Statistical Privacy and Consent in Data Aggregation

Nick Scope
DePaul University
Chicago, IL, USA
nscope52884@gmail.com

Alexander Rasin
DePaul University
Chicago, IL, USA
arasin@cdm.depaul.edu

Ben Lenard
DePaul University
Argonne National Laboratory
Chicago, IL, USA
blenard@anl.gov

James Wagner
University of New Orleans
New Orleans, LA, USA
jwagner4@uno.edu

Abstract

As new laws governing management of personal data are introduced, e.g., the European Union's General Data Protection Regulation of 2016 and the California Consumer Privacy Act of 2018, compliance with data governance legislation is becoming an increasingly important aspect of data management. An important component of many data privacy laws is that they require companies to only use an individual's data for a purpose the individual has explicitly consented to. Prior methods for enforcing consent for aggregate queries either use access control to eliminate data without consent from query evaluation or apply differential privacy algorithms to inject synthetic noise into the outcomes of queries (or input data) to ensure that the anonymity of non-consenting individuals is preserved with high probability. Both approaches return query results that differ from the ground truth results corresponding to the full input containing data from both consenting and non-consenting individuals. We present an alternative framework for group-by aggregate queries, tailored for applications, e.g., medicine, where even a small deviation from the correct answer to a query cannot be tolerated. Our approach uses provenance to determine, for each output tuple of a group-by aggregate query, which individual's data was used to derive the result for this group. We then use statistical tests to determine how likely it is that the presence of data for a non-consenting individual will be revealed by such an output tuple. We filter out tuples for which this test fails, i.e., which are deemed likely to reveal non-consenting data. Thus, our approach always returns a subset of the ground truth query answers. Our experiments successfully return only 100% accurate results in instances where access control or differential privacy would have either returned less total or less accurate results.

CCS Concepts

• **Security and privacy** → **Data anonymization and sanitization**; *Information accountability and usage control*; • **Mathematics of computing** → *Probability and statistics*; Statistical software.

SSDBM 2024, July 10–12, 2024, Rennes, France

ACM ISBN 979-8-4007-1020-9/24/07
https://doi.org/10.1145/3676288.3676298

Keywords

GDPR, Compliance, Processing consent, Privacy

ACM Reference Format:
Nick Scope, Alexander Rasin, Ben Lenard, and James Wagner. 2024. Statistical Privacy and Consent in Data Aggregation. In *36th International Conference on Scientific and Statistical Database Management (SSDBM 2024), July 10–12, 2024, Rennes, France.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3676288.3676298

1 Introduction

Laws are currently being passed and refined, placing new restrictions on how organizations have to store, process, and destroy customer data to improve consumer privacy. Two examples of these new laws are the European General Data Protection Regulation (GDPR) [2] and the California Consumer Privacy Act (CCPA) [1]. Among other restrictions, these laws require that organizations only use an individual's data for purposes the individual has explicitly consented to. Organizations which do not comply with these consent requirements are subject to heavy fines. For example, in 2020 a €28.7 million fine was imposed on the Italian communications company TIM [3] for (among other things) using customer data for marketing without having customer's consent for using their data for marketing purposes. To avoid such fines and comply with legislation, organizations must implement data privacy support for compliant data processing. For example, GDPR outlines multiple requirements for processing customer data. Any business with customers in the European Union must comply with GDPR requirements, regardless of where the organization is headquartered [2]. GDPR Article 6 outlines when and how an organization can process customer's personal data: "the data subject has given consent to the processing of his or her personal data for one or more specific purposes;". Only in a few cases can consent be assumed; for example, one may use customer's address to ship the order for which the address was provided without collecting explicit consent. GDPR is one of several laws that may apply to data processing in an organization. Many organizations target satisfying GDPR requirements as a way to ensure compliance with all other policy sources, because GDPR typically has the strictest requirements.

In this paper, we propose and evaluate a statistical privacy-preserving framework for group-by aggregate queries over data from individuals that have *consented* for their data to be used for

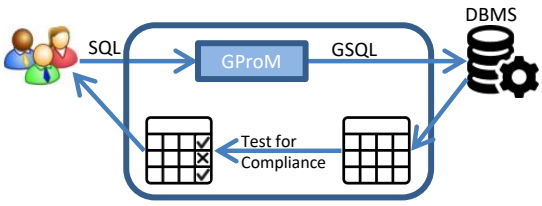

Figure 1: Summary of consent validation steps. GProM is a middleware created by Arab et al. [8] that computes provenance in relational databases.

specific *purposes*. We consider the following setting. An organization has collected data from a set of individuals $\mathcal{I}$. Each *individual* I whose data D_I was collected by the organization, has consented for this data to be used for a set of purposes P_I which is a subset of a set $\mathbb{P}$ of possible purposes relevant to the organization.[1] Each query Q run by the organization is associated with a purpose p. [2] Based on each individual's set of consented purposes P_I, we can now distinguish between tuples from individuals that consented to the query's purpose ($D_C = \bigcup_{I \in \mathcal{I}: p \in P_I} D_I$) and those who do not ($D_{\mathcal{N}} = D - D_C$). Thus, each tuple from the database D falls into either of these two categories:

$$D = D_C \cup D_{\mathcal{N}}$$

We define *consent-abiding query processing* for a group-by query Q as returning a filtered answer $A_C \subseteq Q(D)$ such that for each tuple $t \in A_C$ (the result for one group) we can demonstrate that it is unlikely that information from any non-consenting individuals that contributed to the result t (i.e., that belong to the provenance of t, get exposed by t). We test for leakage of information from non-consenting individuals by analyzing whether the inclusion of data from the individual can be inferred from the distribution of input values for the aggregation of the group for tuple t. While the user issuing the query will only observe the aggregated result t, testing that the existence of data from a non-consenting individual is unlikely to be inferred from the input data for a group is a strictly stronger condition that prevents privacy violations when the user running the query has background knowledge about the distribution of values in a group. We test whether it is safe to return t as follows. First, we verify the number of individuals contributing to the aggregation result for t is sufficiently large, so that subsequent calculations have a large enough dataset to execute reliable statistical testing to verify privacy. If so, we perform a statistical test on the distributions used in query's aggregations to verify that the distribution for any single non-consenting individual does not meaningfully differ from the other individuals in the corresponding group for t. Although this framework is agnostic, we use both sample size and goodness-of-fit tests to ensure privacy is preserved for non-consenting customers.

[1]We assume here that the set $\mathbb{P}$ is static. However, the techniques we develop are also applicable when new purposes are added to $\mathbb{P}$ over time.
[2]Our approach can trivially be extended to allow queries that are associated with multiple purposes P_Q. The only impact this change would have on our approach is that we would have to test whether $P_Q \subseteq P$

Prior work (e.g., [22, 23]) has approached the problem of consent-abiding query processing as an extension to access control. For example, if the query purpose is p ="marketing", the records of individuals I which do not have consented for marketing ($marketing \notin P_I$) are excluded from the query's input. Although this guarantees compliance, this approach will produce results for a group that differ significantly from the ground truth, the result computed for the group over the full database containing data from both consenting and non-consenting customers. In some domains (e.g., healthcare), inaccurate query results (even those off by fractions of a percent), may not satisfy data accuracy requirements. In contrast, our approach excludes results that are deemed privacy risks with regard to consent, but every tuple returned is part of the ground truth query answer $Q(D)$.

Figure 1 shows our consent verification process. Using GProM [8], for each group (corresponding to a result tuple t), we calculate its provenance, i.e., the input tuples from D that contributed to the group. Furthermore, we determine which of these tuples belong to data from consenting / non-consenting individuals. Using the source tuples, we implement the necessary statistical testing to determine whether each aggregation calculation can be considered sufficiently privacy preserving. In sum, our contributions are:

(1) We outline the requirements a database must support to guarantee consent compliance in data processing
(2) We develop a framework for *consent-abiding query processing* that preserves privacy of non-consenting individuals with high probability by removing query result tuples for which we cannot guarantee with high likelihood that the privacy of non-consenting individuals is preserved. Instead of perturbing query results, as is the case for DP mechanisms and access control based solutions, we return a subset of the ground truth query results. Our approach utilizes well-known statistical tests to determine whether the distribution of input values for aggregation for a group is significantly affected by the data of a non-consenting individual.
(3) We implement this framework in a relational DBMS (PostgreSQL) to evaluate its overhead and validate compliant processing.

2 Privacy Requirements

2.1 Domain Terminology

Anonymous Data: GDPR Recital 26 defines anonymized data as "information which does not relate to an identified or identifiable natural person or to personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable" [2]. Therefore, in this paper, we check whether a customer can be statistically detected (by identifying the presence of their value in aggregate output) without using additional information [2].

Business Record: Data governance policies operate in terms of "business record" units. In a database, a business record may span combinations of rows across multiple tables (e.g., a purchase order consisting of a buyer, a product, and the purchase transaction from three different tables). In this paper, we use select-project-join (SPJ) views to define business record units.

2.2 GDPR Processing

Not all types of processing of customer data requires explicit consent. For example, an organization does not need consent to use a customer's address for the purpose of shipping the order, after the customer placed the order and provided their shipping address. Thus, not all queries will require consent validation.

Storage requirements with privacy laws are different than processing requirements. An organization can store data to process it for the purpose for which it was collected, but this does not necessarily mean that the organization can process the data for another purpose. For example, a company may store a customer's address to ship them an order; conversely, using this address to send marketing materials may result in a privacy compliance violation (without explicit permission). We consider data storage privacy compliance functionality research to be beyond the scope of this paper.

With respect to processing, GDPR defines *pseudonymisation* as "the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information, [and] the personal data are not attributed to an identified or identifiable natural person" [2]. If the underlying customers are not identified by the query output, there is no need to verify consent for the business records used in processing. However, simply because customer data is aggregated does not automatically mean that their data is sufficiently unidentifiable. A sufficiently small or sufficiently skewed input sample may allow inferring identifying information about the underlying customers. This framework provides a statistical approach to probabilistically determine if there is a risk of being able to infer information about the customers contributing to an aggregation.

2.3 Related Work

Ataullah et al. [9] developed a compliance framework that enforces data retention policies, blocking the deletion of records that are protected by data retention rules. They defined protection policies as a SQL query, using SPJ views. Scope et al. [24] used a similar approach to define compliance policies for purging of data. Neither of these frameworks limit how business records can be used.

Wuyts et al. [27] implemented access control to achieve consent compliance in healthcare data. Although this does guarantee that non-consenting data is not processed, access control may overly filter data, affecting the output query's correctness and usefulness. Our framework only filters out data of the entire aggregation if the result would risk the privacy of the contributing customers. With access control, either the result would be filtered (matching our restrictiveness) or a subset of the input would be filtered, biasing the results. Pappachan et al. [23] used a framework to rewrite queries to enforce privacy processing compliance via access control, limiting query access to data whose owner has not given consent. We expand on access control mechanisms by implementing statistical testing to verify exposure risk. Our goal is to maximize query correctness by returning precise aggregation results or excluding the individual non-compliant aggregation results if that's not possible.

Brahem et al. [10], discussed how to protect consent by requiring a ℓ-completeness. They used a system similar to k-anonymity with multiple data sources. Specifically, this differs from our research by aiming to maintain privacy across multiple sources, where our framework focuses on maximally compliance of a single aggregation query in a database. Furthermore, our consent-abiding query processing framework uses statistics to determine identifiability among other customers instead of some determined k amount used with k-anonymity approaches.

Hozel [18] investigated how differential privacy may satisfy GDPR processing requirements. Differential privacy ensures customer anonymity by adding synthetic noise into the data. Depending on the technique, some apply this noise post calculation, while other research adds noise before the aggregation; regardless, neither is guaranteed to return completely accurate results. The overall goal of differential privacy is to balance privacy with accuracy (i.e., injecting a higher volume of noise into the data should increase the privacy of the customer data). In instances where introducing output noise is considered acceptable, differential privacy may satisfy both organizational priorities and privacy compliance requirements. In settings like the medical field, the tolerance for any intentional error is deemed unacceptable. For example, injecting noise into the weight of a patient may bias the results of a study.

In real-world setting, the habits of malicious attackers may not be easily discernible from user activity. Multiple works of research that focus on the identification of malicious attackers behavior patterns (e.g., Stevanovic et al. [25]). However, our framework focuses on protecting privacy from non-malicious user behavior (and thus we do not need to differentiate between the two).

Cheney et al. [12] provide a comprehensive overview of provenance in databases. They describe provenance information as, "[the] origins and the history of data in its life cycle." Ni et al. [22] used provenance to develop an access control framework. In their solution, they developed a system to determine how data evolves within a database which supports fine-grained access policies that are based on data provenance, e.g., all data derived from sensitive data should be subject to the same access control restrictions.

Konstantinidis et al. [20] investigated how provenance can be used to achieve consent compliance in databases. Their framework used query rewrites to achieve policy-compliant query outputs by filtering out values whose inclusion would result in a non-compliant output. Specifically, they use provenance to map a type of access control to remove outputs whose sources are not permissioned. Our framework expands on this work with statistical tests to determine if query aggregations sufficiently preserve privacy before applying any (potentially unnecessary) filtering. Drien et al. [16] implemented consent compliance by comparing output tuples to the input tuples to determine if the output tuples of a query can be mapped to the inputs used. Their research achieves the level of fine-grained compliance necessary to minimally remove tuples whose inclusion would result in non-compliance. Unlike our research, the work by Drien et al. [16] does not apply to aggregations but select-project-join queries.

Arab et al. [8] developed GProM, a provenance middleware for relational databases. We use GProM to compute provenance of queries in our framework to determine which input data was used to computed a query result tuple. This information is then used to test for consent compliance (see Section 6).

3 Privacy and Threat Model

We present the privacy model on which our framework is based and discuss the threat model, i.e., what kind of background knowledge we assume that the user issuing the query has access to.

3.1 Privacy Model

GDPR requires that data belonging to a non-consenting customer is excluded from processing. However, as per Article 4 of GDPR, customer records may be used for processing as long as customer anonymity is preserved. We define *anonymity* as the inability to associate personal data with a specific underlying customer and the inability to discover the presence of a single customer's data from query results. Chaudhuri and Mishra [11] defined the privacy of two tables as *"[...] for any pair of tables T and T′ that differ in only one position, privacy is preserved if a hypothetical attacker upon seeing the transcript is unable to distinguish between the case when the actual table is T or T′"*. Non-consenting data can still be used in an aggregate (e.g., average income per state), as long as the underlying customer data is not identifiable without sources of additional information (i.e., additional queries or knowledge from outside the database). As mentioned earlier we target group-by aggregate queries. Specifically, we support queries Q of the form where f is an aggregation function (we support `MEAN`, `MEDIAN`, `MODE`, `MIN`, and `MAX`), a is the attribute we are aggregating over and G is a set of group-by attributes (e.g., `SELECT f(a), G FROM R GROUP BY G;`).

We assume that the organization's database stores information about a set of individuals: $\mathcal{I} = \{I_1, \ldots, I_n\}$. Each individual $I \in \mathcal{I}$ has consented for their data to be utilized for a specific set of purposes $P_I \subseteq \mathbb{P}$ where $\mathbb{P}$ is a set of purposes of interest to the organization. For an individual I, we use D_I to denote the set of tuples from the organizations database D that contain information about I. Thus, we assume that $D_{I_1} \cap D_{I_2} = \emptyset$if $I_1 \neq I_2$.

An SQL extension for specifying consent will be discussed in Section 4. We assume that each query Q is associated with a purpose $p \in \mathbb{P}$. Given such a purpose, the database can be divided into two subsets D_C, which contains data from consenting individuals wrt. p and all data that is not associated with any individual (e.g., the inventory of an organization), and $D_{\mathcal{N}}$, which contain data from individuals I which have not consented for their data to be used for the purpose p of the query ($p \notin P_I$). Formally,

$$D_{\mathcal{N}} = \bigcup_{I \in \mathcal{I}: p \notin P_I} D_I \qquad D_C = D - D_{\mathcal{N}}$$

For a query Q and database D, consider the answer of the query evaluated over the full database (containing data from both consenting and non-consenting individuals):

$$A = Q(D)$$

Consider a tuple $t \in A$, we use $Prov(Q, D, t)$ to denote the Lineage [8] of t which is the set of input tuples from D that were used to compute t by Q. For a group-by aggregation query these are all tuples from R that belong to the group for t, i.e., that have the same values in the group-by attributes. We use $\mathcal{G}[t]$ to denote this set of tuple and $\mathcal{G}_C[t]$ ($\mathcal{G}_{\mathcal{N}}[t]$) to denote the subsets of $\mathcal{G}[t]$ corresponding to consenting (non-consenting) individuals.

$$\mathcal{G}[t] = \{t' \mid t' \in D \wedge t'.G = t.G\}$$

$$\mathcal{G}_C[t] = \mathcal{G}[t] \cap D_C \qquad \mathcal{G}_{\mathcal{N}}[t] = \mathcal{G}[t] \cap D_{\mathcal{N}}$$

We use a predicate $private(Q, \alpha, t, \mathcal{G}_C[t], \mathcal{G}_{\mathcal{N}}[t])$ that determines whether t is statically unlikely (with confidence α) to leak information about any data from $\mathcal{G}_C[t]$, i.e., data from individuals that have not consented to the purpose of the query. We will provide a concrete realization of this predicate based on statistical tests in **??**. With the predicate we are ready to define a concrete semantics for consent-abiding query processing.

Definition 3.1 (Consent-abiding Query Processing). Consider a group-by query Q of the form shown above with purpose p evaluated over a database D with individuals $\mathcal{I}$. We define the *consent-abiding answer* A_C of Q for a given confidence threshold α to be:

$$A_C = \{t \mid t \in Q(D) \wedge private(Q, \alpha, t, \mathcal{G}_C[t], \mathcal{G}_{\mathcal{N}}[t])\}$$

We define *query correctness* Γ as the fraction of ground truth query results that are returned by consent-abiding query processing:

$$\Gamma = \frac{|\ A_C\ |}{|\ A\ |}$$

Most of related work focused on filtering non-consenting customer data from queries; in this paper, we seek to return as many correct query output records as possible. We connect this definition of privacy to statistical techniques by arguing that sufficiently large and balanced groups in aggregations preserve the anonymity of the underlying customers (and, by extension, their privacy). Our framework allows organizations to apply statistical techniques to set thresholds for privacy compliance in their aggregations. Specifically, by having aggregations calculated on groups of sufficient size and balanced contributions from the underlying members, we argue that these groups are able to sufficiently protect the identifiability and privacy of their underlying members. Figure 2 illustrates sample steps of refining query aggregation.

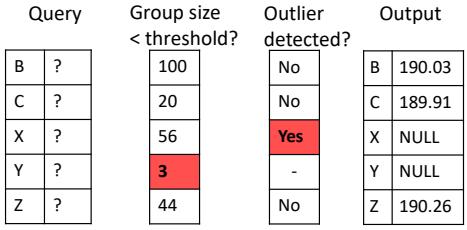

Figure 2: Anonymity-preserving query processing.

3.2 Motivation and Threat Model

Differential privacy approaches (see Section 2.3) by their inherent nature inject some level of noise into data in order to maintain the privacy of the underlying individuals. Although various approaches inject the minimum amount of noise possible, in instances where any level of inaccuracy is not allowed, a differential privacy approach will not satisfy zero accuracy compromise requirements. Instead, an approach is needed to ensure that a query can be computed without any noise injection, while still guaranteeing the privacy of the underlying contributing individuals.

The approach outlined in this paper ensures privacy by applying statistical techniques to data used in the aggregation. Intuitively, if a customer cannot be identified (via inclusion and impact) within a group, their identifiability has been masked. First, we ensure that aggregate group sizes are sufficiently large to prevent the determination of whether a particular customer's data is included in the computation. Second, we ensure each group's non-consenting contributing members cannot be detected because they bias the aggregation results by a statistically detectable amount. Finally, we ensure that each customer is not detectable because it is not overly represented in aggregate computation, independent of their individual value's statistical impact on the aggregation. If a single non-consenting customer's inclusion would detectably change aggregation output, it may be possible to identify the underlying individual's input and risk their privacy.

We assume that a user is well-intentioned and has a general domain knowledge of the underlying data. Furthermore, that user has access to the data for legitimate business purposes but is not permitted (due to privacy rules), to have access to all customer data for all purposes. Therefore, the user who has a rough understanding of what an outlier would look like must not be able to identify underlying customers within the aggregation results. For example, if the user expected an average of 10 with a standard deviation of 1, if the average returned were 14, the user could safely assume the presence of an extreme outlier in the group.

Our analysis only considers computations done in an individual single query. Additional information can be discovered by comparing the results of multiple queries. One current limitation of this framework is the database has no way of verifying the purpose (e.g., "marketing") of the query provided at execution time, thus we assume no nefarious users attempting to circumvent the controls.

4 Business Records and Consent

Our framework uses a business record to map customer consent for their data; we use SPJ definitions, previously used in [9, 24] for defining business records with additional requirements, such as defining the "primary key". `CREATE CONSENT` (a new SQL statement type, similar to `CREATE VIEW`) defines a business record to corresponding consent permissions. Our implementation requires that a business record definition includes a primary key using `CUSTOMER` command to determine how an individual is reflected in the schema:

```
CREATE CONSENT customerweights CUSTOMER c.custID, v.vitalID
SELECT c.custID, c.fullname, c.race, v.vitalID, v.weight
FROM customer AS c NATURAL JOIN vital AS v;
```

We then store consent collected per customer in the database (which is ultimately a parameter that is collected and maintained throughout a customers lifecycle in the database). This is done by using an `INSERT CONSENT` command to denote the business record, customers, and type of consent given (with other similar commmands for deleltion and updates).From this, we use provenance of a query to determine when a customer's data is included in a query and what consent they have given by determining the overlap of the returned source tuples with the customer's business records.

5 Determining Privacy Leakage

We now are ready to discuss how, given a result tuple $t \in Q(D)$ of a query Q, $\mathcal{G}_C[t]$ and $\mathcal{G}_N[t]$, and a confidence level α, to determine whether exposing t to the user issuing the query does not violate privacy with a confidence of at least α. With respect to α, we use this value as the corresponding p-value threshold used to either reject or accept the null hypothesis (i.e., if a customer's presence risks identifiability). We apply two categories of statistical tests: *Group Size* (Section 5.1) and *Group Balance* (Section 5.2). The aggregation is considered sufficiently anonymous with level α if it passes an appropriate test in both categories. *Group* refers to the cohorts defined by the `GROUP BY`. We measure group size with respect to the number of customers within each cohort. With balance, we measure the number of business records (i.e., output tuples) of each member to the aggregation calculations for each cohort.

With respect to group size, a larger sample size naturally offers some basic anonymity protection for its members due to the impact of each member decreasing as the sample size increases (per the central limit theorem). For example, organizations that execute employee surveys typically will not give a manager the results of their direct reports if a group is too small.

With group balance, even with sufficiently large groups, a single member's inclusion may change the distribution of their group by a statistically significant margin (and thus impact the subsequent aggregations on the distribution). This may risk privacy leakage, letting database user infer if an individual is a member of an aggregated group. Therefore, we claim that by insuring a non-consenting individual does not impact a metric's distribution by a statistically significant margin (and that the same group has a sufficiently large sample), that privacy has been preserved.

For example, suppose an organization has a customer who is a CEO at another organization and does not consent to marketing. Marketing wants to determine the average salary of their customers by occupation. Assuming this organization has a sufficiently large number of CEOs to satisfy the minimum group size requirement, we claim that the addition of a single CEO does not introduce a statistically significant impact. Using statistical testing with a significance level (i.e., α) of 0.05, if the CEO's salary were replaced with another CEO's salary, the average of the resulting distribution would not differ significantly enough to conclude a divergence in the calculated aggregation; therefore, we would fail to reject that the mean salary of that specific group with and without the particular CEO are different. Thus, the CEO's identifiability (due to inclusion of the aggregation) has not been violated, offering evidence for privacy preserving processing.

A distribution preserves the privacy of its members when it ensures that the individual contributions to the dataset are indistinguishable to a well-intentioned observer, thereby safeguarding personal information. An aggregation performed over such distribution would also be privacy preserving because the aggregation results would not reveal contributions of individual data points. This framework offers statistical guarantees that when data is analyzed in aggregate, no additional information about any individual can be inferred, maintaining the privacy of all members of the dataset, as long as our statistical testing conditions are satisfied.

This differs from access control methods (where data may be unnecessarily filtered) and differential privacy techniques (which inject artificial noise) by simply returning 100% correct and inclusive results or filtering out the aggregations who would risk disclosing information about non-consenting members.

5.1 Group Sizes

In statistical analysis, larger samples result in more accurate calculations. Thus, individual members of a sample have a lesser impact on the calculations as the sample size grows. We connect this to the concept of consent compliance by arguing that larger groups better protect the privacy of the underlying data elements (as the influence of outliers is reduced). With a group of significant size, any single customer's data value would not observably impact the aggregate calculation and allowe the underlying customer to be identified. Therefore, by using a similar definition of privacy by Chaudhuri and Mishra [11], we use size of the aggregated dataset (instead using statistical testing) as a consideration for protecting underlying customer privacy.

Assuming a balanced group, drawing conclusions or deriving any single underlying value is statistically improbable. We use *population* to refer to an organization's entire customer base and *group* to refer to a subset of customers (which belong to the same `GROUP BY` bucket). Although these groups are not randomly selected from the population, our goal of privacy compliance aims to ensure that we can neither determine if a customer is in a given group nor what their contributions were (for an individual query). Since we do not attempt to use the groups to draw statistical conclusions about the overall customer base population, the absence of sample randomness does not invalidate our process.

We use Yamane's Formula [28] to determine a minimum `GROUP BY` size required to guarantee that an underlying customer's business record privacy is sufficiently preserved to process the data without explicit consent. Using N (the population size of an organization's customers) and e (the precision parameter), Yamane's Formula of $n = \frac{N}{1+(N\times e^2)}$ provides the recommended group size we use as the minimum threshold for group size compliance.Because a recommended sample size is used as guidance for stability in a distribution, we argue this provides guidance for when a single new observation (i.e., customer) will not drastically alter the distribution.

Setting α scores or precision threshold in statistics has many considerations. For general statistical studies, an α of 0.05 is considered the default (i.e., most domains use this as a best practice value). By having a standardized accepted α, there is a lower probability of an individual changing the score until their finding becomes "statistically significant" (i.e., p-value hacking).In some domains, a less sensitive (e.g., smaller e or α value) is considered acceptable (resulting in a higher group size requirement). Furthermore, research by Di at al. [15] has led to the suggestion that the default should be set closer or at 0.005. Making a less sensitive precision comes with the cost of requiring a larger group size, but it also lowers the influence that each value in the group has on an aggregation.

Overall, we recommend a default precision value e set at a maximum of 0.05; a larger value risks an unacceptable amount of privacy leakage (which interacts with the other function parameters).Moreover, this value should never be changed to achieve a specific result in a query; the precision must remain consistent for all queries (which is set by an organizations privacy requirements determined by their legal and data experts). The precision parameter of 0.05 translates to: after sampling a population 20 times, the aggregate output of a sample would negligibly differ that of the population 19 out of 20 times. In this case, we use this to compare the stability of the group.From this, we argue that the different groups would have a statistically insignificant difference when compared to each other (19 out of 20 times), thereby preserving the privacy of the underlying customers used to calculate the aggregate.

Depending on an organization's policies, an alternative formula for the group size threshold calculation may be preferred. For example, some policies have a specific threshold recommendation (e.g., HIPAA [17] explicitly recommends a minimum of 20,000 "inhabitants" for data aggregations). Although we default to Yamane's Formula due to its low amount of input parameters, there are instances when a more complex formula (e.g, Cochran's Formula [13]) results in the ability to process a lower group size. One trade-off is a more computationally intensive calculation (i.e., a higher overhead) for equations which require additional information (e.g., standard deviation).Our framework allows both a pre-set threshold and alternative group size formula calculations to be used.

We default to using Yamane's Formuala, which simply requires a count calculation on the data. Running the group size before the group balance allows us to potentially skip the more costly group balance calculations if the group size requirement is not met.Therefore, running group size before group balance offers optimisation benefits. While statistical tests that we subsequently apply offer protections, they offer more statistical power using sufficiently large sample sizes.A foundational element of statistical testing is understanding the power of a statistical test [14]. The power of a statistical test is the probability of rejecting H_0 (i.e., the null hypothesis) when the null hypothesis is false. The power of a statistical test is—in part—determined by the sample size.If the sample size is too small, any subsequent statistical test may not have enough power to accurately derive results. In this case, we do not apply tests for distribution balance testing if the underlying group size is not large enough to guarantee a statistically stable result.

5.2 Group Balance

Once sufficient group size has been verified, we further test to see if anon-consenting customer stands out in their group as an outlier or due to contributing a disproportionate amount of values in the aggregation. If group size was not verified, this step is not executed. For group balance, we apply two types of tests: 1) no non-consenting customer has a statistically discernable difference in data distribution (compared to the aggregate) and 2) no non-consenting customer is over-represented in the aggregation. For example, consider the following SQL query:

```
SELECT p.race, MEAN(v.weight)
FROM patient AS p NATURAL JOIN vital AS v GROUP BY race;
```

If we have a sufficient group size of patients within a race to meet our anonymity threshold, we test for a possibility that a single non-consenting patient may overly influence the result or make up a disproportionate amount of the weight measurements within a race (thus risking their privacy by identifiability).

Statistical Discernability of Outlier Customer Distributions: To determine whether a single customer's records are statistically differentiable from the other records in their group, we use a Two-Sample Kolmogorov-Smirnov (KS) test [21]. A KS test determines the probability that two observed groups originate from the same underlying population distribution. If the results are statistically significant (i.e., the probability is less than the accepted alpha), we can conclude that the underlying distributions are different and it is possible to derive information about the underlying groups (potentially leaking personal information of a customer).

A KS test has the basic assumptions that the groups are independent and identically distributed within each group. Although we accept that there is a bias simply by defining each group (e.g., the definition of the `GROUP BY`). Because the two groups of individuals are mutually exclusive with their data (i.e., the customer in one group does not appear in the other group), we believe the first assumption is at least minimally satisfied.Although the values themselves are not independent (e.g., the weight of a customer at one time is not independent from their weight at another time), because our intent is to judge the distributions to each other, we do not believe this hinders the application of the test. In instances where the data was captured over time, a time-series model with differencing may be more appropriate. Furthermore, we believe the identically distribution within each group is satisfied due to the aforementioned purpose. The KS test does require a distribution of data (i.e., a single data point would not satisfy this requirement). In instances where an outlier non-consenting customer only has a single value of data, an alternative test (i.e., other goodness-of-fit test) is necessary). For the purposes of this initial research and our experiments, we assume that this criteria has been satisfied.

Our framework compares the following groups: 1) the records belonging to the single non-consenting customer and 2) the entire group (with the exception of the aforementioned non-consenting customer). The KS test does not require any specific distribution nor equal variances in the underlying data.

Whitenall et al. [26] analyzed information leakage using a KS test and compared these results to other entropy-based Mutual Information testing techniques. Under our requirements, information leakage of a non-consenting customer would violate privacy compliance; thus, applying a KS test as a means to test for information leakage is considered a standard use-case. Overall, Whitenall et al. [26] found that a KS test detected information leakage even with a weak signal difference (although there are instances where other tests perform similarly or may be preferable).

For example, consider a single non-consenting customer with five weight measurements. The customer's average weight was 200 while the total average of their corresponding group was 180 (with a standard deviation of 40). These distributions are illustrated in Figure 3. Our framework runs the KS test on the entire group without the customer's inclusion compared to the single customer's data to examine if there is a statistically significant difference of the underlying distributions. If there is a statistically significant difference, there is a risk of information leak. Consequently, we do not allow the group's aggregate calculation to be returned. In this example, the test did not yield a statistically significant result, allowing the group calculation to proceed.

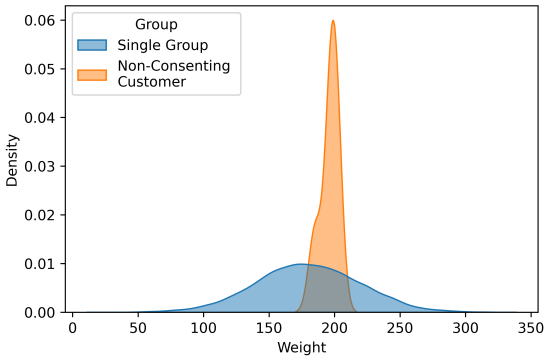

Figure 3: Distribution Comparison Example

To determine the customer, one would need to find the worst-case customer that is non-consenting. This would be done by applying the test using a leave-one-out approach using the desired statistical test (which we default to the KS test). For our initial performance evaluation, we assume the furthest aggregation from the remainder of the group is the most likely to risk privacy leakage.

Identifying Over-Represented Customers: We further evaluate the number of records each customer contributes to an aggregation function. If a customer contributes a disproportionate number of records to an aggregation, there is a risk information leakage about the customer. We define *outliers* as customers who contribute a disproportionate number of records to the aggregate.

We use the interquartile range rule (IQR) for detecting outliers (using the number of records contributed). This IQR rule does not require any particular data distribution. We determine how many records each customer contributes and determine what is considered the typical amount. Customers who contribute a number of record that fall above the allowed number are classified as outliers (low contributing non-consenting customers are not a concern).

Given Q_3 as the value of the third quartile, Q_1 as the value of the first quartile, and M as the value of the median, the IRQ is computed using $IQR = M \pm (Q_3 - Q_1) * 1.5$. Although typically the IQR rule logic uses Q_1 and Q_3 as a method for calculating the acceptable range for non-outliers, Q_1 and Q_3 may be adjusted to use alternative quantiles (depending on an organization's privacy needs and requirements). If a non-consenting customer is considered an outlier on the upper bounds, we consider the query's group to be imbalanced and, consequently, in violation of consent processing rules. For this, only need to compare the non-consenting customer with the most business records to the rest of the customers in the aggregate group. We use the IQR rule due to its generalizable application and a relatively simple formula. As with group size calculations, this formula may be substituted with another outlier detection technique depending on organizational needs, privacy threshold, and underlying data distributions.

In addition to the prior statistical evaluation, identifying over-represented customers is essential for privacy, particularly when considering certain aggregates, such as the mode. Even if a customer's data distribution aligns with the overall distribution of their aggregate group, a disproportionate representation can pose a privacy risk for that customer.

5.3 Verifying Privacy

We propose a new command `SELECT CONSENT [consent type]` to activate our framework. Our framework is currently implemented as a Python-based middleware. Regular `SELECT` queries (i.e., queries without `CONSENT`) would not require additional computation and thus do not impact performance. Only the queries with the input `SELECT CONSENT [consent type]` activate our consent verification computations, calculate the provenance of the query (and corresponding data), intercept the output of the query, and remove data from the output whose inclusion would lead to non-compliance. Using our previous business record example, the following query would be used to verify consent compliance for a marketing query:

```
SELECT CONSENT MARKETING customer.race, AVG(vital.weight)
FROM customer NATURAL JOIN vital GROUP BY customer.race;
```

First, our framework determines if the input query targets any columns defined under a customer business record definition (e.g., if a weight column is aggregated and covered by consent rules). If none were targeted, the query would proceed as normal. Otherwise, our framework must verify that the `GROUP BY`s contain a sufficient number of records and that each groups is sufficiently balanced (i.e., no non-consenting customer makes up a disproportionate amount of contributing records). Any group which only contains consenting customers may proceed without the group size and group balance checks. With Yamane's Formula, we use the total number of records defined under the applicable business record policy to determine the minimum group size required. For example, given an organization which has 10,000 customers (and each customer makes up a single business record), with a precision of 0.05, each group is required to have at least 385 customers (regardless of the number of tuples used for the aggregation). Any group which does not meet this requirement would have their aggregate calculation excluded.

For the groups which do contain a sufficient number of records, we examine whether any single non-consenting customer overly influences a groups distribution. If so, we conclude that privacy leakage is possible, and thus, we do not proceed to calculate and return the corresponding aggregation. If a KS test determines the distributions of the customer and the other group data are statistically indistinguishable, we conclude that there is no risk of privacy leakage and thus, the calculation can proceed.

Finally, for the groups which passed the previous verification steps, we determine if a non-consenting customer makes up a disproportionate number of the records. If no customers are considered outliers or a consenting customer is considered the most egregious outlier, the computation can proceed; if a non-consenting customer is an outlier, the framework returns NULL instead of the aggregate.

5.4 Supported Aggregations

Our framework supports numerical aggregations (e.g., string aggregations are beyond the scope of this paper). We currently support `MEAN`, `MEDIAN`, `MODE`, `MIN`, and `MAX`. The nature of `COUNT` aggregation prevents a single customer from being overly influential. Thus, to support `COUNT`, we still require a sufficient sample size and an adequately balanced number of underlying records contributed per customer, but do not evaluate the underlying customer data distribution influence.

Our framework is designed to be customizable, allowing organizations to incorporate various statistical functions and parameters. Tests can be chosen based on the required aggregations and organizational needs. Typically, a KS test or another goodness-of-fit test meets the privacy testing requirements supported by this framework. For other specific needs (e.g., a domain specific testing approach), alternative statistical tests can be used. For example, a KS test would not sufficiently analyze the difference of the maximums of two groups. Although the KS test is designed to test for overall distribution distinguishability, organizations may wish to implement an alternative information leakage test (e.g., an alternative mutual information differential power analysis).

An alternative statistical test supported by this framework for a `MIN`, and `MAX` aggregation is a different goodness-of-fit test on the customer's data. In that test, the framework excludes the non-consenting customer whose values are the minimum or maximum of the data. It then would compare whether or not the customer's data appears to be drawn from the same underlying population (in this case, the corresponding `GROUP BY` group). For a goodness-of-fit test, the null hypothesis (i.e., the default) is that the customer's data might belong to the same distribution of the group; the alternative hypothesis is the customer's data does not belong to the same pattern distribution of the group. If the goodness-of-fit test fails to reject that the customer belongs to the group, we then can safely use the customer's data in the aggregate calculation. These alternative tests were implemented but not evaluated in this paper.

In this paper, simple statistical tests (specifically, IQR and KS tests) were chosen to provide a proof-of-concept for this framework's process. In specific domains, other statistical approach may be preferred due to industry best practices. The tests we chose were designed to be general (i.e., domain agnostic) with the least amount of underlying assumptions (i.e., non-parametric).

6 Experiments

6.1 Framework Integration in a Database

We implemented a prototype system in PostgreSQL to demonstrate how our framework enforces consent compliance and to evaluate the associated overhead costs. Since databases do not support triggers on `SELECT` queries, this framework is implemented as a proxy (see Figure 1). In our evaluation, we implemented the framework in Python as a standalone application. The application co-existed on the same virtual machine as the database; while it is not required for it to be co-located on the same system, we chose this setup in order to minimize network traffic. Most databases (e.g., Db2 LUW) allow invoking User Defined Functions in languages other then PL/SQL (e.g., C/C++) [7], which facilitate better performance if the fucntionality were developed directly in the database.

We use GProM [8] to determine the input tuples which contributed to each output `GROUP BY`. Using this information, our framework passes the output of the GProM query (which contains both the original requested columns and the additional input tuple columns) to a Python script to run our privacy analysis.

6.2 Experimental Setup

For our experiments, we used a server with dual Intel Xeon E5645, each with 6 physical cores and Hyper Threading enabled, and 64GB

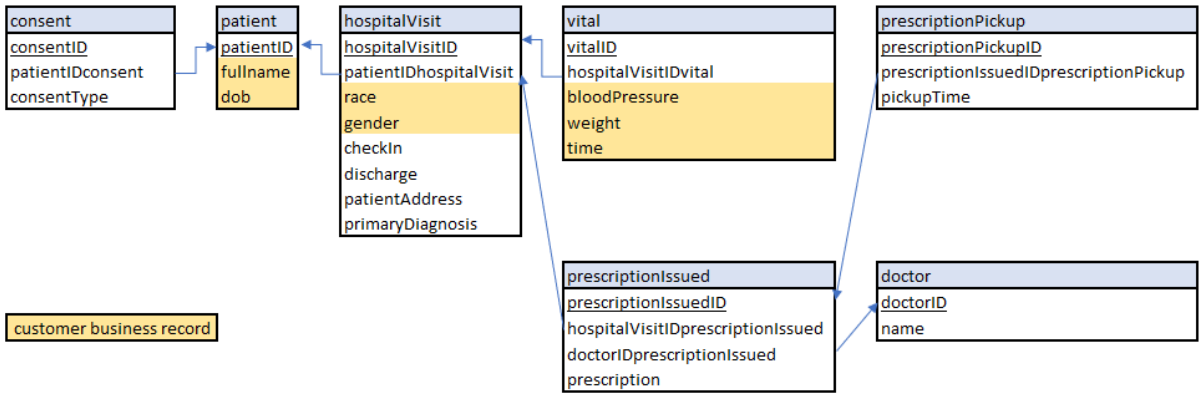

Figure 4: Experimental Schema (based on MIMIC)

of RAM, and a SSD for storage; the server was running CentOS 8 Stream x86_64 with Kernel Virtual Machine [5] (KVM) as the hypervisor software. Within this setup, there was a Virtual Machine (VM) used for our experiments. The VM was built with CentOS 8 Stream x86_64 and Postgres 14.4, 1 x vNIC and a 25GB QEMU Copy On Write [4] (QCOW2) disk image file on an SSD. The QCOW2 file was partitioned into: 350MB/boot, 2GB swap, and the remaining storage was used for the / partition; this was done with standard partitioning and ext4 filesystem. While the experiments were running, only this VM was running on the hypervisor to ensure the least amount of fluctuation in the runtime measurements.

Using the schema in Figure 4, we simulated the performance of a real-world system. This schema design was derived from a subset of the MIMIC schema [19]. Some design elements which may not intuitively make sense (e.g., a patient's race being collected upon every hospital visit) are from the MIMIC database which is based on a real-world medical database. Note that we refer to patients as customers for the purposes of this paper. Additionally, we did not include our full consent-compliant schema, but focused only on the minimum number of tables and fields required to evaluate query performance. For all database sizes used, we assume only 25% of our customers consent to marketing processing. Throughout our experiments, we used synthetically generated data designed to ensure some groups in our query processing are removed while others remain due to having a sufficiently large and balanced samples.

Because this database contains a mix of consenting and non-consenting customers, if an access control mechanism that filtered out any non-consenting customers or differential privacy application were applied, then none of the aggregation results would have been guaranteed accurate.

6.3 Adjusting Table Counts

In this analysis, we run 3 different queries on the database loading with 3 different data set sizes (table sizes are summarized in Table 1). The "realistic" database size was determined based on the size of the original MIMIC tables. Our three queries (with inner joins) were as follows (with post consent filtering):

(1) **2T:** Average weight by race
83% were not filtered
$\texttt{vital} \bowtie \texttt{hospitalVisit}$

(2) **3T:** Average weight by race and prescription
97% were not filtered
$\texttt{vital} \bowtie \texttt{hospitalVisit} \bowtie \texttt{prescriptionIssued}$

(3) **4T:** Average weight by race, prescription, and doctor
99% were not filtered
$\texttt{vital} \bowtie \texttt{hospitalVisit} \bowtie \texttt{prescriptionIssued} \bowtie \texttt{doctor}$

With the large memory requirements of "realistic" data, Postgres was unable to run 2T's GSQL and, therefore, it was not evaluated.

We analyze our framework capturing the runtimes for: 1) the original query (to use as a baseline), 2) the provenance analysis (i.e., running the GProM query), 3) sample size confirmation, and 4) sample balance verification. For all queries, we verified the results met our compliance requirements of sample size and sample balance and were correctly processed to ensure compliant processing.

Provenance Analysis. In order to determine the full provenance of a requested query, we input a SQL query into GProM [8], which rewrites the SQL into a provenance computation (which we refer to as *GSQL*) to include the input tuples alongside the original query's output (i.e., calculating the query provenance). Because this rewrite would only change if the underlying schema or SQL changes, the process of running GProM's rewrite would only need to occur once. In other words, this step is not required during every query execution but instead only during the set-up of a new query. Thus, we do not include the process of GProM query rewrite in our overhead analysis. Instead, we discuss the overhead of running the GSQL compared to the original SQL independently.

GProM was developed as a generalized tool for calculating provenance, which "[does not] require any changes to the backend database system." [8] Thus, per the authors' discussion on GProM's current development, an alternative provenance framework with a more dedicated purpose (and less generalized implementation), would have a lower performance cost. For the purposes of this paper, we used GProM to validate our framework and compute an upper bound of compliance evaluation performance without a dedicated compliance provenance approach. Any provenance calculation mechanism which provides the source data required by our analysis may be implemented as an alternative approach.

The average cost of running the GSQL was an overhead factor of approximately 23.6X compared to the original query runtime. An investigation of provenance framework optimization is beyond the scope of this paper. Although we later detail opportunities for

Database Size	Memory Size (MB)	Total Customers	Business Records	2T Records	3T Records	4T Records
Small	1,011	40,002	241,002	6 (87% returned)	31 (97% returned)	7,995 (99% returned)
Large	1,321	400,002	2,201,002	6 (87% returned)	30 (97% returned)	73,954 (99% returned)
Realistic	3,583	400,002	24,000,002	6 (87% returned)	30 (97% returned)	73,954 (99% returned)

Table 1: Experimental Database Sizes

performance improvement in Section 7.1, the execution time for the provenance analysis provided by GProM makes up the majority of overhead incurred by our framework.

Group Size. We evaluate the computation cost of determining whether a `GROUP BY` in a query met the required size output by Yamane's formula. For each of the queries, this consisted of the following steps:

(1) Calculating the total population size (i.e., number of weight measurements)
(2) Inputting the population size into Yamane's Formula to calculate the sample size required for statistical anonymity
(3) Determine the sample size for each group
(4) Replacing the calculation with `NULL` for groups that did not meet the sample size requirements

The runtimes of the Group Size computation step are shown in Figure 5. The largest cost visualized is associated with the 2T Large combination. This is due to the framework having to calculate larger number of records per `GROUP BY` – the number of input records is the same, while 2T queries has the fewest number of groups. When the number of groups increases in 3T and 4T queries, the framework has more groups to determine whether or not they meet the criteria for the group size threshold. Additionally, with the Realistic database, because fewer groups fall below the required size output by Yamane's Formula, fewer groups have the additional step of removing their aggregation calculation from the output. As the database size increases, the cost of the computation increases as well. This is particularly noticeable in 2T query (with only 6 groups), which causes GSQL computation to run out of memory in 8G VM; therefore, we were unable to run this combination with our current set up of hardware, data, and experimental parameters resulting in not having enough RAM. The average overhead for the Group Size step across our tested combinations took approximately 40% as long as the original SQL runtime.

Group Balance. For the groups that met the minimum sample size threshold, we proceed to evaluate the sample size balance. In our analysis, we assume that the customer whose average is furthest from the rest of their group is the most likely to have their privacy exposed. However, in some instances, a customer who has a smaller divergence but a larger number of observations may have a more statistically significant KS test p-value. Therefore, in practice (and future research), we recommend using (and optimizing) an approach to test all non-consenting customers for group balance. We evaluate each group defined by the `GROUP BY` clause to determine if it meets our defined balance criteria. Figure 6 illustrates the runtimes of the framework's balancing process (for both the influence and record contributions as discussed in Section 5.2).

Influence: For the influence, we compared each group's distribution against the distribution of their most differentiable non-consenting customer (using the two-sample KS test from SciPy). With each group we identified the non-consenting customer whose average weight is furthest from their group's overall mean computed with their weight included (assuming the highest probability of identifiability belongs to the customer most divergent from their group). We used the same precision value for our alpha score (0.05) in order to determine the threshold for statistical significance.

Record Contributions: As with the influence, we determined the non-consenting customer who had the highest number of records (per group). We calculated the adjusted IQR (adjusting the percentiles to be 0.025 and 0.975 instead of the default Q_1 and Q_3) to capture all but 0.05 percent of the data in the adjusted IQR. We then multiplied that range by 1.5 and determined if any non-consenting customer contributed a great number of records than the calculated threshold. Groups who did have a non-consenting

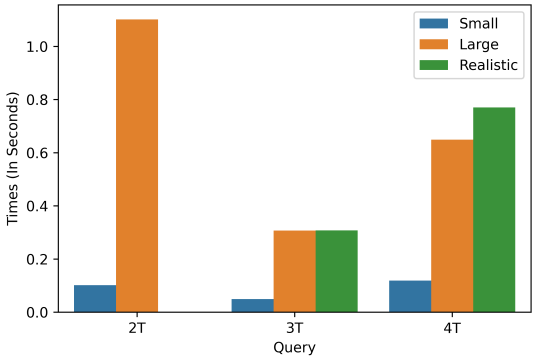

Figure 5: Group Size Runtimes

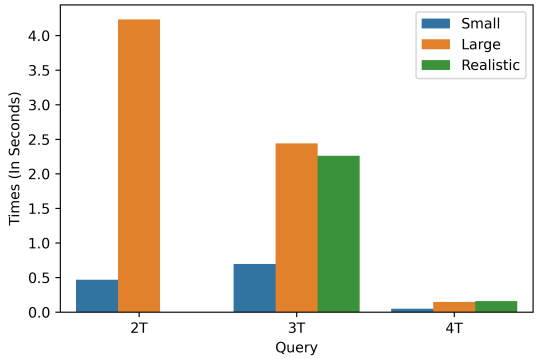

Figure 6: Group Balance Runtimes

customer who contributed more than the calculated threshold did not return an aggregate calculation.

Balance Performance Overhead: The cost of balancing was, on average, 1.8x that of the original SQL runtime. For query 2T with our large database, because there was relatively few groups, many groups met the minimum required Group Size requirement, resulting in a higher number of groups requiring group balance verification. With query 4T, because many of the groups did not meet the group size requirement, less groups required a balance check (thus reducing the performance cost). For query 3T, this cost was above 4x the original SQL due to the data size and the high amount of groups which required group balance analysis (as a result of the large number of columns used in the `GROUP BY`).

6.4 Modified 2T Queries

To further evaluate how our framework works across different queries, we adjust our 2T query (Average weight by race). We ran two different modified queries across our Small and Large databases:

(1) Average weight by race and gender with 18 groups and 94% remain after compliance filtering
(2) Average weight by race filtered on an individual gender resulting in one group which was not filtered

By modifying the original query, we can examine the impact of the framework on queries that have smaller groups (but consistent tables corresponding tuples). Furthermore, the second additional query that filters on an individual gender should return the same results but with only a subset of the data. Again, the realistic data was unable to be run due to Postgres and hardware constraints. For both queries, we verified that any data processing that would result in risking customer privacy has been properly filtered from the outputs. The plots shown in Figure 7 illustrate the runtimes of the two sizes and queries across three steps.

GProM. The first step was running GProM to rewrite the query to return the necessary provenance calculations. With this, we can see that GProM depends more on the schema of the database and not the size of the data. Furthermore, because the queries being run required the same data, the cost was negigible between the two.

GSQL Execution. Next, the database must rerun the modified query to run the query and return the provenance of the source tuples (denoted as *GSQL Execution* in Figure 7). For both queries, when comparing the small to the large, both averaged an approximate increase by a factor of 3.5. This is more due to the complexity of the query combined with the schema then the framework (and as such, would be more of an evaluation of the performance of GProM which is beyond the scope of this paper). When adjusting the data that is considered in-scope, we can see that a limited dataset using the gender filter query does greatly reduce the overall runtime for both small and large databases.

Python Testing and Filtering. The Python element of this framework applies the statistical testing and filters out any data whose inclusion would result in a policy violation. As with the Output Rewrite, we can see that a reduced dataset (even with the same columns) drastically reduces the overall runtime. Therefore, even with a larger database size, if a user were to have more precise query, the runtime can be reduced drastically.

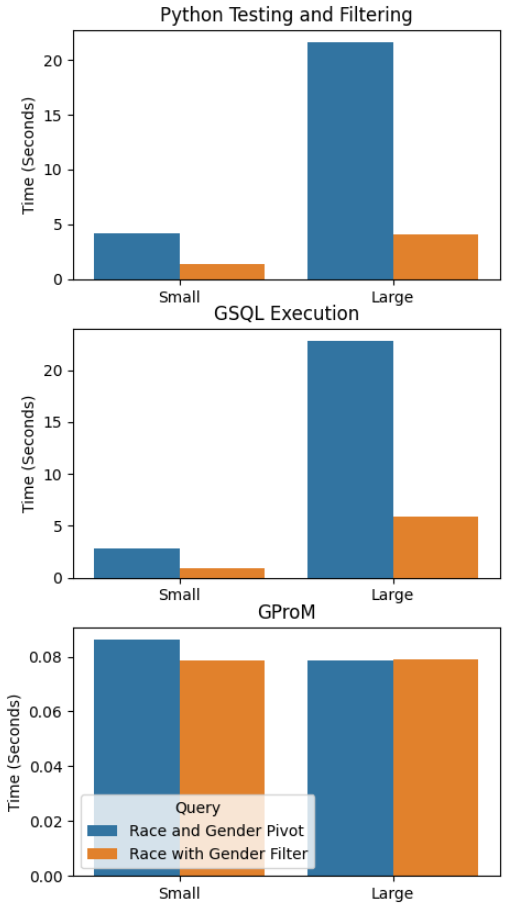

Figure 7: GProM, GSQL Execution, and Python Filtering Runtimes

7 Discussion

7.1 Performance Optimization Opportunities

Due to the lack of `SELECT` triggers in RDBMS platforms, we implement the filtering logic as a proxy using a Python implementation. While Postgres has the ability to implement Python within PL/SQL [6], our framework depends on additional Python packages (e.g., NumPy, Pandas, SciPy). There is a high potential for performance optimizations with a dedicated compliance functionality.

Currently, the framework overhead performance uses Python 3.6.8 with Pandas 1.1.5 (using NumPy as the underlying memory store). In full scale implementation, we would either implement this framework directly using NumPy or Arrow to reduce overhead. During the execution of the KS test, we used Python's SciPy package. This package calculates additional metrics during the test, which are not necessary for our framework. Reducing the function to the minimally required calculations would reduce overhead cost.

Depending on the business needs of the organization, one can run our logic against a static set of queries and store the results as a materialized view; by doing so, the performance of the external application could be minimized to an infrequent activity. Additionally, we did not alter GProM's original output GSQL. Thus, the output GSQL includes full provenance functionality which results

in additional columns being calculated that are not used during our frameworks consent verification process. These additional columns come with a potentially significant processing cost (an evaluation of the full performance metrics of GProM is beyond the scope of this paper). Although we do later remove these additional columns during our Python processing, the cost of calculating these fields is still incurred. Thus, designing and developing a provenance functionality to calculate input tuples (but with the minimum columns required) would offer significant performance benefits.

7.2 Statistical Considerations

In this paper, we leverage Yamane's Formula, the KS test, and the IQR Rule for detection of sample size and sample balance. In practice, many statistical techniques and tests have assumptions that must be verified before implementation. For example, a common t-test requires normally distributed data. Thus, when incorporating advanced statistical techniques into this framework, additional verification steps may be required (unless using non-parametric statistical tests). Overall, organizations must have their DBAs, legal experts, and statisticians determine what the appropriate needs are for their organization in order to ensure compliant processing.

Although we use a KS test for our implementation, because a KS test evaluates the entire distribution, it may detect differences that are not applicable to a particular aggregate function. For example, a KS is given two distributions with equal averages but extremely different variances may yield a statistically significant result. Thus, one may want to limit the comparison of influence and distinguishability to a narrower scope on individual aggregation functions instead of a comprehensive comparison of distributions.

8 Future Work and Conclusion

Although in this paper we use a KS test and the IQR rule for balance verification baseline, a further analysis of other techniques would provide stronger guidance for which techniques organizations should implement. GProM was never intended to be used for this purpose and performs more computation than strictly necessary. Other tools may satisfy the processing requirements with a lower overhead cost (requiring fewer calculations for the statistical analysis) by retrieving the minimum amount of data from the input tuples required to facilitate the statistical tests. Furthermore, developing the same functionality within the database application would reduce the amount of third-party framework dependencies.

Automated solutions must be developed to guarantee compliance without requiring a system overhaul or rewriting all queries. In this paper, we outlined a framework which determines when customer anonymity has been preserved in aggregate computations. This framework automatically adjusts the query outputs to achieve compliance while returning a maximally correct query (by removing the rows with non-compliant results). Our paper has shown how this framework can be incorporated into an existing database, automating the processing with a non-intrusive changes.

Acknowledgments

This work was partially funded by US National Science Foundation Grant IIP-2016548, CME Group, Argonne National Laboratory, and Louisiana Board of Regents Grant AWD-10000153. Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357.

References

[1] 2020. California Consumer Privacy Act. oag.ca.gov/privacy/ccpa
[2] 2020. GDPR Archives. gdpr.eu/tag/gdpr/
[3] 2021. €27,8 million GDPR fine for Italian Telecom -TIM. dataprivacymanager.net/e278-million-gdpr-fine-for-italian-telecom-tim/
[4] 2022. Qcow. en.wikipedia.org/wiki/Qcow
[5] 2023. www.linux-kvm.org/page/Main_Page
[6] 2023. 46.1. PL/Python Functions. www.postgresql.org/docs/current/plpython-funcs.html
[7] 2023. Create function (external scalar) statement. www.ibm.com/docs/en/db2/11.5?topic=statements-create-function-external-scalar
[8] Bahareh Sadat Arab, Su Feng, Boris Glavic, Seokki Lee, Xing Niu, and Qitian Zeng. 2018. GProM-a swiss army knife for your provenance needs. *A Quarterly bulletin of the Computer Society of the IEEE Technical Committee on Data Engineering* 41, 1 (2018).
[9] Ahmed A Ataullah, Ashraf Aboulnaga, and Frank Wm Tompa. 2008. Records retention in relational database systems. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 873–882.
[10] Mariem Brahem, Guillaume Scerri, Nicolas Anciaux, and Valerie Issarny. 2021. Consent-driven data use in crowdsensing platforms: When data reuse meets privacy-preservation. In *2021 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 1–10.
[11] Kamalika Chaudhuri and Nina Mishra. 2006. When random sampling preserves privacy. In *Advances in Cryptology-CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006. Proceedings 26*. Springer, 198–213.
[12] James Cheney, Laura Chiticariu, and Wang-Chiew Tan. 2009. *Provenance in databases: Why, how, and where*. Now Publishers Inc.
[13] William G Cochran. 1977. *Sampling techniques*. John Wiley & Sons.
[14] Jacob Cohen. 1992. Statistical power analysis. *Current directions in psychological science* 1, 3 (1992), 98–101.
[15] Giovanni Di Leo and Francesco Sardanelli. 2020. Statistical significance: p value, 0.05 threshold, and applications to radiomics—reasons for a conservative approach. *European radiology experimental* 4, 1 (2020), 1–8.
[16] Osnat Drien, Antoine Amarilli, and Yael Amsterdamer. 2021. Managing Consent for Data Access in Shared Databases. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 1949–1954.
[17] Centers for Medicare & Medicaid Services et al. 1996. The health insurance portability and accountability act of 1996 (hipaa). *Online at http://www.cms.hhs.gov/hipaa* (1996), 158.
[18] Julian Holzel. 2019. Differential Privacy and the GDPR. *Eur. Data Prot. L. Rev.* 5 (2019), 184.
[19] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3, 1 (2016), 1–9.
[20] George Konstantinidis, Jet Holt, and Adriane Chapman. 2021. Enabling personal consent in databases. *Proceedings of the VLDB Endowment* 15, 2 (2021), 375–387.
[21] Frank J Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 46, 253 (1951), 68–78.
[22] Qun Ni, Shouhuai Xu, Elisa Bertino, Ravi Sandhu, and Weili Han. 2009. An access control language for a general provenance model. In *Workshop on Secure Data Management*. Springer, 68–88.
[23] Primal Pappachan, Roberto Yus, Sharad Mehrotra, and Johann-Christoph Freytag. 2020. Sieve: A middleware approach to scalable access control for database management systems. *arXiv preprint arXiv:2004.07498* (2020).
[24] Nick Scope, Alexander Rasin, James Wagner, Ben Lenard, and Karen Heart. 2021. Purging Data from Backups by Encryption. In *International Conference on Database and Expert Systems Applications*. Springer, 245–258.
[25] Dusan Stevanovic, Natalija Vlajic, and Aijun An. 2013. Detection of malicious and non-malicious website visitors using unsupervised neural network learning. *Applied Soft Computing* 13, 1 (2013), 698–708.
[26] Carolyn Whitnall, Elisabeth Oswald, and Luke Mather. 2011. An exploration of the kolmogorov-smirnov test as a competitor to mutual information analysis. In *Smart Card Research and Advanced Applications: 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers 10*. Springer, 234–251.
[27] Kim Wuyts, Riccardo Scandariato, Griet Verhenneman, and Wouter Joosen. 2011. Integrating patient consent in e-health access control. *International Journal of Secure Software Engineering (IJSSE)* 2, 2 (2011), 1–24.
[28] Taro Yamane. 1967. An introductory analysis of Statistics.