Private Approximate Query over Horizontal Data Federation*

Ala Eddine Laouir
Université de Lorraine, CNRS and Inria
Nancy, France
ala-eddine.laouir@loria.fr

Abdessamad Imine
Université de Lorraine, CNRS and Inria
Nancy, France
abdessamad.imine@loria.fr

ABSTRACT

In many real-world scenarios, multiple data providers need to collaboratively perform analysis of their private data. The challenges of these applications, especially at the big data scale, are time and resource efficiency as well as end-to-end privacy with minimal loss of accuracy. Existing approaches rely primarily on cryptography, which improves privacy, but at the expense of query response time. However, current big data analytics frameworks require fast and accurate responses to large-scale queries, making cryptography-based solutions less suitable. In this work, we address the problem of combining *Approximate Query Processing* (AQP) and *Differential Privacy* (DP) in a private federated environment answering range queries on horizontally partitioned multidimensional data. We propose a new approach that considers a data distribution-aware online sampling technique to accelerate the execution of range queries and ensure end-to-end data privacy during and after analysis with minimal loss in accuracy. Through empirical evaluation, we show that our solution is able of providing up to 8 times faster processing than the basic non-secure solution while maintaining accuracy, formal privacy guarantees and resilience to learning-based attacks.

1 INTRODUCTION

The extensive reliance of individuals on software solutions in daily and professional life has led to an exponential growth of data collected by companies, corporations, government organisations, and even hospitals. These vast mines of data, if carefully and efficiently analysed, can provide valuable insights that guide decision-making and business development. In large-scale studies and research, the analysis must be conducted on several data sources to obtain meaningful conclusions. An example of such a case is during a pandemic, where many hospitals jointly conduct studies to have a global view of the contagion problem.

One of the most commonly used tools to analyse and explore these huge volumes of data are OLAP tasks, where various aggregation queries (`SUM`, `COUNT`, etc.) can be issued to learn existing patterns and trends within the data. These aggregation queries may seem simple, but they are very time-consuming in big databases. The analysis of data from multiple data providers comes with two main challenges: privacy and resource/time efficiency. The privacy issue arises from the fact that this data is personal and sensitive to individuals, and sharing it with other parties can be very harmful. Many regulations and restrictions like GDPR are imposed by governments on how to process and share such sensitive data. In the case of a federated environment, where a joint study requires the collaboration of many data providers, data sharing is highly restricted. Each data provider must ensure the security and privacy of the data collected from their users during and after the analysis.

To satisfy the requirement of end-to-end privacy, many solutions have been proposed in the literature, and most of them rely on cryptography to ensure there is no data leakage during the exchange and query evaluation. Secure multiparty computation (SMC) solutions appear to be a prominent solution in federated environments [7, 8]. Others use oblivious operations [6] or secure hardware [17, 35, 43] so that, during query evaluation, each data provider can maintain the confidentiality of their data. Additionally, for securing the final result of any OLAP query, Differential Privacy (DP) [15] is generally considered the gold standard by government and private institutions [1, 10, 16, 37]. Due to its strong formal confidentiality guarantees, DP allows individuals to deny their participation in the database. These query evaluation solutions in a federated environment meet end-to-end security and privacy requirements. However, what they have in common is their reliance on encryption. This causes a huge processing time overhead; and for time-sensitive tasks, utility is measured by both accuracy and speed. They certainly address the privacy issue, but they are time and resource consuming.

The issue of reducing query response time has been widely addressed in the literature, through the need to obtain Approximate Query Processing (AQP). Existing AQP methods can be classified into two types, online approximation and offline synopsis creation. In online approximation, there is *Online Aggregation* based solutions [21, 25, 34] that provide fast and reliable approximation of the query continuously, and other solutions based on applying *online sampling* to reduce the processed data and obtain an approximation from a sample [19, 36, 42]. In offline synopsis creation, views are generated offline using query workloads or/and data statistics [2, 3, 12]. In this area of research, the main focus is on efficiency, but privacy has not been considered.

In our work, we address the challenge of answering OLAP aggregation range queries in a federated environment, while preserving end-to-end privacy and improving resource and time consumption for query processing. Our solution relies heavily on DP to secure collaboration and end results, and ensure no information leaks. To speed up queries, we implement a cluster-based sampling method using a well-known statistical estimator that provides accurate estimates for range queries (such as `SUM` and `COUNT`) while processing minimal data portions. While existing systems ensure either privacy or speedup for query approximation, to the best of our knowledge, our solution is the first to offer speedup over plain-text execution with end-to-end privacy in a federated environment. Our main contributions can be listed as follows:

(1) Definition of a lightweight collaboration method that determines optimal sampling decisions for data providers to maximize accuracy without access to their full datasets or information leakage.
(2) Introduction of data distribution-aware cluster sampling method with DP guarantees for individual privacy.

*This work is funded by DigiTrust (http://lue.univ-lorraine.fr/fr/article/digitrust/).

(3) Meticulous integration of DP at every step with minimal loss of precision.
(4) Extensive experimentation to empirically validate the performance of our approach in terms of accuracy, time efficiency and resilience against learning-based attacks.

Roadmap. The paper is structured as follows: Section 2 reviews some existing works. Section 3 introduces the notions used throughout our paper. Section 4 gives a description of the problem solved by our approach. Section 5 presents our proposed solution in detail. The extensive evaluation of our approach is given in Section 6. In Section 7, we discuss the limitations/extensions of our solution and we conclude in Section 8 by giving some future works.

2 RELATED WORKS

Due to the increasing size and distribution of databases, querying such vast volumes for analytical purposes, quickly and without revealing sensitive information, has become a challenge. Here, we describe the state-of-the-art related to our work.

Approximate Query Processing (AQP). As the quality of a query is based on its accuracy and response time, especially for time-sensitive tasks like OLAP [38] and Business Intelligence, approximating the query offers the best way to strike a balance between these two quality factors.

In the early 1990s, [21] proposed a new interactive method for query processing that provides a quick initial answer with a certain error, refining it as processing continues. Other works followed in this direction [25, 34, 39, 40], each enhancing specific aspects of the method by including support for *group by* or propose parallel and distributed versions. Another research direction focuses on processing a small subset of the original data, thereby reducing query run-time. In [31–33, 36], uniform row-level random sampling is applied online before query processing. Although row-level sampling may improve processing time for complex queries, it can introduce overhead and slow down queries that require a full table scan [20] (e.g. Bernoulli sampling). To avoid such overhead, the solutions from [2, 3, 12] create the samples offline. Cluster sampling, also referred to as *page-sampling* [20], is utilized to speed-up aggregation queries in big databases. Methods in [5, 19, 42] use this sampling in the context of Hadoop Map-Reduce framework[1], as it proves to be fast and I/O efficient compared to row-level sampling.

Federated query answering. Data is often distributed across multiple locations (e.g. data providers like hospitals and companies) and the collaboration among all parties is necessary to answer range aggregation queries. But for privacy and security reasons, each data provider cannot disclose its data to third parties.

Some solutions rely on secure hardware modules (i.e. *enclaves*), in which all sensitive code and data are processed. Methods in [4, 17, 43] focus on aggregation queries in this setting, and [17, 43] use intel's SGX for secure processing. These solutions are generally efficient, but their reliance on trusted hardware and weakness to side-channel attacks constitute a limitation. Recently in [35], the notion of *Differential Obliviousness* was used to mitigate the risk of side channel attacks.

Other recent works presented Secure Multiparty Computation (SMC) query processing engines [6–8]. These engines enable data providers to respond to OLAP queries securely by joining data with end-to-end privacy. Differential Privacy (DP) is used to perturb the final results, thereby mitigating any inference attacks based on the results. While these solutions incur computational overhead, [8] introduced online random sampling to improve secure computing performance by reducing the size of shared data for query processing. In [11], sampling is performed offline to create a synopsis to further improve performance. Another solution [26] focused on reducing the cost of SMC operation thus obtaining significant improvement in performances. All of these SMC (or *enclaves*)-based protocols are encryption-based, which prevents them from outperforming plain-text query execution. Even with significant improvements introduces in the past years, on real world big tables they still expensive for real-time queries [26].

To highlight the scale of this problem, we performed a simulation[2] using a synthetic Adult dataset [9], horizontally distributed on 4 data providers as a federated environment. We ran a set of random range queries, which are the type of queries we focus on. For the query processing, we considered two solutions using SMC: (i) sharing the rows and collectively evaluating the query; and (ii) evaluating the query locally and only sharing the results and computing the final result.

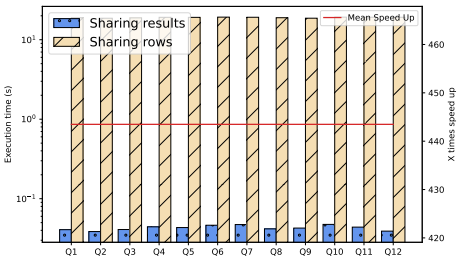

Figure 1: Runtime cost of data sharing in SMC.

We measured the time required to share the rows/results in SMC. The results in Figure 1 show that sharing only local results incurs an insignificant overhead of 0.04 seconds. On average, this is less than 440 times the time required for row sharing in SMC. Additionally, the cost of sharing only results remains constant and independent of the dataset, whereas the cost of sharing rows will increase with larger tables.

In our work, we propose a framework to approximate query processing in a federated environment, enabling accelerated query execution compared to plain text execution while ensuring end-to-end DP guarantees.

3 PRELIMINARIES

In this section, we give the notation and explain briefly notions used throughout the paper.

Data model. In a tabular database T defined over a set of n *dimensions* (or *attributes*) $D = \{d_1, d_2, ..., d_n\}$, each individual is a *row* with values on each dimension. We assume that each dimension d is associated with a domain $|d|$ containing discrete and totally ordered values. The size of the domain is $\|d\|$. For performance purposes during *online analytics* tasks, the table T is transformed into a *multidimensional data* (or a *count tensor*) T^a of dimensions $D^a \subset D$, which has an attribute *Measure* storing the number of *aggregated* rows of T. Figure 2 illustrates how to construct a

[1] https://hadoop.apache.org

[2] https://github.com/AlaEddineLaouir/Federated-Range-Queries.git

count tensor T^a from table T by aggregating dimension *Service*. For simplicity, we use term "table" for "tabular data" and "count tensor".

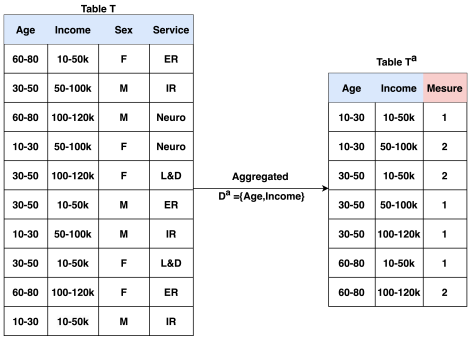

Figure 2: Count tensor

Queries. To analyze and extract insights from these tables, the analyst can issue aggregation queries, helping to explore the data and gain a general understanding of patterns and trends. In this work, we consider a *range query* Q defined as:

`SELECT Aggregation FROM Table WHERE Range`, where:

- `Aggregation` is `COUNT(*)` or `SUM(Measure)`.
- `Range` is a set of intervals $r_d = [l_b^d, u_b^d]$ on each dimension $d \in D^Q$ where $D^Q \subseteq D$ in `Table`, such that $l_b^d \leq v \leq u_b^d$ for every value $v \in |d|$.

In our work, we focus on `COUNT` and `SUM` queries because they are used in several analytics applications. For instance, in a big database aggregating per-stock order data for the NASDAQ exchange, these queries are typically used to analyze order data from past days. Additionally, aggregations, such as average, standard deviation, and variance, can be derived from `COUNT` and `SUM`.

Query Approximation and Sampling. The goal of query approximation is generally to speed up execution at the expense of answering the query exactly, while preserving answer accuracy as much as possible. Online sampling is employed for time-sensitive tasks to reduce the overhead of evaluating queries on large databases. *Note that in this case, the sampling differs from one query to another.* In statistical terms, random sampling is essentially the process of selecting a subpopulation *SP* from the total population *P* where a sampling rate *sr* dictates the size of *SP*. This subpopulation contains sufficiently representative individuals and properties, capturing various characteristics of *P* such that the analysis conducted on *SP* can be generalized to *P*. All random sampling techniques can be categorized based on three main features:

- Granularity: sampling elements are individuals or a cluster of individuals.
- Uniformity: elements are sampled with equal/unequal probabilities.
- Replacement: sampling elements can be chosen multiple times or only once.

Nowadays, all modern systems choose to split/store a big table T into a set of smaller, manageable entities $T = \{C_1, C_2, ..., C_N\}$ where each entity has a maximum size S. The entity could be *Table pages*[3], *HDFS file Blocks*[4], etc.

[3]https://www.postgresql.org/docs/current/storage-page-layout.html
[4]https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

In this paper, we call these storage entities *Clusters* and we assume that our tables are already stored as a set of clusters. Given this storage format, sampling on databases can be done at two levels: Row/Cluster level [20].

In tabular databases with range queries, it is particularly challenging to find an online sampling algorithm that offers speed-up while maintaining accuracy.

Data providers. For many real-world use cases, multiple organizations or institutions, called *data providers*, publish access to their databases for joint analysis. Let $\mathbb{S}$ be the set of data providers. In this work, we assume that a large table T is *horizontally* distributed over $\mathbb{S}$ such that all data providers share the same schema (i.e. a set of dimensions) of T but each contains different rows. All data providers use clusters of the same size to store their local tables. *More importantly, for privacy reasons, data providers collaborate on joint analyzes without revealing their data.*

Differential Privacy (DP). A privacy model that provides formal guarantees of *indistinguishability* such that the query results do not yield much information about the presence or absence of any particular individual. Consequently, it hides information about which of the *neighbouring tables* [15] was used to answer the query.

Definition 3.1 (Neighbouring Tables [15]). Two tables T and T' are *neighbouring* if we can obtain one of them by inserting at most a row into the other.

We use $d(T, T')$ to represent the *distance* between two tables T and T' and we say that two tables are *neighbouring* if their distance is 1 or less.

Definition 3.2 ((ϵ, δ)-Differential Privacy [15]). A mechanism M satisfies (ϵ, δ)-*Differential Privacy* (or (ϵ, δ)-*DP*) if, for any two neighboring tables T, T' and for any possible output V of M:

$$\Pr\left[M\left(T\right) \in V\right] \leq exp(\epsilon) \times \Pr\left[M\left(T'\right) \in V\right] + \delta$$

where δ represents the failure probability. We refer to (ϵ, δ) as the *privacy budget*.

In practice, M is a *randomized* algorithm, which has many possible outputs under the same input. It is well known that DP is used to answer specific queries on databases. Let f be a query on a table T whose its answer $f(T)$ returns a number. The *global sensitivity* of f is the amount by which the output of f changes for all neighboring tables.

Definition 3.3 (Global Sensitivity [15]). For any two neighboring tables T and T', the *global sensitivity* of function f is:

$$GS_f = \max_{T,T':d(T,T')\leq 1} \left\| f(T) - f(T') \right\|_1$$

where $\|\cdot\|_1$ is the L_1 norm.

For instance, if f is a `COUNT` range query then GS_f is 1.

The *Laplace Mechanism* is a randomized mechanism for enforcing ϵ-DP (or $(\epsilon, 0)$-DP referred to as pure DP), which adds calibrated noise to f based on its global sensitivity GS_f.

Definition 3.4 (Laplace Mechanism [15]). The *Laplace Mechanism* adds noise to $f(T)$ as:

$$S = f(T) + \text{Lap}\left(\frac{GS_f}{\epsilon}\right)$$

where GS_f is the *global sensitivity* of f, and $Lap(\alpha)$ denotes sampling from the Laplace distribution with center 0 and scale α.

Unlike the Laplace Mechanism, which is used to release noisy numerical values, the *Exponential Mechanism* can be used for biased selection of elements from a set based on a scoring function while preserving $(\epsilon, 0)$-DP [15].

Definition 3.5 (Exponential Mechanism [15]). Given a set of elements SE and a scoring function L, the *Exponential Mechanism* randomly selects $e \in SE$ with the probability of the element e being proportional to:

$$\exp\left(\frac{\epsilon L(e)}{2\Delta_L}\right)$$

where Δ_L is the sensitivity of L.

Local and Smooth Sensitivity. In many applications of DP, the global sensitivity GS_f cannot be bounded. In this case, there is an alternative definition of sensitivity called *local sensitivity*, where the maximum difference between the query's results is based on a fixed database T and any database T' neighbouring to it:

Definition 3.6 (Local Sensitivity [30]). Given a database T and T' as any of its possible neighbouring tables, the *local sensitivity* of function f is:

$$LS_f(T) = \max_{T' : d(T,T') \leq 1} \left\| f(T) - f(T') \right\|_1$$

The local sensitivity $LS_f(T)$ is often much less than the global sensitivity GS_f because it is based on a specific instance of the data T. This also makes it unsafe to use, as it can leak information about T on which it is based. Nassim et *al.* [30]. suggest the use of a smoothing function that finds a safe upper bound for $LS_f(T)$ and can be used to calibrate the randomness (noise) without any risk. These functions usually require that the local sensitivity be computed at any arbitrary distance k from T.

Definition 3.7 (Local Sensitivity at Distance k [30]). Given a table T, the *local sensitivity* of function f is:

$$LS_f(T)^k = \max_{T' : d(T,T') \leq k} \left\| f(T) - f(T') \right\|_1$$

A safe approximate upper bound $S\_LS_f(T)$ of $LS_f(T)$, which is insensitive to small variations of data, can be obtained by the *smooth sensitivity framework* [30].

Definition 3.8 (Smooth Sensitivity Framework [30]).

$$S\_LS_f(T) = max_{k=0,1,\ldots n}\{exp(-\beta k) LS_f(T)^k\}$$

where $\beta = \frac{\epsilon}{2log(2/\delta)}$.

After a number of n iterations, this upper bound can be used to calibrate noise for the Laplace mechanism to ensure (ϵ, δ)-DP.

DP Properties. Combining several DP mechanisms is possible, and the privacy accounting is managed using the sequential and the parallel composition properties of DP. Let $M_1, \ldots, M_n$ be mechanisms satisfying $(\epsilon_1, \delta_1), \ldots, (\epsilon_n, \delta_n)$ -DP.

Theorem 3.9 (Sequential Composition [15]). *Applying sequentially $M_1, \ldots, M_n$ satisfies $\left(\sum_{j=1}^{n} \epsilon_j, \sum_{j=1}^{n} \delta_j\right)$-DP.*

Theorem 3.10 (Parallel Composition [15]). *A mechanism that applies $M_1, \ldots, M_n$ on disjoint parts of the data satisfies: $\left(max_{i\in[n]}(\epsilon_i), max_{i\in[n]}(\delta_i)\right)$-DP*

The *post-processing* property states that it is safe to execute any function on the output of a DP mechanism.

Theorem 3.11 (Post-Processing [15]). *For any (ϵ, δ)-DP mechanism M and any function f, $f(M)$ satisfies (ϵ, δ)-DP.*

In the context of online query answering, each query consumes (ϵ, δ) to secure the results. To manage/ the information released to the analyst, a total budget (ξ, ψ) is given which will be consumed by N queries such that $\xi = N\epsilon$ and $\psi = N\delta$. The analyst can continue sending queries until their total budget is consumed.

Secure Multiparty Computation (SMC). it refers to cryptographic protocols that enable a set of independent parties to collaboratively evaluate a query without revealing their private inputs to each other. It also allows them to avoid trusting a third party with the union of their data for query evaluation. However, this safety assurance comes at the cost of resources and processing time. Using SMC is several times slower than insecure alternatives.

4 PROBLEM STATEMENT

Given a federated system in which n data providers pool their private data for analysis querying. Consider a private table T (as in Figure 2) which is horizontally partitioned among data providers as tables $T_1, \ldots, T_n$. Each data provider wants to keep the individual tuples of their local table confidential and only the schema of T is public. Suppose an end user sends the following range query Q:

```
SELECT COUNT(*) FROM Table WHERE 20 <= Age <= 40
```

where Q is performed on the union of tables stored at the data providers, $\cup_{i=1}^{n} T_i$. However, even though Q may seem very simple at first glance, the big data associated with T_i makes Q very complex and time-consuming.

To solve the problem of slow query response time, we can resort to Approximate Query Processing (AQP) to find a trade-off between accuracy and speed of results via approximation. One very straightforward technique of AQP is to perform *random sampling*, given a sampling rate sr, to obtain a set of tuples from T. For example, an end user can request an answer for Q based only on $sr = 20\%$ of the entire T. Even for a single table T, to obtain a good approximation of Q, the sampled tuples must contain meaningful data in the ranges of Q. Random sampling can be done at the row or cluster level. Although cluster-level sampling is faster than row-level sampling, both have linear performance with respect to sampling rate. The larger the sample, the more accurate and slower the result, and vice versa.

Consider T is stored as a set of clusters. To get an accurate estimate of Q when processing a few parts of the data, we use a statistical estimator [27]. To do this, we need to consider the distribution of rows between all clusters. It should be noted that the assumption of a uniform distribution of rows among all clusters is rarely valid in real databases. Indeed, the rows generally follow a skewed distribution. In contrast, unequal probability cluster sampling is more effective at providing better estimates, where the probability of a cluster being sampled is based on the data distribution for Q.

Assume that each partition T_i is stored using clusters. How to apply the unequal probability cluster sampling in our federated context? Note that each cluster within each data provider should have a specific probability p of being sampled to estimate Q, taking into account all other clusters (even those from other data providers). As a result, capturing the inter/intra data distribution will bias the sampling toward clusters or data providers that hold most of the data related to Q. We refer to this sampling as global sampling.

The other solution is local sampling, where each data provider computes the sampling probabilities for its clusters (without considering other data providers). In this sampling, the sample size is distributed uniformly on data providers, so it does not require a collaboration between data providers. This lack of global data distribution awareness makes this solution less appealing than global sampling.

To apply global data distribution-aware sampling and approximation, data providers must provide appropriate information about their data to quickly and accurately estimate Q. The optimal solution to capture the data distribution in this context is achieved if data providers have access to each other's data and sampling probabilities are computed collectively. This collaboration will lead to an overhead in processing time. The challenge is then to define the summarized and small pieces of information that data providers can share and be sufficient to capture the data distribution while producing negligible overhead. Once this global data distribution is captured, each data provider can locally sample clusters, estimate the query, and send its result. All results from data providers will be added together and the final result will be returned to the end user.

Another dimension of our problem concerns privacy and data protection. In the federated context, the end-to-end privacy property must be guaranteed. This essentially ensures that data is protected (i) during and after query execution, (ii) for intermediate results during collaboration, and (iii) for the final response. DP is a widely accepted privacy model, typically applied to query results to prevent any inference about the presence or absence of individuals. As for the intermediate results produced during collaboration between data providers, they must also be protected, with each data provider seeking to prevent any leakage of information on its table. Even if the exchange is limited to summarized (aggregated) information, there will be no privacy guarantee. Thus, DP can also be used to publish intermediate results between data providers.

An alternative solution to DP is the use of SMC to implement collaboration between data providers. This solution has two major drawbacks: if data providers use the summary information for sampling in SMC, query approximation (which includes running the query on each cluster) must also be done in SMC because the sampling is based on sensitive information and its results may disclose information to other data providers. Second, SMC relies heavily on cryptography, which will significantly reduce the utility of the query in terms of processing time, thereby diluting the purpose of approximations.

In this work, we aim to provide fast and accurate responses to range queries in a federated setup while preserving end-to-end privacy. The challenges we address are: defining a lightweight sampling algorithm considering data distribution for query approximation in a federated environment and carefully applying DP to ensure end-to-end privacy with minimal loss of query accuracy.

5 OUR SOLUTION

In this section, we present in detail our solution. Due to limited space, all proofs of theorems are given in our long version [23].

5.1 Overview

In our work, we combine DP with lightweight SMC to protect intermediate results when collaborating between data providers. This allows us to obtain significantly better performance in terms of speed-up and achieve end-to-end privacy, while maintaining high utility answers for online range queries. To achieve these goals, we propose an efficient and lightweight collaboration method, allowing data providers to decide how many samples to extract from each, guided by the summary information shared during this collaboration. To integrate knowledge of the data distribution into our sampling and approximation steps, we use the *probability proportional to size* (pps) method [27]. Here, the probability p of including (or sampling) a cluster C is determined by the proportion R of rows in C falling within the ranges of the query Q. Computing R is expensive and requires similar overhead as running the query. To minimize the processing time of Q, we will approximate each R of any cluster C using lightweight *metadata* associated with C.

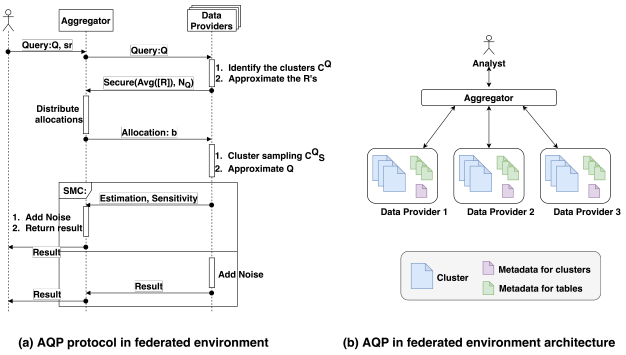

Figure 3: Protocol and Architecture

Our solution has two main phases: offline data preprocessing and online query answering. In the offline data preprocessing phase, each data provider constructs global and individual metadata for its clusters. This metadata makes query approximation easier without imposing a significant overhead in terms of processing time. All data providers agree on the same maximum cluster size S (more details are given in Section 7) before initiating the system. The size S may not reflect the actual size of their clusters, but it would be used to calculate the R of each cluster. The offline phase and metadata creation are detailed in Section 5.2, and Figure 3.(b) shows the general architecture of our system where each data provider is equipped with metadata.

Once preprocessing is complete for all data providers, the system goes online. In the online query response phase, the end user interacts with an aggregator by sending their query Q and desired sampling rate sr and receives a secure response in return. The aggregator manages the rest of the exchanges with the data providers. The query lifecycle (see Figure 3.(a)) as well as the collaboration (exchange of summary data) are described as follows:

(1) First, the aggregator sends the query Q to the data providers. Each data provider performs two tasks: i) identify the set of clusters C^Q covering Q such that $N^Q = |C^Q|$, ii) compute the proportion R of rows for each $C \in C^Q$. The data provider uses previously stored metadata to avoid overhead when performing these two tasks.
(2) Each data provider securely (using DP) sends to the aggregator the summarized data needed for collaboration: the number of clusters N^Q and average of proportions $Avg(\widehat{R})$ where $\widehat{R} = \{R_1, ..., R_{N^Q}\}$.

(3) The aggregator computes and sends the best allocation (sample size s) for each data provider while respecting the total sample size given by sr.
(4) Each data provider tests the condition $N^Q < N^{min}$ in order to compute Q "regularly" without approximation. The N^{min} is a threshold set by each data provider to trigger the approximation only if the query is significantly large (more details about N^{min} are given in Section 5.2).
(5) If the previous condition does not hold, each data provider randomly and securely with DP samples C_S^Q, where $C_S^Q \subset C^Q$.
(6) After sampling, each data provider estimates Q over C_S^Q locally and then securely sends the result to the aggregator with DP guarantees.
(7) Alternatively, data providers may use SMC to share their local estimations and "*sensitivities*". Then, the aggregator obliviously sums the estimations and applies DP using the maximum sensitivity before safely releasing the final result.

In Section 5.2, we will focus on the approximation via cluster sampling and the metadata created offline. Afterward, Section 5.3 will be dedicated to the second phase of our solution. In Section 5.3.1, we will describe the allocation step and how it preserves the same semantics as the naive (sharing all data) method of collaboration by keeping the sampling data distribution aware without an overhead. In Section 5.3.2, we will present the privacy-preserving sampling used by each data provider locally to create C_S^Q. In Section 5.3.3, we detail how to obtain a calibrated DP noise for the final result obtained by using a statistical estimator. Finally in Section 5.4, we explain how the privacy budget for each query is managed and consumed.

5.2 Query Approximation and sampling

As previously mentioned in Section 5.1, our unequal probability sampling is based on the proportion R of rows in cluster C that corresponds to Q. Computing the exact R for each cluster is as costly as evaluating the query itself, rendering the approximation useless. Inspired by [42], we will only approximate R to avoid an overhead in response time. Given a query $Q = \{\forall d \in D^Q \mid r_d = [l_b^d, u_b^d]\}$ where each dimension is defined by a range. We assume that the dimensions are not correlated (independent). We will compute the sub-proportions R^d on each dimension as follows:

$$R^d = R^{d\geq}(l_b^d) - R^{d\geq}(u_b^d)$$

$$\text{where } R^{d\geq}(x) = \frac{|rows^d \geq x|}{S} \text{ and } S \text{ is the cluster size.}$$

The proportion R^d is computed based on the proportions $R^{d\geq}(l_b^d)$ and $R^{d\geq}(u_b^d)$ of records whose dimension d values are $\geq l_b^d$ and $\geq u_b^d$, respectively. Based on the assumption of independence between dimensions, R can be obtained as follows:

$$R = \prod_{d\in D^Q} R^d \text{ and } p_j = \frac{R_j}{\sum_{i=0}^{N^Q} R_i} \quad (1)$$

where N^Q is the number of clusters covering Q. The approximated R can then be used to obtain the sampling probabilities p_j for the jth cluster as shown in Equation 1. Even this approximation requires a lot of calculations, which may cause similar overhead as the exact R. To bypass this limitation, we associate each cluster with a set of metadata that accelerates these computations for any given query (see Algorithm 1).

Algorithm 1 Cluster metadata

Require: $T = \{C_1, C_2, ..., C_N\}$: Set of clusters
1: **Clusters_metas** ← *create_global_meta*()
2: **for** each $C \in T$ **do**
3: *cluster_meta* ← []
4: **datas_meta** ← *create_datas_meta_*(C)
5: **for** each $d \in D$ **do**
6: **for** each $v \in |d|_C$ **do**
7: $R^{d\geq}(v)$ ← *portions_greater_*(C, d, v)
8: **datas_metas**.*add*($d, v, R^{d\geq}(v)$)
9: **end for**
10: v_{min}^d, v_{max}^d ← *min_max*($|d|_C$)
11: *cluster_meta.add*(v_{min}^d, v_{max}^d)
12: **end for**
13: **Clusters_metas**.*add*(*cluster_meta*)
14: *save*(**datas_metas**)
15: **end for**
16: *save*(**Clusters_metas**)

In Algorithm 1, for each cluster C and for each distinct value v of dimension $d \in D$ in $C \in T$ (Lines 5-6), $R^{d\geq}(v)$ is stored in the dedicated meta file for the cluster where the entry is in the form $\{d, v, R^{d\geq}(v)\}$ (Line 8). These metadata will be used by each data provider to quickly access precomputed proportions that correspond to the range of a given Q, thus significantly reducing the overhead in the online phase. To further improve the performances, Algorithm 1 stores additional global metadata about the clusters **Clusters_metas**, enabling the system to easily identify the clusters C^Q that correspond to Q before even computing the proportions. In a dedicated global file ***Clusters metas***, for each dimension $d \in D$ in cluster C, Algorithm 1 (Lines 11-13) stores $v_{\min}^d$ ($v_{\max}^d$), the minimum (maximum) value of d in C. Based on these metadata in ***Clusters metas***, the system is able to focus only on a small subset of the database C^Q that actually contains rows matching Q instead of T, thus reducing the processing time of Q. The set C^Q is defined as follows:

$$C^Q = \{\forall C \in T \mid \forall d \in D^Q, [v_{min}^d, v_{max}^d] \cap r_d \neq \emptyset\} \quad (2)$$
where r_d is the interval of Q in dimension d.

Since we are able to identify the clusters C^Q concerned by Q, it only makes sense to approximate Q only when N^Q is bigger than a certain threshold N^{min}. This threshold can be set independently by each data provider based on the size of the clusters, the processing time required for a single cluster, and the hardware and software infrastructures. The cost of saving these metadata is very negligible compared to the actual table and clusters. We used the same data structure like [42] which is very efficient. In Section 6, we show the space needed for each database.

Once the sampling is applied according to the probability computed using Equation 1, the *Hansen-Hurwitz* estimator [27] is used to obtain the final estimation of Q. The estimation is done as follows:

$$E(Q, C_S^Q) = \frac{1}{N_S} \sum_{i=1}^{N_S} \frac{Q(C_i)}{p_i} \quad (3)$$

where p_i is the sampling probability of the ith cluster and $Q(C_i)$ is the query execution result on the ith cluster

5.3 Federated protocol

In this section, we will review all the steps of online query approximation and how we were able to carefully integrate DP into each step.

5.3.1 Allocation phase. In this step, the data providers $\mathbb{S}$ need to jointly decide the number of clusters to be sampled from each one of them based on the distribution (R's) of data related to Q. So upon receiving the query, each data provider identifies C^Q and computes the R for each $C \in C^Q$ using the metadata stored locally. Then each one sends to the aggregator its N^Q and $Avg(\widehat{R})$, where $\widehat{R}$ is the set of R's of the clusters in C^Q. N^Q indicates the number of clusters within that data provider that overlap with Q, while $Avg(\widehat{R})$ shows the average proportion of rows within those clusters that corresponds to Q. Based on this information, we obtain an aggregated (summary) view of the data distribution of records corresponding to Q in each data provider. Using these insights, the aggregator finds the best sample size s_i for the ith data provider using an optimization problem given in Equation 4 that aims to assign a bigger allocation to the data provider with the most data related to Q.

$$\begin{array}{ll} \textbf{maximize} & \sum_{i=0}^{|\mathbb{S}|} \text{Avg}(\widehat{R})_i \times s_i \\ \\ \textbf{where} & \sum_{i=0}^{|\mathbb{S}|} s_i = sr \times \sum_{i=0}^{|\mathbb{S}|} N_i^Q \\ \textbf{and} & sr \in]0,1[\text{ is the sampling rate} \\ \textbf{and} & s_i \in]1, N_i^Q[\end{array} \tag{4}$$

In Equation 4, the data provider that holds the most data related to Q (has the bigger $\text{Avg}(\widehat{R})_i$) gets more allocation, thus sampling more clusters to approximate Q locally. This reflects the same behaviour as the original collaboration method (described in Section 4): sampling probabilities are computed globally and the clusters of the data provider with the bigger $R's$ are more likely to be sampled than others (higher probabilities, Equation 3). So with our collaboration method, we are able to reproduce similar results and behaviour. It is important to highlight that comparing the $\text{Avg}(\widehat{R})$ from each data provider is only possible because we imposed they use the same S in order to compute the proportions during the metadata creation phase.

To solve the problem in Equation 4, each data provider shares the N^Q and $\text{Avg}(\widehat{R})$. Both are sensitive pieces of information that may reveal insights about the individuals within the database. Even if the optimisation in Equation 4 is done over encrypted data, the released allocation s_i might give a data provider insights about the other data providers. To secure the release of this information, each data provider uses *Laplace mechanism* to ensure formal guarantees of privacy. Given a privacy budget ϵ^O, each data provider perturbs these two values as follows:

$$\begin{aligned} \widetilde{\text{Avg}}(\widehat{R}) &= \text{Avg}(\widehat{R}) + \text{Lap}\left(\frac{2\Delta_{\text{Avg}(\widehat{R})}}{\epsilon^O}\right) \\ \widetilde{N^Q} &= N^Q + \text{Lap}\left(\frac{2}{\epsilon^O}\right) \end{aligned} \tag{5}$$

where the sensitivity of N^Q to the absence/presence of an individual is 1, and the sensitivity of $\text{Avg}(\widehat{R})$, is $\Delta_{\text{Avg}(\widehat{R}}$.

Theorem 5.1 (Sensitivity of estimator $\Delta_{\text{Avg}(\widehat{R})}$). *For any two neighbouring databases T, T' the sensitivity of $Avg(\widehat{R})$ is defined as:*

$$\Delta_{Avg(\widehat{R})} = max(\frac{\Delta_R}{N^{min}}, \frac{1}{N^{min}+1}) \text{ where } \Delta_R = 1 - (1 - \frac{1}{S})^{|D|}$$

With this perturbation, the collaboration between data providers for deciding the allocation does not reveal any sensitive information. So the optimization problem is formulated as follows:

$$\begin{array}{ll} \textbf{maximize} & \sum_{i=0}^{|\mathbb{S}|} \widetilde{\text{Avg}}(\widehat{R}_i) \times s_i \\ \textbf{where} & \sum_{i=0}^{|\mathbb{S}|} s_i = sr \times \sum_{i=0}^{|\mathbb{S}|} \widetilde{N_i^Q} \\ \textbf{and} & sr \in]0,1[\text{ is the sampling rate} \\ \textbf{and} & s_i \in]1, \widetilde{N_i^Q}[\end{array} \tag{6}$$

The test of $N^Q < N^{min}$ comes after the allocation (collaboration) phase in order to encourage all data providers to participate. Otherwise, if a data provider does not participate because locally approximating Q is not possible, this may reveal information about the size of its data to other data providers.

5.3.2 Sampling phase. After the allocation phase, each data provider receives an allocation s: the number of clusters to process for the Q's approximation. Using the $\widehat{R}$ computed locally, the data provider computes the sampling probabilities for C^Q and then performs unequal probability sampling to randomly select s clusters. Since the sampling probabilities are computed based on the rows (individuals) in the database, the result of the sampling (choices) may leak information about the presence/absence of any individual. To guarantee DP, our system uses the Exponential Mechanism (EM) to select the s clusters $C_S^Q \subset C^Q$ (see Algorithm 2) while consuming ϵ^S privacy budget.

Algorithm 2 *EM_sampling*

Require: C^Q: set of clusters, $\widehat{R}$: set of corresponding $R's$ to C^Q, s: sample size, ϵ^S: total budget

1: $P \leftarrow$ get_sampling_probabilities($\widehat{R}$) ▷ Equation 1
2: $P^{EM} \leftarrow []$
3: $\epsilon^s \leftarrow \epsilon^S / s$
4: **for** $i \in [1, N^Q]$ **do**
5: $\quad P^{EM}[i] \leftarrow \exp\left(\frac{\epsilon^s P[i]}{2\Delta p}\right)$
6: **end for**
7: $C_S^Q \leftarrow$ random_choice(C^Q, P^{EM}, s)
8: **Return** C_S^Q, P

The score of the ith cluster $C_i \in C^Q$ is its own sampling probability p_i (Algorithm 2, Line 1), which means the scoring function L of EM is defined by the computation in Equation 1. So to calibrate the noise (randomness) of EM, we must find the sensitivity of this function L to the absence/presence of any individual in the database.

Consider two neighbouring databases T and T', where T' is obtained by adding any random record (which represents an individual) to T at any possible cluster. Given a range query Q, in order to measure Δp_i (sensitivity of p_i, which is the same as L) we assume the worst case scenario for T and T': all clusters of $C^Q (C^Q \subset T)$ each have a record that corresponds to Q. In this case, their probabilities are the same: $p = \frac{1}{N^Q}$. In T', one record is added to another cluster C' outside of C^Q that matches Q. Thus $C'^Q = C^Q \cup \{C'\}$, $N'^Q = N^Q + 1$, and for Q all the clusters have the same sampling probability: $p' = \frac{1}{N^Q+1}$. So the Δp can be computed as follows:

$$\Delta p \leq \left|\frac{1}{N^Q} - \frac{1}{N^Q+1}\right| = \frac{1}{N^Q(N^Q+1)} \tag{7}$$

We notice that Δp is dependent on the query Q. To find the global maximum value for Δp, we replace N^Q by its minimum possible value N^{min}.

Theorem 5.2 (Sensitivity of sampling probability). *For any two neighbouring databases T and T', the sensitivity of the sampling probability p_C of any cluster C is bounded by :*

$$\Delta p = \max_{T,T'} \left\| p_C(T) - p_C(T') \right\|_1 = \frac{1}{N^{min} \times (N^{min} + 1)}$$

In Algorithm 2, this sensitivity Δp is used for sampling using *EM* (Line 5). To manage the total budget ϵ^S allocated for *EM* in order to safely make s selections (Line 7), we set $\epsilon^s = \frac{\epsilon^S}{s}$ the budget of each random selection (Line 3).

5.3.3 Approximation phase. To obtain the final result from C_S^Q, each data provider uses the estimator E defined in Equation 3. In order to release the final results securely and have DP guarantees, a well-calibrated noise will be added to the final answer using *Laplace Mechanism*. To apply *Laplace Mechanism*, we need to find the sensitivity Δ_E of the estimator. Let us define $\mathbb{E}(C, Q, p) = \frac{Q(C)}{p}$. We can rewrite E as follows :

$$E(Q, C_S^Q) = \frac{1}{s} \sum_{i=1}^{s} \mathbb{E}(Q, C_i, p_i) \quad \text{where } s \text{ is the size of } C_S^Q \tag{8}$$

Which implies that :

$$\Delta_E = \frac{1}{s} \sum_{i=1}^{s} \Delta_{\mathbb{E}} \tag{9}$$

We will focus on finding $\Delta_{\mathbb{E}}$ to deduce Δ_E. Given that $\mathbb{E}(C, Q, p)$ is a fraction of two real values, it gives a hint that its sensitivity might be unbounded similarly to *Average* operator [28]. Upon further analysis (see [23]), we find that $\Delta_{\mathbb{E}}$ is unbounded, which implies Δ_E is also unbounded.

Theorem 5.3 (Sensitivity of estimator $\mathbb{E}$). *For any two neighbouring databases T, T' the sensitivity of the estimator $\mathbb{E}$ for any cluster C and query Q is unbounded:*

$$\Delta_{\mathbb{E}} = \max_{T,T'} \left\| \mathbb{E}(Q, C) - \mathbb{E}(Q, C') \right\|_1 \geq \frac{N \times S^D}{2} - 1$$

5.3.

Given that a global sensitivity does not exist, we resort to the *Local Sensitivity (LS)* which is measured based on the database instance T. For any database T' neighbouring to T obtained by adding 1 row (one individual) that matches the query Q, we can distinguish four scenarios for a cluster $C \in C^Q$ (we focus on one cluster C because we are looking for $\Delta_{\mathbb{E}}$) that might affect $\mathbb{E}$:

- Scenario 1: Cluster C did not receive the new row, but another cluster did.
- Scenario 2: Cluster C did receive the new row.
- Scenario 3: Cluster C did not receive the new row but another cluster has been added to C^Q, such that $N'^Q = N^Q + 1$.
- Scenario 4: Cluster did receive the new individual, but only add +1 to the *Measure* attribute of existing aggregate row.

Our aim is to find the upper bound of $LS_{\mathbb{E}}$, thus we must consider the distance that provides the largest sensitivity. An analysis of each of these scenarios (see [23]) showed that under a certain condition, either scenario 1 or scenario 4 will yield the biggest distance. For a given cluster C, we can choose the *dominant scenario* (which will yield the biggest $LS_{\mathbb{E}}$) between scenarios 1 and 4 without needing to compute any of them.

Theorem 5.4 (Dominant distance LS). *The neighbouring scenario 1 will give bigger distance than scenario 4 iff:*

$$Q(C) > \frac{\sum^{R \in \widehat{R}} R}{\Delta_R}$$

Since the $LS_{\mathbb{E}}$ is computed based on T, it cannot be used directly to inject noise because the scale of the noise may reveal sensitive information about T [28]. To avoid such information leakage, we will use the *smooth sensitivity framework* [30] for finding a safer upper bound $S\_LS_{\mathbb{E}}$ for the *local sensitivity* $LS_{\mathbb{E}}$. So we redefine our $LS_{\mathbb{E}}$ in terms of a distance k between T and T':

- Scenario 1: $LS_{\mathbb{E}}^k = k \times \frac{Q(C) \times \Delta_R}{R}$
- Scenario 4: $LS_{\mathbb{E}}^k = k \times \frac{1}{p}$

The safe smooth upper $S\_LS_{\mathbb{E}}$ is defined as follows:

$$S\_LS_{\mathbb{E}} = max_{k=0,1,\ldots n} \{ e^{-\beta k} \times LS_{\mathbb{E}}^k \} \tag{10}$$

where $\beta = \frac{\epsilon^E}{2 \ln(2/\delta)}$ and (ϵ^E, δ) is the privacy budget allocated for releasing the final result.

Algorithm 3 *Estimate_Q*

Require: $Q : query, C_S^Q : clusters, (\epsilon^E, \delta) : budget, SMC : bool$

1: $result \leftarrow$ approximate_Q(Q, C_S^Q) ▷ Equation 3
2: $S\_LS \leftarrow []$
3: **for** $i \in [1, N_S^Q]$ **do**
4: $\quad S\_LS[i] \leftarrow smooth\_LS(Q, C_S^Q[i], \epsilon^E, \delta)$ ▷ Equation 10
5: **end for**
6: $LS\_smooth \leftarrow average(S\_LS)$ ▷ Equation 9
7: **if** SMC **then**
8: $\quad send\_secure(result, LS\_smooth)$
9: **else**
10: $\quad dp\_result \leftarrow result + Lap(\frac{2 \times LS\_smooth}{\epsilon^E})$
11: $\quad send(dp\_result)$
12: **end if**

Based on the definitions we gave for $LS_{\mathbb{E}}^k$, the computational overhead to compute the *smooth sensitivity* for each cluster $C \in C_S^Q$ is very negligible because: i) All the R's and p's are computed before this step, and will be reused for each iteration over k; ii) the maximum value of k (steps) is also bounded by $k = \frac{1}{1-e^\beta} + 1$, which guarantees that the process will terminate; iii) Theorem 5.4 allows to determine which scenario is *dominant* for any given cluster, thus only computing one $S\_LS_{\mathbb{E}}$.

Algorithm 3 describes the process of estimating Q over the subset of cluster C_S^Q. It starts by estimating Q according to Equation 3 (Line 1). Then it proceeds to compute the *smooth sensitivity* (Lines 2-6), where the function *smooth_LS* is responsible for computing the smooth sensitivity $S\_LS_{\mathbb{E}}$ for each cluster $C \in C_S^Q$ as described in Equation 10. Depending on the chosen setup by the data providers, either they compute and send a DP result to the aggregator (Lines 10–11) that returns ultimately the sum to the user. The second option is that data providers share their estimations and computed sensitivities (Line 8) with the aggregator that obliviously computes the sum of estimations and the maximum sensitivity to perturb the final result with *Laplace Mechanism*.

5.4 Privacy accounting

In the online query answering settings under DP, the end user is limited by a total privacy budget of (ξ, ψ). For each query Q, a budget (ϵ, δ) is consumed in order to publish the answer and the end user can interact with system as long as the total budget (ξ, ψ) is not consumed. In this section, we will track the privacy budget ϵ consumption for each query.

In our proposed protocol the data providers do not share their data, and Q is processed (data access and publishing) in parallel by each data provider. We can just track the consumption on one data provider, and based on the *parallel composition property* of DP we can deduce the budget consumption for Q on the full system. A data provider starts by publishing the N^Q and $\text{Avg}(\widetilde{R})$ using *Laplace mechanism* for the allocation phase, while consuming a total budget of ϵ^O. Based on the *post-processing property* of DP, obtaining the sample size s is DP. Afterwards, each data provider uses *Exponential Mechanism* to sample a subset $C_S^Q \subset C^Q$ while consuming a budget of ϵ^S. To publish an estimation of Q, each data provider uses *Laplace mechanism* once more, and consumes a budget of ϵ^E. The final step does not in fact guarantee pure DP, since the smooth sensitivity has a δ *failure probability*. Based on the *sequential composition property* of DP, the total budget is: $(\epsilon = \epsilon^O + \epsilon^S + \epsilon^E, \delta)$. Given the *parallel composition property*, the budget consumption for Q is (ϵ, δ).

In case the data providers use SMC to inject a single noise, and based on *parallel composition property*, we deduce that data providers consumed $\epsilon^O + \epsilon^S$ for the local computation. Then they collectively consumed (once) ϵ^E for publishing the result. By the *sequential composition property* of DP, the budget consumption for Q is $(\epsilon = \epsilon^O + \epsilon^S + \epsilon^E, \delta)$.

Based on these results, a set of hyperparameters can be set in our system (for example, by database admin) that regulates the ϵ budget distribution at each step of the query processing. Let hp_1, hp_2 and hp_3 be this set of hyperparameters (where each $hp_i \in]0, 1[$ and $hp_1 + hp_2 + hp_3 = 1$) such that: $\epsilon^O = \epsilon hp_1$, $\epsilon^S = \epsilon hp_2$ and $\epsilon^E = \epsilon hp_3$.

6 EVALUATION

6.1 Setup

Datasets. We used two big datasets: (i) *Adult* [9] contains demographic and income information for individuals with 15 dimensions and 48×10^3 records, synthetically scaled up 4×10^6 records. (ii) *Amazon Review* [29] is about reviews from Amazon clients across different product categories, with only three "range queriable" dimensions and 231×10^6 records (~ 120 Gb). We synthetically added three randomly populated dimensions and random records to reach $4 \times 231 \times 10^6$ records.

A count tensor with column *Measure* is created from each dataset, aggregating six dimensions of *Adult* and one dimension of *Amazon Review*.

Queries and Workloads. We generated random ranges for the queries and ran only those that lead to the approximation $(N^{min} < N^Q)$ on all data providers. A workload (m, n) is a set of m *distinct* queries with ranges over n dimensions.

Metrics. An online query is useful if it has a low error rate and low processing time. To measure the query error, we used *Relative error* $= \frac{|\text{answer} - \text{estimation}|}{\text{answer}}$. For performance in terms of *response time*, we used: *Speed-UP*= $\frac{\text{time of normal computation}}{\text{time of estimate computation}}$.

Configuration. In our experiments, we assumed that there are one aggregator and four data providers and that each data provider has its own database. Datasets *Adult* and *Amazon Review* are horizontally partitioned equally across data providers.

Source code. Based on PostgreSQL[5], our solution[6] coded in Python uses the libraries: (i) OrTools[7] as solver; (ii) Pyro5[8] as communication medium; and, (iii) MPyC[9] as SMC environment. Our implementation is a proof-of-concept in which the clusters of the original table are other smaller tables.

Hyperparameters. In our experiments, the total privacy budget (ϵ, δ) for each query is set with $\delta = 10^{-3}$ and $\epsilon = 1$ (unless other values are indicated for ϵ). The budget ϵ is shared between each step of our solution as follows: $\epsilon^O = 0.1 \times \epsilon$, $\epsilon^S = 0.1 \times \epsilon$ and $\epsilon^E = 0.8 \times \epsilon$. To get clusters of the same size, we set the cluster size S to 1% and 0.5% of the total size T^a of each data provider for *Adult* and *Amazon Review*, respectively.

Metadata space allocation. The metadata for *Amazon Review* dataset was about 11 MB (56 KB/cluster). As for *Adult* dataset, it occupied 6.4 MB (64 KB/cluster).

Hardware[10]**.** For each of the data providers and the aggregator, we allocated a dedicated server with the following configuration: 2 X Intel Xeon E5−2630 v3 8 cores/CPU x86_64, RAM 128 GB and 1.2 TB HDD, and a network with 1 Gbps + 4 x 10 Gbps (SR-IOV).

6.2 Dimension-based analysis

In these experiments, we evaluated the impact of the number of dimensions in queries on accuracy. To this end, we generated random workloads (m, n) with $m = 100$ distinct queries (`SUM` and `COUNT`) and dimension $n \in [2, 7]$ for *Adult* and $n \in [2, 5]$ for *Amazon Review*. For the sampling rate, we set it to 5% and 20% for *Amazon Review* and *Adult* datasets, respectively.

The results presented in Figure 4 show that our solution achieves very high accuracy for `COUNT` and `SUM` queries. The relative error is less than 2.5% (resp. 11%) on average for `COUNT` queries on *Amazon Review* (resp. *Adult*). As for `SUM` queries, the error is less than 5% (resp. 17%) on *Amazon Review* (resp. *Adult*). This performance difference is due to the size difference between the databases. In big tables, query results are larger (contain more data), therefore less affected by *Laplace Mechanism* noise. Interestingly, the results also indicate that queries become more accurate as the number of dimensions decreases. Specifically, with workloads having only 2 dimensions on both datasets, we reached an error close to 0%. This observed behavior corresponds to our expectations. Because in Equation 1, we approximate R of each cluster and the accuracy of this approximation improves as the number of dimensions decreases, bringing the approximation closer to the exact R. Thus, we have more accurate sampling probabilities which affect the estimation of the final result. For the speedup, the results in Figure 7 show that the higher the number of dimensions, the less speedup is gained. From the results in Figure 7, the speedup drops from approximately $8x$ to $6x$ as the number of dimensions increases from 2 to 5 on *Amazon Review* dataset. This drop is attributed to the sampling probabilities approximation phase, where our algorithm looks up the preprocessed metadata. The higher the number of dimensions, the more metadata it needs to look up. However, this effect becomes negligible on larger databases. Because even in these results, the speedup remains very significant.

[5] https://developers.google.com/optimization
[6] https://github.com/AlaEddineLaouir/Federated-Range-Queries.git
[7] https://developers.google.com/optimization
[8] https://pyro5.readthedocs.io/en/latest/index.html
[9] https://mpyc.readthedocs.io/en/latest/mpyc.html
[10] Grid5000: Grisou cluster https://www.grid5000.fr/w/Nancy:Hardware

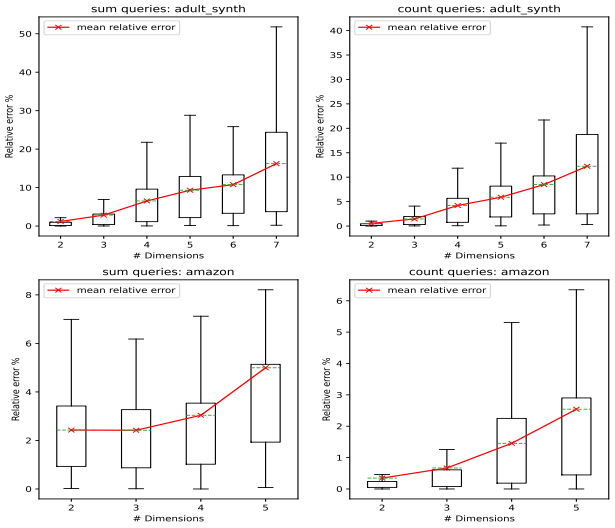

Figure 4: Dimension-based analysis

6.3 Sampling rate-based analysis

In this analysis, we examined the effect of sampling rate on query quality. For each database, we generated two random workloads for COUNT and SUM queries of $m = 100$ and $n = 4$. We varied the sampling rate between 5% and 20% for each experiment and measured the quality obtained in terms of accuracy and speed-up. From the results in Figure 5, we observe that a higher sampling rate provides slightly better accuracy: reaching a relative error of less than 1% with a 20% sampling rate for COUNT queries on *Amazon Review* dataset.

Regarding the speed-up, we note that our solution reaches up to a $7x$ compared to a normal execution (without approximation) on *Amazon Review* (with 4 dimensional queries). Additionally, the speed-up gains in *Amazon Review* are $4x$ more significant than those in *Adult*. This result indicates that our solution provides more speed for larger datasets. Also based on the results in Figure 5, the tradeoff between speed-up and accuracy is noticeable. We observe that the larger the sampling, the less the speed-up is gained. On the other hand, accuracy improves with higher sampling rates. We can say that, based on the results shown in this experiment, accuracy gains with higher sampling are very costly in terms of speed-up. But it is up to the users (data analysts) to define the sampling rate according to their needs.

6.4 Privacy budget-based analysis

In these experiments, we analyzed the effect of the privacy budget ϵ on query quality. We generated two random workloads of $m = 100$ and $n = 4$ for COUNT and SUM queries and set the sampling rate to 5% and 10% for *Amazon Review* and *Adult*, respectively. We varied ϵ between 0.1 and 1.3 and captured the performance on each workload. From the results in Figure 6, we can immediately observe the typical trend of any DP mechanism (larger ϵ leads to better accuracy).

Interestingly, SUM queries are able to provide better utility (lower relative error) than COUNT queries. This happens because SUM queries yield more substantial results (larger query responses) than COUNT queries, making them less affected by noise added to the response. A similar observation applies when comparing results between the two databases, with workloads on *Amazon Review* preserving more accuracy than those on *Adult*. This is

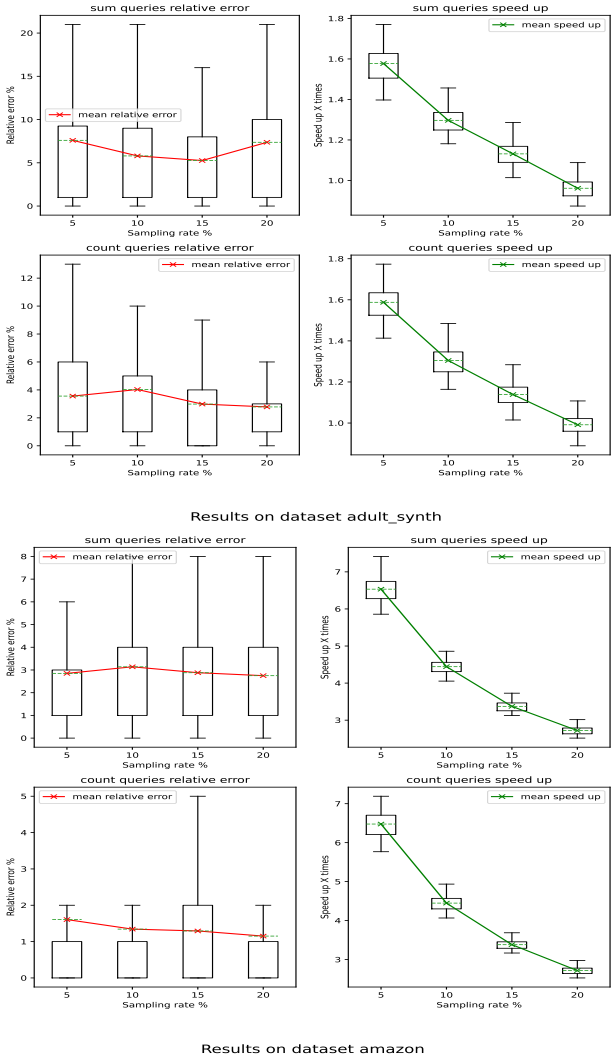

Figure 5: Sampling rate-based analysis

attributed to the fact that the *Amazon Review* dataset is much larger than *Adult*, causing queries to be less affected by the added noise. Based on this observation, we can predict that as the database size increases, the accuracy of our solution will improve by using smaller values for ϵ. Regarding speed-up, the results in Figure 7 show that ϵ levels have no effect.

6.5 SMC vs DP in terms of sharing results

To examine the performance of our SMC-based solution to share final results, we conducted experiments using an *Adult* dataset split across four data providers. We generated five random two-dimensional COUNT queries. Each query was repeated five times (with and without SMC) and we measured the speed-up and the the range of noise added using the *Laplace mechanism* at each iteration.

The results in Figure 8 show, for each query, the range of noise sampled using the *Laplace mechanism* for both solutions at each iteration and speed-up. We notice in Figure 8 that using SMC to share only the sensitivity and the local result does not produce significant overhead, which corresponds to the simulation results in Figure 1. Concerning the injected noise, which affects the precision of the query result, the use of SMC allows a more

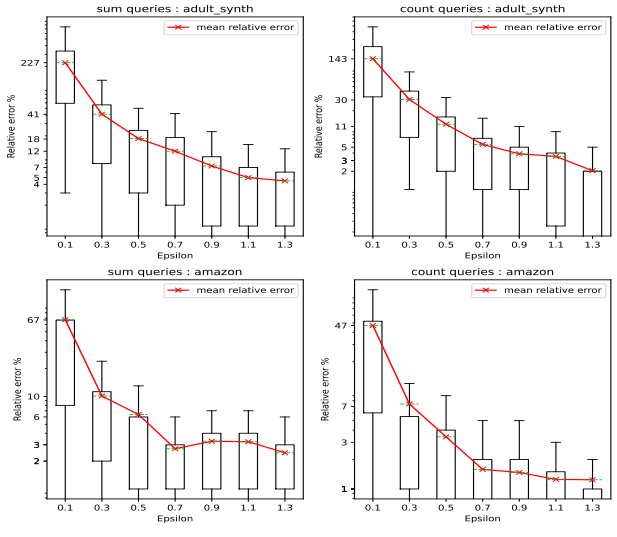

Figure 6: Epsilon-based analysis

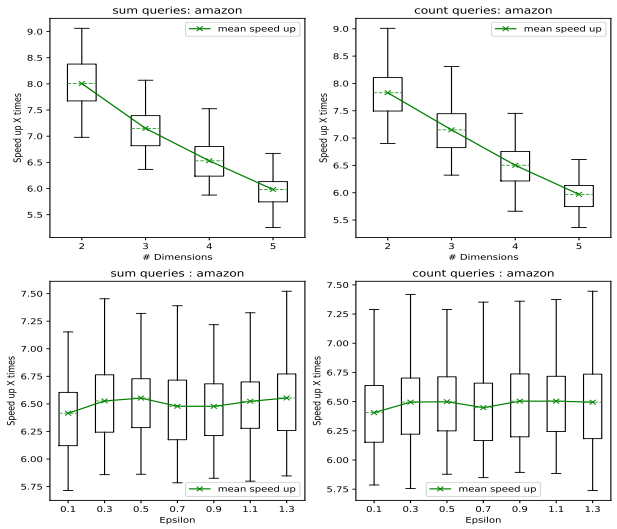

Figure 7: Impact of dimension and ϵ on speed-up

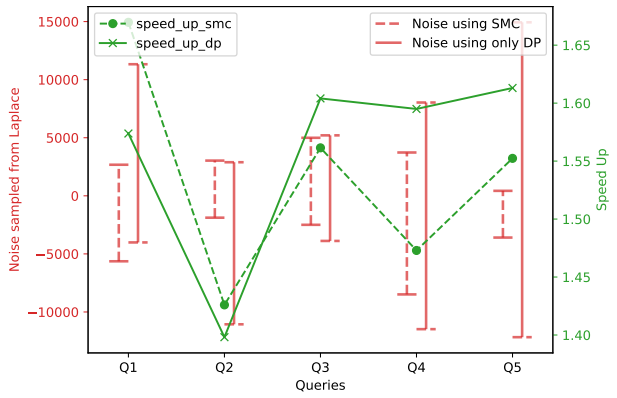

Figure 8: SMC effect on speed-up and accuracy

restricted range of perturbation. Meanwhile, if each data provider perturbs its local data without SMC, there could be two cases: (i) the noises from the data providers cancel each other out, or (ii) the noise accumulates. In the first case, the sum of noises is close to zero because some are positive and others negative, which will help improve accuracy. In the second case, which represents the worst case where most of the noise is positive or negative, the accuracy of the results will be greatly affected. Based on the experiment results, a user/data provider can choose the appropriate query execution process (with or without SMC) based on their needs, preferring accuracy over speed-up or vice versa.

6.6 Resilience to Learning-Based Attacks

DP prevents membership attacks revealing the presence/absence of an individual in the database. In [13], the author introduced a simple attack that allows the disclosure of an individual's sensitive *SA* attribute based on anonymized data. This attack relies on training a *Naive Bayes Classifier* (NBC) using the results of `COUNT` queries from a noisy database, and this classifier will be used to predict the value of *SA* based on a given set of *QI* (quasi-identifiers) attribute values of an individual. In our data model, *SA* corresponds to one of the dimensions $d_{SA} \in D$, and *QI* is the subset $D_{QI} \subseteq D \setminus \{d_{SA}\}$. Given $V_{QI} = \{v_1, ..., v_{|D_{QI}|}\}$ for D_{QI}, a NBC attaches a probability to each possible value y of d_{SA} ($y \in |d_{SA}|$). The predicted value $\hat{y}$ is the one with the highest probability according to Bayes Theorem [13]:

$$\hat{y} = \underset{y \in |d_{SA}|}{\arg\max}\, P(y) \prod_{i=1}^{|D_{QI}|} P(v_i|y)/P(v_i)$$

To make these predictions, the classifier goes through a training phase during which it learns the conditional probabilities using the queries `COUNT(*)` (or `SUM(Measure)`) issued by the attacker to the database. The learned probabilities are saved and later used to make predictions. The number of queries *nQueries* needed is:

$$nQueries = 1 + ||d_{SA}|| + ||d_{SA}|| \times \sum_{dQI \in D_{QI}} ||dQI||$$

which is used to compute the *size* of the database, $P(y)$ and $P(v|y)/P(v)$ for all values and dimensions. For instance, consider a table `T` with 10000 rows and $|d_{SA}| = [20, \ldots, 60]$ is the dimension for *Age* attribute. To compute $P(Age = 25)$, we use the following `COUNT` query:

```
SELECT COUNT(*) FROM T WHERE 25 <= Age <= 25 )/ 10000.
```

This huge number of queries can be easily issued to a published database using a DP algorithm with a fixed privacy budget (e.g. PrivBayes[41]), and from which the attacker can infer some knowledge [13, 18].

However, the database is *not published* in our system. As we showed in Section 5.4 the attacker has a limited budget ($\xi > 0, \psi > 0$), from which each issued query consumes a privacy budget ($\epsilon > 0, \delta > 0$) based on a sequential composition 3.9. Since *nQueries* can be very large, ϵ must be very small $\epsilon = \xi/nQueries$ and $\delta = \psi/nQueries$, thus losing the utility of query answers. An alternative to sequential composition is *Advanced composition* [22, 27], which allows the queries to have a greater budget ϵ without exceeding ξ. With the advanced composition, the budget of each query is: $\epsilon = \xi/\left(2 \times \sqrt{2 \times nQueries \times log(\frac{1}{\delta})}\right)$ and $\delta = \psi/nQuesries$. We notice that $\xi/\left(2 \times \sqrt{2 \times nQueries \times log(\frac{1}{\delta})}\right) > \xi/nQueries$, which means queries have better utility.

To evaluate the resilience of our system against this learning-based attack, we tested both sequential compositions and the two allowed queries COUNT and SUM. We also considered *parallel composition* which allows multiple attackers to create a coalition, where each of them executes only one query (to maximize utility) and combines it with those of other attackers to train the classifier. The ingredients of our experiments are as follows:

Setup: We used *Adult* dataset with four data providers. We selected 3 dimensions of our table to be D_{QI} and 1 dimension to be d_{SA} where $||d_{SA}||$ =100 (i.e. the number of classes for NBC). We also set $\psi = 10^{-6}$ and we varied ξ between 1 and 100 since there is no standard value [24, 27].

Evaluation: To assess the quality of the learning attack, we measured the accuracy of the NBC in predicting the value of SA for each row in the original table $accuracy = \frac{\text{number of correct predictions}}{\text{total number of predictions}}$.

	$\xi = 1$	$\xi = 20$	$\xi = 50$	$\xi = 100$
Sequential / COUNT	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$
Sequential / SUM	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$
Advanced / COUNT	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$
Advanced / SUM	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$
Coalition / COUNT	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$
Coalition / SUM	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$

Table 1: Inference accuracy based on ξ

The results in Table 1 show that in all scenarios the accuracy is $< 1\%$. Since the SA we used had 100 possible values, this means that the trained classifier is given similar accuracy as randomly assigning a value for SA in each row. Three reasons can be put forward to explain the failure of the learning-based attack: i) our system is interactive (the database is not released) and the budget is limited, thus it is difficult to have good accuracy for large numbers of queries by a single attacker; ii) query answers in our system are approximated with random sampling, which will introduce some error; iii) the smooth sensitivity has a considerable scale, and in the case of queries that collects small values, the accuracy can be lost even for large values of ϵ.

Similar results were obtained when fixing the $\xi = 100$ and changing the number of dimensions in D_{QI} from 1, 3, 5 to 8. This shows the resilience of our system in different settings.

7 DISCUSSION

To approximate the sampling probabilities (see Section 5.2), we assumed that the dimensions are independent and that there is no correlation between them. However, this assumption is not valid in some cases. For example, if an individual's *Age* is less than 25, this implies with a high probability that he/she is still studying ($profession = student$). Likewise, if $Age > 65$, the attribute $profession = retired$. When it comes to range queries, capturing and managing these dependencies is non-trivial; so we will leave it for future work.

In our solution, we focused on protecting the intermediate (summary information) and final result from inference attacks with the use of *Differential Privacy*. However, we have not directly addressed the risks associated with side-channel attacks. It is easy to see that thanks to the collaboration method that we propose, we manage to avoid certain risks mentioned in [35], such as: *memory access models* and *communication volumes* since all data-based computations are performed locally at each data provider and the communication cost is constant and independent of the query. But we have postponed further consideration of this aspect of the problem to dedicated work.

Our solution serves as the first building block towards a more comprehensive solution that handles more complex queries, such as GROUP-BY queries. Integrating such clauses in the SQL query is not so trivial, and adding noise to the final result will not be enough to guarantee privacy [14]. Other aggregations, such as average, standard deviation, and variance, can be derived from SUM and COUNT using the sequential composition of DP. However, to handle other aggregations (such as Min, Max and Mode), different estimators are required.

Finally, during our evaluation, we built a proof of concept of our solution on *PostgreSQL*. It would be interesting to incorporate it directly into any DBMS, which would further improve our results.

8 CONCLUSION

In our study, we introduced a lightweight collaborative approach for online range query approximation in a federated environment. Our experimental results demonstrated the performance improvements our solution is capable of delivering, with processing times improved by up to 8x compared to plain-text execution, while ensuring end-to-end privacy with minimal loss of accuracy. Our solution uses cluster sampling and query estimation techniques that take into account data distribution to preserve query utility in terms of speed and accuracy. This work lays a solid foundation for future work to handle more complex queries while maintaining the same level of performance.

REFERENCES

[1] John M Abowd. 2018. The US Census Bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2867–2867.

[2] Swarup Acharya, Phillip B Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. 1999. The aqua approximate query answering system. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*. 574–576.

[3] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European conference on computer systems*. 29–42.

[4] Rakesh Agrawal, Dmitri Asonov, Murat Kantarcioglu, and Yaping Li. 2006. Sovereign joins. In *22nd International Conference on Data Engineering (ICDE'06)*. IEEE, 26–26.

[5] Hossein Ahmadvand, Maziar Goudarzi, and Fouzhan Foroutan. 2019. Gapprox: using gallup approach for approximation in big data processing. *Journal of Big Data* 6 (2019), 1–24.

[6] Johes Bater, Gregory Elliott, Craig Eggen, Satyender Goel, Abel N Kho, and Jennie Rogers. 2017. SMCQL: Secure Query Processing for Private Data Networks. *Proc. VLDB Endow.* 10, 6 (2017), 673–684.

[7] Johes Bater, Xi He, William Ehrich, Ashwin Machanavajjhala, and Jennie Rogers. 2018. Shrinkwrap: efficient sql query processing in differentially private data federations. *Proceedings of the VLDB Endowment* 12, 3 (2018).

[8] Johes Bater, Yongjoo Park, Xi He, Xiao Wang, and Jennie Rogers. 2020. Saqe: practical privacy-preserving approximate query processing for data federations. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2691–2705.

[9] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

[10] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. 2017. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th symposium on operating systems principles*. 441–459.

[11] Lei Cao, Dongqing Xiao, Yizhou Yan, Samuel Madden, and Guoliang Li. 2021. ATLANTIC: making database differentially private and faster with accuracy guarantee. (2021).

[12] Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. 2007. Optimized stratified sampling for approximate query processing. *ACM Transactions on Database Systems (TODS)* 32, 2 (2007), 9–es.

[13] Graham Cormode. 2010. Individual privacy vs population privacy: Learning to attack anonymization. *arXiv preprint arXiv:1011.2511* (2010).

[14] Damien Desfontaines, James Voss, Bryant Gipson, and Chinmoy Mandayam. 2020. Differentially private partition selection. *arXiv preprint arXiv:2006.03684* (2020).

[15] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[16] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.

[17] Saba Eskandarian and Matei Zaharia. 2017. Oblidb: Oblivious query processing for secure databases. *arXiv preprint arXiv:1710.00458* (2017).

[18] Olga Gkountouna, Katerina Doka, Mingqiang Xue, Jianneng Cao, and Panagiotis Karras. 2022. One-off disclosure control by heterogeneous generalization. In *31st USENIX Security Symposium (USENIX Security 22)*. 3363–3377.

[19] Inigo Goiri, Ricardo Bianchini, Santosh Nagarakatte, and Thu D Nguyen. 2015. Approxhadoop: Bringing approximations to mapreduce frameworks. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*. 383–397.

[20] Peter J Haas and Christian König. 2004. A bi-level bernoulli scheme for database sampling. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. 275–286.

[21] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. 1997. Online Aggregation. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*. 171–182.

[22] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2015. The composition theorem for differential privacy. In *International conference on machine learning*. PMLR, 1376–1385.

[23] Ala Eddine Laouir and Abdessamad Imine. 2024. Private Approximate Query over Horizontal Data Federation. arXiv:2406.11421

[24] Peeter Laud and Alisa Pankova. 2019. Interpreting epsilon of differential privacy in terms of advantage in guessing or approximating sensitive attributes. *arXiv preprint arXiv:1911.12777* (2019).

[25] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. 2016. Wander join: Online aggregation via random walks. In *Proceedings of the 2016 International Conference on Management of Data*. 615–629.

[26] John Liagouris, Vasiliki Kalavri, Muhammad Faisal, and Mayank Varia. 2021. Secrecy: Secure collaborative analytics on secret-shared data. *arXiv preprint arXiv:2102.01048* (2021).

[27] Sharon L. Lohr. 2009. *Sampling : Design and Analysis*.

[28] Joseph P. Near and Chiké Abuah. 2021. *Programming Differential Privacy*. Vol. 1. https://programming-dp.com/

[29] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 188–197.

[30] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 75–84.

[31] Frank Olken and Doron Rotem. 1986. Simple random sampling from relational databases. (1986).

[32] Frank Olken and Doron Rotem. 1995. Random sampling from databases: a survey. *Statistics and Computing* 5 (1995), 25–42.

[33] Gregory Piatetsky-Shapiro and Charles Connell. 1984. Accurate estimation of the number of tuples satisfying a condition. *ACM Sigmod Record* 14, 2 (1984), 256–276.

[34] Chengjie Qin and Florin Rusu. 2014. PF-OLA: a high-performance framework for parallel online aggregation. *Distributed and Parallel Databases* 32 (2014), 337–375.

[35] Lina Qiu, Georgios Kellaris, Nikos Mamoulis, Kobbi Nissim, and George Kollios. 2023. Doquet: Differentially Oblivious Range and Join Queries with Private Data Structures. *Proceedings of the VLDB Endowment* 16, 13 (2023), 4160–4173.

[36] Guangxuan Song, Wenwen Qu, Xiaojie Liu, and Xiaoling Wang. 2018. Approximate calculation of window aggregate functions via global random sample. *Data Science and Engineering* 3 (2018), 40–51.

[37] ADP Team et al. 2017. Learning with privacy at scale. *Apple Mach. Learn. J* 1, 8 (2017), 1–25.

[38] Lingyu Wang and Sushil Jajodia. 2008. Security in Data Warehouses and OLAP Systems. In *Handbook of Database Security - Applications and Trends*, Michael Gertz and Sushil Jajodia (Eds.). Springer, 191–212.

[39] Sai Wu, Beng Chin Ooi, and Kian-Lee Tan. 2010. Continuous sampling for online aggregation over multiple queries. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 651–662.

[40] Fei Xu, Christopher M. Jermaine, and Alin Dobra. 2008. Confidence bounds for sampling-based group by estimates. *ACM Trans. Database Syst.* 33, 3 (2008), 16:1–16:44.

[41] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)* 42, 4 (2017), 1–41.

[42] Xuhong Zhang, Jun Wang, and Jiangling Yin. 2016. Sapprox: Enabling efficient and accurate approximations on sub-datasets with distribution-aware online sampling. *Proceedings of the VLDB Endowment* 10, 3 (2016), 109–120.

[43] Wenting Zheng, Ankur Dave, Jethro G Beekman, Raluca Ada Popa, Joseph E Gonzalez, and Ion Stoica. 2017. Opaque: An oblivious and encrypted distributed analytics platform. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 283–298.