供应商*、*客户*、*订单*和*项目*。它是一个*链连接*，但不再是外键连接，因此该链连接的卡入度比任何表都要大得多。
- 查询 Y (QY)：是*多向循环连接查询*，它连接了*供应商*、*客户 1*、*订单 1*、*行项目 1*、*行项目 2*、*订单 2*、*用户 tomer2*，它连接回*供应商*表。与输入表的大小相比，这种循环连接查询的连接大小也非常大。

社交图谱数据。第二个数据集是[7]中使用的 twitter 友情链接和用户配置文件集。该数据集包含一个 "名人 "表，其中包含粉丝数超过 10,000 的 twitter 用户，称为 "*热门用户*"；另一个表包含数据集中的所有 twitter 用户，称为 "*twitter 用户*"。每个表中的一条记录代表一个友谊链接，该链接以一个用户 ID 为来源，以另一个用户 ID 为目的地。我们对该数据集进行了以下三次查询：

- 查询 T (QT)：popular-user、twitter-user 和 twitter-user 之间的*三角连接*。
- 查询 S (QS)：popular-user、twitter-user、twitter-user 和 twitter-user 之间的*平方连接*。
- 查询 F (QF)：两个流行用户表和两个 twitter 用户表之间的*雪花连接*，结果是*一棵树*为底层连接超图的*拓扑结构*。

popular-user表有1.65亿条记录，占用3.1GB；twitter-user表有10亿条记录，占用19GB。我们还对 twitter-user 表进行了向下采样，以获得不同大小的表，用于可扩展性测试。附录 D 显示了 SQL 查询。

6.2 TPC-H 实验

采样效率。我们在默认规模因子为 10（约 10GB）的 TPC- H 数据集上运行每种算法，测试它们收集 10^3、10^4、10^5 和 10^6 个样本的速度。对于 Q3，我们还采用了反向采样法 (RS)，而 QX 和 QY 无法采用这种方法。实验结果如图 3 所示。请注意，X 轴和 Y 轴都是对数刻度。如图 3a 所示，由于 Q3 是一个简单的多向外键链连接，在这种情况下，RS 的性能最好。

此外，由于 FJ 是简单的多向外键链连接，因此也相当高效。尽管如此，EW 和 OE 都表现出了良好的性能，EO 在产生较少样本数时效率较高，但随着所需样本数的不断增加，其较低的采样效率（由于高拒绝率）会导致比其他方法高得多的成本。

QX 是一般的多向链连接（即某些连接属性不再是主键、外键关系），因此连接大小远大于输入关系大小。因此，如图 3b 所示，FJ 的成本太高，一天内无法完成。EO 表现出与 Q3 实验类似的模式。OE 和 EW

在这种情况下，两者的性能相似；都能在 35 秒内生成 ^{106} 个样本。

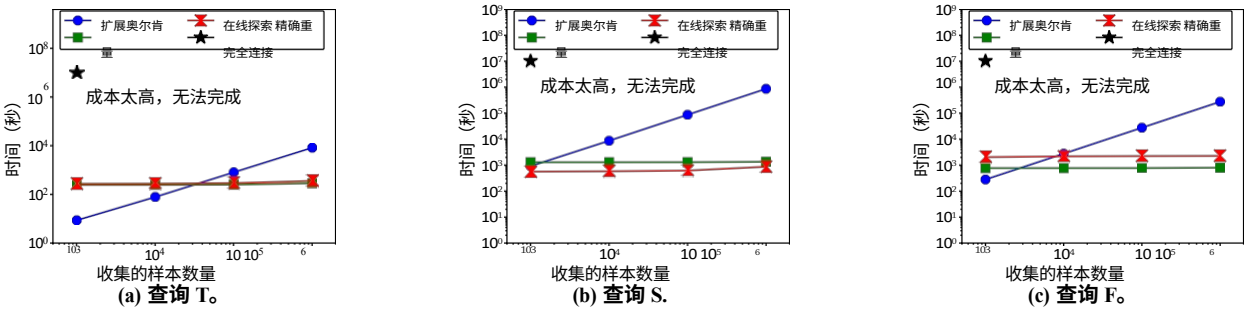
最后，QY 是一个循环查询，并非所有连接条件都是主键、外键关系，这使得 FJ 非常昂贵。如图 3c 所示，EO 仍显示出类似的模式，但当它需要返回更多样本时，成本就会高得多，这是因为循环查询（比非循环查询）的拒绝率通常更高。在这种情况下，OE 的性能总是优于 EW。由于连接中的关系数多于 QX 中的关系数，EW 变得更加昂贵。OE 制作 ^{106} 个样本需要 245 秒，而 EW 需要 381 秒。对于 QX 和 QY，OE 和 EW 的效率都比 FJ 高几个数量级。

可扩展性我们使用 QY 测试不同方法的可扩展性，当数据库以不同的规模因子生成时

(从 1 到 40，即数据量从 1GB 到 40GB）。我们要求每种方法收集 ^{106} 个样本，结果如图 5a 所示。FJ 的成本过高，无法完成除

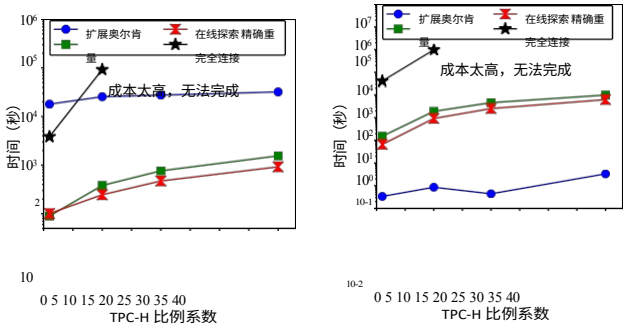
随着数据量的增大，最小的情况也会出现。另一方面，根据我们的框架实例化的方法表现出非常好的可扩展性。EO 主要只依赖于连接拓扑（AGM 约束）和最大度数信息，即使数据量增加（因为数据分布相同）也相对稳定。OE 和 EW 都与输入表大小的增加呈大致线性关系，这符合他们的分析预期。但 EW 与总输入关系大小呈严格的线性关系，而 OE 由于其在线探索的性质，情况未必如此。因此，OE 的性能最好，而且随着关系越大，OE 和 EW 之间的性能差距也越大。对于比例因子 40，OE 的性能几乎是 EW 的 2 倍。

我们还测量了每种算法报告第一个样本的时间，结果如图 5b 所示。毫不奇怪，由于初始化开销较低，EO 报告第一个样本的速度最快，而 OE 报告第一个样本的速度比 EW 快。



(a) 查询 T。 (b) 查询 S. (c) 查询 F。

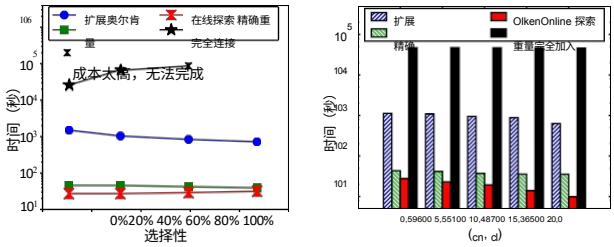
图 4：完整数据集的社交图谱收集时间。



(a) 收集 10^6 个样本的时间。 (b) 采集第一个样本的时间。

图 5：查询 Y 的 TPC-H 可扩展性测试。

因为它是在线探索。EW 报告第一个样本的速度最慢，因为它只有在运行动态编程公式计算出所有 $w(t)$ 之后才能开始报告样本。但是，一旦 EW 开始报告样本，它的采样效率是最高的，不会出现剔除样本的情况，而 OE 总是会剔除一些样本，EO 的剔除率最高。请注意，OE 的拒绝率实际上非常低，这是因为 wander 连接估计器返回了精确的上限估计值，以及 EW 所探索子树的精确 权重。



(a) 一个选择谓词。 (b) 两个选择谓词。

图 6：带有选择谓词的 QX

选择谓词。为了显示在存在选择谓词时各种方法的有效性，我们测量了在 QX 中添加 1 或 2 个选择谓词时收集 10^6 个样本所需的时间。

在第二个实验中（如图 6b 所示），我们添加了两个关于*国家*和 *l_extendedprice* 的选择谓词 $n\_nationkey \geq c_n \geq c_l$ on *lineitem*。我们将选择性固定在大约 20%，并测量使用不同方法收集 10^6 个样本所需的时间。FJ 仍然的成本比其他三种取样方法高出几个数量级。当国家密钥的谓词变得更具选择性时，EO、EW 和 OE 获取样本所需的时间都更短。这这是因为我们从国家表开始采样，而第一张表中的图元越少，被拒绝的次数也就越少。请注意，在这种情况下由于初始化成本低，OE 始终快于 EW 它们的运行速度至少比 EO 快一个数量级。

6.3 社交图谱实验

图 4a、4b 和 4c 显示了不同采样算法在完整社交图数据集上从三角形查询 (QT)、正方形查询 (QS) 和树形结构查询 (QF) 中收集 10^3、10^4、10^5 和 10^6 个样本所需的运行时间。

当样本总数增加时，EW 的运行时间并不会增加很多，因为它需要先建立权重表，然后才能生成样本。OE 在生成样本前的预热阶段付出的代价要小得多。当开始生成样本时，它按需建立权重，因此其初始采样率较小

但随着时间的推移会增加。当所需样本总数较少时，OE 和 EW 孰快孰慢取决于连接结构和输入大小。查询 S 有 3 张大表，而查询 T 和 F 只有 2 张表，因此查询 S 的 EW 动态编程（DP）将比其他两张表昂贵得多。因此，在查询 S 上，OE 比 EW 运行得更快。查询 T 在我们通过删除一个关系打破循环后，只是一个双向连接。查询 T OE 探索大部分连接图的成本大致为

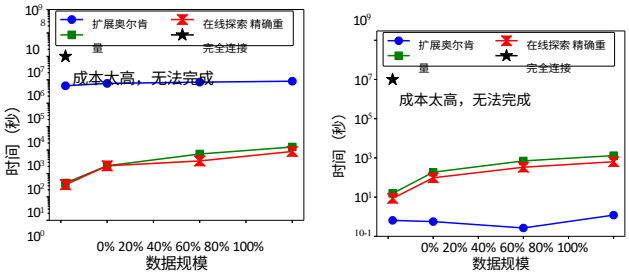
在第一个实验中（如图 6a 所示），我们只在表 *lineitem* 上添加了一个选择谓词 $l\_extendedprice \geq c_l$，并改变了查询的选择性（图元数目和查询次数之间的比率）。

查询中的选择谓词）。当选择性大于 50%时，FJ 仍然非常昂贵，而且无法完成。当查询的选择性降低时，EO 的采样效率

更高，因为在这种情况下拒绝率更低。另一方面，OE 和 EW 都能很好地适应不同选择性的查询。

与 EW 执行 DP 的成本相同。在查询 F 中，由于连接是非循环的，而且只有两个大表，EW 的 DP 成本实际上小于 OE 的探索成本，因此 DP 运行速度比 OE 快。然而，一旦需要足够多的采样，总运行时间将是相似的。EO 因其静态结构而具有恒定的采样率，但采样率远小于 EW 和 OE，即使只需要适量的采 样 ， 也会比其他两种方法慢。由于输入和输出的大小都很大，所以在所有三个查询中，完全连接都不会完成。

图 7a 显示了在不同大小的向下采样社交图上从正方形查询 (QS) 中获取一百万个样本所需的时间。随着数据大小的增加 ，EW 和 OE 的运行时间也在增加。



(a) 采集 10^6 个样本的时间。 (b) 采集第一个样本的时间。

图 7：查询 S 的社交图谱可扩展性测试。

算法将大致呈线性增长，而 EO 则呈对数增长。EW 和 OE 具有线性可扩展性的原因是，EW 和 OE 的大部分时间都花在 DP 上（OE 使用 EW 探索某些级别下的子树），而 DP 是线性扩展的。由于索引大小的增加，EO 的时间呈对数增长。但是，由于 EO 的拒绝率极高，它仍然比其他两种方法慢至少 3 个数量级。即使在 最小的数据规模上，完全连接仍然无法运行完成。

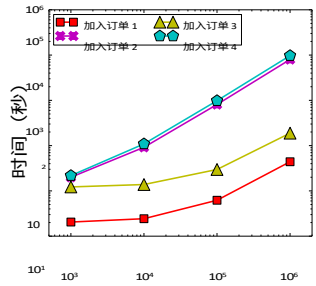


图 8：QS 不同连接顺序下获取 10^6 个样本所需的时间。

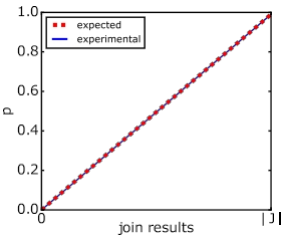


图 9：QX 的 KS 检验。

图 7b 显示了获取第一个样本的时间。EO 算法获取第一个样本的速度比 EW 和 OE 快得多，因为它不需要付出建立权重或预热的成本。但是，只要所需的样本数量足够多，EO 算法的总运行时间就会比 EW 和 OE 长很多。

图 8 显示了 OE 使用 QS 的不同连接顺序收集不同数量样本所需的时间，从图中可以看出，使用连接顺序 1 所需的时间最少，因为连接顺序 1 的连接结果数量最少，这是我们优化所选择的。

6.4 Kolmogorov-Smirnov (KS) 检验

回想一下，我们的抽样框架*保证返回真正的简单随机样本，只要使用的上限 $W(t)$ 确实是抽样过程中遇到的以图元 t 为根*

表 2：K-S 值，其中 $n = 100$ 万。

	采样算法	实验 d
查询 X	扩展奥尔肯	0.0003276
	精确重量	0.000774
	在线探索	0.000357
查询 T	扩展奥尔肯	0.00130
	精确重量	0.00131
	在线探索	0.00155

确实是均匀的（显然，它们是从

抽样方法的构建）。为此，我们从非循环查询 QX 和循环查询 Twitter 三角形查询 QT 中收集了 100 万个样本，并使用 Kolmogorov-Smirnov 检验（K-S 检验）测试了返回样本的累积分布是否符合预期的均匀分布。

我们对 $M = (m_1, m_2, \ldots, m_n)$ 进行排序，排序顺序与完全连接算法对 $J = (t_1, t_2, \ldots, t_{|J|})$ 中的图元进行排序的顺序相同。设 $o_J(m_i) = k$、
其中 $t_k = m_i$，则 M 的排序为 $o_J(m_i) > o_J(m_k)$。
所有 $k > i$。
如果我们的算法报告的样本是统一的，给定

任何 $m_i \in M$ 的比例 $\frac{i}{|J|}$ 将近似等于
比例 $\frac{o_J(m_i)}{|J|}$。在分析整个 M 的同时，通过比较这些比例作为累积概率分布的进展情况，我们可以比较我们的样本匹配程度有多接近
与预期的均匀分布一致。

这些预期累积分布与实验累积分布之间的最大差异为 d，即

K-S 评分。d 值越小，实验结果*越*接近预期结果（均匀分布）。此外，我们还计算出，当 $n = 100$ 万时，我们可以接受在 $\alpha = 0.01$ 时分布均匀的假设。

的子树的上限，这在我们针对不同连接拓扑的所有实例中都是如此，唯一的例外是游走连接可能会以极小的概率返回小于子树实际连接大小的估计值。由于我们使用的是 wander join 估计值的上限（即估计值+置信区间），因此理论上这种情况发生的概率几乎可以忽略不计。实际上，在我们的所有实验中，这种情况从未发生过。

尽管如此，我们还是使用著名的 KS 检验来验证基于我们的抽样框架的每种方法返回的样本

如果 $d < 0.00163$，则为显着水平。

我们在图 9 中绘制了一幅图，显示了使用 OE 从 QX 采集样本时的实验结果和预期的均匀分布。

表 2 列出了使用每种抽样算法进行两次查询后得出*的* d 实验值。所有实验都通过了 K-S 检验，显著性水平为 $\alpha = 0.01$。

6.5 从何处打破循环

从循环查询中采样时，一个重要的决定因素是不同的切分循环方式有多有效。我们按照第 5.2 节所述的方法实施了算法，但为了了解这一策略的有效性，我们手动配置了不同的循环中断方式，并测量了不同循环中断方式的采样率。有两种方法可以切断循环：一种是通过将一个连接条件改为选择条件来切断循环；另一种是删除一个关系，然后使用复合索引检查删除的关系中是否存在相应的边。虽然第二种方法更好，但并不总是可以使用，这将在第 5.2 节中介绍。

我们使用基于游走连接的优化算法来估算将循环连接排序为非循环连接的每种不同方法的大小。表 3 是查询 Y 的方向连接估算表。

表 3：| J | 打破循环的不同方案。

选择权	| J | 增加条件	| J | 打破关系
1	5.853×1017	1.53×1014
2	4.532×1018	1.45×1014
3	4.765×1017	4.27×1014
4	7.127×1012	7.57×1010
5	1.133×1013	— 无效 -
6	4.534×1017	9.24×1010
7	4.534×1018	— 同 1 -

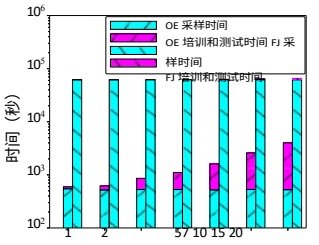


图 10：运行 SVM 所需的时间

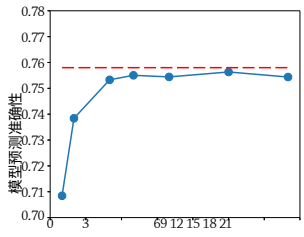


图 11：SVM 的精确度与 SVM 的差异

抽样数据 不同的样本量

我们的优化策略是剪切一个循环，使无循环查询的中间连接规模最小。我们在第 5.2 节中的优化策略总能找到最优切分。

6.6 机器学习的取样连接

如第 1 节所述，在连接点上进行随机抽样的一个有用应用是将连接点处理和机器学习以统一的方式结合起来，同时改进两者。最近关于联接学习的研究 [21, 32] 仅适用于线性回归模型的梯度下降。它们无法应用于 SVM 等更通用的模型。另一方面，经典的统计学习理论指出，训练一个模型所需的样本量为 $\frac{D+\ln(1/\delta)}{\epsilon}$），其中 D 是模型的 VC 维度[34]。这对任何分类模型都成立，因此通过连接取样，可以更普遍地解决这个问题。

在本小节中，我们进行了一项案例研究，训练 SVM 二进制分类器来预测 7 个 TPC-H 表连接结果中的 退货状态字段。我们修改了 TPC-H 生成器，使每个客户在随机选择的整个时间段的一半时间内向其运送商品时，商品退货率较高。价格在前 1%的商品的退货率也更高。该连接的完整连接结果有大约 17 万亿条记录，从中选择了 14 个特征来训练模型。TPC-H 数据生成器和连接查询的详情见附录 E。我们以 10K 到 200K 不等的样本大小对连接进行采样，并使用样本训练 SVM 模型当样本量达到 60K 左右时，该值已经非常接近于 0。

在效率方面，我们运行了采样框架的在线探索实例，以检索不同大小的样本。相比之下，我们还运行了完全连接，并直接从完全连接结果中采样。图 10 显示了两种方法在采样、训练和测试时间方面的比较。在线探索的运行速度比完全连接方法至少快两个数量级。请注意，这仅仅是采样时间的差异造成的；在相同样本量下，两种方法的训练和测试时间是相同的（但由于图中的对数比例不同，完全连接方法的训练和测试时间几乎不可见）。

7 备注和延期

实验结果清楚地证明了联接抽样框架的有效性、高效性、可扩展性和可扩展性。鉴于许多连接查询都是
正如我们的实验所显示的那样，该系统的运行成本极其昂贵。对任意连接查询进行支配采样极为重要。

在我们抽样框架的三个实例中，EO

。我们还独立抽取了 10 万个图元作为测试数据集。

图 11 显示了样本量变化时精确度的变化情况。随着样本量的增加，准确率先是急剧上升，然后在样本量足够大时收敛到一个恒定值。在本例中，准确率收敛到约 76%。这与 VC 理论非常吻合，即样本量与 $1/\epsilon$ 成正比。请注意，剩下的 24% 误差是模型本身的固有误差（如图 11 中红色虚线所示），也就是说，即使它是在所有数据上训练出来的。VC 理论中的ϵ表示由以下因素引入的额外误差

在初始化方面的开销最小，因为它只需要 AGM 约束（只取决于连接拓扑和关系的大小）和最大度数信息（或重击频率阈值）。因此，它的采样速度最快。但它的采样效率最差，原因是连接大小估计的上限较松，导致拒绝率较高。

当连接中的关系大小较小或适中时，EW 是有效的，因为它使用动态编程公式来精确计算所有 $w(t)$，其成本与连接中所有关系的大小之和成线性关系。但如果一个或多个关系的表大小很大，计算成本就会很高。此外，它的采样速度最慢（但一旦开始采样，由于拒绝率为零，它的采样效率最高）。

在大多数情况下，OE 是对 EW 的改进，因为它是一种混合方法，结合了在线聚合估计（任何连接子树的连接大小）和使用 EW 对某些连接子树的 $w(t)$ 按需探索。因此，它具有较低的拒绝率和相当快的响应速度来开始生成样本。由于它只对一小部分连接子树进行在线、按需的 EW 探索，因此当连接中某些关系的大小开始增加时，它也具有良好的可扩展性。

在所有情况下，EW 和 OE 都能轻松超越 FJ 和 EO，尤其是当连接大小远大于输入关系和/或连接拓扑结构复杂时。

扩展和限制。我们提出的连接抽样框架有一些扩展和限制。*首先*，它支持带有选择条件的自然连接或等连接，通过将 ϑ - 连接条件视为额外的选择谓词，它可以扩展到支持 θ - *连接，*但这种扩展在某些情况下并不容易，值得全面研究。

其次，通过在采样操作符的基础上添加逐组操作符和投影操作符，可以轻松支持不重复的逐组查询和投影查询。对于逐组查询，我们可以对样本执行逐组操作。当一个组包含少量记录时，该组中的样本可能很少。在这种情况下，我们可以将 group-by 子句转换为选择谓词，然后对样本进行取样。

在每组中独立进行。不带重复数据删除的投影是 SQL 的默认设置，可以通过在样本上执行投影来完成。但是，带去重的投影可能会改变连接结果的卡入度，因此不能直接在样本上执行，这需要进一步研究。

第三，与大多数随机抽样技术一样，我们的随机抽样框架依赖于对底层数据的高效随机访问。因此，当从一个关系进入另一个关系时，我们确实需要连接属性上的索引。这意味着我们的框架主要适用于内存数据库，在内存数据库中，通常需要为连接属性建立和维护索引结构。

当其中一些索引缺失时，即在某个关系 R 中的连接属性 A 上没有索引时，我们的抽样框架可以在启动抽样过程之前为 A 建立一个索引。这只会在 R 的大小上增加线性成本，总体采样成本仍会比计算完整连接小得多。

对于磁盘驻留数据，由于其随机访问，采样效率将大幅下降。不过，在这种情况下，完全连接也会变得更加昂贵。

最后，我们的采样框架可以处理更新。特别是，AGM 约束不受数据更新的影响；最大度和重击点可以在更新时有效地保持。因此，EO 方法无需改变。在 EW 中，每当数据接收到更新时，使用完整的动态编程方法重新计算所有 $w(t)$ 的权重会很昂贵，但是我们可以观察到，我们只需要更新权重（$w(t)$ 的权重），使用我们动态编程公式中相同的后向传播，对受更新影响的子树沿线的图元进行更新。因此，EW 只需稍作调整即可应对更新。对于 OE 方法，由于它依赖于 EW 和在线聚合技术（如 wander join），我们只需调整 wander join 以支持更新。这可以通过记住通过元组 t *的*成功和失败行走次数来实现，并在连接超图实例中的子图 t 发生更新时调整这些次数。尽管如此，支持更新仍然不是一件容易的事，我们将把对这一主题的探索作为今后的一项有趣的工作。

需要注意的是，Chaudhuri 等人[8]（EW 是对它的概括）之前关于在连接上采样的结果，以及使用奥尔肯方法的结果[25]（EO 是其延伸）也存在同样的局限性。

8 相关工作

Chaudhuri 等人在 1999 年 SIGMOD [8] 会议上的开创性工作中探讨了联接采样这个基本而又具有挑战性的问题。同一会议上还研究了从外键连接中抽样的特殊情况 [2]。在此之前，Olken 等人在 Olken 的博士论文 [25] 中总结了一系列工作，研究了从单个数据库表中随机抽样的问题。事实上，Chaudhuri 等人在研究中也探讨了 Olken 的技术。我们在第 2 节回顾了这些工作，它们不支持对任意多向连接（可能是也可能不是循环连接）进行随机抽样。最近的一些研究使用分层抽样来解决这个问题 [19]，通过连接运算符向下推送抽样，但不再保证总是返回随机样本。

我们的联接抽样框架利用了最近在推导联接查询的联接大小上限方面取得的进展。为此，AGM 边界是一项最新成果，它能得出

我们在连接抽样框架的一个实例中使用了任意连接 [6]。我们还使用了表的最大度数来帮助得出比 AGM 约束更严格的上限，其中表中元组的度数定义为其连接属性值在该表中的频率。文献 [18] 也探索了类似的原理，以便在给定度（频率）信息的情况下更好地估计连接大小，尽管这是在一个更理论化的环境中。

另一种获取连接规模上限的方法是使用*在线聚合*的思想 [15, 22, 26, 28, 38]。特别是，我们可以使用联接的在线聚合 [14, 22, 23] 来估计联接大小的置信区间（通过在联接查询中使用 count(*)）；取这个置信区间的上界就能高概率地得到联接大小的上界。关于联接大小的卡入度估计，还有许多其他工作[4, 6, 12, 29, 31, 36, 41]；只要我们的联接抽样框架能从联接超图中高效、高概率地生成任何部分联接的联接大小上界，它就可以采用其中任何一种技术来实例化一种新方法。

需要注意的是，有多种采样技术被用于开发在线聚合和近似聚合[10, 14, 22, 27, 37, 39]，然而，在以波纹连接[14]和徘徊连接[22]为代表的连接查询中，它们分别产生*均匀但非独立的样本*或*独立但非均匀的样本*。但随机抽样需要*均匀和独立的样本*。

采样是构建交互式近似系统的一种有用构造，例如 Aqua [1]、DBO [30]、BlinkDB [3]、Quickr [20]、G-OLA [42]、ABS [43] 等。但这些系统都无法支持连接的随机抽样。

人们对设计各种环境下最坏情况下的最优连接算法也重新燃起了兴趣 [9，35]，但通常情况下，除了外键密钥连接外，连接规模可能非常大，随机抽样将比计算完全连接更有效率和效果，正如我们的实验评估所显示的那样。

最后，随机抽样是一种基本机制，对许多分析任务都很有用，例如中位数、定量、推理、聚类、分类、核密度估计等复杂统计。其中许多任务都需要*独立和统一的*样本，以便构建良好和可扩展的近似值，或进行严格和主要的分析以确定性能。随机抽样对查询优化也很有用 [40]。

9 结论

本文重新探讨了连接取样这一经典而重要的问题。本文提出了一个通用的连接抽样框架，将现有的研究作为一个特例纳入其中，并可通过不同的方法对连接大小进行实例化，从而产生一个上界。我们证明了所提出的框架非常灵活和通用，可以处理任意多向非循环和循环查询，无论是否有选择谓词。大量实验证明了所提方法的效率和有效性。未来有趣的工作方向包括将所提工作集成到数据库引擎中，并扩展我们的研究和结果，以处理更复杂的查询，如ϑ连接和涉及嵌套查询块的连接。

参考文献

[1] Swarup Acharya, Phillip B. Gibbons, and Viswanath Poosala.1999.Aqua：使用近似查询答案的快速决策支持系统。In *VLDB*.754-757.

[2] Swarup Acharya 、 Phillip B. Gibbons 、 Viswanath Poosala 和 Sridhar Ramaswamy。1999.用于近似查询回答的联合综述。*SIGMOD 1999，ACM SIGMOD 数据管理国际会议论文集，1999 年 6 月 1-3 日，美国宾夕法尼亚州费城*。275-286.

[3] Sameer Agarwal、Barzan Mozafari、Aurojit Panda、Henry Milner、Samuel Madden 和 Ion Stoica。2013.BlinkDB：超大数据上具有有界误差和有界响应时间的查询。In *EuroSys*.29-42.

[4] Noga Alon, Phillip B. Gibbons, Yossi Matias, and Mario Szegedy.1999.在有限存储中跟踪连接和自连接大小。*第 18 届 ACM SIGMOD-SIGACT-SIGART 数据库系统原理研讨会论文集 (PODS '99)*.ACM, New York, NY, USA, 10-20. https://doi.org/10.1145/303976.303978

[5] Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, and Matei Zaharia.2015.Spark SQL： Spark 中的关系数据处理。In *Proc.ACM SIGMOD 国际数据管理大会*。

[6] Albert Attsertia、Martin Grohe 和 Daniel Marx。2013.关系连接的大小界限和查询 计划。*SIAM J. Comput.*42, 4 (2013), 1737-1767.

[7] Meeyoung Cha, Hamed Haddadi, Fabrício Benevenuto, and P. Krishna Gummadi.2010. 衡量 Twitter 中用户的影响力：百万追随者谬论》。*ICWSM*.

[8] Surajit Chaudhuri、Rajeev Motwani 和 Vivek Narasayya。1999.论连接上的随机抽 样 。In *Proc.ACM SIGMOD 数据管理国际会议*.

[9] Shumo Chu, Magdalena Balazinska, and Dan Suciu.2015.从理论到实践：并行数据库系统中的高效连接查询评估。In *SIGMOD*.63-78.

[10] Bolin Ding, Silu Huang, Surajit Chaudhuri, Kaushik Chakrabarti, and Chi Wang.2016. 采样 + 寻找：具有分布精度保证的近似聚合 Guarantee.In *SIGMOD*.679-694.

[11] Alin Dobra、Chris Jermaine、Florin Rusu 和 Fei Xu。2009.DBO 中的涡轮增压估计收敛。*Proc. International Conference on Very Large Data Bases*.

[12] Sumit Ganguly, Phillip B. Gibbons, Yossi Matias, and Abraham Silberschatz.1996. 用于抗偏斜连接尺寸估计的双焦采样。In *SIGMOD*.271-281.

[13] P.J. Haas.1997.在 线 聚合的大样本和确定性置信区间。In *Proc.Ninth Intl.科学与统计数据库 管理*。

[14] P.J. Haas and J. M. Hellerstein.1999.在线聚合的波纹连接。在 *Proc.ACM SIGMOD 国际数据管理大会*。287-298.

[15] J.J. M. Hellerstein, P. J. Haas, and H. J. Wang.1997.在线聚合。In *Proc.ACM SIGMOD 数据管理国际会议*。

[16] Christopher Jermaine、Subramanian Arumugam、Abhijit Pol 和 Alin Dobra。2007.使用 DBO 引擎的可扩展近似查询处理。In *Proc.ACM SIGMOD 数据管理国际会议*。

[17] Madhav Jha、C. Seshadhri 和 Ali Pinar。2015.路径采样：用于估算 4 顶点子图数量的快速且可证明的 方法。In *WWW*.

[18] Manas Joglekar and Christopher Re.2016.程度决定一切：使用程度 信息优化多向连接。In *ICDT*.

[19] Niranjan Kamat 和 Arnab Nandi.2016. 分层抽样中的完美随机性和最大随机性。*CoRR* abs/1601.05118 (2016)。

[20] Srikanth Kandula、Anil Shanbhag、Aleksandar Vitorovic、Matthaios Olma、Robert Grandl、Surajit Chaudhuri 和 Bolin Ding。2016.Quickr：在大数据集群中轻松逼近 复杂的 AdHoc 查询。In *SIGMOD*.631-646.

[21] Arun Kumar, Jeffrey Naughton, and Jignesh M. Patel.2015.在归一化数据上学习广义 线性模型。In *SIGMOD*.

[22] 李飞飞、吴斌、易珂、赵卓悦。2016.Wander Join： Wander Join: Online Aggrega- tion via Random Walks.In *SIGMOD*.615-629.

[23] Gang Luo, Curt J. Ellmann, Peter J. Haas, and Jeffrey F. Naughton.2002.可扩展哈希波纹连接算法。In *Proc. 数据管理国际会议*。

[24] Hung Q Ngo, Christopher Ré, and Atri Rudra.2013.Skew Strikes Back：联接算法理论的新发展 .(ArXiv:1310.3314v2

[25] F.Olken.1993.*Random Sampling from Databases*.博士论文。加州大学伯克利分校 。

[26] Niketan Pansare、Vinayak R. Borkar、Chris Jermaine 和 Tyson Condie。2011. 大型 MapReduce 作业的在线聚合。In *Proceedings of the VLDB En- dowment*, Vol. 4.

[31] Florin Rusu、Zixuan Zhuang、Mingxi Wu 和 Chris Jermaine。2015.工作量驱动的反连接卡丁性估计。*ACM Trans.Database Syst.*40, 3 (2015), 16.

[32] Maximilian Schleich 、Dan Olteanu 和 Radu Ciucanu 。2016. 通过因子化连接学习线性回归模型》。*2016 年国际 数据管理大会（SIGMOD ）论文集*。

[33] C Seshadhri、Ali Pinar 和 Tamara G Kolda。2013. 图上的三元测量： 楔取样的力量。在 *SDM* 中。

[34] V.V. N. Vapnik 和 A. Y. Chervonenkis.1971. 论事件相对频率对其概率的均匀收敛.*概率论及其 应用* 16 (1971)，264-280。

[35] Todd L. Veldhuizen.2012.Leapfrog Triejoin ： 一种最坏情况下的最优连接算法。*CoRR* abs/1210.0481 (2012)。

[36] David Vengerov、Andre Cavalheiro Menck 和 Mohamed Zait。2015.受滤波器条件限制的连接大小估计。*Proc. International Conference on Very Large Data Bases*.

[37] Lu Wang, Robert Christensen, Feifei Li, and Ke Yi.2016.空间在线采样 和聚合。*超大规模数据库国际会议论文集》*。

[38] Sai Wu、Shouxu Jiang、Beng Chin Ooi 和 Kian-Lee Tan。2009. 分布式 在线聚合。*PVLDB* 2, 1 (2009), 443-454.

[39] Sai Wu、Beng Chin Ooi 和 Kian-Lee Tan。2010. 多查询在线 聚合的连续采样。*SIGMOD*.651-662.

[40] Wentao Wu, Jeffrey F. Naughton, and Harneet Singh.2016. 基于采样的 查询再优化。In *SIGMOD*.1721-1736.

[41] Feng Yu 、Wen-Chi Hou 、Cheng Luo 、Dunren Che 和 Mengxia Zhu.2013.CS2：用于查询估计的 新数据库概要。In *SIGMOD*.469-480.

[42] Kai Zeng、Sameer Agarwal、Ankur Dave、Michael Armbrust 和 Ion Stoica。2015.G-OLA： 用于大数据交互式分析的通用在线聚合。In *SIGMOD*.913-918.

[43] Kai Zeng 、Shi Gao 、Jiaqi Gu 、Barzan Mozafari 和 Carlo Zaniolo 。2014.ABS ： 具有准确性保证的可扩展近似查询系统。In *SIGMOD*.1067-1070.

[27] Chengjie Qin and Florin Rusu.2013.用于并行在线计算的采样估计器聚合。In *BNCOD*.204-217.

[28] Chengjie Qin and Florin Rusu.2014.PF-OLA：并行在线聚合的高性能框架。*分布式与并行数据库》* 32，3（2014）， 337-375。

[29] Florin Rusu 和 Alin Dobra。2008.用于连接大小估计的草图。*ACM TODS* 33, 3 (2008).

[30] Florin Rusu, Fei Xu Luis Leopoldo Perez, Mingxi Wu, Ravi Jampani, Chris Jermaine, and Alin Dobra.2008.DBO 数据库系统。In *SIGMOD*.1223-1226.

A 致谢

李飞飞、赵卓悦和罗伯特-克里斯滕森（Robert Christensen）部分获得了国家自然科学基金（NSF）1251019、1302663、1443046、1619287和国家自然科学基金（NSFC）的资助。61729202。胡晓和易可得到香港研究资助局的资助：GRF-16211614、GRF-16200415 和 GRF-16202317。邵逸夫奖作者非常感谢 SIGMOD 匿名审稿人提供的宝贵反馈意见。

B 定理 2 的证明

证明*部分树 P* 是如下递归定义的关系的子集：

(1) $\{R_0\}$ 是一棵局部树。

(2) 对于任意一棵部分树 P 和任意 $R_i \in P$，$P \cup \{R_k \text{ for all } k\}$ 是一棵部分树。让 ∂P 成为 P 的*边界，*即它包含所有的 P 中其子关系（如果有的话）不属于 P *的*关系。

如果 $T(P)$ (resp. $T(\partial P)$) 恰好包含每个 $R_i \in P$ (resp. ∂P) 中的一个元组，则称 $T(P)$ (resp. $T(\partial P)$) 为 P (resp. ∂P) 的*部分连接结果*。

我们将证明，对于任何局部树 P，它的任何局部连接结果 $T(P)$ 被算法采样的概率为 $\sum_{t \in T(\partial P)} w(t) / W(r_0)$.那么将 P 作为整棵树，并应用不变式 (6) 证明定理。

证明是根据 P 的大小归纳得出的。当 P 只包含 R_0 时的基本情况是微不足道的。现在假定，对于严格包含在 P *中的*任何部分树，证明都是成立的。 我们删除 P 的所有子树。中的一个关系，例如 R_i，得到一个更小的局部树 P'。通过根据归纳假设，我们知道 $T(P)$ 中的元组有的概率 $\sum_{t \in T(\partial P')} w(t) / W(r_0)$.

要从 $T(P')$ 到 $T(P)$，算法必须在每个 R_k 中成功获取相应的元组。假设 $t_i \in R_i$ 是在 R_i 中采样的元组，$t_k \in R_k$ 是在 R_k 中采样的元组。算法 *i* *i* *i*

i

首先决定它是否应该转移到 R_i 的子代上，因为 R_i 的子代有可能
第 7 行中的 bility $\sum_k W(t_i, R^k)/W(t_i)$。如果是这样，它就会递归

对每个 R_i 调用 ACYCLIC-SAMPLE(t_i, R_i)。在这个调用中，算法首先决定是否从 R^k 中以概率
第 3 行中的 $W(t_i \rtimes R^k)/W(t_i, R^k)$，然后以概率采样 t^k。
第 4 行中的 $W(t^k_i)/W(t_i \rtimes R^k)$。因此，$T$（$P$）的概率是所有采样为

$$\frac{\sum_{t \in T(\partial P')} W(t)}{W(r_0)} \cdot \frac{\sum_k W(t_i, R^k_i)}{W(t_i)} \cdot \sum_k \frac{W(t_i \rtimes R^k)}{\sum_k Z\, W(t_i, R^k_i)} \cdot \frac{W(t^k_i)}{W(t_i \rtimes R^k_i)}$$

$$= \frac{\sum_{t \in T(\partial P')} W(t)}{W(r_0)} \cdot \frac{\sum_k W(t^k_i)}{W(t_i)}$$

$$= \frac{\sum_{t \in T(\partial P)} W(t)}{W(r_0)}. \quad \square$$

□

C TPC-H 系列

Q3 - 行项目与其关联的订单和下订单的客户相连：

```
SELECT c_custkey, o_orderkey, l_linenumber
FROM customer, orders, lineitem
WHERE  c_custkey  =  o_custkey
    AND  l_orderkey  =  o_orderkey
    ;
```

QX - 同一个国家的供应商和客户，以及客户的购买历史。

```
SELECT nationkey, s_suppkey, c_custkey,
        o_orderkey, l_linenumber
FROM 国家、供应商、客户、订单
    、行项目
WHERE  n_nationkey  =  s_nationkey
    AND s_nationkey = c_nationkey
    AND c_custkey = o_custkey
    AND o_orderkey = l_orderkey;
```

QY - 与曾经订购过相同产品的同一国家的一对客户位于同一国家的供应商：

```
SELECT l1.l_linenumber, o1.o_orderkey, c1.c_custkey,
        l2.l_linenumber, o2.o_orderkey, s_suppkey,
        c2.c_custkey
FROM 行项目 l1、 订单 o1、客户 c1、
    行项目 l2，订单 o2，客户 c2，供应商 s WHERE
        l1.l_orderkey = o1.o_orderkey
    AND o1.o_custkey = c1.c_custkey
    AND l1.l_partkey = l2.l_partkey AND
    l2.l_orderkey = o 2 . o_orderkey AND
    o2.o_custkey = c2.c_custkey AND
    c1.c_nationkey = s.s_nationkey
    AND s.s_nationkey = c2.c_nationkey;
```

选择 *
FROM popuar-user A，twitter-user B、
twitter 用户 C、twitter 用户 D
WHERE A.dst = B.src
AND B.dst = C.src
AND C.dst = D.src
AND D.dst = A.src;

QF - 两名用户，一名是受欢迎用户，另一名是紧随其后的是一个热门用户，然后是第一个热门用户：

选择 *
FROM popular-user A, twitter-user B、
twitter-用户 C、流行用户 D
WHERE A.src = B.src
AND C.dst = A.src
AND C.src = D.src;

E 训练数据生成器和连接学习实验查询

第 6.6 节中使用的训练数据由修改后的

D 社交图谱查询

QT - twitter 粉丝图中的三角形：

```
SELECT * FROM
popuar-user A、twitter-user B、twitter-user C
WHERE  A.dst = B.src
    AND B.dst = C.src AND
    C.dst = A.src;
```

QS - twitter 粉丝图中的正方形：

TPC-H 数据生成器。在 TPC-H 基准的 lineitem 表中，有一个 return_flag 字段，表示下订单的客户是否退回了已交付的物品。在原始基准中，如果项目已交付，则该字段遵循均匀分布。我们修改了退货标志的分布如下，使其与客户、发货日期和商品价格相关。

首先，所有商品被退回的概率为 10%。然后，我们把整个发货日期分成 13 个 200 天的窗口（0-12），最后一个窗口短于 200 天（因为日期值的范围）。每位顾客以 1/2 的概率选择一组奇数装运日期窗口，以 1/2 的概率选择另一半窗口。如果商品在所选窗口内发货，则退货概率为 50%。这是为了模拟客户的退货率经常表现出很强的时间局部性。

最后，如果某件商品的折扣后价格在所有商品中大约属于前 1%（l_extendedprice * (1 - l_discount) / 1000 >=85），则该商品将有额外 30% 的机会被退回。这是为了模拟昂贵的订单会有更高退货率的情况。

提取特征以训练 SVM 模型的查询如下所示。为了预测运送给客户 c 的商品 ℓ_1 的退货标志，我们希望查看运送给同一客户 c 的另一个商品 ℓ_2，并将 ℓ_2 的退货标志作为一个特征。我们还包括两件商品（ℓ_1 和 ℓ_2）的发货日期差异，以及两件商品的数量和折扣后价格。此外，我们还将商品供应商的地区、供应商的账户余额、总价和订单的优先级作为特征。

请注意，机器学习算法无法知道哪些特征对有效预测非常重要。因此，它无法 "神奇地 "只选择那些会影响我们生成器中使用的 return_flag 值的特征（即发货日期和商品价格的差异）。相反，它必须利用我们的查询中显示的一系列看似有用的特征来训练其模型（例如，来自某些地区的客户有可能对其商品的价格产生影响）。

退货率也可能与订单数量有关。

```
选择
    l1.l_returnflag, n_regionkey, s_acctbal,
    l1.l_quantity, l1.l_extendedprice, l1.l_discount,
    l1.l_shipdate, o1.o_totalprice, o1.o_orderpriority,
    l2.l_quantity, l2.l_extendedprice, l2.l_discount,
    l2.l_returnflag, l2.l_shipdate
FROM 国家、供应商、行项目 l1、订单 o1、客户、订
    单 o2、行项目 l2
    WHEREs_nationkey =
    n_nationkey AND s_suppkey =
    l1.l_suppkey
    AND l1.l_orderkey = o1.o_orderkey AND
    o1.o_custkey = c_custkey
    AND c_custkey = o2.o_custkey
    AND o2.o_orderkey = l2.l_orderkey;
```

F 附加实验

表 4：随机抽样接受率比较

	精确重量	在线探索	扩展奥尔肯
Q3	1.00	1.00	0.0092
QX	1.00	1.00	0.0084
QY	0.071	0.071	0.00033
QT	0.15	0.15	0.00078
QS	0.13	0.0076	0.000043
QF	1.00	1.00	0.000031

表 4 显示了我们的抽样框架的三个实例对所有六个查询的接受率。如表所示，精确加权和在线探索的接受率最高，因为它们的上限相当严格。而扩展 Olken 的接受率通常要低 1 或 2 个量级，因为 AGM 约束和每个表中最大度数乘积的较小值所设定的上界比较宽松。