Private SQL: a Differentially Private SQL Engine

Ios Kotsogiannis

Microsoft Research, April 10 2019
Overview

- Introduction
- Private SQL
- Empirical Evaluation
- Ongoing and Future Work
Introduction

We live in a data-fueled world

Want to share this data:

- US Census data releases (e.g., SF-1)
- Train predictive ML algorithms based on Skype logs
- Share data within the organization (e.g., Uber)

Traditional databases
Introduction

U.S. Census:
- Congressional apportionment
- Redistricting
- SF-1 Release

Title 13, chapter 9:
Neither the secretary nor any officer or employee ... make any publication whereby the data furnished by any particular establishment or individual under this title can be identified ...
Introduction

U.S. Census:
- Congressional apportionment
- Redistricting
- SF-1 Release

```
SELECT COUNT(*) FROM (
    SELECT hid, COUNT(*) AS CNT
    FROM ( SELECT hid, COUNT(*) AS CNT
    WHERE p.hid = h.hid AND p.Rel = 'child'
    AND p.Age < 18
    GROUP BY hid)
WHERE CNT >= 1
```

Count of the number of households where the householder age in [18..64] AND it's a husband-wife family AND there is at least one related child under 18.

Title 13, chapter 9:
Neither the secretary nor any officer or employee ... make any publication whereby the data furnished by any particular establishment or individual under this title can be identified ...
Differential Privacy

Presence or absence of a tuple in a dataset does not affect the output of a DP mechanism by too much.

More specifically, a mechanism $M$ is $\varepsilon$-DP iff:

$$\forall S \in \text{Range}(M), \forall D' \in \text{nbrs}(D):$$

$$\Pr[M(D) \in S] \leq e^{\varepsilon} \Pr[M(D') \in S]$$

Where, $D$ and $D'$ are neighboring if they differ in one tuple:

$$D' \in \text{nbrs}(D), \text{ then } |D - D'| \cup |D' - D| = 1$$
**Post processing:** Execution of any algorithm on the output of a DP algorithm does not incur additional privacy loss.

**Composition:** The sequential execution of differentially private algorithms is also differentially private.

For algorithms $M_1, \ldots, M_k$ each satisfying $\epsilon_i$ — Differential Privacy, their sequential execution satisfies $\epsilon$ — Differential Privacy.

With: $\epsilon = \sum_i \epsilon_i$

Similarly, if $M_1, \ldots, M_k$ are executed on a different partition $D_i$ of the data, then their parallel execution satisfies $\max\{\epsilon_i\}$ — DP.
Differential Private Algorithms

Work by adding noise to the query answers.

- High values of $\varepsilon \rightarrow$ less noise, less private
- Low values of $\varepsilon \rightarrow$ more noise, more private

Scale of noise calibrated to sensitivity of query.

Sensitivity of a query is the maximum change of that query for all neighboring datasets.

$$S(Q) = \max_{\forall D, D' \in \text{enbrs}(D)} \|Q(D) - Q(D')\|_1$$
Remove row from Household, from Person, or from both?

\[ D' \text{ nebrs}(D), \text{ then } |D - D'| \cup |D' - D| = 1 \]
We want to build a system that:

- Answers complex SQL queries on a DB
- Use a common privacy budget for all of them
- Privacy requirements defined from the data owner
Remove row from Household, from Person, or from both?

\[ D' \text{ nbrs}(D) \text{, then } |D - D'| \cup |D' - D| = 1 \]
Goal

We want to build a system that:

- Answers complex SQL queries on a DB
- Use a common privacy budget for all of them
- Privacy requirements defined from the data owner
Prior Work Solutions

One query at a time:

Private Database

DP Algorithm

Query 1
Answer 1

Query 2
Answer 2

Flex [VLDB18]
Employed at Uber

PINQ [SIGMOD09]
Prior Work Solutions

One query at a time:

Private Database

Unbounded privacy loss -- or stop query answering.

No consistency between query answers.
Prior Work Solutions

Synthetic data:

Private Database

HDMM [VLDB18], MWEM [NIPS12] ...
Output a histogram tuned to query workload

PrivBayes [SIGMOD14], Private Synthetic Data using GANs [NIST Challenge 18]
Generates a synthetic database in the same schema as input
Prior Work Solutions

One query at a time:

Private Database

Unbounded privacy loss -- or stop query answering.
No consistency between query answers.
Prior Work Solutions

Synthetic data:

Private Database

HDMM [VLDB18], MWEM [NIPS12] ...
Output a histogram tuned to query workload

PrivBayes [SIGMOD14], Private Synthetic Data using GANs [NIST Challenge 18]
Generates a synthetic database in the same schema as input
Prior Work Solutions

Synthetic data:

Private Database

No support for multi-relational tables
Joins computed on synthetic tables incur high error $O(\sqrt{n})$
[McGregor FOCS 2010]
Goal

We want to build a system that:

- Answers complex SQL queries on a DB
- Use a common privacy budget for all of them
- Privacy requirements defined from the data owner
Overview

- Introduction
- Private SQL
- Empirical Evaluation
- Ongoing and Future Work
→ **Release of private synopses**
   Constant privacy loss, prevention of side-channel attacks, and consistency among query answers

→ **Synopses over views of the schema**
   [Mironov 2009] Queries involving joins cannot be accurately answered using synopses of the base tables

→ **Views are tuned to a representative workload**
   [Dinur 2003] We cannot accurately answer an arbitrarily large set of queries under strong privacy guarantees.

→ Full design principles: [K CIDR 2019]
Private SQL Overview

Private Synopses Generation

Query Engine

- Query 1
  - Answer 1
- Query 2
  - Answer 2
Private SQL Overview

- Tuned synopses to Q
- Correctly guarantees requirements of P, ε
Private SQL Overview

- Provides SQL interface to analyst
- Knows how to answer SQL queries based on synopses generated
Generating Synopses

Private Synopsis Generation

VSSELECTOR

VRewriter

SensCalc

PrivSynGen

BudgetAlloc

P, Q, ε

v = {v}

ξ = {ξ}

Σ = {Δ}

Private Synopses
View Selection

- Generates views based on Q
- Captures join structures of q
- Partitions Q to partial workloads

Private Synopsis Generation

VSSELECTION

VREWRITE

SensCalc

PrivSynGen

$\mathcal{E} = \{ e_i \}$

$\mathcal{V} = \{ v \}$

$\mathcal{D}$
Budget Allocation

Private Synopsis Generation

VSELECTOR → VRewriter → SensCalc → PrivSynGen

Assign privacy budget to each view

\v = \{v\} \quad \Sigma = \{\Delta\} \quad \mathcal{E} = \{\varepsilon\}
View Selection

- Generates views based on Q
- Captures join structures of q
- Partitions Q to partial workloads

Private Synopsis Generation
Sensitivity Calculation

- Compute sensitivity of each view
- Satisfy privacy requirements

Private Synopsis Generation
Budget Allocation

Private Synopsis Generation

VSelector

VRewrite

SensCalc

PrivSynGen

BudgetAlloc

Assign privacy budget to each view

\[ v = \{ V \} \]

\[ \mathcal{E} = \{ \varepsilon_i \} \]

\[ \Sigma = \{ \Delta \} \]
Execute a DP algorithm on the input \((V, D, Q, \varepsilon)\)

Can be any algorithm from prior work: DAWA, PrivBayes, etc.
Sensitivity Calculation

- Compute sensitivity of each view
- Satisfy privacy requirements

Private Synopsis Generation
### Sensitivity Revisit

**Person**

<table>
<thead>
<tr>
<th>pid</th>
<th>age</th>
<th>...</th>
<th>hid</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>p100</td>
<td>45</td>
<td>...</td>
<td>h02</td>
</tr>
<tr>
<td>p101</td>
<td>46</td>
<td>...</td>
<td>h02</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Household**

<table>
<thead>
<tr>
<th>hid</th>
<th>...</th>
<th>gid</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>h02</td>
<td>...</td>
<td>g04</td>
</tr>
<tr>
<td>h03</td>
<td>...</td>
<td>g04</td>
</tr>
</tbody>
</table>

Q := SELECT COUNT(*) FROM PERSON WHERE PERSON.AGE > 17;

\[
S(Q) = \max_{\forall D, D' \in nbs(D)} \| Q(D) - Q(D') \|_1
\]
Sensitivity Revisit

<table>
<thead>
<tr>
<th>Person</th>
<th></th>
<th></th>
<th>Household</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>age</td>
<td>...</td>
<td>hid</td>
<td>gid</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>p100</td>
<td>45</td>
<td>...</td>
<td>h02</td>
<td></td>
</tr>
<tr>
<td>p101</td>
<td>46</td>
<td>...</td>
<td>h02</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Q := SELECT COUNT(*) FROM PERSON WHERE PERSON.AGE > 17;

\[ S(Q) = \max_{D, D' \in \text{enbr}(D)} \| Q(D) - Q(D') \|_1 \]
Sensitivity Revisit

<table>
<thead>
<tr>
<th>pid</th>
<th>age</th>
<th>...</th>
<th>hid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p100</td>
<td>45</td>
<td>...</td>
<td>h02</td>
</tr>
<tr>
<td>p101</td>
<td>46</td>
<td>...</td>
<td>h02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Household

<table>
<thead>
<tr>
<th>hid</th>
<th>...</th>
<th>gid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h02</td>
<td>...</td>
<td>g04</td>
</tr>
<tr>
<td>h03</td>
<td>...</td>
<td>g04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Q := \texttt{SELECT COUNT(*) FROM PERSON WHERE PERSON.AGE > 17};

D and \(D'\) differ in a row of Person \(\rightarrow S(Q) = 1\)

Neighboring if they differ in one table
What Flex does
## Sensitivity Revisit

### Person

<table>
<thead>
<tr>
<th>pid</th>
<th>age</th>
<th>...</th>
<th>hid</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>p100</td>
<td>45</td>
<td>...</td>
<td>h02</td>
</tr>
<tr>
<td>p101</td>
<td>46</td>
<td>...</td>
<td>h02</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Household

<table>
<thead>
<tr>
<th>hid</th>
<th>...</th>
<th>gid</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>h02</td>
<td>...</td>
<td>g04</td>
</tr>
<tr>
<td>h03</td>
<td>...</td>
<td>g04</td>
</tr>
</tbody>
</table>

Q := **SELECT COUNT(*) FROM PERSON WHERE PERSON.AGE > 17**;

D and D′ differ in a row of Household → S(Q) = 0

**Neighboring if they differ in one table**

**What Flex does**
Sensitivity Revisit

Q := \texttt{SELECT COUNT(*) FROM PERSON WHERE PERSON.AGE > 17};

D and D’ differ in a row of Household \(\rightarrow S(Q) = 0\)

Foreign key constraint \(\rightarrow\) removing one row from Household will affect multiple rows of the Person table
Sensitivity Revisit

<table>
<thead>
<tr>
<th>pid</th>
<th>age</th>
<th>...</th>
<th>hid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p100</td>
<td>45</td>
<td>...</td>
<td>h02</td>
</tr>
<tr>
<td>p101</td>
<td>46</td>
<td>...</td>
<td>h02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>hid</th>
<th>...</th>
<th>gid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h02</td>
<td></td>
<td>g04</td>
</tr>
<tr>
<td>h03</td>
<td></td>
<td>g04</td>
</tr>
</tbody>
</table>

Q := SELECT COUNT(*) FROM PERSON WHERE PERSON.AGE > 17;

D and D' differ in a row of Household → S(Q) = 0

Neighboring if they differ in one table
What Flex does
Sensitivity Revisit

Q := \textbf{SELECT COUNT(*) FROM PERSON WHERE PERSON.AGE > 17;}

D and D’ differ in a row of Household \(\rightarrow\) \(S(Q) = 0\)

\textbf{Foreign key constraint} \(\rightarrow\) removing one row from Household will affect multiple rows of the Person table
Privacy for Relational Data

Policy: specifies the primary private relation $\mathcal{R}$

Key constraints: secondary private relations.

Neighboring databases: Keep track of changes in private relations as we add/remove tuples from $\mathcal{R}$ via cascade deletions.

Privacy defined in terms of $(\mathcal{R}, \varepsilon)$

→ Schema needs to be acyclic
Sensitivity Revisit

```
Q := SELECT COUNT(*) FROM PERSON WHERE PERSON.AGE > 17;
```

D and D' differ in a row of Household $\rightarrow$ S(Q) = 0

- **Foreign key constraint** $\rightarrow$ removing one row from Household will affect multiple rows of the Person table
### Privacy for Relational Data

#### Person Table

<table>
<thead>
<tr>
<th>pid</th>
<th>age</th>
<th>...</th>
<th>hid</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>p100</td>
<td>45</td>
<td>...</td>
<td>h02</td>
</tr>
<tr>
<td>p101</td>
<td>46</td>
<td>...</td>
<td>h02</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

#### Household Table

<table>
<thead>
<tr>
<th>hid</th>
<th>...</th>
<th>gid</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>h02</td>
<td>...</td>
<td>g04</td>
</tr>
<tr>
<td>h03</td>
<td>...</td>
<td>g04</td>
</tr>
</tbody>
</table>

**Policy**: specifies the primary private relation $R$.

**Key constraints**: secondary private relations.

**Neighboring databases**: Keep track of changes in private relations as we add/remove tuples from $R$ via cascade deletions.

Privacy defined in terms of $(R, \varepsilon)$

$\rightarrow$ Schema needs to be acyclic
Sensitivity Calculation

- Compute sensitivity of each view
- Satisfy privacy requirements

Private Synopsis Generation
Addressing View Sensitivity

- View is a complex SQL query, estimation is hard
  [Arapinis 2016]

- Global sensitivity unbounded in presence of joins

- Calculation depends on privacy resolution
Addressing View Sensitivity

- View is a complex SQL query, estimation is hard [Arapinis 2016]
- Global sensitivity unbounded in presence of joins
- Calculation depends on privacy resolution

Rule based sensitivity
Truncate “outliers”
View Rewriting -- automatic policy enforcement
Addressing View Sensitivity

- View is a complex SQL query, estimation is hard [Arapinis 2016]
- Global sensitivity unbounded in presence of joins
- Calculation depends on privacy resolution

Rule based sensitivity

- Truncate “outliers”
- View Rewriting -- automatic policy enforcement
Sensitivity Calculator

Bottom-up rule based algorithm on a query plan

Builds on top of Elastic sensitivity rules

Extends rules via tracking of keys

\[ V := \text{SELECT relp, race, cnt FROM Person P,} \]
\[ (\text{SELECT count(*) AS cnt, hid FROM Person GROUP BY hid}) \text{ AS P2 WHERE P.hid = P2.hid;} \]
V := SELECT relp, race, cnt FROM
    Person P,
    (SELECT count(*) AS cnt, hid FROM Person GROUP BY hid) AS P2
WHERE P.hid = P2.hid;

<table>
<thead>
<tr>
<th>relp</th>
<th>race</th>
<th>cnt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>head</td>
<td>Asian</td>
<td>3</td>
</tr>
</tbody>
</table>
Sensitivity Calculator

\[
\begin{align*}
\text{(SELECT relp, race, cnt FROM Person P,} \\
\text{(SELECT count(*) AS cnt, hid FROM Person GROUP BY hid) AS P2} \\
\text{WHERE P.hid = P2.hid) AS V;}
\end{align*}
\]

\[S(\text{Person}) = 1\]

\[S(R) = S(\text{Person}) \times 2 = 2\]

hid becomes key → max multiplicity = 1 in R
Sensitivity Calculator

\[
\text{S(Person)} = 1
\]

\[
\text{S(R}_1) = \text{S(Person)} \times 2 = 2
\]

hid becomes key \(\rightarrow\) max multiplicity = 1 in R

\[
\text{S(R}_2) = \text{S(R}_1) \times F(\text{hid, Person}) + \text{S(Person)} = 2F + 1
\]

New rule for join on key attributes
Sensitivity Calculator

```
(SELECT relp, race, cnt FROM Person P,
 (SELECT count(*) AS cnt, hid FROM Person GROUP BY hid) AS P2
WHERE P.hid = P2.hid) AS V;
```

Our rules: \( S(V) = 2F + 1 \)

Without key tracking: \( 3F + 2 \)

This difference is only getting larger for more complex views with additional joins.
SELECT relp, race, cnt FROM Person P,
(SELECT count(*) AS cnt, hid FROM Person GROUP BY hid) AS P2
WHERE P.hid = P2.hid;

What happens for different privacy policy?

We would need a different algorithm to correctly compute it.
Goal: allow sensitivity calculator to automatically enforce privacy policies (Person, Household)

→ via addition of semijoin operators.

Main idea: add semijoin operators on secondary private relations → Sensitivity calculator will correctly update the base sensitivities of all secondary private relations in the query plan.
Semijoin Rewrite

\[ V := \text{SELECT * FROM PERSON WHERE PERSON.AGE > 17} ; \]

Policy: Household

\[
\begin{array}{c}
\pi \\
\sigma_{\text{age}>17} \\
\text{Person}
\end{array}
\]
Semijoin Rewrite

\[ V := \text{SELECT } * \text{ FROM PERSON WHERE PERSON.AGE } > 17; \]

Policy: Household

SensCalc computes \( S(V) = 0 \) for policy Household
\[ V := \text{SELECT} * \text{FROM PERSON WHERE PERSON.AGE > 17}; \]

Policy: **Household**

SensCalc computes \( S(V) = 0 \) for policy Household

SensCalc computes \( S(V) = F \) for policy Household
Goal: allow sensitivity calculator a bound independent on F (i.e., global vs local sensitivity)

→ via addition of truncation operators.

Main idea: adding a truncation operator after private relations. This bounds the max frequency of join attributes and removes dependency on F → gives bound for global sensitivity instead of local.
$V := \texttt{SELECT * FROM PERSON WHERE PERSON.AGE > 17;}$

Policy: \textbf{Household}

SensCalc computes $S(V) = k$ for policy Household
Sensitivity Calculator

```
(SELECT relp, race, cnt FROM
     Person P,
     (SELECT count(*) AS cnt, hid FROM Person GROUP BY hid) AS P2
WHERE P.hid = P2.hid) AS V;
```

\[ S(\text{Person}) = 1 \]
\[ S(R) = S(\text{Person}) \times 2 = 2 \]

hid becomes key → max multiplicity = 1 in R
\[ V := \text{SELECT} \ast \text{FROM} \text{PERSON} \text{WHERE} \text{PERSON.AGE} > 17; \]

Policy: **Household**

SensCalc computes \( S(V) = k \) for policy Household
Overview

- Introduction
- Private SQL
- Empirical Evaluation
- Ongoing and Future Work
**U.S. Census Use Case**

**Dataset:** NC households and people. (5.4M, 2.7M tuples)

**Queries:** 3,685 queries describing the SF-1 data release

Report relative per query error (10 independent trials)

**Private SQL instantiation:**
- representative: full
- psg: w-nnls
- pba: wsize

“Number of people living in owned houses of size 3 where the householder is a married Hispanic male.”

```
<table>
<thead>
<tr>
<th>Person</th>
<th>Household</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>hid</td>
</tr>
<tr>
<td>age</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>hid</td>
<td>gid</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

Main Results

Policy: Person

Error stratified by true query answer

First group all queries

→ for 60% of all queries we achieve less than 10% relative error

→ Outliers with high error due to high sensitivity

→ Error drops for larger query answers
Main Results

Policy: Household

Error stratified by true query answer

First group all queries

→ Errors boosted across all groups

→ Effect of removing a household larger than removing a person
Main Results

Policy: Person

Error stratified by true query answer

First group all queries

→ for 60% of all queries we achieve less than 10% relative error

→ Outliers with high error due to high sensitivity

→ Error drops for larger query answers
Main Results

Policy: Household

Error stratified by true query answer

First group all queries

→ Errors boosted across all groups

→ Effect of removing a household larger than removing a person
Comparison with Prior Work

Comparison w/ baseline adapted from prior work [Flex]

- Flex did not support all queries of workload, we report error on Flex supported alone.
- Flex supports only Persons policy.

Results stratified by true query answers.

Improvement due to 3 compounding factors:
- Queries answered on views
- Tighter sensitivity analysis
- No need for smoothing
Overview

- Introduction
- Private SQL
- Empirical Evaluation
- Ongoing and Future Work
Comparison w/ baseline adapted from prior work [Flex]

- Flex did not support all queries of workload, we report error on Flex supported alone.
- Flex supports only Persons policy.

Results stratified by true query answers.

Improvement due to 3 compounding factors:
- Queries answered on views
- Tighter sensitivity analysis
- No need for smoothing
Overview

- Introduction
- Private SQL
- Empirical Evaluation
- Ongoing and Future Work
**Ongoing**

- Policies that extend to multiple primary private relations
- Support for aggregate queries like $\text{AVG}(\text{Salary})$
- Tighter sensitivity analysis → Better SensCalc rules
- Add support for multiple PSGs and algorithm selection at runtime [K SIGMOD 2016]

**Future**

- Synopsis updater: new $(Q, D, \varepsilon)$ [Cummings NIPS 2018]
- Richer SQL grammar support from VSelector
- Tighter sensitivity analysis → VRewriter find an ‘optimal’ view rewriting w.r.t sensitivity calculations
- Explore other truncation techniques, connection with Lipschitz extension
- Provide error bounds
Comparison w/ baseline adapted from prior work [Flex]

→ Flex did not support all queries of workload, we report error on Flex supported alone.
→ Flex supports only Persons policy.

Results stratified by true query answers.

Improvement due to 3 compounding factors:
- Queries answered on views
- Tighter sensitivity analysis
- No need for smoothing
**Ongoing**

- Policies that extend to multiple primary private relations
- Support for aggregate queries like $\text{AVG(Salary)}$
- Tighter sensitivity analysis $\rightarrow$ Better SensCalc rules
- Add support for multiple PSGs and algorithm selection at runtime [K SIGMOD 2016]

**Future**

- Synopsis updater: new $(Q, D, \epsilon)$ [Cummings NIPS 2018]
- Richer SQL grammar support from VSelector
- Tighter sensitivity analysis $\rightarrow$ VRewriter find an ‘optimal’ view rewriting w.r.t sensitivity calculations
- Explore other truncation techniques, connection with Lipschitz extension
- Provide error bounds
Thank You!
Comparison with Prior Work

Comparison w/ baseline adapted from prior work [Flex]

- Flex did not support all queries of workload, we report error on Flex supported alone.
- Flex supports only Persons policy.

Results stratified by true query answers.

Improvement due to 3 compounding factors:
- Queries answered on views
- Tighter sensitivity analysis
- No need for smoothing
**Ongoing**

- Policies that extend to multiple primary private relations
- Support for aggregate queries like \( \text{AVG}(\text{Salary}) \)
- Tighter sensitivity analysis → Better SensCalc rules
- Add support for multiple PSGs and algorithm selection at runtime [K SIGMOD 2016]

**Future**

- Synopsis updater: new (Q, D, \( \varepsilon \)) [Cummings NIPS 2018]
- Richer SQL grammar support from VSelector
- Tighter sensitivity analysis → VRewriter find an ‘optimal’ view rewriting w.r.t sensitivity calculations
- Explore other truncation techniques, connection with Lipschitz extension
- Provide error bounds