PrivLava：差分隐私下带外键的关系数据合成

Kuntai Cai
caikt@comp.nus.edu.sg 新加坡国立大学 新加坡，新加坡

Xiaokui Xiao
xkxiao@nus.edu.sg 新加坡国立大学 新加坡，新加坡

Graham Cormode
rmodeg.cormode@warwick.ac.uk
华威大学 考文垂，英国

摘要

在保护隐私的同时回答数据库查询，是近年来受到广泛关注的重要问题。解决这一问题的一种经典方法是使用合成数据。也就是说，我们将输入数据库 R 替换为合成数据库 R^*，该数据库保留了 R 的特征，并使用 R^* 回答查询。然而，现有的关系数据合成解决方案要么无法提供强有力的隐私保护，要么假设 R 只包含单一关系。此外，将现有的单一关系解决方案扩展到多个关系的情况颇具挑战，因为这些方案无法对外键所引发的复杂相关性进行建模。因此，如何在提供强隐私保证的同时进行多关系数据合成，仍是一个开放问题。

在本文中，我们通过提出 PrivLava 解决上述开放问题。PrivLava是首个在差分隐私下合成带外键的关系数据的解决方案；差分隐私是一种在学术界和工业界均被广泛采用的严格隐私框架。PrivLava 的核心思想是使用图模型对 R 中的数据分布进行建模，并加入潜变量 以捕捉由外键引起的关系间相关性。我们证明，PrivLava支持形成有向无环图的任意外键引用，并能够处理 R 包含公有关系和私有关系混合这一常见情况。在人口普查数据集和 TPC-H 基准测试上的大量实验表明，就在合成数据上处理的聚合查询的准确性而言，PrivLava 显著优于其竞争方法。

CCS 概念

• **安全与隐私 → 数据匿名化与清理**。

关键词

差分隐私, 数据合成

ACM参考文献格式：
Kuntai Cai、Xiaokui Xiao 和 Graham Cormode。2023年。PrivLava：在差分隐私下合成带外键的关系数据。发表于《会议论文集》（请从权利确认邮件中填写正确的会议名称（会议简称 *'XX*）。ACM，纽约州纽约市，18页。https://doi.org/XXXXXXX.XXXXXXX

允许为个人或课堂使用而免费制作本作品的部分或全部数字或纸质副本，前提是不得为营利或商业利益制作或分发副本，且副本须附有本声明及首页上的完整引用信息。本作品中第三方组件的版权必须得到尊重。其他用途请联系所有者/作者。会议简称 *'XX, 2023*年*6*月*3*日至*5*日，*XXXXXXXXX, XX* © 2023年版权归所有者/作者所有。ACM ISBN 978-x-xxxx-xxxx-x/YY/MM。
https://doi.org/XXXXXXX.XXXXXXX

1 引言

在保护个人隐私的同时回答数据库查询，是一个重要问题，近年来吸引了大量研究关注[2, 4–6, 8, 12, 15, 22, 25, 27, 28, 33, 35, 37, 38, 40, 41, 43, 45, 46] 。现有解决方案大多采用输出扰动方法 [35], ，通过修改数据库引擎，为每个查询答案添加噪声以实现隐私保护。所需噪声量根据查询答案可能泄露的敏感信息量进行校准，通常依据差分隐私（DP）等形式化隐私框架 [16] 或其推广形式。

尽管输出扰动广受欢迎，但它存在一个重大缺陷，限制了其在实践中的应用：数据库只能以较高准确度回答有限数量的查询。这一观察（由 Dinur 和 Nissim [14] 形式化）源于数据库返回的每个噪声答案都不可避免地会泄露有关底层数据的一些信息。处理的查询越多，泄露的信息总量就会单调增加，最终达到某个阈值；此时，回答任何新查询都会导致整体隐私风险过高，数据库也就无法再准确回答新的查询。因此，现有研究可能主要关注提供单次查询隐私保证，而不涉及为所有查询的组合提供总体界限 [2, 12, 35, 40, 45, 46]。然而，要获得有意义的保证，我们应当寻求一个在数据库整个生命周期内均有效的强隐私界限。

一种自然的缓解方法是使用合成数据 构建隐私保护数据库，使其能够被无限次查询，同时不会损害相对于原始真实数据的隐私。具体而言，给定一个包含敏感信息的数据库 R，我们生成一个模仿 R 但隐藏其中私人数据的合成版本 R^*；此后，我们可以使用 R^* （而不是 R）来回答查询，无须限制用户可以发出的查询数量。其原理是，由于所有查询都是在合成数据库 R^*上处理的，查询答案只会揭示 R^* 中的信息，而非 R 中的信息；因此，只要 R^* 本身能够保护隐私，我们就无需限制使用 R^*回答查询的次数。此外，与输出扰动不同，使用合成数据 R^*既无需更改数据库引擎，也不会产生额外的查询处理开销。

然而，为了确保 R^* 中强有力的隐私保护，R^* 应使用能够提供严格隐私保证的算法（例如差分隐私）生成。关于这一点，已有多种用于合成关系数据的差分隐私方法 [7, 9, 13, 18, 21, 23, 29, 34, 36, 47, 49, 50] 。遗憾的是，这些方法都只适用于一种受限情形：输入

数据库 R 仅包含 单一关系。具体而言，这些方法假设数据库元组是来自某个未知多维分布的独立样本；它们根据输入表中观测到的元组推断该分布，然后从推断出的分布中采样以生成合成数据。将这些方法扩展到一般情形并不容易：在这种情形下，R 由多个带有外键约束的关系组成，因为外键会导致数据库不同部分中的元组之间存在复杂的相关性，而单关系合成方法忽略了这类相关性。因此，在差分隐私下利用外键生成合成数据仍是一个未解决的问题。

我们的贡献。本文介绍了 PrivLava1，这是首个用于在外键约束下合成关系数据的差分隐私解决方案。PrivLava 的核心思想是使用图模型 [44], 对输入数据库中的数据分布建模，并纳入潜变量以捕获由外键导致的关系间相关性。例如，考虑一个包含两个关系 R_I和 R_H的人口普查数据库，其中 R_I（分别是 R_H）中的每个元组都表示一个个人（分别是一个家庭），且 R_I包含一个引用 R_H的外键，用于将每个个人关联到其所属的家庭。为对 R_I和 R_H建模，我们假设：(i) 每个家庭都关联一个潜变量 Z，用于表示其类型（例如，是否为核心家庭）；(ii) Z 还决定家庭规模、属性和成员的分布。基于这一假设，我们构建了一个含有潜变量的图模型，根据成员组成推断每个家庭的类型 Z，进而估计家庭类型的总体分布 $p(Z)$，以及给定家庭类型时家庭规模、属性和成员的条件分布 Z。得到这些分布的估计值后，我们可以按如下方式生成合成家庭：首先从 $p(Z)$ 中抽取一个家庭类型 Z，然后依据 Z 抽取属性，生成一个合成家庭元组；之后，在给定 Z的条件下抽取家庭规模 s，再依据 Z生成 s个个人元组。换言之，我们利用 Z 衔接合成家庭和个人的生成过程，从而保留由 R_I和 R_H之间的外键所形成的家庭组成分布。

一般而言，PrivLava 使用含有潜变量的图模型，分别对输入数据库 R 中的每个外键进行建模，并在模型训练算法中注入经过精细校准的噪声，以确保差分隐私。其建模方法具有足够的通用性，可支持构成有向无环图的任意外键引用。此外，它还能够处理 R 同时包含公有关系和私有关系这一常见情况。

我们使用来自综合公共使用微观数据系列 [10, 42] 的两个人口普查数据集以及 TPC-H 基准测试 [1]，对 PrivLava 与最先进的差分隐私数据合成方法 [9, 34, 49], 进行了实验评估。实验结果表明，在对这些方法生成的合成数据执行聚合查询时，PrivLava 的准确性显著优于竞争方法。此外，PrivLava 能够在标准机器上于数小时内处理大型基准数据集；对于只需执行一次以发布数据集的流程而言，这一耗时是合理的。

2 问题定义

设 R 是一个包含关系 R_0、R_1、R_2、..... 的数据库。对于任意R_i, $R_j \in$R，如果 R_i具有引用 R_j主键的外键，我们就称 R_i引用 R_j。相应地，如果元组 t_i的外键等于元组 t_j的主键，我们就称元组 $t_i \in R_i$引用元组 $t_j \in R_j$（记作 $t_i \to t_j$）。此外，如果元组 $t \to t_j$ 或元组 t引用了另一个依赖于 t_j的元组，那么元组 t依赖于 t_j（记作 $t \rightsquigarrow t_j$）。我们假设 R 中的外键引用构成一个有向无环图，即，$t \rightsquigarrow t$永远不会出现。

不失一般性，假设 R_0 是一个包含敏感信息的私有关系。令 t_0 为 R_0中的一个元组，并令R′ 为从 R 中移除 t_0 以及依赖于 t_0的所有其他元组后得到的数据库。我们将 R 和 R′ 称为相邻数据库。基于相邻数据库的这一概念，我们对差分隐私作出如下形式化定义：

定义 *2.1*（差分隐私（差分隐私）*[16]*）。设 F为一个以数据库为输入的算法。当且仅当对于任意两个相邻数据库 R 和 R′ 以及 F的任意可能输出集合 O，F 满足 (ϵ, δ)-差分隐私（差分隐私），

$$\Pr[F(\mathcal{R}) \in O] \leq e^{\epsilon} \cdot \Pr[F(\mathcal{R}') \in O] + \delta.$$

直观而言，上述差分隐私的概念旨在确保：当对手观察到 F 的输出时，即使考虑外键所引起的元组间依赖关系，他也无法推断出关系 R_0 中是否存在任何单个元组 t。为方便起见，我们将 R_0 称为主要私有关系。对于任何其他关系 R_i，如果 R_i 中的元组依赖于 R_0 中的元组，我们将 R_i 称为次要私有关系；否则，R_i 是公开关系，即可以直接发布而无需付出任何隐私成本。

继之前的工作 [15, 25, 43], 之后，我们假设数据库中的每个外键都具有有界的倍数。也就是说，每当 R_i具有引用 R_j的外键时，R_j中的每个元组最多会被 R_i中的τ_{ij}个元组引用，其中 τ_{ij}是预定义的常数。该假设旨在确保主要隐私关系 R_0 中的每个元组所依赖的其他元组数量是有界的；否则，在 R_0 中插入或删除一个元组 t 可能会导致数据库发生无界变化，这使得实施差分隐私变得不可行，因为对手可以轻易推断 t 是否存在于 R_0中。出于这一原因，所有先前的工作都假设每个外键具有有界的倍数。在实践中，我们可以通过截断数据来实施这一假设，即，对于每个 R_j，我们移除在任何 R_i [25, 43]中被超过τ_{ij} 个元组引用的元组。当每个阈值τ_{ij} 足够大时，截断操作只会移除数据集中的离群值，不会显著影响数据的保真度。

我们的目标是开发一种合成数据生成算法，该算法 (i) 将数据库 R 作为输入，(ii) 输出一个与 R 具有相同模式的数据库 R*，并且 (iii) 满足 (ϵ, δ)-差分隐私。

3 解决方案概述

简而言之，PrivLava 使用图模型分别对输入数据库 R 中的每个外键进行建模，并将这些模型组合起来，生成能够保留相关性的合成关系。

[1]隐私保护的带有潜变量的图模型。

表 1：个人表 R_1。

ID	Age	. . .	H-ID
1	28	. . .	1
2	25	. . .	1
3	27	. . .	2
4	29	. . .	2
5	35	. . .	3
6	36	. . .	3
7	5	. . .	3
. . .	. . .	. . .	. . .

表2：家庭表 R_0。

H-ID	. . .
1	. . .
2	. . .
3	. . .
. . .	. . .

表 3：R_1 带有潜在 Z。

ID	Age	。。。	H-ID	Z
1	28	. . .	1	z_1
2	25	. . .	1	
3	27	. . .	2	z_1
4	29	. . .	2	
5	41	. . .	3	
6	39	. . .	3	z_2
7	10	. . .	3	
. . .	. . .	. . .	. . .	. . .

表 4：R_0 包含潜变量 Z。

H-ID	. . .	Z
1	. . .	z_1
2	. . .	z_1
3	. . .	z_2
. . .	. . .	. . .

由外键约束引起。接下来，我们概述 PrivLava 采用的建模方法，首先从 R 仅包含两个私有关系和一个外键的基本情况开始。

3.1 处理单个外键

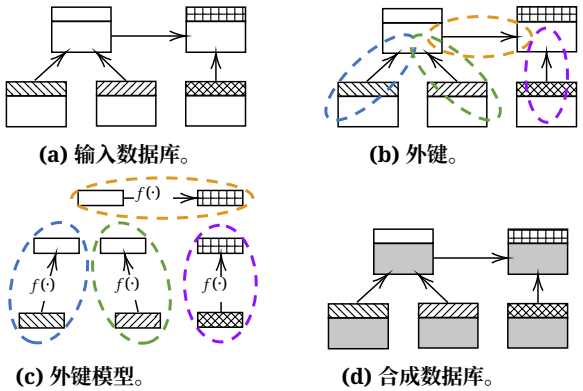
假设 R 由一个主私有关系 R_0 和一个次级私有关系 R_1组成。如果 R_1 中的两个元组指向 R_0中的同一个元组，我们就称它们属于同一个组。例如，表1和表2展示了一个示例，其中 R_0 和 R_1 分别包含有关家庭和个人的信息。此时， R_1 中的每个组都是同一家庭中的个人集合。

直观而言，如果我们要生成 R_0 和 R_1的合成版本，就需要对三类信息进行建模：

(1) 同一关系中各属性之间的属性间相关性（例如，个人的年龄与其收入之间的相关性）；(2) R_1 中同一组内各元组之间的组内相关性（例如，哪些类型的个人倾向于共同居住在同一家庭中）；(3) R_1 和 R_0 中的元组之间的关系间相关性（例如，哪些类型的个人可能居住在郊区家庭中）。

关于单关系合成的现有研究 [7, 9, 29, 34, 49] 已广泛探讨属性间相关性的建模。典型方法是：(i) 找出彼此高度相关的属性集合；(ii) 在差分隐私下测量每个属性组的联合分布；然后 (iii) 利用得到的联合分布生成合成元组。然而，目前尚无研究探讨数据合成中组内相关性和关系间相关性的保留。

为了解决上述问题，我们的思路是识别 R_1中元组组构成的模式，并将其作为代理，以捕捉组内相关性和关系间相关性。例如，在表 1 中，元组组的构成可能呈现某些



(a) 输入数据库。 (b) 外键。 (c) 外键模型。 (d) 合成数据库。

图1：解决方案概述。

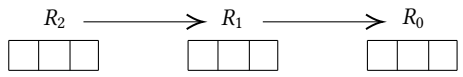
模式，例如，没有孩子的年轻夫妇，或有孩子的中年父母。我们假设，每个组的模式由 R_1中的一个潜在（即隐藏）属性 Z 决定，如表 3 所示。请注意，表 3 中的第一个和第二个元组组都具有 $Z = z_1$，这表明它们的组构成具有相同的模式。同时，第三个元组组具有 $Z = z_2$，这意味着其组构成与前两个组的组构成有显著差异。

假设我们能够推断出 R_1 中每个元组组的潜在属性 Z 的值。在这种情况下，我们可以使用 R_1 来估计 R_1 中 P_1 of Z 与其他属性的联合分布。此外，我们可以将每个组的潜在属性与 R_0 中该组所指向的元组关联起来，从而得到一个添加了 Z 的 R_0 版本，如表 4 所示。这使我们能够估计 R_0 中 Z 与各属性的联合分布。利用上述联合分布，我们可以按如下方式生成合成元组：首先，从增强后的 R_0 中 Z 的分布采样一个值 z，然后基于 P_0 并以 $Z = z$ 为条件，为 R_0 生成一个合成元组 t^*；之后，根据 P_1 和 $Z = z$，合成 R_1 中与 t^* 对应的元组组。直观地说，如果 Z 能准确刻画元组组的模式，那么我们生成的合成元组就能够保留输入数据中的组内相关性和关系间相关性。

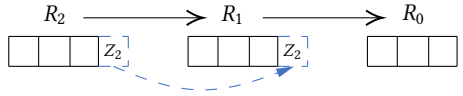
我们如何推断 R_1中每个元组组的潜在属性 Z？我们采用 期望最大化（*EM*） [44], 这一种经典方法，用于估计具有潜变量的统计模型的参数。其高层次的思想是，EM使我们能够根据相似性对 R_1 中的元组组进行聚类，然后我们可以为每个聚类分配一个 Z 值，以表明同一聚类中的元组组遵循相似的模式。在第 4节中，我们将介绍用于建模一个外键的基于 EM 的算法的详细信息。

3.2 处理多个外键

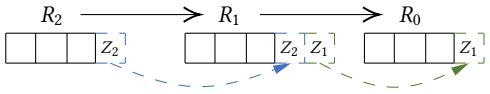
对于输入数据库 R包含多个外键引用的一般情况，PrivLava 按照图1 所示的步骤处理R。它首先识别出R 中与私有关系相关的外键，然后依次采用类似于第 3.1节所述算法的方法对每个外键进行建模。之后，它使用这些模型生成合成数据，以保留外键所引入的相关性。



(a) 三个私有关系 R_0，R_1，R_2。



(b) 对从 R_2 到 R_1的外键进行建模。



(c) 对从 R_1 到 R_0的外键进行建模。

图2：多个外键情形示例。

例如，图2a 显示了一个主私有关系 R_0 和两个次级私有关系 R_1 和 R_2，其中 R_2 引用 R_1，而 R_1又引用 R_0。给定 R_0，R_1，R_2，PrivLava 首先处理从 R_2 到 R_1的外键，这会使 R_1 和 R_2 都增加一个潜在属性 Z_2。之后，PrivLava 处理从 R_1 到 R_0的外键，在刻画 R_1 中的元组组时考虑 Z_2（即，它将 Z_2 视为 R_1的一个属性）。这使得 R_1 和 R_0 增加另一个潜在属性 Z_1。

构建外键模型后，PrivLava按照外键建模的逆序使用这些模型生成合成数据。具体而言，给定图2c中的增强关系，PrivLava 首先采样 Z_1，然后基于此在 R_0 中生成一个合成元组，以及增强后的 R_1中的相应元组 S组。之后，对于 S中的每个合成元组，PrivLava检查其在 Z_2上的值，并基于此在 R_2中构建相应的元组组。该过程不断重复，直到我们在 R_0、R_1和 R_2中获得足够数量的合成元组。

一般而言，给定一组构成有向无环图的外键引用，PrivLava 会根据图边的拓扑序对外键进行建模，然后按照逆序合成数据。当私有关系引用公共关系或多个其他关系时，需要进行特殊处理，因为这类情况会使合成元组生成过程变得复杂。在第5节中，我们将详细阐述处理多个外键的解决方案。

4 单外键解决方案

本节介绍当输入数据库 R 仅包含一个主私有关系 R_0 和一个次级私有关系 R_1时基本情形的解决方案，如表1和表2所示。第4.1至4.3节详细阐述了我们对 R_1 的建模，随后第 4.4 节解释了我们如何对 R_0 进行建模。在此之后，第4.5节讨论了基于我们对 R_0 和 R_1 的建模生成合成数据。

4.1 次级私有关系建模 R_1

我们首先处理次级私有关系 R_1，并说明如何通过图模型对附加了潜在属性的表进行建模。不失一般性，假设 R_1 包含一个

表 5：私有关系 R_1。

	A_{ID}	A_1	A_2	A_3	A_{FK}
t_1	1	0	0	1	1
t_2	2	1	0	0	1
t_3	3	0	1	1	2
t_4	4	1	0	1	2
t_5	5	1	1	0	2

表 6：边缘分布 $\{A_1, A_2\}$。

c_{00}	1	（对于 $A_1 = 0$，$A_2 = 0$）
c_{01}	1	（对于 $A_1 = 0$，$A_2 = 1$）
c_{10}	2	（对于 $A_1 = 1$，$A_2 = 0$）
c_{11}	1	（对于 $A_1 = 1$，$A_2 = 1$）

主键属性 AID和一个引用 R_0的外键属性 AFK，以及 d其他属性 $A_1, A_2, \ldots, A_d$。如第 3.1 节所述，我们假设 R_1 有一个潜在（不可观测）属性 Z，用于表征由 AFK诱导的元组组，如表 3 所示。我们基于其边缘分布对 R_1 建模；这些边缘分布是在 R_1的属性子集上构建的直方图。具体而言，令 $\mathrm{A}_i = \{A_{i_1}, A_{i_2}, \ldots, A_{i_k}\}$为$\{A_1, A_2, \ldots, A_d\}$的一个 k大小的子集，$\text{span}(\mathrm{A}_i) = A_{i_1} \times A_{i_2} \times \cdots \times A_{i_k}$为由 $A_{i_1}, A_{i_2}, \ldots A_{i_k}$张成的 k维空间。假设我们将 R_1 中的元组投影到 $\text{span}(\mathrm{A}_i)$上。那么，对于$\text{span}(\mathrm{A}_i)$中的每个点（$a_{i_1}, a_{i_2}, \ldots, a_{i_k}$），我们都可以统计 R_1中满足对所有$j = 1, 2, \ldots, k$，t具有 $t[A_{i_j}] = a_{i_j}$的元组数量，其中 $t[A_{i_j}]$表示 A_{i_j}上 t的取值。我们将这组计数称为 R_1在A_i上的边缘分布。例如，考虑表 5 中的关系 R_1，其中属性A_1和 A_2的取值域均为$\{0, 1\}$。那么，R_1在$\{A_1, A_2\}$上的边缘分布由四个计数 $c_{00}, c_{01}, c_{10}, c_{11}$组成，其中$c_{xy}$表示 R_1 中同时满足 $A_1 = x$和 $A_2 = y$的元组数量，如表 6 所示。

R_1 在 A_i 上的边缘分布是 A_i中属性联合分布的一种表示。我们在解决方案中利用此类表示，并在概念上将其扩展为考虑潜在属性 Z。具体而言，对于 R_1 中的任意元组 t 以及 Z的取值域中的任意元素 z，令 $p(z \mid t)$表示 t的元组组具有 $Z = z$ 的概率。（注意，由于 Z是潜在的，因此无法直接从 R_1观测到该概率。）我们考虑由 Z和A_i中的属性张成的（$k + 1$）维空间 $\text{span}(\mathrm{A}_i \cup \{Z\})$。对于 $\text{span}(\mathrm{A}_i \cup \{Z\})$中的每个点（$a_{i_1}, \ldots, a_{i_k}, z$），我们将所有满足 $t \in R_1$ 且对于j = 1、2、…、k有 $t[A_{i_j}] = a_{i_j}$的 $p(z \mid t)$相加。换言之，我们计算满足 $t[A_{i_j}] = a_{i_j}$ 且属于具有 $Z = z$的元组组的元组 t的期望数量。这会得到 $\text{span}(\mathrm{A}_i \cup \{Z\})|$中的一组 |期望计数，我们将其定义为 R_1 在 $\mathrm{A}_i \cup \{Z\}$上的边缘分布。我们将此类边缘分布称为潜在边缘分布，并将定义在某个 A_i 上的边缘分布称为观测边缘分布。

假设给定 R_1 的一组边缘分布 M。我们使用一个指数族图模型 [44]来建模 R_1。该模型由向量 θ 参数化，其中每个坐标都对应于 M 中某个边缘分布的一个计数（或期望计数） c（也就是说，θ 的大小等于 M 中各边缘分布的计数总数）。为方便起见，我们用 $\theta[c]$ 表示 θ 中与 c 对应的坐标。设 t 为一个元组，z 为 Z 的取值域中的一个值，而 $p(t, z)$ 表示：如果从 R_1 中抽取一个元组，它具有与 t 相同的属性值且属于 $Z = z$ 的元组组的概率。我们的图模型假设，可以根据与 (t, z)相关联的边缘计数（或期望计数）c，从 θ 推导出 $p(t, z)$，记作 $c \leftrightarrow (t, z)$：

$$p(t, z) \propto \prod_{c \in M \in \mathcal{M} \wedge c \leftrightarrow (t,z)} \exp(\theta[c]). \tag{1}$$

具体而言，在属性集 A_i 上的边缘分布 M 中，若计数 c 对应于点 $(a_{i_1}, \ldots\ldots, a_{i_k}) \in \text{span}(A_i)$，且 $a_{i_j} = t[A_{i_j}]$ 对于 $j = 1, \ldots\ldots, k$ 成立，则称该计数与 (t, z) 相关联。例如，在表 6 的边缘分布中，计数 c_{10} 与表 5 中的 t_2 和 t_4 相关联，因为 c_{10} 对应于点 $(A_1 = 1, A_2 = 0)$，而 t_2 和 t_4 都具有 $A_1 = 1$ 和 $A_2 = 0$。类似地，对于属性集 $A_i \cup \{Z\}$ 上的边缘分布，如果期望计数对应于点 $(t[A_{i_1}], \ldots\ldots, t[A_{i_k}], z) \in \text{span}(A_i \cup \{Z\})$，则称该期望计数与 (t, z) 相关联。

除了公式 1 外，我们还假设 R_1 中元组组的潜在属性 z 遵循分布 p_Z，并且元组组的大小遵循在 z 条件下的分布 $p_{\text{size}}(s \mid z)$。我们认为，$R_1$ 中的每个元组组 G 都具有以下生成过程：

（1）以概率 $p_Z(z)$ 对 Z 采样一个值 z；（2）以概率 $p_{\text{size}}(s \mid z)$ 采样一个组大小 s；（3）采样 s 个元组 $t_1, t_2, \ldots\ldots, t_s$，每个元组的概率为 $p(t_j \mid z) = p_{(t_j, z)} / \sum_t p_{(t, z)}$，$j = 1, 2, \ldots\ldots, s$。

基于这一生成过程，对于 R_1 中的每个元组组 G，我们将其似然表示为

$$p(G) = \sum_z \left(p_Z(z) \cdot p_{\text{size}}(|G| \mid z) \cdot \prod_{t \in G} p(t \mid z) \right). \tag{2}$$

我们的目标是估计 θ、p_Z 和 p_{size}，以最大化在 R_1 中观察到的所有元组组 G 的总体对数似然，即 即，$\log p(G)$（注意，这等价于最大化 $\prod_G p(G)$）。在第4.2节中，我们介绍一种算法，用于在给定 R_1 的边缘分布集合 M 时估计 θ、p_Z 和 p大小。之后，在第 4.3 节中，我们解释如何在我们的解决方案中确定 M。

4.2 估计参数 θ、p_Z 和 p_{size}

接下来，我们假设给定了一个边缘分布集合M，并描述一种用于学习模型参数的算法。假设 M同时包含观测边缘分布和潜在边缘分布。我们假设观测边缘分布中的计数已经根据 R_1 计算得出，并注入高斯噪声 [3] 以实现差分隐私。（我们将在第4.3节中解释噪声注入的详细信息。）与此同时，对于潜在边缘分布，我们将其中每个计数视为一个变量，因为 R_1 中的 Z 值事先是未知的。我们使用 M 来估计 θ、p_Z和p大小，所采用的算法基于期望最大化（EM）方法 [44]，，如算法1所示。

算法1首先从输入中获取初始模型参数 θ，p_Z，$p$size。（我们在第4.3节中解释这些参数的初始化。）算法1的下一部分由 T迭代组成（第1-23行），每次迭代都遵循EM方法的典型框架，并由以下两个步骤组成：

(1) E步(第2-7行)：基于当前模型参数，估计每个元组组 R_1 的潜在属性；(2)M步(第8-23行)：更新模型参数，以优化所有元组组 G的整体对数似然，即即，$\sum_G \log p(G)$。

换言之，该算法迭代地细化对每个元组组的潜在属性的估计，并利用这些属性来提高模型参数的准确性。

具体而言，在每次迭代中，算法1首先检查 R_1 中的元组组（第2-7行）。对于每个元组组 G，它计算该

算法1： 参数估计

输入： 关系 R_1、初始模型参数 θ、p_Z，以及 p大小，迭代次数 T，边缘集合 M，噪声尺度 σ_z, σ大小，以及 σ_ℓ

输出： 更新后的模型参数 θ, p_Z, p大小，潜在属性 z_G，针对每个元组组 G

1 **对于** $j = 1$ **到** T **执行**
2 **对于** 每个元组组 G 在 R_1 **执行**
3 **对于** 每个元组 t 在 G 以及每个 z **执行**
4 计算 $p(t \mid z) = \frac{p(t, z)}{\sum_{t'} p(t', z)}$ 基于 θ;
5 **对**每个 z **执行**
6 计算 $p(z \mid G) = \frac{p(G, z) \sum_{z'} p(G, z')}{\text{，其中}}$，$p(G, z) = p_Z(z) \cdot p_{\text{大小}}(|G| \mid z) \cdot \prod_{t \in G} p(t \mid z)$;
7 令 z_G 为使 $p(z \mid G)$ 最大化的 z;
8 **对于** 每个 z，**执行**
9 计算 R_1 中元组组的数量 $\text{cnt}(z)$ G 满足 $z_G = z$;
10 令$\text{cnt}^*(z) = \max\{0, \text{cnt}(z) + \text{N}(0, \sigma_z^2)\}$;
11 **对** 每个 z **执行**
12 $p_Z(z) = \frac{\text{cnt}^*(z)}{\sum_{z'} \text{cnt}^*(z')}$;
13 **对**每个 z **执行**
14 **对于** 每个组大小 s **执行**
15 统计元组组 G 的数量 $\text{cnt}(s, z)$ 与 $z_G = z$ 和 $|G| = s$;
16 令 $\text{cnt}^*(s, z) = \max\{0, \text{cnt}(s, z) + \text{N}(0, \sigma^2_{\text{size}})\}$;
17 **对于** 每个组大小 s **执行**
18 $p_{\text{大小}}(s \mid z) = \frac{\text{cnt}^*(s,z)}{\sum_{s'} \text{cnt}^*(s',z)}$
19 **对于** 每个潜在边缘分布 $M \in \mathcal{M}$ **执行**
20 根据 z_G计算 M 中的期望计数 每个组 G的;
21 **对于** 每个期望计数 c 在 M **执行**
22 $c = c + \text{N}(0, \sigma_\ell^2)$;
23 根据噪声计数和期望计数更新 θ 当前位于 M 中的边缘分布;
24 **返回** θ，p大小，以及对于每个元组组 G;

条件概率 $p(t \mid z) = p_{(t,z)} / \sum_{t'} p_{(t',z)}$对于每个 $t \in G$和 z基于当前的 θ。然后，基于 $p(t \mid z)$ 和当前的 p_Z和p大小，它推导出条件概率

$$p(z \mid G) = \frac{p(G, z)}{\sum'_z p(G, z')}, \tag{3}$$

其中

$$p(G, z) = p_Z(z) \cdot p_{\text{size}}(|G| \mid z) \cdot \prod_{t \in G} p(t \mid z). \tag{4}$$

基于 $p(z \mid G)$，它识别出对于 G 最可能的潜在值 z_G，并将其作为 G 的潜在属性的当前估计值。

之后，对于每个 z，算法会统计其潜在属性被估计为 z的元组组数量 $\text{cnt}(z)$，并通过注入高斯

噪声 $N(0, \sigma_z^2)$ 得到该数量的含噪版本 cnt $p_Z(Z)$（第 8-10 行）。然后，根据 cnt*(z) 更新估计分布 $p_Z$of Z（第 11-12 行）。类似地，对于每个 z 和每种组大小 s，算法会计算含噪版本 cnt*(s, z)，表示在 R_1 中具有 $z_G = z$ 和 $|G| = s$ 的元组组 G 的数量（第 13-16 行）。随后，算法使用 cnt*(s, z) 更新给定 z 时组大小 s 的估计分布 $p$size$(s \mid z)$。

此外，对于每个潜在边缘分布 $M \in$ M，算法根据每个元组组的估计潜在属性，计算 M 中的期望计数，然后向其中注入高斯噪声 $N(0, \sigma_\ell^2)$。换言之，它将 M 中的每个潜在边缘分布实例化为带噪声的版本。随后，它将所有潜在边缘分布视为观测边缘分布（因为它们已经被实例化），并更新图模型参数 θ，使其拟合 M 中所有边缘分布的带噪声计数（第 23 行）。从观测边缘分布推断 θ 已在图模型文献中得到充分研究 [44]；对于算法1的第 23 行，我们采用 [34], 中的梯度下降法，因为该方法针对边缘分布计数带噪声的情况进行了优化。这里的一个问题是，[34] 中的方法针对的目标函数不同于我们的目标函数，因此无法立即确定该方法是否适用于我们的 EM 框架；然而，我们在附录 A 中证明，该方法返回的解在我们的上下文中仍然是正确的，即，它优化了所有元组组的总体对数似然。

4.3 选择边缘集合$\mathcal{M}$

现在我们去掉边缘集合 M 已给定的假设，并介绍一种选择 M 中观测边缘分布和潜在边缘分布的方法。直观来说，M 中的观测边缘分布和潜在边缘分布用途不同：观测边缘分布有助于我们捕捉 R_1 中的属性间相关性，而潜在边缘分布则使我们能够对 R_1 中的组内相关性进行建模。与此相关，现有的单表合成 [7, 9, 49]研究提出了多种差分隐私方案，用于选择观测边缘分布以对属性间相关性进行建模。然而，在差分隐私下，潜在边缘分布的选择尚未得到研究。

潜在边缘选择中的主要挑战是以下困境。为了根据其有用性选择潜在边缘分布，我们需要检查潜在属性 Z 与每个边缘分布所涉及的其他属性的联合分布；这要求我们估计每个元组组的潜在属性。然而，要使用算法1估计每个元组的潜在属性，我们需要将M 作为算法的输入，这要求我们已经决定了 M 中的潜在边缘分布。

为解决上述困境，我们采用如下迭代方法。首先，我们选择一组初始潜在边缘分布，并将其与若干观测边缘分布一同插入 M。之后，我们调用算法1，基于当前的 M 估计模型参数。利用所得参数，我们使用模型估计每个元组组的潜在属性。然后，我们利用估计出的潜在属性，评估当前不在 M 中的潜在边缘分布候选集 C 的有用性，并据此从 C 中选择最有潜力的潜在边缘分布，将其插入 M。重复这一过程，直到 M 中包含足够多的潜在边缘分布。

算法2展示了我们用于确定 M 的方法的伪代码。它首先调用现有的差分隐私边际选择算法

算法 2： 边际选择和模型构建
对于 R_1

输入： 关系 R_1，候选边缘分布数量n_C，边缘分布数量增量$n$inc，迭代次数 T_C和 T，噪声尺度 σ_z、 σ大小、 σ_ℓ，以及 σ_M

输出： 边缘集合 M、模型参数 θ、p大小，以及每个元组组的潜在属性 z_G G

1 调用差分隐私边际选择算法，在 [9] 上对 R_1 进行处理，以获得一个注入高斯噪声的观测边际集合 Mobs；已注入噪声；
2 令 M = Mobs；
3 **对于** 每个属性 $A \in \{A_1, A_2, \ldots, A_d\}$ **执行**
4 设 M为 R_1 在 $\{A, Z\}$上的潜在边缘分布；
5 将 M 插入 M；
6 初始化模型参数 θ， p_Z， p大小；
7 θ, p大小 $z_G \leftarrow$ 算法1 $(R_1, \theta, p_Z, p$大小T, M σ_z, σ大小 $\sigma_\ell)$；
8 **对于** $t = 1$ **到** T_C **执行**
9 抽取满足以下条件的潜在边缘分布集合 C：n_C 满足λ-有用性且不在 M 中；
10 令 C′ 为与 C 对应的观测边缘集合；
11 **对于**每个观测边缘分布 $M' \in C'$，**执行**
12 生成一个估计版本 M' 的估计版本 M'，方法如下：将每个计数替换为推导出的估计值使用当前 θ；令 $\widetilde{\text{err}}(M') = \|M' - \widehat{M'}\|_1 + N(0, \sigma^2\text{err})$；
13
14 找出 C′ 中使$n$inc 边缘分布 M' 最大化的元素 $\widetilde{\text{err}}(M')$，并插入其对应的潜在值将边缘分布加入 M；
15 通过插入一个新坐标 $\theta[c] = 0$ 来更新 θ娱乐主管在每个新插入的潜在边缘分布中的每个计数 c 到 M 中；
16 θ, p大小 $z_G \leftarrow$ 算法1 $(R_1, \theta,$ M p_Z, p大小 $1, \sigma_z, \sigma$大小 $\sigma_\ell)$；
17 **返回** M，θ, p大小，以及每个元组组的 z_G G；

我们开展工作 [9]，以确定一组能够保留属性间相关性 R_1 且注入了高斯噪声的观测边缘分布（第 1 行）。随后，我们将该边缘集合用作初始的 M（第 2 行）。之后，对于每个属性 $A \in \{A_1, \ldots, A_d\}$，算法会将 R_1 在$\{A, Z\}$上的一个潜在边缘分布插入 M（第 3–4 行）。其理由是，这些潜在边缘分布代表了潜在属性 Z 与其他属性之间最基本的联合分布类型，因此，它们是一个很好的潜在边缘分布起始集合。

然后，算法开始初始化模型参数θ、 p_Z和$p$size（第 6 行）：将 θ 设为零向量，将 p_Z 设为在 Z的预定义取值域上均匀分布，并将$p$size 设为在{1、2、.....、 τ}上均匀分布，其中 τ是 R_1指向 R_0的外键的最大多重度（见第 2 节）。随后，算法调用算法 1，根据当前边缘集合 M 学习 θ、 p_Z和$p$size（第 7 行）。算法的后续部分包含 T_C 次迭代（第 8–16 行），每次迭代都会识别另外$n$inc 个潜在边缘分布，并将其加入 M。具体而言，在每次迭代中，算法都会从不属于 M、但满足λ-有用性 [49]性质的潜在边缘分布中抽取一个包含n_C 个潜在边缘分布的集合 M：

$$\frac{\tilde{n}}{|M|} \geq \lambda \cdot \sqrt{\frac{2}{\pi}} \cdot \sigma_\ell, \tag{5}$$

其中，$\tilde{n}$ 是第 1 行的选择算法得到的 R_1 中元组数量的带噪估计，$|M|$ 是 M中的计数数量，$\lambda > 1$是预先设定的常数，而 $\sqrt{\frac{2}{\pi}}\sigma_\ell$是若 M 被选入 M，算法 1 将注入其中的高斯噪声的绝对值期望（见第 16 行）。直观来说，λ-有用性要求 M 中的平均计数大于注入 M的预期噪声量，从而确保注入噪声后 M 中的信噪比仍然合理。我们设定 $\lambda = 20$。

接下来，算法 2 构造观测边缘分布集合 C′，其对应于 C 中的潜在边缘分布（第 10 行）。具体而言，若观测边缘分布 M' 定义在属性集 $\mathrm{A} \subseteq \{A_1, A_2, \ldots, A_d\}$ 上，而 M 定义在 $\mathrm{A} \cup \{Z\}$ 上，则称观测边缘分布 M' 对应于潜在边缘分布 M。随后，对于 C′ 中的每个观测边缘分布 M'，算法 2 评估当前模型对 M' 中计数的近似准确程度。具体来说，算法首先利用当前模型参数 θ 估计 M' 中的计数（基于公式 1），从而得到 M' 的估计版本 $\widetilde{M'}$（第 12 行）。之后，算法计算 M' 与 $\widetilde{M'}$ 之间的 L_1 距离，并向其中注入高斯噪声 $\mathrm{N}(0, \sigma^2 \mathrm{err})$（第 13 行）。令 $\widetilde{\mathrm{err}}(M')$ 表示所得的带噪 L_1 距离。直观而言，如果 $\widetilde{\mathrm{err}}(M')$ 较大，则说明我们的模型未能准确捕获 M' 中的信息；在这种情况下，将与 M' 对应的边缘分布添加到 M 中有助于改进模型。因此，算法 2 在 C′ 中识别具有最大 $\widetilde{\mathrm{err}}(M')$ 的 n_{inc} 个边缘分布 M'，并将其对应的潜在边缘分布插入 M 中（第 14 行）。随后，算法通过为每个新添加边缘分布中的每个单元格创建一个新坐标 $\theta[c] = 0$ 来更新模型参数 θ（第 15 行）。然后，算法调用算法1，基于当前的 M 更新 θ（第 16 行）。请注意，算法1中的迭代次数设置为 $T = 1$，因为此处模型参数的学习基于先前获得的 θ、p_Z、p 大小（而不是从头开始），因此一次迭代即可对模型进行微调。

4.4 建模主私有关系 R_0

为了完成建模步骤，我们使用现有技术对主要私有关系 R_0进行建模。回顾一下，使用算法1和算法2，我们可以获得次级私有关系 R_1的差分隐私图模型，以及每个元组组 G的推断潜在属性 z_G（属于 R_1）。然后，我们可以按如下方式向主要私有关系 R_0中插入一个新属性 Z：如果 R_1中的元组组 G指向 R_0中的元组 t，那么我们设置 $t[Z] = z_G$，如表 4 所示。在这种情况下，Z可以视为观测属性（而不是潜在属性），因为 R_0 中每个元组的该属性值都是确定的。因此，为了对 R_0建模，我们可以应用任意现有的单关系合成差分隐私方法。在我们的解决方案中，我们使用 [9]中的最先进的方法对 R_0 进行建模。

4.5 合成 R_0 和 R_1

现在，我们已经对 R_0 与 R_1之间的外键进行了建模，接下来讨论如何通过随机抽样过程，根据该模型生成合成数据。这基于标准方法，并且可以高效地完成。首先，我们使用 R_0 的模型生成一个合成元组 r^*，其中包含原本属于 R_0 的所有属性，以及新增的属性 Z。之后，我们获取 R_1的模型，该模型由 θ、p_Z和 p大小参数化。我们首先检查 $r^*[Z]$，并以概率 $p\mathrm{size}(s \mid z = r^*[Z])$抽取一个元组组大小 s。然后，我们为 R_1生成一个由 s个元组构成的组 G。G中的每个元组 t^*都以概率 $p(t^* \mid z = r^*[Z])$抽取，其中 $p(t \mid z)$的定义见第 4.1 节。借助图模型 [44], 的连接树表示，可以高效地进行元组抽样；这是先前研究 [9]采用的一种标准方法。为简洁起见，我们省略具体细节。构建好 G 中的元组后，我们设置它们的外键，使其指向 r^*。通常，可以重复上述生成过程，为 R_0 创建任意数量的元组，并在 R_1中生成相应的元组组。

4.6 扩展到多个潜在属性

第4.1至4.5节中的讨论假设，在对 R_1 和 R_0建模时，我们只使用一个潜在属性 Z。然而，通常情况下，我们可以使用多个潜在属性 Z_1、Z_2，……，并将它们视为一个复合属性，以类似于单个潜在属性 Z的方式对元组组进行建模。使用多个潜在属性的主要优势在于，它为模型中边缘分布的选择提供了更多灵活性，因为每个观测属性都可以与不同的潜在属性组合，形成潜在边缘分布。不过，潜在属性越多，建模难度也越大，因为我们需要在 R_1 和 R_0中处理高维数据。

在我们的解决方案中，我们选择为每个外键使用两个潜在属性 Z_1 和 Z_2，因为总体而言，这样能带来更好的实证结果。因此，我们对算法1和算法2作如下修订：

（1）在算法1中，我们使用复合属性 $\{Z_1, Z_2\}$ 来代替 Z。

（2）在算法 2 的第3-5 行中，我们考虑定义在 $\{A, Z_1\}$、$\{A, \ldots, Z_2\}$ 或 $\{Z_1, Z_2\}$ 上的潜在边缘分布，其中 $A \in \{A_1, \ldots, A_d\}$。

（3）在算法 2 的第9行中，我们允许每个潜在边缘分布包含 Z_1 或 Z_2 或两者。

4.7 潜在属性取值域的选择

我们的解决方案要求每个潜在属性 Z_i 都是具有给定取值域的分类属性。虽然 Z_i 的取值域中的元素可以是任意的，但取值域 $|Z_i|$ 的大小会对我们解决方案的性能产生显著影响。具体来说，请回想一下，我们的解决方案使用 Z_i 来刻画 R_1 中每个元组组的构成，即，每个 Z_i 值代表元组组的一种类型。当每个 $|Z_i|$ 较小时，我们将没有足够多的类型来对 R_1 中的元组组进行分类，这会导致对数据建模不准确。另一方面，如果 $|Z_i|$较大，那么每个潜在边缘分布将包含过多的计数；在这种情况下，潜在边缘分布中的“信号”过于稀疏，无法满足我们的 λ-实用性要求（参见算法2 的第 9 行），也就是说，我们无法利用潜在边缘分布来建模 R_1。

为解决上述问题，我们将潜在属性 Z_1 和 Z_2 的取值域大小设置如下。令 A 为 R_1 中取值域最大的属性。我们令 $|Z_1| = |Z_2| = k$，其中 k 是最大的

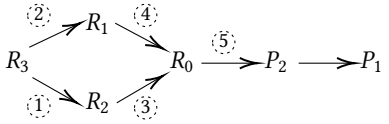


图 3：多个外键引用的示例。

算法 3： 对多个外键进行建模

输入： 数据库 R，严格全序 $\prec$

对于 每个私有外键 FK($R, R_\prime$) 按 $\prec$ 升序 **do**
  应用算法 2 对 FK(R, $R_\prime$) 建模，并使用估计的潜在属性扩充 $R_\prime$ 与估计的潜在属性；

对于 每个没有私有外键但被至少一个私有关系引用的关系 R **执行**
  为 R 构建差分隐私单关系模型（参见第 4.4 节）；

返回所有已构建的模型；

数，使得 $\{Z_1, Z_2\}$ 和 $\{Z_i, A\}$ 上的潜在边缘分布均满足 λ-有用性。也就是说，我们将 Z_i的取值域大小设为尽可能大，同时确保我们的方案能够使用潜在边缘分布来建模 Z_1 与 Z_2之间的联合分布，以及每个观测属性与 Z_i之间的相关性。

5 多个外键的解决方案

假设输入数据库 R 包含一个主私有关系 R_0、次级私有关系 R_1、R_2、.....以及公共关系 P_1、P_2、.....。对于任意两个关系 R、$R_\prime \in$ R，若 R 引用 $R_\prime$，我们用 FK(R、$R_\prime$) 表示从 R 到 $R_\prime$ 的外键。根据我们的问题定义，公共关系之间的外键不会造成任何隐私问题。此外，公共关系 P 不应存在指向私有关系的外键；否则，P 就应当是次级私有关系。因此，R 中只有两类需要处理的外键：(i) FK(R_i、R_j)，其中 R_i 和 R_j 都是私有关系；以及 (ii) FK(R_i、P)，其中 P 是公共关系。我们将这两类外键称为私有外键。

注意，在 R 中的私有外键集合上存在一个严格全序 $\prec$ 在，使得 FK($R, R_\prime$) $\prec$FK($R_\prime R_{\prime\prime}$) 始终成立。例如，图 3 展示了一个包含四个私有关系 $R_0, \ldots, R_3$和两个公开关系 P_1, P_2的数据库中的外键引用。与每个私有外键关联的圆圈数字表示该外键在全序 $\prec$中的索引。注意，FK(R_1, R_0) $\prec$FK(R_0, P_2) 以及 FK(R_2, R_0) $\prec$FK(R_0, P_2)。一般而言，可以通过对表示外键引用的有向无环图进行拓扑排序来构造这样的严格全序。

下面，我们将说明如何基于上述全序 $\prec$扩展第 4 节中的解决方案，以处理多个外键的情况。

5.1 多重外键建模

算法 3 展示了我们用于建模 R 的伪代码。该算法对私有外键进行线性扫描

算法 4： 无公共关系的合成

输入： 不含公共关系的数据库 R，严格全序 $\prec$

输出： R 的合成版本 R*

根据 R_0 的单关系模型合成 R_0^*；

对于每个私有外键 外键(R_i, R_j) 按降序排列 $\prec$ **do**
  如果 R_i^*尚未生成**那么**
    $R_i^* = \emptyset$;
    对于每个合成元组 t^*，在 R_j^***中执行**
      采样一个元组组 G^* 以用于 t^* 使用该用于FK(R_i, R_j) 的图模型；
      将 G^* 中的元组插入 R_i^*；
  else
    令 $S = R_i^*$；
    对于每个合成元组 t^* 在 R_j^***执行**
      使用.....的图模型，为 t^* 采样一个元组组 G^* FK(R_i, R_j)的图模型，但将样本空间限制为 S;
      对于 每个元组 t_i^*在 R_i^*且也在 G^* **执行**
        令 t_i^*指代 t^*;
        从 S 中移除 t_i^*;

返回 所有生成的合成关系；

按 $\prec$升序排列的 R。对于每个私有外键 FK($R, R_\prime$)，算法应用算法 2 对 FK($R, R_\prime$) 进行建模，该算法还会将与 R 中每个元组组关联的估计潜在属性添加到 $R_\prime$ 中。随后，在处理遇到的任何外键 FK($R_\prime R_{\prime\prime}$) 时，使用增强后的 $R_\prime$ 代替原始的 $R_\prime$。线性扫描终止后，算法继续检查 R 中满足以下条件的关系：(i) 没有私有外键，但 (ii) 至少被一个私有关系引用。这类关系不由算法 2 建模，但会添加数据合成所需的潜在属性。我们按照第 4.4 节中的流程，使用 [9], 中的差分隐私单关系模型建模这类关系。

例如，给定图 3 中的数据库 R，算法 3 首先处理FK(R_3, R_2)，并向 R_2 添加估计的潜在属性。之后，它分别继续对 FK(R_3, R_1)、FK(R_2, R_0) 和 FK(R_1, R_0) 建模，并将估计的潜在属性插入 R_1 和 R_0。请注意，在此阶段，R_0 会分别因 FK(R_2, R_0) 和 FK(R_1, R_0) 而增补两组潜在属性。当算法随后处理 FK(R_0, P_2) 时，这两组潜在属性都会被纳入考量。

5.2 无公开关系的合成

下面，我们将解释如何基于算法 3 构建的模型生成 R 的合成版本。为便于说明，我们首先考虑 R 不包含任何公开关系的情况。在这种情况下，主关系 R_0 不具有引用其他关系的任何外键，因此，算法 3 将为 R_0构建一个单关系模型。

算法 4 展示了在 R 中不存在公共关系的情况下，我们的数据合成方法的伪代码。该方法首先利用为 R_0 构建的单关系模型，生成 R_0 的合成版本 R_0^*（第 1 行）。随后，它按照 $\prec$ 的降序检查 R 中的外键（第 2–14 行）。对于遇到的每个外键 FK(R_i, R_j)，算法根据在处理其他外键时 R_i^* 是否已被创建来区分两种情况（第 3 行）。如果 R_i^* 不存在，则算法初始化 $R_i^* = \emptyset$，并获取 R_j 的合成版本 R_j^*（由于外键的检查顺序，该版本必定已经生成）。对于 R_j^* 中的每个合成元组 t^*，算法使用 FK(R_i, R_j) 的图模型为 t^* 采样一个元组组 G^*，并将 G^* 中的元组插入 R_i^*（第 3–7 行）。

另一方面，如果 R_i^*已经存在，那么处理FK(R_i, R_j) 就会变得更加复杂。例如，考虑图 3 中所示的外键，并假设公共关系 P_1 和 P_2 不存在。如果我们按照 $\prec$ 的降序应用算法 4 来处理这些外键，那么 FK(R_3, R_1) 将在 FK(R_3, R_2) 之前被处理。因此，当算法检查 FK(R_3, R_2) 时， R_3的一个合成版本 R_3^*将已经构建完成。在这种情况下，如果算法将 FK(R_3, R_2)当作 R_3^* 不存在来处理，那么它将生成 R_3的另一个合成版本。我们不清楚如何协调 R_3的两个合成版本之间的差异。

为了解决上述问题，当 R_i^*已存在时，算法 4 将按如下不同方式对合成元组组进行采样。首先，它构建集合 $S = R_i^*$（第 9 行）。然后，从模型中为 FK(R_i, R_j)采样元组组时，将样本空间限制为 S，即只有 S中的元组才有机会被采样（第 11 行）。此外，每次从 S中采样一个元组 t_i^*后，都会将其从 S中移除，以避免算法再次采样该元组（第 14 行）。因此，为 FK(R_i, R_j)生成的元组组只包含先前生成的 R_i^*中的元组，因此我们无需处理 R_i的两个不同合成版本。

5.3 使用公共关系进行合成

第 5.2 节中的解决方案仅处理私有关系之间的外键。接下来，我们将扩展这一解决方案，以考虑形如 FK(R_i, P)的外键，其中 R_i 和 P 分别是私有关系和公共关系。

请注意，当算法 3 处理 FK(R_i, P) 时，它将：(i) 为 FK(R_i, P) 构建图模型；(ii) 为 P 增加与 R_i 相关的估计潜在属性；(iii) 在增强后的 P 上创建差分隐私单关系模型（因为 P 没有指向其他关系的私有外键）。如果我们要生成与 FK(R_i, P) 相关的合成数据，一种朴素方法是首先根据 P 的单关系模型构建 P^* 的合成版本，然后从 FK(R_i, P) 的图模型中采样合成元组组。然而，就准确性而言，这种方法并不理想，因为它没有利用 P 是公共关系、可以直接发布这一事实。另一种朴素方法是发布增强后的 P，然后利用其中的潜在属性，从图模型中为 R_i 采样合成元组组。遗憾的是，这会违反差分隐私，因为 P 中每个元组的潜在属性都传达了私人信息，不能直接公开。

为解决上述朴素方法的局限性，我们采用如下改进方案。首先，对于 P 中的每个元组 t，我们使用 P 的差分隐私单关系模型来推断 t 的潜在属性。这不会产生任何隐私成本，因为它仅使用公共信息（即 P）和差分隐私模型。然后，对于带有其推断潜在属性的每个元组 t，我们通过从 FK(R_i, P) 的图模型中进行采样，为 t 生成一个合成元组组。这样，我们无需合成 P，同时还能确保差分隐私。

我们将这一改进方案纳入算法 4，具体如下。首先，我们修改算法 4 的输入，使 R 可以同时包含私有关系和公共关系。其次，如果主私有关系 R_0 具有指向公共关系的任何外键，我们就跳过算法 4 的第 1 行，并将 R_0^* 的生成留给第 3 至 14 行。之后，当算法按 $\prec$降序处理私有外键时，我们会根据 R_j是私有还是公共，采用不同方式处理每个外键 FK(R_i, R_j)。如果 R_j 是私有的，我们就按照算法 4 第 3 至 14 行的方式处理 FK(R_i, R_j)。否则，我们使用上文提到的改进方案处理 FK(R_i, R_j)，但有一处调整：如果 R_i 的合成版本 R_i^*已经存在，那么我们就使用 R_i^*作为生成元组组的样本空间（见算法 4 第 9 至 14 行）。

6 确保差分隐私

在本节中，我们将说明如何通过为调用的每个算法选择合适的参数，确保我们的解决方案满足 (ϵ, δ)-差分隐私。我们首先介绍 L_2 敏感度 的概念，以及关于高斯分布噪声的一个已知结果。

定义 *6.1*（L_2 敏感度 *[16]*）。设 f是一个将数据库映射为实向量的函数。 f的 L_2 敏感度记为 $\Delta(f)$，对于任意两个相邻数据库 R 和 R$\prime$，其最大值为 $\|f(\mathrm{R}) - f(\mathrm{R}\prime)\|_2$，其中 $\|\cdot\|_2$ 表示 L_2 范数。

定理 6.2（[3]）。令$\{f_1, \ldots, f_k\}$为一个函数集。对于任意 $i = 1, \ldots, k$，假设我们向 f_i 的输出中的每个元素注入独立的高斯噪声$\mathrm{N}(0, \sigma_i^2)$。那么，扰动后的函数整体满足（$\epsilon, \delta$）-差分隐私，当且仅当

$$\Phi\left(\frac{\gamma}{2} - \frac{\epsilon}{\gamma}\right) - e^{\epsilon} \cdot \Phi\left(-\frac{\gamma}{2} - \frac{\epsilon}{\gamma}\right) \leq \delta, \tag{6}$$

其中，Φ 是标准正态分布的累积分布函数，并且

$$\gamma = \sqrt{\sum_{i=1}^{k} \left(\frac{\Delta(f_i)}{\sigma_i}\right)^2}.$$

注意，公式（6）左侧随γ单调递增。因此，若要向一组函数 {f_1、……、 f_k} 注入高斯噪声以实现（ϵ， δ）-差分隐私，那么根据定理 6.2，可为每个函数 f_i 设置如下噪声尺度 σ_i：

（1）令γmax 为满足公式（6）的最大 γ，其中 ϵ和 δ给定。

（2）选择 σ_i，以确保 $\sum_{i=1}^{k} \left(\frac{\Delta(f_i)}{\sigma_i}\right)^2 \leq \gamma^2$） max。

为方便起见，我们将 $\sum_i \left(\frac{\Delta(f_i)}{\sigma_i}\right)^2$）称为函数集 {$f_i$} 的隐私消耗。

请注意，每当我们的算法从 R 访问私人信息时，它们总是向其中注入高斯噪声以实现隐私保护。因此，如果我们能够量化隐私消耗

我们可以分别分析解决方案中每个组件的ϵ，δ，然后据此设置参数，以确保满足 (ϵ, δ)-DP。对此，我们注意到，算法的隐私消耗取决于其所应用的私有关系，因为在主私有关系 R_0 中添加或删除一个元组，通常会导致不同次级私有关系发生不同程度的变化。为考虑这些依赖关系，我们将每个关系的元组乘数和组乘数 定义如下。

定义*6.3*（元组和组乘数）。对于R 中的任意关系 R，其元组乘数 是两个相邻数据库之间可能发生变化的元组的最大数量。此外，相对于外键 FK(R，$R_\prime$)，其 组乘数 是两个相邻数据库之间可能发生变化的、由 FK(R，$R_\prime$) 诱导的元组组的最大数量。

接下来，我们根据输入关系的元组和组乘数来量化算法的隐私消耗。我们从算法 1 开始。（由于篇幅限制，证明见附录 C。）

引理6.4。 假设我们在关系 R 上应用算法*1*，并考虑由外键 FK($R, R_\prime$) 诱导的元组组。令 μ_g 为相对于 FK($R, R_\prime$) 的 R 的组乘数。令 τ 为 FK($R, R_\prime$) 的最大多重度。那么，算法*1*的隐私消耗为：

$$C_1(T, m_{\mathcal{M}}) = T \cdot \mu_g^2 \cdot \left(\frac{m_{\mathcal{M}} \cdot \tau^2}{\sigma_l^2} + \frac{1}{\sigma_{size}^2} + \frac{1}{\sigma_z^2} \right), \tag{7}$$

其中 T，$\mathcal{M}$ σ_z、σ_{size}、σ_l是算法*1*的输入参数，$m_{\mathcal{M}}$是$\mathcal{M}$中的潜在边缘分布数量。

接下来，我们考虑算法 2 第 1 行和算法 3 第 4 行使用的差分隐私单关系建模方法 [9] 。根据 [9], 中的引理 2，我们有以下结果。

推论 6.5。 假设我们在 *[9]* 中对一个关系 R应用算法*6*，其元组乘数为 μ_t。设 d为 R中的属性数量。那么，该算法的隐私消耗为：

$$C_{single}(d) = \mu_t^2 \cdot \left(\frac{1}{\sigma_{\mathcal{U}}^2} + \frac{2d(d-1)}{\sigma_R^2} + \frac{t \cdot k}{\sigma_h^2} + \frac{d+t}{\sigma_m^2} \right), \tag{8}$$

其中 $\sigma_{\mathcal{U}}$ σ_R、σ_h、σ_m、t、k是该算法的参数 。

接下来，我们给出算法 2 的以下结果。

引理 6.6. 假设我们在关系 R 上应用算法 *2*，并考虑由外键 FK($R, R_\prime$) 所诱导的元组组。令 μ_t 为 R 的元组乘数，d 为 R 中属性的数量。那么，算法 *2* 的隐私消耗为：

$$C_2(\mathrm{FK}(R, R')) = C_{single}(d) + C_1(T, 2d) + \sum_{i=1}^{T_C} C_1(1, 2d + i \cdot n_{inc}) + \mu_t^2 \cdot \frac{n_C \cdot T_C}{\sigma_{err}^2}. \tag{9}$$

根据引理6.6 和推论 6.5，我们可以通过将以下两部分的隐私消耗相加，量化算法3 的隐私消耗：(i)应用算法2处理 R 中的每个私有外键；以及 (ii) 对每个不包含私有外键但被至少一个私有关系引用的关系应用单关系方法。然后我们可以设置.....的参数

相应地配置 PrivLava，以确保其在预先设定 ϵ和 δ的情况下满足 (ϵ, δ)-差分隐私。请注意，一旦我们从算法 3 获得差分隐私模型，使用这些模型生成合成数据不会产生任何隐私成本。此外，我们可以使用合成数据回答任意查询，而无需承担任何额外的隐私开销。这是因为差分隐私的后处理性质 [17]：如果某个算法满足 (ϵ, δ)-差分隐私，那么对该算法的输出进行任何后处理都不会降低其隐私保证。

最后，我们说明如何在算法1和算法2的各个组成部分之间分配主要消耗。在算法1中，我们将生成噪声潜在边缘分布（第19—22行）所占的主要消耗设为80%，并按照4：1的比例，将剩余的20%分配给$p$size的生成（第13—18行）和$p$size的生成（第8—12行）。换言之，$\frac{m_{\mathcal{M}} \cdot \tau^2}{\sigma_l^2} : \frac{1}{\sigma_{size}^2} : \frac{1}{\sigma_z^2} = 20 : 4 : 1$。此外，我们设置 $T = 6$。在算法2中，我们将20%的隐私消耗分配给第1行，75%分配给算法1的调用，剩余的5%分配给第9—14行。我们还设置 $T_C = 2$、$n_C = 400$以及$n_{\mathrm{inc}} = d/4$。我们根据在一系列不同数据集上的经验校准来确定这些分配比例，但这些比例不一定是最优的。确定隐私预算的最优分配是未来工作中一个有趣的方向。

7 实验

7.1 设置

数据集。 我们使用来自 [10, 42]和 TPC- H基准测试 [1] 的两个人口普查数据库进行实验。这些人口普查数据库分别包含在加利福尼亚和法兰西岛收集的人口普查数据。每个数据库都包含一个人员关系 R_P和一个家庭关系 R_H，其中外键 $FK(R_P, R_H)$表示每个人所属的家庭。我们分别将 R_H和 R_P视为主要私有关系和次级私有关系。表 7 展示了这两个人口普查数据库的统计信息。

TPC-H 基准测试包含 8 个关系和 8 个外键，如图 4所示。我们分别将 Orders 和Lineitem 作为主要私有关系和次级私有关系，并将其余关系视为公开关系。需要注意的是，原始的 TPC-H 基准测试基于均匀分布生成每个外键；因此，其关系间相关性最小，不适合用于评估多关系合成的有效性。基于此，我们对 TPC-H 进行了修改，以引入如下关系间相关性。

首先，我们为 Part 中的每种零件类型和零件品牌分配一个权重。然后，对于 Lineitem 中的每个元组，我们将其价格值乘以其零件类型和零件品牌的权重之积。换言之，我们让每个订单明细的价格取决于其零件的类型和品牌。具体而言，每种零件类型（相应地，每种零件品牌）的权重从 $[0.2, 1]$（相应地，$[0.2, 2]$）中选取。其次，对于每个年份 y，我们随机选择客户的一个 15% 子集，并将该年份 y 的订单中的 40% 关联到这些客户；然后，将剩余的 60% 订单随机分配给其他 85% 的客户。这会导致客户与订单年份之间产生相关性。第三，对于每个年份 y，我们以 75% 的概率删除该年份 y 的每个订单，除非该订单包含 $y - 1992$ 个订单明细。这样做是为了引入订单规模与订单年份之间的相关性。

表 7：人口普查数据集概要。(a) 加利福尼亚

关系	# 条记录	# 个属性	域大小
个人	1,690,642	23	$\approx 6.77 \times 10^{12}$
家庭	616,115	10	$\approx 3.24 \times 10^{7}$

(b) 法兰西岛

关系	# 条记录	# 属性	域大小
个人	4,297,133	14	$\approx 1.84 \times 10^{10}$
家庭	1,911,412	10	$\approx 1.24 \times 10^{7}$

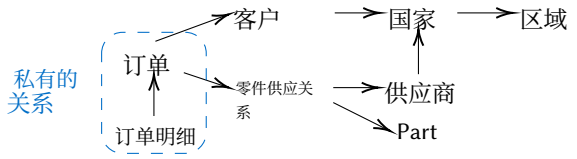


图 4：TPC-H 中的外键依赖关系。

经过上述更改，我们得到一个修订后的 TPC-H基准测试，其中订单明细包含约 4×10^6 个元组，订单包含约 1×10^6 个元组。

回顾一下，我们的解决方案要求给定每个私有外键 $FK(R, R_\prime)$ 的最大重数。在实践中，数据所有者可以根据先验知识选择一个上界[2]；以往工作也采用了这一假设 [46]。我们在实验中沿用相同的假设。具体而言，我们考察 R 中相对于 $R_\prime$ 的所有元组组大小，并计算组大小的 99% 分位数。然后，我们将该分位数用作 $FK(R, R_\prime)$ 的最大重数。

基线。 我们考虑三种最先进的差分隐私数据合成方案：PrivMRF [9]，、PrivBayes [49]，和 PB-PGM [34]。这三种方案均针对单关系合成设计，但我们按如下方式将其应用于数据集。首先，我们使用 80% 的隐私预算为每个私有关系生成一个合成版本。然后，我们将剩余 20% 的预算用于为每个私有外键 $FK(R, R_\prime)$ 生成噪声直方图 H^*，统计 R 中相对于 $FK(R, R_\prime)$ 的元组组大小。之后，我们在保持 H^* 中组大小分布不变的情况下，将 R 的合成元组与 $R_\prime$ 的合成元组随机关联。

此外，我们还纳入了一个基于反规范化的基线方法 Denorm，其工作流程如下。首先，它使用 20% 的隐私预算，为每个外键的组大小生成一个噪声直方图。然后，将输入数据库中所有相关关系连接成一个反规范化关系，并使用剩余 80% 的隐私预算，通过 PrivMRF 为其生成一个合成版本 R^*。之后，再将其规范化 R^* 回一组关系。例如，假设 R^* 是 $R_H \bowtie R_P$的一个合成版本，其中 R_H和 R_P分别是加利福尼亚或法兰西岛数据集中的家庭关系和人员关系。为将 R^*规范化，我们首先按字典序对R^* 中的元组进行排序，家庭属性排在所有个人属性之前。这样便得到一个有序序列，其中相邻元组的家庭属性相似。随后，我们按照噪声组大小直方图中观察到的组大小分布，将排序后的元组划分为若干组。我们将每个组G 视为一个家庭，并通过先随机选择 G 中的一个元组 t，再将 t的家庭属性复制到其中其他所有元组，从而统一 G 中的家庭属性属性到 G中的所有其他元组中。TPC-H 数据库也以类似方式处理。

隐私参数。 在我们的所有实验中，我们改变 ϵ 但固定$\delta = \frac{1}{n}$，其中 n 是次级私有关系中的元组数量。先前的工作 [6, 9, 19, 22]也采用了类似的设置。

7.2 人口普查数据库上的实验结果

我们的第一组实验聚焦于两个人口普查数据库。我们考虑以下形式的聚合查询 $Q(s,\ P_h,\ c,\ \{P_i\})$。

统计 h 满足以下条件的家庭数量：

- h 的大小等于 s，并且
- h满足其属性上的谓词 P_h，并且• h包含 c人员 $p_1, \ldots p_c$，使得每个 $p_i (i = 1, \ldots, c)$都满足其属性上的谓词 P_i。

直观地，当 $c = 1$时，Q 在合成数据库 R* 上的准确性表明R* 是否保留了家庭关系和个人关系之间的相关性。同时，当 $c > 1$时，Q 也可用于评估 R* 是否保留了家庭成员之间的组内相关性。

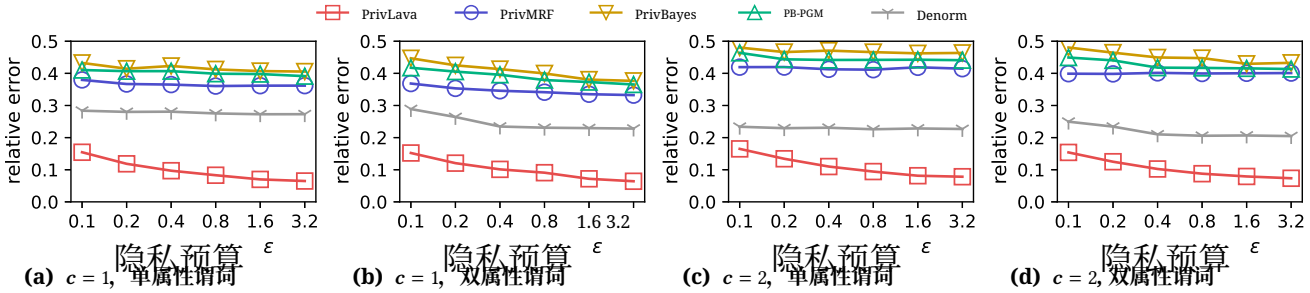
我们考虑 $c = 1$和 $c = 2$的情况，其中s是随机选择的。对于 P_h和$\{P_i\}$，我们考虑包含1个属性和2个属性的合取谓词，使得每个属性 A_j上的条件为 $t[A_j] \in S_j$，其中 t是一个元组，S_j是从A_j的取值域中随机选择的值的集合。我们令 $\frac{|S_j|}{|A_j|} = (0.2)^{1/k}$，其中 k是 P_h和$\{P_i\}$中的条件总数。换句话说，我们将每个S_j的大小设为常数，使得在均匀分布的数据上，所有谓词组合后的选择率为0.2。我们通过其相对误差</style}来衡量每个查询 Q 的准确性，该误差定义为

$$\frac{\text{absolute error of } Q}{\max\{\text{actual result of } Q,\ 0.01 \cdot \text{total number of households}\}},$$

其中，“$0.01 \cdot$ 家庭总数”是一个正则化项，用于减轻查询结果过小所造成的影响。

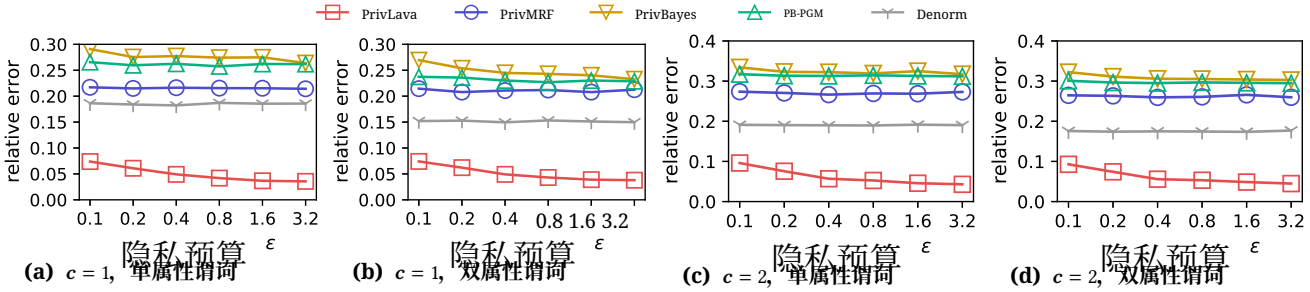
图5（分别为图6）展示了各方法在加利福尼亚（分别为法兰西岛）上对 10,000 个查询取平均后的误差。可以看到，无论是 $c = 1$ 还是 $c = 2$，也无论每个谓词涉及 1 个还是 2 个属性，PrivLava 在查询准确度方面都显著优于其竞争方法。这证明了 PrivLava 在对输入数据中的关系间相关性和组内相关性建模方面的有效性。随着隐私预算 ϵ 的增加，PrivLava 的准确度会提高，因为较大的 ϵ 使 PrivLava 能够在数据合成时使用较小的噪声。相比之下，基线方法的准确度几乎不会随着 ϵ 的增加而提高，这表明它们无法对外键建模是导致查询误差的主要原因。此外，Denorm 优于其他基线方法，但仍不及 PrivLava，这表明 Denorm 的反规范化方法有助于保留关系间相关性，但不如 PrivLava 的潜变量方法有效。

[2]当此类先验知识不可用时，数据所有者可以使用差分隐私算法来确定一个合适的上界，例如，通过计算外键度数的近似 99% 分位数；先前的工作 [15, 25] 已提出用于此目的的算法。



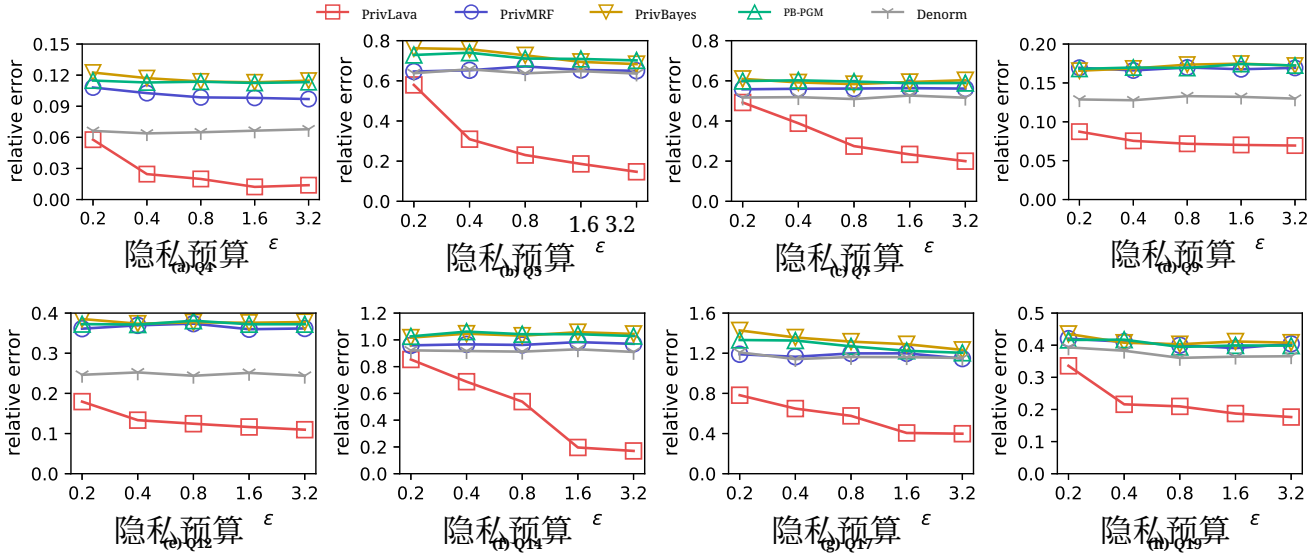
(a) $c = 1$, 单属性谓词 (b) $c = 1$, 双属性谓词 (c) $c = 2$, 单属性谓词 (d) $c = 2$, 双属性谓词

图5：加利福尼亚：相对误差与 ϵ。



(a) $c = 1$, 单属性谓词 (b) $c = 1$, 双属性谓词 (c) $c = 2$, 单属性谓词 (d) $c = 2$, 双属性谓词

图6：法兰西岛：相对误差与 ϵ。



(a) Q4 (b) Q5 (c) Q7 (d) Q9 (e) Q12 (f) Q14 (g) Q17 (h) Q19

图 7：TPC-H：相对误差与 ϵ的关系。

7.3 TPC-H上的实验结果

在第二组实验中，我们在 TPC-H 基准测试中的 8 个聚合查询上评估各方法。每个查询 Q 都返回一个值或一组通过 GROUP BY 生成的值。我们以其返回值的平均相对误差来衡量 Q 的准确度。此外，我们从相对误差的定义中去除了正则化项，因为 TPC-H 查询返回的所有值都足够大。

图 7 展示了各方法的准确率随 ϵ 变化的情况，结果取 10 次运行的平均值。可以看到，PrivLava 始终优于其他方法，而且大多数情况下优势明显。这是

与图5和图6中的结果一致。有趣的是，对于少数查询（例如，Q7、Q14、Q17），当 ϵ 较小时，PrivLava 的准确性会明显下降。主要原因是，这些查询的某些组件具有较小的选择率，因此只有在我们对输入数据的建模非常准确时，才能准确回答这些组件，而在隐私预算较小时这很难实现 ϵ。例如，Q17 的结果包含一个值，该值仅根据 Lineitem 中0.01%的元组计算得出。尽管如此，即使隐私预算 $\epsilon = 1.6$并不大，PrivLava 也已经能够在所有查询上显著优于其竞争方法。

在Q4、Q9 和 Q12 中，Denorm 相较于其他基线表现出显著的改进。然而，对于所有其他查询，Denorm 仅提供

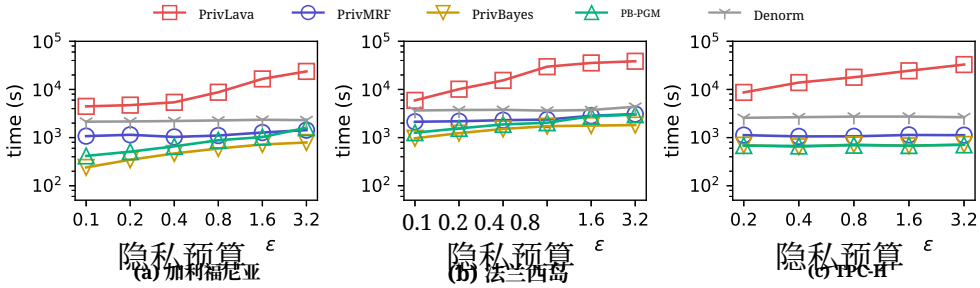


图 8：计算成本与 ϵ的关系。

查询准确度与其他基线方法相当。这与我们在第 7.2 节实验结果中的观察一致：Denorm 能够捕捉一些关系间相关性，但在建模准确度方面不如 PrivLava。

7.4 计算成本

我们最后一组实验评估了各方法在三个数据集上的计算时间。实验使用配备 32 核 2.9GHz CPU 和 Nvidia A10 GPU 的 Linux 机器进行，所有方法均使用 Python 实现。图 8 展示了各方法 10 次运行的平均处理成本。PrivLava 的计算成本会随着隐私预算 ϵ 的增加而上升，因为：(i) 更大的 ϵ 使 PrivLava 能够使用更大的边缘分布进行建模（见公式 5）；(ii) 处理这类边缘分布耗时更长。PrivLava 的计算时间高于其他基线方法，但在所有情况下，它仍能在 12 小时内完成输入数据集的处理。PrivLava 的计算时间是可以接受的，因为数据合成只需一次性付出成本，生成的合成数据则可供下游应用反复使用。

8 相关工作

如第 1节所述，已经存在相当数量的基于差分隐私进行数据合成的方法 [7, 9, 13, 21, 23, 29, 34, 36, 39, 47–50], 但它们全部聚焦于单关系案例。在这些方法中，与我们所提出的方法最为接近的是 PrivMRF [9],，它同样使用一类相似的图模型对输入数据进行建模。然而，PrivMRF 与 PrivLava 存在显著差异，其图模型不包含任何潜变量，而PrivLava中潜变量的使用正是其对外键约束进行建模的关键。潜变量的使用也使得PrivLava的模型构建算法比 [9]中的方法复杂得多。已有工作 [13, 21, 23, 29, 36, 47] 还提出了使用其他模型的差分隐私单关系合成技术（例如，Copula 函数[29], 生成对抗网络 [23, 39, 48]），但如 [9] 所示，这些技术的性能通常不如基于图模型的方法。

在非差分隐私数据合成的背景下，已有研究 [19, 20, 24, 26, 31, 32] 提出了多种处理外键的方案。具体而言， [20] 中的技术假设每个表的非键属性都受到硬约束，并利用这些约束在合成关系之间生成外键。然而，在实践中，我们并不清楚如何以保护隐私的方式获取此类约束，也不清楚它们是否足以保留由外键引起的相关性。同时，先前的工作 [24, 26, 31, 32] 提议保留输入数据中的外键，并扰动非键属性以实现隐私保护。遗憾的是，这些方法无法提供强有力的隐私保证。

此外，还有一系列研究 [5, 6, 8, 12, 15, 22, 25, 33, 35, 37, 40, 41, 45, 46] 致力于研究如何在差分隐私下使用输出扰动构建数据库引擎。正如我们在第 1节中所解释的那样，然而，这类数据库引擎可能只能准确回答数量有限的查询，因为每个查询都不可避免地会泄露有关底层数据的部分私人信息。需要注意的是，这一限制是输出扰动所固有的：Dinur 和 Nissim [14] 已证明，在任何合理强度的隐私概念下，所有输出扰动方法都会面临同样的问题。

最后，先前的工作 [30]提出了一种在差分隐私下生成数据库基准的算法。具体而言，给定数据库 D 和由查询组成的工作负载 Q，该算法在差分隐私下生成另一个数据库 D′，使得如果我们使用数据库引擎分别在 D 和 D′上处理Q，则每个查询在两个数据库上的处理时间都将相近。换言之，该算法旨在保留 Q 中查询的成本，而不是查询结果的准确性。因此， [30] 中的解决方案不适用于我们的场景。

9 结论

本文介绍了 PrivLava，这是首个在差分隐私（差分隐私）下利用外键合成关系数据的解决方案。PrivLava使用图模型对输入数据库进行建模，并纳入潜变量，以捕捉由外键约束引起的复杂相关性。它支持同时包含私有关系和公共关系的数据库，其中的任意外键都形成有向无环图。在真实数据上进行的大量实验表明，与竞争方法相比，PrivLava 在合成数据上提供了显著更高的查询准确度。

PrivLava 的主要局限在于，它需要大量数据来训练准确的外键差分隐私模型。具体而言，它需要从输入数据库中生成相对大量的观测边缘分布和潜在边缘分布，并向每个边缘分布注入高斯噪声，以实现差分隐私。当输入数据库中的元组数量较少时，注入每个边缘分布的噪声可能会淹没其中的信号（尤其是在潜在边缘分布中）；在这种情况下，PrivLava 无法有效地对外键进行建模。在未来的工作中，我们计划研究如何缓解这一问题，并扩展 PrivLava 以考虑

关系数据上的其他类型约束，例如，拒绝约束 [11]。

致谢。 本研究/项目由新加坡国家研究基金会根据其战略能力研究中心资助计划提供支持。本文中表达的任何意见、研究结果、结论或建议均属于作者个人观点，不代表新加坡国家研究基金会的立场。本工作还部分得到了英国研究与创新署繁荣伙伴计划（FAIR）的支持，该计划由EPSRC资助，资助编号为EP/V056883/1，同时也得到了艾伦·图灵研究所的支持。

参考文献

[1] 2022. TPC-H 基准测试主页。https://www.tpc.org/tpch/。[2] Gergely Acs、Claude Castelluccia 和 Rui Chen。2012 年。通过有损压缩发布差分隐私直方图。载于《2012 年 IEEE 第 12 届数据挖掘国际会议》。IEEE，1–10。[3] Borja Balle 和 Yu-Xiang Wang。2018 年。改进用于差分隐私的高斯机制：解析校准与最优去噪。载于《国际机器学习会议》。PMLR，394–403。[4] Johes Bater、Gregory Elliott、Craig Eggen、Satyender Goel、Abel N Kho 和 Jennie Rogers。2017 年。SMCQL：私人数据网络的安全查询处理。VLDB 基金会会刊 10，6（2017），673–684。[5] Johes Bater、Xi He、William Ehrich、Ashwin Machanavajjhala 和 Jennie Rogers。2018 年。Shrinkwrap：差分隐私数据联邦中的高效 SQL 查询处理。VLDB 基金会会刊 12，3（2018）。[6] Johes Bater、Yongjoo Park、Xi He、Xiao Wang 和 Jennie Rogers。2020 年。Saqe：面向数据联邦的实用隐私保护近似查询处理。VLDB 基金会会刊 13，12（2020），2691–2705。[7] Vincent Bindschaedler、Reza Shokri 和 Carl A Gunter。2017 年。隐私保护数据合成中的合理可否认性。VLDB 基金会会刊 10，5（2017）。[8] Jaroslaw Błasiok、Mark Bun、Aleksandar Nikolov 和 Thomas Steinke。2019 年。迈向实例最优的私人查询发布。载于《第三十届 ACM-SIAM 离散算法年度研讨会论文集》。SIAM，2480–2497。[9] Kuntai Cai、Xiaoyu Lei、Jianxin Wei 和 Xiaokui Xiao。2021 年。通过差分隐私马尔可夫随机场进行数据合成。VLDB 基金会会刊 14，11（2021），2190–2202。[10] 明尼苏达州人口中心。2020 年。综合公共用途微观数据系列，国际版：第 7 版。3 [数据集]。明尼阿波利斯，明尼苏达州：IPUMS。https://doi.org/10.18128/D020.V7.3。[11] Jan Chomicki 和 Jerzy Marcinkowski。2005 年。使用元组删除进行最小更改完整性维护。Information and Computation 197，1-2（2005），90–121。[12] Graham Cormode、Tejas Kulkarni 和 Divesh Srivastava。2018 年。本地差分隐私下的边缘分布发布。载于《2018 年数据管理国际会议论文集》。131–146。[13] Graham Cormode、Cecilia Procopiuc、Divesh Srivastava 和 Thanh TL Tran。2012 年。稀疏数据的差分隐私摘要。载于《第 15 届数据库理论国际会议论文集》。299–311。[14] Irit Dinur 和 Kobbi Nissim。2003 年。在保护隐私的同时揭示信息。载于《第二十二届 ACM SIGMOD-SIGACT-SIGART 数据库系统原理研讨会论文集》。202–210。https://doi.org/10.1145/ 773153.773173[15] Wei Dong、Juanru Fang、Ke Yi、Yuchao Tao 和 Ashwin Machanavajjhala。2022 年。R2T：带外键的差分隐私查询求值的实例最优截断。载于《ACM SIGMOD 数据管理国际会议论文集》。[16] Cynthia Dwork、Frank McSherry、Kobbi Nissim 和 Adam Smith。2006 年。为私人数据分析中的敏感度校准噪声。载于《密码学理论会议》。Springer，265–284。[17] Cynthia Dwork、Aaron Roth 等。2014 年。差分隐私的算法基础。Foundations and Trends® in Theoretical Computer Science 9，3–4（2014），211–407。[18] Marco Gaboardi、Emilio Jesús Gallego Arias、Justin Hsu、Aaron Roth 和 Zhiwei Steven Wu。2014 年。双重查询：面向高维数据的实用私人查询发布。载于《国际机器学习会议》。PMLR，1170–1178。[19] Chang Ge、Shubhankar Mohapatra、Xi He 和 Ihab F Ilyas。2021 年。Kamino：具备约束感知能力的差分隐私数据合成。VLDB 基金会会刊 14，10（2021），1886–1899。[20] Amir Gilad、Shweta Patwa 和 Ashwin Machanavajjhala。2021 年。在基数约束和完整性约束下合成关联数据。载于《2021 年数据管理国际会议论文集》。619–631。[21] Moritz Hardt、Katrina Ligett 和 Frank McSherry。2012 年。用于差分隐私数据发布的简单实用算法。《神经信息处理系统进展》第 25 卷（2012 年）。[22] Noah Johnson、Joseph P Near 和 Dawn Song。2018 年。迈向适用于 SQL 查询的实用差分隐私。《VLDB 基金会会刊》11，5（2018 年），526–539。[23] James Jordon、Jinsung Yoon 和 Mihaela Van Der Schaar。2018 年。PATE-GAN：利用差分隐私保证生成合成数据。载于《国际学习表征会议》。[24] Christopher T Kenny、Shiro Kuriwaki、Cory McCartan、Evan TR Rosenman、Tyler Simko 和 Kosuke Imai。2021 年。差分隐私在普查数据中的应用及其对选区重划的影响：以2020年美国人口普查为例。《科学进展》7，41（2021 年），eabk3283。[25] Ios Kotsogiannis、Yuchao Tao、Xi He、Maryam Fanaeepour、Ashwin Machanavajjhala、Michael Hay 和 Gerome Miklau。2019 年。PrivateSQL：差分隐私 SQL 查询引擎。《VLDB 基金会会刊》12，11（2019 年），1371–1384。[26] Amy Lauger、Billy Wisniewski 和 Laura McKenna。2014 年。美国人口普查局的披露规避技术：当前实践与研究。披露规避研究中心，美国人口普查局（2014 年）。[27] Chao Li、Michael Hay、Gerome Miklau 和 Yue Wang。2014 年。差分隐私下适用于范围查询的数据与工作负载感知算法。《VLDB 基金会会刊》7，5（2014 年）。[28] Chao Li、Gerome Miklau、Michael Hay、Andrew McGregor 和 Vibhor Rastogi。2015 年。矩阵机制：在差分隐私下优化线性计数查询。《VLDB 期刊》24，6（2015 年），757–781。[29] Haoran Li、Li Xiong 和 Xiaoqian Jiang。2014 年。使用 Copula 函数对多维数据进行差分隐私合成。载于《数据库技术进展：论文集——国际数据库技术扩展会议》第 2014 卷。美国国立卫生研究院公共访问，475。[30] Wentian Lu、Gerome Miklau 和 Vani Gupta。2014 年。为不可信系统评估生成隐私合成数据库。载于《IEEE 第30届国际数据工程会议（2014）》。IEEE，652–663。[31] Paul M Massell 和 Jeremy M Funk。2007 年。使用噪声保护数量数据表的最新进展：通过平衡改善数据质量，以及保留保护效果的舍入。美国人口普查局 [保管人]]。[32] Laura McKenna。2018 年。1970至2010年人口与住房十年一次普查所采用的披露规避技术。技术报告。美国人口普查局、经济研究中心。[33] Ryan McKenna、Gerome Miklau、Michael Hay 和 Ashwin Machanavajjhala。2018 年。差分隐私下高维统计查询误差的优化。《VLDB 基金会会刊》11，10（2018 年），1206–1219。[34] Ryan McKenna、Daniel Sheldon 和 Gerome Miklau。2019 年。用于差分隐私的基于图模型的估计与推断。载于《国际机器学习会议》。PMLR，4435–4444。[35] Frank D McSherry。2009 年。隐私集成查询：用于隐私保护数据分析的可扩展平台。载于《2009年 ACM SIGMOD 数据管理国际会议论文集》。19–30。[36] Noman Mohammed、Rui Chen、Benjamin CM Fung 和 Philip S Yu。2011 年。用于数据挖掘的差分隐私数据发布。载于《第17届 ACM SIGKDD 知识发现与数据挖掘国际会议论文集》。493–501。[37] Prashanth Mohan、Abhradeep Thakurta、Elaine Shi、Dawn Song 和 David Culler。2012 年。GUPT：让隐私保护数据分析变得简单。载于《2012年 ACM SIGMOD 数据管理国际会议论文集》。349–360。[38] Aleksandar Nikolov、Kunal Talwar 和 Li Zhang。2016 年。差分隐私的几何学：小型数据库与近似情形。《SIAM 计算杂志》45，2（2016 年），575–616。[39] Noseong Park、Mahmoud Mohammadi、Kshitij Gorde、Sushil Jajodia、Hongkyu Park 和 Youngmin Kim。2018 年。基于生成对抗网络的数据合成。《VLDB 基金会会刊》11，10（2018 年）。[40] Davide Proserpio、Sharon Goldberg 和 Frank McSherry。2014 年。将数据校准至隐私数据分析中的敏感度：用于差分隐私分析加权数据集的平台。《VLDB 基金会会刊》7，8（2014 年），637–648。[41] Wahbeh Qardaji、Weining Yang 和 Ninghui Li。2014 年。Priview：边际列联表的实用差分隐私发布。载于《2014年 ACM SIGMOD 数据管理国际会议论文集》。1435–1446。[42] Steven Ruggles、Sarah Flood、Ronald Goeken、Megan Schouweiler 和 Matthew Sobek。2022 年。IPUMS 美国：第 12版。0 [数据集]。美国明尼苏达州明尼阿波利斯：IPUMS。https://doi.org/10.18128/D010.V12.0。[43] Yuchao Tao、Xi He、Ashwin Machanavajjhala 和 Sudeepa Roy。2020 年。计算带连接的计数查询的局部敏感度。载于《2020年 ACM SIGMOD 数据管理国际会议论文集》。479–494。[44] Martin J Wainwright、Michael I Jordan 等。2008 年。图模型、指数族与变分推断。《机器学习基础与趋势》1，1–2（2008 年），1–305。[45] Tianhao Wang、Bolin Ding、Jingren Zhou、Cheng Hong、Zhicong Huang、Ninghui Li 和 Somesh Jha。2019 年。在

本地差分隐私下回答多维分析查询。载于 *2019*年数据管理国际会议论文集。159–176。[46] Royce J Wilson、Celia Yuxin Zhang、William Lam、Damien Desfontaines、Daniel Simmons-Marengo 和 Bryant Gipson。2020 年。具有有界用户贡献的差分隐私 SQL。隐私增强技术会刊 2（2020 年），230–250。[47] Chugui Xu、Ju Ren、Yaoxue Zhang、Zhan Qin 和 Kui Ren。2017 年。DPPro：通过随机投影实现差分隐私高维数据发布。*IEEE* 信息取证与安全汇刊 12，12（2017 年），3081–3093。[48] Lei Xu、Maria Skoularidou、Alfredo Cuesta-Infante 和 Kalyan Veeramach aneni。2019 年。使用条件生成对抗网络建模表格数据。神经信息处理系统进展 32（2019 年）。[49] Jun Zhang、Graham Cormode、Cecilia M Procopiuc、Divesh Srivastava 和 Xiaokui Xiao。2017 年。PrivBayes：通过贝叶斯网络实现隐私数据发布。*ACM* 数据库系统汇刊（*TODS*）42，4（2017 年），1–41。[50] Jun Zhang、Xiaokui Xiao 和 Xing Xie。2016 年。Privtree：用于层次分解的差分隐私算法。载于*2016*年数据管理国际会议论文集。155–170。

参数估计

在本节中，我们详细阐述算法1第23行中针对 θ 的参数估计方法。回顾一下，我们的参数估计目标是最大化总体似然 $\sum_G \log p(G)$，其中 $p(G)$在公式（2）中定义。为此，我们使用期望最大化，其在每次迭代 $j \in [T]$ 中尝试最大化以下期望对数似然：

$$Q^{(j)} = \sum_{G \in R} \sum_{z} p^{(j)}(z \mid G) \log p(G, z), \tag{10}$$

其中 $p^{(j)}(z \mid G)$ 是第 j 次迭代中由公式 (3)定义的条件潜变量分布 $p(z \mid G)$，而 $p(G, z)$ 的定义见公式 (4)。

令 $p(t, z)$ 为公式 (1)中定义的内容，且 θ 是 p 的参数。令 p元组$(t, z; \theta)$ 表示使用 θ 参数化时的 $p(t, z)$，并令 p元组$(t \mid z; \theta) = p_{\text{tuple}}(t,z;\theta)/\sum_t p_{\text{tuple}}(t,z;\theta)$。~~为方便~~起见，当上下文明确时，我们在 p元组$(\cdot)$ 中省略 θ。我们将证明，算法1第 j 次迭代中的估计结果为：

$$p_Z^{(j)}, p_{\text{size}}^{(j)}, p_{\text{tuple}}^{(j)} = \underset{p_Z, p_{\text{size}}, p_{\text{tuple}}}{\arg\max}\ Q^{(j)}. \tag{11}$$

如第4.2节所述，算法1第23行调用了 [34], 中的梯度下降法，该方法称为 PGM。我们先简要介绍 PGM，然后证明算法1第23行对 PGM 的调用会返回元组 $p^{(j)}$的参数 $\theta^{(j)}$。给定一组边缘分布 M，PGM 通过最小化 $L = \|\mathbf{M}_R - \mathbf{M}_\theta\|_2$ 来确定元组 p 的参数 θ，其中 $\mathbf{M}_R$ 是关系 R 的噪声计数，$\mathbf{M}_\theta$ 是元组 p 的边缘分布计数。接下来，我们首先将最大化式（10）的问题转化为最大化一个关于元组($p$tuple(t, z)的新目标函数的问题，如引理 A.1 所示。然后，我们将在定理 A.2 中证明该新目标函数是凹函数，因此可以使用 PGM 等梯度上升方法将其最大化。

根据式（10）、（11）和（4），我们有：

$$\begin{aligned}
&p_Z^{(j)}, p_{\text{size}}^{(j)}, p_{\text{tuple}}^{(j)} \\
&= \underset{p_Z, p_{\text{size}}, p_{\text{tuple}}}{\arg\max} \sum_{G \in R} \sum_{z} p^{(j)}(z \mid G) \log p(G, z) \\
&= \underset{p_Z, p_{\text{size}}, p_{\text{tuple}}}{\arg\max} \sum_{G \in R} \sum_{z} p^{(j)}(z \mid G) \\
&\qquad \cdot \log \left(p_Z(z) \cdot p_{\text{size}}(|G| \mid z) \cdot \prod_{t \in G} p(t \mid z) \right) \\
&= \underset{p_Z, p_{\text{size}}, p_{\text{tuple}}}{\arg\max} \sum_{G \in R} \sum_{z} p^{(j)}(z \mid G) \Bigg(\log p_Z(z) + \log p_{\text{size}}(|G| \mid z) \\
&\qquad + \sum_{t \in G} \log p_{\text{tuple}}(t \mid z; \theta) \Bigg).
\end{aligned}$$

注意，我们可以分别如下计算 $p^{(j)}Z, p^{(j)}$大小 $p^{(j)}$元组：

$$\begin{aligned}
p_Z^{(j)} &= \underset{p_Z}{\arg\max} \sum_{G \in R} \sum_{z} p^{(j)}(z \mid G) \log p_Z(z), \\
p_{\text{size}}^{(j)} &= \underset{p_{\text{size}}}{\arg\max} \sum_{G \in R} \sum_{z} p^{(j)}(z \mid G) \log p_{\text{size}}(|G| \mid z), \\
p_{\text{tuple}}^{(j)} &= \underset{p_{\text{tuple}}}{\arg\max} \sum_{G \in R} \sum_{z} \sum_{t \in G} p^{(j)}(z \mid G) \log p_{\text{tuple}}(t \mid z; \theta).
\end{aligned} \tag{12}$$

令 N 表示 R中的组数。根据吉布斯不等式，有：

$$\begin{aligned}
p_Z^{(j)}(z) &= \frac{1}{N} \sum_{G \in R} p^{(j)}(z \mid G), \\
p_{\text{size}}^{(j)}(s \mid z) &= \frac{\sum_{G \in R} p^{(j)}(z \mid G) \mathbb{I}(|G| = s)}{\sum_{G \in R} p^{(j)}(z \mid G)}.
\end{aligned}$$

我们首先求解公式 (12)。根据公式 (1)，我们可以将 p元组$(t, z; \theta)$ 重写为：

$$p_{\text{tuple}}(t, z; \theta) = \frac{1}{A(\theta)} \prod_{c \in M \in \mathcal{M} \wedge c \leftrightarrow (t,z)} \exp(\theta[c]), \tag{13}$$

其中 $A(\theta)$ 是一个常数，使得$\sum_{t,\ z} p$元组$(t,\ z; \theta) = 1$。类似地，p元组$(t \mid z; \theta)$ 可以重写为：

$$p_{\text{tuple}}(t \mid z; \theta) = \frac{1}{A'(\theta, z)} \prod_{c \in M \in \mathcal{M} \wedge c \leftrightarrow (t,z)} \exp(\theta[c]), \tag{14}$$

其中 $A'(\theta,\ z)$ 也是一个常数，使得 $\sum_t p$元组$(t \mid z; \theta) = 1$ 对于任意 z。

注意，当结构 M 固定时，p元组由 θ确定。因此，式（12）等价于：

$$\theta^{(j)} = \underset{\theta}{\arg\max} \sum_{G \in R} \sum_{z} \sum_{t \in G} p^{(j)}(z \mid G) \log p_{\text{tuple}}(t \mid z; \theta). \tag{15}$$

为求解此方程，我们有以下引理。（证明见附录 B。）

引理 A.1。令 M_Z 表示潜变量的集合。给定每个组的某个潜变量分布 $p(z|G)$ 以及一个边缘集 $\mathcal{M}$，使得存在 $M \in \mathcal{M}$ 和 $M_Z \subseteq M$，令：

$$L_1(\theta) = \sum_{G\in R}\sum_{z}\sum_{t\in G} p(z \mid G)\log p_{tuple}(t \mid z;\theta), \tag{16}$$

$$L_2(\theta) = \sum_{G\in R}\sum_{z}\sum_{t\in G} p(z \mid G)\log p_{tuple}(t,z;\theta), \tag{17}$$

$$\theta^* = \arg\max_{\theta} L_2. \tag{18}$$

然后，θ^* 是$\arg\max_\theta L_1$的解。

在只有一个潜变量的情况下，M_Z 是任意潜在边缘分布的子集。在有多个潜变量的情况下，我们要求 M 在第 4.6 节中包含 M_Z。因此，我们可以应用此引理来求解公式（15）。令：

$$L^{(j)}(\theta) = \sum_{G\in R}\sum_{z}\sum_{t\in G} p^{(j)}(z \mid G)\log p_{\text{tuple}}(t,z;\theta).$$

我们有：

$$\begin{aligned}\theta^{(j)} &= \arg\max_{\theta} L^{(j)}(\theta)\\ &= \arg\max_{\theta} \sum_{G\in R}\sum_{z}\sum_{t\in G} p^{(j)}(z \mid G)\log p_{\text{tuple}}(t,z;\theta).\end{aligned} \tag{19}$$

事实证明，这是一个凸优化问题，如下面的定理所示。

定理 A.2. 给定每个组的某个潜变量分布 $p(z \mid G)$ G，令：

$$L(\theta) = \sum_{G\in R}\sum_{z}\sum_{t\in G} p(z \mid G)\log p_{tuple}(t,z;\theta). \tag{20}$$

那么，L 是凹函数，且其关于每个计数 c 的偏导数为：

$$\frac{\partial L}{\partial \theta[c]} = \sum_{G\in R}\sum_{t\in G,z:\, c\leftrightarrow(t,z)} p^{(j)}(z \mid G) - n\sum_{t,z:\, c\leftrightarrow(t,z)} p_{tuple}(t,z;\theta). \tag{21}$$

证明。由式 (13) 和 (20) 可得，

$$\begin{aligned}L(\theta) &= \sum_{G\in R}\sum_{z}\sum_{t\in G} p^{(j)}(z \mid G)\left(\sum_{c\in M\in\mathcal{M}\wedge c\leftrightarrow(t,z)}\theta[c] - \log A(\theta)\right)\\ &= \sum_{G\in R}\sum_{z}\sum_{t\in G} p^{(j)}(z \mid G)\sum_{c\in M\in\mathcal{M}\wedge c\leftrightarrow(t,z)}\theta[c] - n\log A(\theta),\end{aligned}$$

其中，n 是 R中的元组数。由于 $p^{(j)}(z \mid G)$ 均为固定值，第一项是线性函数。根据线性性，它也是凹函数。此外，图模型领域的文献中已知，$\log A(\theta)$ 关于 θ是凸的。因此，$L(\theta)$是凹的。对于每个计数 c，$\theta[c]$ 的偏导数为：

$$\frac{\partial L}{\partial \theta[c]} = \sum_{G\in R}\sum_{t\in G,z:\, c\leftrightarrow(t,z)} p^{(j)}(z \mid G) - n\sum_{t,z:\, c\leftrightarrow(t,z)} p_{\text{tuple}}(t,z;\theta),$$

其中，第一项由线性性得到，第二项则由性质

$$\frac{\partial}{\partial\theta[c]}\log A(\theta) = \frac{\sum_{t,z:\, c\leftrightarrow(t,z)} p_{\text{tuple}}(t,z;\theta)}{\sum_{t,z} p_{\text{tuple}}(t,z;\theta)}$$

给出。□

定理 A.2 表明，式 (19) 是一个凸优化问题，因此我们可以应用任意梯度上升方法来估计 $\theta^{(j)}$。特别地，导数式 (21) 的第一项是关系的计数值 c，第二项是 p元组的边缘分布计数。我们可以使用任意边缘分布推断算法（例如连接树算法）计算第二项。

此外，由于该导数是数据边缘分布计数与模型边缘分布计数之差，我们的梯度方法与 PGM 的算法1 [34] 相同，只是将边缘分布的概念扩展到包含潜在边缘分布。这解释了算法1的第 23 行。

B 引理 A.1 的证明

根据式 (13) 和 (17)，

$$\begin{aligned}L_2(\theta) &= \sum_{G\in R}\sum_{z}\sum_{t\in G} p(z \mid G)\left(\sum_{c\in M\in\mathcal{M}\wedge c\leftrightarrow(t,z)}\theta[c] - \log A(\theta)\right)\\ &= \sum_{G\in R}\sum_{z}\sum_{t\in G} p(z \mid G)\sum_{c\in M\in\mathcal{M}\wedge c\leftrightarrow(t,z)}\theta[c] - n\log A(\theta),\end{aligned}$$

其中，n是 R中的元组数量。第一项是线性函数，因为 $p(z \mid G)$都是固定值。根据线性性，该项也是凹函数。此外，在图模型相关文献中，众所周知，$\log A(\theta)$关于 θ是凸函数。因此，L_2是凹函数，且在 θ^*处，偏导数 $\partial L_2/\partial\theta[c]$均为零。对于每个参数 $\theta[c]$，其偏导数为：

$$\frac{\partial L_2}{\partial\theta[c]} = \sum_{G\in R}\sum_{t\in G,z:\, c\leftrightarrow(t,z)} p(z \mid G) - n\sum_{t,z:\, c\leftrightarrow(t,z)} p_{\text{tuple}}(t,z;\theta),$$

其中，第一项源于线性性，第二项则由性质

$$\frac{\partial}{\partial\theta[c]}\log A(\theta) = \frac{\sum_{t,z:\, c\leftrightarrow(t,z)} p_{\text{tuple}}(t,z;\theta)}{\sum_{t,z} p_{\text{tuple}}(t,z;\theta)}$$

得出。于是，我们有：

$$\begin{aligned}\frac{\partial}{\partial\theta[c]}L_2(\theta^*) &= 0,\\ \sum_{G\in R}\sum_{t\in G,z:\, c\leftrightarrow(t,z)} p(z \mid G) - n\sum_{t,z:\, c\leftrightarrow(t,z)} p_{\text{tuple}}(t,z;\theta^*) &= 0.\end{aligned} \tag{22}$$

这意味着，M 上该关系的期望计数等于 $M \in$ 处 p元组的边缘分布计数。特别地，对于每个 $c \in M_Z$，这一结论也成立，因为 M 包含边缘分布 M_Z。对于每个可能的 z，它都恰好对应一个 $c \in M_Z$，因此式 (22) 可简化为：

$$\begin{aligned}\sum_{G\in R}|G|p(z \mid G) &= n\sum_{t} p_{\text{tuple}}(t,z;\theta^*),\\ \sum_{G\in R}|G|p(z \mid G) &= np_{\text{tuple}}(z;\theta^*).\end{aligned} \tag{23}$$

也就是说，M_Z 关系的期望计数等于 M_Z上p元组的边缘分布计数。

现在，我们来考察 $\arg\max_\theta L_1$ 的解。我们有：

$$L_1 = \sum_{G\in R}\sum_{z}\sum_{t\in G} p(z \mid G)\log p_{\text{tuple}}(t \mid z;\theta).$$

根据公式 (14)，我们有：

$$L_1 = \sum_{G\in R}\sum_{z}\sum_{t\in G} p(z \mid G)\left(\sum_{c\in M\in\mathcal{M}\wedge c\leftrightarrow(t,z)} \theta[c] - \log A'(\theta,z)\right)$$
$$= \sum_{G\in R}\sum_{z}\sum_{t\in G} p(z \mid G)\sum_{c\in M\in\mathcal{M}\wedge c\leftrightarrow(t,z)} \theta[c] - \sum_{G\in R}\sum_{z}\sum_{t\in G} p(z \mid G)\log A'(\theta,z). \quad (24)$$

与 $\log A(\theta)$ 类似，对数归一化因子 $\log A'(\theta,z)$ 关于 θ 是凸的，而式 (24) 的第一项是线性的。因此，L_1 是凹函数。要证明 θ^* 是 $\arg\max_\theta L_1$ 的解，只需证明对于所有 c，当 θ^* 时，$\partial L_1/\partial\theta[c] = 0$。利用 $\partial A'(\theta,z)/\partial\theta[c] = \sum_{t':c\leftrightarrow(t',z)} p$ 元组$(t' \mid z;\theta)$，我们有：

$$\frac{\partial L_1}{\partial\theta[c]} = \sum_{G\in R}\sum_{t\in G,z:\,c\leftrightarrow(t,z)} p(z \mid G) - \sum_{G\in R}\sum_{z}\sum_{t\in G} p(z \mid G)\sum_{t':c\leftrightarrow(t',z)} p_{\text{tuple}}(t' \mid z;\theta)$$
$$= \sum_{G\in R}\sum_{t\in G,z:\,c\leftrightarrow(t,z)} p(z \mid G) - \sum_{z}\left(\sum_{G\in R}\sum_{t\in G} p(z \mid G)\right)\left(\sum_{t':c\leftrightarrow(t',z)} p_{\text{tuple}}(t' \mid z;\theta)\right)$$
$$= \sum_{G\in R}\sum_{t\in G,z:\,c\leftrightarrow(t,z)} p(z \mid G) - \sum_{z}\left(\sum_{G\in R}|G|p(z \mid G)\right)\left(\sum_{t':c\leftrightarrow(t',z)} p_{\text{tuple}}(t' \mid z;\theta)\right). \quad (25)$$

根据公式（23），我们有：

$$\frac{\partial}{\partial\theta[c]}L_1(\theta^*) = \sum_{G\in R}\sum_{t\in G,z:\,c\leftrightarrow(t,z)} p(z \mid G) - \sum_{z} np_{\text{tuple}}(z;\theta^*)\left(\sum_{t':c\leftrightarrow(t',z)} p_{\text{tuple}}(t' \mid z;\theta^*)\right)$$
$$= \sum_{G\in R}\sum_{t\in G,z:\,c\leftrightarrow(t,z)} p(z \mid G) - \sum_{z}\sum_{t':c\leftrightarrow(t',z)} np_{\text{tuple}}(t' \mid z;\theta^*)p_{\text{tuple}}(z;\theta^*)$$
$$= \sum_{G\in R}\sum_{t\in G,z:\,c\leftrightarrow(t,z)} p(z \mid G) - \sum_{z}\sum_{t':c\leftrightarrow(t',z)} np_{\text{tuple}}(t',z;\theta^*)$$
$$= \sum_{G\in R}\sum_{t\in G,z:\,c\leftrightarrow(t,z)} p(z \mid G) - n\sum_{t',z:\,c\leftrightarrow(t',z)} p_{\text{tuple}}(t',z;\theta^*).$$

由式 (22)，对于每个 c，我们有：

$$\frac{\partial}{\partial\theta[c]}L_1(\theta^*) = 0. \quad (26)$$

因此，θ^* 也是 $\arg\max_\theta L_1$ 的一个解。

C 隐私消耗的界定

在本节中，我们量化算法1和算法2的隐私消耗，以及 [9]中算法6的隐私消耗。然后，算法 3 的隐私消耗是其子程序隐私消耗之和。

C.1 引理6.4的证明

算法1在私人数据上有以下查询：

(1) 它在第10行的每个EM步骤中查询cnt*(z)。cnt*(z)是由组构成的列联表，其敏感度为 μ_g。cnt*(z) 的查询总数为 T。
(2) 它在第16行的每个EM步骤中查询 cnt*(s, z)。cnt*(s, z) 是由组构成的列联表，其敏感度为 μ_g。cnt*(s, z)的查询数量为 T。(3) 它在第19-22行查询潜在边缘分布。潜在边缘分布是一个列联表，每个组的贡献最多为 τ。因此，其敏感度为 $\tau\mu_g$。由于我们执行 EM T 步，并且有 $m_{\mathcal{M}}$ 个潜在边缘分布，因此潜在边缘分布的查询总数为 $m_{\mathcal{M}}T$。

因此，算法1的隐私消耗为：

$$C_1(T, m_{\mathcal{M}}) = T\cdot\mu_g^2\cdot\left(\frac{m_{\mathcal{M}}\cdot\tau^2}{\sigma_l^2} + \frac{1}{\sigma_{size}^2} + \frac{1}{\sigma_z^2}\right).$$

C.2 推论 6.5 的证明

当元组乘数为1时， [9] 中的引理2表明，其算法6的隐私消耗为：

$$g = \frac{1}{\sigma_{\mathcal{U}}^2} + \frac{2d(d-1)}{\sigma_R^2} + \frac{t \cdot k}{\sigma_h^2} + \frac{(d+t)}{\sigma_o^2}. \tag{27}$$

注意，其隐私消耗与敏感度的平方成正比，也就是说，与元组乘数的平方成正比。由于我们的元组乘数至多为 μ_t，因此我们算法中算法 6 [9] 的各个实例的隐私消耗为 $C\text{single}(d) = \mu_t^2 g$。

C.3 引理 6.6 的证明

算法 2 的隐私消耗如下：

(1) 在第 1 行，它调用了 [9] 的算法6，其隐私消耗为 C单个(d)。(2) 在第 7 行，当 M 仅包含初始潜在边缘分布时，它使用算法1来估计参数。由于我们具有两个潜变量，如第 4.6 节所述，初始潜在边缘分布的数量为 $2d$，隐私消耗为 $C_1(T, 2d)$

(3) 在第16行，选择新的潜在边缘分布后，使用算法1估计参数。隐私消耗为 $\sum_{i=1}^{T_C} C_1(1, 2d + i \cdot n\text{inc})$。

(4) 在第13行，数学个 $\widetilde{C} \cdot \text{err}(M)$ 进行查询 M' 的 C 查询，总数为 $n_C \cdot T_C$。由于 M' 是元组的私有列联表，而 M' 是公开可用的，因此， $\widetilde{\text{err}}(M') = \|M' - \widetilde{M'}\|_1$ 的敏感度为 μ_t。于是，隐私消耗为 $\mu_t^2 \cdot \frac{n_C \cdot T_C}{\sigma_{\text{err}}^2}$。

因此，总隐私消耗为：

$$\begin{aligned} C_2(\text{FK}(R, R')) &= C_{\text{single}}(d) + C_1(T, 2d) \\ &\quad + \sum_{i=1}^{T_C} C_1(1, 2d + i \cdot n_{\text{inc}}) + \mu_t^2 \cdot \frac{n_C \cdot T_C}{\sigma_{\text{err}}^2}. \end{aligned} \tag{28}$$