Information Control by Policy-Based Relational Weakening Templates

Joachim Biskup(✉) and Marcel Preuß(✉)

Technische Universität Dortmund, Dortmund, Germany
{joachim.biskup,marcel.preuss}@cs.tu-dortmund.de

Abstract. We conceptually design, formally verify and experimentally evaluate a sophisticated information control mechanism for a relational database instance. The mechanism reacts on access requests for data publishing or query answering with a granularity of either the whole instance or individual tuples. The reaction is based on a general read access permission for the instance combined with user-specific exceptions expressed as prohibitions regarding particular pieces of information declared in a confidentiality policy. These prohibitions are to be enforced in the sense that the user should neither be able to get those pieces directly nor by rational reasoning exploiting the interaction history and background knowledge about both the database and the control mechanism. In an initial off-line phase, the control mechanism basically determines instance-independent weakening templates for individual tuples and generates a policy-compliant weakened view on the stored instance. During the system-user interaction phase, each request to receive data of the database instance is fully accepted but redirected to the weakened view.

Keywords: Distortion · Confidentiality · Background knowledge · History-awareness · Information control · Read access · Relational database · Query access · View generation · Weakened information

1 Introduction

Early versions of access control deal with objects as containers on the layer of an *operating system*. Basically, the control intercepts any request issued by a process to read, write or execute the content of a container and then either accepts or denies the request. The decision is taken according to previously granted access rights, but without inspecting the actual content of the container. Access control primarily aims at enforcing requirements of confidentiality, integrity and availability. In this article, we focus on *confidentiality* regarding processes of a *single user* or a group of potentially colluding users. Accordingly, requests to read or, more generally, to *receive data* are our main concern.

Since early days, many refinements of access control have been proposed and have come into operation. In particular, the concepts of *granularity*, *history-awareness* and *content-sensitivity* are important for access control on the layer of a *database management system*. Going even further, managing data can be seen as the fundament of providing knowledge or some kind of belief, by assigning some well-defined meaning to raw data. Typically, such semantics are defined for the syntax of a formal logic. For example, first-order logic is employed for query answering in a relational database management system. Dealing with sophisticated notions of information – whether seen as knowledge or as belief – rather than with raw data might be even more ambitious, leading to a further layer of a *knowledge-and-belief management system*. Accordingly, access control for such a system demands for further concepts, namely of *information* control and *entailment*.

If a process running on behalf of an intelligent agent issues access requests, the results of an accepted access might be further exploited by computational rational reasoning, in order to determine the information actually gained. Roughly described, this gain is the new information inferred by reasoning about recently directly received data together with the already previously held information. Hence, the control has to confine the information content of data delivered such that any information gain by a "too curious" receiver does not comprise information to be kept confidential.

To still achieve best availability of information, the control should then be further enhanced by more sophisticated reactions on a request: rather than simply either accepting or denying a request, the control can react by a larger range of options, including the mediation of *distorted* data. However, distortions might lead to new vulnerabilities by so-called *meta-inferences*. Accordingly, on the layer of a *multi-(intelligent-)agent system*, it is necessary to also deal with adversarial reasoning including meta-inferences based on advanced background knowledge about the protection mechanism.

During this development rather straightforward access control gradually matured to highly sophisticated inference control. Unfortunately, the increase of functionality comes along with a decline of efficiency and scalability. One line of answers to this challenge is known as *confidentiality/privacy-preserving data publishing* [11], which in particular includes the technique of value generalization by k-anonymization as a special case of information weakening. In a first precomputation *offline phase*, the control system generates a sanitized view such that all concerns regarding inferences are already provably captured. In a second *system-user interaction phase*, access to the original data is completely prohibited, but full read access rights on the view are granted.

A particular instantiation of this approach applied to relational databases even goes a step further. In this instantiation [5], access rights for receiving data are expressed by the combination of (i) a *general permission* to see the tuples of a fixed database instance and (ii) exceptions in the form of *user-specific prohibitions* to acquire specific pieces of information. These forbidden pieces are expressed as queries in terms of the database schema and declaratively stated in a *confidentiality policy*. Notably, a security officer should declare such prohibitions independently of the actual instance. Given a confidentiality policy and the database instance, the control system splits the offline phase into two stages, which can be roughly rephrased as follows:

- For each forbidden piece of information listed in the policy, the system generates a suitable *weakening* by individually assigning a disjunctive and thus a less informative template to it, such that all these templates seen together with any non-distorted data are totally non-interferential regarding the information to be kept confidential.
- In any sequence inspecting each *actual* tuple of the instance in turn, the system checks whether the tuple is related to one or more of the disjunctive templates generated from the policy, and if this is the case the system replaces the tuple by the set of all pertinent templates.

Our contributions generalize and substantially extend that particular instantiation of confidentiality/privacy-preserving data publishing:

- We propose a *generic approach* consisting of first generating weakening templates from the policy and afterwards applying these templates on the instance tuple-wise, whether *dynamically* and interactively while reacting on query requests, or *statically* for defining a view.
- We design and verify a powerful method to handle *a priori knowledge*, in particular in the form of relational *data dependencies*.
- We employ a flexible scheme to declare and enforce *prohibitions*.
- We reduce the conceptual requirements to graph problems for which well-established *scalable graph algorithms* are known.

In the remainder of this article, we first introduce an example in Sect. 2. In Sect. 3, we briefly summarize basic notions and present the new generic approach. In Sect. 4, we refine the generic approach for data dependencies as a priori knowledge. Moreover, in Sect. 5, we discuss the practical efficiency on the basis of an experimental evaluation of a prototype implementation. Finally, we further relate our contributions to previous work and conclude in Sect. 6.

2 Running Example

We consider a simple relation instance r over a schema $R(A, B, C)$, so far without any data dependencies. For confining the interactions with some user, the security officer declares a confidentiality policy *ppol* with the *prohibitions* shown in Fig. 1a, formalized as sentences of first-order logic as any other items. This confidentiality requirement implies in particular that the user should neither be able to infer that the hidden instance contains any of the listed ground facts, nor should he be able to reason that the instance contains any ground fact that *entails* some of the listed existential facts. But the requirement still accepts that the user infers the validity of strict disjunctions of the listed items, as long as the user cannot strengthen such a disjunction to just one disjunct, i.e., to one of the prohibited items.

Accordingly, in the *first stage* of the weakening method, the prohibited items are suitably clustered into mutually independent groups in order to define for each of these groups a *weakening template* in the form of the *disjunction* of the

$$\begin{aligned} ppol = \{\, & R(a,a,a),\ R(a,b,a),\ R(a,b,c),\ R(a,b,d),\ R(a,b,e),\ R(a,c,a), \\ & (\exists X)\, R(a,e,X),\ (\exists X)\, R(b,e,X),\ (\exists X)\, R(c,e,X),\ (\exists X)\, R(b,X,e) \,\} \end{aligned}$$

(a) Confidentiality policy *ppol*

$$\begin{array}{l} \{\, R(a,b,c),\ R(a,b,d) \,\}, \\ \{\, (\exists X)\, R(b,X,e),\ (\exists X)\, R(b,X,d)^A \,\}, \\ \{\, (\exists X)\, R(a,e,X),\ (\exists X)\, R(a,f,X)^A \,\} \end{array}$$

(b) Groups for templates that are satisfied by the instance

$$\begin{array}{l} \{\, R(a,a,a),\ R(a,c,a) \,\}, \\ \{\, R(a,b,a),\ R(a,b,e) \,\}, \\ \{\, (\exists X)\, R(b,e,X),\ (\exists X)\, R(c,e,X) \,\} \end{array}$$

(c) Groups for templates that are *not* satisfied by the instance

$$r = \{\, (a,b,c),\ (a,f,g),\ (b,a,e),\ (b,b,d),\ (b,d,f),\ (g,e,i),\ (g,h,i) \,\}$$

(d) Original database instance r

$$\begin{array}{l} R(b,d,f) \\ R(g,e,i) \\ R(g,h,i) \\ \\ R(a,b,c) \vee R(a,b,d) \\ (\exists X)\, R(a,e,X) \vee (\exists X)\, R(a,f,X) \\ (\exists X)\, R(b,X,d) \vee (\exists X)\, R(b,X,e) \\ \\ \neg\, [\, R(a,a,a) \vee R(a,c,a) \,] \\ \neg\, [\, R(a,b,a) \vee R(a,b,e) \,] \\ \neg\, [\, (\exists X)\, R(b,e,X) \vee (\exists X)\, R(c,e,X) \,] \end{array}$$

$$\begin{array}{l} (\forall X)(\forall Y)(\forall Z)\ [\\ (\, X \equiv a \wedge Y \equiv b \wedge Z \equiv c \,) \vee \\ (\, X \equiv a \wedge Y \equiv b \wedge Z \equiv d \,) \vee \\ (\, X \equiv a \wedge Y \equiv e \,) \vee \\ (\, X \equiv a \wedge Y \equiv f \,) \vee \\ (\, X \equiv b \wedge Z \equiv d \,) \vee \\ (\, X \equiv b \wedge Z \equiv e \,) \vee \\ (\, X \equiv b \wedge Y \equiv d \wedge Z \equiv f \,) \vee \\ (\, X \equiv g \wedge Y \equiv e \wedge Z \equiv i \,) \vee \\ (\, X \equiv g \wedge Y \equiv h \wedge Z \equiv i \,) \vee \\ \neg R(X,Y,Z) \ ] \end{array}$$

(e) Confidentiality-preserving weakened view

Fig. 1. Groups for weakening templates generated from a confidentiality policy, a database instance, and the resulting confidentiality-preserving weakened view

group members. In case that the clustering leaves some items isolated, suitable further items are added, in our example $(\exists X)\, R(a,f,X)^A$ and $(\exists X)\, R(b,X,d)^A$. Figure 1b and c show the resulting groups, though at this stage the partitioning into the two parts is not relevant.

The weakening method computes that partitioning only in the *second stage*, when the stored instance r as shown in Fig. 1d is treated: one part contains the templates that are entailed by the instance; the other part contains the remaining templates. Finally, the weakening method generates the confidentiality-preserving *weakened view* that consists of three kinds of sentences, as shown in Fig. 1e (though in the presence of a priori knowledge, we might need to deal with a fourth kind of totally *refused knowledge*).

- *positive knowledge* about the instance, $R(b,d,f)$, $R(g,e,i)$ and $R(g,h,i)$;
- *disjunctive knowledge*, the templates of the first part;
- *negative knowledge*, a first sentence capturing all facts not entailing the other knowledge and further sentences capturing all templates of the second part.

3 Generic Approach

Stored Data. We consider data stored by means of a *relational database* management system, for which a single relational *schema* is declared. A schema comprises a relation *symbol* (table name) R, a finite set of *attributes* (column names) $\mathcal{A} = \{A_1, \ldots, A_n\}$, each of which has the same *infinite domain* Dom of constants, and some set SC of semantic *constraints*. In the running example we have three attributes A, B and C and, so far, an empty set of constraints.

The system maintains a database *instance* r, which is a finite set of tuples over $\mathcal{A}$ with values in Dom, satisfying the semantic constraints in SC. Intuitively, such an instance is treated as being *complete* in the following sense: each tuple in r represents a fact that is *true* in some fictitious "real world"; whereas, by Closed World Assumption (CWA), each other tuple over $\mathcal{A}$ with values in Dom represents a possible fact which is *false* in that world. Figure 1d shows an example of a database instance, leaving the CWA implicit.

We follow a foundation of the relational model of data in terms of first-order logic with equality, as also used in [3]. Syntactically, the logic is specified by a language $\mathscr{L}$ over $\equiv$, R, $\mathcal{A}$, Dom, variables, propositional connectives and first-order quantifiers in the usual way. Semantically, for this logic we treat a database tuple $(a_1, \ldots, a_n)$ as a ground fact $R(a_1, \ldots, a_n) \in \mathscr{L}$ and a database instance as a finite Herbrand interpretation of $\mathscr{L}$ with the infinite universe Dom assuming *unique names*. Using an instance in this way, we can inductively assign a truth value to each sentence in $\mathscr{L}$. This foundation also provides us with the pertinent notions of *satisfaction* and *entailment*: an instance r, seen as an Herbrand interpretation of the kind described above, *satisfies* a sentence $\Phi \in \mathscr{L}$ (r is a model of Φ, $r \models \Phi$) iff the truth evaluation according to r returns the truth value *true*; a set $\mathcal{S} \subseteq \mathscr{L}$ of sentences *entails* a sentence $\Phi \in \mathscr{L}$ ($\mathcal{S} \models \Phi$) iff each instance r satisfying $\mathcal{S}$ also satisfies Φ.

Given an instance $r = \{(a_{1,1}, \ldots, a_{1,n}), \ldots, (a_{m,1}, \ldots, a_{m,n})\}$ consisting of m tuples $(a_{j,1}, \ldots, a_{j,n})$, we can formalize our completeness assumption, specified above in natural language, by the following sentence in $\mathscr{L}$, denoted by $Comp(r)$:

$$(\forall X_1) \ldots (\forall X_n)[\bigvee_{(a_{j,1},\ldots,a_{j,n}) \in r} (\bigwedge_{i \in \{1,\ldots,n\}} X_i \equiv a_{j,i}) \vee \neg R(X_1, \ldots, X_n)].$$

The control system should be effective for any fixed instance r. But the user is assumed to have some *a priori knowledge* $prior \subseteq \mathscr{L}$ that includes the semantic constraints SC, i.e., only instances with $r \models prior$ are seen as being possible. Extending the running example in Sect. 4 below, we will consider the a priori knowledge shown in Fig. 2b.

Confidentiality Policy. Confidentiality requirements are expressed in the form of *user-specific prohibitions*. Syntactically, most generally each prohibition would just be a sentence Ψ in $\mathscr{L}$. However, facing the well-known difficulty of the computational unsolvability of the general entailment problem for the full first-order logic language $\mathscr{L}$, in this work we restrict prohibitions to sentences in the sublanguage $\mathscr{L}_{exist}$ of existential facts. A security officer is assumed to declare all prohibitions as a finite subset $ppol \subseteq \mathscr{L}_{exist}$. The prohibitions dealt within the running example are gathered in the confidentiality policy shown in Fig. 1a.

More formally, an *existential fact* is a sentence of the form $(\exists X_{i_1}) \ldots (\exists X_{i_m}) R(t_1, \ldots, t_n)$ with pairwise different variables $X_{i_1}, \ldots, X_{i_m}$ and terms $t_{i_j} = X_{i_j}$ for $i_j \in \{i_1, \ldots, i_m\} \subseteq \{1, \ldots, n\}$ and $t_i \in Dom$ otherwise. Such a sentence corresponds to a subtuple where the components for the attributes in $\{A_{i_1}, \ldots, A_{i_m}\}$ are dropped. We also see ground facts as elements of $\mathscr{L}_{exist}$. The *entailment problem* for existential facts, i.e., whether for some Ψ_1 and Ψ_2 in $\mathscr{L}_{exist}$ we have $\Psi_1 \models \Psi_2$, is known to be easily solvable by a simple *term matching*, namely if and only if the following holds: whenever Ψ_2 has a constant $a \in Dom$ for an attribute A, then Ψ_1 has the same constant for that attribute. Consequently, we have both $\Psi_1 \not\models \Psi_2$ and $\Psi_2 \not\models \Psi_1$ if and only if at least one of the following alternatives holds: Ψ_1 and Ψ_2 have different constants $a_1 \neq a_2$ on some attribute A, or Ψ_1 has a constant a_1 for some attribute A_1 but Ψ_2 has a variable there and, vice versa, Ψ_2 has a constant a_2 for a different attribute $A_2 \neq A_1$ but Ψ_1 has a variable there. Each $\Psi \in \mathscr{L}_{exist}$ determines its *sphere* (of ground facts) defined by $Sp(\Psi) := \{\Phi \,|\, \Phi \text{ is ground fact in } \mathscr{L} \text{ and } \Phi \models \Psi\}$. Obviously, we have $\Psi_1 \models \Psi_2$ if and only if $Sp(\Psi_1) \subseteq Sp(\Psi_2)$. Moreover, even if both $\Psi_1 \not\models \Psi_2$ and $\Psi_2 \not\models \Psi_1$, the spheres might be overlapping, i.e., $Sp(\Psi_1) \cap Sp(\Psi_2) \neq \emptyset$, namely if only the second alternative discussed above holds.

Semantically, a prohibition sentence $\Psi \in ppol$ intuitively requires the following: from the point of view of the user, it should always appear to be *possible* that the prohibition sentence Ψ is *not true* [12]. More formally, the *view generation* mechanism to be designed gets three inputs, namely (i) the actually stored database instance r, together with (ii) the (assumed) a priori knowledge *prior* with $r \models prior$, and (iii) a confidentiality policy *ppol* with $prior \not\models \Psi$ for each $\Psi \in ppol$. Thus in the running example the input consists of the database instance shown in Fig. 1d, the empty a priori knowledge, and the confidentiality policy shown in Fig. 1a.

Given the inputs, the mechanism should return a consistent weakened view $v(r, prior, ppol)$ on r such that for all prohibition sentences $\Psi \in ppol$ there exists an alternative instance r^{Ψ} that

1. satisfies the a priori knowledge *prior*, i.e., $r^{\Psi} \models prior$,
2. does not satisfy Ψ, i.e., $r^{\Psi} \not\models \Psi$, and
3. generates the same weakened view, i.e., $v(r, prior, ppol) = v(r^{\Psi}, prior, ppol)$.

Since the view $v(r, prior, ppol)$ will be both a joint weakening of all these alternative instances and consistent, it should not entail any prohibition sentence $\Psi \in ppol$. In particular, this implies that for all $\Phi \in Sp(\Psi)$ we should have $v(r, prior, ppol) \not\models \Phi$. Notably, in general the latter property is only *necessary* for achieving our strong notion of *semantic confidentiality*, but it is *not sufficient* to guarantee the third property of *indistinguishability*.

Weakened Views. We aim at designing a control mechanism that applied to any possible instance r generates a sanitized *view* by *weakening* the information content of individual tuples as far as needed to preserve confidentiality. Such a view will again be formally specified in terms of the first-order logic language $\mathscr{L}$, in particular employing the sublanguage $\mathscr{L}^{\vee}_{exist}$ of strict and non-redundant *disjunctions* over $\mathscr{L}_{exist}$, i.e., all sentences of the form $\Psi_1 \vee \Psi_2 \vee \ldots \vee \Psi_k$ such that $k \geq 2$, $\Psi_i \in \mathscr{L}_{exist}$ and $\Psi_i \not\models \Psi_j$ for $i \neq j$.

As far as needed for confidentiality, a tuple/ground fact $R(a_1, \ldots, a_n)$ in the stored instance is disjunctively *weakened* by replacing it in a *context-free* way by a disjunction $\Psi_1 \vee \Psi_2 \vee \ldots \vee \Psi_k$ taken from a predefined finite set of *templates* $\mathcal{T} \subset \mathscr{L}^{\vee}_{exist}$ such that $R(a_1, \ldots, a_n) \models \Psi_1 \vee \ldots \vee \Psi_k$. In fact, in order to conveniently capture many simultaneous threats to confidentiality, the replacement is performed with *all* such disjunctions. To avoid unnecessary distortions, the disjunctions should only be formed by prohibitions of the confidentiality policy. Moreover, all disjunctions for all tuples seen together should be *mutually independent* in the following sense: for each two different disjunctions $\Psi_1 \vee \Psi_2 \vee \ldots \vee \Psi_k$ and $\bar{\Psi}_1 \vee \bar{\Psi}_2 \vee \ldots \vee \bar{\Psi}_{\bar{k}}$ we have $\Psi_i \not\models \bar{\Psi}_j$ and $\bar{\Psi}_j \not\models \Psi_i$.

Lines 1 to 6 of the left side of Fig. 1e indicate that the last three tuples of the example database instance remain undistorted, whereas the first four tuples are replaced by suitable disjunctions. Each of the distorted tuples entails a prohibition sentence and is thus replaced by the disjunction of the pertinent group for templates shown in Fig. 1b.

All replacements have to be reflected in a corresponding *partial* completeness assertion. Now, a tuple/ground fact Φ is treated as *false* if at least one of the following two properties holds: (i) Φ does neither entail an unreplaced tuple/ground fact nor an existential fact occurring in the weakening disjunctions; (ii) Φ does entail an existential fact occurring in $\mathcal{T}$ but not in the weakening disjunctions. If $\mathcal{G}$ is the set of unreplaced ground facts and $\mathcal{R} \subseteq \mathcal{T}$ is the set of weakening disjunctions used for replacements, then the completeness assertion is expressed by the following two sentences in $\mathscr{L}$, denoted by $Comp(\mathcal{G}, \mathcal{R}, \mathcal{T})$:

$$\begin{aligned}
&(\forall X_1)\ldots(\forall X_n)[\ \textstyle\bigvee_{R(a_1,\ldots,a_n)\in\mathcal{G}}\ (\bigwedge_{i\in\{1,\ldots,n\}} X_i \equiv a_i)\\
&\quad\vee\ \textstyle\bigvee_{(\exists X_{i_1})\ldots(\exists X_{i_m})R(t_1,\ldots,t_n)\text{ occurs in }\mathcal{R}}\ (\bigwedge_{i\in\{1,\ldots,n\}\text{ with }t_i\in Dom} X_i \equiv t_i)\\
&\quad\vee\ \neg R(X_1,\ldots,X_n)\,],\\
&(\forall X_1)\ldots(\forall X_n)[\ \textstyle\bigvee_{(\exists X_{i_1})\ldots(\exists X_{i_m})R(t_1,\ldots,t_n)\text{ occurs in }\mathcal{T}\text{ but not in }\mathcal{R}}\\
&(\textstyle\bigwedge_{i\in\{1,\ldots,n\}\text{ with }t_i\in Dom} X_i \equiv t_i)\quad\Rightarrow\quad \neg R(X_1,\ldots,X_n)\,].
\end{aligned}$$

Lines 7 to 9 of the left side of Fig. 1e show an equivalent reformulation of the second completeness assertion, and the right side of Fig. 1e exemplifies the first completeness assertion.

For some syntactically possible tuples the exact status of being either *true* or *false* deliberately remains unknown. Instead, the status is only determined up to the specified entailment relationships to disjunctions in $\mathcal{T}$, and in this sense the view might become only *partially complete*. Moreover, we will have to ensure that a weakened view is *consistent* even under consideration of the semantic constraints SC and possibly further *a priori knowledge.*

Two-Stage Weakening Method for View Generation. To achieve the goal of employing weakened views to enforce the confidentiality requirements of the prohibition sentences by means of context-free replacements of ground facts by weakening disjunctions, we propose the following two-stage *weakening method.* Given a database instance r, the a priori knowledge $prior$, and a confidentiality policy $ppol$ such that $prior \not\models \Psi$ for all $\Psi \in ppol$, a weakened view $v(r, prior, ppol)$ is created as follows:

Stage 1 (independent of r) **Safe Templates**
Determine a finite set $\mathcal{T} \subset \mathscr{L}^{\vee}_{exist}$ with the following properties:

Property 1. $\mathcal{T}$ *covers* $ppol$, i.e., for each prohibition sentence $\Psi \in ppol$ there is a template $\tau \in \mathcal{T}$ such that $\Psi \models \tau$.
Property 2. The templates in $\mathcal{T}$ are *independent*, i.e., for each two different elements $\Psi_1 \vee \Psi_2 \vee \ldots \vee \Psi_k$ and $\bar{\Psi}_1 \vee \bar{\Psi}_2 \vee \ldots \vee \bar{\Psi}_{\bar{k}}$ of $\mathcal{T}$ we have $\Psi_i \not\models \bar{\Psi}_j$ and $\bar{\Psi}_j \not\models \Psi_i$.
Property 3. $\mathcal{T}$ is *non-interferential* under $prior$, i.e., for each finite set $\mathcal{G}$ of ground facts Φ such that $\Phi \not\models \tau$ for all $\tau \in \mathcal{T}$, for each finite set $\mathcal{R} \subseteq \mathcal{T}$ such that $\mathcal{R} = \{\tau \mid \tau \in \mathcal{T}$ and there exists $\Phi \in \mathcal{D} : \Phi \models \tau\}$ for some set $\mathcal{D}$ of ground facts, and for each $\Psi \in \mathscr{L}_{exist}$ occurring in $\mathcal{T}$, we have $\mathcal{G} \cup \mathcal{R} \cup \{Comp(\mathcal{G}, \mathcal{R}, \mathcal{T})\} \cup prior \not\models \Psi$.

Stage 2 (dependent on r) **Weakened View**
Define and (to block any information gain from the syntactic appearances) suitably normalize the following outputs:

1. *positive knowledge:*
   $v(r, prior, ppol)^+ := \{\Phi \mid \Phi \in r$ and for all $\tau \in \mathcal{T} : \Phi \not\models \tau\}$;
2. *disjunctive knowledge:*
   $v(r, prior, ppol)^\vee := \{\tau \mid \tau \in \mathcal{T}$ and there exists $\Phi \in r : \Phi \models \tau\}$;
3. *negative knowledge:*
   $v(r, prior, ppol)^- := Comp(\, v(r, prior, ppol)^+, v(r, prior, ppol)^\vee, \mathcal{T}\,)$.

Quite obviously, the task of achieving the non-interferential Property 3 of $\mathcal{T}$ is the only conceptually difficult one. However, Stage 1 can be executed as a precomputation without even having an actual instance so far. In many applications we expect the costs to be affordable, at least under some reasonable restrictions. This claim will be further treated in the remaining sections.

Regarding the running example, still not considering a priori knowledge, in Stage 1 the safe templates are determined by the groups shown in Fig. 1b and c, which are straightforwardly formed by putting together two prohibitions that differ in exactly one attribute with constants. As explained before, the weakened view of our running example generated in Stage 2 is shown in Fig. 1e as follows: lines 1 to 3 of the left side form the positive knowledge; lines 4 to 6 of the left side comprise the disjunctive knowledge; and lines 7 to 9 of the left side together with the right side yield the negative knowledge.

Total Refusals. In some cases, it is impossible to achieve the wanted weakening of information *only* by means of weakening disjunctions. Intuitively, such an unfortunate event can be caused by a prohibition sentence Ψ that in some sense is conflicting with the a priori knowledge such that *every candidate* for a covering template is *not* safe, i.e., including it into the set $\mathcal{T}$ to be determined would violate the non-interferential property. We escape from this seemingly hopeless situation by complementing the set of templates $\mathcal{T}$ with a set $\mathcal{C} \subset \mathscr{L}_{exist}$ of such *conflicting* prohibition sentences and by adapting the generic approach accordingly, as sketched in the following. In Stage 1, we now require that

1. *ppol* is *covered* by $\mathcal{T} \cup \mathcal{C}$,
2. the *independence* property also applies for $\mathcal{C}$, and
3. the *non-interferential* property is adapted by (i) considering sets $\mathcal{G}$ of ground facts that additionally do not entail any prohibition sentence in $\mathcal{C}$, (ii) modifying the definition of $\mathcal{R}$ accordingly, and (iii) inserting the clauses corresponding to $\mathcal{C}$ into the first completeness sentence (thus *excluding* the ground facts in their spheres from known to be *not true*).

And in Stage 2, we generate an additional output $v(r, prior, ppol)^{?} := \mathcal{C}$ representing *refused knowledge*, meaning that any nontrivial information about the truth value of a ground fact in the sphere of an element of $\mathcal{C}$ is totally refused. Accordingly, we (i) strengthen the positive knowledge into $v(r, prior, ppol)^{+?}$ by additionally requiring that no prohibition sentence in $\mathcal{C}$ is entailed, (ii) change the disjunctive knowledge into $v(r, prior, ppol)^{\vee ?}$ by insisting that only those $\Phi \in r$ are replaced that do not entail a prohibition sentence in $\mathcal{C}$, and (iii) modify the negative knowledge into $v(r, prior, ppol)^{-?}$ as just outlined.

Information Control. The output of the weakening method can be employed in essentially two ways: The weakened view $v(r, prior, ppol)$ is used for *data publishing* and thus the anticipated user is granted the full read access right to it, whereas all rights on the actually stored instance r are revoked. Alternatively, the anticipated user keeps his previously granted rights for *reading* or *querying*, but his requests are redirected to the weakened view. In the latter case we can even easily implement *content-dependent* query access rights with the granularity of single tuples/ground facts. More specifically, a *query request* regarding the (truth evaluation by the instance) of a ground fact Φ is handled as follows:

- If $\Phi \in v(r, prior, ppol)^{+?}$, then return Φ.
- If $v(r, prior, ppol)^{-?} \models \neg\Phi$, then return $\neg\Phi$.
- If $\Phi \models \Psi$ for some $\Psi \in v(r, prior, ppol)^{?} = \mathcal{C}$, then return MUM (a refusal).
- Otherwise, implying that there exists $\tau \in v(r, prior, ppol)^{\vee ?}$ such that $\Phi \models \tau$, then return the pertinent weakening disjunctions in $v(r, prior, ppol)^{\vee ?}$.

For each possible tuple/ground fact Φ exactly one of the four cases applies. Moreover, the third case applies for all tuples – whether in r or not – that entail an element in the refused knowledge. Similarly, the fourth case applies not only for the replaced tuples of r but also for all tuples – whether in r or not – that entail an element in the disjunctive knowledge about r without being affected by the second completeness sentence. Furthermore, since the fact of a refusal is explicitly indicated, a total refusal is only slightly related to simple tuple suppressions, which cannot be recognized in general.

Basic Assurance. To complete the presentation of our generic approach to generate weakened views, we formally verify the following assurance.

Theorem 1. *The weakening method of Subsect. 3.4 always returns a view that complies with the semantic confidentiality property defined in Subsect. 3.2.*

Proof. We consider appropriate inputs r, *prior* and *ppol* such that Stage 1 of the method successfully determines a finite set $\mathcal{T}$ of templates together with a set $\mathcal{C}$ of conflicting prohibitions with the required properties and Stage 2 defines the view $v := v(r, \mathit{prior}, \mathit{ppol})$. Let then $\Psi \in \mathit{ppol}$ be a prohibition sentence. The non-interferential Property 3 guarantees that $v^{+?} \cup v^{\vee ?} \cup \{v^{-?}\} \cup \mathit{prior} \not\models \Psi$. Hence, there exists an alternative instance r^{Ψ} such that $r^{\Psi} \models v^{+?} \cup v^{\vee ?} \cup \{v^{-?}\} \cup \mathit{prior}$, but $r^{\Psi} \not\models \Psi$. Define $v^{\Psi} := v(r^{\Psi}, \mathit{prior}, \mathit{ppol})$ to be the view generated for r^{Ψ}. It remains to show that $v = v^{\Psi}$.

In fact, the mutually exclusiveness of the four cases for a query request implies that r and r^{Ψ} can only differ in tuples for which the fourth case applies. Regarding that case, $r^{\Psi} \models v^{\vee ?}$ means that for each disjunction $\tau \in v^{\vee ?}$ there exists a tuple/ground fact $\Phi \in r^{\Psi}$ such that $\Phi \models \tau$. So, we verify that r^{Ψ} does not satisfy any further disjunctions in $\mathcal{T}$.

Assume indirectly that there is some $\tau = \Psi_1 \vee \ldots \vee \Psi_k \in \mathcal{T} \setminus v^{\vee ?}$ such that $r^{\Psi} \models \tau$. On the other hand, since $\tau \notin v^{\vee ?}$ and by step 2 of Stage 2, none of the existential facts Ψ_i of τ does occur in any of the disjunctions in $v^{\vee ?}$. Hence, since $r^{\Psi} \models v^{-?}$, the second completeness sentence in $v^{-?}$ implies that for all $\Phi \in r^{\Psi}$ we have $\Phi \not\models \Psi_i$ for each Ψ_i of τ, and thus $r^{\Psi} \not\models \tau$, resulting in a contradiction.□

The non-interferential Property 3 of Stage 1 is also *necessary* to uniformly guarantee semantic confidentiality of the view constructed in Stage 2 for all situations. For assume that there are $\mathcal{G}$, $\mathcal{R}$, $\mathcal{D}$ and Ψ violating that property. Then the construction of Stage 2 for the instance $r := \mathcal{G} \cup \mathcal{D}$ would return a view that entails Ψ, and thus semantic confidentiality could not be achieved.

Availability, Admissibility and Interchangeability. In general *formal confidentiality* has to be balanced with and complemented by further possibly conflicting goals. First of all, we comply with *availability* by weakening information only if seen to be (locally) necessary. Moreover, best availability is achieved if the templates in $\mathcal{T}$ are as short as possible, i.e., are disjunctions of length 2, and additional prohibitions to complete a clustering are avoided as far as possible.

However, favoring better confidentiality than formally required, we might want to generate longer templates. Furthermore, as already discussed in [5], a weakening disjunction used as replacing template should be *admissible* in some application-oriented sense. In the next section, we will instantiate admissibility by *interchangeability* (of length 2), requiring that a template should be formed from two existential facts that only differ in *one* attribute with *constants.*

4 Data Dependencies as a Priori Knowledge

As captured by the non-interferential Property 3 of Stage 1, controlling information requires us to consider the *a priori knowledge.* Of course, for arbitrary a priori knowledge expressed in first-order logic we cannot algorithmically decide in general whether or not the crucial non-entailment actually holds. Thus, to come up with algorithmic solutions, we have to suitably restrict the expressiveness of the a priori knowledge that we aim to consider. In this section, we elaborate an example of such a restriction, focussing on single-premise tuple-generating dependencies as an important class of sentences capturing background knowledge about an application. Other examples would have to be treated in a similar way.

A *single-premise tuple-generating dependency* (called *dependency* for short) is a sentence Γ in the underlying first-order logic $\mathscr{L}$ of the syntactic form

$$(\forall X_1)\dots(\forall X_k)\,[\,R(t_1,\dots,t_n) \Rightarrow (\exists Y_1)\dots(\exists Y_l)\,R(\bar{t}_1,\dots,\bar{t}_n)\,],$$

where $X_1,\dots,X_k,Y_1,\dots,Y_l$ are pairwise different variables, each universally quantified variable X_i occurring exactly once in $R(t_1,\dots,t_n)$ and at most once in $R(\bar{t}_1,\dots,\bar{t}_n)$, each existentially quantified variable Y_j occurring exactly once in $R(\bar{t}_1,\dots,\bar{t}_n)$, and – preferably to avoid an overall refusal – in both $R(t_1,\dots,t_n)$ and $R(\bar{t}_1,\dots,\bar{t}_n)$ at least one constant of *Dom* occurs. We will extract from Γ two existential facts in $\mathscr{L}_{exist}$, basically by taking the *existential closure* of each of the atomic formulas occurring in Γ:

$$\begin{aligned}&\mathit{prem}^{\exists}(\Gamma) := (\exists X_1)\dots(\exists X_k)\ R(t_1,\dots,t_n) \text{ and}\\ &\mathit{concl}^{\exists}(\Gamma) := (\exists X_{\bar{i}_1})\dots(\exists X_{\bar{i}_k})\,(\exists Y_1)\dots(\exists Y_l)\,R(\bar{t}_1,\dots,\bar{t}_n).\end{aligned}$$

Instead of converting originally universally quantified variables into existentially quantified ones, we might want to replace them by constants, basically by applying a *constant substitution* $\sigma : \{\,X_1,\dots,X_n\,\} \to \mathit{Dom}$:

$$\begin{aligned}&\mathit{prem}^{\sigma}(\Gamma) := R(t_1,\dots,t_n)[\sigma] \text{ and}\\ &\mathit{concl}^{\sigma}(\Gamma) := (\exists Y_1)\dots(\exists Y_l)\,R(\bar{t}_1,\dots,\bar{t}_n)[\sigma].\end{aligned}$$

A dependency establishes knowledge about the relationships between the validity of one single fact with another single fact, and can be used for reasoning in two ways. By *forward chaining*, knowing the validity of a fact that can be unified with the premise, we can infer the *validity* of the fact resulting from applying the unifier involved to the conclusion. By *backward chaining*, knowing the *non-validity* of a fact unifiable with the conclusion – as possibly enabled by our treatment of *partial completeness sentences* – we can infer the non-validity of the fact resulting from applying the unifier involved with the premise.

Regarding weakened views, basically, we have to avoid in an *instance-independent* way that for some possible instance such kinds of reasoning enable the adversary to exploit what we call an *interference* of a dependency with a prohibition: namely, to infer from the validity of both a weakening disjunction $\Psi_1 \vee \Psi_2$ of two prohibitions and a dependency Γ – together with the validity of positive or negative knowledge – that either Ψ_1 or Ψ_2 is *not* valid, i.e., that the other one is entailed. Technically, it can be shown that this unwanted effect can happen under three conditions: (i) a prohibition Ψ entails the existential closure $prem^{\exists}(\Gamma)$ of a dependency Γ; (ii) the sphere $Sp(\Psi)$ of a prohibition and the sphere of the existential closure $concl^{\exists}(\Gamma)$ of the conclusion of a dependency Γ have a nonempty intersection (which includes the case that $concl^{\exists}(\Gamma)$ entails Ψ); and (iii) a prohibition Ψ at the same time equals the existential closure of the conclusion of some dependency and for some constant substitution σ, $\Psi[\sigma]$ entails the existential closure $prem^{\exists}(\Gamma)$ of the premise of a dependency Γ.

These conditions will be blocked (step 1 below) by extending the policy with both the existential closure of the premise and the existential closure of the conclusion of the dependency Γ involved, thus excluding their spheres from published positive or negative knowledge. Unfortunately, in some cases this main measurement has to be complemented by further ones (steps 3 and 4/5 below). In a nutshell, the refinement for data dependencies proceeds as follows:

Refined Stage 1 (independent of r) **Safe Templates**

1. *extend* the policy by *implicit prohibitions* caused by a single dependency;
2. *clean* the policy from semantically *redundant prohibitions*;
3. *reject* conflicting prohibitions and establish total *refusals* instead;
4. *partition* the set of dependencies according to interactions with prohibitions;
5. respecting the partitioning, *cluster* prohibitions into admissible groups;
6. if possible, *add synthetic prohibitions* for completing a partial match;
7. *reject* prohibitions remained isolated and establish additional total *refusals*;
8. *form templates* of $\mathcal{T}$ as disjunctions, one for each group of the clustering.

We will only briefly explain the many subtle details by means of an example, reusing the confidentiality policy and the database instance of Sect. 2.

0. Input: The input is now given in Fig. 2. One can easily see that the given instance r complies with the given a priori knowledge *prior*.

1. Policy extension: As a basic step to achieve the most crucial non-interferential Property 3, the given confidentiality policy *ppol* is exhaustively extended according to each dependency in the given a priori knowledge *prior*. The dependencies Γ_1, Γ_2, Γ_3 and Γ_5 immediately interfere with *ppol*, and thus we have to add $prem^{\exists}(\Gamma_i)$ and $concl^{\exists}(\Gamma_i)$ for $i = 1, 2, 3, 5$. Afterwards, the dependency Γ_4 interferes with an added element, due to a suitable constant substitution of $concl(\Gamma_4) = (\exists X)\, R(g, e, X)$ and $(\exists X)(\exists Y)\, R(X, e, Y)$, requiring to add $prem^{\exists}(\Gamma_4)$ and $concl^{\exists}(\Gamma_4)$ as well. This leads to the *extended* policy

$$\begin{aligned} ppol_{prior} = ppol \cup \{\, & (\exists X)\, R(a, X, c),\ (\exists X)\, R(X, d, f),\ (\exists X)\, R(X, a, e),\\ & (\exists X)\, R(a, X, d),\ (\exists X)\, R(X, b, e),\ (\exists X)\, R(a, X, a),\\ & (\exists X)(\exists Y)\, R(X, e, Y),\ (\exists X)\, R(g, h, X),\ (\exists X)\, R(g, e, X)\}. \end{aligned}$$

$$r = \{\, (a,b,c),\ (a,f,g),\ (b,a,e),\ (b,b,d),\ (b,d,f),\ (g,e,i),\ (g,h,i) \,\}$$

(a) Original database instance r (complying with *prior*)

$$\begin{aligned} prior = \{\ \ \Gamma_1 &= (\forall X)\,[\,R(a,X,c) \Rightarrow R(X,d,f)\,] \\ \Gamma_2 &= (\forall X)\,[\,R(X,d,f) \Rightarrow R(X,a,e)\,] \\ \Gamma_3 &= (\forall X)\,[\,R(a,X,d) \Rightarrow R(X,b,e)\,] \\ \Gamma_4 &= (\forall X)\,[\,R(g,h,X) \Rightarrow R(g,e,X)\,] \\ \Gamma_5 &= (\forall X)\,[\,R(a,X,a) \Rightarrow (\exists Y)\,R(X,e,Y)\,]\ \ \} \end{aligned}$$

(b) Adversary's a priori knowledge *prior*

$$\begin{aligned} ppol = \{\ & R(a,a,a),\ R(a,b,a),\ R(a,b,c),\ R(a,b,d),\ R(a,b,e),\ R(a,c,a), \\ & (\exists X)\,R(a,e,X),\ (\exists X)\,R(b,e,X),\ (\exists X)\,R(c,e,X),\ (\exists X)\,R(b,X,e)\ \} \end{aligned}$$

(c) Confidentiality policy *ppol*

Fig. 2. Example input with a priori knowledge for refined weakening method

2. Policy cleaning: We then ensure the *independence* Property 2 of Stage 1 but without affecting the *covering* Property 1 of Stage 1. To do so, the extended policy $ppol_{prior}$ is cleaned by removing those elements that entail another element, which is still kept. Thus, the policy is reduced to the "core" subset of its weakest sentences. This leads to the *cleaned* (extended) confidentiality policy

$$\begin{aligned} \widehat{ppol}_{prior} = \{\ & (\exists X)\,R(b,X,e),\ (\exists X)\,R(a,X,c),\ (\exists X)\,R(X,d,f), \\ & (\exists X)\,R(X,a,e),\ (\exists X)\,R(a,X,d),\ (\exists X)\,R(X,b,e), \\ & (\exists X)\,R(a,X,a),\ (\exists X)(\exists Y)\,R(X,e,Y),\ (\exists X)\,R(g,h,X)\}. \end{aligned}$$

3. Rejecting prohibitions and establishing refusals: If a prohibition of $\widehat{ppol}_{prior}$ is entailed by $concl^{\sigma}(\Gamma)$ for some constant substitution σ for some dependency Γ, then it always needs to be rejected. This results in the set of *conflicting* prohibitions to be refused:

$$\begin{aligned} \mathcal{C} = \{\ & (\exists X)\,R(b,X,e),\ (\exists X)\,R(X,d,f), \\ & (\exists X)\,R(X,a,e),\ (\exists X)\,R(X,b,e),\ (\exists X)(\exists Y)\,R(X,e,Y)\ \}. \end{aligned}$$

4. Partitioning dependencies: To decisively ensure the crucial non-interferential Property 3, we have to take provisions against unwanted *joint* effects of two or more dependencies. Accordingly, we partition the given a priori knowledge *prior* with respect to $\widehat{ppol}_{prior}$, with the intention to block forming templates of prohibitions that are affected by dependencies of the same partition.

The dependencies Γ_1 and Γ_2 need to be in the same partition, as the existential closure of the conclusion of Γ_1 implies the existential closure of the premise of Γ_2. Further, Γ_3 also needs to be in this partition, because there is the prohibition $(\exists X)\,R(b,X,e) \in \widehat{ppol}_{prior}$, for which both implications $concl^{\sigma_2}(\Gamma_2) \models (\exists X)\,R(b,X,e)$ and $concl^{\sigma_3}(\Gamma_3) \models (\exists X)\,R(b,X,e)$ hold under constant substitutions σ_2 and σ_3 with $\sigma_2(X) = \sigma_3(X) = b$. Similarly, the dependencies Γ_4 and Γ_5 need to be in the same partition, as $\widehat{ppol}_{prior}$ contains the prohibition $(\exists X)(\exists Y)\, R(X,e,Y)$ with both $concl^{\sigma_4}(\Gamma_4) \models (\exists X)(\exists Y)\, R(X,e,Y)$ and $concl^{\sigma_5}(\Gamma_5) \models (\exists X)(\exists Y)\, R(X,e,Y)$ under arbitrary constant substitutions σ_4 and σ_5. As a consequence, the algorithm creates the partitioning $\mathcal{P} = \{P_1, P_2\}$ with $P_1 = \{\Gamma_1, \Gamma_2, \Gamma_3\}$ and $P_2 = \{\Gamma_4, \Gamma_5\}$.

$R(a,f,g)$
$R(b,b,d)$

$(\exists X)\, R(a,X,a) \vee (\exists X)\, R(a,X,c)$
$(\exists X)\, R(g,c,X) \vee (\exists X)\, R(g,h,X)$

$$\begin{array}{ll}\text{Refused: } \{ & (\exists X)\, R(X,a,e), \\ & (\exists X)\, R(X,b,e), \\ & (\exists X)\, R(X,d,f), \\ & (\exists X)(\exists Y)\, R(X,e,Y), \\ & (\exists X)\, R(a,X,d), \\ & (\exists X)\, R(b,X,e) \quad \} \end{array}$$

$$\begin{array}{llllll}
\multicolumn{6}{l}{(\forall X)(\forall Y)(\forall Z)\ [} \\
(& & Y \equiv a & \wedge\ Z \equiv e &) \vee \\
(& & Y \equiv b & \wedge\ Z \equiv e &) \vee \\
(& & Y \equiv d & \wedge\ Z \equiv f &) \vee \\
(& & Y \equiv e & &) \vee \\
(& X \equiv a\ \wedge & & Z \equiv a &) \vee \\
(& X \equiv a\ \wedge & & Z \equiv c &) \vee \\
(& X \equiv a\ \wedge & & Z \equiv d &) \vee \\
(& X \equiv a\ \wedge & Y \equiv f & \wedge\ Z \equiv g &) \vee \\
(& X \equiv b\ \wedge & & Z \equiv e &) \vee \\
(& X \equiv b\ \wedge & Y \equiv b & \wedge\ Z \equiv d &) \vee \\
(& X \equiv g\ \wedge & Y \equiv c & &) \vee \\
(& X \equiv g\ \wedge & Y \equiv h & &) \vee \\
\multicolumn{6}{l}{\neg R(X,Y,Z)\ ]}
\end{array}$$

Fig. 3. Inference-proof weakened view for inputs of Fig. 2

5. Admissible clustering: To prepare the clustering by means of an efficient graph algorithm, the prohibitions in the set $\widehat{ppol}_{prior} \setminus \mathcal{C}$ are used as vertices to generate an *indistinguishability-graph*. Although the prohibitions $(\exists X)\, R(a,X,c)$ and $(\exists X)\, R(a,X,d)$ are obviously interchangeable, the indistinguishability-graph does *not* contain an edge connecting the corresponding vertices of the graph, as both of these prohibitions entail an existential closure of a premise of the *same* partition P_1 due to $prem^{\exists}(\Gamma_1) = (\exists X)\, R(a,X,c)$ and $prem^{\exists}(\Gamma_3) = (\exists X)\, R(a,X,d)$. Then we employ a suitable *graph algorithm* to compute a clustering as a maximum matching on the considered indistinguishability-graph, getting (in this simple example trivially) $M = \{\ \{\ (\exists X)\, R(a,X,a),\ (\exists X)\, R(a,X,c)\ \}\ \}$.

6. Adding synthetic prohibitions. To tentatively maintain the covering Property 1, the prohibition $(\exists X)\, R(g,h,X)$, which is uncovered by the matching M, is admissibly paired with the additional synthetic prohibition $(\exists X)\, R(g,c,X)$.

7. Rejecting isolated prohibitions: To decisively maintain the covering Property 1, we still have to treat the prohibition $(\exists X)\, R(a,X,d)$, which remains isolated so far. Each interchangeable additional prohibition must differ either in the constant symbol at first position or in the constant symbol at the third position, and in both of these cases a dependency of *prior* would interfere with it. Accordingly, the prohibition $(\exists X)\, R(a, X, d)$ is additionally rejected and added to the set of conflicting prohibitions $\mathcal{C}$, to be used for total refusals.

8. Forming templates: We form a disjunction for each group of the clustering.

Stage 2 of the weakening method: We get the view given in Fig. 3.

Theorem 2. *The output $\mathcal{T}$ of the refined Stage 1 complies with the required properties of the generic weakening method, namely (i) covering the confidentiality policy ppol, (ii) having mutually independent templates, and (iii) being non-interferential under the a priori knowledge prior.*

Proof. Elaborated arguments following the explanations given for the example.

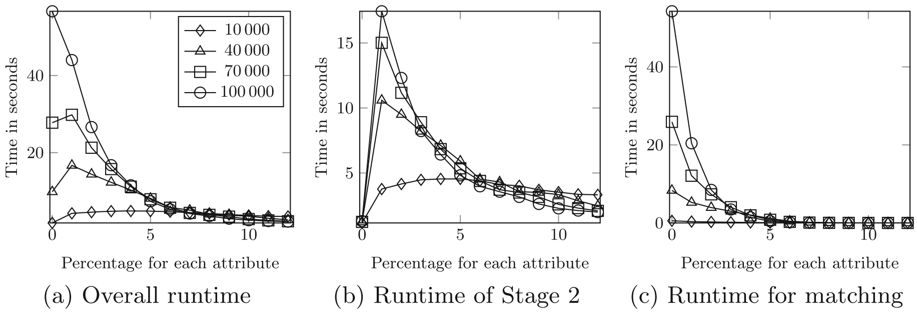


Fig. 4. Experiment 1: Varying *existential quantification* in confidentiality policy

5 Experimental Evaluation and Practical Efficiency

To experimentally confirm the practical efficiency of the generic approach under the refinement for data dependencies we provided a prototype implementation and performed several experiments. The prototype is implemented in Java 8, except for the C++ implementation of the matching algorithm. All experiments were run under Ubuntu 14.04 on a machine with 2 CPU sockets, each of which is equipped with an "Intel Xeon E5-2690" with 8 physical cores running at 2.9 GHz. As each CPU core can logically handle two threads due to hyperthreading, the machine has a total number of 32 logical CPU cores. To benefit from the modern hardware, the algorithms used for cleaning the policy, partitioning the a priori knowledge and constructing the weakened view have been parallelized (but we could not find a suitable parallelization for the maximum matching algorithm).

To compute a maximum matching (cf. [13,15]), the prototype benefits from the "Boost"-library [8]. Although a maximum matching on a general graph

Table 1. Parameters of experiments (varying parameter values in boldface)

Parameter	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5
Relation instance:					
Number of tuples	10^6	10^6	10^6	10^6	10^6
Number of constants used	20	20	20	20	20
Instance generation	fully random	fully random	random/chased	random/chased	random/chased
Confidentiality policy:					
Number of prohibitions	$1, 4, 7, 10 \times 10^4$	$1, 4, 7, 10 \times 10^4$	$1, 4, 7, 10 \times 10^4$	$1, 4, 7, 10 \times 10^4$	$1, 4, 7, 10 \times 10^4$
Number of constants used	12	**from 10 to 22**	12	12	12
Existential quantification	**from 0 % to 12 %**	5 %	5 %	5 %	5 %
Policy generation	fully random	fully random	fully random	fully random	fully random
A priori knowledge:					
Number of dependencies	-	-	**from 100 (200) to 2500**	1200	1200
Constants used	-	-	as for instance	as for instance	as for instance
Universal quantification	-	-	15 %	**from 5 % to 29 %**	15 %
Universal variables in concl.	-	-	10 %	**5 % less than above**	10 %
Existential variables in concl.	-	-	5 %	5 %	5 %
Knowledge generation	-	-	random/corrected	random/corrected	random/corrected
Parallelization:					
Number of threads	64	64	64	64	**from 1 (2) to 25**

Explanations:
existential quantification: for each attribute, the percentage of existential facts used as a prohibition which have an existentially quantified variable in that attribute
universal quantification: for each attribute, the percentage of premises of a dependency which have a universally quantified variable in that attribute
universal variables in conclusion: for each attribute, the wanted percentage of conclusions of a dependency which have a universally quantified variable in that attribute, subject that the dependency has enough such variables
existential variables in conclusion: for each attribute, the percentage of conclusions of a dependency which have an existentially quantified variable in that attribute
randomness: always coupled with the removal of semantic(ally equivalent) duplicates

$G = (V, E)$ can be computed in $O(\sqrt{|V|} \cdot |E|)$ (cf. [19]), common implementations as provided by “LEDA” [16] or “Boost” [8] prefer an algorithm performing in $O(|V| \cdot |E| \cdot \alpha(|E|, |V|))$ with $\alpha(|E|, |V|) \leq 4$ for any feasible input.

We only outline and briefly comment on five experiments, in each of them *varying one* specific generation parameter. We always employ the schema $R(A, B, C, D, E)$ and fix the generation parameters of the database *instances* (but not the instances themselves!): complying with the schema, having about 1 000 000 tuples, using 20 constants in *Dom* as an active domain, and for each repetition being fully randomly generated and then chased to enforce compliance with the a priori knowledge. Moreover, we always vary the following parameter of the confidentiality policy: having either 10 000, 40 000, 70 000 or 100 000 semantically different prohibitions, in each case being fully randomly generated. Table 1 provides an overview about the parameters considered. In the figures below, each evaluation curve is based on the average results of 100 experiments.

Experiment 1 studies the impact of allowing arbitrary existential facts (corresponding to subtuples) rather than only ground facts (corresponding to full tuples) in the confidentiality policy, thus leading to improved *flexibility to declare prohibitions*. First of all, the results shown in Fig. 4 clearly indicate the practical feasibility of our weakening method, which needs at most 1 min for both stages together. If about so much time is needed at all, then it is spent in the instance-independent first stage, in particular for the matching computation, while for the instance-dependent second stage only a few seconds suffice.

Experiment 2 deals with the number of *constants affected by prohibitions*. Figure 5 indicates an increase of the runtime nearly linear in that number at the beginning, but a somehow surprising pique for the matching computation.

Experiment 3 starts investigating the runtime consequences of taking care of *data dependencies*, at the beginning varying the number of dependencies considered. Figure 6 confirms that introducing a priori knowledge essentially affects the overall runtime, but fortunately still keeps it practically feasible. For the instance-dependent second stage the runtime even decreases when more dependencies are considered, among others caused by the strong impact of cleaning the policy. Moreover, Fig. 6d shows that the impact of rejecting conflicting

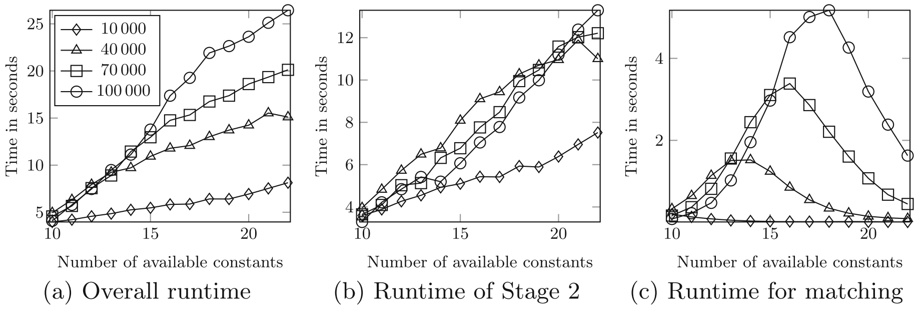
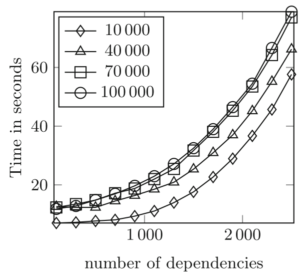
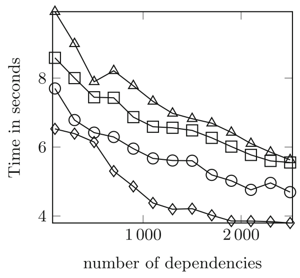


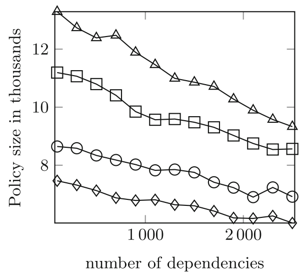
Fig. 5. Experiment 2: Varying *number of constants used* in confidentiality policy



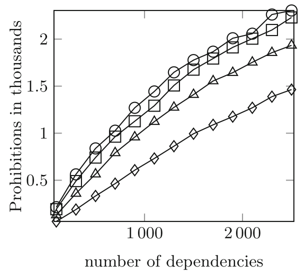
(a) Overall runtime



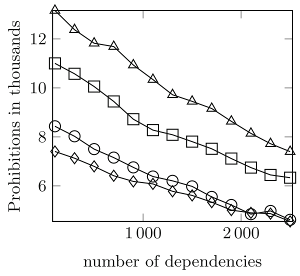
(b) Runtime of Stage 2



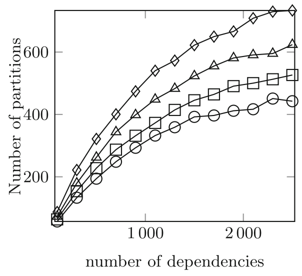
(c) Size of cleaned policy



(d) Rejected prohibitions

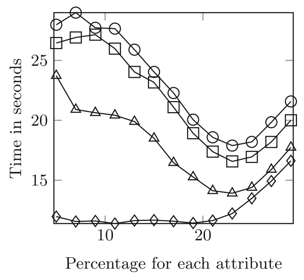


(e) Clustered prohibitions

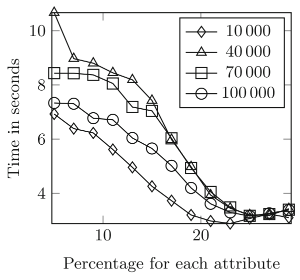


(f) Number of partitions

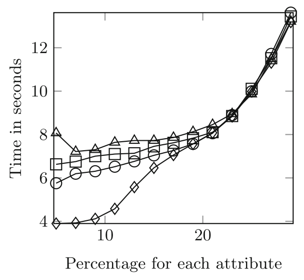
Fig. 6. Experiment 3: Varying *number of dependencies* in a priori knowledge



(a) Overall runtime



(b) Runtime of Stage 2



(c) Runtime for partitioning

Fig. 7. Experiment 4: Varying *universal quantification* in a priori knowledge

prohibitions is acceptable. And Fig. 6e and f indicate the behavior of partitioning the dependencies, which causes increasing costs but nevertheless keeps the number of the resulting partitions manageably bounded.

Experiment 4 serves for a closer look on the *syntactic structure of data dependencies* in terms of the occurring quantifications. Intuitively, a data dependency is the more powerful the more universally quantified variables are used. As can be seen from Fig. 7 among others, if we vary the percentage of universal variables as indicated, then at about a percentage of 20 % the interferential effects

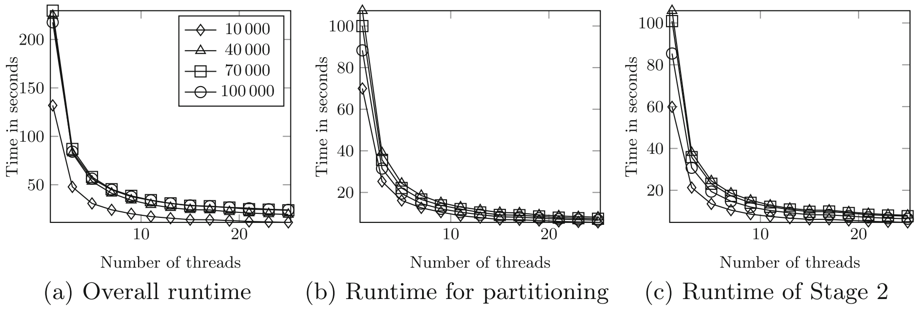


Fig. 8. Experiment 5: Varying number of threads

substantially grow leading in Stage 1 to more runtime needed for (parallelized) partitioning and thus overall as well, whereas the task of Stage 2 becomes easier.

Experiment 5 inspects the merits of *parallelization*. The curves shown in Fig. 8 indicate that we profit nearly optimally from parallelization, reducing the runtime to a half when the number of threads is doubled.

6 Conclusion

Our contribution constitutes a successful compromise between several potentially conflicting requirements: strong semantic confidentiality in terms of indistinguishability according to a declared policy representing pieces of information prohibited to be gained from interactions of data transfer, background knowledge and rational reasoning; expressive language for a "too curious" user's assumed a priori knowledge about the database content; flexible language for declaring prohibitions; uniform applicability for both data publishing and query answering; high availability of only correct information; application-dependent admissibility of weakening distortions; conformity to completeness assumptions regarding the actually stored data; and last but not least practical feasibility and scalability.

Though the concrete weakening method is novel, both its aims and its structure have been inspired by various previous work in the already very wide field of confidentiality preservation. Though often neglected, the ambitious aim of semantic confidentiality has already been considered in early work on confidentiality-preserving statistical databases, see [9], has explicitly and uniformly been used for the framework of Controlled Interaction Execution [2], and is in the spirit of many other approaches as highlighted by [12]. The further aim of providing only correct but if necessary explicitly weakened information has a long research tradition as well, in particular including the seminal work on refusals in information systems [17] and the extensive research on k-anonymity [14,18].

The structure of our weakening method generalizes a more special case [5] and is related to k-anonymization [14,18] by replacing sensitive data in a context-free way. Accordingly, our method also shares their complexity restrictions identified in [1,7]. Basically, the optimization problem for k-anonymization by maximum generalizations of values in the form of complete suppressions, aiming at a minimum number of suppressed values, is NP-hard when choosing $k \geq 3$, but solvable in polynomial time for $k = 2$. In contrast to the work on k-anonymity, we explicitly and formally deal with a priori knowledge, as throughout the framework of Controlled Interaction Execution [2], which also includes a view generation method based on distortion by lying [6]. The specific technique to deal with dependencies as a priori knowledge is related to similar efforts in the field of database fragmentation [4,10], another example of a weakening approach.

References

1. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Anonymizing tables. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 246–258. Springer, Heidelberg (2005)
2. Biskup, J.: Inference-usability confinement by maintaining inference-proof views of an information system. Int. J. Comput. Sci. Eng. **7**(1), 17–37 (2012)
3. Biskup, J., Bonatti, P.A.: Controlled query evaluation with open queries for a decidable relational submodel. Ann. Math. Artif. Intell. **50**(1–2), 39–77 (2007)
4. Biskup, J., Preuß, M.: Database fragmentation with encryption: under which semantic constraints and a priori knowledge can two keep a secret? In: Wang, L., Shafiq, B. (eds.) DBSec 2013. LNCS, vol. 7964, pp. 17–32. Springer, Heidelberg (2013)
5. Biskup, J., Preuß, M.: Inference-proof data publishing by minimally weakening a database instance. In: Prakash, A., Shyamasundar, R. (eds.) ICISS 2014. LNCS, vol. 8880, pp. 30–49. Springer, Heidelberg (2014)
6. Biskup, J., Wiese, L.: A sound and complete model-generation procedure for consistent and confidentiality-preserving databases. Theoret. Comput. Sci. **412**(31), 4044–4072 (2011)
7. Blocki, J., Williams, R.: Resolving the complexity of some data privacy problems. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6199, pp. 393–404. Springer, Heidelberg (2010)
8. Boost Graph Library: Maximum cardinality matching (2014). http://www.boost.org/doc/libs/1_55_0/libs/graph/doc/maximum_matching.html
9. Denning, D.E.: Cryptography and Data Security. Addison-Wesley, Reading (1982)
10. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Fragmentation in presence of data dependencies. IEEE Trans. Dependable Sec. Comput. **11**(6), 510–523 (2014)
11. Fung, B.C.M., Wang, K., Fu, A.W.-C., Yu, P.S.: Introduction to Privacy-Preserving Data Publishing - Concepts and Techniques. Chapman & Hall/CRC, Boca Raton (2010)
12. Halpern, J.Y., O'Neill, K.R.: Secrecy in multiagent systems. ACM Trans. Inf. Syst. Secur. **12**(1), 5.1–5.47 (2008)
13. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms. Algorithms and Combinatorics, 5th edn. Springer, Heidelberg (2012)
14. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: ℓ-diversity: privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data **1**(1) (2007)
15. Magun, J.: Greedy matching algorithms: an experimental study. ACM J. Exp. Algorithmics **3**(6) (1998)

16. Mehlhorn, K., Näher, S.: LEDA: a platform for combinatorial and geometric computing. Cambridge University Press, Cambridge (1999)
17. Sicherman, G.L., de Jonge, W., van de Riet, R.P.: Answering queries without revealing secrets. ACM Trans. Database Syst. **8**(1), 41–59 (1983)
18. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertainty Fuzziness Knowl. Based Syst. **10**(5), 557–570 (2002)
19. Vazirani, V.V.: A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{|V|} \cdot |E|)$ general graph maximum matching algorithm. Combinatorica **14**(1), 71–109 (1994)