Disclosure-Compliant Query Answering

RUDI POEPSEL-LEMAITRE, BIFOLD & Technische Universität Berlin, Germany
KAUSTUBH BEEDKAR, Indian Institute of Technology Delhi, India
VOLKER MARKL, BIFOLD, Technische Universität Berlin & DFKI, Germany

In today's data-driven world, organizations face increasing pressure to comply with data disclosure policies, which require data masking measures and robust access control mechanisms. This paper presents Mascara, a middleware for specifying and enforcing data disclosure policies. Mascara extends traditional access control mechanisms with data masking to support partial disclosure of sensitive data. We introduce data masks to specify disclosure policies flexibly and intuitively and propose a query modification approach to rewrite user queries into disclosure-compliant ones. We present a utility estimation framework to estimate the information loss of masked data based on relative entropy, which Mascara leverages to select the disclosure-compliant query that minimizes information loss. Our experimental evaluation shows that Mascara effectively chooses the best disclosure-compliant query with a success rate exceeding 90%, ensuring users get data with the lowest possible information loss. Additionally, Mascara's overhead compared to normal execution without data protection is negligible, staying lower than 300ms even for extreme scenarios with hundreds of possible disclosure-compliant queries.

CCS Concepts: • **Security and privacy → Data anonymization and sanitization**; • **Information systems → Middleware for databases**.

Additional Key Words and Phrases: Data Masking, Privacy, Access Control, Information Loss

ACM Reference Format:
Rudi Poepsel-Lemaitre, Kaustubh Beedkar, and Volker Markl. 2024. Disclosure-Compliant Query Answering. *Proc. ACM Manag. Data* 2, 6 (SIGMOD), Article 233 (December 2024), 28 pages. https://doi.org/10.1145/3698808

1 INTRODUCTION

Over the past decade, there has been an unprecedented surge in the volume of data being collected, stored, and processed across various sectors and industries [58]. The ability to harness data has presented numerous societal and economic benefits in the form of data-driven applications and decision-making across domains, including health [50], transportation [64], energy [39], or manufacturing [14]. However, this also has raised concerns about data privacy and information misuse [30, 53, 59]. Large parts of the data that is being collected and processed can be sensitive personal, medical, or governmental data, and, as a result, has led to data regulations—most prominent being European Union's General Data Protection Regulation (GDPR) [2] and the California Consumer Privacy Act (CCPA) [1]—that regulate the handling of data.

Data privacy concerns and potential information misuse underscore the critical need for robust access control mechanisms in data handling practices. In particular, access control mechanisms to limit use and restrict disclosure of sensitive data are crucial in safeguarding data against unauthorized access and misuse and complying with data regulations. Additionally, disclosing data after

Authors' addresses: Rudi Poepsel-Lemaitre, BIFOLD & Technische Universität Berlin, Berlin, Germany, r.poepsellemaitre@tu-berlin.de; Kaustubh Beedkar, Indian Institute of Technology Delhi, Delhi, India, kbeedkar@cse.iitd.ac.in; Volker Markl, BIFOLD, Technische Universität Berlin & DFKI, Berlin, Germany, volker.markl@tu-berlin.de.

ACM 2836-6573/2024/12-ART233
https://doi.org/10.1145/3698808

anonymizing sensitive or personally identifiable information attributes using masking functions provides data protection while preserving the data's utility for analysis or research purposes.

Current access control mechanisms [12, 54, 55] to limit data disclosure, however, do not support data masking functions, thereby impeding an organization's data-driven decision-making and collaborations. Therefore, extending access control mechanisms with data masking capabilities is essential. Recently, many commercial database systems, such as Microsoft Azure [41], Oracle [45], and open source systems such as Postgres [36] have included support for data masking functions as a helpful feature to protect sensitive data. However, they adopt a query-agnostic approach, which limits their effectiveness by trading off data utility with protection.

Extending access control mechanisms with query-aware data masking capabilities entails three major challenges. First, we need an easy and flexible way to specify disclosure policies subjected to data masking requirements. Often, masking attributes using different masking functions depending upon which combination of attributes are being accessed offers better utility of anonymized data without compromising disclosure requirements. For example, consider disclosing three personal attributes: zip code, gender, and date of birth. Using current access control mechanisms, one would either entirely restrict access to the three attributes [6, 27, 42] or use query-agnostic data masking to anonymize the data regardless of the query issued by the user [20, 34, 40, 52]. Both approaches limit the utility of data. Instead, it may be desirable to mask data differently based on which attributes a query is accessing and the query semantics. For instance, it may be desirable to suppress the date of birth and blur the zip code when computing aggregate gender information across zip codes, and conversely suppress the gender information and only disclose the year of birth when filtering data on date of birth.

The second challenge pertains to extending query rewriting techniques used by access control mechanisms to consider both query semantics and properties of masking functions. Existing mechanisms can only support disclosure policies that require total restrictions by either filtering out sensitive data from query answers [6, 12, 21, 56] or performing some validation/rewriting on the submitted query to only release compliant results [5, 27, 42, 54, 57]. Supporting partial restrictions with dynamic data masking requires carefully interleaving existing query operators with masking functions. For instance, masking the zip code after evaluating the filter may lead to possible singling out records, or applying data masking functions before other query operations may break the query semantics by changing the data type of attributes (e.g., blurring the zip code may change its type from an integer to a string).

The third challenge is minimizing information loss when attributes can be masked differently. For example, when masking via noise addition, a rewritten aggregate query will carry a higher utility than masking by blurring. To this end, we need a way to estimate the information quality of the rewritten query, particularly when attributes can be masked using different masking functions depending upon which attributes are accessed.

In this paper, we present Mascara, a middleware that allows data officers and database administrators to specify and enforce data disclosure policies that require data masking. We propose *data masks* as a simple and intuitive way to specify which information attributes and how those attributes should be masked prior to disclosure. We also show how to modify a user-specified query into a disclosure-compliant query that adheres to disclosure policies. As a result, one can easily incorporate our solution into existing access control mechanisms provided by database systems. Mascara allows queries to be written in a disclosure-transparent manner, i.e., user queries can be written against the database without referring to disclosure policies. Further, Mascara also allows context-based data masking, i.e., it allows specifying masking of attributes depending on which combination of attributes are being accessed. To this end, we propose a utility estimator, which

estimates the similarity between the user and modified queries. Our utility estimator enables the modification of a user query into a disclosure-compliant one with the best information quality.

In sum, after further motivating the need for supporting query-aware data masking and limitations of current access control mechanisms (Section 2), we make the following major contributions:

- We formalize the concept of disclosure policies and formally define the problem of disclosure-compliant query answering. (Section 3)
- We introduce data masks to specify which information attributes and how those attributes should be masked concisely and intuitively. Our data masks offer the flexibility of masking attributes differently depending on which other attributes are accessed by the query. (Section 4)
- We show how to modify a user query into a disclosure-compliant query by considering both the query semantics and certain properties of the masking functions specified in the policies. This allows the underlying database to adhere to disclosure policies when answering user queries. (Section 5)
- We propose relative entropy as a distance metric and present a utility estimator that allows modifying a user query into a disclosure compliant query with the least information loss. (Section 6)
- We experimentally evaluate the effectiveness and efficiency of Mascara using data based on the TPC-H benchmark [61] and the Folk US-census dataset [18]. Overall, our experiments show that Mascara's utility metric can successfully capture the decaying utility of masked data, even for downstream applications. Furthermore, Mascara can choose one of the best compliant queries using our approximation methods with acceptable overhead.

2 BACKGROUND AND MOTIVATION

We begin with a motivating example illustrating disclosure policies that require data masking. We then discuss the shortcomings of current access control mechanisms in supporting such policies.

2.1 Motivating Example

Consider `MedIns`, a medical institute that maintains a database D to store information about its patients. The database has a `patient` and `diagnosis` table with the following information attributes

```
Patient (p_id, name, age, sex, address, zipcode, expenses)
Diagnosis (d_id, weight, height, disease, p_id, dr_id)
```

Figures 1a and 1b show an excerpt of the patients' data. Here the attribute `disease` contains values following the ICD-10-GM classification [23], e.g., C25.0 stands for "Malignant neoplasm of the head of pancreas". Based on the recommendation of `MedIns`'s data officer (DO), the database administrator (DBA) would like to enforce the following data disclosure policies for admins, doctors, and analysts.

P_1: Attributes comprising only `p_id`, `name`, `address`, `zipcode` and `expenses` can be disclosed with administrative personnel.

P_2: A patient's medical information such as `name`, `age`, `sex`, `weight`, `height`, and `disease` can only be disclosed with their doctor.

P_3: Attributes `age`, `sex`, `zipcode`, and `disease` can be disclosed to an analyst only after masking the `age` and `zipcode` by "bucketizing" the age values in ranges of size 5 (e.g., $23 \rightarrow [20, 25)$), and by "blurring" the last three digits of the zipcode code values (e.g., $13347 \rightarrow 13xxx$).

P_4: Personal information like `age`, `height`, and `weight` along with `disease` and `expenses` can also be disclosed to an analyst, but only after masking attributes `disease` and `expenses` by "generalizing" the ICD-10-GM classification (e.g., C25.0: Malignant neoplasm of the head of pancreas $\rightarrow$ C25:Malignant neoplasm of pancreas) and by adding random "noise" to the patients' expenses.

p_id	name	age	sex	address	zipcode	expenses
1	Ana	65	F	Berliner Str. 2	13159	38,035.00
2	Klaus	54	M	Reichen Str. 145	10999	37,117.00
3	Carla	73	F	Plauener Str. 43	13055	15.335,00
4	Lucy	52	F	Ritterstraße 98	10969	21.887,00
5	Andres	61	M	Gounodstraße 12	13088	18.726,00
6	Nele	69	F	Friedrichstraße 129	10117	24.975,00

(a) Patients table

d_id	weight	height	disease	p_id	dr_id
1	75	173	C25.0	1	1
2	60	173	A01.1	1	2
3	91	183	C25.0	2	2
4	54	163	C18.4	3	1
5	62	178	C18.7	4	1
6	80	179	D55.2	5	3
7	65	171	C25.1	6	3
8	87	177	C18.5	5	3

(b) Diagnosis table

p_id	name	address	zipcode	expenses
1	Ana	Berliner Str. 2	13159	38,035.00
2	Klaus	Reichen Str. 145	10999	37,117.00
3	Carla	Plauener Str. 43	13055	15.335,00
4	Lucy	Ritterstraße 98	10969	21.887,00
5	Andres	Gounodstraße 12	13088	18.726,00
6	Nele	Friedrichstraße 129	10117	24.975,00

(c) Data disclosure as per policy P_1.

name	age	sex	weight	height	disease	dr_id
Ana	65	F	60	173	C25.0	1
Carla	73	F	54	163	C18.4	1
Lucy	52	F	62	178	C18.7	1

(d) Data disclosure as per policy P_2.

age'	sex	zipcode'	disease
[65, 70)	F	13XXX	C25.0
[65, 70)	F	13XXX	A01.1
[50, 55)	M	10XXX	C25.0
[70, 75)	F	13XXX	C18.4
[50, 55)	F	10XXX	C18.7
[60, 65)	M	13XXX	D55.2
[65, 70)	F	10XXX	C25.1
[65, 70)	M	13XXX	C18.5

(e) Disclosre as per P_3

age	weight	height	disease'	expenses'
65	75	173	C25	36,133.25
65	60	173	A01	38,642.87
54	91	183	C25	38,972.85
73	54	163	C18	15,641.70
52	62	178	C18	21,011.52
61	80	179	D55	17,763.71
69	65	171	C25	24,725.25
61	87	177	C18	19,475.04

(f) Disclosure as per P_4

Fig. 1. Data disclosure based on different disclosure policies: (a) and (b) original data; (c) and (d) disclosing certain patients' information (`dr_id=1`); (e) masking age and zipcode; and (f) masking diagnosis and expenses.

2.2 Current Limitations & Challenges

State-of-the-art access control in DBMSes do not lend themselves to support flexible data masking requirements (such as in example policies P_3 and P_4 above).

Status quo: Current access control mechanisms include view-based access control [12, 16, 55, 56] or query-based access control [54, 57]. The former employs database views—to specify a "version" of data that can be disclosed—and query modification [16] to match a policy. The latter on the other hand, either accepts or rejects a query as deemed by the policy. Furthermore, prior art on view-based control only support relational operators in view definitions, and therefore, can only support disclosure policies of type P_1 and P_2. For example, to enforce policy P_1, the database administrator can create an *authorization view* [16] that only projects the permissible information attributes. We note here that authorization views can also be defined using other relational operators including aggregates and joins. Likewise, policy P_2 can also be enforced using parameterized or access pattern views [54], which can have parameters that can bound to some values (for example `dr_id = $user_id`).

Challenges: View-based access control mechanisms, however, fall short of supporting policies such as P_3 and P_4. Although, in principle, the DBA can create authorization views using user-defined functions (UDFs) to mask desired attributes, it faces the following two challenges in the evolving landscape of data regulations.

1. Query modification. Disclosure policies based on data protection regulations often prescribe "masking functions" [1, 2] to regulate access to sensitive information. This makes current query

rewriting techniques unsuitable as masking functions may potentially change both the data type and value domain of an attribute. For example, consider the following query Q_1 issued by an analyst.

select disease, **avg** (age) **from** diagnosis **group by** disease

Further, consider the table ("view" of the patient's data) shown in Figure 1f, that can be accessed while complying to policy P_4. Rewriting the above query using such a view is straightforward in that the age attribute is not masked. However, this is not always the case. For instance, consider the absence of policy P_4 and the view shown in Figure 1e that can be accessed while complying to policy P_3. In this case, rewriting the above query is far from trivial as the attribute age is masked by bucketizing values.

2. Context-based anonymization. The second challenge pertaining to supporting disclosure policies in that often information of quasi-identifiers (e.g., attributes such as date-of-birth, zip-code code, or sex) is considered sensitive when combining them with other quasi-identifiers [3] (Recital 26). This offers data officers the flexibility of anonymizing quasi-identifiers based on the combination of attributes that are being disclosed, i.e., by context. For example, in the case of MedIns, it allows to have policies P_3 and P_4, and views in Figure 1e and 1f, respectively when disclosing data to analysts. However, choosing the right view(s) to answer a query can quickly become complex. Consider query Q_1 again. In addition to how to rewrite a query, which view to answer from also becomes a challenge. For instance, is it better to have the precise disease and an approximate age aggregate, or is it better to have more general diagnoses with exact aggregates.

Overall, current access control mechanisms do not support disclosure policies that require employing masking functions and masking attributes differently based on which combination of attributes are being accessed.

3 DISCLOSURE-COMPLIANT QUERY ANSWERING

We propose Mascara, which enables data officers and DBAs to specify and enforce data disclosure policies. Mascara is designed as an access control middleware between the user (querier) and the database system. Mascara intercepts the user query and translates it to make data access compliant with disclosure policies. In the following, we give an overview of Mascara and formalize the problem of disclosure-compliant query answering that we consider in this paper.

3.1 Mascara Overview

Figure 2 illustrates the schematics of Mascara. It allows declarative specification of disclosure policies as a set of data masks. A *data mask* specifies which information attributes can be disclosed to whom (i.e., the user) and how (i.e., using which masking function). For instance, a data officer can specify the disclosure policies in the example above, where disclosure of information must be subjected to masking attributes using certain masking functions (e.g., blurring or bucketization). Consequently, for each masking function specified in the data mask, Mascara additionally requires defining a corresponding User-Defined Masking Function (UDF) (e.g., by using the CREATE FUNCTION statement). Note that specifying disclosure policies and defining masking functions is an offline process. While the set of data masks is stored in Mascara, the data masking UDFs must be added to the DBMS's catalog.

At query time, Mascara validates a query by first checking if the query is accessing any of the "restricted" information attributes (such as example query Q_1 by an analyst above). It then modifies the queries by parsing the query into its logical plan and then modifies each plan operator so that the query respects the disclosure policy. If the query modifier fails to rewrite a query, it rejects it. Additionally, for scenarios where more than one policy is applicable, for instance, disclosure policies

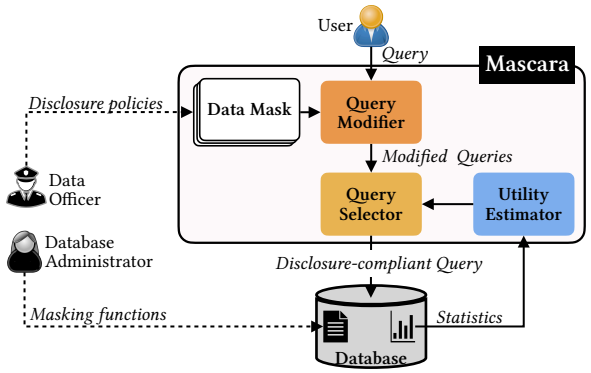


Fig. 2. Mascara overview

P_3 and P_4 for the analyst in the example above, the query modifier generates a set of disclosure-compliant queries. The query selector selects the "best" query from the candidate set and sends the disclosure-compliant query to the database. For this, it uses the utility estimator, which estimates the information quality based on available database statistics and the masking function's properties. The query selector leverages these estimates to choose a modified query (disclosure-compliant) that respects the disclosure policy and is also "optimal" regarding information loss over the original query (nondisclosure-compliant).

3.2 Disclosure-compliant Queries

We consider a setting where a set $\mathcal{U}$ of user groups (e.g., analysts or doctors) access data is stored in a database $D = \{ R_1, \ldots, R_{|D|} \}$ and each relation $R(a_1, \ldots, a_{\deg(R)})$, where $\deg(R)$ denotes the degree of a relation R may be subjected to one or more disclosure policy.

Disclosure Policies. Given a set $\mathscr{P}$ of disclosure policies, a policy $P \in \mathscr{P}$ specifies which information attributes and how those attributes should be *masked* before they can be disclosed to a user group $U \subseteq \mathcal{U}$. Abstractly, we model P as a 4-tuple $\langle \mathcal{R}, \rho, \phi, U \rangle$, where: $\mathcal{R}$ specifies a set of relations; ρ specifies the conditions that select tuples in R that can be disclosed; $\phi = \{(a, \mu) : a$ is an attribute of R, μ is some masking function$\}$ specifies the projection attributes and how they should be masked; and $U \subseteq \mathcal{U}$ specifies the set of user group for which the policy applies.

Masking Functions. Our goal in this paper is to support disclosure policies that require masking data. A data masking function $\mu : \mathbf{dom}(a) \rightarrow \mathbf{dom}(a')$ takes as input an attribute value (along with other parameters) and outputs its masked value. In this work, we only focus on scalar and value-based functions. For example, a masking function $\texttt{blur}(z, n)$ to blur zip codes takes as input a zipcode z, and a number of characters n and outputs a masked zipcode, e.g., $\texttt{blur}(13347, 3) = 13xxx$.

Anonymized Relations. Intuitively, each policy $P = \langle \mathcal{R}, \rho, \phi, U \rangle$ states that a relation R can only be disclosed as an *anonymized relation* $R_P \equiv \pi_{\mu_1(a_1),\ldots,\mu_n(a_n)}(\sigma_\rho(\mathcal{R}))$ to the user group U. Further, let $\mathscr{P}_U$ denote the set of policies applicable to users U. It follows that the database $D_{\mathscr{P}_U} = \left\{ R_{P_1}, \ldots, R_{P_{|\mathscr{P}_U|}} \right\}$ is also anonymized. Note that $|D| \leq |D_{\mathscr{P}_U}|$ as multiple policies may mask the same relation differently (e.g., policies P_3 and P_4 in the example from Section 2.1). Henceforth, for expository reasons, we assume that there is only one user group; this, however, does not scarify the problem formulation below.

Disclosure-compliant Queries. For a given database D subjected to disclosure policies $\mathscr{P}$ and a `SELECT-FROM-WHERE-GROUP BY` (SPJG) query Q, denote by $Q(D)$ the set of tuples as the query answer. We say that $Q(D)$ is disclosure compliant if and only if there exists a query Q' such that $Q(D) = Q'(D_{\mathscr{P}})$.

Our goal in this work is to modify, i.e., to rewrite, a given user query Q to a disclosure-compliant query Q^c. Since multiple policies may anonymize the same relation differently, multiple ways to modify a query can exist. To this end, we additionally define $\texttt{dist}_D(Q, Q^c)$[1] as the "distance" between the query answer in terms of information loss. In other words, the distance measures the similarity between the intended answer ($Q(D)$) and the disclosure compliant answer ($Q^c(D)$)..

We are now ready to define the disclosure-compliant query answering problem that we focus on in this paper.

Problem Statement. *Given a database D subjected to disclosure policies $\mathscr{P}$, a user-specified query Q, a space of disclosure compliant queries $\mathcal{Q}$, and a distance function $\texttt{dist}_D(Q, Q^c)$ that assigns a numeric information loss to a compliant query $Q^c \in \mathcal{Q}$, find the disclosure compliant query with the least information loss.*

Discussion. The problem requires considering all possible combinations of anonymized relations, whose size is equal to the cartesian product of the number of available policies for every relation, and checking its correctness. Denote by $Q_{\mathcal{R}}$ the set of relations required to answer Q, by $\mathscr{P}_R$ the set of policies for every relation $R \in Q_{\mathcal{R}}$, and by $\mathscr{P}_Q \subseteq \mathscr{P}$ the set of policies applicable to Q (we will formally define them below). Then, the number of policy combinations that Mascara must consider is $O(\prod_{R \in Q_{\mathcal{R}}} |\mathscr{P}_R|)$, i.e., the Cartesian product of the number of policies for every relation. Also, $|\mathscr{P}_Q| = \sum_{R \in Q_R} |\mathscr{P}_R|$. It also follows, under the uniform distribution assumption, that in a worst-case scenario the complexity is $O(|\overline{\mathscr{P}_Q}|^{|Q_{\mathcal{R}}|})$, where $|\overline{\mathscr{P}_Q}| = \frac{|\mathscr{P}_Q|}{|Q_{\mathcal{R}}|}$.

The following theorem captures the hardness of the problem.

Theorem 1. *The disclosure-compliant query answering problem is NP-Hard.*

Proof Sketch. We can prove the hardness of the problem by reducing it to the problem of answering queries using views (AQUV), which is known to be NP-Hard [26, 27]. The AQUV problem involves rewriting a query Q into Q' using a set of views $\mathcal{V}$. In our setting, consider that for every policy $P \in \mathscr{P}$, we use identity functions as masking functions, i.e., $\mu(a) = a$, then the problem of finding a disclosure-compliant Q^c becomes equivalent to finding a rewriting Q' of Q using the views $\mathcal{V}$, where each view $V \in \mathcal{V}$ is an anonymized relation (with the identity function as the masking function). This is because we must explore combinations of all relevant views and check for the equivalence of Q in Q', which is analogous to setting the distance $\texttt{dist}_D(Q, Q') = 0$. □

It follows from above that the problem is also undecidable, as query determinacy for AQUV is known to be undecidable [24, 25]. An important direction for future work includes identifying classes of disclosure policies where the problem is decidable, similar to the related works on AQUV [4, 43, 49].

4 SPECIFYING DISCLOSURE POLICIES

In this section, we propose *data mask*, which are DDL-like statements that allow data officers and DBAs to conveniently specify disclosure policies $\mathscr{P}$. We define the syntax of a data mask for a policy $P := \langle \mathcal{R}, \rho, \phi, U \rangle$ as follows:

```
disclose attribute list from table [, ...]
[with mask on attribute using masking_function] [,..]
[where condition]
```

Here, the mask specifies the attributes the database can disclose and how, i.e., using which masking function those should mask. This maps in our formalism to the set ϕ. With the `from` command, the data officers can define the set of relations ($\mathcal{R}$) for which the data mask applies.Additionally, the

[1]We drop D whenever it is clear from the context.

Listing 1. Data masks for example policies of Section 2.1

P_1	**disclose** p_id, name, address, zipcode, expenses **from** patient **where** $user.role='admin'
P_2	**disclose** name,age,sex,weight,height,disease,dr_id **from** patient, diagnosis **where** patient.p_id=diagnosis.p_id **and** $user.id=dr_id
P_3	**disclose** age,sex,zipcode,disease **from** patient,diagnosis **with mask on** age **using** bucketize(5) **with mask on** zipcode **using** blur(3) **where** patient.p_id=diagnosis.p_id **and** $user.role='analyst'
P_4	**disclose** age, weight, height, disease, expenses **from** patient, diagnosis **with mask on** disease **using** generalize_disease() **with mask on** expenses **using** perturbation(2) **where** patient.p_id=diagnosis.p_id **and** $user.role='analyst'

`where` condition specifies the rows of the table on which the mask should apply (ρ). It also allows limiting disclosure to certain users (U). For this, Mascara relies on query metadata (e.g., user.id or user.role), which can be specified in the `where` clause. Note that attributes in the `disclose` clause without a corresponding masking function can be disclosed as-is.

Examples Consider the disclosure policies from Section 2.1. Listing 1 shows how we specify the policies in Mascara.

For each masking function specified in the data mask, Mascara requires the DBA to define a corresponding User-Defined Masking Function (UDF) (e.g., by using the `CREATE FUNCTION` statement). For example, using PostgreSql DB, for P_2, it requires the DBA to define two UDFs: `bucketize(integer, integer) returns int4range` and `blur(text, integer) returns text`.

Disclosure Model Note that we adopt a conservative approach when specifying policies, i.e., we assume that unless a data mask is specified, no attributes are allowed to be disclosed. For this reason, the `with mask on` clause is optional in the data mask specification. Negative instances may be more convenient for certain policies, i.e., specifying what is not allowed. An additional pre-processing step can handle this.

Data masks provide flexibility in specifying disclosure policies that require data masking. Additionally, it allows for context aware anonymization, i.e., one can specify the masking of attributes differently based on the combination of attributes and to whom the data is closed. For example, for P_3, and P_4, we can specify masking the `age` attribute when accessed by analysts depending upon which other attributes are being accessed.

Discussion. Data masks are reminiscent of authorization views [16, 54]. The key difference lies in supporting data masking functions and query modifications that consider the semantics of masking functions. We also assume that the policies are well-defined and do not include contradictions between each other. More formally, we say that a policy $P_1 := \langle \mathcal{R}_1, \rho_1, \phi_1, U_1 \rangle$ contradicts policy $P_2 := \langle \mathcal{R}_2, \rho_2, \phi_2, U_2 \rangle$ if $\mathcal{R}_1 = \mathcal{R}_2$, $U_1 \subseteq U_2$, $\sigma_{\rho_1}(\mathcal{R}_1) \cap \sigma_{\rho_2}(\mathcal{R}_2) \neq \emptyset$, and $\forall (a, \mu) \in \phi_1, \exists (a', \mu') \in \phi_2$ such that whenever $a = a'$ and $\mu \neq \mu'$ then either of μ or μ' should not be an identity function.

5 QUERY MODIFICATION

We now discuss how Mascara's query modifier, given a set of data masks $\mathscr{P}$, rewrites a given user SPJG query Q into one or more disclosure-compliant queries Q^c. Our approach assumes that selection predicates have been converted into conjunctive normal form (CNF). We also assume that join elimination and subquery de-correlation have been performed.

Notations. Before delving into our query modification approach, we first introduce some necessary notations. For a SPJG query Q, denote by: $Q_{\mathcal{R}}$ the set of relations that appear in the `FROM` clause; A_Q the output attributes; $F_Q = \{ p_1, p_2, \ldots, p_k \}$ the set of filter predicates; G_Q the set of attributes in the `GROUP BY` clause; and f_a the aggregate function associated with the attribute $a \in A_Q \setminus G_q$. Likewise, for a disclosure policy P, we denote by: $P_{\mathcal{R}}$ the set of relations that appear in its `FROM` clause; A_P the set of disclosure attributes; $A_m \subseteq A_P$ the set of masked attributes; and μ_a the masking function associated with the attribute $a \in A_m$.

Query Rewrite Check. Mascara first checks if the user query can be modified to meet the disclosure policies. For this, a data mask must be specified for all relations appearing in the query. Recall that we adopt a conservative approach. More formally, denote by $\mathscr{P}_Q = \{\mathcal{P}_Q \mid \mathcal{P}_Q \subseteq \mathscr{P}$ and $\cup_{P \in \mathcal{P}_Q} P_{\mathcal{R}} = Q_{\mathcal{R}}$ and $\forall P', P'' \in \mathcal{P}_Q, P' \neq P'' : P'_{\mathcal{R}} \cap P''_{\mathcal{R}} = \emptyset\}$

the subsets of policies applicable to a query Q. Mascara rejects the query if the above set is an empty set, otherwise for each $\mathcal{P}_Q \in \mathscr{P}_Q$ it incrementally builds a relational expression by adding "modified" joins, filters, projections, and group-by/aggregations.

Note that the query rewrite does not allow combining policies over the same relation. This allows data officers to specify which attributes can be accessed together. However, there is one case where policies from the same relation can be combined. This is when given two policies $P_1\langle \mathcal{R}_1, \rho_1, \phi_1, U_1 \rangle$ and $P_2\langle \mathcal{R}_2, \rho_2, \phi_2, U_2 \rangle$, where $\mathcal{R}_1 = \mathcal{R}_2$ and $\phi_1 = \phi_2$ and $U_1 \subseteq U_2$ and $\rho_2 \Rightarrow \rho_1$. This means that the policies allow access to the same attributes in the same way for the same user, while one of the predicate sets is self-contained in the other. Hence, we can keep P_1 and discard P_2.

Modification Process. After the query rewrite check for a given query Q, Mascara must produce a complaint query Q^c for every possible policy combination $\mathcal{P}_Q \in \mathscr{P}_Q$. Algorithm 1 summarizes our modification process to produce a single complaint query Q^c that considers each policy $P \in \mathcal{P}_Q$ and adds an anonymized relation (recall Section 3.2) that can be indeed be disclosed instead of the one specified in the query (lines 3 and 4). This step is akin to rewriting queries in the presence of authorization views [54], making the rewriting process query aware as the masking does not occur only in the resulting relation but when accessing the data. It then proceeds to modify all filter predicates (lines 5–25). This is a crucial step where Mascara tries to preserve the user-specified predicates, ensuring correct query semantics. To this end, for each predicate, the algorithm performs a *predicate computability test*, which, when successful, modifies the query predicate into a "weaker" one, ensuring correct query semantics yet adhering to the disclosure policies. A similar test, that is the *aggregate computability test*, is also performed if the query is an aggregate query (lines 26–33). The algorithm modifies the grouping attributes and the aggregate functions , considering policies and query semantics. Lastly, for the projection attributes, the algorithm simply preserves the attributes that can be disclosed.

We now detail the predicate computability and aggregate computability tests.

Predicate Computability Test. This check is required for query predicates that access masked attribute(s). For all other predicates the algorithm (lines 8 and 12, and 14 and 16) either drops the predicate (i.e., when the attribute (s) cannot be disclosed at all) or preserves the predicate (i.e., when the attribute(s) can be disclosed and do not require masking, e.g., `zipcode` in policy P_3).

For predicates of the form $\alpha\theta c$, where α is some attribute in Q, and c is some constant, the algorithm tests if the predicate can be modified to preserve query semantics and disclosure policy. In particular, it checks the following property of the masking function.

Property 1 (θ-preserving). *For a predicate $\alpha\theta c$, the masking function μ_α is θ-preserving if $\alpha\theta c \implies \mu_\alpha(x \in \boldsymbol{dom}(\alpha))\theta\mu_\alpha(c)$ and for predicate $\alpha\theta\beta$, the masking functions μ_α and μ_β are θ-preserving if $\alpha\theta\beta \implies \mu_\alpha(x \in \boldsymbol{dom}(\alpha))\theta\mu_\beta(y \in \boldsymbol{dom}(\beta))$*

Algorithm 1 Query Modification.

Require: $Q, \mathcal{P}_Q$
Ensure: Q^c
1: $Q^c \leftarrow$ `null` ▷ *Initialize Q^c to a null expression*
2: **for all** $P \in \mathcal{P}_Q$ **do**
3: $T \leftarrow \{\pi_{\alpha_1,\ldots,\alpha_{|A_P|},\mu_{\beta_1}(\beta_1),\ldots,\mu_{\beta_{|A_m|}}(\beta_{|A_m|})}(\sigma_\rho(\mathcal{R}))\}$ ▷ *$\alpha_i \in A_P; \beta_i \in A_m$*
4: addRelation(Q^c, T) ▷ *add anonymized relation*
5: **for all** $p \in F_Q$ **do**
6: **switch** (p)
7: **case** $\alpha\theta c$: ▷ *α is an attribute and c is some constant*
8: **if** $\alpha \in A_P$ **then**
9: **if** $\alpha \in A_m$ and testPredicateComputability($\alpha, \mu_\alpha, \theta, c$) **then**
10: addFilter(Q^c,buildPredicate($\alpha, \mu_\alpha, \theta, c$))
11: **else if** $\alpha \notin A_m$ **then**
12: addFilter(Q^c, p)
13: **case** $\alpha\theta\beta$: ▷ *both α and β are attributes*
14: **if** $\alpha \in A_P$ and $\beta \in A_P$ **then**
15: **if** $\alpha \notin A_m$ and $\beta \notin A_m$ **then**
16: addFilter(Q^c, p) ▷ *both attributes can be disclosed*
17: **if** $\alpha \in A_m$ and $\beta \in A_m$ **then**
18: **if** testPredicateComputability($\alpha, \mu_\alpha, \theta, \beta, \mu_\beta$) **then**
19: addFilter(Q^c,buildPredicate($\alpha, \mu_\alpha, \theta, \beta, \mu_\beta$)) ▷ *both attributes require masking*
20: **if** $\alpha \in A_m$ and $\beta \notin A_m$ **then**
21: **if** testPredicateComputability($\alpha, \mu_\alpha, \theta, \beta$) **then**
22: addFilter(Q^c,buildPredicate($\alpha, \mu_\alpha, \theta, \beta$)) ▷ *only one requires masking*
23: **if** $\alpha \notin A_m$ and $\beta \in A_m$ **then**
24: **if** testPredicateComputability($\alpha, \theta, \beta, \mu_\beta$) **then**
25: addFilter(Q^c,buildPredicate($\alpha, \theta, \beta, \mu_\beta$)) ▷ *only one requires masking*
26: **if** Q is an aggregate query **then**
27: **for all** $a \in A_Q \setminus G_Q$ **do**
28: **if** $a \in A_P \setminus A_m$ **then**
29: AddAggregate(Q^c, (a, f_a)) ▷ *preserve aggregate*
30: **else if** $a \in A_m$ and testAggregateComputability(a, f_a, m_a) **then**
31: AddAggregate(Q^c, buildAggregate(a, f_a, m_a))
32: **for all** $a \in G_Q \cap A_P$ **do**
33: AddGroupBy(Q^c, a) ▷ *all group by attributes that can be disclosed*
34: **else**
35: **for all** $a \in A_Q \cap A_P$ **do**
36: addProjection(Q^c, a) ▷ *project on attributes that can be disclosed*

Whenever the masking function is θ-preserving, the algorithm modifies the predicate into $\mu_\alpha(\alpha)\theta\mu_\alpha(c)$ (line 10). For example, consider the predicate $age \geq 18$ when $\mu_{age} = bucketize(5)$, the algorithm (line 10) modifies the predicate to $bucketize(age, 5) \geq bucketize(18, 5)$, which is equivalent to $bucketize(age, 5) \geq [15, 20)$. Note that we do not expect the user to provide a buildPredicate function. Mascara only needs to know whether the masking function changes the attribute's datatype.

Likewise, for the predicates that are of the form $\alpha\theta\beta$, where both α and β are attributes, the algorithm tests the θ-preserving property and modifies the predicate either as above, i.e., when only one attribute requires masking or as $\mu_\alpha(\alpha)\theta\mu_\beta(\beta)$ when both attributes require masking.

Aggregate Computability Check. This check is required when aggregated attributes cannot be disclosed without masking. For other attributes, the algorithm either preserves or drops the aggregates (lines 28 and 29). For attributes requiring data masking, the algorithm tests if the aggregate can be modified using the masking function based on the following property.

Property 2 (γ-preserving). *Given an attribute a and an aggregate function f_a, a masking function $\mu_a : \boldsymbol{dom}(a) \rightarrow \boldsymbol{dom}(a')$ is aggregate preserving if there exists a function $\tau : \boldsymbol{dom}(a') \rightarrow X$ such that the aggregation $f_a(\tau(\mu_a(x \in \boldsymbol{dom}(a))))$ is computable.*

In general, the algorithm preserves the aggregation if μ_a does not change the datatype of the original attribute. For other cases, we try to convert the datatype into an aggregable datatype. For example, for f_a = `AVG` and `a=age` and μ_a = `bucketize(5)` we create an aggregate using a new

Listing 2. Query modification example.

Example User Query Q_2

```
select disease, avg (age) from patient, diagnosis
where patient.p_id = diagnosis.p_id and zipcode = 10117
group by disease
```

Disclosure Compliant Query Q^c_{2a}

```
select p3_t.disease, avg (p3_t.age_t) from (
  select p3.disease, int4range(p3.age_m) as age_t from (
    select bucketize(age, 5) as age_m, blur(zipcode, 3) as zipcode_m, disease from patient, diagnosis
    where patient.p_id = diagnosis.p_id) as p3
  where p3.zipcode_m = 10xxx) as p3_t
group by p3_t.disease
```

Disclosure Compliant Query Q^c_{2b}

```
select p4.disease_m, avg (p4.age) from (
   select age, generalize_disease(disease) as disease_m from patient, diagnosis
   where patient.p_id = diagnosis.p_id) as p4
group by p4.disease_m
```

scalar function τ : `int4range` $\rightarrow$ `int` that changes a range with the value in the middle of the range. Now we modify the aggregate as $\texttt{AVG}(\tau(\mu_a(a))$. Note that in contrast to BUILDPREDICATE the user must now provide an extra τ function for BUILDAGGREGATE. However, MASCARA already provides multiple transformations to change some traditional types to aggregable types. DBAs can extend this collection to enrich the modification of aggregate functions.

θ- and γ-preserving properties allow us to handle schema changes caused by some masking functions. This ensures that MASCARA always generates a semantically correct disclosure compliant query.

THEOREM 2. *Given a user query Q and a non empty set of policies $\mathcal{P}_Q$, the query modification algorithm (Algorithm 1) always generates a disclosure-compliant query Q^c that does not contradict any of the policies in $\mathcal{P}_Q$.*

PROOF. We prove Theorem 2 and the correctness of Algorithm 1 by contradiction. Assume that the algorithm generates a query Q^c that contradicts at least a policy $P\langle \mathcal{R}, \rho, \phi, U \rangle \in \mathcal{P}_Q$. This can only happen in two scenarios: i) Q^c projects at least one attribute in a different way as described by ϕ, or ii) Q^c does not filter the tuples as described by ρ. Neither of these cases can happen as Algorithm 1 builds the query Q^c bottom-up. Line 3 enforces compliant access to relations $\mathcal{R}$ since the beginning, masking sensitive data as described in ϕ and filtering out tuples as described in ρ. The other operators only process data that is already compliant with the policies. Therefore, the algorithm always generates a disclosure-compliant query Q^c that does not contradict any of the policies in $\mathcal{P}_Q$. □

Example. Listing 2 shows an example query Q_2 issued by an analysts and the modified queries Q^c_{2a} and Q^c_{2b}. Here the query rewrite check determines the data masks $\mathscr{P}_{Q_2} = \{\{P_3\}, \{P_4\}\}$ that are applicable and generates the following two modified (disclosure-compliant) queries.

For Q^c_{2a}. the algorithm first builds an anonymized relation for P_3 (line 3) and then modifies filter predicate `zipcode = 10117` to `blur(zipcode, 3) = 10xxx` (line 10). Since the data type of the

age attribute has changed as it is masked by the `bucketize(5)` function. Therefore, it injects the scalar function `int4range` to convert the range back into an integer (line 29). Likewise, for Q^c_{2b}, the algorithm (line 3) builds an anonymized relation for P_4, which does not provide access to the `zipcode` attribute. Therefore, the algorithm drops the predicate `zipcode = 10117` (line 8). Finally, the references to aggregating attributes (line 31) and the grouping attributes (line 33) are added to Q^c_{2b}.

6 UTILITY ESTIMATION

We now turn attention to Mascara's utility estimator, which allows estimating the "information loss" due to modifying the user query. Mascara then uses these estimates to select the disclosure-compliant query with the least information loss.

6.1 Relative Entropy as a Distance Metric

Relative entropy, also called Kullback-Leibler (KL) divergence, is a class of statistical distance that measures the similarity between two distributions [35]. A relative entropy of zero indicates that both distributions have identical quantities of information, and the higher the relative entropy, the more significant their difference. Following this definition, we propose to measure the relative entropy between the original and the anonymized data and take that value to measure the $\texttt{dist}(Q, Q^c)$. We chose relative entropy for measuring information loss as (a) it is a well known metric for quantifying the similarity between two distributions; (b) it can be adapted when the attribute's domain changes as a result of data masking; and (c) it is computed upon values' frequencies, which can be estimated using traditional database statistics.

Consider an attribute a of relation (R) and let $f(x, a)$ denote the frequency of $x \in \mathbf{dom}(a)$. We can then define the probability distribution $p(x) = \frac{f(x,a)}{|R|}$ where $|R|$ denotes the cardinality ($\#tuples$) of the relation R. Now consider a masking function $\mu : \mathbf{dom}(a) \rightarrow \mathbf{dom}(a')$ to mask attribute a to a masked attribute a'. For now, assume that $\mathbf{dom}(a) = \mathbf{dom}(a')$ (we will drop this assumption later). We can define the distance between the two queries $Q \equiv \pi_a(R)$ and $Q^c \equiv \pi_{\mu(a)}(R) = \pi_{a'}(R)$ as

$$D_{KL}(p(x) \| p'(x)) = \sum_{x \in \mathbf{dom}(a)} p(x) \ln \frac{p(x)}{p'(x)} \tag{1}$$

where $p'(x) = \frac{f(x,a')}{|R|}$ is the probability distribution of masked values. We can then compute the distance $\texttt{dist}(R, R')$ between a relation R and a disclosure compliant relation R' as

$$\sum_{i=1}^{\deg(R)} D_{KL}(p(x_i) \| p'(x_i)) \tag{2}$$

where $x_i \in \mathbf{dom}(a_i)$.[2]

Computing the KL divergence is straightforward in principle, however, computing Equation 1 in our setting is far from trivial due the following four challenges.

- Masking functions often change the domain, i.e., $\mathbf{dom}(a) \neq \mathbf{dom}(a')$ which makes computing computing $p'(x)$ non trivial.
- The definition of relative entropy holds true only when, for every value of x, $p'(x) = 0 \implies p(x) = 0$ (ensuring absolute continuity). In cases where the condition is not met, relative entropy is commonly expressed as $+\infty$. The challenge arises when the masking function does not ensure this absolute continuity, which would cause a relative entropy of $+\infty$.
- The third challenge arises when dealing with attributes with continuous domains.

[2]We assume statistical independence among all attributes in a relation.

- The fourth challenge is in estimating $p(x)$ and $p'(x)$ as it is infeasible to first execute every compliant query and then to determine the best one.

To address the first two challenges, we categorize masking functions into generalization- and perturbation-based masking functions. We then discuss how to deal with continuous domains and estimate $p(x)$ and $p'(x)$ for Q and the modified query Q^c.

6.2 Generalization-Based Data Masking

Generalization-based masking functions, such as blurring or bucketization, are essentially "many-to-one" functions that map multiple values from the input domain $\mathbf{dom}(a)$ to values in the generalized domain $\mathbf{dom}(a')$. Masking via generalization implies that every value from the generalized domain represents a set of values from the input domain, causing that without external information, it is impossible to map back a value from $\mathbf{dom}(a')$ to $\mathbf{dom}(a)$ with absolute certainty. In most of the cases, the input domain does not contain any value from the generalization domain, i.e., $\mathbf{dom}(a) \cap \mathbf{dom}(a') = \emptyset$, meaning that a generalization can potentially change the domain and data type of every input value.

To better illustrate the effect of generalizations on sensitive data, consider the `bucketize()` masking function denoted by μ_1 from the policy P_3 (Section 2.1) that should be applied to the patient's `age` (Figure 1b). μ_1 buckets each value in ranges of size 5, e.g., $\mu_1(62) = [60, 65)$. We initially only consider generalizations on discrete domains. Hence, a user trying to guess the original age could only do it with 20% of certainty since there are only five possible values inside the bucket for this domain.

We can observe that the masked values of attribute `age'` after applying μ_1 are not contained in the age attribute's domain $\mathbf{dom}(\text{age}) = \left\{ x | \forall x \in Z^{0+} \wedge x \in [0, 125] \right\}$. This is a problem, as mentioned above, where the masking function changes the domain of values. To compute Equation. 1 we redefine the probability distribution of a masked attribute a' in terms of the original domain instead of the masked domain as follows.

For $x \in \mathbf{dom}(a)$, let $G_x = \{ v \mid \mu(v) = \mu(x) \}$ denote the set of values that map to the same generalized value. We can then define the probability distribution $p'(x)$ as

$$p'(x) = \frac{1}{|R|} \sum_{x' \in G_x} \frac{f(x', a)}{|G_x|} \tag{3}$$

. We can now compute the relative entropy between $p(x)$ and $p'(x)$ to compute information loss when an attribute is masked with a generalization-based masking function.[3] To tackle the second challenge when both $p(x)$ and $p'(x)$ have to fulfill the absolute continuity property, we leverage the following theorem.

Theorem 3. *Let $\mu : \mathbf{dom}(a) \rightarrow \mathbf{dom}(a')$ be a generalization-based masking function. Then $p'(x) = 0 \implies p(x) = 0$.*

Proof. For $x \in \mathbf{dom}(a)$, let $p'(x) = 0$,

$$\begin{aligned} &\implies \sum_{x' \in G_x} f(x', a) = 0 \quad \text{From Equation 3} \\ &\implies f(x, a) = 0 \quad \text{as } x \in G_x \\ &\implies p(x) = 0 \end{aligned}$$

□

[3]For most of generalizations used in practice $|G_x|$ is either a constant or straightforward to compute. Based on this, we use $|G_x|$ Eq. 3 for simplifying exposition; in practice, we collect statistics from the masked attribute a' and approximate $p'(x)$.

6.3 Perturbation-based Data Masking

Perturbation-based masking functions protect sensitive data by adding random noise and do not change the data type and domain of the original data, i.e., $\mathbf{dom}(a) = \mathbf{dom}(a')$.

Take as an example the policy P_4 (Section 2.1), where the patient's `expenses` attribute must be masked by adding random noise. Let μ_4 be the function that adds up to 5% of random noise to each value. For instance, for input the value 38,035.00 we get a random value in the range between 36,133.25 and 39,936.75.

Since perturbation-based masking functions do not change the original value domain, we can use Equation 1 to compute the relative entropy. However, based on the unpredictable nature of perturbations, is not possible to ensure that the masked distribution fulfills the absolute continuity property. This can become a problem because we would qualify the entire masked attribute with an infinite information loss based on a single value where the property does not hold.

More precisely, the random behavior of perturbation-based data masking can cause a value $x \in \mathbf{dom}(a)$ to be present in a but not in a', i.e., $p'(x) = 0$ but $p(x) \neq 0$, breaking the absolute continuity property. To address this, we adapt $p'(x)$ as follows:

$$p'(x) = \begin{cases} \frac{1}{|R|+\varepsilon} & \text{if } x \notin a' \wedge x \in a \\ \frac{f(x,a')}{|R|+\varepsilon} & \text{otherwise} \end{cases} . \tag{4}$$

The key idea is to "force" $p'(x)$ to fulfill the absolute continuity property at the cost of sacrificing precision in computing some probabilities. We ingest a fake value into $p'(x)$ when $x \notin a' \wedge x \in a$. As shown in Equation 4 this forces us also to add these fake elements to the total number of elements in a, which was originally $|R|$. We denote this error as ε, which is the number of fake values required to fulfill the absolute continuity property, i.e., $\varepsilon = |\{x | \forall x \in \mathbf{dom}(a) \wedge x \notin a' \wedge x \in a\}|$. This hurts the probability of other values as the expected error for every value $x \in a'$ is $\frac{f(x,a')}{|R|} \frac{\varepsilon}{(|R|+\varepsilon)}$. In the best case scenario, $p'(x)$ already fulfills the absolute continuity property, making $\varepsilon = 0$ and the error in every value. Moreover, in the worst case scenario, all values in a have to be added as fake values, making $\varepsilon = |R|$ and the probability error for every value $\frac{f(x,a')}{2|R|}$, which is a relative error of 50%.

6.4 Dealing with Continuous Domains

So far, the above adaptations only work for discrete domains. To deal with attributes with continuous domain, we first discretize the values and proceed as above. Note that we only discretize to compute the relative entropy while preserving the original and masked attributes.

Discretizing a continuous domain is akin to applying a generalization-based masking function. We first define a discretization function δ, which maps values from a continuous domain $\mathbf{dom}(a)$ into a discrete domain $\mathcal{D}$. Denote by δ^{-1} the mapping from the discrete domain into a set of values from the continuous domain. We adapt the probability distributions $p(x)$ and $p'(x)$ in terms of the discrete domains $p(d)$ and $p'(d)$ as follows.

$$p(d) = \sum_{x \in \delta^{-1}(d)} \frac{f(x,a)}{|R|} \tag{5}$$

For generalized-based masking, we have

$$p'(d) = \frac{1}{|R|} \sum_{x \in \delta^{-}1(d)} \sum_{x' \in G_x} \frac{f(x',a)}{|G_x|} \tag{6}$$

and for perturbation-based masking, we have

$$p'(d) = \begin{cases} \frac{1}{|R|+\varepsilon} & if \delta^{-1}(d) \cap a' = \emptyset \wedge \delta^{-1}(d) \cap a \neq \emptyset \\ \sum_{x \in \delta^{-1}(d)} \frac{f(x,a')}{|R|+\varepsilon} & else \end{cases} \tag{7}$$

6.5 Estimating Information Loss

We now describe how we leverage the above estimation framework in practice to compute $\texttt{dist}(Q, Q^c)$. First, consider simple queries of the form $Q \equiv R_1 \times \cdots \times R_n$. Likewise, $Q^c \equiv R'_1 \times \ldots R'_n$ a disclosure compliant query where R' corresponds to the anonymized relation R. We can compute $\texttt{dist}(Q, Q^c)$ following Equation 2.

For queries of the form $Q \equiv \pi_A(\sigma_C(\boldsymbol{R}))$ and corresponding $Q^c \equiv \pi_{A'}(\sigma_{C'}(\boldsymbol{R'}))$, where $\boldsymbol{R} = R_1 \times \cdots \times R_n$ and $\boldsymbol{R'} = R'_1 \times \cdots \times R'_n$, we compute the distance as

$$\texttt{dist}(Q, Q^c) = \left(1 + \left|1 - \frac{|Q^c|}{|Q|}\right|\right) \texttt{dist}(\pi_A(\boldsymbol{R}), \pi_{A'}(\boldsymbol{R'})) \tag{8}$$

where $|Q|$ denote the cardinality of the query. The intuition behind the above equation is to reward modified queries that return a most accurate set of tuples compared to the original query, for which we use cardinality estimates to determine the set of tuples.

When queries additionally have aggregations, we replace A with attributes in the select clause. This is because the loss of information due to modified aggregations directly depends on the underlying distribution of values.

In an ideal scenario, one could execute every compliant query and then use Equation 1. However, this is not feasible as this would require processing all compliant queries to generate the resulting tables and then counting the frequency of all tuples to compute. Therefore, to enable a practical solution, we piggyback on available database statistics to estimate the values' frequencies and then compute the relative entropy upon these estimates. In particular, for an attribute a, we estimate its values based on commonly available database statistics such as number of distinct values, most common values, most common frequencies, and histogram bounds (considering equi-depth histograms).

Example. Now consider the two compliant queries presented in Table 2 to answer Q_2. The first one, Q^c_{2a}, accesses the disease attribute's exact distribution while masking the age in ranges of 5 and performing the filter only on a masked zip domain. Q^c_{2b}, on the other hand, does not filter any tuple because the zip attribute is not available and then projects the exact distribution of age while having a generalized disease attribute. Following Eq. 8, we start by only computing (or estimating) the relative entropy of the projected attributes. Regardless of the distribution, we could intuitively argue that, on average, the relative entropy of Q^c_{2a} is lower than Q^c_{2b}. This is because Q^c_{2a} generalizes the age attribute in buckets of 5 possible elements, while Q^c_{2b} generalizes the disease attribute into buckets of 10 possible values (as shown in Figure 1f). The unmasked attributes get a relative entropy of 0. Still, to be sure, the system must estimate these values based on the database statistics. Finally, we want to reward queries that return a closer amount of tuples than the original query. Here again, Q^c_{2b} returns a better result, as Q^c_{2b} removes the predicate because the zip attribute is not available, i.e., $\left(1 + \left|1 - \frac{|Q^c_{2a}|}{|Q_2|}\right|\right) < \left(1 + \left|1 - \frac{|Q^c_{2b}|}{|Q_2|}\right|\right)$. Hence, Mascara would choose Q^c_{2a} as the best compliant query to answer Q_2.

7 EXPERIMENTAL EVALUATION

We evaluated Mascara with the goal of investigating four main aspects: (i) suitability of relative entropy as a distance metric; (ii) its overhead of modifying a user query and selecting a disclosure compliant one; (iii) accuracy of its approximation methods to select the disclosure-compliant query

Table 3. List of value-based masking functions.

Name	Type	Parameters
suppress	suppression	–
bucketize	generalization	bucket_size
generalize_number	generalization	divisor
blur_phone	generalization	–
generalize_date	generalization	level
add_relative_noise	perturbation	percentage
add_absolute_noise	perturbation	max_noise
add_laplace_noise	perturbation	sensitivity, epsilon
add_noise_date	perturbation	level, max_noise
add_noise_cow	perturbation	probability
add_noise_mar	perturbation	probability

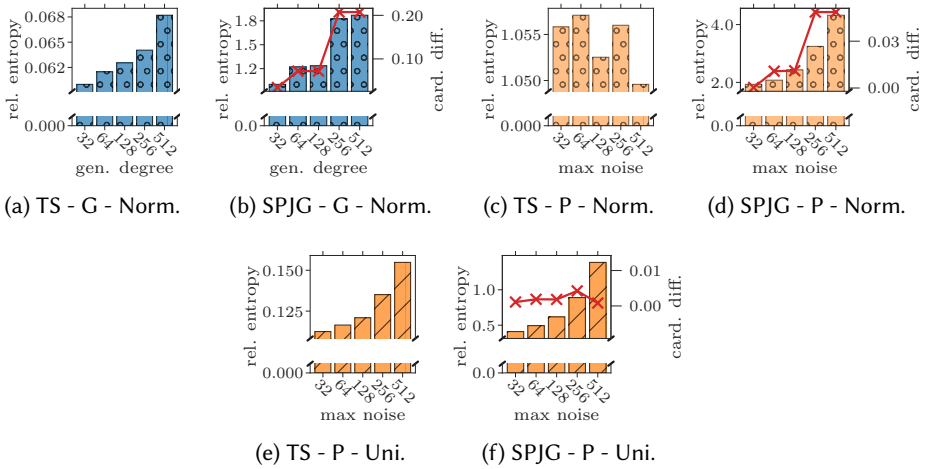
with least information loss; and (iv) cost of disclosure compliant queries with respect to unmodified queries (without data protection). We found that:

- Relative entropy is a good information loss metric as it linearly increases with the degree of masking. This reflects that the more a masking function generalizes or perturbates sensitive data, the higher the relative entropy with respect to the original.
- system's efficiency is almost independent of the size of the collected statistics, and the overhead incurred by our approach is lower than 300ms compared to the normal execution of a query without data protection.
- Our approximations allow Mascara to select one of the best three disclosure complaint queries with 90% of probability. Furthermore, for single table queries, Mascara always chooses the best compliant query.
- Cost of disclosure compliant queries over unmodified is up to 50× when masking on the fly. However, using materialized views, the cost is similar to that of unmodified queries. [60]

7.1 Experimental Setup

Setup. We implemented Mascara in Java (JDK 1.11) using JDBC to connect to an underlying PostgreSQL (v14.10) [60] database. We used Apache Calcite (v1.34.0) [11] to parse SQL queries into logical plans, which Mascara uses to rewrite them into compliant queries before submitting them to the underlying database system. We implemented all masking functions using the PL/pgSQL procedural language [51]. We performed the microbenchmark of all efficiency experiments using Java Microbenchmark Harness (JMH) (v1.33) [32]. We configured JMH to call every tested operation hundreds of times after ten sets of warmup iterations and report the average time along with the 99.9 percentile for the error range. We ran all experiments on a machine equipped with an Intel i7-1370P CPU and 64GB of main memory running Microsoft Windows 11.

Datasets. We used two datasets. First, the TPC-H benchmark dataset [61] with a scale factor (SF) of one. Note that the SF has no impact as long as the attributes' value ranges remain the same (except for attributes with unique values). All the synopses (equi-depth histograms) used to approximate relative entropy can only lose accuracy when expanding the range of possible values. We also made a modification in the customer table for the experiments in Section 7.2, where we generated the c_acctbal attribute following four different distributions: first a sparse (50% of possible values present) and a dense (100% of possible values present) uniform distribution, and a sparse and dense normal distribution. Second, we used the Folk US-census dataset from 2017 [18] (*census_data*), to test our solutions on a real-world application (Section 7.4).



(a) TS - G - Norm. (b) SPJG - G - Norm. (c) TS - P - Norm. (d) SPJG - P - Norm.

(e) TS - P - Uni. (f) SPJG - P - Uni.

Fig. 3. Effect of data distribution and policies on relative entropy: (a), (c), and (e) tablescans (TS); and (b), (d), and (f) SPJG queries ; (a), (b) masked with generalization (G) and (c), (d), (e), and (f) with perturbation (P).

Query Workload. Mascara currently supports only SPJG queries. We generated 33 heterogeneous queries upon five representative TPC-H queries: Q_1, Q_3, Q_5, Q_6, and Q_{10}. We also prioritize these queries because they mainly focus on three main tables: lineitem, customer, and orders. This way, we can focus on specifying multiple disclosure policies only on those tables and force more combinations to make the compliant query selection more challenging. Finally, for the Folk dataset, we evaluate the impact of system's query modification on predicting the individual's income (ACSIncome prediction task [18]) and classifying if an individual's income is above $50,000 a year (ACSIncome classification task [18]).

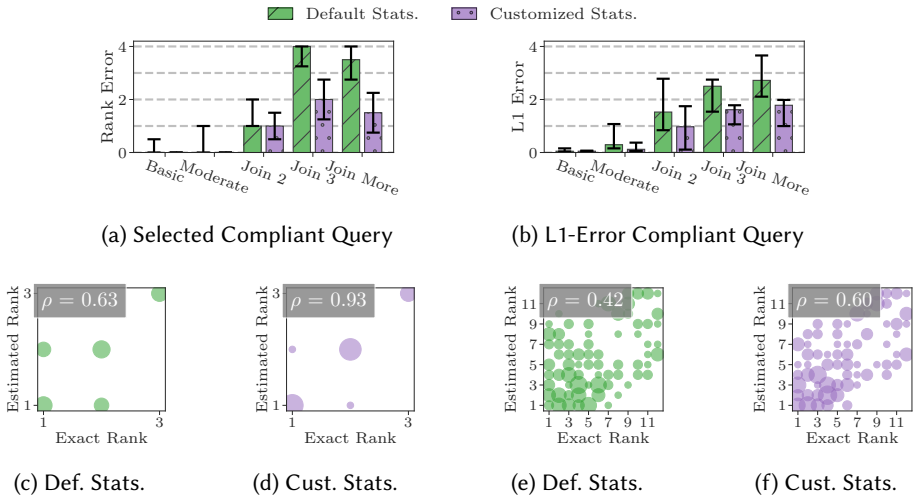
Disclosure Policies and Masking Functions. We implemented a collection of 11 value-based masking functions on PostgreSql (Tab. 3). We defined the disclosure policies on the lineitem, customer, orders, and census_data tables, where different attributes are available or masked depending on the experiment. We also consider a single user role for the experiments because the number of user roles does not affect system's performance.

7.2 Relative Entropy as Information Loss Metric

In our first set of experiments, we evaluate the efficacy of using relative entropy (Equation 1) as a distance metric. Our goal is to evaluate (i) if relative entropy increases when increasing the degree of masking, (ii) the impact of the choice of masking (perturbation or generation) and the data distribution, and (iii) its efficacy for SPGJ queries.

Setup. For this experiment, we use different versions of the customer table with varying distributions, as mentioned above. We use a `bucketize()` as a representative generalization-based masking function and `add_absolute_noise()` as a perturbation-based. We gradually increase the degree of masking in terms of the bucket size and maximum noise to be added. We also repeated the experiment, increasing the complexity of the query by adding more operators to the query, such as joins, selections, and aggregates.

Results. Figure 3 shows selected results from our experiments. Due to space constraints, we omitted discussion on all configurations. Most configurations show the same pattern as using a generalization on a dense normal distribution (Fig. 3a and Fig. 3b), with relative entropy increasing with the degree of anonymization. The exception is applying perturbation on a sparse normal distribution (Fig. 3c), where the error results from our adaptation for relative entropy to ensure



(a) Selected Compliant Query (b) L1-Error Compliant Query

(c) Def. Stats. (d) Cust. Stats. (e) Def. Stats. (f) Cust. Stats.

Fig. 4. Accuracy of utility estimation: (c) and (d) for single table queries; and (e) and (f) for join queries.

absolute continuity for perturbations (Section 6.3). In this sparse distribution, the probability of finding a value that does not meet the absolute continuity property increases, reducing precision for other attribute values. This effect dissipates with more operators in the query (Fig. 3d with join, selections, and aggregates). The red lines in Fig. 3b, 3d, and 3f show the relative cardinality difference between the original and compliant results. Generally, there is a correlation between the cardinality difference and increasing relative entropy, motivating our use of cardinality estimates and database statistics (Section 6.5). The exception is perturbation in a uniform distribution (Fig. 3f), where the cardinality difference remains constant and near zero, indicating minimal impact on information loss estimates.

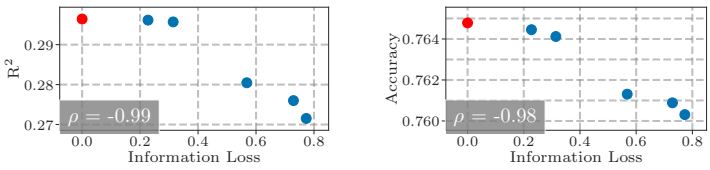
Overall, our adaptation of relative entropy is a reliable metric to measure the distance between a disclosure-compliant and user query.

7.3 Accuracy of Utility Estimator

We now evaluate the accuracy of the utility estimator. To do so, we first execute all disclosure-compliant queries and rank them by their real relative entropy. We then compare these real ranks to the queries' rank based Mascara's estimated relative entropy.

Setup. We used the TPC-H dataset, created policies to mask the orders, customers, and lineitem tables, and then used the collection of TPC-H queries as explained in Section 7.1. We classify these queries into five categories depending on their complexity: **Basic** queries of the form `select * from [T]`, **moderate** queries `select [Attrs.] from [T] where [condition]`, **join-2** are queries `select [Attrs.] from [T1,T2] where [condition]`, finally, **join-3** and **join-more** are complex queries that are defined upon three or more tables.

Results. Figure 4 shows the performance of system's estimation methods based on query complexity and statistics used. We ran the experiment with two configurations. In green, we see system's performance using PostgreSQL's default statistics size (100) for all attributes. In purple, the results are shown when collecting more statistics (size of 10,000) for masked attributes in the lineitem table, the largest in our experiment. Figure 4a shows the median rank error (`estimated_rank - 1`) of Mascara 's chosen compliant query. Mascara almost always selects the best compliant query for single table requests (basic and moderate queries). However, as the number of tables in the query increases, the median rank error can rise to 4 (join-3 with default statistics) or 2 (join-3 with



(a) Prediction task. (b) Classification task.

Fig. 5. Impact on downstream application.

customized statistics). This occurs because Mascara cannot predict how attribute distributions change after joining tables, relying only on base table statistics. This issue is similar to cardinality estimation [28], as databases cannot foresee intermediate table states within a query. This effect is confirmed by observing the L_1-error of the chosen compliant query, which increases with query complexity (Fig. 4b).

In Figures 4c - 4f, we can observe the correlation of the estimated and exact ranking, evaluating not only the chosen query but the whole ranking. We specify Spearman's rank correlation coefficient in each figure's top right corner. Again, we can see that with the customized statistics, Mascara provides a much better ranking closer to the ground truth. However, we can observe that the correlation decreases as the complexity of the query increases.

Overall, system's approximate solutions to estimate relative entropy perform well on simple queries, but when increasing the complexity of the requested queries, the accuracy of system's estimates depends on the size and quality of the collected statistics.

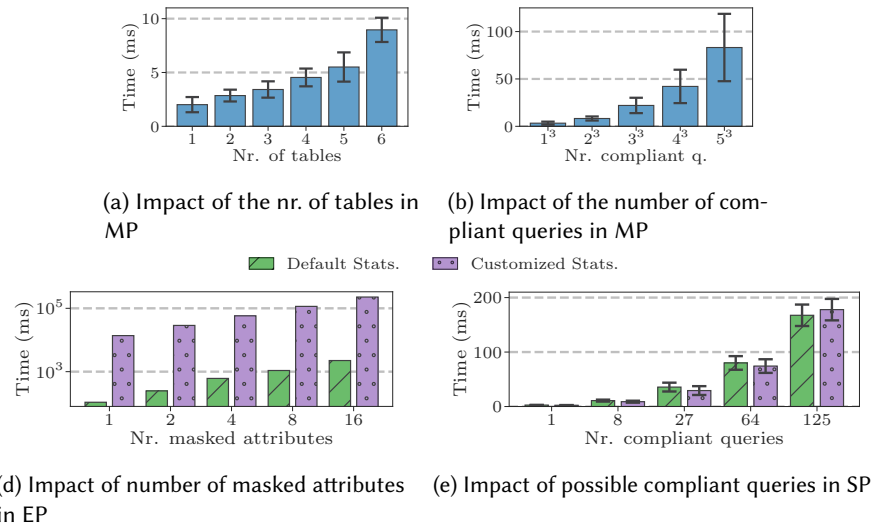
7.4 Example on Real-World Applications

We now evaluate if our information loss estimation can also be used as an indicator to foresee the performance of downstream applications consuming anonymized data. We do so by measuring the correlation between system's information loss metric and the score of ML models trained with anonymized data.

Setup. For this experiment, we use the Folk dataset [18] and commonly machine learning: the ACSIncome prediction task, which requires a model to predict an individual's yearly income, and the ACSIncome classification task, which classifies if the individual's yearly income is above $50,000. We stored 80% of the data in our database as the *census_data* table to be used as our training data. We created five policies masking various attributes from census_data for which Mascara produced five disclosure-compliant queries, which we executed individually to train a simple logistic regression model (classification task) and a linear regression model (prediction task) upon each resulting table. Finally, we used the remaining 20% of the data to test the score of each ML model.

Prediction Task. Figure 5a shows the results of our experiments with the prediction task. Colored in red, we can find our baseline, accessing all data without any data protection. Each blue point represents the result of one of the modified compliant queries. The x-axis denotes the information loss computed by Mascara, and the y-axis shows the score (R^2) of the model trained with the corresponding complaint query. We can observe that the complaint queries that obtained a higher information loss performed worse in the downstream application. The correlation between the information loss and the model's score obtained a value of -0.99, showing that the proposed information loss metric for Mascara can successfully capture the decaying utility of higher anonymized data, even for real-world applications.

Classification Task. Figure 5b shows the results for the classification task. Again, we can see the same pattern as in the prediction task. The compliant queries with higher information loss



(a) Impact of the nr. of tables in MP

(b) Impact of the number of compliant queries in MP

(d) Impact of number of masked attributes in EP

(e) Impact of possible compliant queries in SP

Fig. 6. Overhead of Mascara: (a) and (b) modification phase (MP), (c) estimation phase (EP), and (d) selection phase (SP)

performed worse in the downstream application. The correlation between the information loss and the model's score obtained a value of −0.98.

In sum, system's information loss metric has a strong correlation with the result's quality of the selected downstream applications.

7.5 Overhead of Query Modification

We measured the overhead of our modification approach (Section 5) depending on the number of tables requested in the query and the number of possible compliant queries generated.

Setup. We ran these experiments using the TPC-H dataset. For the first part of the experiment, we submitted different queries joining an increasing number of tables. For the second part, we submitted a query joining the lineitem, orders, and customer tables. Then, we defined an increasing number of available policies for each base table, forcing Mascara to generate exponentially more disclosure complaint queries.

Impact of the # Tables in the Query. Figure 6a shows how the runtime of system's modification phase when generating a single disclosure-compliant query for queries with an increasing number of tables. We can see that the runtime grows almost linearly with the number of tables involved in the submitted query. This occurs because the more tables are involved in the query, the more iterations Mascara has to perform to modify the requested query. Still, to generate a single complaint query, the runtime remains lower than 10 ms, even for queries accessing six tables.

Impact of the # Disclosure Policies. Figure 6b shows the duration of the modification phase depending on the number of disclosure-compliant queries generated. We can see that the runtime linearly increases with the number of complaint queries. This is because Mascara generates a compliant query for every possible combination of the available policies for the submitted query. Even for extreme scenarios of 125 possible queries, the runtime of the modification phase remains around 80 ms.

Overall, system's query modification has an acceptable overhead.

7.6 Estimating Information Loss

Mascara estimates the information loss of masked attributes offline when new statistics have been collected (Section 6.5). Here, we evaluate the cost of this estimation phase, which depends on the size of the collected statistics and the number of masked attributes.

Setup. For this experiment, we defined a set of disclosure policies using the TPC-H dataset, where we gradually increased the number of masked attributes.

Results. Figure 6d shows the pre-processing time that Mascara takes to estimate the information loss of the masked attributes. The green bars show the performance when using the statistics of the default size (100) and the purple bars show the performance when using the statistics of the customized size (10,000). We can observe that the runtime of the estimation phase grows almost in the same degree as the size of the collected statistics (2 orders of magnitude). However, this task has to be performed only once, when the statistics of the protected table are updated. Then, Mascara reuses these estimates for the following queries.

In summary, the runtime of the estimation phase grows linearly with the size of the statistics. Moreover, this task has to be performed only once, distributing its cost over multiple queries.

7.7 Selecting the Best Complaint Query

In this section, we evaluate the efficiency of the query selection phase by measuring the runtime of the selection phase and varying the number of disclosure-compliant queries.

Setup. Similar to the experiment described in Section 7.5, we submitted a query joining the lineitem, orders, and customer tables and increased number of available policies for each base table.

Results. Figure 6e shows the results where the green bars represent system's performance when using default statistics (size of 100) and the purple bars show when using customized statistics (size of 10,000). We can observe that the performance of the selection phase is independent of the size of the statistics because this only affects the efficiency of the estimation phase (Fig. 6d). The selection phase only reuses the information loss estimates from the estimation phase, making this process only dependent on the number of disclosure-compliant queries. Even for extreme scenarios where a user query can be modified into 125 complaint queries, the runtime of the selection phase remains around 180ms.

Overall, system's selection phase is efficient and independent of the size of the collected statistics.

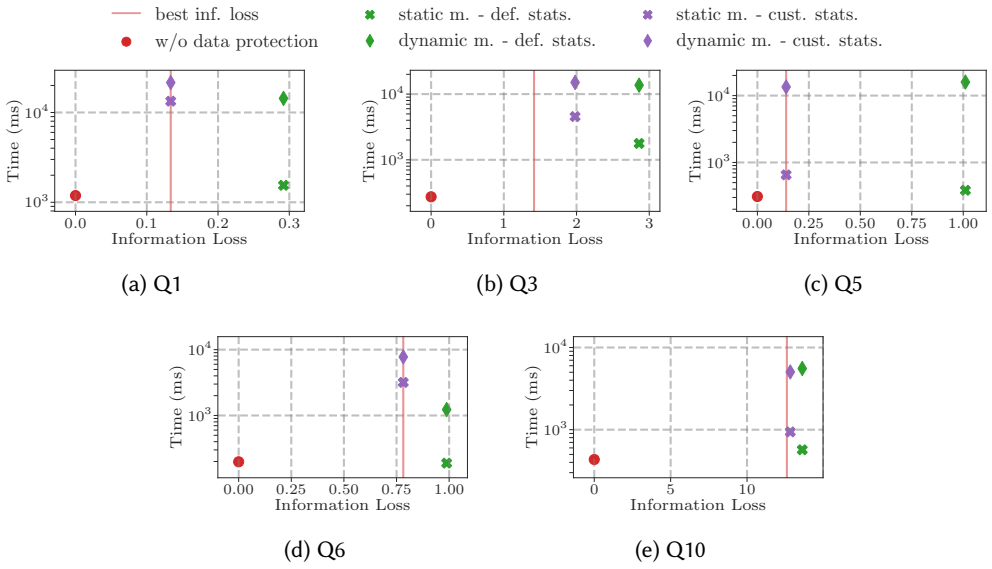
7.8 Cost of Compliance

Finally, we evaluate the cost of compliance by execution time of a complaint query with a non-compliant query.

Setup. For this set of experiments, we used the same tables and policies as in Section 7.3.

Results. Figure 7 shows the results of running TPCH queries Q_1, Q_3, Q_5, Q_6, and Q_{10} with and without Mascara. The y-axis indicates query runtime, and the x-axis shows the information loss of the selected compliant query. The red dot represents running the query without Mascara, resulting in zero information loss but without protecting accessed data. The red line indicates the information loss of the optimal compliant query. Green markers show performance using default statistics size (100), while purple markers show performance with customized statistics (size 10,000). Marker shapes distinguish between dynamic masking (diamonds) and static masking (cross).

We observed that most queries show similar runtimes for normal execution and static masking, as Mascara maintains a materialized view for each data mask. However, there are exceptions when Mascara needs to reconvert anonymized values for aggregates, causing overhead even with materialized views Figures 7a, 7b, and 7d. Running compliant queries with dynamic masking is always more expensive, up to an order of magnitude, because sensitive data must be anonymized



(a) Q1 (b) Q3 (c) Q5

(d) Q6 (e) Q10

Fig. 7. Cost of compliance: comparison between running a query with and without Mascara

during execution. Larger datasets do not affect system's query modification and selection phases but can significantly impact dynamic masking performance, as runtime is dominated by masking operations implemented as UDFs.

Regarding information loss, our conservative approach (Section 5) may lead to higher information loss when the modifier drops predicates. We observe for Q_{10}, where the optimal compliant query has an information loss of 12.6. This generally depends on how the policies are defined based on user access. Additionally, it is crucial to estimate the information loss accurately to select the optimal plan. In Q_5, the plan selected by Mascara with the default statistics shows a ten times higher loss than when using the customized statistics.

The last conclusion we can draw from our results is that analog to the experiments in Section 7.3, Mascara always chooses a better query when using more detailed statistics. For this experiment, it even chooses the optimal compliant query for queries Q_1 (Fig 7a), Q_5 (Fig 7c), and Q_6 (Fig 7d). The reason why Mascara does not always choose the optimal complaint query (Q_3 - Fig 7b and Q_{10} - Fig 7b) is because we take the assumption of statistical independence to estimate the information loss, which is a common practice for cost and cardinality estimation tasks [28, 65, 68]. In order to improve these results further, we should investigate multi-dimensional statistics.

Overall, disclosure-compliant queries do have an overhead. However, Mascara can minimize this overhead by using static masking whenever possible. Additionally, Mascara can always reliably translate a user query into a disclosure-compliant one that has the least information loss.

8 RELATED WORK

We now discuss ideas put forward in this paper to existing prior work, which we categorize into:

Data Masking. Table 4 compares current database systems that support data masking with Mascara. We consider eight dimensions: (i) *Anonymization Techniques:* All systems support suppression, perturbation-based, and generalization-based masking functions, as recommended by data regulations such as GDPR; (ii) *Data Masking:* Db2, Azure, and Redshift do not support static masking. They mask sensitive data whenever accessed. A workaround involves using data masking

Table 4. Comparison to current data masking solutions. S: Suppression, P: Perturbation, G: Generalization, and Sh: Shuffling. D: Dynamic Masking, S: Static Masking.

System	Anonymization Techniques	Data Masking	Supported Queries	User-Defined Masking	Query Aware	Schema Changes	Context Aware	Policies Across Multiple Tables
Db2 [31]	S,P,G	D	*	✓	✗	✗	✗	✗
Azure [41]	S,P,G	D	*	✗	✗	✗	✗	✗
Dalibo [36]	S,P,G,Sh	D,S	*	✓	(✓)	✗	✗	✗
Oracle [45]	S,P,G,Sh	D,S	*	✓	(✓)	✗	✗	✗
Redshift [7]	S,P,G	D	*	✓	(✓)	✗	✗	✗
Mascara	S,P,G	D,S	SPJG	✓	✓	✓	✓	✓

functions to define materialized views and grant users access to these views, but this does not allow for authorization transparent querying; (iii) *Supported Queries:* Only Mascara supports Select-Project-Group By-Join (SPGJ) queries; (iv) *User-Defined Masking:* All systems except Azure allow the user to implement and use new user-defined masking functions; (v) *Query Aware:* Query awareness involves modifying queries based on the semantics of the masking function. Db2 and Azure lack this, allowing attackers to bypass protection using inference or brute-force techniques. Other systems are query-aware and always mask sensitive data before processing but don't allow masking functions that change data types. This assumption lets traditional query modification techniques suffice to create syntactically correct, compliant queries; (vi) *Schema Changes:* Generalizations can change data at the schema level, requiring different query modification approaches for consistent, semantically correct, compliant queries. Only Mascara supports schema-changing masking functions; (vii) *Context Aware:* Only Mascara allows for the specification of multiple policies for the same relation, providing context-aware data protection; and (viii) *Policies Across Multiple Tables:* Only Mascara allows for the specification of policies across multiple tables.

Database Access Control. In addition to the discussion in Section 2.2, [15] underscores the importance combining access control with data protection policies. In this work, we extend upon view-based access control methods [12, 16, 55, 56]. Upon these concepts, we create novel query modification technique that can handle changes at the schema level (such as anonymizing using generalizations), which has not been studied before [27, 54].

Policy-aware Data Processing. Different notions of policy have been studied in the literature. For example, [8–10] studied policies pertaining to cross-border data transfers. [33] studied fine-grained access control policies that reflect personal consent in databases. Sieve [47], an access control middleware, focuses on managing a large number of policies efficiently. DataLawyer [62] uses provenance logs to audit compliance regarding accessing sensitive attributes. [48] discusses techniques to protect sensitive data when these data depend on other disclosable attributes. Data Station [66] is a data escrow that provides a policy framework to coordinate and execute data-sharing tasks in a trustworthy environment. IoT Notary [46] is a privacy-preserving system that enforces data-capturing policies in IoT devices. Extending Mascara along these works in an interesting direction for future work.

Cardinality Estimation. Our work also relates to cardinality estimation (CE) in databases. Recall that we use cardinality estimates and database statistics to approximate the calculation of relative entropy. CE is still considered an open problem as data correlation and functional dependencies are not always captured by the statistics built upon single columns [28, 65, 68], which we observe in our

results. While we piggyback on underlying database statistics, an exciting direction for future work will be to explore how cardinality estimation methods such as synopses-based methods [13, 37], ML-based data driven [44, 68], or hybrid techniques [65] can adapted to our information loss estimation framework.

Privacy-Preserving Query Processing. Differential privacy uses a privacy budget to decide how much noise is required to successfully protect the privacy of every single value while keeping the highest data utility possible [19, 40]. In contrast to differential privacy, Mascara's goal is to rewrite queries based on a given set of disclosure policies. Mascara does not guarantee privacy requirements beyond the ones defined by data officers with every disclosure policy. With Mascara data, officers can also define a disclosure policy adding noise to a sensitive attribute following a differential privacy approach. Future works could extend our disclosure policies to reflect more complex differential privacy policies, such as privacy-preserving join-aggregate queries [63], textual perturbations [22], and multi-dimensional range queries [67] in addition to handling temporal policies that can change over time by integrating data auditing techniques [5, 22, 66] Deshpande proposes a vision of a middleware called Sypse [17] to provide privacy-aware data management. In contrast to Mascara, it proposes to use pseudonymization, synthetic data, and data partitioning to achieve user-defined privacy goals. Exploring how to combine Sypse's vision with Mascara features could provide a richer solution for privacy-preserving query processing.

Measuring Information Loss. While our work focuses on relative entropy as an information loss metric when estimating the utility of compliant queries, other metrics, such as Avg. Relative Error [34], Kendall-Tau correlation [29], or the total mean squared error [38] could also be potentially used. We, however, note that none of these metrics can be used as-is when masking functions change the attribute's domain. Developing utility estimation frameworks supporting these metrics is an interesting direction for future work.

9 CONCLUSION

Data masking and access control mechanisms are pivotal in the evolving landscape of data regulations. In this paper, we proposed the problem of disclosure-compliant query answering. We proposed data masks as an intuitive way to specify disclosure policies that allow flexibility in masking information attributes. We showed how access control mechanisms can be extended to support data masking requirements. We presented a query modification algorithm that considers query semantics and properties of masking functions to translate a user query into a disclosure-compliant one. We also proposed a theoretical framework for estimating the information loss from data masking and approximate methods to select the optimal disclosure-compliant query. We have implemented the above techniques in a prototypical system called Mascara, which can be used as an access-control middleware atop existing relational database systems. Our experimental evaluations showed that Mascara is effective in supporting disclosure policies and is efficient in translating user queries into disclosure-compliant ones.

ACKNOWLEDGMENTS

We gratefully acknowledge funding from the German Federal Ministry for Education and Research as BIFOLD—Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A and ref. 01IS18037A).

REFERENCES

[1] 2018. California Consumer Privacy Act (CCPA). https://www.caprivacy.org/

[2] 2018. General Data Protection Regulation (GDPR). https://gdpr-info.eu/

[3] 2018. General Data Protection Regulation (GDPR): Recital 26. https://gdpr-info.eu/recitals/no-26/

[4] Foto N Afrati. 2011. Determinacy and query rewriting for conjunctive queries and views. *Theoretical Computer Science* 412, 11 (2011), 1005–1021.

[5] Rakesh Agrawal, Roberto Bayardo, Christos Faloutsos, Jerry Kiernan, Ralf Rantzau, and Ramakrishnan Srikant. 2004. Auditing compliance with a Hippocratic database. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30* (Toronto, Canada) *(VLDB '04)*. VLDB Endowment, 516–527.

[6] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. 2002. Hippocratic databases. In *Proceedings of the 28th International Conference on Very Large Data Bases* (Hong Kong, China) *(VLDB '02)*. VLDB Endowment, 143–154.

[7] Amazon Redshift. 2024. *Amazon Redshift: Dynamic data masking.* Retrieved April 5, 2024 from https://docs.aws.amazon.com/redshift/latest/dg/t_ddm.html

[8] Kaustubh Beedkar, David Brekardin, Jorge-Anulfo Quiané-Ruiz, and Volker Markl. 2021. Compliant geo-distributed data processing in action. *Proc. VLDB Endow.* 14, 12 (July 2021), 2843–2846. https://doi.org/10.14778/3476311.3476359

[9] Kaustubh Beedkar, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. 2022. Navigating Compliance with Data Transfers in Federated Data Processing. *IEEE Data Eng. Bull.* 45, 1 (2022), 50–61.

[10] Kaustubh Beedkar, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. 2021. Compliant Geo-distributed Query Processing. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 181–193. https://doi.org/10.1145/3448016.3453687

[11] Edmon Begoli, Jesús Camacho-Rodríguez, Julian Hyde, Michael J. Mior, and Daniel Lemire. 2018. Apache Calcite: A Foundational Framework for Optimized Query Processing Over Heterogeneous Data Sources. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 221–230. https://doi.org/10.1145/3183713.3190662

[12] Ji-Won Byun and Ninghui Li. 2008. Purpose based access control for privacy protection in relational database systems. *The VLDB Journal* 17, 4 (July 2008), 603–619. https://doi.org/10.1007/s00778-006-0023-0

[13] Walter Cai, Magdalena Balazinska, and Dan Suciu. 2019. Pessimistic Cardinality Estimation: Tighter Upper Bounds for Intermediate Join Cardinalities. In *Proceedings of the 2019 International Conference on Management of Data* (Amsterdam, Netherlands) *(SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 18–35. https://doi.org/10.1145/3299869.3319894

[14] Rose Clancy, Dominic O'Sullivan, and Ken Bruton. 2021. Data-driven quality improvement approach to reducing waste in manufacturing. *The TQM Journal* 35, 1 (2021), 51–72. https://doi.org/10.1108/TQM-02-2021-0061

[15] Pietro Colombo and Elena Ferrari. 2015. Privacy Aware Access Control for Big Data: A Research Roadmap. *Big Data Research* 2, 4 (2015), 145–154. https://doi.org/10.1016/j.bdr.2015.08.001

[16] Dorothy E. Denning, Selim G. Akl, Matthew Morgenstern, Peter G Neumann, Roger R. Schell, and Mark Heckman. 1986. Views for Multilevel Database Security. (1986), 156–156. https://doi.org/10.1109/SP.1986.10012

[17] Amol Deshpande. 2021. Sypse: Privacy-first Data Management through Pseudonymization and Partitioning. In *11th Conference on Innovative Data Systems Research, CIDR 2021, Virtual Event, January 11-15, 2021, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2021/papers/cidr2021_paper24.pdf

[18] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. 2021. Retiring Adult: New Datasets for Fair Machine Learning. (2021), 6478–6490. https://proceedings.neurips.cc/paper/2021/hash/32e54441e6382a7fbacbbbaf3c450059-Abstract.html

[19] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 4052)*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer, 1–12. https://doi.org/10.1007/11787006_1

[20] Mohamed Y Eltabakh, Jalaja Padma, Yasin N Silva, Pei He, Walid G Aref, and Elisa Bertino. 2012. Query processing with K-anonymity. *International Journal of Data Engineering (IJDE)* 3, 2 (2012), 48–65.

[21] David F. Ferraiolo, Ravi S. Sandhu, Serban I. Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. 2001. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.* 4, 3 (2001), 224–274. https://doi.org/10.1145/501978.501980

[22] Oluwaseyi Feyisetan, Borja Balle, Thomas Drake, and Tom Diethe. 2020. Privacy- and Utility-Preserving Textual Analysis via Calibrated Multivariate Perturbations. In *Proceedings of the 13th International Conference on Web Search and Data Mining* (Houston, TX, USA) *(WSDM '20)*. Association for Computing Machinery, New York, NY, USA, 178–186. https://doi.org/10.1145/3336191.3371856

[23] Federal Institute for Drugs and Medical Devices (BfArM). 2023. International Statistical Classification of Diseases - German Modification (ICD-10-GM). https://www.bfarm.de/EN/Code-systems/Classifications/ICD/ICD-10-GM/_node.

html
[24] Tomasz Gogacz and Jerzy Marcinkowski. 2015. The Hunt for a Red Spider: Conjunctive Query Determinacy Is Undecidable. In *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (LICS '15)*. IEEE Computer Society, USA, 281–292. https://doi.org/10.1109/LICS.2015.35
[25] Tomasz Gogacz and Jerzy Marcinkowski. 2016. Red Spider Meets a Rainworm: Conjunctive Query Finite Determinacy Is Undecidable. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (San Francisco, California, USA) *(PODS '16)*. Association for Computing Machinery, New York, NY, USA, 121–134. https://doi.org/10.1145/2902251.2902288
[26] Alon Y. Halevy. 2000. Theory of answering queries using views. *SIGMOD Rec.* 29, 4 (Dec. 2000), 40–47. https://doi.org/10.1145/369275.369284
[27] Alon Y. Halevy. 2001. Answering queries using views: A survey. *The VLDB Journal* 10, 4 (Dec. 2001), 270–294. https://doi.org/10.1007/s007780100054
[28] Yuxing Han, Ziniu Wu, Peizhi Wu, Rong Zhu, Jingyi Yang, Liang Wei Tan, Kai Zeng, Gao Cong, Yanzhao Qin, Andreas Pfadler, Zhengping Qian, Jingren Zhou, Jiangneng Li, and Bin Cui. 2021. Cardinality estimation in DBMS: a comprehensive benchmark evaluation. *Proc. VLDB Endow.* 15, 4 (Dec. 2021), 752–765. https://doi.org/10.14778/3503585.3503586
[29] Michael Hay, Liudmila Elagina, and Gerome Miklau. 2017. Differentially Private Rank Aggregation. In *Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, Texas, USA, April 27-29, 2017*, Nitesh V. Chawla and Wei Wang (Eds.). SIAM, 669–677. https://doi.org/10.1137/1.9781611974973.75
[30] Michael Hill and Dan Swinhoe. 2021. The 15 biggest data breaches of the 21st century. https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html
[31] IBM Corporation. 2024. *DB2 Version 12.1: User Guide.* Retrieved April 5, 2024 from https://www.ibm.com/docs/en/db2
[32] JMH 2023. *Java Microbenchmark Harness (JMH).* Retrieved April 5, 2024 from https://github.com/openjdk/jmh
[33] George Konstantinidis, Jet Holt, and Adriane Chapman. 2021. Enabling personal consent in databases. *Proc. VLDB Endow.* 15, 2 (Oct. 2021), 375–387. https://doi.org/10.14778/3489496.3489516
[34] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2019. PrivateSQL: A Differentially Private SQL Query Engine. 12 (2019), 1371–1384. Issue 11. https://doi.org/10.14778/3342263.3342274
[35] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.
[36] Dalibo Labs. [n. d.]. Anonymization Data Masking for PostgreSQL. https://postgresql-anonymizer.readthedocs.io/en/stable/
[37] Viktor Leis, Bernhard Radke, Andrey Gubichev, Alfons Kemper, and Thomas Neumann. 2017. Cardinality Estimation Done Right: Index-Based Join Sampling. In *8th Biennial Conference on Innovative Data Systems Research, CIDR 2017, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2017/papers/p9-leis-cidr17.pdf
[38] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. 2010. Optimizing linear counting queries under differential privacy. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (Indianapolis, Indiana, USA) *(PODS '10)*. Association for Computing Machinery, New York, NY, USA, 123–134. https://doi.org/10.1145/1807085.1807104
[39] Han Li, Hicham Johra, Flavia de Andrade Pereira, Tianzhen Hong, Jérôme Le Dréau, Anthony Maturo, Mingjun Wei, Yapan Liu, Ali Saberi-Derakhtenjani, Zoltan Nagy, Anna Marszal-Pomianowska, Donal Finn, Shohei Miyata, Kathryn Kaspar, Kingsley Nweye, Zheng O'Neill, Fabiano Pallonetto, and Bing Dong. 2023. Data-driven key performance indicators and datasets for building energy flexibility: A review and perspectives. *Applied Energy* 343 (2023), 121217. https://doi.org/10.1016/j.apenergy.2023.121217
[40] Frank McSherry and Kunal Talwar. 2007. Mechanism Design via Differential Privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. 94–103. https://doi.org/10.1109/FOCS.2007.66
[41] Microsoft. [n. d.]. Dynamic Data Masking. https://learn.microsoft.com/en-us/azure/azure-sql/database/dynamic-data-masking-overview?view=azuresql
[42] Amihai Motro. 1989. An Access Authorization Model for Relational Databases Based on Algebraic Manipulation of View Definitions. In *Proceedings of the Fifth International Conference on Data Engineering*. IEEE Computer Society, USA, 339–347.
[43] Alan Nash, Luc Segoufin, and Victor Vianu. 2010. Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst.* 35, 3, Article 21 (July 2010), 41 pages. https://doi.org/10.1145/1806907.1806913
[44] Parimarjan Negi, Ziniu Wu, Andreas Kipf, Nesime Tatbul, Ryan Marcus, Sam Madden, Tim Kraska, and Mohammad Alizadeh. 2023. Robust Query Driven Cardinality Estimation under Changing Workloads. *Proc. VLDB Endow.* 16, 6 (Feb. 2023), 1520–1533. https://doi.org/10.14778/3583140.3583164

[45] Oracle. [n. d.]. Oracle Data Masking and Subsetting. https://www.oracle.com/security/database-security/data-masking/
[46] Nisha Panwar, Shantanu Sharma, Guoxi Wang, Sharad Mehrotra, Nalini Venkatasubramanian, Mamadou H. Diallo, and Ardalan Amiri Sani. 2021. IoT Notary: Attestable Sensor Data Capture in IoT Environments. *ACM Trans. Internet Things* 3, 1, Article 3 (Oct. 2021), 30 pages. https://doi.org/10.1145/3478290
[47] Primal Pappachan, Roberto Yus, Sharad Mehrotra, and Johann-Christoph Freytag. 2020. Sieve: a middleware approach to scalable access control for database management systems. *Proc. VLDB Endow.* 13, 12 (July 2020), 2424–2437. https://doi.org/10.14778/3407790.3407835
[48] Primal Pappachan, Shufan Zhang, Xi He, and Sharad Mehrotra. 2022. Don't be a tattle-tale: preventing leakages through data dependencies on access control protected data. *Proc. VLDB Endow.* 15, 11 (July 2022), 2437–2449. https://doi.org/10.14778/3551793.3551805
[49] Daniel Pasailă. 2011. Conjunctive queries determinacy and rewriting. In *Proceedings of the 14th International Conference on Database Theory* (Uppsala, Sweden) *(ICDT '11)*. Association for Computing Machinery, New York, NY, USA, 220–231. https://doi.org/10.1145/1938551.1938580
[50] Roberta Pastorino, Corrado De Vito, Giuseppe Migliara, Katrin Glocker, Ilona Binenbaum, Walter Ricciardi, and Stefania Boccia. 2019. Benefits and challenges of Big Data in healthcare: an overview of the European initiatives. *European journal of public health* 29, Supplement_3 (2019), 23–27.
[51] PostgreSQL 2023. *PL/pgSQL — SQL Procedural Language*. Retrieved April 5, 2024 from https://www.postgresql.org/docs/current/plpgsql.html
[52] Fabian Prasser, Johanna Eicher, Helmut Spengler, Raffael Bild, and Klaus A Kuhn. 2020. Flexible data anonymization using ARX—Current status and challenges ahead. *Software: Practice and Experience* 50, 7 (2020), 1277–1304.
[53] MEDIA Protocol. 2018. What The Facebook/Cambridge Analytica Scandal Means To The Digital Content Ecosystem. https://medium.com/@mediaprotocolsm/what-the-facebook-cambridge-analytica-scandal-means-to-the-digital-content-ecosystem-7ddcba19761
[54] Shariq Rizvi, Alberto Mendelzon, S. Sudarshan, and Prasan Roy. 2004. Extending query rewriting techniques for fine-grained access control. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data* (Paris, France) *(SIGMOD '04)*. Association for Computing Machinery, New York, NY, USA, 551–562. https://doi.org/10.1145/1007568.1007631
[55] R.S. Sandhu and P. Samarati. 1994. Access control: principle and practice. *IEEE Communications Magazine* 32, 9 (1994), 40–48. https://doi.org/10.1109/35.312842
[56] Ravi S Sandhu. 1998. *Role-based access control*. Vol. 46. Elsevier. 237–286 pages.
[57] Richard Shay, Uri Blumenthal, Vijay Gadepally, Ariel Hamlin, John Darby Mitchell, and Robert K. Cunningham. 2019. Don't Even Ask: Database Access Control through Query Control. *SIGMOD Rec.* 47, 3 (Feb. 2019), 17–22. https://doi.org/10.1145/3316416.3316420
[58] Samiksha Shukla, Kritica Bisht, Kapil Tiwari, and Shahid Bashir. 2023. Comparative study of the global data economy. In *Data economy in the digital age*. Springer, 63–86.
[59] Daniel J Solove and Danielle Keats Citron. 2017. Risk and anxiety: A theory of data-breach harms. *Tex. L. Rev.* 96 (2017), 737.
[60] Michael Stonebraker and Lawrence A. Rowe. 1986. The design of POSTGRES. *SIGMOD Rec.* 15, 2 (June 1986), 340–355. https://doi.org/10.1145/16856.16888
[61] Transaction Processing Performance Council. 2017. *TPC Benchmark H (Decision Support) Standard Specification*. Technical Report. Transaction Processing Performance Council. http://www.tpc.org/tpch/ Accessed: 2024-04-16.
[62] Prasang Upadhyaya, Magdalena Balazinska, and Dan Suciu. 2015. Automatic Enforcement of Data Use Policies with DataLawyer. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (Melbourne, Victoria, Australia) *(SIGMOD '15)*. Association for Computing Machinery, New York, NY, USA, 213–225. https://doi.org/10.1145/2723372.2723721
[63] Yilei Wang and Ke Yi. 2021. Secure Yannakakis: Join-Aggregate Queries over Private Data. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 1969–1981. https://doi.org/10.1145/3448016.3452808
[64] Yinhai Wang and Ziqiang Zeng. 2018. *Data-driven solutions to transportation problems*. Elsevier.
[65] Ziniu Wu, Parimarjan Negi, Mohammad Alizadeh, Tim Kraska, and Samuel Madden. 2023. FactorJoin: A New Cardinality Estimation Framework for Join Queries. *Proc. ACM Manag. Data* 1, 1, Article 41 (May 2023), 27 pages. https://doi.org/10.1145/3588721
[66] Siyuan Xia, Zhiru Zhu, Chris Zhu, Jinjin Zhao, Kyle Chard, Aaron J. Elmore, Ian Foster, Michael Franklin, Sanjay Krishnan, and Raul Castro Fernandez. 2022. Data station: delegated, trustworthy, and auditable computation to enable data-sharing consortia with a data escrow. *Proc. VLDB Endow.* 15, 11 (July 2022), 3172–3185. https://doi.org/10.14778/3551793.3551861

[67] Jianyu Yang, Tianhao Wang, Ninghui Li, Xiang Cheng, and Sen Su. 2020. Answering multi-dimensional range queries under local differential privacy. *Proc. VLDB Endow.* 14, 3 (Nov. 2020), 378–390. https://doi.org/10.14778/3430915.3430927

[68] Rong Zhu, Ziniu Wu, Yuxing Han, Kai Zeng, Andreas Pfadler, Zhengping Qian, Jingren Zhou, and Bin Cui. 2021. FLAT: fast, lightweight and accurate method for cardinality estimation. *Proc. VLDB Endow.* 14, 9 (May 2021), 1489–1502. https://doi.org/10.14778/3461535.3461539

Received April 2024; revised July 2024; accepted August 2024