Towards Practical Differential Privacy for SQL Queries

Noah Johnson, Joseph P. Near, Dawn Song
UC Berkeley
Outline

1. **Discovering** real-world requirements
2. **Elastic sensitivity** & calculating sensitivity of SQL queries
3. **Our experience**: lessons & challenges
Part 1

Discovering Real-world Requirements
Our collaboration with Uber

• Uber’s goal: **deploy differential privacy**
  • Internally (for some analysts)
  • Externally (for partners & regulators)

• Our goals
  • Explore **real-world requirements** for differential privacy
  • Build **open-source systems**
Previous work on differential privacy for analytics: insufficient for real-world applications

**Previous work**: either...
- Theoretical (does not explore practical applications)
- Targets specialized analytics tasks
  - Google RAPPOR: browsing statistics
  - Apple: keyboard & emoji trends

**Result**: little use in real-world analytics environments
- No practical, scalable systems for DP in analytics
Empirical study: understanding real-world data analytics

• Conducted large-scale empirical study of real-world analytics queries

• Dataset: 8 million SQL queries written by data analysts at Uber
  • Covers wide range of use cases: fraud detection, marketing, business metrics, etc.

• Goal: identify DP requirements for real-world workload
Empirical study results

The most common aggregations are **COUNT, SUM, AVG, MAX, and MIN:**

- **COUNT:** 39.3%
- **SUM:** 22.6%
- **AVG:** 6.5%
- **MAX:** 4.6%
- **MIN:** 3.8%
- **MEDIAN:** 0.2%
- **STDDEV:** 0.1%

**Most existing DP mechanisms support only counting queries**
Empirical study results

62% of queries use JOIN, and some queries use many joins:

- Very few existing mechanisms support join
Empirical study results

Many different databases in use

<table>
<thead>
<tr>
<th>Database</th>
<th># queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertica</td>
<td>6,362,631</td>
</tr>
<tr>
<td>Postgres</td>
<td>1,494,680</td>
</tr>
<tr>
<td>MySQL</td>
<td>94,206</td>
</tr>
<tr>
<td>Hive</td>
<td>81,660</td>
</tr>
<tr>
<td>Presto</td>
<td>39,521</td>
</tr>
<tr>
<td>Other</td>
<td>29,387</td>
</tr>
</tbody>
</table>

Existing approaches require modifying/replacing DB
Part 2
Elastic Sensitivity & Analyzing SQL Queries
Global sensitivity vs. local sensitivity for joins

Global sensitivity

- **Unbounded** for queries with joins
  - Single added join key in one table could match an unbounded number of keys in another

Local sensitivity

- **Bounded** for queries with joins
  - Data in true database bounds number of possible new matches
  - Computationally expensive
    - Must consider every possible change to true database
Elastic sensitivity

**Upper bound on local sensitivity**
- Efficient, compositional calculation from query

**Supports queries with equijoins**
- **Insight**: increase in size of joined relation tightly bounded by multiplicities of join keys
- Key multiplicities queried from database in advance

**Supports more than just count**
- Works well for COUNT
- Works less well for SUM
Example: elastic sensitivity of join

```
SELECT COUNT(*) FROM A JOIN B ON A.k = B.k
```

Duplicate join key 1 causes duplicate rows in joined relation

Maximum change in COUNT: add another 1 to A

Local sensitivity = 2

In general: local sensitivity bounded by maximum multiplicities of k in A and B
A static analysis framework for SQL queries

Built a practical framework for analyzing real-world queries

Challenge: these queries are complex

Our framework:
  • Solve complexity once
  • Enable many different analyses
Differential privacy for SQL queries using Elastic Sensitivity
Empirical evaluation results

Dataset: 9862 Uber queries, run on production database
Part 3
Lessons Learned & Future Challenges
Value of close collaboration

• Opportunity to examine **real use cases**
  • Dataset of queries: what analysts actually *did*

• Insight into **privacy goals** in the real world
  • e.g. concern about external *and* internal sharing

• Discover **requirements** & infrastructure restrictions
  • e.g. we *really can’t* modify the database engine
Challenges of close collaboration

• **Analysts** skeptical about need for privacy protections
  • Concerned about utility
  • Believe privacy is already protected
  • e.g. machine learning teams believe models protect privacy

• **Privacy team** unsure of privacy goals
  • Belief that de-identification is enough, *or*
  • Differential privacy seen as a silver bullet
  • Would like to “have differential privacy” all in one go

• **Infrastructure teams** want a one-size-fits-all solution
  • Multiple solutions = more work
Conclusions

• Perfect deployment will take time, experimentation
  • Early versions will be limited
  • There will be bugs

• We can accelerate the process
  • Encouragement
  • Constructive engagement

• We should encourage transparency
  • Secrecy encourages bugs, discourages adoption

https://github.com/uber/sql-differential-privacy
https://arxiv.org/abs/1706.09479

jnear@berkeley.edu

Thank you!