面向 SQL 查询的实用差分隐私保护

葛云清

摘要

差分隐私承诺在保护个人隐私的同时可以实现一般的数据分析，但现有的差分隐私机制并不支持现实中基于 SQL 的分析系统中使用的各种特征和数据库。本文首次提出了面向实际应用场景的 SQL 查询差分隐私保护方法。为了达到这一目标，文章提出了弹性敏感度，一种用于近似一般等连接查询局部敏感度的全新方法。本文证明了弹性敏感度是局部敏感度的上界，因此可以应用于任何基于局部敏感度的差分隐私机制。最后，文章构建了一个实用的端到端系统，使用弹性敏感度对 SQL 查询进行差分隐私化处理。

关键词：差分隐私；SQL 查询；局部敏感度；隐私保护

1 引言

各类机构在越来越广泛地收集关于个人的敏感信息。虽然这些组织在道德和法律上都有义务保护隐私安全，然而机构内的数据分析人员已经开始不受限制地获取数据以获得最大的生产力，而这势必会导致隐私泄露问题的加剧。这类访问通常在支持 SQL 查询的关系数据库中发生。现有的数据安全和隐私保护方法已无法在为分析人员提供通用访问权限的同时保证个人的隐私安全。

正如近年发生的内部攻击[1-2]那样，机构成员被允许不受限制地访问数据是隐私泄露的主要原因。传统的访问控制策略可以限制对特定数据库的访问，但是一旦分析人员拥有访问权限，这些策略就无法控制数据的使用方式。数据匿名化策略试图在提供隐私的同时允许通用分析，但并不可靠，如一些重识别攻击[3]所示。

差分隐私[4-5]是解决这些问题的一种很有前途的技术。差分隐私允许对数据进行一般的统计分析，同时保护关于个人的数据，具有很强的隐私形式保证。

由于其可取的正式保证，差分隐私受到了越来越多包括谷歌和苹果在内的机构的关注。然而，差分隐私实用技术的研究主要集中在特殊用途的用例上，例如收集关于网页浏览行为的统计数据[6]和键盘和表情符号的使用情况[7]，而面向通用数据分析的差分隐私仍然是一个开放的挑战。

2 相关工作

多种机制[8-11]为类 SQL 查询的某些子集提供了差分隐私，但在实际应用中并不支持大多数查询。这些机制还需要对数据库引擎进行修改，使其在实际中的应用变得十分复杂。

此外，尽管差分隐私的理论方面已经得到了广泛的研究，但关于差分隐私对现实查询的定量影响却知之甚少。最近的工作已经评估了实际数据上的差分隐私机制，然而这项工作使用有限的查询来表示单一的、特殊用途的分析任务，如直方图分析或范围查询。据我们所知，目前尚未有工作探索面向一般现实查询的差分隐私技术的设计和评估。

现有的几种通用差分隐私机制支持带有 join 的查询。图 1总结了这些机制及其支持的特性，并与我们提出的机制进行了比较 (最后一行)。

2.1 隐私集成查询

2.1.1 PINQ

隐私集成查询 (Privacy Integrated Queries，PINQ)[9]是一种为用增强型 SQL 语言编写的查询提供差分隐私的机制。PINQ 支持一个受限的连接算子，用同一个密钥将结果分组。对于一对一连接，该算子等价于标准语义。另一方面，对于一对多和多对多连接，PINQ 查询可以统计唯一连接键的个数，但不能统计连接结果的个数。此外，PINQ 引入了标准 SQL 中不存在的新操作符，因此该方法与标准数据库不兼容。

2.1.2 wPINQ

加权 PINQ (Weighted PINQ，wPINQ)[11]扩展了支持一般等分连接的 PINQ，通过为数据库中的每一行分配一个权重，然后缩小连接中行的权重来保证整体灵敏度为 1。在 wPINQ 中，计数查询的结果是被计数记录的权重加上由拉普拉斯分布得出的噪声之和。该方法允许 wPINQ 支持所有三种类型的连接。然而，wPINQ 并不能满足我们的数据库兼容性要求。普罗塞尔皮奥等[11]描述的系统使用自定义的运行时；在现有数据库中应用 wPINQ 需要修改数据库以在执行过程中传播权重。

2.2 受限敏感度

受限敏感性[8]旨在通过使用辅助数据模型的属性，将计数查询的全局敏感性与连接绑定在一起。该方法要求在全局上限定每个连接键的频率 (即对于所有可能的未来数据库)。这对于一对一和一对多的连接很有效，因为连接的” 一” 边上的唯一密钥具有全局界。但是，它不能处理多对多连接，因为连接两侧键的频率可能是无界的。Blocki 等人[8]对受限敏感性方法进行了形式化，但没有描述如何在与现有数据库兼容的系统中使用它，并且没有提供可用的实现。

2.3 DJoin

DJoin[12]是一种针对分布在多方的数据集进行差分隐私查询的机制。由于与此设置相关的额外限制，DJoin 只支持一对一连接，因为它将连接查询重写为关系交叉点。例如，考虑如下查询：

SELECT COUNT (*) FROM X JOIN Y ON $X.A = Y.B$.

DJoin 将该查询改写为以下 (在关系代数中)，只有当连接为一对一连接时，该查询才与原始查询语义等价：$|\pi_A(X) \cap \pi_B(Y)|$.

此外，该方法在查询执行过程中需要使用特殊的加密函数，因此与现有数据库不兼容。为了解决现有机制的局限性，我们提出弹性敏感性，下面进行讨论。弹性敏感性能够兼容任意已有的数据库，并且支持具有全谱连接关系的通用等值连接。这种结构允许弹性灵敏度能够在现实场景中使用。

	Database compatibility	One-to-one equijoin	One-to-many equijoin	Many-to-many equijoin
PINQ [41]		✓		
wPINQ [48]		✓	✓	✓
Restricted sensitivity [14]		✓	✓	
DJoin [43]		✓		
Elastic sensitivity (this work)	✓	✓	✓	✓

图 1: 支持等值连接的通用差分隐私机制的比较。

3 本文方法

3.1 差分隐私背景

这里简要总结了描述本文的方法所需的现有差分隐私概念。为了更透彻地概述差分隐私，我们参考了 Dwork 和 Roth 的优秀工作[5].

差分隐私提供了不可区分性的形式化保证：差分隐私结果不会产生关于两个相邻数据库中的哪一个被用于计算结果的大量信息。

从形式上讲，差分隐私考虑一个建模为向量 $x \in D^n$ 的数据库，其中 x_i 表示用户 i 提供的数据。两个数据库 $x, y \in D^n$ 之间的距离为 $d(x,y) = |\{i \mid x_i \neq y_i\}|$。若 $d(x,y) = 1$，则两个数据库 x，y 是相邻的。

定义 1 (差分隐私)。任意机制 K：$\mathcal{K} : D^n \to \mathbb{R}^d$ 满足 (ε, δ)-差分隐私，如果对于任意一对数据库 $x, y \in D^n$，使得 $d(x,y) = 1$，并且对于所有可能输出的集合 S 有：

$$\Pr[\mathcal{K}(x) \in S] \leqslant e^{\varepsilon} \Pr[\mathcal{K}(y) \in S] + \delta$$

直觉上，查询的敏感度对应于当数据库发生变化时其结果可以改变的量。敏感性的一个度量是全局敏感性，即查询结果在任意两个相邻数据库上的最大差异。

定义 2 (全局敏感度)。对于 $f : D^n \to \mathbb{R}^d$ 和所有 $x, y \in D^n$，f 的全局敏感度为：

$$GS_f = \max_{x,y:d(x,y)=1} \|f(x) - f(y)\|$$

麦克谢里[9]定义了数据库上稳定变换的概念，我们将在后面使用。直觉上，一个转换是稳定的，如果它的隐私影响是可以被限制的。

敏感度的另一个定义是局部敏感度[13]，它是查询在真实数据库上的结果与它的任何邻居之间的最大差值：

定义 3 (局部敏感度)。对于 $f : D^n \to \mathbb{R}^d$ 且 $x \in D^n$，f 在 x 处的局部敏感度为：

$$LS_f(x) = \max_{y:d(x,y)=1} \|f(x) - f(y)\|$$

局部敏感性往往远低于全局敏感性，因为它是单个真实数据库的属性，而不是所有可能数据库的集合。

多表数据库的差分隐私保护。本文考虑有界差分隐私，其中 x 可以通过改变 (但不添加或去除) 为单个元组从其邻居 y 中获得。我们的设置涉及一个以元组的多重集表示的数据库，我们希望保护单个元组的存在或不存在。如果从域 D 中抽取元组，数据库中包含 n 个元组，则该集合可以表示为向量 $x \in D^n$，其中如果数据库中第 i 行包含元组 v，则 $x_i = v$。

通过这种映射，差分隐私提供了与单表情况相同的保护：它保护数据库中任何一个元组的存在或不存在。然而，当单个用户贡献了多个受保护元组时，保护单个元组可能不足以提供隐私。

3.2 弹性敏感度

文章在查询的结构上递归地定义查询的弹性敏感度。为了允许使用平滑函数，我们的定义描述了如何计算距离真实数据库 (在此定义下, 查询的局部敏感度定义在 $k = 0$ 处) 任意距离 k 处的弹性敏感度。图 2 包含了完整的定义，分为四个部分：(a) 核心关系代数，(b) 弹性灵敏度的定义，(c) 距离 k 处的最大频率，(d) 关系的祖先。接下来将对每一部分进行描述。

3.2.1 核心关系代数

本文以标准关系代数的子集给出弹性灵敏度的形式化定义，定义如图 2(a) 所示。这个子集包括选择 (σ)、投影 (π)、连接 ($\bowtie$)、计数 (Count) 和分组计数 (Count $_{G_1..G_n}$)。它承认任意等值连接，包括自连接，以及所有连接关系 (一对一、一对多、多对多)。

为了简化表示，文章的表示法假设查询执行一个计数作为最外层的操作，但是该方法自然地扩展到嵌套在查询中任何地方的聚合，只要查询不执行算术运算或对聚合结果进行其他修改。例如，下面的查询统计出行总次数，并投影” 计数” 属性：

$$\pi_{\text{count}}\ \text{Count(trips)}$$

本文的方法可以通过将内部关系作为查询根来支持该查询。

3.2.2 弹性敏感性

图 2(b) 给出了距离 k 处弹性灵敏度的递推定义。本文将查询 q 在距离真实数据库 x 为 k 处的弹性敏感度记为 $\hat{S}^{(k)}(q, x)$. $\hat{S}$ 函数定义为关系变换 (记为 $\hat{S}_R$) 的弹性稳定性.

$\hat{S}_R^{(k)}(r, x)$ 定义为关系 r 在距离数据库 x 在 k 处的局部稳定性，$\hat{S}_R^{(k)}(r, x)$ 定义为属性 a 在距离数据库 x,k 处的最大频率 $\text{mf}_k(a, r, x)$。

3.2.3 距离 k 处的最大频率

最大频率度量用于约束连接的敏感性。本文定义最大频率 $\text{mf}(a, r, x)$ 为数据库实例 x 中关系 r 中属性 a 的最频繁值的频率。若要在距离真实数据库 k 处绑定查询的本地敏感性，还必须在距离真实数据库 k 处绑定每个连接键的最大频率。对于真实数据库 x 中关系 r 的属性 a，用于表示这个值 $\text{mf}(a, r, x)$，在图 2(c) 中给出定义。

3.2.4 关系的起源

图 2(d) 中的定义是识别自连接的形式化。自我联接比非重叠关系联接对敏感度的影响大得多。在自连接中，增加或删除底层数据库中的一行可能会引起两个连接关系的变化，而不仅仅是一个或另一个。因此，弹性敏感性的连接情况定义为两种情况：一种为自连接，一种为非重叠关系连接。为了区分这两种情况，文章使用 $\mathcal{A}(r)$ (定义如图 2(d)))，它表示可能为 r 贡献行的表的集合。两个关系 r_1 和 r_2 的连接是当 r_1 和 r_2 重叠时的自连接，当底层数据库中的某个表 t 同时为 r_1 和 r_2 贡献行时发生。当 $|\mathcal{A}(r_1) \cap \mathcal{A}(r_2)| = 0$ 时，行 r_1 和 r_2 是非重叠的。

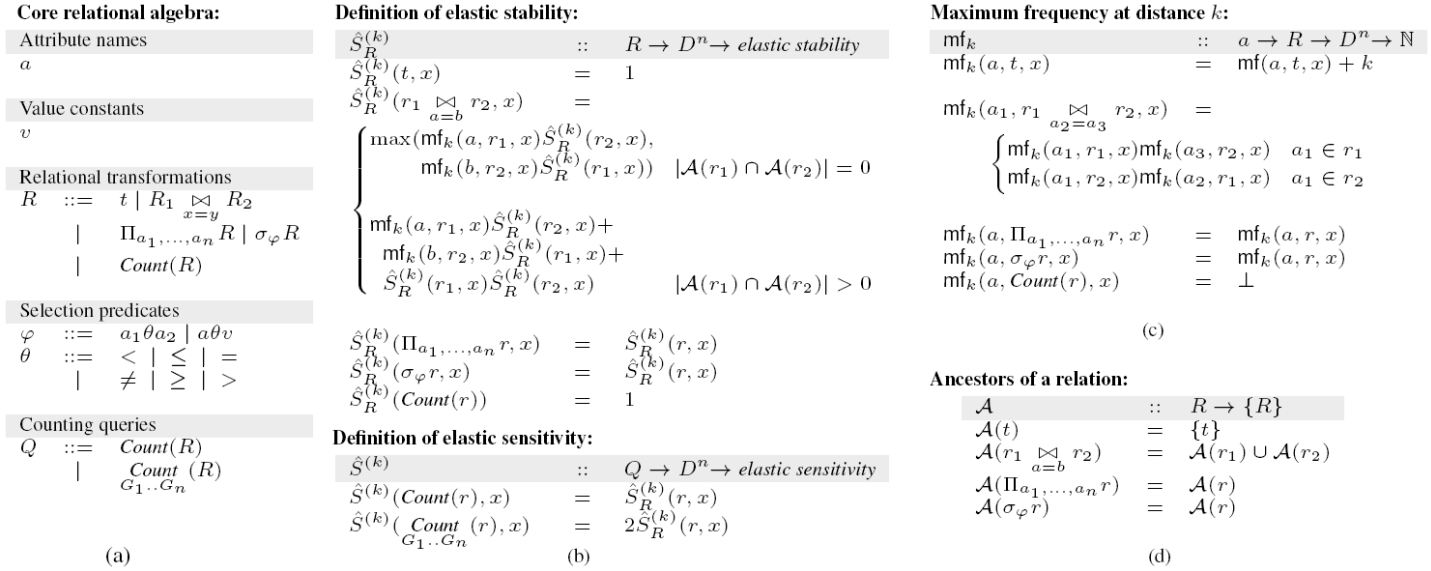

图 2: (a) 核心关系代数，(b) 弹性灵敏度的定义，(c) 距离 k 处的最大频率，(d) 关系的起源。

4 复现细节

4.1 实验环境搭建

构建与运行这个框架是用 Scala 编写并使用 Maven 构建的。代码已经在 Mac OS X 上进行了测试。要构建代码：

$ mvn package

4.2 使用说明

本文的工作重点在于为通用的 SQL 查询实施差分隐私机制，因此界面并非文章的主要工作之一。实现在终端即可进行，以下的实验结果也都是在终端直接运行得到。

代码文件夹中已包含几个编写完成的运行示例。文件 examples/QueryRewritingExample.scala 包含用于查询重写的示例代码，并使用几个简单的查询演示了支持的机制。要运行此示例：

$mvn exec:java -Dexec.mainClass=”examples.QueryRewritingExample”

类似地，还可以运行：

$mvn exec:java -Dexec.mainClass=”examples.ElasticSensitivityExample”

和

$mvn exec:java -Dexec.mainClass=”examples.MechanismExample”

分别得到弹性敏感度的计算结果和机制的测试结果。

4.3 创新点

4.3.1 弹性敏感度的复现

首先，我对这篇文章的核心创新点弹性敏感度进行的学习和理解，梳理其逻辑后进行了验证复现工作。详细运行结果可参看本文第 5 节实验结果分析部分。

4.3.2 差分隐私机制的优化与改进

参考如图 3 的全局系统架构，我注意到原文采用的方法是最基础的 Laplace 差分隐私机制，这是一种自差分隐私问世以来就有的经典方案，我对其进行了扩展，增加了更新的差分隐私机制，增强了这一架构的可扩展性。

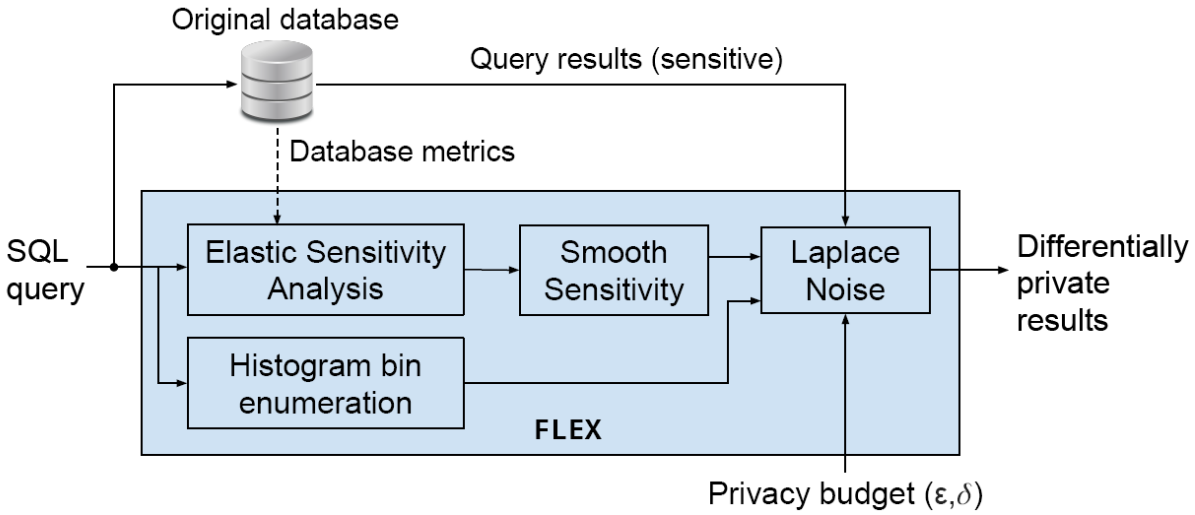

图 3: 系统框架

5 实验结果分析

5.1 弹性敏感度的复现

原代码测试结果如图 3 所示，而我得到的实验结果如图 4，可以看到，误差范围相差无几，说明本文的弹性敏感度也基本达到了原文提出的敏感度要求。

```
Query:
    SELECT COUNT(*) FROM orders
    JOIN customers ON orders.customer_id = customers.customer_id
    WHERE orders.product_id = 1 AND customers.address LIKE '%United States%'

Private result: 100000

Noisy result (run 1): 94628
Noisy result (run 2): 103864
Noisy result (run 3): 105402
Noisy result (run 4): 113289
Noisy result (run 5): 101134
Noisy result (run 6): 100177
Noisy result (run 7): 99473
Noisy result (run 8): 99538
Noisy result (run 9): 106394
Noisy result (run 10): 95710
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.815 s
[INFO] Finished at: 2022-11-30T10:52:32+08:00
```

图 4: 原弹性敏感度应用结果

```
Query:
    SELECT COUNT(*) FROM orders
    JOIN customers ON orders.customer_id = customers.customer_id
    WHERE orders.product_id = 1 AND customers.address LIKE '%United States%'

Private result: 100000

Noisy result (run 1): 98999
Noisy result (run 2): 102378
Noisy result (run 3): 89760
Noisy result (run 4): 102593
Noisy result (run 5): 104211
Noisy result (run 6): 114310
Noisy result (run 7): 102694
Noisy result (run 8): 103720
Noisy result (run 9): 94836
Noisy result (run 10): 99780
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.714 s
[INFO] Finished at: 2022-12-06T22:31:36+08:00
```

图 5: 复现弹性敏感度应用结果

5.2 差分隐私机制的优化与改进

更换差分隐私机制后，我修改了相应代码并进行了实验验证测试，如图 6，新的结果能够对原本的 SQL 查询语句进行隐私化的改写，达到实验预期。

```
*** Elastic sensitivity example ***
Original query:

  SELECT COUNT(*) FROM orders
  JOIN customers ON orders.customer_id = customers.customer_id
  WHERE orders.product_id = 1 AND customers.address LIKE '%United States%'

> Epsilon: 0.1
> Elastic sensitivity of this query: 215.4674322436261
> Required scale of Laplace noise: 2 * 215.4674322436261 / 0.1 = 4309.348644872522

Rewritten query:

  SELECT COUNT(*) + 4309.348644872522 * (CASE WHEN RAND() - 0.5 < 0 THEN -1.0 ELSE 1.0 END * LN(1 - 2 * ABS(RAND() - 0.5)))
  FROM (SELECT customer_id, product_id
  FROM public.orders) t
  INNER JOIN (SELECT customer_id, address
  FROM public.customers) t0 ON t.customer_id = t0.customer_id
  WHERE t.product_id = 1 AND t0.address LIKE '%United States%'
```

图 6: SQL 查询语句的改写

此外，我还参考原文进行了 TPC-H 测试，这是一套通用的 SQL 查询测试。如图 7，可以看到，随着查询范围的增加，所需的误差显著降低，即需要添加的差分隐私扰动与查询范围是负相关的，这也满足原文的预期实验结果。

```
jean@geyunqingdeMacBook-Pro elastic-sensitivity-experiments-master % sbt run
[info] welcome to sbt 1.8.0 (Homebrew Java 19.0.1)
[info] loading project definition from /Users/jean/IdeaProjects/elastic-sensitivity-experiments-master/
[info] loading settings for project elastic-sensitivity-experiments-master from build.sbt ...
[info] set current project to elastic-sensitivity-experiments (in build file:/Users/jean/IdeaProjects/e
[warn] there's a key that's not used by any other settings/tasks:
[warn]
[warn] * elastic-sensitivity-experiments-master / mainClass
[warn]   +- /Users/jean/IdeaProjects/elastic-sensitivity-experiments-master/build.sbt:7
[warn]
[warn] note: a setting might still be used by a command; to exclude a key from this `lintUnused` check
[warn] either append it to `Global / excludeLintKeys` or call .withRank(KeyRanks.Invisible) on the key
[info] compiling 1 Scala source to /Users/jean/IdeaProjects/elastic-sensitivity-experiments-master/targ
[info] running Experiment
Query    Coverage          Error
q4.sql   10487             0.416712
q1.sql   1478682           0.002653
q21.sql  10                125343813.308134
q13.sql  2017              902.642371
q16.sql  4                 220329.415335
[success] Total time: 8 s, completed 2022年12月7日 12:56:19
```

Table 3: Evaluated TPC-H queries.

Query	Description	# Joins
Q1	Billed, shipped, and returned business	0
Q4	Priority system status and customer satisfaction	0
Q13	Relationship between customers and order size	1
Q16	Suppliers capable of supplying various part types	1
Q21	Suppliers with late shipping times for required parts	3

图 7: TPC-H 查询测试

6 总结与展望

本文首先介绍了在现代数据库管理系统中存在的隐私泄露问题并未提供了较为可靠的解决办法，差分隐私机制。其次，分析了以往将差分隐私技术应用于数据库中所面临的困难与阻碍，以及已有技术的不足。接着，本文介绍了弹性敏感度这一有效的局部敏感度优化方案，它利用等值连接中的频繁度这一概念提供了局部敏感度的限制，能够有效的提升计算效率，降低出错概率，并能够很好的兼容已有的各种数据库管理系统，实现通用性。

此外，我对于原文的核心创新点弹性敏感度进行了复现工作，并在此基础上对系统框架进行了扩展。

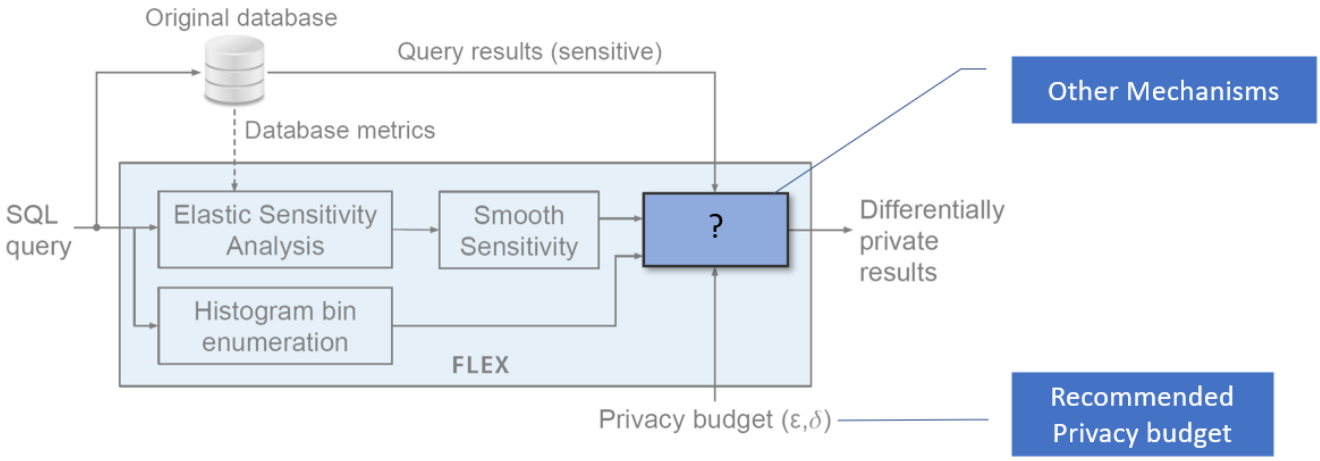

图 8: 对于其他机制的拓展

在我看来，这篇工作的可扩展之处还有很多，比如将其应用至具体的类似 PostgreSQL 等 DBMS 中，或是继续我的第二项创新工作，增加更多的差分隐私机制（如图 8），这些工作就有待我后期继续进行了。

参考文献

[1] PHYSICS H E. Theory collaboration network[Z]. https://snap.stanford.edu/data/ca-HepTh.html.

[2] A REMINDER OF INSIDER RISKS M S B. TP Toolbox[Z]. https://securityintelligence.com/news/morgan-stanley-breach-reminder-insider-risks/.

[3] DE MONTJOYE Y A, HIDALGO C A, VERLEYSEN M, et al. Unique in the crowd: The privacy bounds of human mobility[J]. Scientific reports, 2013, 3(1): 1-5.

[4] DWORK C. Differential privacy[C]//Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33. 2006: 1-12.

[5] DWORK C, ROTH A, et al. The algorithmic foundations of differential privacy[J]. Foundations and Trends® in Theoretical Computer Science, 2014, 9(3–4): 211-407.

[6] ERLINGSSON Ú, PIHUR V, KOROLOVA A. Rappor: Randomized aggregatable privacy-preserving ordinal response[C]//Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. 2014: 1054-1067.

[7] Apple. Apple previews iOS 10, the biggest iOS release ever[Z]. http://www.apple.com/newsroom/2016/06/apple-previews-ios-10-biggest-ios-release-ever.html.

[8] BLOCKI J, BLUM A, DATTA A, et al. Differentially private data analysis of social networks via restricted sensitivity[C]//Proceedings of the 4th conference on Innovations in Theoretical Computer Science. 2013: 87-96.

[9] MCSHERRY F D. Privacy integrated queries: an extensible platform for privacy-preserving data analysis[C]//Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. 2009: 19-30.

[10] MOHAN P, THAKURTA A, SHI E, et al. GUPT: privacy preserving data analysis made easy[C]//Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. 2012: 349-360.

[11] PROSERPIO D, GOLDBERG S, MCSHERRY F. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets[J]. Proceedings of the VLDB Endowment, 2014, 7(8): 637-648.

[12] NARAYAN A, HAEBERLEN A. DJoin: Differentially private join queries over distributed databases [C]//Presented as part of the 10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12). 2012: 149-162.

[13] NISSIM K, RASKHODNIKOVA S, SMITH A. Smooth sensitivity and sampling in private data analysis [C]//Proceedings of the thirty-ninth annual ACM symposium on Theory of computing. 2007: 75-84.