Provable data privacy

Kilian Stoffel[1] and Thomas Studer[2]

[1] Université de Neuchâtel,
Pierre-à-Mazel 7, CH-2000 Neuchâtel, Switzerland
kilian.stoffel@unine.ch
[2] Institut für Informatik und angewandte Mathematik,
Universität Bern, Neubrückstrasse 10, CH-3012 Bern, Switzerland,
tstuder@iam.unibe.ch

Abstract. In relational database systems a combination of privileges and views is employed to limit a user's access and to hide non-public data. The data privacy problem is to decide whether the views leak information about the underlying database instance. Or, to put it more formally, the question is whether there are certain answers of a database query with respect to the given view instance. In order to answer the problem of provable date privacy, we will make use of query answering techniques for data exchange. We also investigate the impact of database dependencies on the privacy problem. An example about health care statistics in Switzerland shows that we also have to consider dependencies which are inherent in the semantics of the data.

1 Introduction

Data privacy refers to the relationship between technology and the legal right to, or public expectation of privacy in the collection and sharing of data [16]. Although technology alone cannot address all privacy concerns, it is important that information systems take responsibility for the privacy of the data they manage [1]. The main challenge in data privacy is to share some data while protecting other personally identifiable information. Our aim is to formally prove that under certain circumstances none of the protected data can be logically inferred from the data which is made public.

We investigate this problem in the context of relational database systems. Assume we are given a database whose contents are supposed to be hidden and a view which is exposed to the public. We want to decide whether the view leaks information about the underlying database. Leaking of information means that the set of facts which can be inferred with certainty about the database is non-empty.

Data stored in a relational database system usually is protected from unauthorized access. Some database system users may be allowed to issue

queries only to a limited portion of the database. This can be achieved by introducing views. A view can hide data a user does not need to see. Although, we can deny a user direct access to a relation, that user may be allowed to access part of that relation through a view. Thus the combination of privileges and views can limit a user's access to precisely the data that the user needs.

Silberschatz et al. [13] present the following banking example. Consider a clerk who needs to know the names of all customers who have a loan at each branch. This clerk is not authorized to see information regarding specific loans that the customers may have. Thus, the clerk must be denied access to the `loan` relation. But, if she is to have access to the information needed, the clerk must be granted access to a view that consists of only the names of customers and the branches at which they have a loan. For this example, we want to formally prove that the clerk cannot infer any information about specific loans from the view instances to which she has access.

Let us now study a simple example which shows how it can happen that private data may be derived from a published view instance. Consider a database schema that consist of a table A with two attributes and a table P with one attribute. We define two views by the following queries

$$V_1(x) \leftarrow A(x, y) \wedge P(y) \text{ and } V_2(x) \leftarrow A(x, x).$$

Assume that we have a view instance which contains $V_1(a), V_2(a)$. This view instance does not allow us to infer that a certain element belongs to P. The fact $V_1(a)$ might stem from entries $A(a, b)$ and $P(b)$ and we do not know what b is. Hence, there is no way to say something about P except that it is non-empty. This situation completely changes if we add a unique key constraint to A. Assume that the first attribute of A is a unique key for this table. Then $V_2(a)$ implies that $\forall y.(V(a, y) \rightarrow a = y)$. Therefore, by $V_1(a)$, we get that $P(a)$ must hold in any database instance which satisfies the key constraint and which yields the view instance. We see that equality generating dependencies may give rise to unwanted inferences of private data. Hence, in our setting, we have to take into account also the constraints of the database schema.

We formally model the problem of provable data privacy in the following framework: a data privacy setting consists of a database schema $\mathcal{R}$, a set of constraints Σ_r on $\mathcal{R}$, and a set of view definitions $\mathcal{V}$ over $\mathcal{R}$. In addition, assume we are given a view instance I over $\mathcal{V}$ and a query q over $\mathcal{R}$ which is asking for non-public information. In this setting, the data privacy problem consists of the following question: is there a tuple $\boldsymbol{t}$

of constants of I such that $\boldsymbol{t} \in q(J)$ for every database instance J over $\mathcal{R}$ which satisfies the constraints in Σ_r and which yields the view instance I.

Making use of the notion of *certain answer*, we can state the question as: is there a tuple $\boldsymbol{t}$ of constants of I such that $\boldsymbol{t}$ is a certain answer of q with respect to I? That means, is there a tuple $\boldsymbol{t}$ such that $\boldsymbol{t} \in q(J)$ for every 'possible' database instance J? The problem of answering queries against a set of 'possible' databases was first encountered in the context of incomplete databases [15]. Today, certain answer is a key notion in the theory about data integration [5, 11, 12] and data exchange [8–10].

We show that the techniques employed for query answering in the area of data exchange can be applied also to the data privacy problem. In a data exchange setting, we use a standard chase to produce a so-called universal database instance. The certain answers of a query are then computed by evaluating the query on this universal instance. We can make use of the same procedure to find certain answers in the context of data privacy which solves the data privacy problem. Certain answers in the presence of key constraints are also studied in [14]. There, we independently developed a method similar to, but less general than the one presented in [9] in order to solve the data privacy problem.

As we have seen, it is important to consider dependencies in a data privacy setting. This refers not only to constraints on the database schema; we may also encounter so-called *semantic dependencies* which are not explicitly stated but which are inherent in the semantics of the data. We study the following example: the Swiss Federal Statistical Office annually creates statistics about health care in Switzerland. The published data contains semantic dependencies which give rise to equality generating dependencies in the corresponding data privacy setting.

The rest of the paper is organized as follows. In Section 2, we present a formal definition of the data privacy problem. In Section 3, we summarize some results of [9] about query answering in the context of data exchange. We apply these data exchange techniques to solve the data privacy problem in Section 4. We also investigate the impact of dependencies on the privacy problem. This is illustrated in Section 5 with an example about semantic dependencies. Finally, Section 6 concludes the paper.

2 The data privacy problem

In this section, we present the data privacy problem. We need some preliminary definitions in order to formally state the privacy problem.

Definition 1. *1. A* schema *is a finite collection* $\mathcal{R} = \{R_1, \ldots, R_k\}$ *of relation symbols.*

2. *An* instance I over $\mathcal{R}$ *is a function that associates to each relation symbol* R_i *a relation* $I(R_i)$.
3. *A* view over a schema $\mathcal{R}$ *is a query of the form*

$$V_i(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y} \phi_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y})$$

where $\phi_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y})$ *is a conjunction of atomic formulas over* $\mathcal{R}$. *We say that* V_i belongs to a set of views $\mathcal{V}$ *if* $\mathcal{V}$ *contains a view* $V_i(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y} \phi_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y})$.
4. *A* view instance *over a set of views* $\mathcal{V}$ *is a finite collection of facts of the form* $V_i(\boldsymbol{t})$ *where* V_i *belongs to* $\mathcal{V}$ *and* $\boldsymbol{t}$ *is a tuple of constants.*
5. *We say* a database instance J over $\mathcal{R}$ yields a view instance I over $\mathcal{V}$ *if for all facts* $V_i(\boldsymbol{t})$ *in* I, *such that* $V_i(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y} \phi_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y})$ *belongs to* $\mathcal{V}$, *there exists a tuple* $\boldsymbol{s}$ *of constants of* J *such that each conjunct of* $\phi_{\mathcal{R}}(\boldsymbol{t}, \boldsymbol{s})$ *is an element of* J.

Consider the following setting for the data privacy problem:

Definition 2. *A* data privacy setting $(\mathcal{R}, \Sigma_r, \mathcal{V})$ *consists of*

1. *a schema* $\mathcal{R}$,
2. *a set of dependencies* Σ_r *on* $\mathcal{R}$. *Each element of* Σ_r *is either a tuple generating dependency [2] of the form*

$$\forall \boldsymbol{x}(\phi_{\mathcal{R}}(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y} \psi_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y}))$$

or an equality generating dependency [2] of the form

$$\forall \boldsymbol{x}(\phi_{\mathcal{R}}(\boldsymbol{x}) \rightarrow (x_1 = x_2)),$$

where $\phi_{\mathcal{R}}(\boldsymbol{x})$ *and* $\psi_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y})$ *are conjunctions of atomic formulas over* $\mathcal{R}$, *and* x_1, x_2 *are among the variables of* $\boldsymbol{x}$,
3. *a set of views* $\mathcal{V}$ *over* $\mathcal{R}$.

The *data privacy problem* for this setting can be stated as: given a view instance I over $\mathcal{V}$ and a query q over $\mathcal{R}$, are there tuples $\boldsymbol{t}$ of constants such that $\boldsymbol{t} \in q(J)$ must hold for any instance J over $\mathcal{R}$ which respects Σ_r and which yields the view instance I? In that case, knowledge about $\mathcal{R}$, Σ_r, and $\mathcal{V}$ together with the data presented in I make it possible to infer non-public information. Hence, even if a user is denied to issue the query q, he can infer some of the answers of this query from the information which is presented to him in the view instance I.

To put this problem more formally, we introduce the notion of a certain answer in the context of the data privacy setting.

Definition 3. *Let $(\mathcal{R}, \Sigma_r, \mathcal{V})$ be a data privacy setting.*

- *If I is a view instance over $\mathcal{V}$, then a* possible database instance for I *is an instance J over $\mathcal{R}$ such that J respects Σ_r and J yields the view instance I.*
- *If I is a view instance over $\mathcal{V}$ and q is a query over $\mathcal{R}$, then a* certain answer of q with respect to I, *denoted by* $\mathsf{certain}_{\mathsf{p}}(q, I)$, *is the set of all tuples $\boldsymbol{t}$ of constants from I such that for every possible database instance J for I, we have $\boldsymbol{t} \in q(J)$.*
- *The data privacy problem for this setting is to compute* $\mathsf{certain}_{\mathsf{p}}(q, I)$.

Assume q is a query asking for non-public information. If $\mathsf{certain}_{\mathsf{p}}(q, I)$ is empty, that is if there are no certain answers, then it is not possible to infer any information about tuples in q from the view instance I and knowledge about the schema $\mathcal{R}$, the constraints Σ_r and the view definition $\mathcal{V}$. Conversely, if there is a tuple $\boldsymbol{t}$ in $\mathsf{certain}_{\mathsf{p}}(q, I)$, then some hidden information can be inferred.

3 Query answering for data exchange

Fagin et al. [9] study query answering in the context of data exchange. We summarize some of their definitions and results which are relevant for the data privacy problem.

Let $\mathcal{S}$ and $\mathcal{T}$ be two disjoint schemata. We refer to $\mathcal{S}$ as *source* schema and to $\mathcal{T}$ as *target* schema. If I is an instance over $\mathcal{S}$ (a *source instance*) and J is an instance over $\mathcal{T}$ (a *target instance*), then $\langle I, J\rangle$ is the corresponding instance over $\mathcal{S} \cup \mathcal{T}$.

A *source-to-target dependency* is a tuple generating dependency of the form

$$\forall \boldsymbol{x}(\phi_{\mathcal{S}}(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y} \psi_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{y})),$$

where $\phi_{\mathcal{S}}(\boldsymbol{x})$ is a conjunction of atomic formulas over $\mathcal{S}$ and $\psi_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atomic formulas over $\mathcal{T}$. A *target dependency* is a tuple generating dependency or an equality generating dependency:

$$\forall \boldsymbol{x}(\phi_{\mathcal{T}}(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y} \psi_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{y})), \qquad \forall \boldsymbol{x}(\phi_{\mathcal{T}}(\boldsymbol{x}) \rightarrow (x_1 = x_2)).$$

Here, $\phi_{\mathcal{T}}(\boldsymbol{x})$ and $\psi_{\mathcal{T}}(\boldsymbol{x}, \boldsymbol{y})$ are conjunctions of atomic formulas over $\mathcal{T}$, and x_1, x_2 are among the variables of $\boldsymbol{x}$.

Definition 4. *[9, Def. 1] A* data exchange setting $(\mathcal{S}, \mathcal{T}, \Sigma_{st}, \Sigma_t)$ *consists of a source schema $\mathcal{S}$, a target schema $\mathcal{T}$, a set Σ_{st} of source-to-target*

dependencies, and a set Σ_t of target dependencies. The data exchange problem *associated with this setting is the following: given a finite source instance I, find a finite target instance J such that $\langle I, J\rangle$ satisfies Σ_{st} and J satisfies Σ_t. Such a J is called a* solution for I.

Next, we specify the class of so-called universal solutions which have several 'good' properties. Before presenting its definition, we introduce some terminology and notation.

Let Const denote the set of all values, called *constants*, that occur in source instances. In addition, we assume an infinite set Var of values, called *labeled nulls*, such that $\mathsf{Const} \cap \mathsf{Var} = \emptyset$. If K is an instance over a schema $\mathcal{R}$ with values in $\mathsf{Const} \cup \mathsf{Var}$, then $\mathsf{Var}(K)$ denotes the set of labeled nulls occurring in relations in K.

Definition 5. *[9, Def. 2] Let K_1 and K_2 be two instances over a schema $\mathcal{R}$ with values in $\mathsf{Const} \cup \mathsf{Var}$. A* homomorphism $h : K_1 \to K_2$ *is a mapping from $\mathsf{Const} \cup \mathsf{Var}(K_1)$ to $\mathsf{Const} \cup \mathsf{Var}(K_2)$ such that*

1. *$h(c) = c$, for every $c \in \mathsf{Const}$,*
2. *for every fact $R_i(\boldsymbol{t})$ of K_1, we have that $R_i(h(\boldsymbol{t}))$ is a fact of K_2, where, if $\boldsymbol{t} = (t_1, \ldots, t_n)$, then $h(\boldsymbol{t}) = (h(t_1), \ldots, h(t_n))$.*

Definition 6. *[9, Def. 3] Let $(\mathcal{S}, \mathcal{T}, \Sigma_{st}, \Sigma_t)$ be a data exchange setting. If I is a source instance, then a* universal solution for I *is a solution J for I such that for every solution J' for I, there exists a homomorphism $h : J \to J'$.*

Fagin et al. [9] show that the classical chase can be used to produce a universal solution: start with an instance $\langle I, \emptyset\rangle$ that consists of I, for the source schema, and of the empty instance, for the target schema; then chase $\langle I, \emptyset\rangle$ by applying the dependencies in Σ_{st} and Σ_t for as long as they are applicable. This process may fail (for instance, if an attempt to identify two constants is made) or it may not terminate. However, if it does terminate and if it does not fail, then the resulting instance is guaranteed to satisfy the dependencies and to be universal.

Theorem 1. *[9, Th. 1] Let $(\mathcal{S}, \mathcal{T}, \Sigma_{st}, \Sigma_t)$ be a data exchange setting. Let $\langle I, J\rangle$ be the result of some successful finite chase of $\langle I, \emptyset\rangle$ with $\Sigma_{st} \cup \Sigma_t$. Then J is a universal solution.*

Definition 7. *[9, Def. 7] Let $(\mathcal{S}, \mathcal{T}, \Sigma_{st}, \Sigma_t)$ be a data exchange setting. Let q be a k-ary query over the target schema $\mathcal{T}$ and I a source instance. The* certain answers of q with respect to I, *denoted by $\mathsf{certain}_{\mathsf{e}}(q, I)$, is the set of all k-tuples t of constants from I such that for every solution J of this instance of the data exchange problem, we have $t \in q(J)$.*

There are situations in which the certain answers of a query q can be computed by evaluating q on a particular fixed solution and then keeping only the tuples that consist entirely of constants. If q is a query and J is a target instance, then $q(J)\downarrow$ is the set of all tuples t of constants such that $t \in q(J)$.

Theorem 2. *[9, Prop. 2] Let $(\mathcal{S}, \mathcal{T}, \Sigma_{st}, \Sigma_t)$ be a data exchange setting such that the dependencies in the sets Σ_{st} and Σ_t are arbitrary. Let q be a union of conjunctive queries over the target schema $\mathcal{T}$. If I is a source instance and J is a universal solution, then $\mathsf{certain}_{\mathsf{e}}(q, I) = q(J)\downarrow$.*

4 Proving Privacy

We can reduce the data privacy problem to the data exchange problem: starting from a data privacy setting $(\mathcal{R}, \Sigma_r, \mathcal{V})$, we create a corresponding data exchange setting $(\mathcal{S}, \mathcal{T}, \Sigma_{st}, \Sigma_t)$ such that $\mathcal{R} = \mathcal{T}$, $\Sigma_r = \Sigma_t$, and $\mathcal{V}$ defines $\mathcal{S}$ and Σ_{st}. Hence, a query q over $\mathcal{R}$ may be issued to $\mathcal{T}$. Moreover, a view instance I over $\mathcal{V}$ is a valid database instance over the schema $\mathcal{S}$. We will show that the certain answers of q with respect to I under the data privacy setting will be the same as the certain answers of q with respect to I under the corresponding data exchange setting.

Definition 8. *The data exchange setting $(\mathcal{S}, \mathcal{T}, \Sigma_{st}, \Sigma_t)$ that* corresponds *to a data privacy setting $(\mathcal{R}, \Sigma_r, \mathcal{V})$ is given by:*

- *$\mathcal{S}$ consists of all relation symbols V_i which belong to $\mathcal{V}$,*
- *$\Sigma_t := \Sigma_r$,*
- *Σ_{st} contains $V_i(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y} \phi(\boldsymbol{x}, \boldsymbol{y})$ if $V_i(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y} \phi(\boldsymbol{x}, \boldsymbol{y})$ is an element of $\mathcal{V}$,*
- *$\mathcal{T} := \mathcal{R}$.*

Obviously, if q is query over $\mathcal{R}$, then it is also a query over $\mathcal{T}$; and if I is a view instance over $\mathcal{V}$, then it is also a database instance over $\mathcal{S}$. The previous definition immediately leads to the following theorem:

Theorem 3. *Let $(\mathcal{R}, \Sigma_r, \mathcal{V})$ be a data privacy setting and $(\mathcal{S}, \mathcal{T}, \Sigma_{st}, \Sigma_t)$ its corresponding data exchange setting. Assume q is a query over $\mathcal{R}$ and I is a view instance over $\mathcal{V}$. The certain answers $\mathsf{certain}_{\mathsf{p}}(q, I)$ of q with respect to I for $(\mathcal{R}, \Sigma_r, \mathcal{V})$ are exactly the certain answers $\mathsf{certain}_{\mathsf{e}}(q, I)$ for $(\mathcal{S}, \mathcal{T}, \Sigma_{st}, \Sigma_t)$.*

This theorem, together with Theorem 2, provides a solution for the data privacy problem. Given a data privacy setting $(\mathcal{R}, \Sigma_r, \mathcal{V})$, a query q over $\mathcal{R}$, and a view instance I over $\mathcal{V}$. We can compute the certain answers of q with respect to I for $(\mathcal{R}, \Sigma_r, \mathcal{V})$ as follows:

1. compute the corresponding data exchange setting $(\mathcal{S}, \mathcal{T}, \Sigma_{st}, \Sigma_t)$,
2. construct a universal solution J for I with respect to $(\mathcal{S}, \mathcal{T}, \Sigma_{st}, \Sigma_t)$,
3. take $q(J)\downarrow$, that is evaluate q on J and keep only the tuples that consist entirely of constants.

5 Semantic dependencies

A standard way to guarantee privacy while sharing data is to publish data only in the aggregate. However, this practice does not necessarily ensure data privacy. It is still possible that there are semantic dependencies which may lead to unwanted leaking of information.

If we only have view definitions, it is easy to see what information they reveal. However, if views can interact with dependencies, then it is less obvious what amount of information one may obtain. In a privacy setting, it is important to consider as many dependencies as possible: the more dependencies there are, the more data may be inferred. A dependency can be a constraint which is defined on a database schema. For example, as we have seen in the introduction, the presence of a unique key constraint can make it possible to derive private data. However, it is possible that a dependency is not explicitly stated; maybe it is inherent in the semantics of the data.

Let us investigate the following example of a semantic dependency. The Swiss Federal Statistical Office (SFSO) annually creates statistics on health care in Switzerland [3]. The collected data includes sociodemographic information about the patients (age, sex, region of residence, etc.) as well as administrative data (kind of insurance, length of stay in hospital, disease, etc.). Naturally, these data are highly sensitive and there are strict privacy policies about their treatment. For instance, only the year of birth and the region of residence of a patient are recorded; the date of birth and the precise place of residence are considered as too detailed.

The latest results about health care are from 2002. One of the published tables [4] contains information about the length of stay in hospital. The table has a row for each disease (encoded according to ICD-10 [17]); and basically three columns (the ICD-10 code, the number of inpatients with the corresponding disease, the average length of stay). In that table, we have 7644 rows with different ICD-10 codes. 903 of them are diseases which occurred only once; the corresponding number of inpatients is 1. Let us restrict the view to these rows. Then the value of the attribute 'average length of stay' is the precise length of stay of the one patient with the corresponding disease. Hence, the length of stay depends on the

ICD-10 code and is uniquely determined by it. This dependency may be used in a data privacy setting to infer more information.

On an abstract level, the situation is as follows: consider a relation `R` with two attributes `A` and `B`. Let `V` be the view defined by the query

```
select A, count(A), avg(B) from R group by A.
```

Assume we have a view instance I over `V`. We can restrict I to an instance I' which contains only those rows where the value of `count(A)` is 1. If we use I' instead of I to compute the possible instances of `R`, then each value of the `A` attribute determines a unique value for the `B` attribute. Hence, we can add the equality generating dependency

$$\forall x, y, z(\mathtt{R}(x, y) \wedge \mathtt{R}(x, z) \rightarrow y = z)$$

to the data privacy setting. It is possible that more data may be inferred by making use of this additional dependency.

Semantic dependencies are not easy to find. In the above example, the reason for their occurrence is that the view is defined by a query with small counts [6, 7]. That means the grouping is too fine grained. There are many rows where the value of `count(A)` is 1. We can assign the average values of those rows as `B` values to the corresponding `A` values; and the `A` value uniquely determines the `B` value. It is important that we also consider this kind of semantic dependencies when we are studying data privacy settings.

6 Concluding remarks

We have defined the data privacy problem to be the problem of deciding whether information about a database instance can be inferred from a given view instance. There is a related problem, namely to decide whether already the data privacy setting guarantees that there is no possible leaking of information. That is, whether $\mathsf{certain}_{\mathsf{p}}(q, I)$ is empty for every possible view instance I. In this case, we do have data privacy, no matter what the view instance is. Privacy follows already from the setting $(\mathcal{R}, \Sigma_r, \mathcal{V})$. We call such a setting *safe with respect to* q.

There is a trivial example of a data privacy setting which ensures privacy. Let $\mathcal{R}$ be a schema with a non-public relation symbol P. Further, let $\mathcal{V}$ be a set of views over $\mathcal{V}$ such that every view V in $\mathcal{V}$ satisfies: if $P(\boldsymbol{x})$ is an atom occurring in the definition of V, then each variable of $\boldsymbol{x}$ is existentially quantified in V. Let q be the query

$$q(\boldsymbol{x}) \leftarrow P(\boldsymbol{x}).$$

Given the data privacy setting $(\mathcal{R}, \emptyset, \mathcal{V})$, we find that $\mathsf{certain}_{\mathsf{p}}(q, I)$ must be empty for every view instance I. Hence, $(\mathcal{R}, \emptyset, \mathcal{V})$ is safe with respect to q. The reason for this is that the information stored in P is hidden by the existential quantifier in the view definitions and that there are no dependencies to infer that information.

We plan to further investigate safe data privacy settings. It would be nice to have a collection of patterns for safe privacy settings. Finally, an important direction is extending the data privacy problem to more complex view definitions and other forms of dependencies.

References

1. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *Proc. of 28th VLDB Conference*, 2002.
2. C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *Journal of the ACM*, 31(4):718–741, 1984.
3. Bundesamt für Statistik. Medizinische Statistik der Krankenhäuser.
4. Bundesamt für Statistik. Beilage ICD-10/2002, 2002. Available at `http://www.bfs.admin.ch/bfs/portal/de/index/themen/gesundheit/gesundheitsversorgung/behandlungen/analysen_berichte/stand/01.html`.
5. A. Calì, D. Calvanese, G. D. Giacomo, and M. Lenzerini. Data integration under integrity constraints. In *Proc. of CAiSE 2002*, volume 2348 of *LNCS*, pages 262–279. Springer, 2002.
6. F. Y. Chin. Security in statistical databases for queries with small counts. *ACM Transactions on Database Systems*, 3(1):92–104, 1978.
7. L. Cox. Suppresion methodology and statistical disclosure control. *J. Am. Stat. Assoc.*, 75:377–395, 1980.
8. R. Fagin, P. G. Kolaitis, R. Miller, and L. Popa. Data exchange: Semantics and query answering. To appear in *Theoretical Computer Science*.
9. R. Fagin, P. G. Kolaitis, R. Miller, and L. Popa. Data exchange: Semantics and query answering. In *Proc. of ICDT'03*, pages 207–224, 2003.
10. R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. In *ACM PODS'03*, pages 90–101, 2003.
11. A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.
12. M. Lenzerini. Data integration: a theoretical perspective. In *ACM PODS '02*, pages 233–246. ACM Press, 2002.
13. A. Silberschatz, H. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill, fourth edition, 2002.
14. K. Stoffel and T. Studer. Canonical databases and certain answers under key constraints, 2004. Technical report IAM-04-009.
15. R. van der Meyden. Logical approaches to incomplete information: a survey. In *Logics for databases and information systems*, pages 307–356. Kluwer Academic Publishers, 1998.
16. Wikipedia The Free Encyclopedia. Data privacy. Available at `http://en.wikipedia.org/wiki/Data_privacy`.
17. World Health Organization WHO. International statistical classification of diseases and related health problems. 10th Revision.