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

NH 2
VT 1
MA 9
RI 2
CT 5
NJ 12
DE 1
MD 8
Gained
Lost
No Change

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**

NH 2
VT 1
MA 9
RI 2
CT 5
NJ 12
DE 1
MD 8
Gained
Lost
No Change
PRIVACY

```
SELECT COUNT(*)
FROM ( SELECT hid, COUNT(*) AS CNT
       FROM Persons p, (SELECT hid
                        FROM Persons p1, Persons p2
                        WHERE p1.hid = p2.hid
                              AND p1.Rel = 'householder'
                              AND p1.Age in [18, …, 64]
                              AND p2.Rel = 'spouse'
                              AND ( (p1.sex= 'M' AND p2.sex = 'F')
                                    OR (p1.sex= 'F' AND p2.sex = 'M'))
                        GROUP BY hid) AS h
       WHERE p.hid = h.hid AND p.Rel = 'child'
       AND p.Age < 18
       GROUP BY hid)
WHERE CNT >= 1
```

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**

NH 2
VT 1
MA 9
RI 2
CT 5
NJ 12
DE 1
MD 8
Gained
Lost
No Change

```
SELECT COUNT(*)
FROM ( SELECT hid, COUNT(*) AS CNT
```

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.

```
                  GROUP BY hid) AS h
        WHERE p.hid = h.hid AND p.Rel = 'child'
        AND p.Age < 18
        GROUP BY hid)
WHERE CNT >= 1
```

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 ε-DP iff:

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

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

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

$$D' \ nbrs(D), \quad \text{then} \quad |D - D'| \cup |D' - D| = 1$$

Differential Privacy -- Composition

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 ϵ_i—Differential Privacy, their sequential execution satisfies ϵ—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{ ϵ_i}– DP*

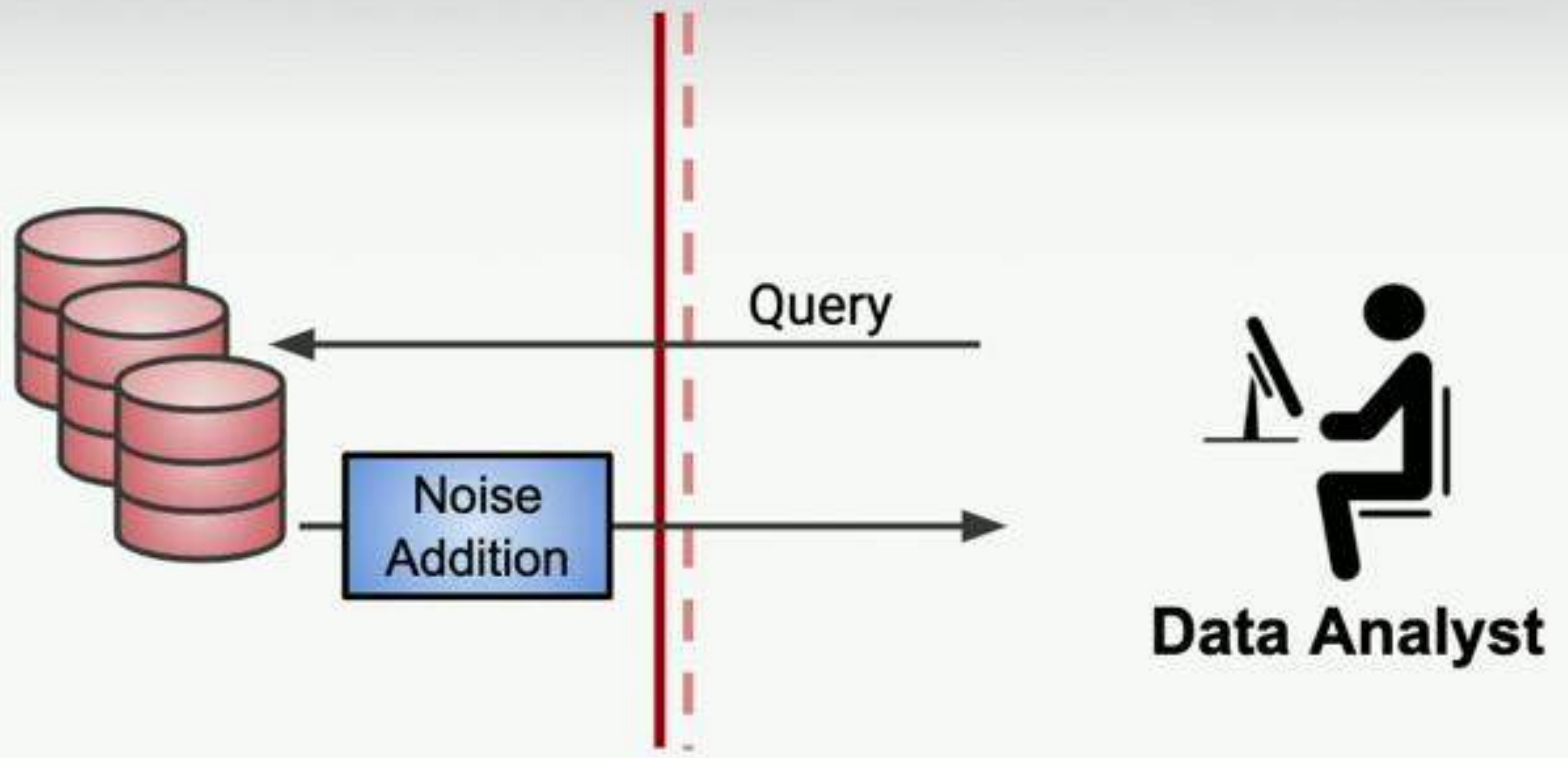
Differential Private Algorithms

Work by adding noise to the query answers.

High values of ε → less noise, less private

Low values of ε → 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.

$$S(Q) = \max_{\forall D, D' \in nbrs(D)} \|Q(D) - Q(D')\|_1$$

Neighboring Databases

Person

pid	**age**	**...**	**hid**
...	...	...	...
p100	45	...	h02
p101	46	...	h02
...	...	...	...

Household

hid	**...**	**gid**
...	...	...
h02	...	g04
h03	...	g04

?

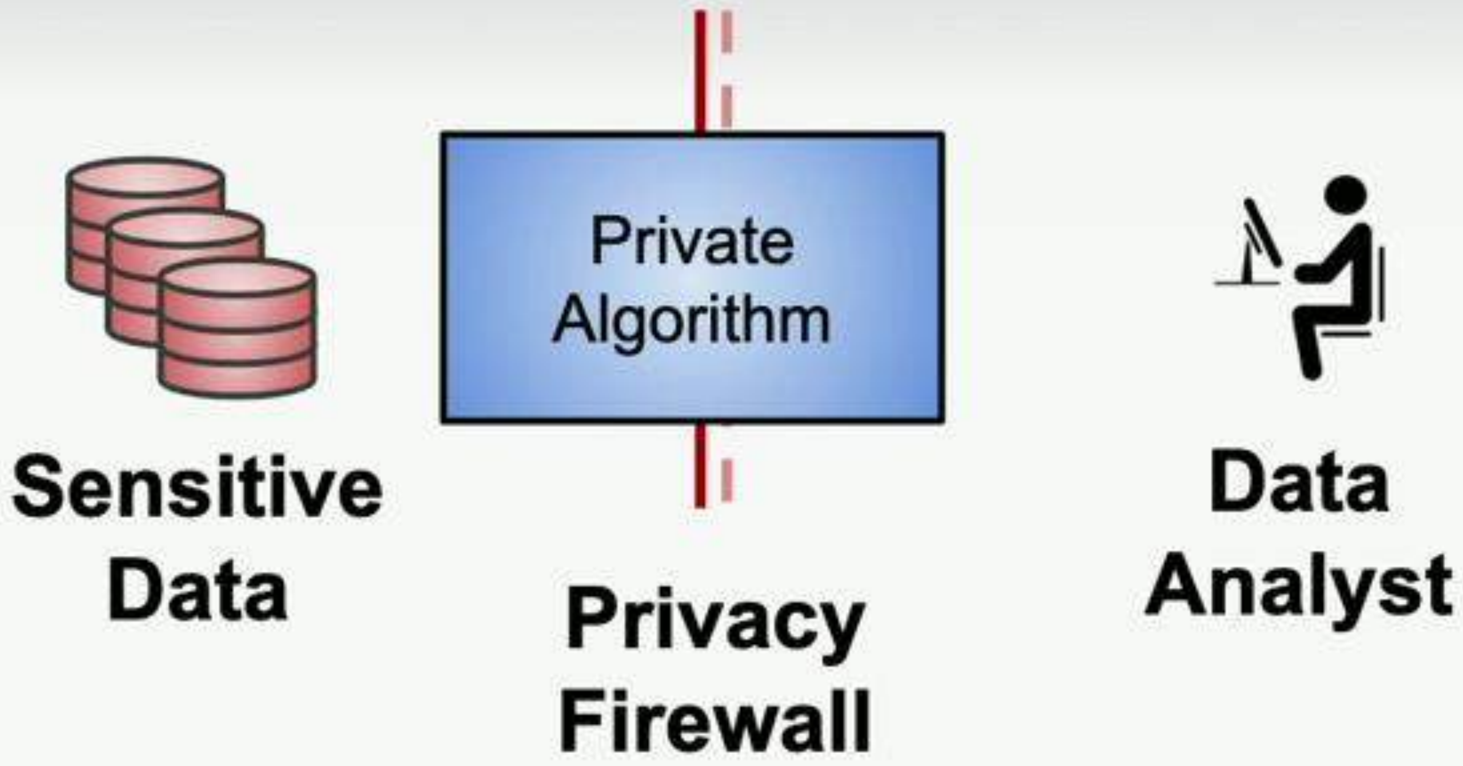
Remove row from Household, from Person, or from both?

$$D' \; nbrs(D), \quad \text{then} \quad |D - D'| \cup |D' - D| = 1$$

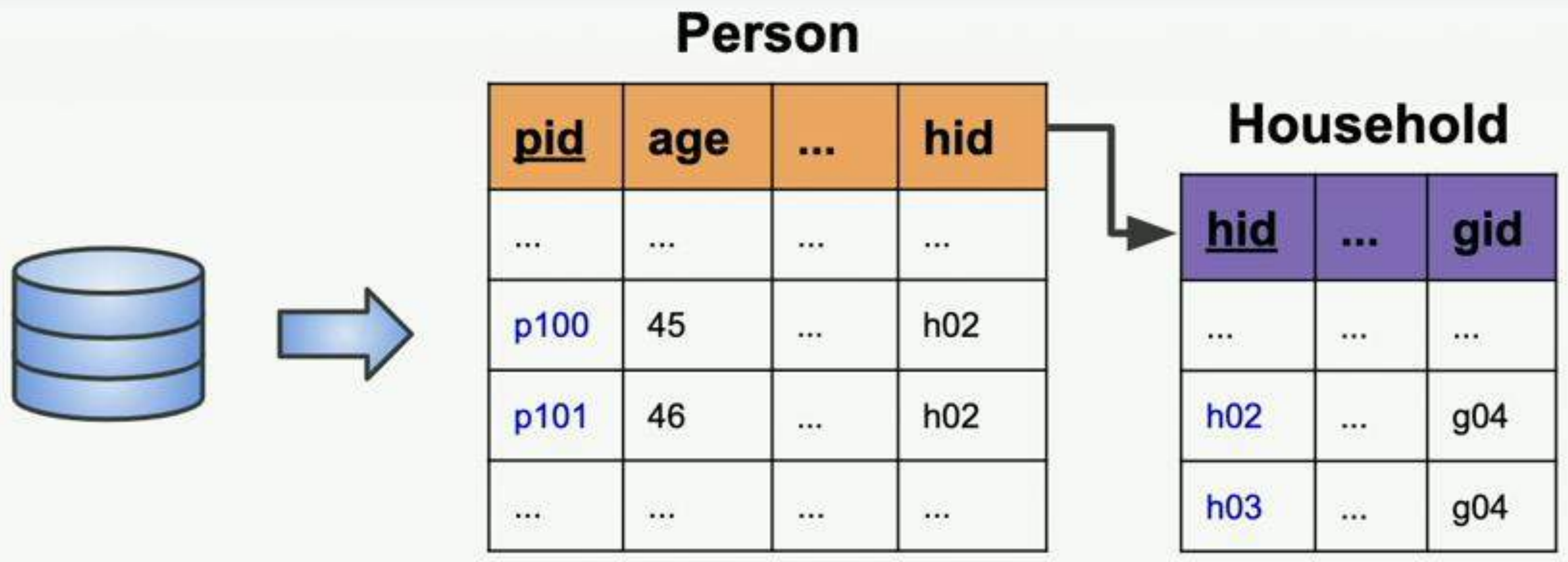
Goal

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



Neighboring Databases

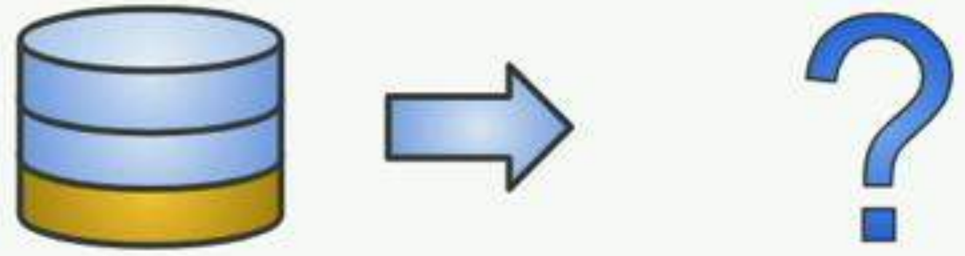


Person

pid	**age**	**...**	**hid**
...	...	...	...
p100	45	...	h02
p101	46	...	h02
...	...	...	...

Household

hid	**...**	**gid**
...	...	...
h02	...	g04
h03	...	g04



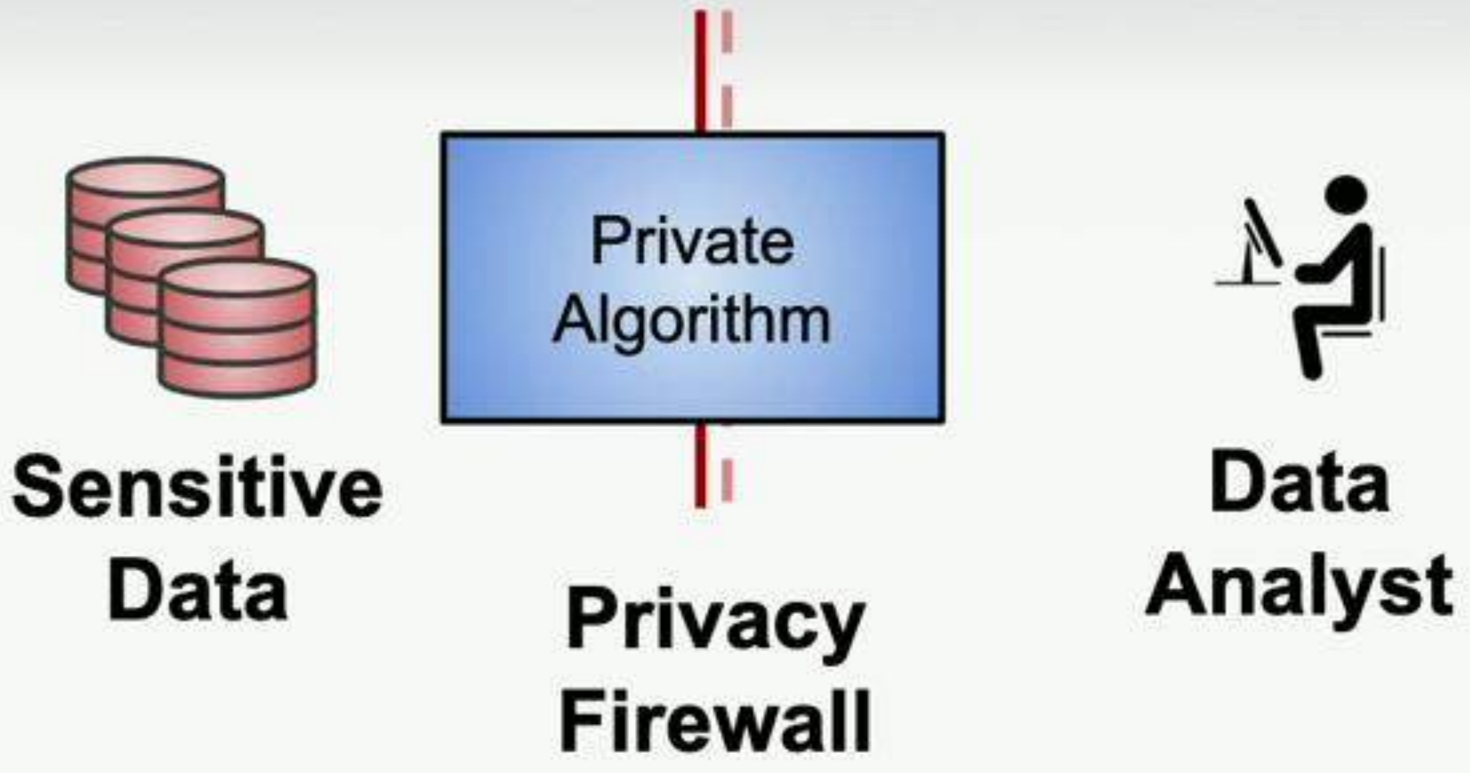
Remove row from Household, from Person, or from both?

$D'\ nbrs(D), \quad \text{then} \quad |D - D'| \cup |D' - D| = 1$

Goal

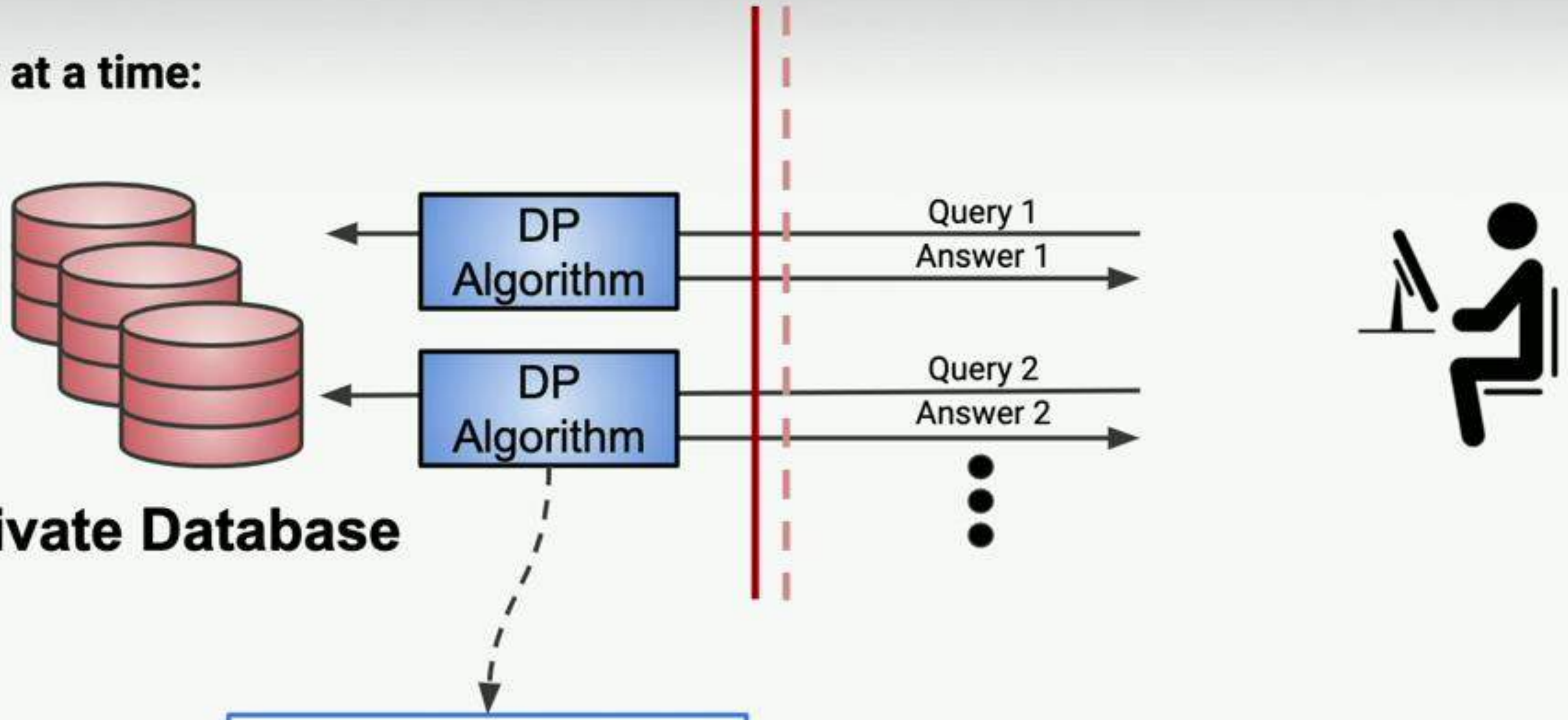
We want to build a system that:

- 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:

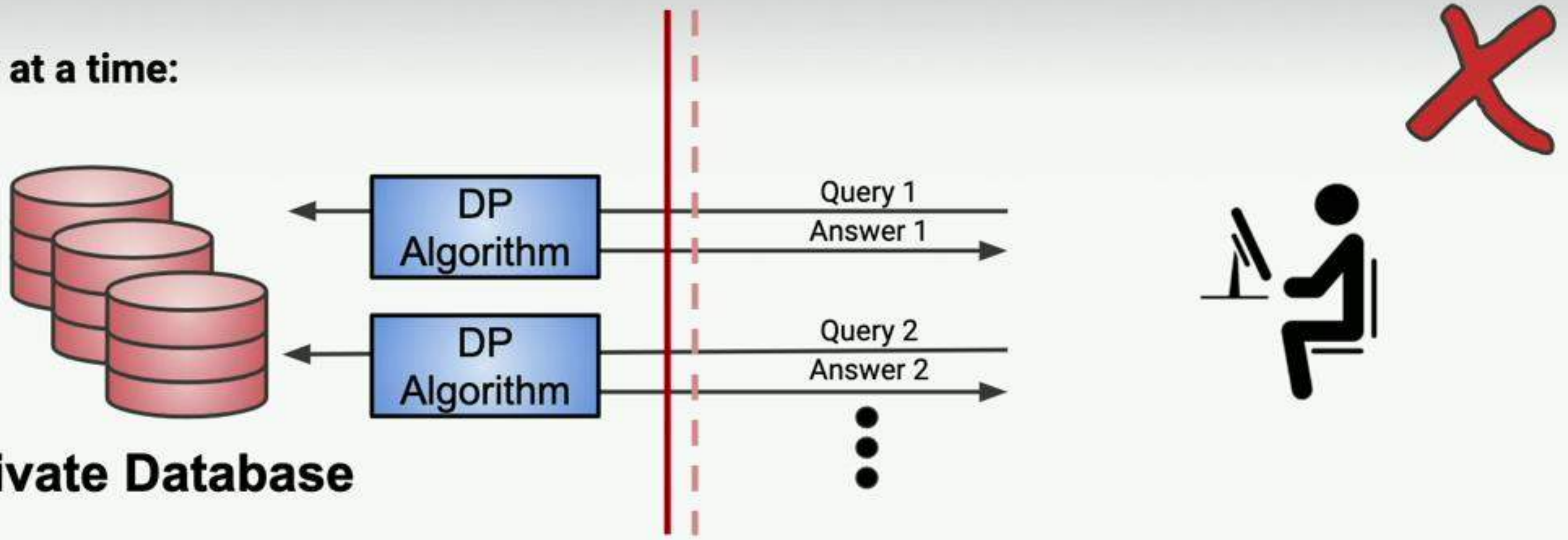


Flex [VLDB18]
Employed at Uber

PINQ [SIGMOD09]

Prior Work Solutions

One query at a time:

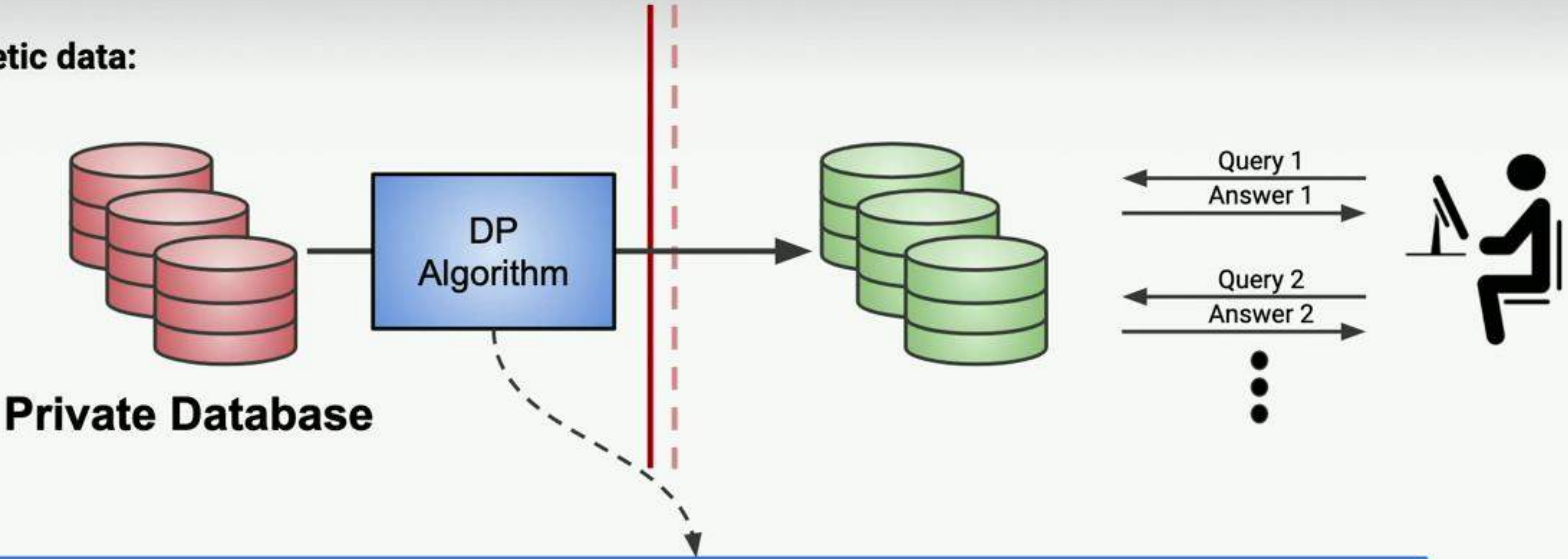


Unbounded privacy loss -- or stop query answering.

No consistency between query answers.

Prior Work Solutions

Synthetic data:



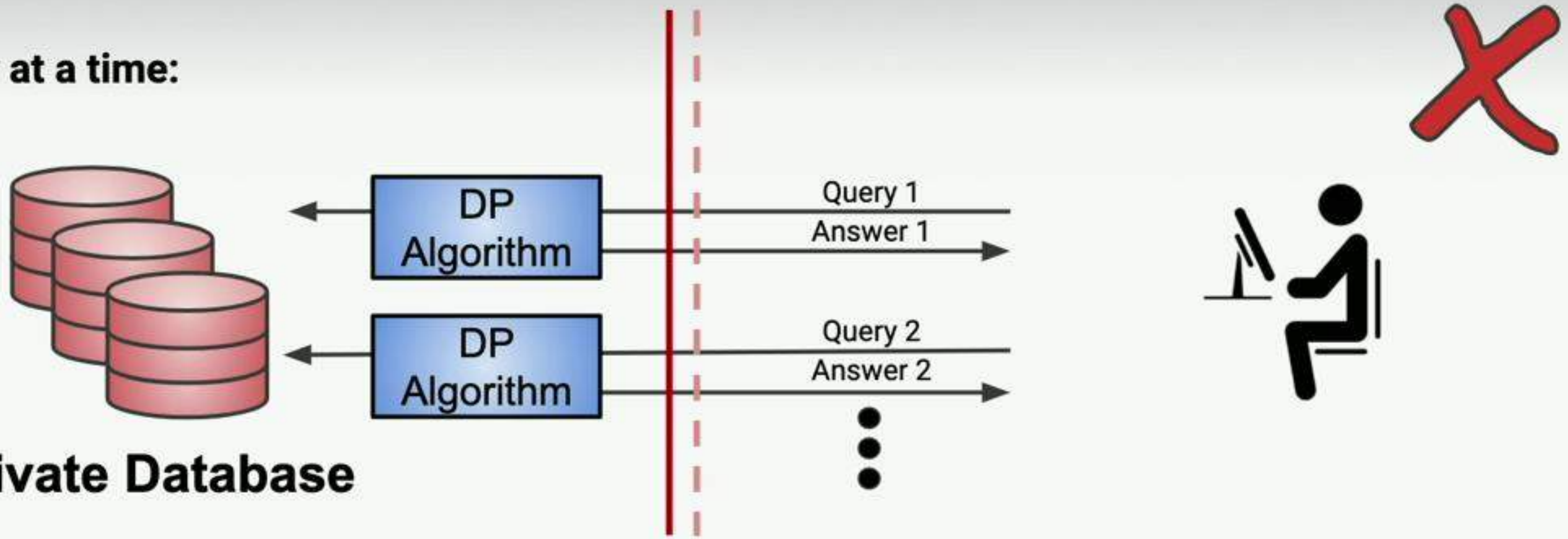
Private Database

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:

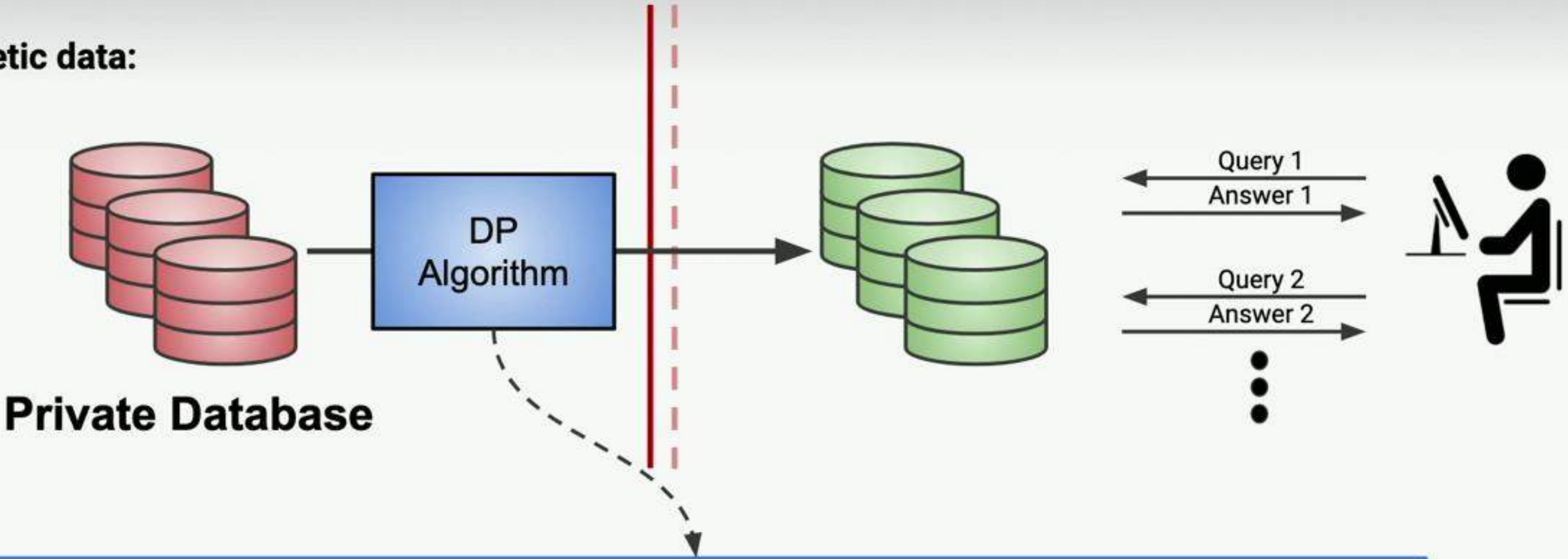


Unbounded privacy loss -- or stop query answering.

No consistency between query answers.

Prior Work Solutions

Synthetic data:

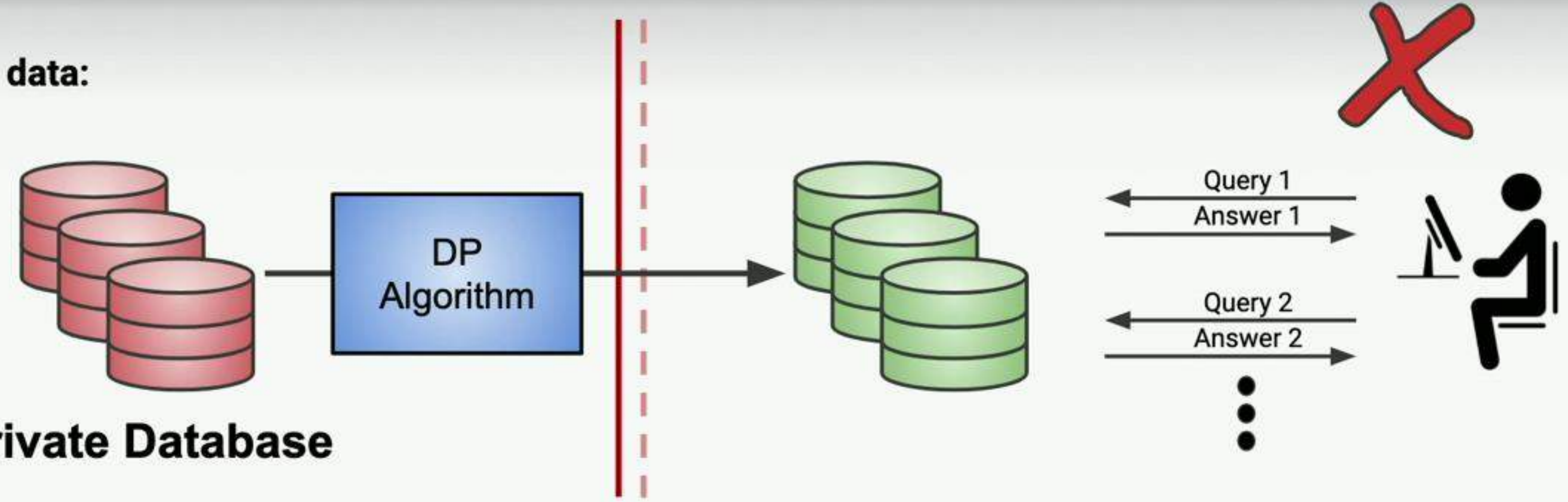


HDMM [VLDB18], MWEM [NIPS12] …
Output a histogram tuned to query workload

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:



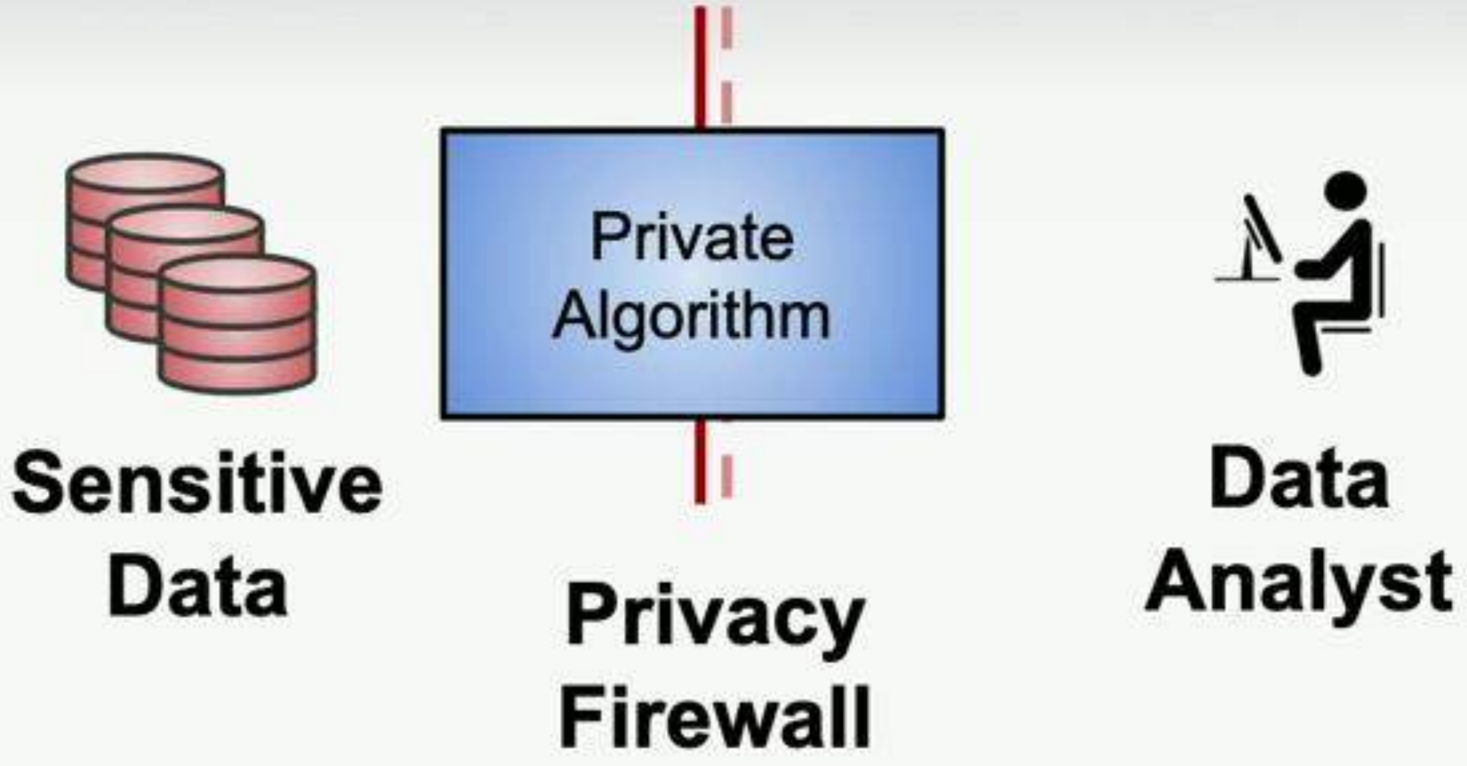
No support for multi-relational tables

Joins computed on synthetic tables incur high error O(√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

Design Principles

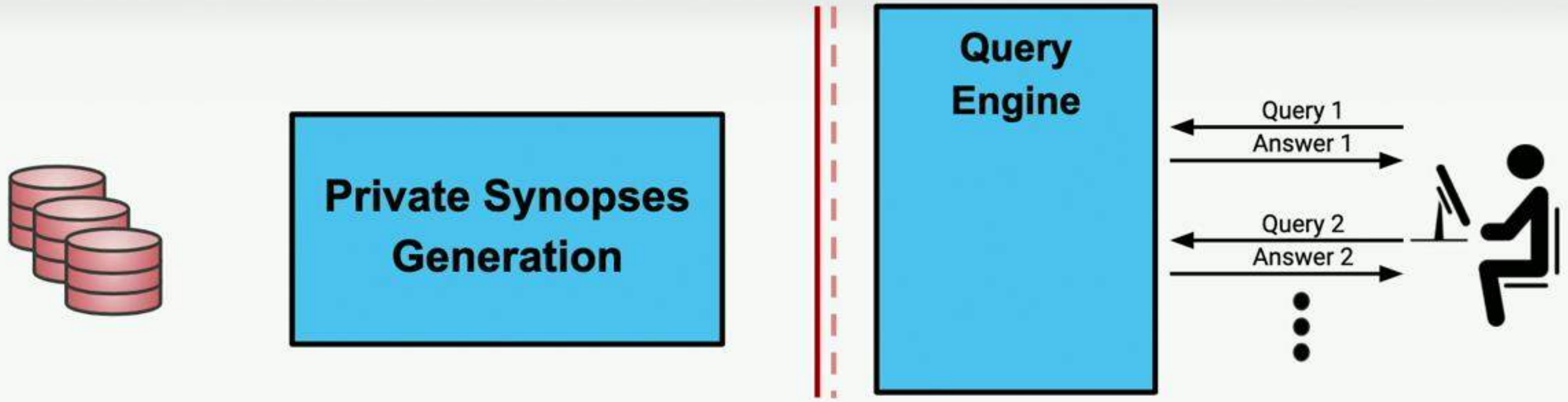
- ➔ **Release of private synopses**
  Constant privacy loss, prevention of side-channel attacks, and consistency among query answers

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

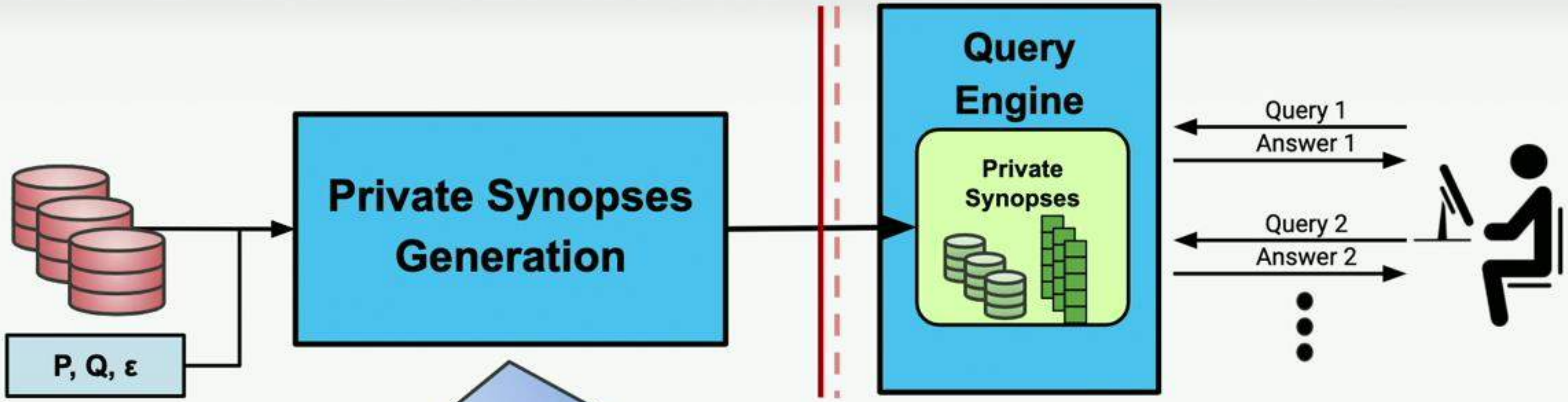
- ➔ **Views are tuned to a representative workload**
  [Dinur 2003] We cannot accurately answer an arbitrarily large set of queries under strong privacy guarantees.

- ➔ Full design principles: [**K** CIDR 2019]

Private SQL Overview

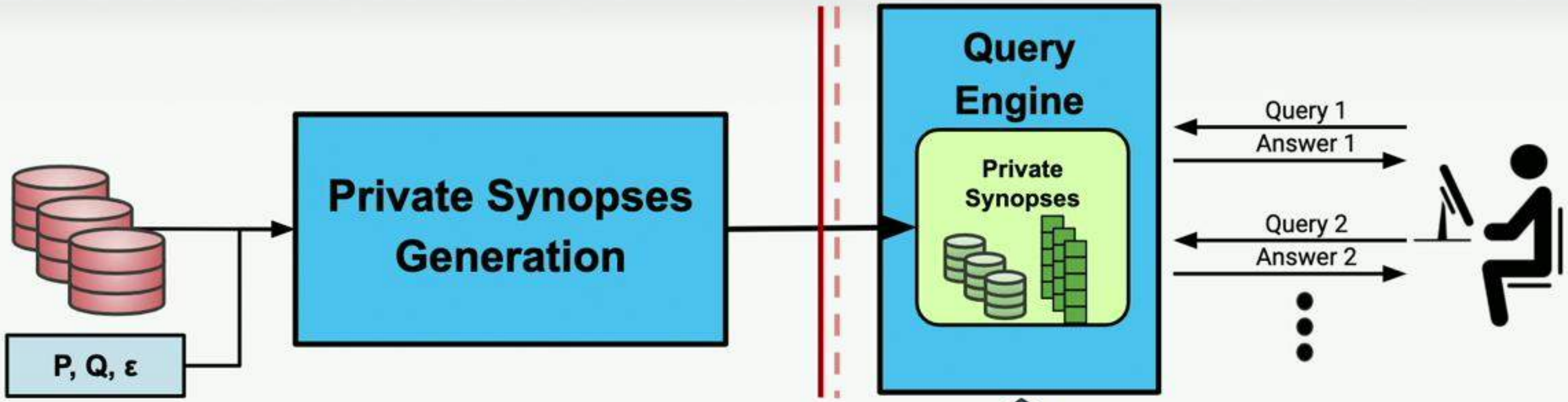


Private SQL Overview



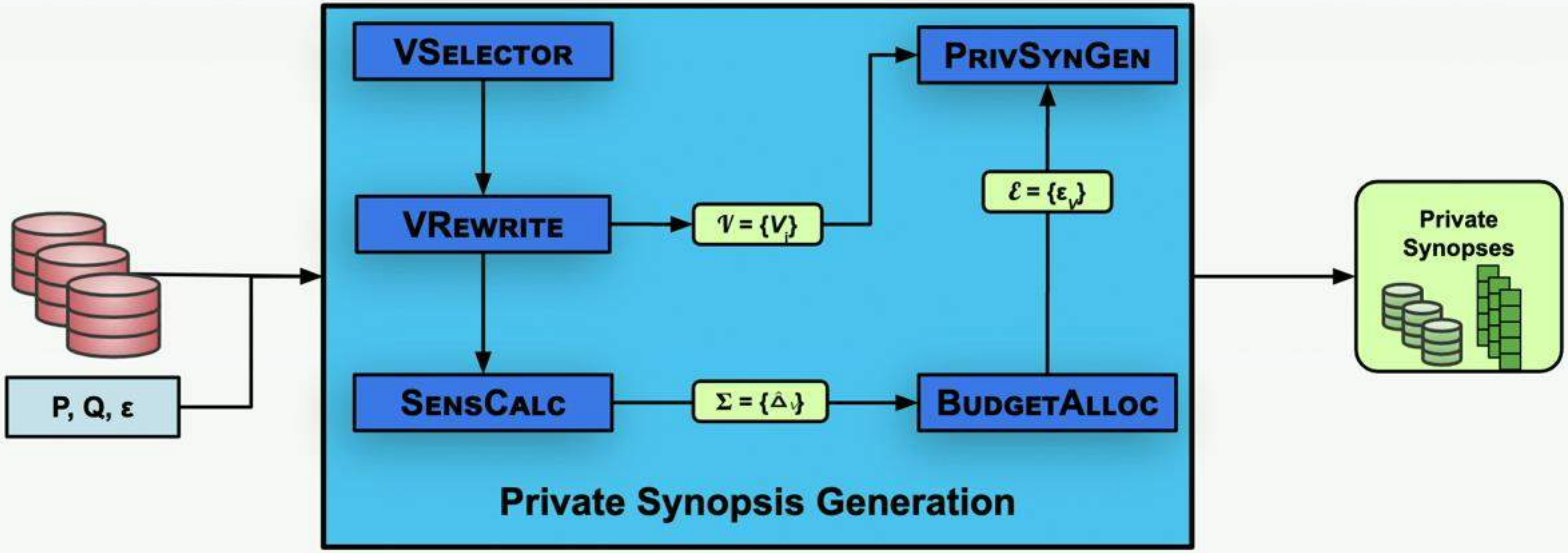
- Tuned synopses to Q
- Correctly guarantees requirements of P,ε

Private SQL Overview

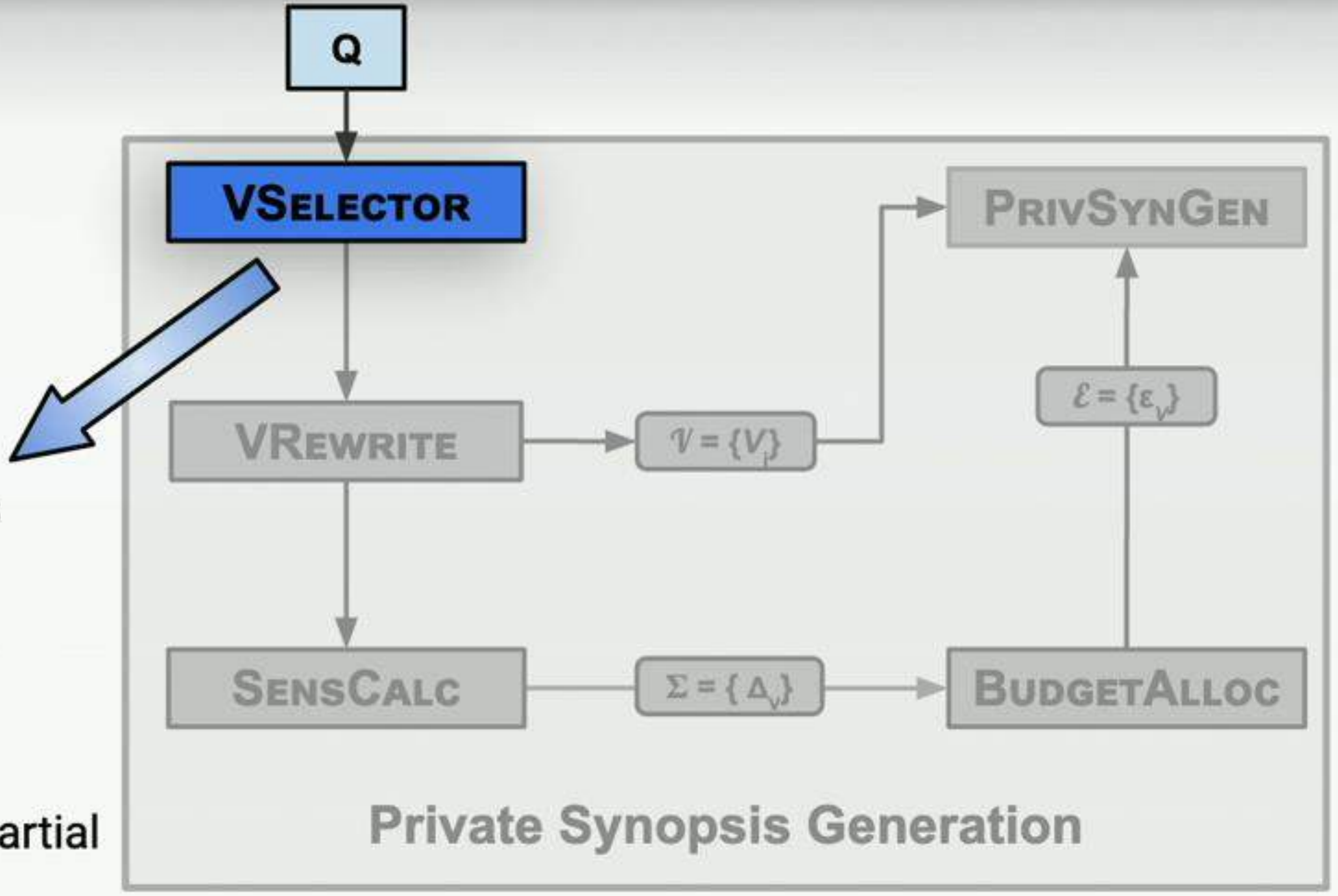


- Provides SQL interface to analyst
- Knows how to answer SQL queries based on synopses generated

Generating Synopses

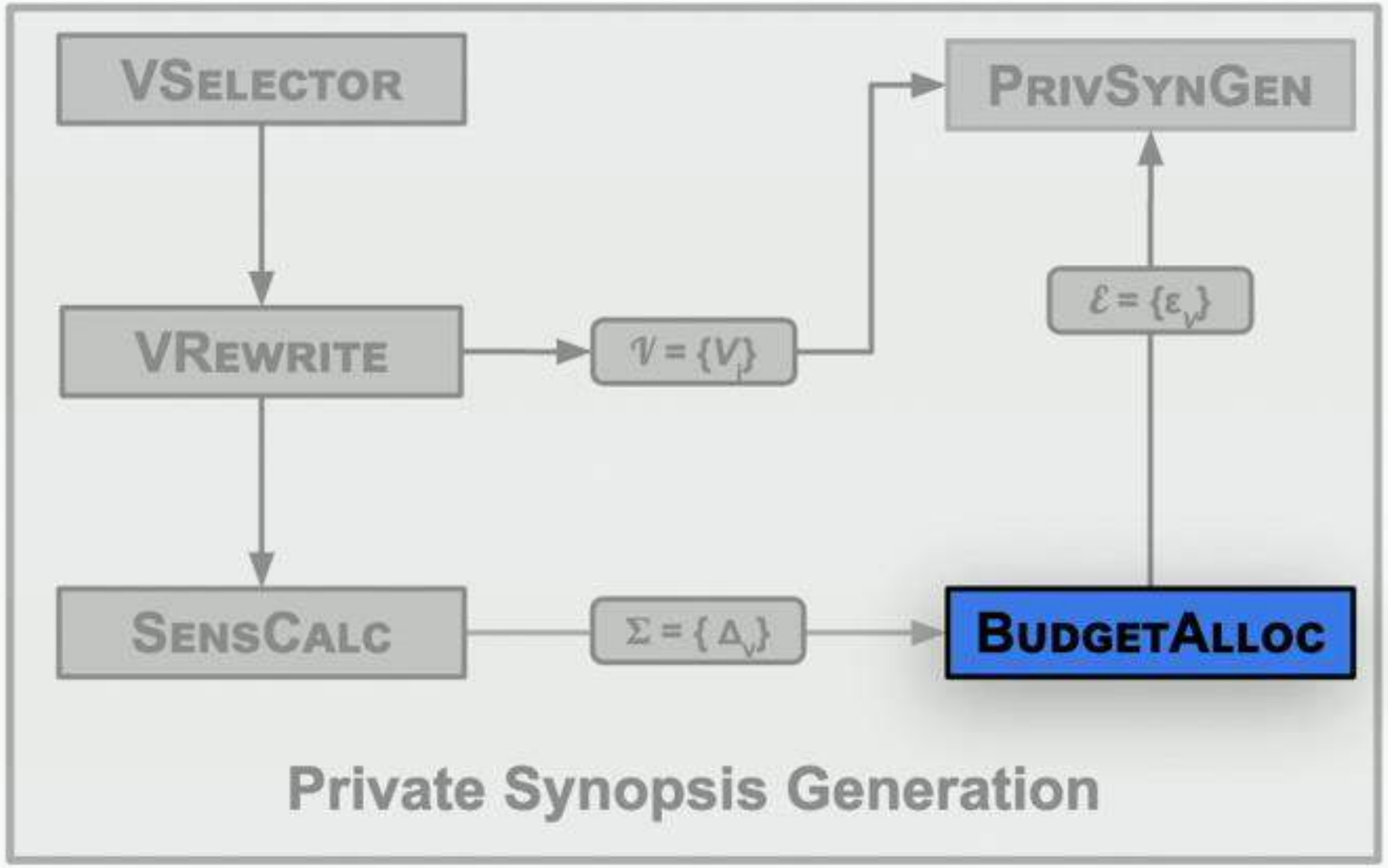


View Selection



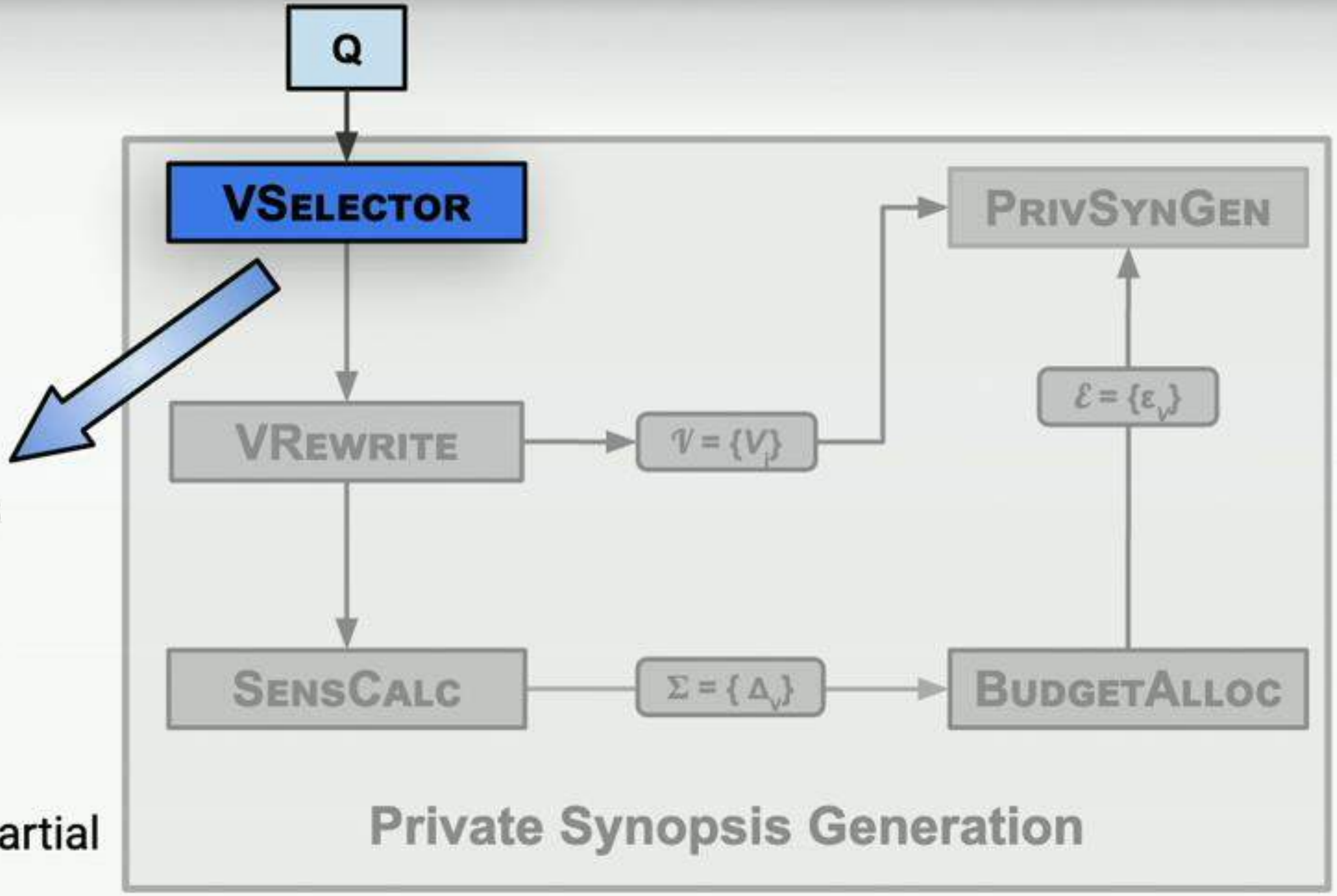
- Generates views based on Q
- Captures join structures of q
- Partitions Q to partial workloads

Budget Allocation



Assign privacy budget to each view

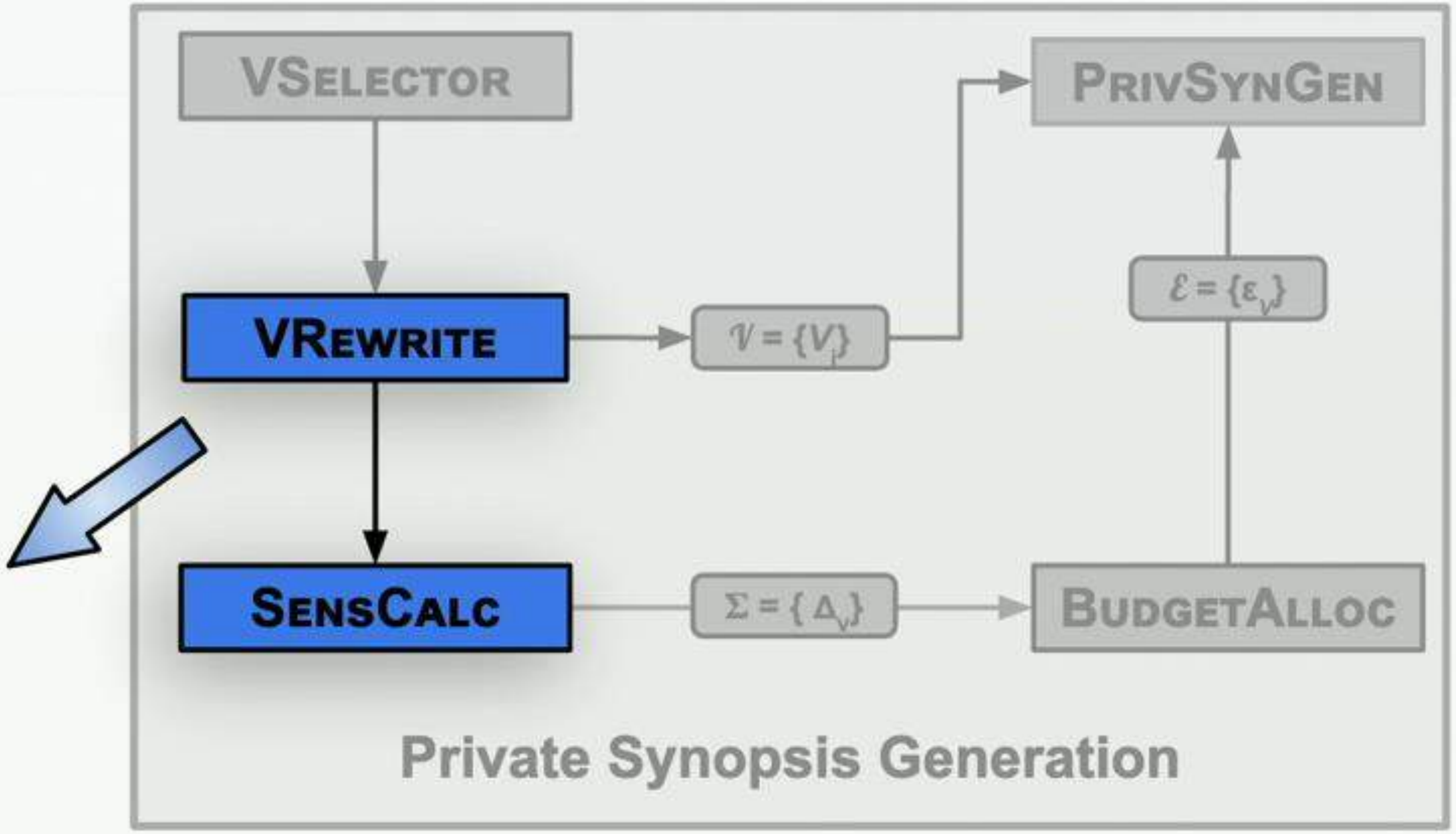
View Selection



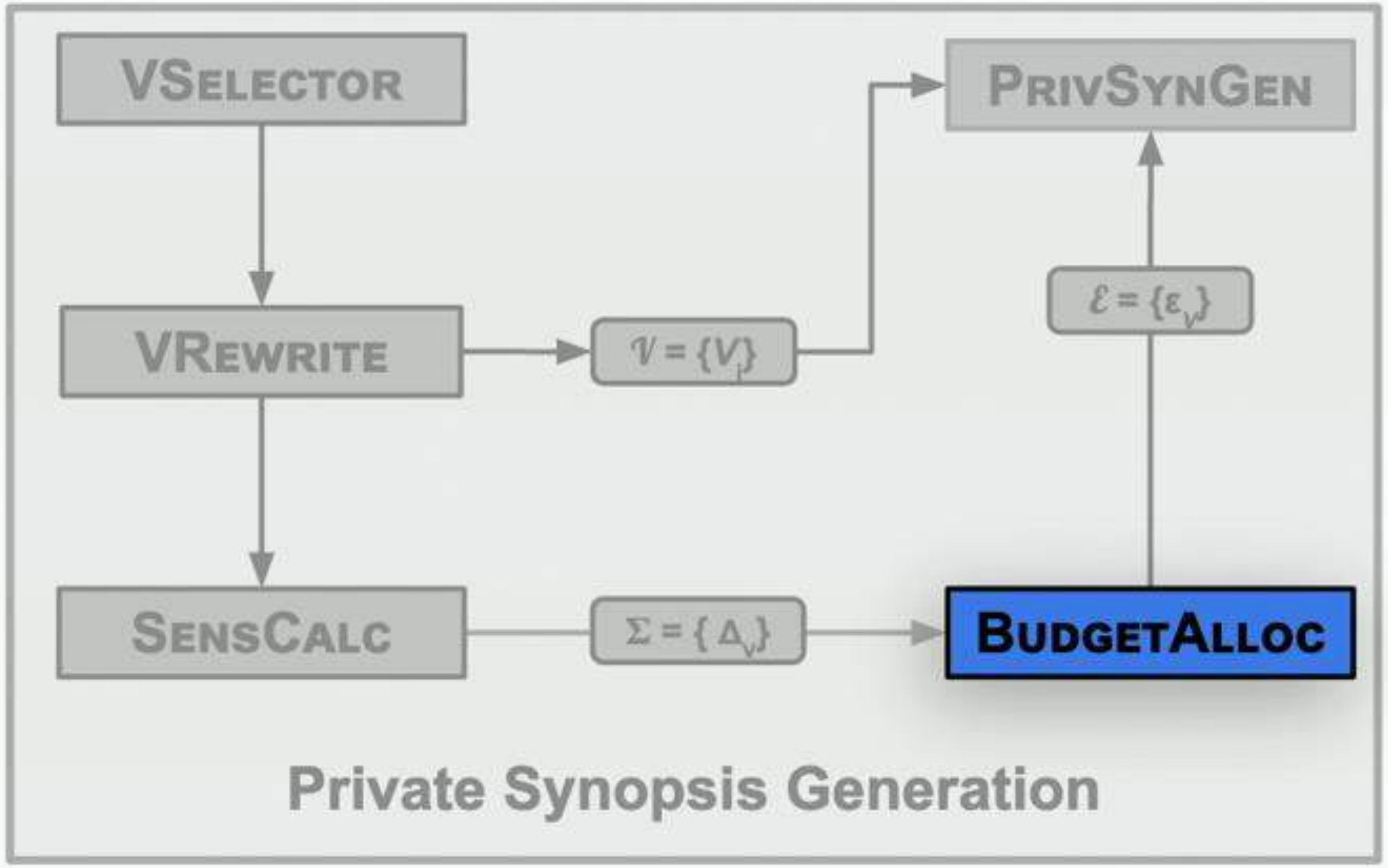
- Generates views based on Q
- Captures join structures of q
- Partitions Q to partial workloads

Sensitivity Calculation

- Compute sensitivity of each view
- Satisfy privacy requirements

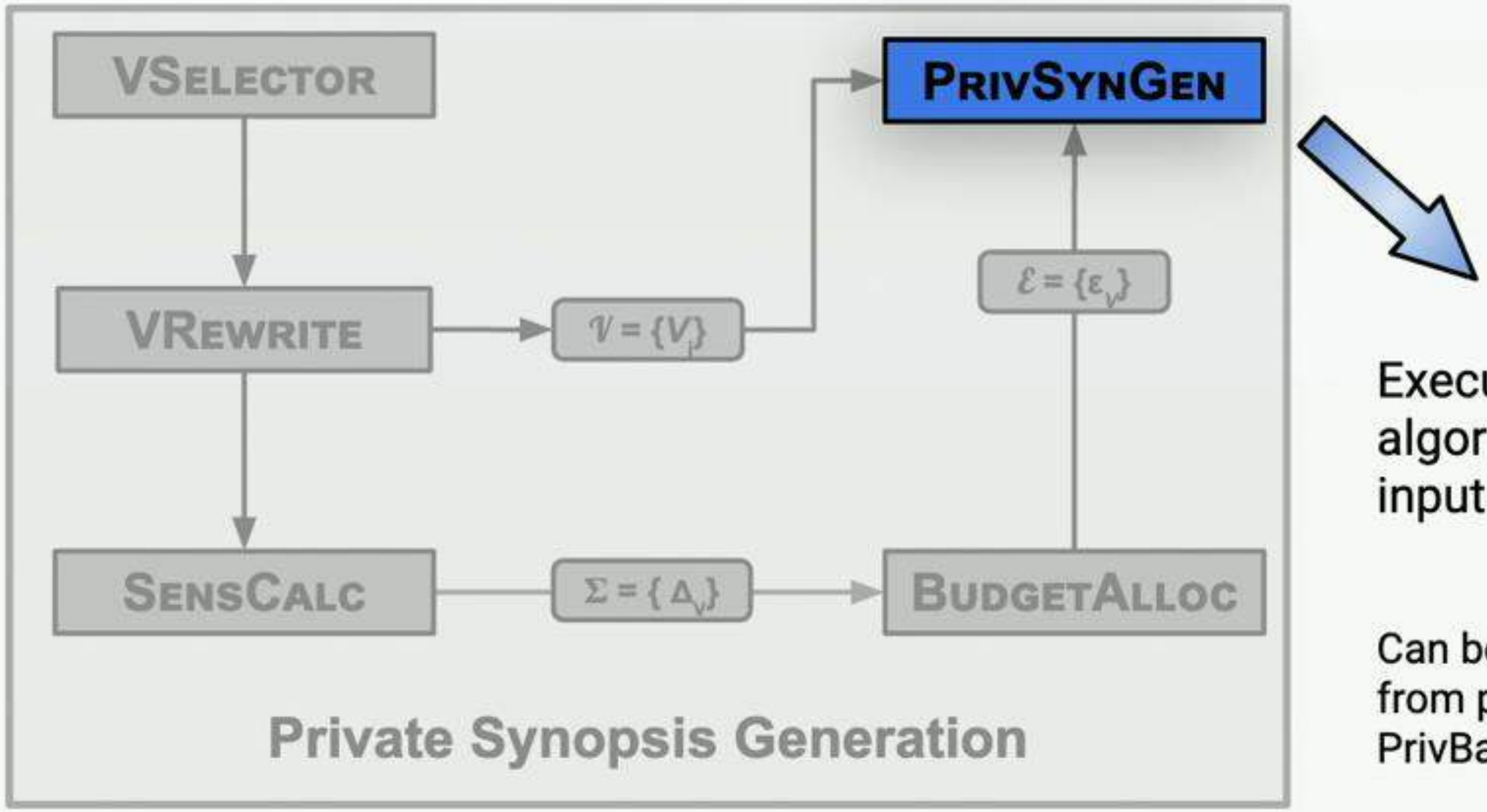


Budget Allocation



Assign privacy budget to each view

Synopsis Generator

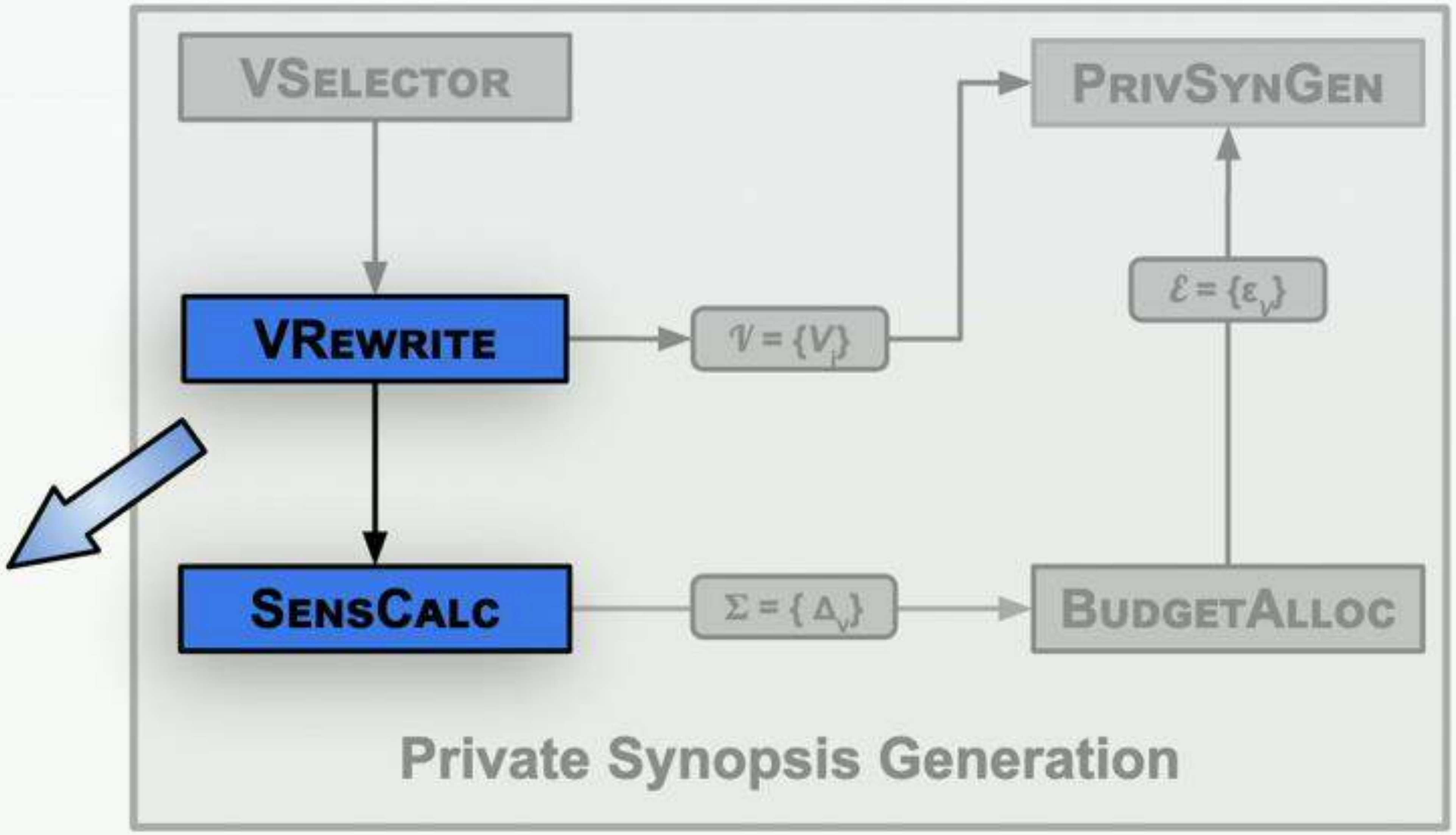


Execute a DP algorithm on the input $(V_i(D), Q_i, \varepsilon_i)$

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

Sensitivity Calculation

- Compute sensitivity of each view
- Satisfy privacy requirements



Sensitivity Revisit

Person

pid	age	...	hid
...	...	...	...
p100	45	...	h02
p101	46	...	h02
...	...	...	...

Household

hid	...	gid
...	...	...
h02	...	g04
h03	...	g04

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

$$S(Q) = \max_{\forall D, D' \in nbrs(D)} \|Q(D) - Q(D')\|_1$$

Sensitivity Revisit

Person

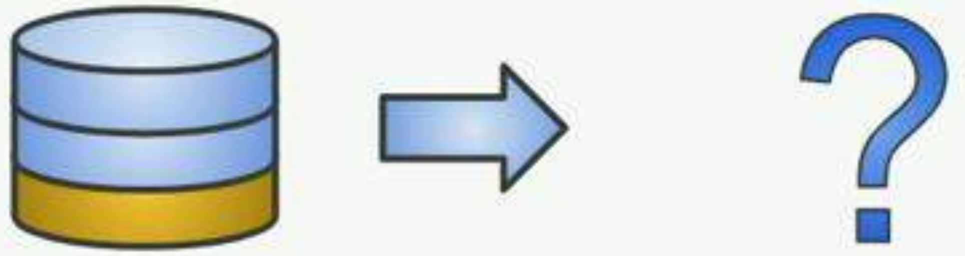
<u>pid</u>	age	...	hid
...	...	...	...
p100	45	...	h02
p101	46	...	h02
...	...	...	...

Household

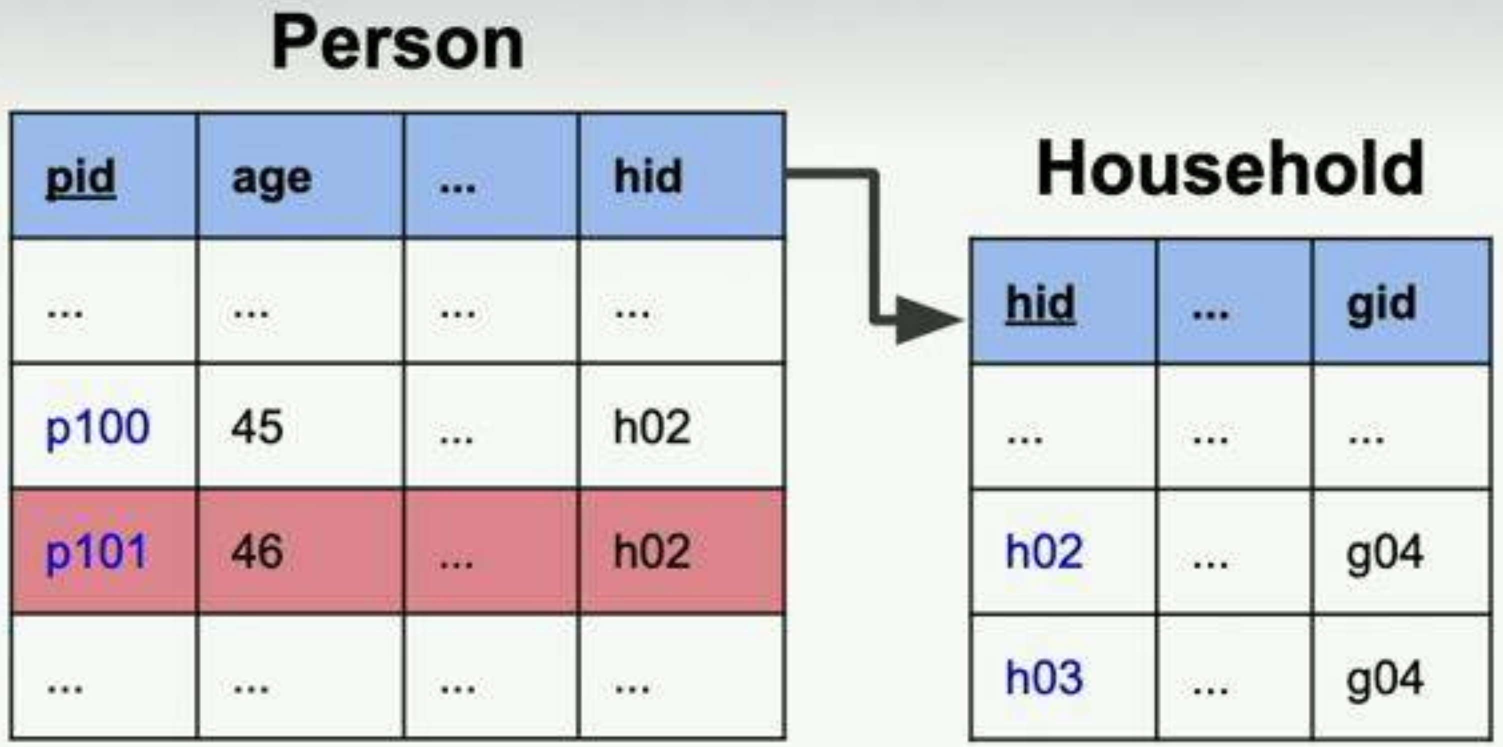
<u>hid</u>	...	gid
...	...	...
h02	...	g04
h03	...	g04

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

$$S(Q) = \max_{\forall D, D' \in nbrs(D)} \|Q(D) - Q(D')\|_1$$



Sensitivity Revisit



Person

<u>pid</u>	age	...	hid
...	...	...	...
p100	45	...	h02
p101	46	...	h02
...	...	...	...

Household

<u>hid</u>	...	gid
...	...	...
h02	...	g04
h03	...	g04

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

D and D’ differ in a row of Person → S(Q) = 1

Neighboring if they differ in one table
What Flex does

Sensitivity Revisit

Person

pid	age	...	hid
...	...	...	...
p100	45	...	h02
p101	46	...	h02
...	...	...	...

Household

hid	...	gid
...	...	...
h02	...	g04
h03	...	g04

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

Person

pid	age	...	hid
...	...	...	...
p100	45	...	h02
p101	46	...	h02
...	...	...	...

Household

hid	...	gid
...	...	...
h02	...	g04
h03	...	g04

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

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

Foreign key constraint → removing one row from Household will affect multiple rows of the Person table

Sensitivity Revisit

Person

pid	age	...	hid
...	...	...	...
p100	45	...	h02
p101	46	...	h02
...	...	...	...

Household

hid	...	gid
...	...	...
h02	...	g04
h03	...	g04

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

Person

pid	**age**	**...**	**hid**
...	...	...	...
p100	45	...	h02
p101	46	...	h02
...	...	...	...

Household

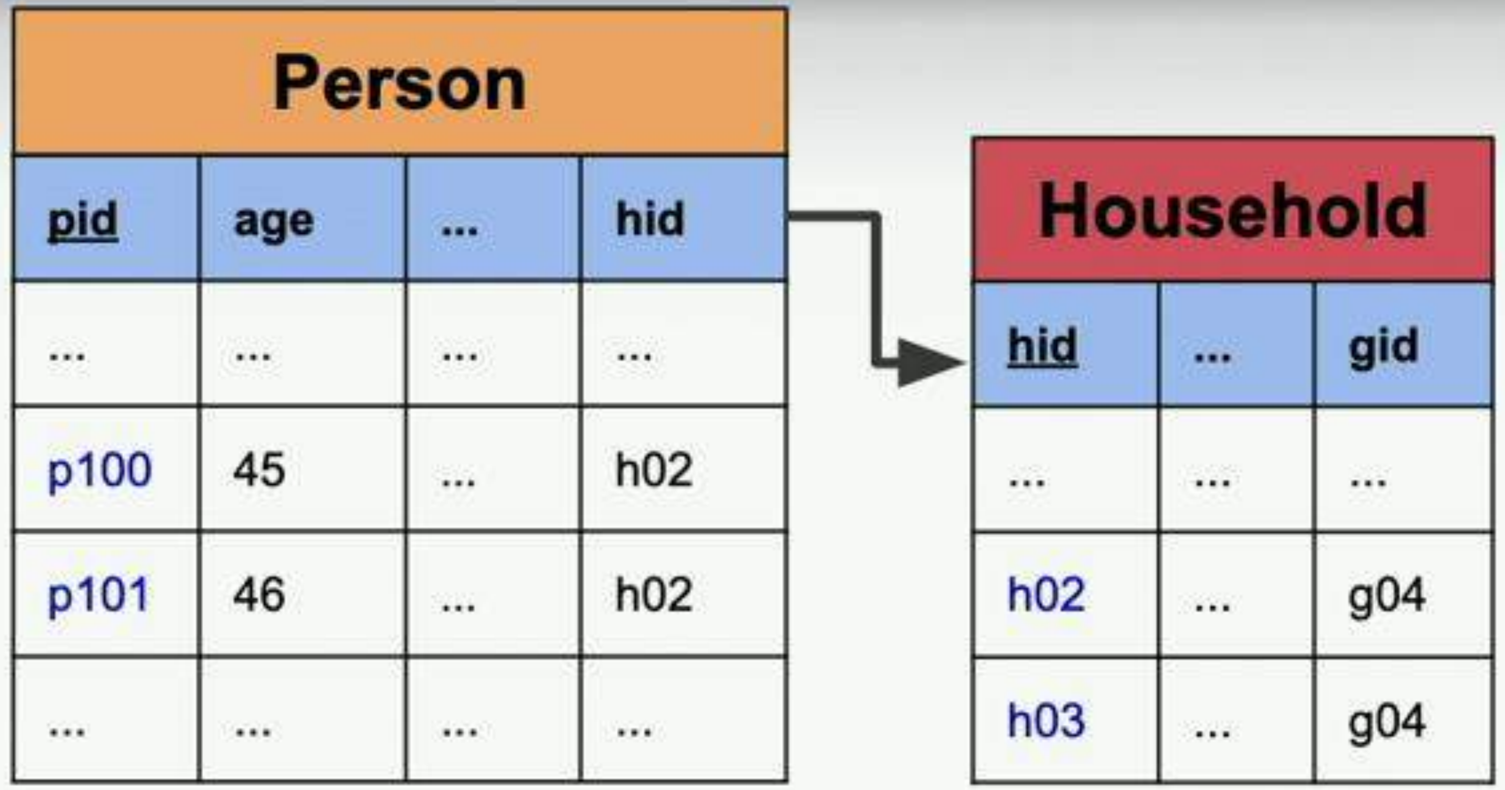
hid	**...**	**gid**
...	...	...
h02	...	g04
h03	...	g04

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

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

Foreign key constraint → removing one row from Household will affect multiple rows of the Person table

Privacy for Relational Data



Person

pid	age	...	hid
...	...	...	...
p100	45	...	h02
p101	46	...	h02
...	...	...	...

Household

hid	...	gid
...	...	...
h02	...	g04
h03	...	g04

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**,ε)

→ Schema needs to be acyclic

Sensitivity Revisit

Person

<u>pid</u>	age	...	hid
...	...	...	...
p100	45	...	h02
p101	46	...	h02
...	...	...	...

Household

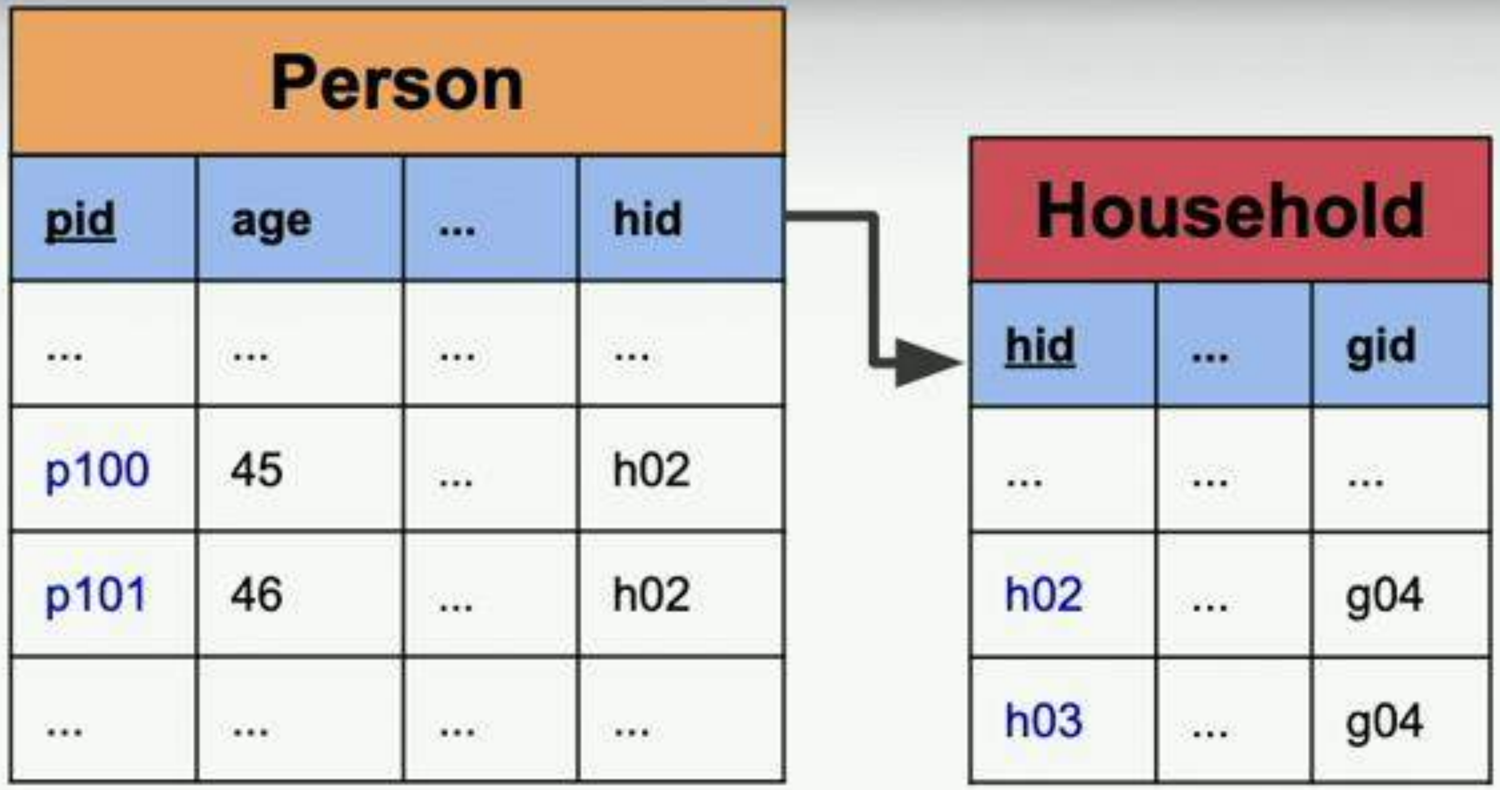
<u>hid</u>	...	gid
...	...	...
h02	...	g04
h03	...	g04

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

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

Foreign key constraint → removing one row from Household will affect multiple rows of the Person table

Privacy for Relational Data



Person

pid	age	...	hid
...	...	...	...
p100	45	...	h02
p101	46	...	h02
...	...	...	...

Household

hid	...	gid
...	...	...
h02	...	g04
h03	...	g04

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

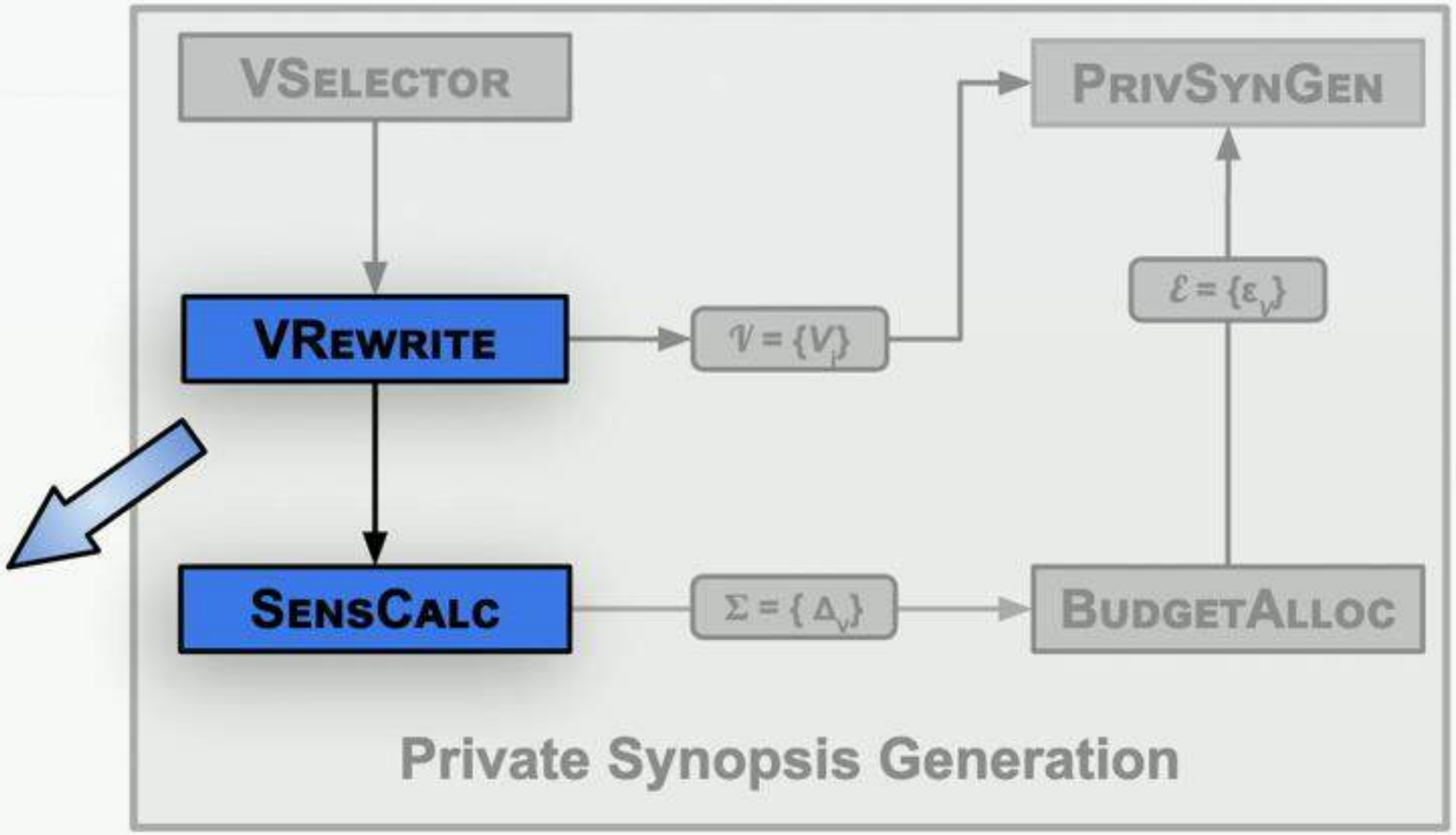
Key constraints: secondary private relations.

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**,ε)

→ Schema needs to be acyclic

Sensitivity Calculation



- Compute sensitivity of each view
- Satisfy privacy requirements

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]	⇨	Rule based sensitivity
● Global sensitivity unbounded in presence of joins	⇨	Truncate “outliers”
● Calculation depends on privacy resolution	⇨	View Rewriting -- automatic policy enforcement

Addressing View Sensitivity

• View is a complex SQL query, estimation is hard [Arapinis 2016]	⇨	Rule based sensitivity
• Global sensitivity unbounded in presence of joins	⇨	Truncate "outliers"
• Calculation depends on privacy resolution	⇨	View Rewriting -- automatic policy enforcement

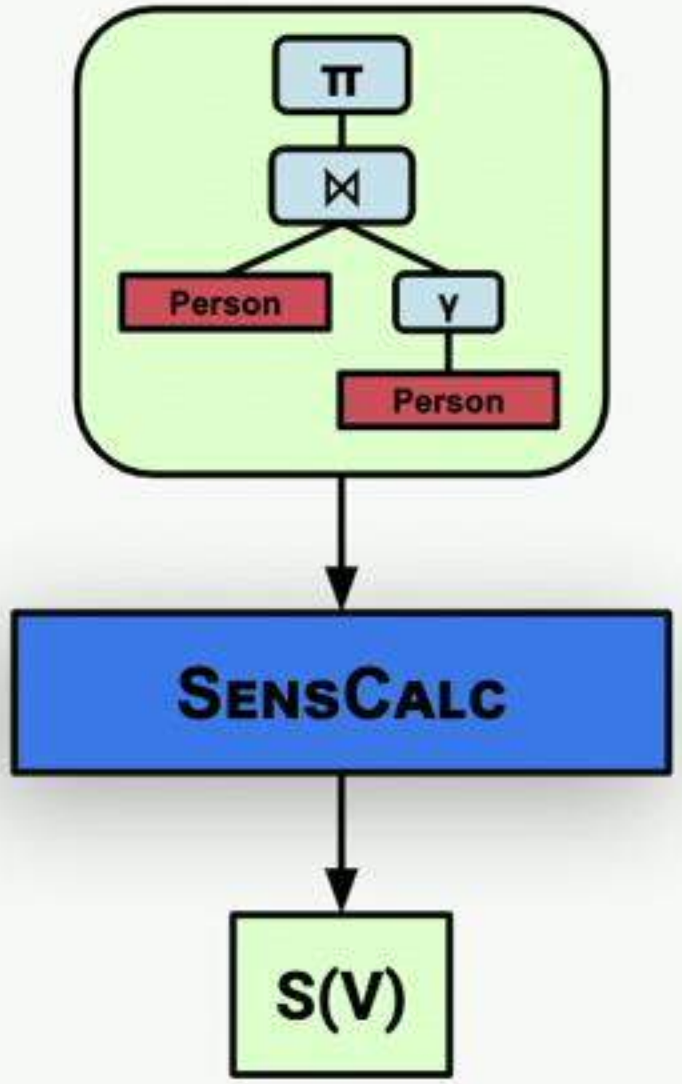
Sensitivity Calculator

Bottom-up rule based algorithm on a query plan

Builds on top of Elastic sensitivity rules

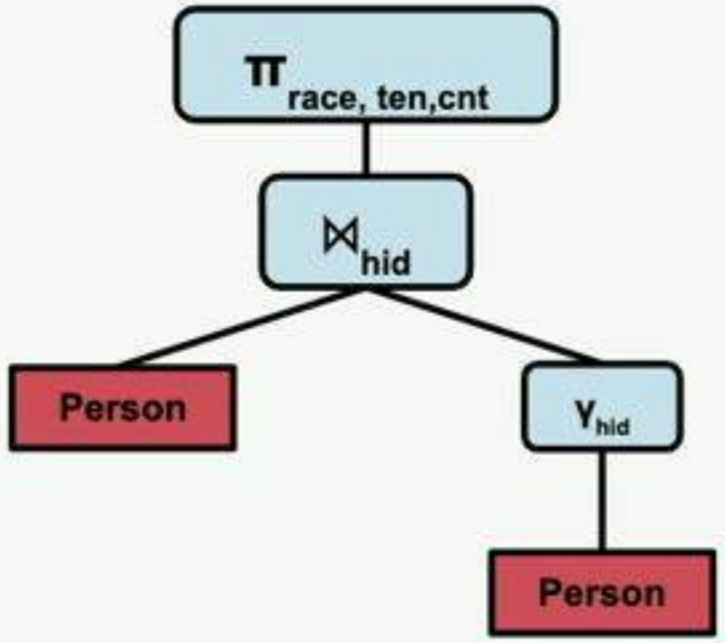
Extends rules via tracking of keys

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



Sensitivity Calculator

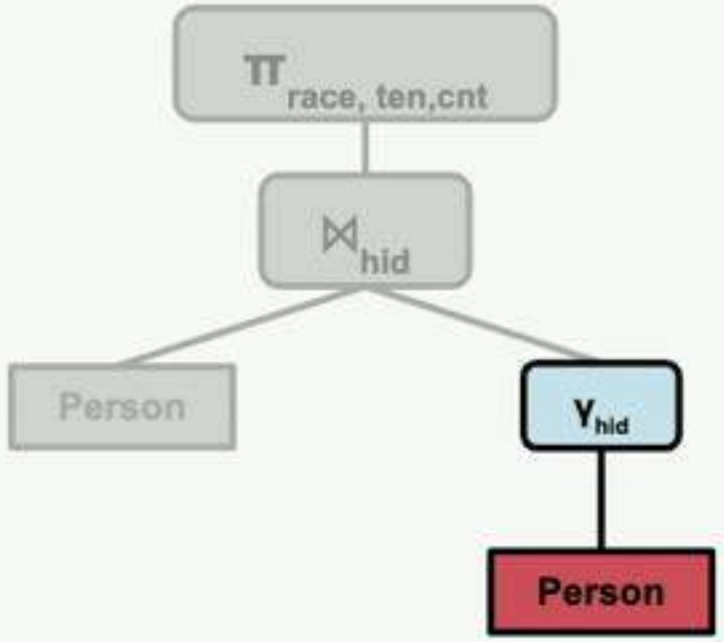
```
V := SELECT relp, race, cnt FROM
        Person P,
        (SELECT count(*) AS cnt, hid FROM Person GROUP BY hid) AS P2
WHERE P .hid = P.hid;
```



V		
relp	race	cnt
...	...	...
head	Asian	3

Sensitivity Calculator

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



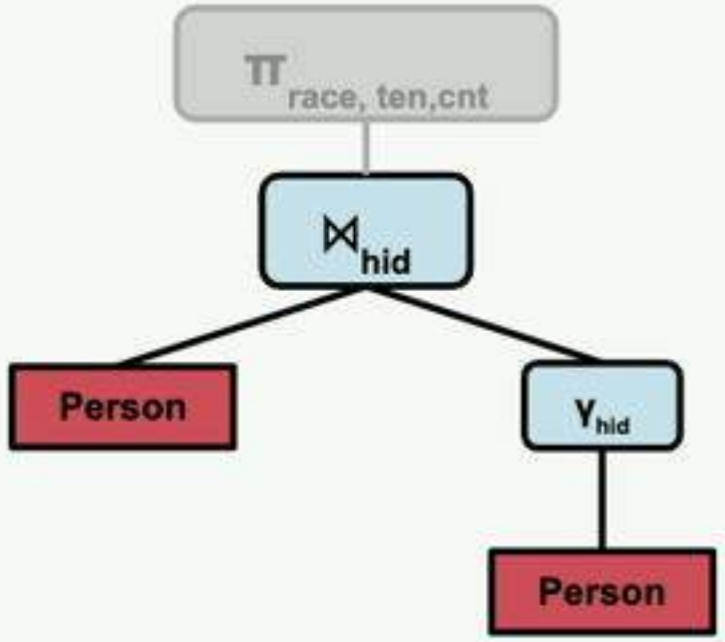
S(Person) = 1

S(R) = S(Person) * 2 = 2

hid becomes key → max multiplicity = 1 in R

Sensitivity Calculator

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



$S(\text{Person}) = 1$

$S(R_1) = S(\text{Person}) * 2 = 2$

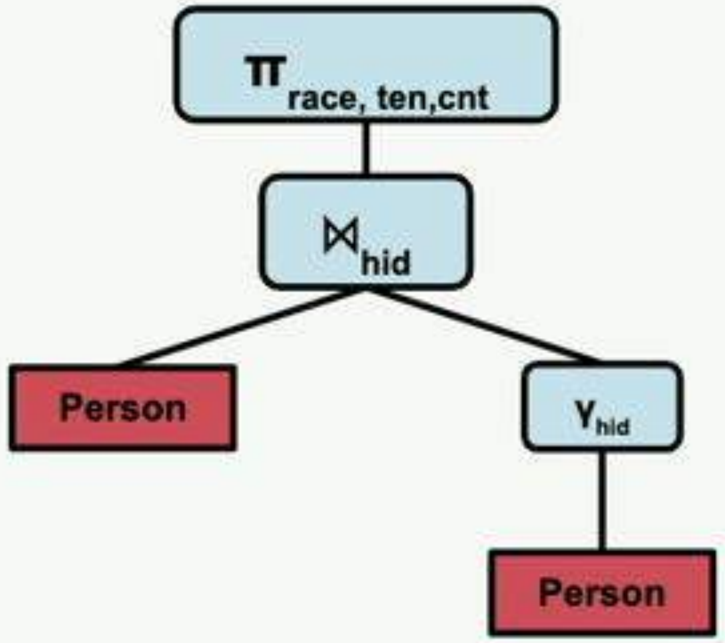
hid becomes key → max multiplicity = 1 in R

$S(R_2) = S(R_1) * F(\text{hid, Person}) + S(\text{Person}) = 2F + 1$

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 = P.hid) AS V;
```



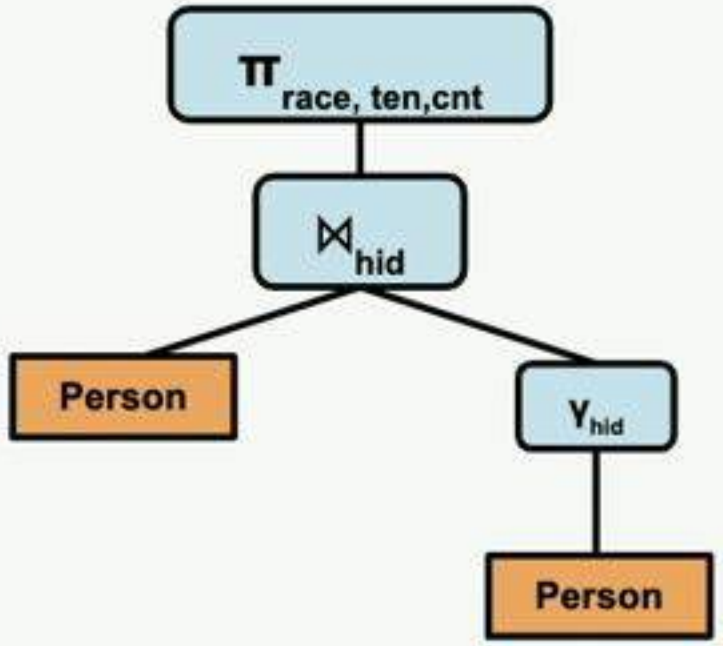
Our rules: S(V) = 2F + 1

Without key tracking: 3F + 2

This difference is only getting larger for more complex views with additional joins.

Sensitivity Calculator

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



What happens for different privacy policy?

We would need a different algorithm to correctly compute it..

View Rewriting — Enforcing Policies

Goal: allow sensitivity calculator to automatically enforce privacy policies (Person, Household)

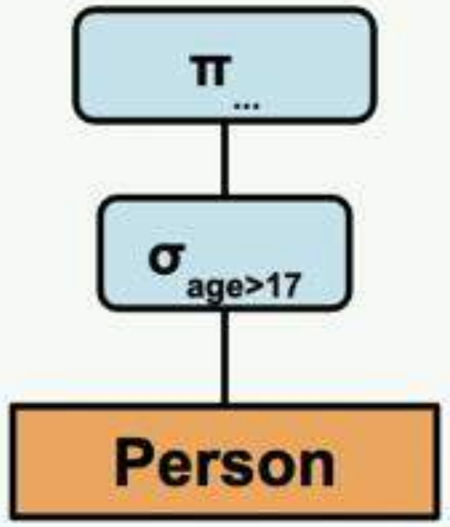
→ via addition of semijoin operators.

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 := **SELECT** * **FROM** PERSON **WHERE** PERSON.AGE > **17**;

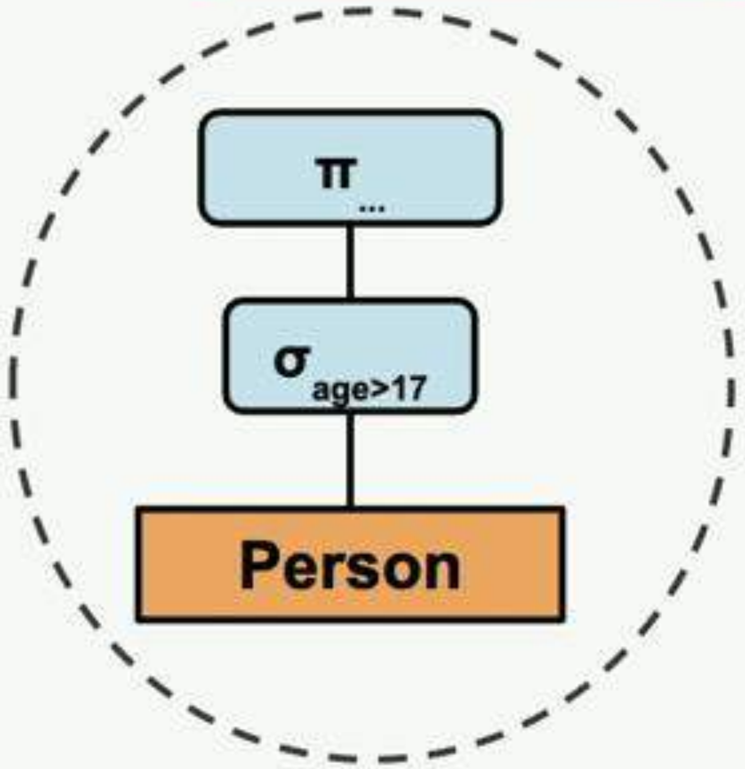
Policy: **Household**



Semijoin Rewrite

V := **SELECT** * **FROM** PERSON **WHERE** PERSON.AGE > **17**;

Policy: **Household**

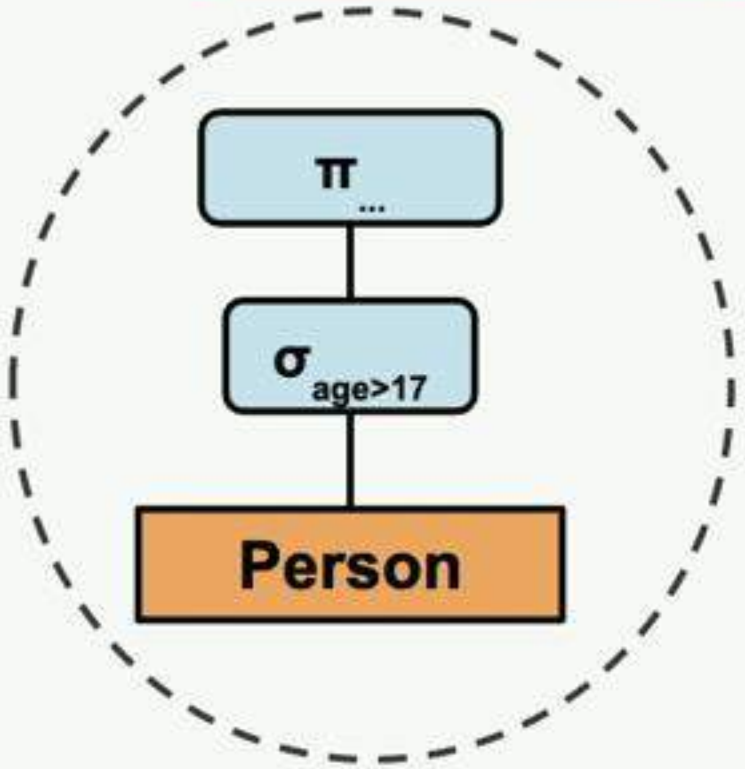


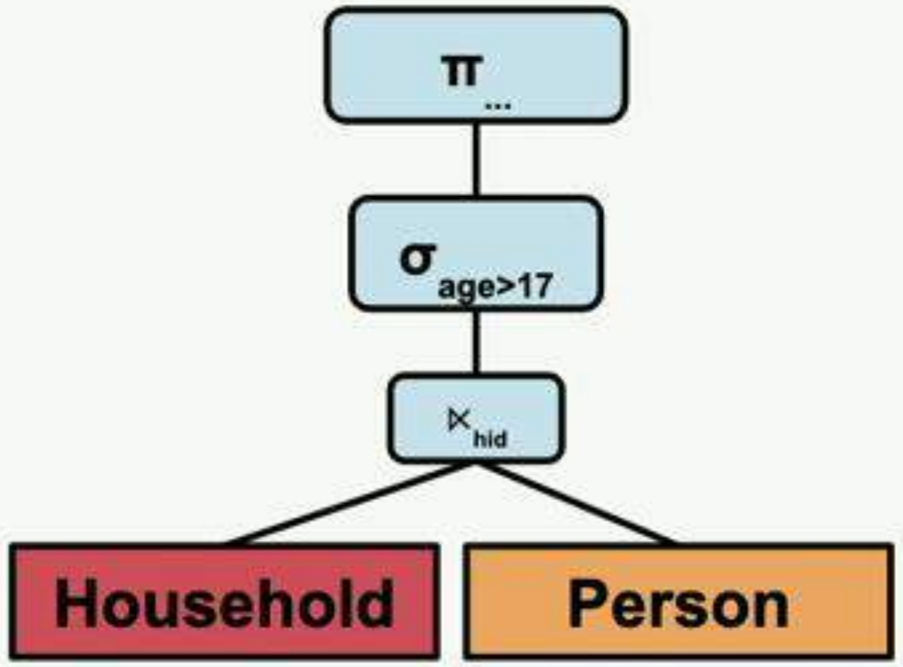
SensCalc computes S(V) = 0 for policy Household

Semijoin Rewrite

V := **SELECT** * **FROM** PERSON **WHERE** PERSON.AGE > **17**;

Policy: **Household**





SensCalc computes S(V) = 0 for policy Household

SensCalc computes S(V) = F for policy Household

View Rewriting — Max Frequency

Goal: allow sensitivity calculator a bound independent on F (i.e., global vs local sensitivity)

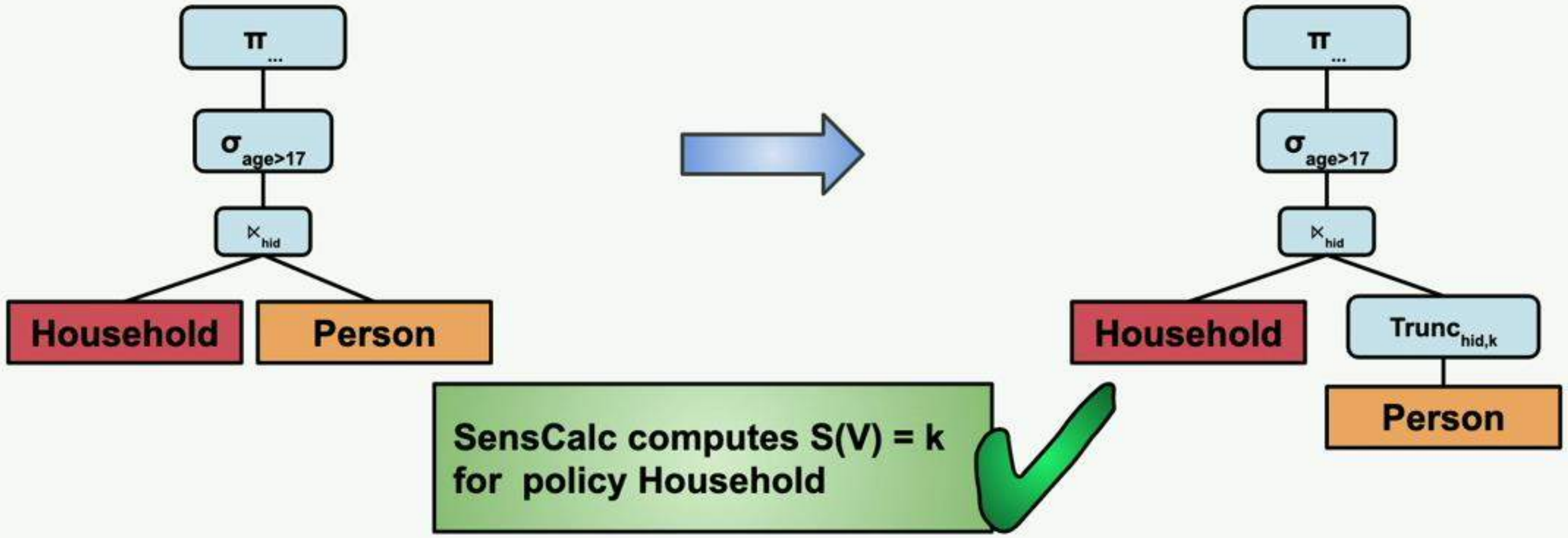
→ via addition of truncation operators.

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.

Truncation Rewrite

V := **SELECT** * **FROM** PERSON **WHERE** PERSON.AGE > **17**;

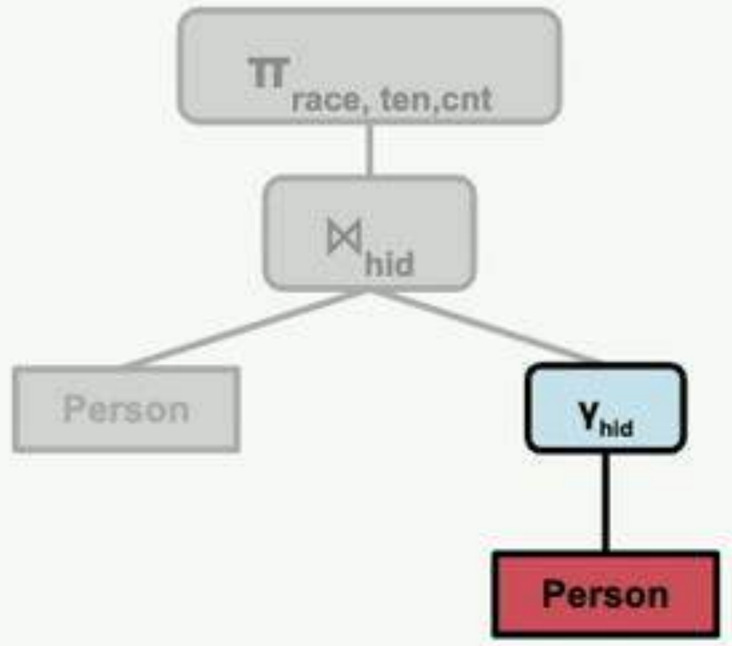
Policy: **Household**



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 = P.hid) AS V;
```



S(Person) = 1

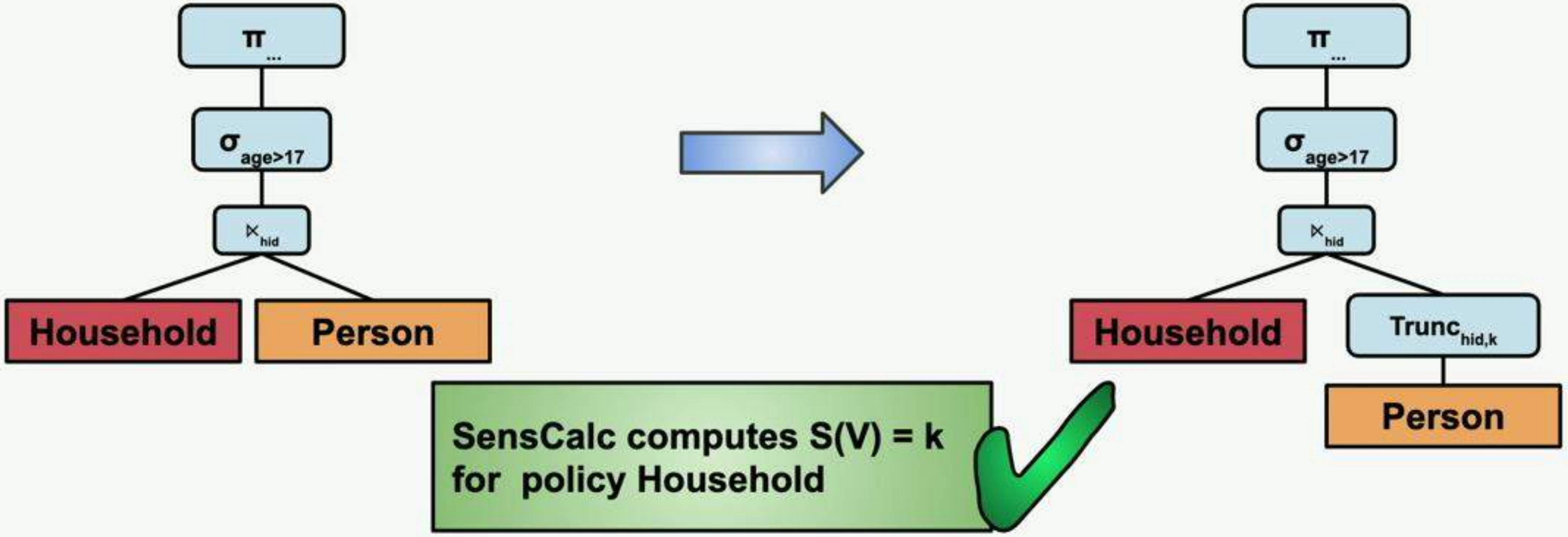
S(R) = S(Person) * 2 = 2

hid becomes key → max multiplicity = 1 in R

Truncation Rewrite

V := **SELECT** * **FROM** PERSON **WHERE** PERSON.AGE > **17**;

Policy: Household



SensCalc computes S(V) = k for policy Household

Overview

U.S. Census Use Case

SF-1

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

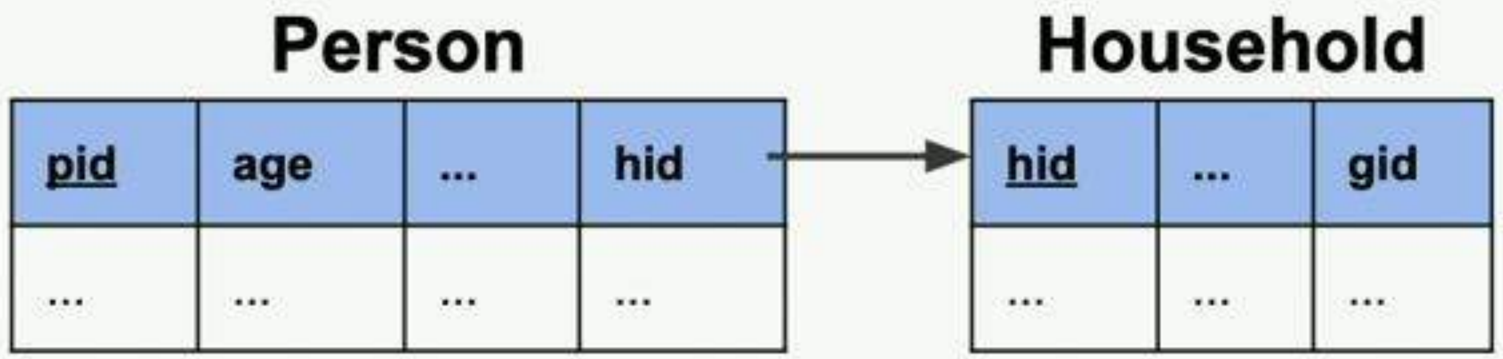
Queries: 3,685 queries describing the SF-1 data release

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

Report relative per query error (10 independent trials)

Private SQL instantiation:

representative: full
psg: w-nnls
pba: wsize



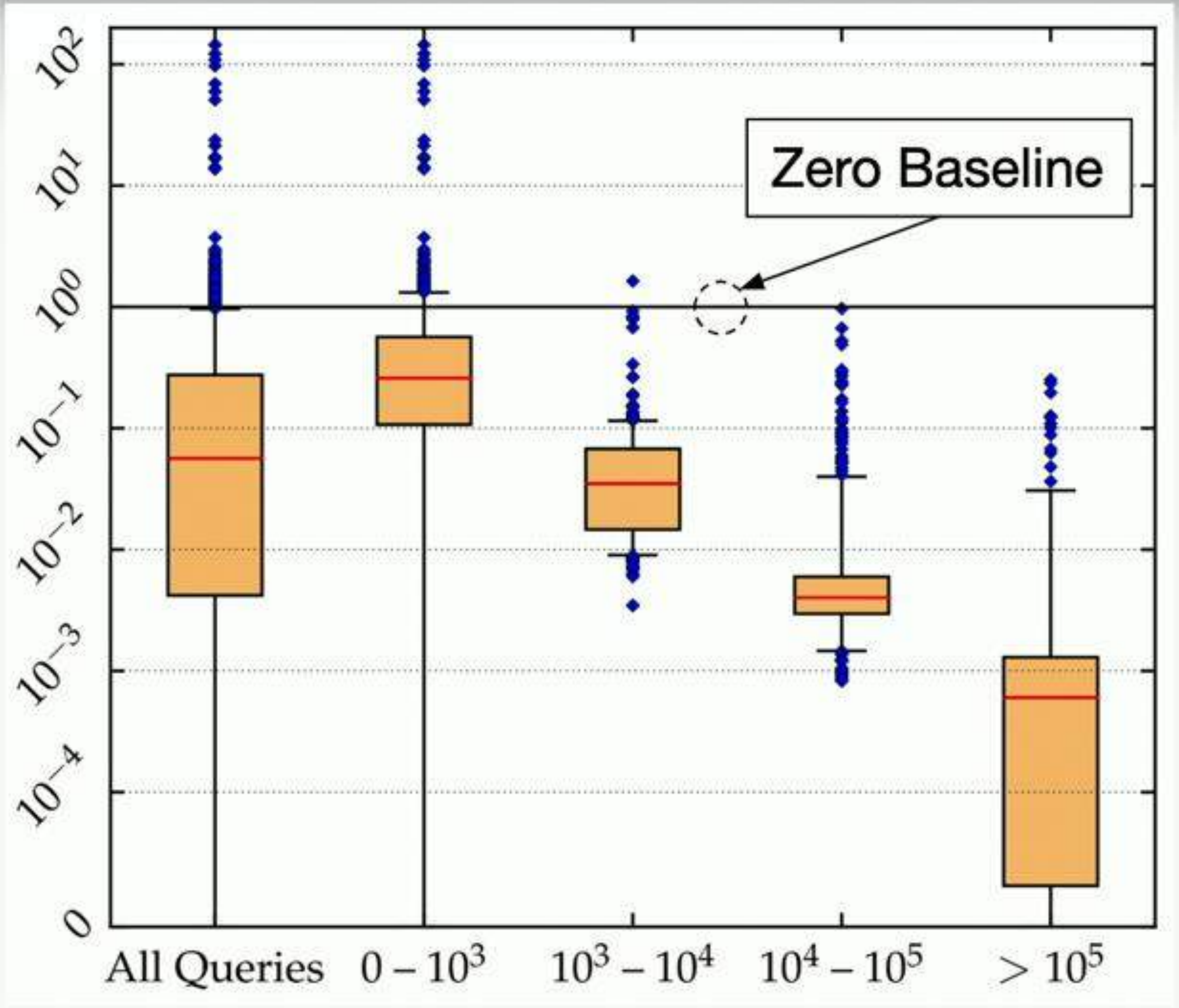
Person

<u>pid</u>	age	...	hid
...	...	...	...

Household

<u>hid</u>	...	gid
...	...	...

Main Results



Policy: Person

Error stratified by true query answer

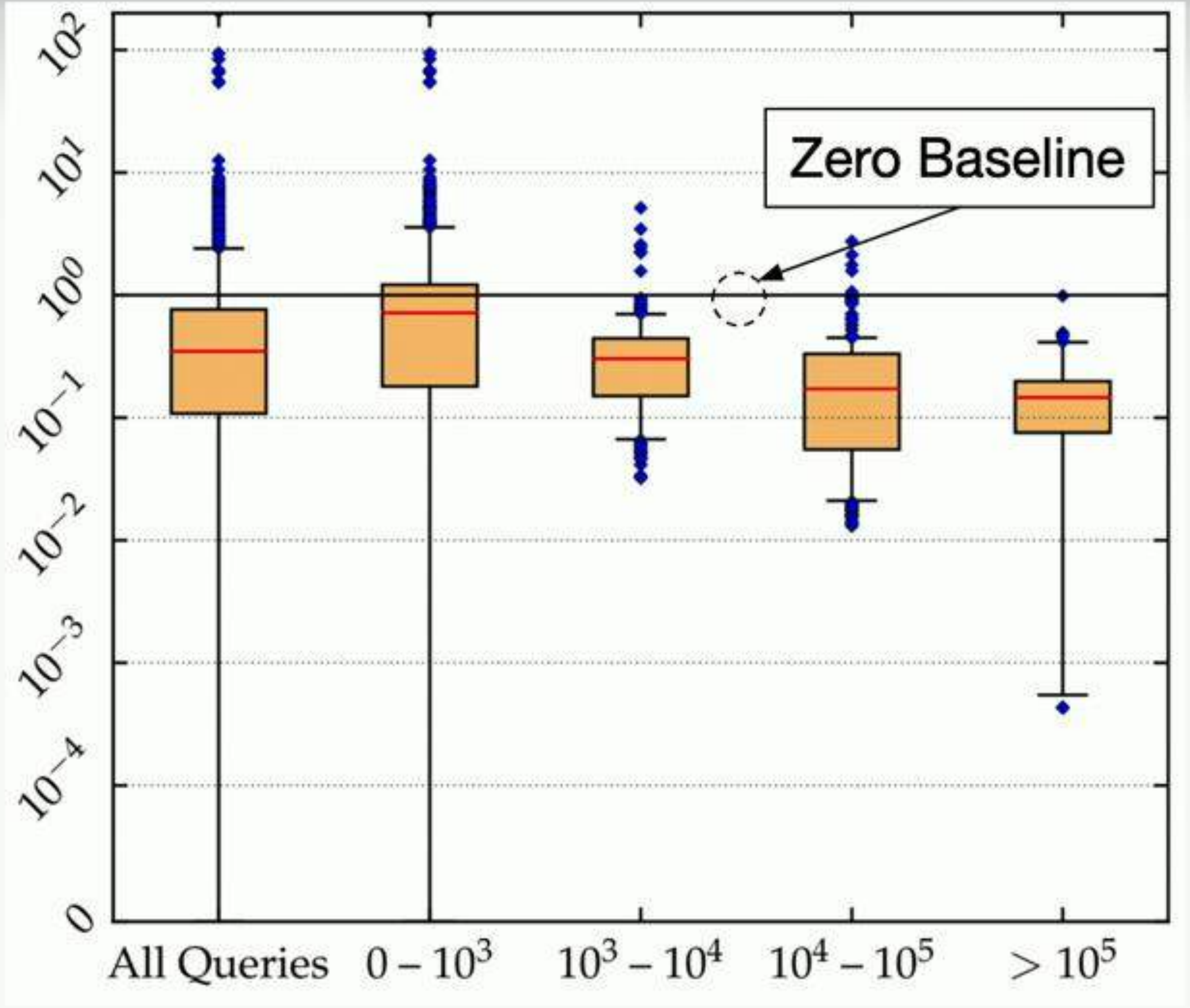
First group all queries

→ 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

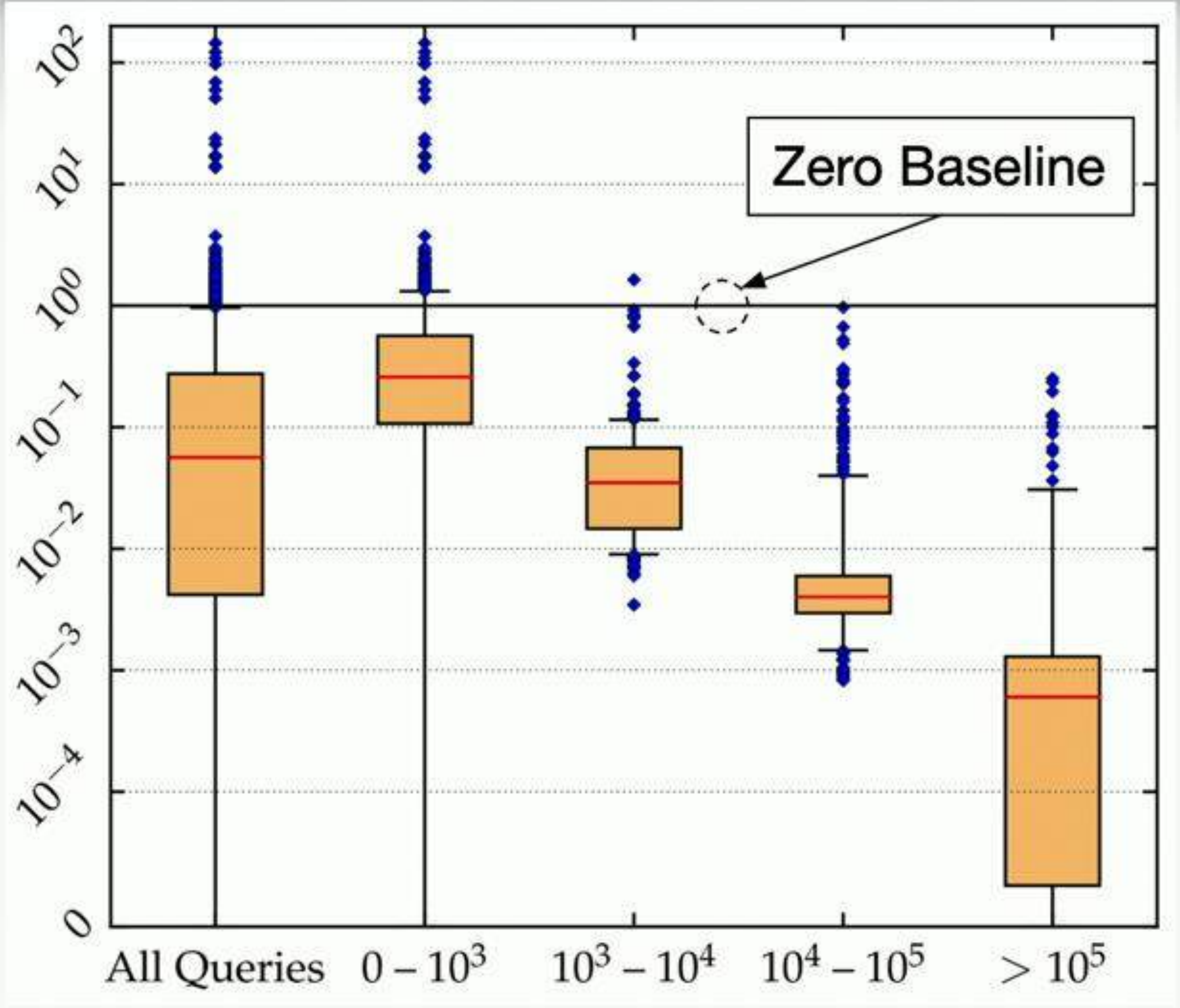
Error stratified by true query answer

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

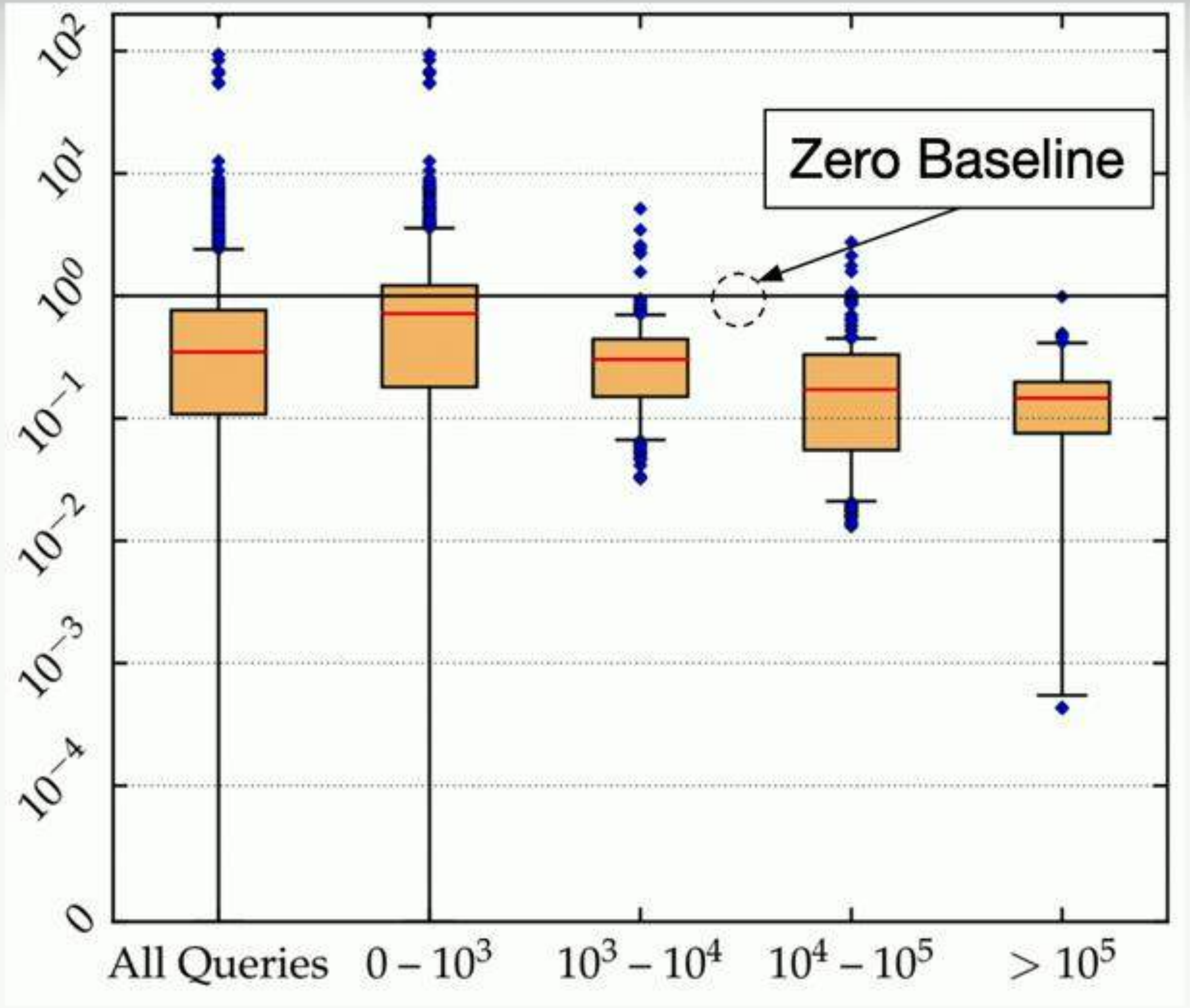
First group all queries

→ **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

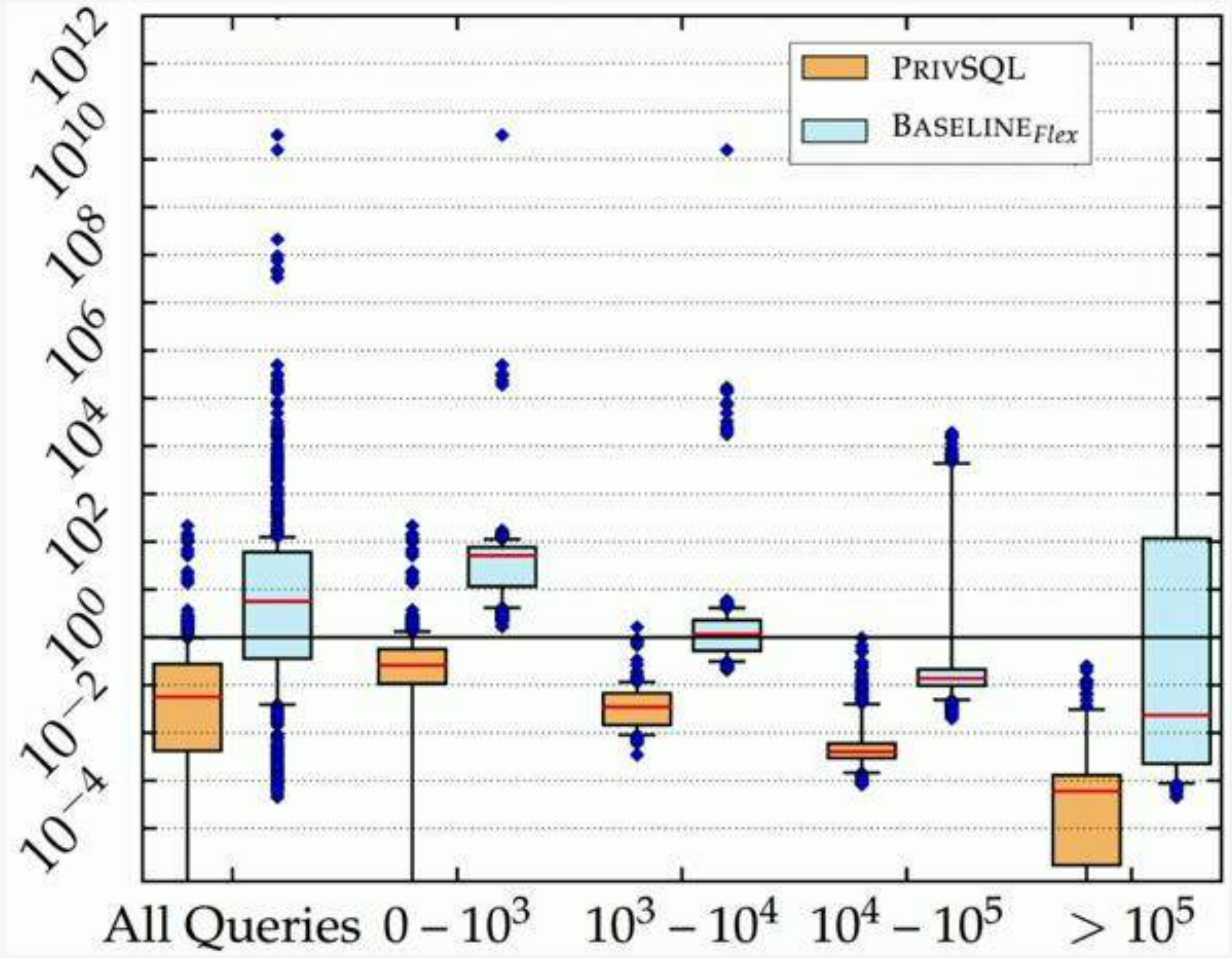
Comparison w/ baseline adapted from prior work [Flex]

→ 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 with Prior Work

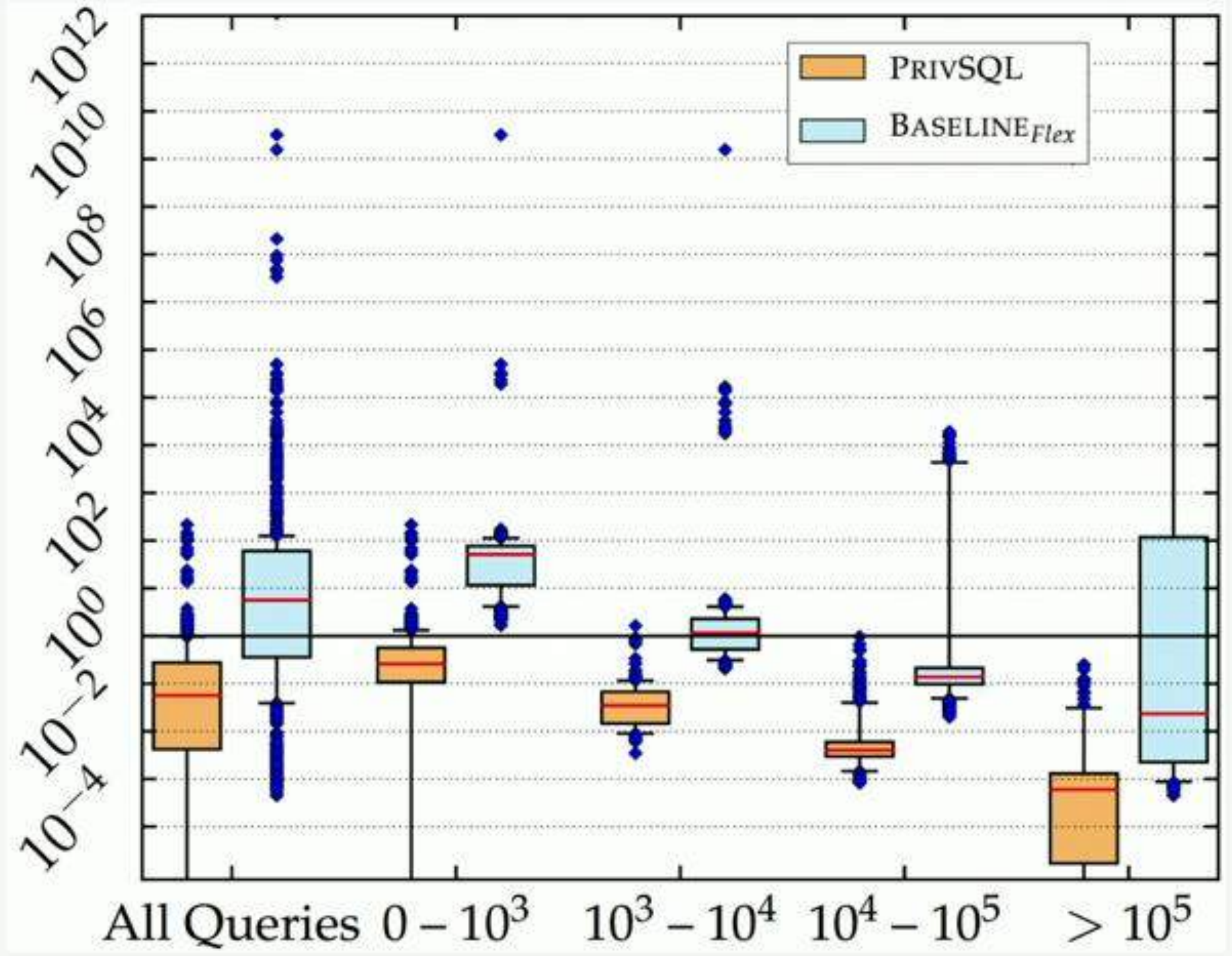
Comparison w/ baseline adapted from prior work [Flex]

→ 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

Ongoing

- Policies that extend to multiple primary private relations
- Support for aggregate queries like **AVG**(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, ε) [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 with Prior Work

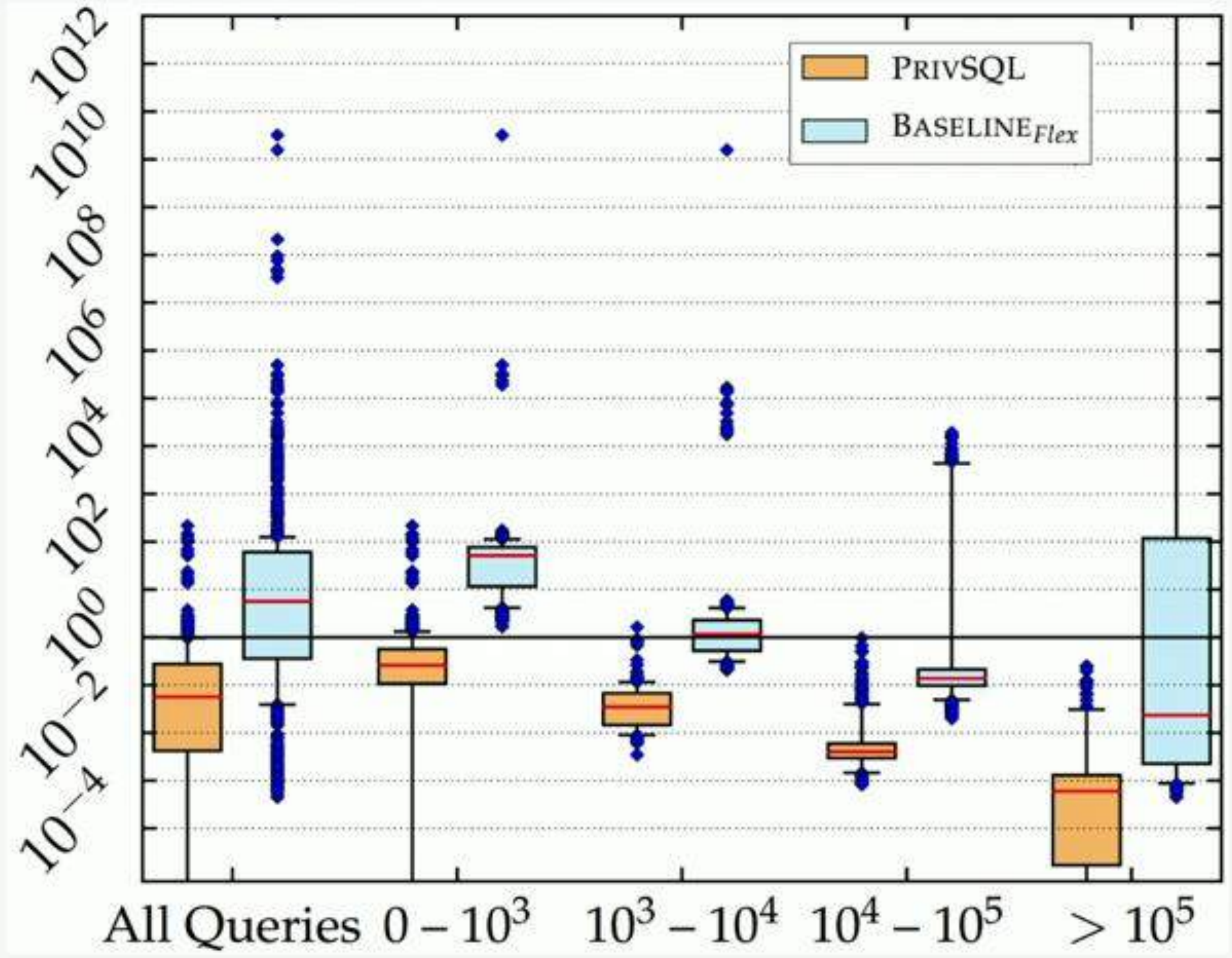
Comparison w/ baseline adapted from prior work [Flex]

→ 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 **AVG**(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, ε) [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

Thank You!

Comparison with Prior Work

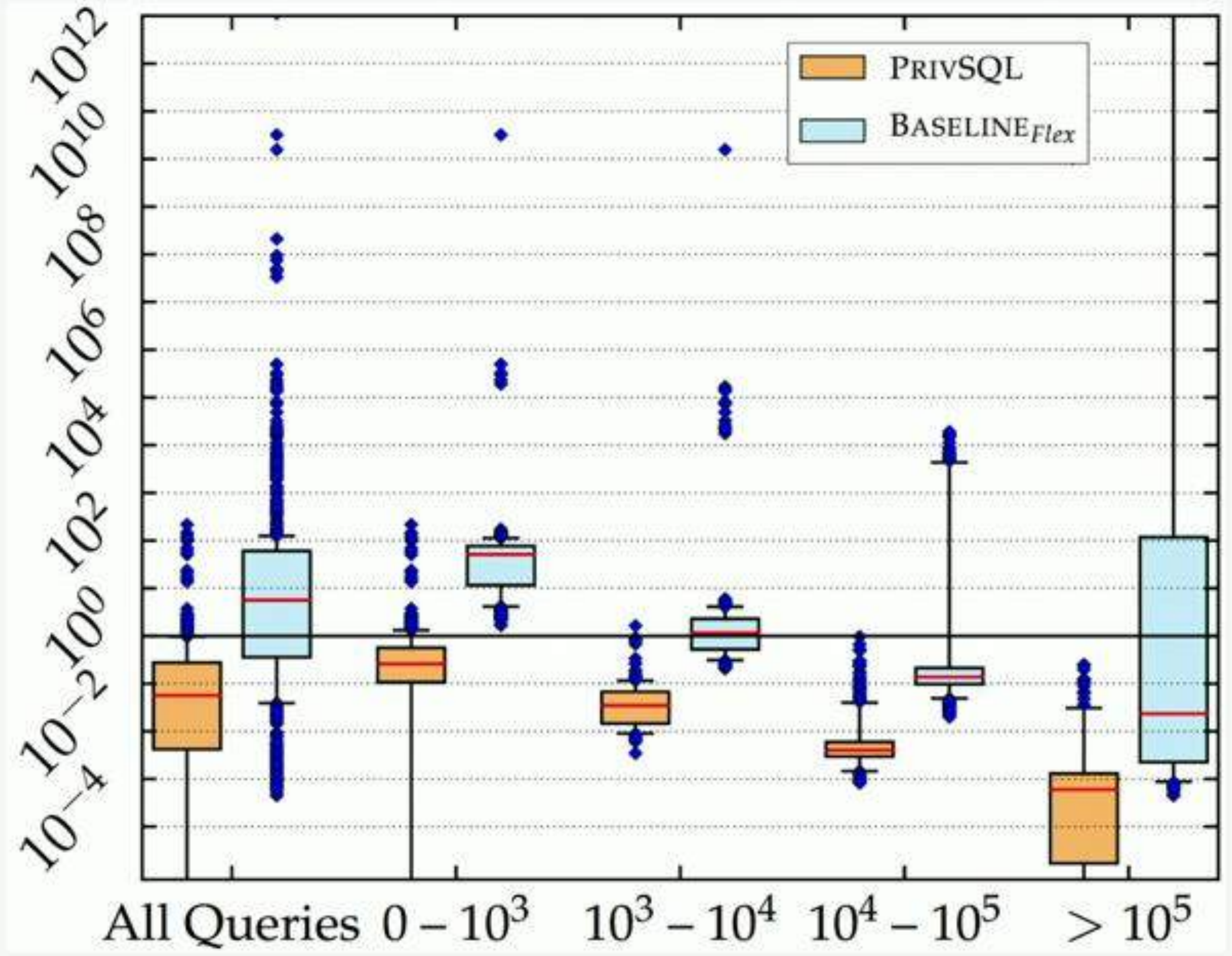
Comparison w/ baseline adapted from prior work [Flex]

→ 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 **AVG**(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, ε) [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