Sensitivity of Counting Queries

Sensitivity of Counting Queries

Myrto Arapinis[1], Diego Figueira[2], and Marco Gaboardi[3]

1 **University of Edinburgh, UK**
marapini@inf.ed.ac.uk
2 **CNRS, LaBRI, France**
dfigueir@labri.fr
3 **University at Buffalo, SUNY, USA**
gaboardi@buffalo.edu

Abstract

In the context of statistical databases, the release of accurate statistical information about the collected data often puts at risk the privacy of the individual contributors. The goal of differential privacy is to maximise the utility of a query while protecting the individual records in the database. A natural way to achieve differential privacy is to add statistical noise to the result of the query. In this context, a mechanism for releasing statistical information is thus a trade-off between utility and privacy. In order to balance these two "conflicting" requirements, privacy preserving mechanisms calibrate the added noise to the so-called *sensitivity* of the query, and thus a precise estimate of the sensitivity of the query is necessary to determine the amplitude of the noise to be added.

In this paper, we initiate a systematic study of sensitivity of counting queries over relational databases. We first observe that the sensitivity of a Relational Algebra query with counting is not computable in general, and that while the sensitivity of Conjunctive Queries with counting is computable, it becomes unbounded as soon as the query includes a join. We then consider restricted classes of databases (databases with constraints), and study the problem of computing the sensitivity of a query given such constraints. We are able to establish bounds on the sensitivity of counting conjunctive queries over constrained databases. The kind of constraints studied here are: functional dependencies and cardinality dependencies. The latter is a natural generalisation of functional dependencies that allows us to provide tight bounds on the sensitivity of counting conjunctive queries.

1998 ACM Subject Classification F.4.1 Mathematical Logic: Model theory; H.2.3: Languages: query languages

Keywords and phrases Differential privacy, sensitivity, relational algebra

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.XXX

1 Introduction

With the emergence of new systems and services such as eHealth, electronic tickets (*e.g.*, London Oyster card), mobile phones, or social networks, important amounts of information concerning our everyday activities are collected in various databases. Statistical analysis of such datasets could be very useful for improving services, or enabling research and market studies for example. But at the same time, the collection and storage of all this data puts at risk our individual privacy. A solution to address this problem is not to release the *exact result* of any query on a sensitive dataset, but rather to perturb the released results by adding some noise. Differential privacy [3, 6] precisely characterises the level of privacy provided by such randomized mechanisms. It offers a worst-case statistical guarantee on the increase in harm that an individual can be exposed to, if deciding to contribute her data to the dataset.

The concept of differential privacy is rooted in the notion of *neighboring databases*, that is, databases that differ in the presence or not of the information regarding one participant. More precisely, a mechanism $\mathcal{M}$ is ε-differentially private, for $\varepsilon \geq 0$ if for any two neighboring databases D and D' and for any subset $S \subseteq \mathcal{R}$ of possible outputs we have:

$$\Pr[\mathcal{M}(D) \in S] \quad \leq \quad \mathbf{e}^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in S].$$

That is, the probability that $\mathcal{M}$ releases an element of S on D is almost the same as the probability that $\mathcal{M}$ releases an element of S on D'. In the definition of differential privacy the parameter ε plays a central role. It gives the concrete bound on the increase in *harm* that an individual I can be exposed to, by contributing her data to the database.

Several mechanisms have been proposed to turn a deterministic query into a differentially private one, like the Laplace mechanism, the exponential mechanism, the Gaussian mechanism, etc. An extended introduction to these and other mechanisms (and more generally to differential privacy) is the recent monograph by Dwork and Roth [7]. In order to provide a good balance between privacy and utility, such ε-differential private mechanisms calibrate the added noise to the so-called *sensitivity* of the query. The sensitivity of a query Q captures the influence that an individual's data can have on the output of the query. More precisely, let us denote by $D \sim D'$ the fact that two databases D and D' are *neighbors*. The sensitivity of a numeric query Q is then

$$\max_{D \sim D'} |Q(D) - Q(D')|.$$

This measure is generally referred to as the *global sensitivity* of the query to distinguish it from other notions of local or smooth sensitivity [13].

To avoid adding too much noise and thus sacrificing too much utility to achieve the intended level of differential privacy, the sensitivity of the query needs to be computed as accurately as possible. However, this problem is undecidable in general as we shall see. In this paper we propose algorithms for computing upper bounds on the sensitivity of queries. Our results hold in a rather general setting: we consider counting conjunctive queries over multi-table databases. Further, our results are not tied to any particular neighboring relation, but hold for any relation of *bounded order*. This work is a first step towards understanding the class of queries and neighboring relations that are amenable to differential privacy.

Relational databases. Most of the works on differential privacy assume the simplified situation where the database is a monolithic table [7]. However, real life databases consist of not one, but many tables containing the information scattered. Of course, one could build a unique table from all these tables, by simply producing the cartesian product of all the tables in the database. Nevertheless, this immediately raises two problems. First, materialising the cartesian product of many—possibly big—tables is impractical, and often plain impossible due to space and time requirements. Second, the notion of neighboring databases now becomes *unbounded* which makes queries have unbounded sensitivity, and thus not amenable to differential privacy mechanisms. For example, given two tables (T_1, T_2) and a neighbor $T_1' = T_1 \setminus \{\bar{t}\}$ of T_1 for some record $\bar{t} \in T_1$, we have that, whereas (T_1', T_2) is the neighbor of (T_1, T_2) resulting from removing *one* record, $T_1 \times T_2$ differs from $T_1' \times T_2$ in a number of records equal to $|T_2|$. This in general makes it impossible for non-trivial queries to have bounded sensitivity, unless further restrictions on the databases are assumed.

Neighboring relation. Most works on differential privacy define neighboring databases as those that differ in exactly one record. This corresponds to assuming that each individual contributes at most one record in the database. However, as pointed out in [10] this

assumption does not hold for many applications such as social networks or tabular data. So the definition of neighboring databases needs to be tailored to the application at hand with privacy in mind. Indeed, neighboring databases should, strictly speaking, differ in the complete set of information pertaining to one individual, which could mean more than one record. Alternative definitions of neighboring have been proposed [10, 5]. In particular, our results are not tied to any particular definition of the neighboring relation.

SQL. SQL is arguably the prevalent query language for relational databases. It is equivalent to first order logic (FO) over relational structures and to Relational Algebra (RA). Here, we focus on SQL with aggregation, and study the static analysis problem of computing the sensitivity of SQL queries. As a first step in the larger programme of studying aggregate queries, we study the *counting* operator. We concentrate our investigation on one of the most prominent fragments of SQL, namely the Conjunctive Queries, corresponding to positive "select-from-where" queries [1].

Contributions. We first establish, in Section 2, that finding the sensitivity of a SQL query with counting is not computable in general. In the remaining sections we restrict our study to counting Conjunctive Queries. Section 3 shows that the sensitivity for this fragment is computable, although the characterisation shows that sensitivity becomes unbounded as soon as we have a 'non-trivial' join.

Now, in most scenarios the class of databases of interest for the application at hand are restricted (or constrained), and oftentimes the sensitivity of a query Q restricted to a constrained class of databases can become bounded. Following this idea, we then study the problem of computing global sensitivity restricted to databases from a constrained class. In Section 4, we focus on *Functional Dependencies* (FD), that allow constraining databases by rules of the form "in the table T, the i-th column determines the j-th column", in other words, "there are no 2 rows of T with the same datum in the i-th column but distinct data in their j-th columns". Further, in Section 5 we study *Cardinality Dependencies*, which are a generalisation of FDs, with rules of the form "there are no more than k rows of T with the same datum in the i-th column but pairwise distinct data in their j-th columns". Finally, Section 6 concludes and discusses future work.

Related work. Several works have studied methods for computing the sensitivity of a given query or program. The work most related to ours is the one of Palamidessi and Stronati [14]. They study the problem of computing the sensitivity of queries in relational algebra. Their approach is based on the use of constraints on attributes: every attribute comes with a bounded range, e.g. $0 \leq \text{age} \leq 100$. They are able to provide tight bounds on the sensitivity of the query Q. This approach can be applied to general SQL queries but it has the drawback that it requires to constrain the ranges for all the attributes. In this paper, instead, we focus on counting queries and on more lax semantic restrictions, namely functional dependencies and cardinality dependencies.

Pierce and Reed [15] and Gaboardi et al. [8] use relational algebra operations with a fixed, predetermined sensitivity, and a linear type system to track the use of the data in programs. This combination permits to have sensitivity analyses that extend, beyond SQL, to a full functional programming language. Their approach can provide "bad" estimates on the sensitivity of given queries due to the use of fixed sensitivity for relational operations. Our approach could provide a kernel query language providing more precise estimates that could be then combined with their type systems.

Chaudhuri et al. in [4] study automatic program analyses that provide bounds on the sensitivity of numerical imperative programs. Their approach is not directly related to specific query languages but our work could, in principle, be combined with their techniques

to design a general purpose programming language for differential privacy.

Several works have pointed out and studied the problem of providing a bound to the sensitivity of queries in disconnected structures. McSherry [12], in the setting of tabular data, considers a restricted form of join where the data of the two tables are grouped by their join keys, and then groups are joined using their group keys. The same solution has been used also in [15, 8]. A similar approach, with different restrictions, has also been used by Palamidessi and Stronati in [14]. This approach limits the situations where differential privacy can be used with a good utility. To overcome this problem, several approaches considered alternative notions of sensitivity such as *local sensitivity* [13] or *empirical sensitivity* [5].

2 Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \dots\}$ and let $\underline{n} = \{1, \dots n\}$ for every $n \in \mathbb{N}$. We write $\bar{a}$ to denote a vector of elements, whose i-th element is denoted by $\bar{a}[i]$. We write A^* [resp. A^+, A^n] for the set of strings [resp. non-empty strings, length-n strings] over A, and ε for the empty string.

2.1 Relational structures

A **relational vocabulary** $\sigma = (\mathcal{K}, \mathcal{R})$ consists of a collection $\mathcal{K}$ of **constants** (usually denoted by $c_1, c_2, \dots$), and a collection $\mathcal{R}$ of **relation symbols**, each with a specified **arity**. By σ_n we denote a vocabulary $\sigma_n = (\mathcal{K}, \mathcal{R})$ where $\mathcal{K} = \{c_1, \dots, c_n\}$. For a relation R we write $\mathit{arity}(R) \in \mathbb{N}$ to denote its arity; and we sometimes write $R^{(r)}$ to specify that R has arity r. A **σ-structure** $\mathbb{A}$ consists of a universe A containing $\mathcal{K}$, or **domain**, and an **interpretation** which associates to each relation symbol $R \in \mathcal{R}$, a relation $R^{\mathbb{A}} \subseteq A^{\mathit{arity}(R)}$, and for each constant $c \in \mathcal{K}$, $c^{\mathbb{A}} = c$. An **isolated element** of $\mathbb{A}$ is an element $a \in A$ which does not appear in any interpretation. Let STR be the set of all finite structures (we write $\mathit{STR}[\sigma]$ to make explicit the vocabulary). We use $\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{A}', \mathbb{B}', \dots$ to denote relational structures from STR, and $A, B, C, A', B', \dots$ to denote their respective domains.

▶ **Example 1.** As our running example, we will consider a database of patients, doctors and hospitals, with tables

- *Hos(id, loc)*, containing the hospitals with their location,
- *Pat(id, sex, hos)*, listing the patients with an identifier, gender and the hospital where they are being treated,
- *Doc(id, specialty, hos)*, listing the doctors with their identifier, their specialty and the hospital where they practice,
- *PatDoc(pat, doc)*, containing the patients and their current attending doctor.

Such a database can be described over the vocabulary $\sigma = (\mathcal{K}, \mathcal{R})$ containing relations $\mathcal{R} = \{\mathit{Hos}^{(2)}, \mathit{Pat}^{(3)}, \mathit{Doc}^{(3)}, \mathit{PatDoc}^{(2)}\}$ and some constants such as $\mathcal{K} = \{c_{\mathrm{F}}, c_{\mathrm{O}}\}$.

A **graph** is a structure $G = (V, E)$, where E is a binary relation that is symmetric and irreflexive. Thus, our graphs are undirected, loopless, and without parallel edges. The **Gaifman graph** of a σ-structure $\mathbb{A}$, denoted by $\mathcal{G}(\mathbb{A})$, is the (undirected) graph whose set of nodes is the universe of $\mathbb{A}$, and whose set of edges consists of all pairs (a, a') of distinct elements of A such that a and a' appear together in some tuple of a relation in $\mathbb{A}$. Recall that the **distance** between two vertices u, v of a graph is the length of the shortest path from u to v. We define the distance between two elements a, b of a structure $\mathbb{A}$ as their distance in $\mathcal{G}(\mathbb{A})$, which we denote by $\mathit{dist}_{\mathbb{A}}(a, b)$. We write $A \sqcup B$ for the disjoint union of A and B.

A **homomorphism** from a $(\mathcal{K}, \mathcal{R})$-structure $\mathbb{A}$ to a $(\mathcal{K}', \mathcal{R}')$-structure $\mathbb{B}$ such that $\mathcal{K} \subseteq \mathcal{K}'$ and $\mathcal{R} \subseteq \mathcal{R}'$ is a mapping $h : A \to B$ so that for each relation symbol $R \in \mathcal{R}$, if $(a_1, \dots, a_r) \in$

$R^{\mathbb{A}}$, then $(h(a_1), \ldots, h(a_r)) \in R^{\mathbb{B}}$, and for every constant $c \in \mathcal{K}$, $h(c) = c$. We will sometimes write $h(a_1, \ldots, a_r)$ as short for $(h(a_1), \ldots, h(a_r))$. We write $\mathbb{A} \to \mathbb{B}$ to denote that there is a homomorphism from $\mathbb{A}$ to $\mathbb{B}$, and we write $h : \mathbb{A} \to \mathbb{B}$ to denote that h is a homomorphism from $\mathbb{A}$ to $\mathbb{B}$. If $\mathbb{A} \to \mathbb{B}$ and $\mathbb{B} \to \mathbb{A}$ we say that $\mathbb{A}$ and $\mathbb{B}$ are **hom-equivalent**. We use $\cong$ for the isomorphism relation. Given a σ-structure $\mathbb{A}$ and a set $B \subseteq A$ there is (up to isomorphism) a unique structure $\mathbb{A}'$ so that

- it is hom-equivalent to $\mathbb{A}$, that is, there are $h : \mathbb{A} \to \mathbb{A}'$ and $h' : \mathbb{A}' \to \mathbb{A}$,
- $h(a) = h'(a) = a$ for all $a \in B$,
- it has the minimal number of elements.

Such a structure $\mathbb{A}'$ is called the **core preserving** B (or simply **core** if $B = \emptyset$). We write $\mathit{core}(\mathbb{A}, B)$ [resp. $\mathit{core}(\mathbb{A})$] to denote the core of $\mathbb{A}$ preserving B [resp. the core of $\mathbb{A}$].

2.2 Logic

Let $\mathcal{V}$ be a collection of first-order variables equipped with a linear order $<$. Let σ be a relational vocabulary. A **term** is either a first order variable $x \in \mathcal{V}$ or a constant from σ. The **atomic formulas** of σ are those of the form $R(t_1, \ldots, t_n)$, where $R \in \sigma$ is a relation symbol of arity r, and $t_1, \ldots, t_r$ are terms. Formulas of the form $t = t'$ are also atomic formulas, and we refer to them as **equalities**. The collection of **first-order formulas** (FO formulas) is obtained by closing the atomic formulas under negation, conjunction, disjunction, universal and existential first-order quantification. The semantics of first-order logic is standard. The set of variables of φ is denoted by $\mathit{var}(\varphi)$, and the set of free variables by $\mathit{free}(\varphi)$. We often write $\varphi(x_1, \ldots, x_n)$ where $\{x_1, \ldots, x_n\} = \mathit{free}(\varphi)$ and $x_1 < \cdots < x_n$, to stress the free variables. If $\mathbb{A}$ is a σ-structure and $\varphi(\bar{x})$ is a first-order formula, we use the notation $\mathbb{A} \models \varphi[\bar{a}]$ to denote the fact that φ is true in $\mathbb{A}$ when its free variables $\bar{x}$ are interpreted by the tuple of elements $\bar{a}$. When φ contains no free variables, we say that it is a **sentence**, and in this case we simply write $\mathbb{A} \models \varphi$. For any formula $\varphi(x_1, \ldots, x_n)$ and structure $\mathbb{A}$, we write $\varphi(\mathbb{A})$ to denote $\{(a_1, \ldots, a_n) \in A^n \mid \mathbb{A} \models \varphi[a_1, \ldots, a_n]\}$. We use '()' to denote the 0-ary tuple of elements. Hence, if φ has no free variables we interpret $\varphi(\mathbb{A})$ as $\{()\}$ if $\mathbb{A} \models \varphi$ or $\emptyset$ otherwise. Note that, in this case, $|\varphi(\mathbb{A})| = 1$ iff $\mathbb{A} \models \varphi$. We use $\equiv$ for the logical equivalence relation and $\equiv_{\mathcal{C}}$ for the equivalence relation restricted to a class of structures $\mathcal{C}$.

Given a class of FO formulas $\mathcal{L}$, by $\mathcal{L}^{\#}$ we denote the class of **counting queries** $\{\#\varphi \mid \varphi \in \mathcal{L}\}$. The evaluation of $\#\varphi$ in $\mathbb{A}$, denoted $\#\varphi(\mathbb{A})$, is defined as $|\varphi(\mathbb{A})|$, that is, as the number of distinct tuples making φ true in $\mathbb{A}$.

▶ **Example 2.** Continuing our running example, we consider the query that counts the number of oncology doctors that are treating female patients in the same hospital as they practice:

```
SELECT count distinct Doc.id
FROM Pat, Doc, PatDoc
WHERE Doc.specialty = 'O' and
      Pat.sex = 'F' and
      Pat.hos = Doc.hos and
      PatDoc.pat = Pat.id and
      PatDoc.doc = Doc.id
```

This can be equivalently expressed with the formula $\#\varphi$, where

$$\varphi(x_{doc}) = \exists x_{pat}, x_{hos} \,.\, Doc(x_{doc}, c_{\mathrm{O}}, x_{hos}) \wedge Pat(x_{pat}, c_{\mathrm{F}}, x_{hos}) \wedge PatDoc(x_{pat}, x_{doc})$$

2.3 Global sensitivity

In its standard formulation, Differential Privacy requires the privacy bound to be valid for every pair of structures that differ in one record. However, it is possible that an individual contributes more than a single record to the database. Further it may be that the database contains tables with public information. For this reason we do not set for our study a particular neighboring relation. Our results hold for any **neighboring relation** $\mathcal{N} \subseteq STR[\sigma] \times STR[\sigma]$.

Having said that, a specific neighboring relation, called **1-neighboring**, will be particularly useful for our proofs. Given two σ-structures $\mathbb{A}, \mathbb{B}$ with $\sigma = (\mathcal{K}, \mathcal{R})$, we say that $\mathbb{A}$ is a **substructure** of $\mathbb{B}$ (noted $\mathbb{A} \subseteq \mathbb{B}$) if $A \subseteq B$, and $R^{\mathbb{A}} \subseteq R^{\mathbb{B}}$ for all $R \in \sigma$. We write $\mathbb{A} \prec \mathbb{A}'$ if $\mathbb{A} \subsetneq \mathbb{A}'$ and there is no $\mathbb{B}$ so that $\mathbb{A} \subsetneq \mathbb{B} \subsetneq \mathbb{A}'$. We say that $\mathbb{A}, \mathbb{B}$ are 1-neighboring structures, noted $\mathbb{A} \sim_1 \mathbb{B}$, if $\mathbb{A} \prec \mathbb{B}$ or $\mathbb{B} \prec \mathbb{A}$. In other words, $\mathbb{A} \sim_1 \mathbb{B}$ if $\mathbb{A}$ can be obtained from $\mathbb{B}$ (and B from A) by removing/adding a tuple or an isolated node.

We say that the neighboring relation $\mathcal{N}$ is **of order** $k \in \mathbb{N}$, if any two neighboring relational structures differ in at most k elements. More formally, $\mathcal{N}$ is of order k if for any $(\mathbb{A}, \mathbb{B}) \in \mathcal{N}$, there exist $\mathbb{A}_0, \ldots, \mathbb{A}_\ell$ such that $\ell \leq k$, $\mathbb{A} = \mathbb{A}_0$, $\mathbb{B} = \mathbb{A}_\ell$ and $\mathbb{A}_{i-1} \sim_1 \mathbb{A}_i$ for all $i \in \underline{\ell}$. We say that the neighboring relation is unbounded if no such k exists.

The **global sensitivity** of a function $f : STR \to \mathbb{N}$ over a class of models $\mathcal{C} \subseteq STR$ with respect to a neighboring relation $\mathcal{N} \subseteq \mathcal{C} \times \mathcal{C}$ is:

$$\mathrm{GS}_{\mathcal{C}}^{\mathcal{N}}(f) \stackrel{def}{=} \max_{(\mathbb{A},\mathbb{A}') \in \mathcal{N}} |f(\mathbb{A}) - f(\mathbb{A}')|.$$

▶ **Example 3.** Suppose now that we want to find out the number of oncological patients in the state of New York with the query

$$\varphi(x_{pat}) = \exists x_{hos}, x_{doc}, x_{sex} \ .$$
$$Doc(x_{doc}, c_{\mathrm{O}}, x_{hos}) \wedge Pat(x_{pat}, x_{sex}, x_{hos}) \wedge PatDoc(x_{pat}, x_{doc}) \wedge Hos(x_{hos}, x_{loc})$$

It is not hard to see that this query has unbounded global sensitivity when all relations are considered sensitive, and thus all databases that differ in any one element are neighbors. Indeed changing the location of a hospital from Indiana to New-York can increase the number of ontological patients in the state of New York by any number.

▶ Observation 1. For any neighboring relation $\mathcal{N}$ of order k and any class of databases $\mathcal{C}$, the global sensitivity of a query Q is bounded with respect to $\mathcal{N}$ over $\mathcal{C}$ iff it is bounded with respect to $\sim_1$ over $\mathcal{C}$. Further, the global sensitivity with respect to $\mathcal{N}$ and relative to the class $\mathcal{C}$ is bounded by $k \cdot \mathrm{GS}_{\mathcal{C}}^{\sim_1}(Q)$. So in the remaining of the paper we focus on 1-neighboring.

We will study the following problem, given a query language $\mathcal{L}$, and a class of relational structures $\mathcal{C}$

Problem:	GlobalSensitivity($\mathcal{L}$, $\mathcal{C}$)
Input:	$Q \in \mathcal{L}$
Output:	$\mathrm{GS}_{\mathcal{C}}^{\sim_1}(Q)$

Unfortunately, this problem is undecidable already for counting first-order logic (and therefore for counting Relational Algebra [1]).

▶ **Theorem 4.** GlobalSensitivity*($FO^{\#}$, STR) is non-computable.*

The fact that the global sensitivity problem for FO is undecidable is not really surprising since most static analysis problems for FO on unrestricted structures are undecidable. This is why in the next sections we will focus on Conjunctive Queries.

3 Conjunctive queries

One of the most studied fragments of FO in relation to database queries is the fragment of *Conjunctive Queries* (CQ). We now, and for the rest of the paper, restrict our study to counting conjunctive queries, and show that sensitivity for this fragment is computable.

The class of Conjunctive Queries (also known as Primitive Positive Logic, or Existential Positive FO) is the fragment of FO corresponding to positive '*select-project-join*' queries of the Relational Algebra or to positive '*select-from-where*' queries of SQL, where by 'positive' we mean that there are no inequalities in the *select* [resp. *where*] conditions (we refer the reader to [1, §4] for more details). These are formulae of the form

$$\varphi(x_1, \ldots, x_n) = \exists y_1, \ldots, y_m\, \theta, \tag{†}$$

where θ is a conjunction of atomic formulae. Since we deal with constants, and, in future sections, with constrained databases, a conjunctive query can also be *false* (noted $\bot$). However, all the results that we show will assume that the input formula is not equivalent to $\bot$ (*i.e.*, that it is satisfiable, which can be checked in polynomial time)—for the particular case where formulae are unsatisfiable all the results are trivial, and this will avoid lengthy statements. For simplicity, we assume that the formulae do not contain equalities.

Every conjunctive query of the form (†) over a relational vocabulary σ_k gives rise to a **canonical structure** (sometimes called *tableau*) $\mathbb{C}_\varphi$ with $n + m + k$ elements, where the elements of $\mathbb{C}_\varphi$ are the variables $x_1, \ldots, x_n, y_1, \ldots, y_m$ plus the constants $c_1, \ldots, c_k$, the relations of $\mathbb{C}_\varphi$ consist of the tuples of terms in the conjuncts of θ. Given a CQ φ, we write $\mathbb{C}_\varphi$ for the canonical structure of φ, and C_φ for its domain (*i.e.*, the variables $x_1, \ldots, x_n, y_1, \ldots, y_m$ and constants $c_1, \ldots, c_k$). We also define $\mathbb{C}_\varphi^-$ as the result of removing all isolated constants from $\mathbb{C}_\varphi$ (note that $\mathbb{C}_\varphi^-$ may not necessarily be a structure over the same vocabulary of φ due to the absence of some constants). Likewise, any σ_k-structure $\mathbb{A}$ with domain $A = \{x_1, \ldots, x_n\} \cup \{c_1, \ldots, c_k\}$ gives rise to a canonical CQ $\varphi(x_1, \ldots, x_n)$ where $var(\varphi) = free(\varphi) = \{x_1, \ldots, x_n\}$, and φ has a conjunct $R(\bar{t})$ iff $\bar{t} \in R^{\mathbb{A}}$. Note that for every σ_k-structure $\mathbb{A}$ there is $\mathbb{A}' \cong \mathbb{A}$ and φ so that φ is the canonical query of $\mathbb{A}'$.

A CQ φ is **acyclic** if $\mathcal{G}(\mathbb{C}_\varphi)$ is acyclic. We say that a CQ φ is **connected** if $\mathcal{G}(\mathbb{C}_\varphi^-)$ is connected, otherwise it is **disconnected**. Note that every disconnected CQ φ so that $\mathcal{G}(\mathbb{C}_\varphi)$ has n connected components can be equivalently written in the form $\varphi = \bigwedge_{i \in \underline{n}} \psi_i(\bar{x}_i)$ so that $\psi_i(\bar{x}_i)$ is a connected CQ for every i, and for all $i \neq j$, $\bar{x}_i$ and $\bar{x}_j$ have no variables in common. We say that ψ_i is a **connected conjunct** of φ, and we say that ψ_i is a **sentential connected conjunct** if it is a sentence (*i.e.*, $\bar{x}_i = ()$). Given $\varphi = \bigwedge_{i \in \underline{n}} \psi_i(\bar{x}_i)$ a disconnected CQ with each ψ_i being a connected conjunct, we further define $\overline{\varphi}^j$ as the conjunction of all the ψ_s's but ψ_j.

▶ **Example 5** (Cont. from Ex. 2)**.** The canonical σ-structure $\mathbb{C}_\varphi$ has universe $\{x_{pat}, x_{hos}, x_{doc}, c_{\mathrm{O}}, c_{\mathrm{F}}\}$ and relations (shown in Figure 1):

$$Doc^{\mathbb{C}_\varphi} = \{(x_{doc}, c_{\mathrm{O}}, x_{hos})\}, \quad Pat^{\mathbb{C}_\varphi} = \{(x_{pat}, c_{\mathrm{F}}, x_{hos})\}, \quad PatDoc^{\mathbb{C}_\varphi} = \{(x_{pat}, x_{doc})\}.$$

Core of CQ's. For a CQ query $\varphi(\bar{x}) = \exists \bar{y}.\theta$ over σ_k we define $core(\varphi)$ as the CQ query $\varphi'(\bar{x}) = \exists \bar{y}'.\theta'$ where θ' is the canonical query of $core(\mathbb{C}_\varphi, \bar{x})$ and $\bar{y}'$ is the set of all non-constant elements of $core(\mathbb{C}_\varphi, \bar{x})$ that are not in $\bar{x}$. Note that $\mathbb{C}_{core(\varphi)} \cong core(\mathbb{C}_\varphi, \bar{x})$. We say that $\varphi(\bar{x})$ is a **core-CQ** if $\mathbb{C}_{core(\varphi)} \cong core(\mathbb{C}_\varphi)$, and we write CQ_{core} for the class of all core-CQ's. We define $core(\#\varphi)$ as $\#core(\varphi)$ for every CQ φ.

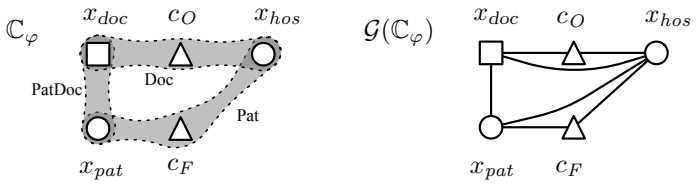

Figure 1 Depiction of the canonical structure of φ as defined in Example 2 as well as its Gaifman graph. Square vertices denote free variables and triangle vertices denote constants.

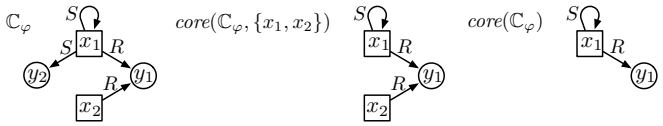

Figure 2 Core of CQ's.

▶ **Example 6.** Given $\varphi(x_1, x_2) = \exists y_1, y_2.S(x_1, x_1) \wedge S(x_1, y_2) \wedge R(x_1, y_1) \wedge R(x_2, y_1)$, whose canonical structure is depicted in Figure 2, we have that $core(\varphi) \equiv \exists y_1.S(x_1, x_1) \wedge R(x_1, y_1) \wedge R(x_2, y_1)$, and that φ is not a core-CQ since $core(\mathbb{C}_\varphi, \{x_1, x_2\})$ is not isomorphic to $core(\mathbb{C}_\varphi)$, as shown in the figure below.

Given a connected CQ φ, let us define

$$\Delta_{STR}(\#\varphi) = \begin{cases} \infty & \text{if } \exists x \in free(\varphi).\ \exists R \in \mathcal{R}.\ \exists \bar{a} \in R^{core(\varphi)}.\ x \notin \bar{a} \\ 1 & \text{otherwise} \end{cases}$$

▶ Proposition 1. For every connected CQ$^\#$ Q, we have $\mathrm{GS}^{\sim_1}_{STR}(Q) = \Delta_{STR}(Q)$.

▶ **Example 7** (Cont. from Ex. 5)**.** Note that we have $\Delta_{STR}(\#\varphi) = \infty$ since $core(\varphi) = \varphi$ and x_{doc} is not in the tuple (x_{pat}, c_F, x_{hos}) of the relation $Pat^{\mathbb{C}_\varphi}$, and thus that $\mathrm{GS}^{\sim_1}_{STR}(Q) = \infty$.

We extend the definition of Δ_{STR} to disconnected CQ$^\#$ as follows. For any $\varphi = \bigwedge_{i \in \underline{n}} \varphi_i$ disconnected CQ so that each φ_i is a connected conjunct, we define

$$\Delta_{STR}(\#\varphi) = \begin{cases} \Delta_{STR}(\#\varphi_k) & \text{if } \exists k \in \underline{n}.\ free(\varphi) = free(\varphi_k) \ \wedge\ \mathbb{C}_{\overline{\varphi}^k} \to \mathbb{C}_{\varphi_k} \\ \infty & \text{otherwise} \end{cases}$$

▶ **Theorem 8.** *For every CQ$^\#$ Q, we have $GS^{\sim_1}_{STR}(Q) = \Delta_{STR}(Q)$.*

The above characterization shows that, even when we deal with *connected* CQ's (arguably the most common), we obtain unbounded sensitivity very easily. Indeed, as soon as one has a 'join' with a free variable which is not the joining attribute, such as $\#\varphi(x) = \#\,\exists y, z\,.\,R(x, y) \wedge S(y, z)$ the global sensitivity is unbounded. Although this means that for every $N \in \mathbb{N}$ there are structures $\mathbb{A} \sim_1 \mathbb{A}'$ so that $\#\varphi(\mathbb{A}) - \#\varphi(\mathbb{A}') > N$, it may be that $\mathbb{A}, \mathbb{A}'$ do not correspond to databases that could arise in the domain of application at hand. However, when restricting the set of considered structures to ones satisfying some constraints, it may well be that the sensitivity becomes bounded. The next two sections will focus on evaluating sensitivity of queries over constrained structures.

4 Functional Dependencies

In this section we show bounds for the sensitivity of queries in the presence of what are called *functional dependencies*. In databases, it is often the case that a set of attributes

determines another attribute. Such constraints are called functional dependencies. In this section functional dependencies where one attribute determine another are considered.

▶ **Example 9** (Cont. from Ex. 2)**.** Note that, the global sensitivity of $\#\varphi$ is unbounded. Indeed, this is a consequence of the possibility of having patients with unbounded number of attending doctors and doctors working in any number of hospitals. However, this does not correspond to databases that could occur in practice, since patients have normally one attending doctor and doctors work in at most one hospital. This is why the use of database constraints becomes useful, to restrict the collection of databases we are interested in, and thus to improve the bounds of the sensitivity of queries.

We write $R[i \to j]$ to denote a **functional dependency of a relation** R **of arity** n **between components** $i \in \underline{n}$ **and** $j \in \underline{n}$. A structure $\mathbb{A}$ satisfies a functional dependency (henceforth "FD") $R[i \to j]$ if $\max_{a \in A}(|\{\bar{b}[j] \mid \bar{b} \in R^{\mathbb{A}}, \bar{b}[i] = a\}|) \leq 1$. We use the symbol Σ to denote a set of FDs, and we write $\#_\Sigma R[i \to j]$ to denote 1 if $R[i \to j] \in \Sigma$, or ∞ otherwise. We write $\mathcal{C}_\Sigma$ for the class of all relational structures satisfying all FDs in Σ.

Given a CQ query φ and a set of FDs Σ we define the **Σ-chase** [11, 2] of φ, noted $chase_\Sigma(\varphi)$, as the closure of the application of the following rule:

- For every $R[i \to j] \in \Sigma$ and every pair of conjuncts $R(\bar{t})$ and $R(\bar{s})$ of φ so that $\bar{t}[i] = \bar{s}[i]$ and $\bar{t}[j] \neq \bar{s}[j]$,
  - if $\bar{s}[j]$ is a variable, replace every occurrence of $\bar{s}[j]$ with $\bar{t}[j]$;
  - if $\bar{s}[j]$ and $\bar{t}[j]$ are constants, output $\bot$.

It can be seen that the application of these rules is terminating and Church-Rosser confluent, up to renaming of variables [1].

The following result shows that, as soon as we have a disconnected query, the sensitivity is likely to be unbounded.

▶ Proposition 2. For every disconnected CQ query φ containing a conjunct without constants and at least one free variable, and for every set Σ of FD's, we have $\mathrm{GS}^{\sim_1}_{\mathcal{C}_\Sigma}(\#\varphi) = \infty$.

Paths. A **path** of a $(\mathcal{K}, \mathcal{R})$-structure $\mathbb{A}$ between an element $a \in A$ and $b \in A$, is a string

$$p = (R_1, i_1, a_1, j_1, b_1) \cdots (R_n, i_n, a_n, j_n, b_n) \in (\mathcal{R} \times \mathbb{N} \times A \times \mathbb{N} \times A)^* \qquad (\star)$$

so that either $p = \varepsilon$ and $a = b$ (*i.e.*, the empty path); or $a_1 = a$, $b_n = b$, $a_i = b_{i-1}$ for all $1 < i \leq n$, and for every $\ell \in \underline{n}$ we have $i_\ell, j_\ell \in \underline{arity(R_i)}$ and there is $\bar{a} \in R^{\mathbb{A}}_\ell$ so that $\bar{a}[i_\ell] = a_\ell$ and $\bar{a}[j_\ell] = b_\ell$. A path of the form $(\star)$ is **simple** if $a_i \neq b_i \neq b_j$ for all $1 \leq i < j \leq n$. Note that in particular the empty path ε is simple. We write $p : A_1 \rightsquigarrow_{\mathbb{A}} A_2$ to denote that p is a simple path of $\mathbb{A}$ from an element of $A_1 \subseteq A$ to an element of $A_2 \subseteq A$. We write $a \rightsquigarrow_{\mathbb{A}} b$, $A_1 \rightsquigarrow_{\mathbb{A}} b$, $a \rightsquigarrow_{\mathbb{A}} A_2$ as short for $\{a\} \rightsquigarrow_{\mathbb{A}} \{b\}$, $A_1 \rightsquigarrow_{\mathbb{A}} \{b\}$, $\{a\} \rightsquigarrow_{\mathbb{A}} A_2$ respectively.

▶ **Example 10.** For a structure $\mathbb{A}$ with relations $R^{\mathbb{A}} = \{(a_1, a_2, a_3), (a_1, a_4, a_6)\}$, $S^{\mathbb{A}} = \{(a_5, a_6)\}$, and $T^{\mathbb{A}} = \{(a_3, a_5, a_4)\}$, we have that $p : a_1 \rightsquigarrow_{\mathbb{A}} a_4$ for $p = (R, 1, a_1, 3, a_3)$ $(T, 1, a_3, 2, a_5)$ $(S, 1, a_5, 2, a_6)$ $(R, 3, a_6, 2, a_4)$, as depicted in Figure 3

Given a vocabulary $\sigma = (\mathcal{K}, \mathcal{R})$ and a path p of the form $(\star)$, let $m \in \underline{n}$ be the greatest index m so that $b_m \in \mathcal{K}$, or 0 otherwise. We define the **cardinality of path** p as

$$\#_\Sigma(p) \stackrel{def}{=} \prod_{m < \ell \leq n} \#_\Sigma R_\ell[i_\ell \to j_\ell] \qquad (1)$$

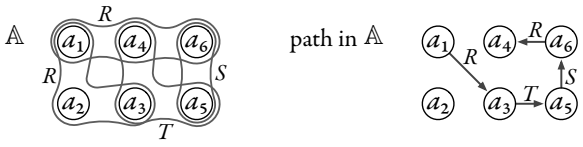

Figure 3 A path in a structure.

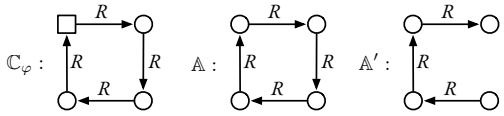

Figure 4 Structures of Example 12. Square vertices denote free variables.

where as usual the product of the empty sequence is 1, and ∞ is absorbing wrt the product ($\infty \cdot N = N \cdot \infty = \infty$). Note that $\#_\Sigma(\varepsilon) = 1$. The intuition is that $\#_\Sigma(p)$ gives a bound on how many different elements b can be reached from a through p on any structure $\mathbb{A} \in \mathcal{C}_\Sigma$ (*i.e.*, so that $p : a \rightsquigarrow_{\mathbb{A}} b$).

Let $Q = \#\psi(x_1, \ldots, x_n)$ be a connected CQ$^\#$ over a vocabulary $\sigma = (\mathcal{K}, \mathcal{R})$, and let $\varphi = \mathit{core}(\mathit{chase}_\Sigma(\psi))$. We define

$$\Delta^+_\Sigma(Q) \stackrel{def}{=} \max_{R \in \mathcal{R}} \sum_{\bar{a} \in R^{\mathbb{C}_\varphi}} \max_{i \in \underline{n}} \left(\min_{p_i : \bar{a} \rightsquigarrow_{\mathbb{C}_\varphi} x_i} \#_\Sigma(p_i) \right)$$

$$\Delta^-_\Sigma(Q) \stackrel{def}{=} \max_{R \in \mathcal{R}} \max_{\bar{a} \in R^{\mathbb{C}_\varphi}} \max_{i \in \underline{n}} \left(\min_{p_i : \bar{a} \rightsquigarrow_{\mathbb{C}_\varphi} x_i} \#_\Sigma(p_i) \right).$$

▶ Observation 2. Note that $\Delta^-_\Sigma(Q)$ is either 1 or ∞ and that $\Delta^-_\Sigma(Q) = \infty$ iff $\Delta^+_\Sigma(Q) = \infty$. Further, observe that $\Delta^+_\Sigma(Q) - \Delta^-_\Sigma(Q) \leq n_Q - 1$, where n_Q is the maximum number of elements in a relation of the canonical structure of $\mathit{core}(\mathit{chase}_\Sigma(\psi))$, assuming $Q = \#\psi$.

▶ **Theorem 11.** *Given a set Σ of functional dependencies and a connected CQ$^\#$ query Q, we have that $GS^{\sim_1}_{\mathcal{C}_\Sigma}(Q) \leq \Delta^+_\Sigma(Q)$. Further, if $Q \in \mathrm{CQ}^\#_{core}$, we have $GS^{\sim_1}_{\mathcal{C}_\Sigma}(Q) \geq \Delta^-_\Sigma(Q)$.*

▶ Observation 3. When computing lower and upper bounds for global sensitivity of query $Q = \#\psi(x_1, \ldots, x_n)$ in the presence of functional dependencies, we consider $\mathit{core}(\mathit{chase}_\Sigma(\psi))$ (rather than $\mathit{core}(\psi)$) as it gives a corresponding *canonical minimal* query. This allows us to obtain tighter bounds than if we hadn't taken the chase of ψ.

▶ **Example 12.** Take for instance the CQ with one free variable of Figure 4. Observe that, for $\Sigma = \{R[1 \to 2],\ R[2 \to 1]\}$, we have that $\mathrm{GS}^{\sim_1}_{\mathcal{C}_\Sigma}(\#\varphi) \leq \Delta^+_\Sigma(\#\varphi) = 4$, which is tight since $\#\varphi(\mathbb{A}) = 4$, and $\#\varphi(\mathbb{A}') = 0$. Further, this example can be easily generalized, obtaining that for every $n \in \mathbb{N}$ there is a CQ Q so that $\mathrm{GS}^{\sim_1}_{\mathcal{C}_\Sigma}(Q) = n = \Delta^+_\Sigma(Q)$.

▶ **Example 13** (Cont. from Ex. 2)**.** As noted in Example 9, $\#\varphi$ has unbounded global sensitivity. However, if every patient has no more than one attending doctor, the sensitivity of $\#\varphi$ becomes bounded. Indeed, if $\Sigma = \{\mathit{PatDoc}[1 \to 2]\}$, then

$$\Delta^-_\Sigma(\#\varphi) \leq \mathrm{GS}^{\sim_1}_{\mathcal{C}_\Sigma}(\#\varphi) \leq \Delta^+_\Sigma(\#\varphi)$$

by Theorem 11—observe that $\varphi \in \mathrm{CQ}_{core}$ since it is unary. Since $\Delta^-_\Sigma(\#\varphi) = \Delta^+_\Sigma(\#\varphi) = 1$, it thus follows that $\mathrm{GS}^{\sim_1}_{\mathcal{C}_\Sigma}(\#\varphi) = 1$.

As we have shown, adding functional dependencies immediately improves the global sensitivity of queries. However, functional dependencies are often very restrictive, and it may not always be possible to impose such restrictions. This leads to a more general notion of dependencies, that we call *cardinality dependencies*. These dependencies bound the number of elements associated with component i of a relation R for each fixed element of a component j. This will be the object of study of our next section.

5 Cardinality Dependencies

While functionality constraints are a very natural restriction of databases, there are many scenarios in which, although we don't have an attribute i functionally determining an attribute j in a relation, we have a **cardinality dependency** nonetheless. This is a dependency of the form "there are at most n different attributes j sharing the same attribute i in the relation R"—functional dependencies being the special case when $n = 1$.

These dependencies arise naturally when modelling relations between entities (such as in ER modelling [9]). For example, the business rules underlying a company database may allow that an employee has more than one manager, but no more than 2. Another example is for bounded domain attributes: whereas the name of a person does not determine the gender, there cannot be more than two possibilities of gender for any given name. As we will see next, cardinality dependencies provide further means to give tighter bounds for the global sensitivity of CQ's.

▶ **Example 14** (Cont. from Ex. 13)**.** We already noticed that constraining each patient to have at most one attending doctor, brings the sensitivity of $\#\varphi$ down to 1. However, it may be that a patient can have more than one attending doctor, although it can't have an *unbounded* number of attending doctors. For example, a scenario in which a patient has *at most* 3 attending doctors.

More formally, we write $R[i \xrightarrow{k} j]$ to denote a **k-cardinality dependency of a relation R of arity n between components $i \in \underline{n}$ and $j \in \underline{n}$**. A structure $\mathbb{A}$ satisfies a cardinality dependency (henceforth "CD") $R[i \xrightarrow{k} j]$ if $\max_{a \in A}(|\{\bar{b}[j] \mid \bar{b} \in R^{\mathbb{A}}, \bar{b}[i] = a\}|) \leq k$. For the particular case where $k = 1$, note that $R[i \xrightarrow{k} j]$ is a *functional* dependency. We use the symbol Σ to denote a set of CD's, and we write $\#_\Sigma R[i \to j]$ to denote the minimum k so that $R[i \xrightarrow{k} j] \in \Sigma$, or ∞ otherwise. As before, we write $\mathcal{C}_\Sigma$ for the class of all relational structures satisfying all CDs in Σ. We define the cardinality of a path $\#_\Sigma(p)$ as in (1), where now Σ is a set of CD's, and in the definition $\#_\Sigma R[i \to j]$ is interpreted as defined above, over CD's.

Upper bound. Given a connected CQ$^\#$ query Q over a vocabulary $\sigma = (\mathcal{R}, \mathcal{K})$ so that $core(Q) = \#\varphi(x_1, \ldots, x_n)$, let us define

$$\Delta^+_\Sigma(Q) \stackrel{def}{=} \max_{R \in \mathcal{R}} \left(\sum_{\bar{a} \in R^{\mathbb{C}_\varphi}} \left(\min_{\substack{p_1,\ldots,p_n \text{ s.t.} \\ p_i : \bar{a} \rightsquigarrow_{\mathbb{C}_\varphi} x_i \text{ for } i \in \underline{n}}} \left(\prod_i \#_\Sigma(p_i) \right) \right) \right).$$

▶ Observation 4. Note that in the presence of cardinality dependencies, when computing upper bounds for global sensitivity of a query $Q = \#\psi(x_1, \ldots, x_n)$, we now consider $core(\psi)$ (rather than $core(chase_\Sigma(\psi))$). This is because $core(chase_\Sigma(\psi))$ is not necessarily a conjunctive query, but rather a union of conjunctive queries which we do not handle.

▶ **Theorem 15.** *Given a set of cardinality dependencies Σ, for all connected CQ$^\#$ queries Q we have $GS^{\sim_1}_{\mathcal{C}_\Sigma}(Q) \leq \Delta^+_\Sigma(Q)$.*

▶ **Example 16** (Cont. from Ex. 14)**.** If every patient has at most 3 attending doctors, the sensitivity of $\#\varphi$ becomes bounded. Indeed, if $\Sigma = \{PatDoc[1 \xrightarrow{3} 2]\}$, then $\mathrm{GS}^{\mathcal{R}}_{\mathcal{C}_\Sigma}(\#\varphi) \leq \Delta^{+}_{\mathcal{R},\Sigma}(\#\varphi) = 3$ by Theorem 15.

6 Conclusion

We have given bounds for the global sensitivity of counting Conjunctive Queries under the functionality or cardinality constraints. These bounds can be used to turn those queries in differentially private ones by using mechanisms like the Laplacian or the Gaussian mechanisms without adding too much noise. The proposed algorithms for computing these bounds have exponential complexity, but since effectively many interesting queries are often small, our results are still practical.

There are several interesting directions that we will pursue in future work. We will study other aggregation operations already present in SQL such as *average* or *sum*. We will also investigate sensitivity of queries with negation, where one can ask for example for the number of patients that are *not* treated by a given doctor. Further, we have focused here on global sensitivity but there are other notions of sensitivity that have been proposed. In particular, the so-called *local sensitivity* is studied in [13]. The local sensitivity is defined by quantifying not over all possible databases but only over the ones in the neighborhood of the particular database under analysis. The local sensitivity is often lower than the global sensitivity, but adding noise proportional to the local sensitivity does not ensure differential privacy. Nevertheless, adding the noise proportional to a smooth approximation of the local sensitivity permits to recover differential privacy.

Acknowledgements Marco Gaboardi's work was partially supported by NSF grants CNS-1237235 and by EPSRC grant EP/M022358/1.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. URL: `http://webdam.inria.fr/Alice/`.

2 A. V. Aho, C. Beeri, and J. D. Ullman. The theory of joins in relational databases. *ACM Trans. Database Syst.*, 4(3):297–314, September 1979. URL: `http://doi.acm.org/10.1145/320083.320091`, `doi:10.1145/320083.320091`.

3 Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: The sulq framework. In *Proceedings of the Twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '05, pages 128–138, New York, NY, USA, 2005. ACM. URL: `http://doi.acm.org/10.1145/1065167.1065184`, `doi:10.1145/1065167.1065184`.

4 Swarat Chaudhuri, Sumit Gulwani, Roberto Lublinerman, and Sara Navidpour. Proving programs robust. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE '11, pages 102–112, New York, NY, USA, 2011. ACM. URL: `http://doi.acm.org/10.1145/2025113.2025131`, `doi:10.1145/2025113.2025131`.

5 Shixi Chen and Shuigeng Zhou. Recursive mechanism: Towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 653–664, New York, NY, USA, 2013. ACM. URL: `http://doi.acm.org/10.1145/2463676.2465304`, `doi:10.1145/2463676.2465304`.

6 Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 1–12, 2006. URL: http://dx.doi.org/10.1007/11787006_1, doi:10.1007/11787006_1.

7 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014. URL: http://dx.doi.org/10.1561/0400000042, doi:10.1561/0400000042.

8 Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C. Pierce. Linear dependent types for differential privacy. *SIGPLAN Not.*, 48(1):357–370, January 2013. URL: http://doi.acm.org/10.1145/2480359.2429113, doi:10.1145/2480359.2429113.

9 Sven Hartmann. On interactions of cardinality constraints, key, and functional dependencies. In *Foundations of Information and Knowledge Systems, First International Symposium, FoIKS 2000, Burg, Germany, February 14-17, 2000, Proceedings*, pages 136–155, 2000. URL: http://dx.doi.org/10.1007/3-540-46564-2_9, doi:10.1007/3-540-46564-2_9.

10 Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 193–204, New York, NY, USA, 2011. ACM. URL: http://doi.acm.org/10.1145/1989323.1989345, doi:10.1145/1989323.1989345.

11 David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, December 1979. URL: http://doi.acm.org/10.1145/320107.320115, doi:10.1145/320107.320115.

12 Frank D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 19–30, New York, NY, USA, 2009. ACM. URL: http://doi.acm.org/10.1145/1559845.1559850, doi:10.1145/1559845.1559850.

13 Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 75–84, New York, NY, USA, 2007. ACM. URL: http://doi.acm.org/10.1145/1250790.1250803, doi:10.1145/1250790.1250803.

14 Catuscia Palamidessi and Marco Stronati. Differential privacy for relational algebra: Improving the sensitivity bounds via constraint systems. In *Proceedings 10th Workshop on Quantitative Aspects of Programming Languages and Systems, QAPL 2012, Tallinn, Estonia, 31 March and 1 April 2012.*, pages 92–105, 2012. URL: http://dx.doi.org/10.4204/EPTCS.85.7, doi:10.4204/EPTCS.85.7.

15 Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger: A calculus for differential privacy. In *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming*, ICFP '10, pages 157–168, New York, NY, USA, 2010. ACM. URL: http://doi.acm.org/10.1145/1863543.1863568, doi:10.1145/1863543.1863568.