Access control and inference problem in data integration systems

Mehdi Haddad

Nº D'ORDRE 2014-ISAL-0107 ANNÉE 2014

THÈSE

ACCESS CONTROL AND INFERENCE PROBLEM IN DATA INTEGRATION SYSTEMS

Présentée devant :
L'Institut National des Sciences Appliquées de Lyon

Pour obtenir :
Le grade de docteur

Spécialité :
Informatique

Formation doctorale :
Informatique

École doctorale :
Informatique et Mathématiques

Par :
Mehdi HADDAD

SOUTENUE PUBLIQUEMENT LE 1ER DÉCEMBRE 2014 DEVANT LE JURY COMPOSÉ DE :

Salima BENBERNOU, Professeur des Universités, Université Paris Descartes ..... Examinateur
Véronique BENZEKEN, Professeur des Universités, Université Paris-Sud 11 ..... Rapporteur
Fabio CASATI, Professeur, Université de Trento ..... Examinateur
Elena FERRARI, Professeur, Université d'Insubria ..... Rapporteur
Mohand-Saïd HACID, Professeur des Universités, Université Claude Bernard Lyon 1 Co-directeur de thèse
Robert LAURINI, Professeur des Universités, INSA de Lyon ..... Co-directeur de thèse

LABORATOIRE D'INFORMATIQUE EN IMAGE ET SYSTÈMES D'INFORMATION

A mes parents et à ma grand mère
A mes frères et à ma sœur
A mes amis

Remerciements

Je tiens à remercier tout d'abord mes encadrants, Pr. Mohand-Saïd Hacid et Pr. Robert Laurini, pour leur soutient et leurs conseils tout au long de ma thèse. J'exprime tous mes remerciements à l'ensemble des membres de mon jury : Pr. Salima Benbernou d'avoir accepté de présider le jury de thèse, Pr. Elena Ferrari et Pr. Véronique Benzaken d'avoir rapporté mon manuscrit de thèse, Pr. Fabio Casati d'avoir examiné le travail effectué durant ma thèse. Je tiens également à remercier les membres du laboratoire LIRIS d'avoir créé un environnement de travail agréable. Je remercie également les membres du département informatique (DISI) de l'université de Trento pour leur accueil ainsi que pour leur ambiance chaleureuse tout au long de mon enrichissante mobilité doctorale. Je n'oublie pas non plus tous ceux qui participaient aux "réunions" footballistiques du vendredi soir.

Abstract

In this thesis we are interested in controlling the access to a data integration system. In a data integration system, a mediator is defined. This mediator aims at providing a unique entry point to several heterogeneous sources. In this kind of architecture security aspects and access control in particular represent a major challenge. Indeed, every source, designed independently of the others, defines its own access control policy. The problem is then: "How to define a representative policy at the mediator level that preserves sources' policies?"

Preserving the sources' policies means that a prohibited access at the source level should also be prohibited at the mediator level. Also, the policy of the mediator needs to protect data against indirect accesses. An indirect access occurs when one could synthesize sensitive information from the combination of non sensitive information and semantic constraints. Detecting all indirect accesses in a given system is referred to as the inference problem.

In this manuscript, we propose an incremental methodology able to tackle the inference problem in a data integration context. This methodology has three phases. The first phase, the propagation phase, allows combining source policies and therefore generating a preliminary policy at the mediator level. The second phase, the detection phase, characterizes the role of semantic constraints in inducing inference about sensitive information. We also introduce in this phase a graph-based approach able to enumerate all indirect access that could induce accessing sensitive information. In order to deal with previously detected indirect access, we introduce the reconfiguration phase which provides two solutions. The first solution could be implemented at design time. The second solution could be implemented at runtime.

Keywords: Access control, Data integration, Inference problem, Authorization policy integration.

Résumé

Dans cette thèse nous nous intéressons au contrôle d'accès dans un système issu d'une intégration de données. Dans un système d'intégration de données un médiateur est défini. Ce médiateur a pour objectif d'offrir un point d'entrée unique à un ensemble de sources hétérogènes. Dans ce type d'architecture, l'aspect sécurité, et en particulier le contrôle d'accès, pose un défi majeur. En effet, chaque source, ayant été construite indépendamment, définit sa propre politique de contrôle d'accès. Le problème qui émerge de ce contexte est alors le suivant : "Comment définir une politique représentative au niveau du médiateur et qui permet de préserver les politiques des sources de données impliquées dans la construction du médiateur?"

Préserver les politiques des sources de données signifie qu'un accès interdit au niveau d'une source doit également l'être au niveau du médiateur. Aussi, la politique du médiateur doit préserver les données des accès indirects. Un accès indirect consiste à synthétiser une information sensible en combinant des informations non sensibles et les liens sémantiques entre ces informations. Détecter tous les accès indirects dans un système est appelé problème d'inférence.

Dans ce manuscrit, nous proposons une méthodologie incrémentale qui permet d'aborder le problème d'inférence dans un contexte d'intégration de données. Cette méthodologie est composée de trois phases. La première, phase de propagation, permet de combiner les politiques sources et ainsi générer une politique préliminaire au niveau médiateur. La deuxième phase, phase de détection, caractérise le rôle que peuvent jouer les relations sémantiques entre données afin d'inférer une information confidentielle. Par la suite, nous introduisant, au sein de cette phase, une approche basée sur les graphes afin d'énumérer tous les accès indirects qui peuvent induire l'accès à une information sensible. Afin de remédier aux accès indirects détectés nous introduisons la phase de reconfiguration qui propose deux solutions. La première solution est mise en œuvre au niveau conceptuel. La seconde solution est mise en œuvre lors de l'exécution.

Mots-clés : Contrôle d'accès, Intégration de données, Problème d'inférence, Intégration de Politiques d'autorisation.

List of publications

[Haddad 2012] Mehdi Haddad, Mohand-Said Hacid and Robert Laurini. *Data Integration in Presence of Authorization Policies*. In 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2012, pages 92-99.

[Haddad 2013a] Mehdi Haddad. *Contrôles d'accès pour l'intégration de données*. In 29eme journées Bases de Données Avancées (BDA), young researcher article. Fsrench conference, 2013.

[Haddad 2014] Mehdi Haddad, Jovan Stevovic, Annamaria Chiasera, Yannis Velegrakis and Mohand-Said Hacid. Access Control for Data Integration in Presence of Data Dependencies. In Database Systems for Advanced Applications - 19th International Conference, DASFAA 2014, pages 203-217.

Contents

Chapter 1

Introduction

Contents

Nowadays, database systems are essential to any business managing information [Garcia-Molina 2002]. For example, even web sites usually make use of databases as data repositories behind the interface. A database is necessary to provide the requested service. In addition to web applications, databases play an important role for many corporations. Indeed, databases are used to manage all the needed data in such corporations in order to achieve their business.

Today database systems [Özsu 2011] are concerned with the management of health data, insurance data, scientific data, legislation data, military data, human communications data, emails and tweets among other kinds of data. This demonstrates the importance databases play in our everyday life. These information should also provide services able to control access to information they handle [Ashley 2003, LeFevre 2004, Agrawal 2002]. Indeed, even though these systems offer great benefits, users are reluctant to use such systems if their privacy is not preserved [Acquisti 2005].

The importance of database systems and their security at economical, scientific and societal levels has led governmental organizations to recognize the need for information security at both technological and societal levels. For instance, the French national research strategy in science and technology report [Allistene 2013] lists security and privacy as a core challenge to information technology. The report aims at identifying the major challenges that computational sciences will have to tackle in the near future in order to improve, among other things, the national industry, the quality of life and the individual well being.

The European Union Horizon 2020 program also recognizes the importance of security to information and communication technology (ICT) [Commission 2014]. Indeed, a whole ICT theme is devoted to cyber security and trustworthy ICT. One main scope of this theme is security by design that aims at providing end-to-end security mechanisms.

The French 2030 innovation commission [Innovation 2013] has listed "Massive data management" as one of the seven ambitions that France needs to emphasize in order to achieve long term prosperity and employment. Among those challenges, data security and accessibility are identified as essential for both individuals and corporations.

In this thesis, we are interested in security issues that arise in database systems. Next, we will present the historical evolution of both database systems and their security concerns.

Historically, first database systems have evolved from file systems. The main goal of such systems is to store and access data over a long period of time. Later on, with the introduction of the relational model [Codd 1970], database systems significantly improved their functionalities. Indeed, the relational model allows to organize data as tables referred to as relations. The idea is to provide a high level language to manipulate data which greatly simplifies the use of databases. The core concept is the one of separating the physical and logical definitions of data [Abiteboul 1995]. This allows users to focus on how to efficiently use data rather than how data need to be physically stored. These database systems were implemented on a single node and are referred to as *centralized databases*.

The combination of (centralized) database systems and network systems

has led to the emergence of *distributed database* systems [Ceri 1983]. In such systems, data is managed by several distributed nodes. These nodes are connected through a computer network. The main goal of distributed databases is to provide a *transparent* access to several local systems. Distributed databases have also been designed to provide more reliability and better performance than centralized systems. Indeed, using replication could offer better fault tolerance and thus better reliability. In addition, splitting and distributing the task on many nodes could improve the processing time and thus the overall performance.

Later on, the concept of *multidatabase* has been introduced. In the literature, this concept also refers to *federated databases* [Sheth 1990] or *data integration* systems [Lenzerini 2002]. The main additional requirements of multidatabases with respect to traditional distributed database systems are heterogeneity and autonomy. Heterogeneity refers to the fact that the local systems have been designed independently using different technologies (*e.g.,* different data representations, different query languages). Autonomy refers to the ability of local systems to continue working independently from the global system. These two additional requirements induce several challenges at different levels (*e.g.,* query processing, query optimization, constraints enforcement). An additional challenge posed by these systems is the one of security [Silberschatz 1996] .

A huge amount of work in the database community has been devoted to provide new paradigms able to manage distributed data. This distribution took different directions. Each direction has been motivated by concrete scenarios and new application requirements. The main directions that have been investigated are:

- Data integration [Lenzerini 2002, Chawathe 1994, Halevy 2006]: It aims at providing a unique entry point to multiple heterogeneous data sources.
- Data exchange [Fagin 2005, Miller 2000, Popa 2002]: It aims at transforming data defined according to a source schema in order to suit a target schema.
- Data fusion [Bleiholder 2008, Naumann 2006]: It aims at resolving conflicts that arose in data integration (*e.g.,* semantic heterogeneity, duplicate detection).

All these systems share a common objective which is combining information provided by different systems. These systems are interesting from information perspective but they also induce some security issues [Becker 2004, Li 2003].

To deal with security issues in computer systems, different fields of study have been introduced to design appropriate approaches able to provide security guarantees [Landwehr 1981]. Notable security fields involve authentication [Lamport 1981, Halevi 1999], access control [Griffiths 1976, Fagin 1978], cryptography [Diffie 1976, Howard 1987, Rivest 1978] and auditing [Agrawal 2004, Motwani 2008]. These fields have followed the evolution of database systems by offering new security paradigms to cope with vulnerabilities induced by new features (*e.g.,* distribution, heterogeneity, autonomy).

In this manuscript, we focus on the security challenge that mainly arise in data integration systems.

Historically, security always comes after database systems have been designed. Indeed, the more database systems evolved providing advanced functionalities the more security issues arose.

First security issues were induced by the fact that several users use the same (centralized) system. The idea in such systems is to ensure that only authorized users could access sensitive information [Jajodia 1991, Lunt 1990]. In other words, ensure that a user could only access the part of data that is allowed to her/him.

Later on, database systems became distributed which also induced new security requirements [Thuraisingham 1992]. In addition to ensuring that users only access authorized data, the use of cryptography has became mandatory to secure network communications. Another issue of distributed databases is the one of authentication. Indeed, authentication has been even more highlighted due to the distribution nature of such systems (*e.g.,* issues related to remote authentication [Chang 1991]).

In addition to the security issues of centralized and distributed databases, multidatabases introduced new security vulnerabilities [Jonscher 1994, di Vimercati 1996]. Indeed, the policy that needs to be enforced by the federation has to be negotiated between the federation participants. This negotiation needs to resolve conflicts among the different specifications used by the

participants. This issue is referred to as authorization autonomy. This means that each participant in the federation defines its own access control policy independently of the other participants.

Another security issue induced by federated databases is the one of inference. An inference, in the context of access control, is the ability of users to obtain sensitive information by combining non sensitive ones. Although this problem also appears in centralized systems, it is highlighted by the federated architecture. Indeed, in such architectures data integration is the basic building bloc. It has been shown that appropriate queries in such architectures could lead to infer prohibited information [Farkas 2002, Su 1987, Thuraisingham 1987].

So far, we have summarized how the evolution of database systems impacts the security of such systems. In particular, the fact that database distribution could induce new security challenges that need to be carefully addressed. Other security challenges are also induced by the following concepts:

- Enforcing laws and regulations: Database systems are requested to protect their data. In particular, when they manage confidential information (*e.g.,* the Health Insurance Portability and Accountability Act the (HIPAA) [Control 2003], the Children's Online Privacy Protection Act (COPPA) [Commission 1998], the Federal Act [Commission 2000]).

  Such regulations have to be respected and thus security administrators need to setup and to enforce appropriate mechanisms that achieve this goal.

- Privacy concerns: People are more and more concerned about their privacy. Indeed, different stories (*e.g.,* de-identification of Netflix users [Narayanan 2008]) showed that people are reluctant to lose their privacy. This also shows that although security mechanisms are already deployed, current mechanisms need to be improved [Liu 2009].

- Inside threat [Spitzner 2003]: This concept is probably one of the most dangerous threat to data security nowadays. Indeed, it considers the case where an authenticated (legitimate) user aims at harming the system (*e.g.,* accessing confidential information).

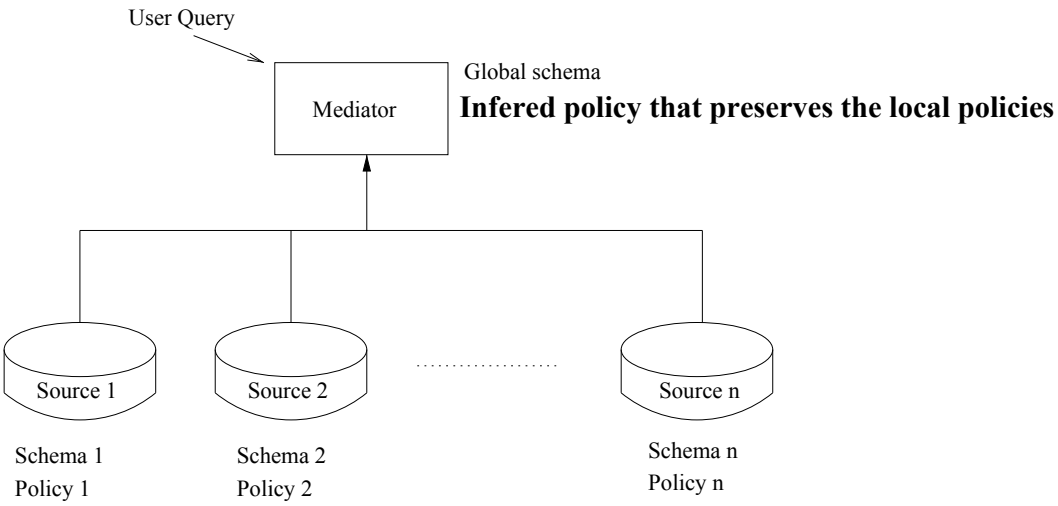

Figure 1.1: Data integration with authorization policies

1.1 Problem statement

In this thesis, we are interested in designing a relevant access control for data integration systems. Our goal is to define an appropriate policy to be attached to a mediator constructed using a data integration approach (see Figure 1.1). In such a system, each data source is autonomous. This means that it has been constructed independently of the other sources. The mediator is then constructed by combining some information extracted from the sources.

In this context, we are interested in the following issues:

- Data distribution: We consider distributed systems constructed using data integration approaches. We assume a mediator that integrates data from different sources.
- Privacy concerns and regulations: We consider sources' policies constructed for protecting local data and/or enforcing regulations. The idea is to preserve such policies at the mediator level. This could encourage sources participating into data integration systems.
- Inside threat: We consider the case of malicious users. In this case, malicious users are legitimate (authenticated) users who aim at accessing prohibited information.

1.2 Objectives

Our main objective is to assist administrators in defining authorization policies at the mediator level. This mediator is built using data integration approaches on top of a set of sources. In data integration, sources are assumed autonomous and therefore have been designed independently. This also means that each source's authorization policy has been designed independently of other sources. The autonomy of sources refers to the fact that local applications running at the sources should continue to work properly after the integration process is completed. Our objective is therefore to provide a comprehensive methodology able to generate a representative policy for the mediator while preserving the policies of the sources. Preserving the policies of the sources aims at verifying that a denied access at the level of a source should also be denied at the level of the mediator.

We investigated these objectives by proceeding as follows. We first studied the different research fields that could provide tools and concepts to tackle our problem. We analyzed the proposed approaches in each field and the way those approaches could be combined. We then formally defined our problem and designed an incremental approach. This approach rests on three main phases. Each phase is built on top of the previous one.

In order to be able to propose a methodology able to generate an appropriate access control policy to a data integration system, we started by considering different fields in the literature. We identified four fields to be relevant to our problem. These fields are:

- Data integration [Lenzerini 2002, Chawathe 1994, Halevy 2001]: Different ways of combining data could induce different access control issues. The idea is to study the different approaches proposed to achieve data integration. In this field, we mainly focused on the approach dealing with the definition of the mediator schema. In particular, how the mapping between the mediator and source elements is performed.

- Access control: In this field we studied the different access control models that have been proposed to protect database systems. Many efficient mechanisms to protect data against direct access exist [Bertino 2011]. In our settings, this helps preserving the source policies. Indeed, pro-

tecting data from direct access, at the mediator level, is the first step towards preserving the policies of the sources. This is achieved by denying prohibited access at the mediator level when such an access is also prohibited at the source level.

- Inference problem [Farkas 2002]: This field is probably the most important to our work. It highlights some limitations of traditional access control models. It mainly considers the impact of indirect access on access control policies. Indirect access represents the core concept of inferences in the context of access control. An indirect access occurs when a user could synthesize sensitive information by combining non sensitive information.

- Policy composition [Bonatti 2002, Jajodia 2001]: In this field we studied how policies designed for independent systems could be combined. Approaches in this field did not consider issues related to the inference problem. Combining sources' policies could be interesting in order to define a preliminary global policy. The synthesized policy could be enriched later on in order to deal with inferences.

We built on existing approaches to propose an incremental methodology for defining an authorization policy of the mediator. This policy should:

- Preserve the policies of the sources: This property ensures that the access constraints attached to the data sources are satisfied at the global level. In other words, if some piece of information is prohibited in a source, it is also prohibited at the mediator level. This property is quite intuitive and it is important to enforce. Indeed, sources are reluctant to participate in a system that does not preserve their data confidentiality.

- Prevent against inferences: This property states that a user should not be able to indirectly access a prohibited information. In particular, this property is concerned with malicious users aiming at obtaining confidential information. Indeed, synthesizing and combining non sensitive information could lead to disclose information about sensitive information.

1.3 Research challenges

As discussed in the previous section access control (or even database security) is closely related to the way data is handled. Classical access control models are not satisfactory to efficiently deal with inferences. In the data integration context the inference problem is highlighted for the following main reasons:

- Each source has been designed independently of the others. Hence, combining data from sources could not be considered by any local administrator while defining local authorization policies.
- New semantic constraints could be attached to the mediator. These constraints could be used to infer sensitive information.

1.4 Methodology

In this section we describe the different phases of our methodology.

1.4.1 Propagating the sources' polices

This phase aims at transferring policies of the sources to the mediator. This phase combines the local policies to generate a preliminary policy for the mediator. This combination aims at preserving the sources' policies by making sure that appropriate local authorization constraints are propagated to the mediator. This combination preserves data sources from direct access.

1.4.2 Detection phase

The aim of this phase is to detect the violations that could occur at the mediator level. This phase is motivated by the inference problem that showed that neither classical access control models nor policy composition approaches are able to protect data against indirect access. This phase focuses on inferences that could arise using semantic constraints. Since we consider the relational model as a reference framework, one of the most used semantic constraints are

functional dependencies. We illustrate how functional dependencies could be used by malicious users in order to synthesize sensitive information. To this end, we propose a graph-based approach able to detect and enumerate a set of violating transactions. Each transaction is a set of queries. This approach allows to know a priori that if a malicious user accesses all the queries of a single transaction then a policy violation could occur.

1.4.3 Reconfiguration phase

This phase proposes solutions to deal with policy violations. Indeed, different methods could be considered to prevent malicious users from completing any transaction. Please recall that violating transactions are identified at design time. In this phase we propose two different solutions for preserving policy violations that could happen if inference mechanisms are used. These solutions differ at the conceptual level. Indeed, the first solution could be implemented at design time while the second solution could be implemented at runtime.

1.4.3.1 Policy revision

This solution could be implemented at design time and it aims at preventing the completion of violating transactions.[1] The idea is to enrich the original policy with a set of new rules. These rules are constructed in such a way to forbid the access for at least one query in each violating transaction. To maintain best data availability, we consider the problem of finding the minimum set of rules to be added while preventing the completion of violating transactions.

1.4.3.2 Query tracking

This solution is history-based. It could be implemented at runtime. This leads to collect and store evaluated queries. When a new query is issued, a merge between past queries and the current query is performed. If the combination leads to infer a prohibited information then the current query is denied.

[1] By completion of a transaction we mean issuing and evaluating all the queries of that transaction.

1.5 Structure

The rest of this manuscript is organized as follows: Chapter 2 describes related work. It discusses data integration approaches, traditional access control models and their limitations, policy composition approaches, and the inference problem that aims at preventing data from indirect access. Chapter 3 introduces a set of relevant definitions and preliminaries. Based on these concepts and definitions, we formally define our problem. Chapter 4 focuses on the detection of policy violations that could occur. A graph-based approach is introduced to enumerate all violations that could occur at the mediator level by exploiting functional dependencies. Chapter 5 describes two solutions intended to prevent the realization of the detected violating transactions. The first solution could be implemented at design time while the second could be implemented at run time. A comparison of these two approaches is also provided. Chapter 6 draws conclusions and discusses future work.

Chapter 2

Related work

Contents

In this chapter, we start by discussing security in information systems in general. Then, we present and discuss four fields of interest that are relevant to our research. The problem we consider is data access in the context of data integration systems by accommodating access control policies. The fields that we identified as being relevant to this problem are the following: data integration, access control, inference problem and policy composition. We discuss proposed approaches in each of those fields and highlight their interactions.

2.1 Security in information systems

Security is an important component of information systems. Indeed, different components need to be put in place in order to provide security and each one needs to provide guarantees about its functioning. According to [Sandhu 1996a] there are three basic components that provide security to information systems: Authentication, Access control and Audit. According to [Bertino 1995] the basic components for security are: authentication, access control and encryption.

- Authentication: it aims at establishing the identity of a user of a system.
- Access control: it aims at preserving the confidentiality, the availability and the integrity of data [Bertino 2005]. In other words, access control gathers three important components of information security:
  - Confidentiality aims at protecting data from unauthorized access.
  - Availability aims at preventing from denials accesses that could make data inaccessible.
  - Integrity aims at preventing unauthorized data modification.
- Audit: it aims at analyzing system activities to detect security violation. This analysis could occur either offline or online.
- Encryption: it protects data when they are sent for communication purposes. It ensures that only intended users will be able to access the data that have been sent.

In this thesis, we focus on one of the major components of security, namely access control. In particular, we study issues that arise when one would enforce access control in data integration systems. To this end, we start by studying existing approaches in the literature. We found that this problem is complex since it necessitates combining different fields. So, to achieve security in data integration scenarios we identified four dimensions, that we believe should to be considered in order to achieve security in the context of information integration systems.

The first dimension is the data integration one. Indeed, the way data is managed could influence the way access control needs to be performed. The more data is accessible, in order to fulfill some utility, the more access control methods need to be sharp and precise. This leads us to the second dimension, namely, access control. Indeed, different access control models and mechanisms have been proposed. One needs to carefully select an appropriate model for the specific application being targeted. This selection could rely on access control properties (*e.g.,* granularity). A well known shortcoming of access control models is that they have been designed to protect data against direct access. They do not consider indirect access that could lead to sensitive information disclosure. This constitutes the third dimension we consider, namely, the inference problem. This problem has been introduced to remedy access control models limitations. Several approaches have been designed to protect data against indirect access. The last dimension is the policy composition. Indeed, this field aims at combining policies belonging to different systems. This is related to data integration scenario where different systems (called sources) are combined. The issue with policy composition approaches is that they do not deal with inference. Nevertheless, they could be used to start reasoning about access control in data integration scenarios and then extend the composition to cope with inferences.

Next, we describe these four dimensions separately. For each dimension we discuss the most relevant approaches that have been proposed in the literature. We classify and compare such approaches for each dimension. Finally, we discuss the common issues shared by the approaches.

2.2 Data integration

The need for combining and merging data provided by different centralized (or distributed) systems has led to the emergence of new approaches for data managements. A first attempt to combing information from different systems is the concept of *superviews* introduced by Motro [Motro 1981]. This approach allows to define views over relations stemming from different systems. Later on, the popular concept of *federated databases* (see [Sheth 1990] for a survey) was introduced in the vein of combining information from different centralized systems. Federated databases put the basis for the field of information integration. Indeed, it introduced several desired properties that are still considered nowadays. Some of these properties are the following:

- Heterogeneity: The sources could have different data models, query languages, etc.
- Autonomy: Each source is designed independently from the others and local applications continue running after the definition of the federation.
- Security: Data confidentiality of the sources should be preserved by the federation.

Given these approaches, the concept of *data integration* was introduced [Deen 1987, Hull 1996]. Data integration aims at providing a unique entry point to a set of data sources. This is achieved by defining a mediator [Wiederhold 1992]. The mediator is defined between the user and the sources. Indeed, users do not have to query each source separately and then combine the results. They only need to query the mediator's global schema. The mediator is in charge of identifying and accessing relevant sources and collecting the partial results. Finally, the mediator combines the partial results into a complete answer, that will be returned to the user. There are two main approaches when it comes to define a mediator, either using a materialized approach or a virtual approach:

- Materialized integration ([Zhuge 1995, Zhou 1995]): This approach allows to define a global schema and to store data provided by the sources. The motivation of this approach is to be able to answer queries without accessing the sources.

- Virtual integration ([Litwin 1990, Chawathe 1994, Halevy 2001]): This approach consists in defining a virtual global schema without materializing any data at the mediator level. The idea is to decompose (rewrite) the user query into a set of queries that are sent to the relevant sources.

These two approaches are complementary and could even be combined [Hull 1996]. Each approach could be considered better for a specific application. Indeed, materialized integration offers a faster query execution than virtual integration. In the other hand, the virtual integration is more relevant for application that are subject to frequent updates. We will focus on virtual integration (see [Lenzerini 2002] for a survey) and in particular on the GAV integration approach.

2.2.1 Virtual data integration

A data integration system can be defined as a triple (G, S, M) where G is the global schema that is attached to the mediator, S is the source schemas that contains the elements of all the sources and M is the mapping between the global schema and the source schemas. In the relational model this mapping could be expressed by means of view definitions. We refer to the mediator's elements as virtual relations since no data is actually stored in the mediator. In the virtual approach, there are two main approaches that differ in the way the mapping between the global schema and the source schemas is defined.

- GAV (Global As View) (*e.g.,* [Carey 1995, Beneventano 2000, Goh 1999, Chawathe 1994]): This approach defines each virtual relation by a conjunctive query over the sources' relations.
- LAV (Local As View) (*e.g.,* [Kirk 1995, Manolescu 2001]): The mapping between the global schema and the sources' relations is defined as follows: each local relation is expressed by a conjunctive query over some relations of the mediator.

2.2.2 Comparison of GAV and LAV

Although GAV and LAV achieve the same goal of defining a global schema of a mediator, the way they define the mapping differs. This induces different

behaviors regarding query rewriting and schema update. Next, we discuss GAV and LAV by considering both of these criteria.

2.2.2.1 Query rewriting

Query rewriting is simple in GAV where it reduces to query unfolding. LAV, in the other hand, requires more complex and time consuming algorithm to perform query rewriting. These rewriting algorithms are discussed and compared in [Halevy 2001]. The three major algorithms that have been proposed are: the Bucket algorithm [Levy 1996, Grahne 1999], the Inverse rules algorithm [Qian 1996, Duschka 2000] and the MiniCon algorithm [Pottinger 2000, Pottinger 2001]. These algorithms allow enforcing the LAV approach. Nevertheless, they are exponential and thus require more processing than simple unfolding algorithms used in GAV.

2.2.2.2 Schema update

In GAV if a source schema evolves or a new source is introduced the global schema needs to be modified. LAV, in the other hand, does not require any modification if a source schema changed or a new source join the integration system.

To summarize, both approaches are well studied and relevant for data integration. Nevertheless, depending on the target application one could either choose GAV or LAV. We also note that these approaches could be combined. Such a combination is introduced in GLAV [Friedman 1999]) where a mapping using both GAV and LAV principals is defined.

In this thesis we focus on security issues. In particular, we focus on access control challenges that arise when data integration is achieved. Indeed, new access control issues could arise at the mediator level even if those of the underlying sources are in place. Next sections discuss access control models, the inference problem and composition of access control policies.

2.3 Access Control

Access control aims at preserving data confidentiality, availability and integrity [Bertino 2005]. In this vein, different concepts have been defined to achieve these goals. These concepts describe how access control should be defined from an abstract level to a concrete level. Different concepts have been introduced. Here, we present those described in [Samarati 2001b]:

- Security policy: It describes the most abstract view of the system. At this level access control rules are defined. The requirements of the system are described in order to comply with some specification (*e.g.,* laws, regulations). This description does not provide any method on how it should be enforced.
- Security model: It formalizes the rules defined in the security policy and describes the way they should work. This level aims at providing a framework where proof of properties could be accomplished.
- Security mechanism: It describes the low level methods that are used to enforce the rules formalized at the security model level.

2.3.1 Access control models

Historically, there has been three main access control models. The first one is DAC (Discretionary Access Control) [Lampson 1974, Graham 1972] where authorizations are provided, individually, for each user. Then, another model emerged, namely, MAC (Mandatory Access Control) [Denning 1976] where both users and resources are assigned, by a centralized entity, a credential level. These levels are organized within a partial order. The access is allowed if the user's level is higher than the level of the underlying resource. Later on, RBAC (Role-Based Access Control) [Ferraiolo 1995] has been introduced. The idea of this model is to define the concept of role that mediates the interaction between users and resources. Indeed, users are given roles and a set of rules state which role could access which resources. Other models have been introduced, such as ABAC [Wang 2004, Bonatti 2000], Or-BAC [Cuppens 2003], TRBAC [Bertino 2001]. These models extend previous models by introducing new features. ABAC (Attribute-Based Access Control) aims at providing

more granularity to the access control. Or-BAC (Organization-Based Access Control) introduces the concept of context (*e.g.,* location, time). In this model, a permission could be considered only in a particular context. TR-BAC (Temporal Role-Based Access Control) model adds the notion of time to RBAC.

2.3.2 Access control enforcement

To enforce these models different mechanisms have been introduced. Here, we describe the main approaches that have been proposed. We follow the chronological order of appearance which is DAC, MAC, RBAC and finally ABAC.

2.3.2.1 DAC (Discretionary Access Control)

The access matrix model has been introduced to enforce DAC policies. In this model a matrix is constructed. The columns of this matrix represent objects. Each line of the matrix corresponds to a user. Then, at each cell is stored the actions that a user could perform on the corresponding object. Later on, this model has been formalized by introducing the HRU[1] model [Harrison 1976]. This model identifies six primitive operations that model the evolution of the system. Different enforcement techniques have been introduced for the access matrix model:

- **Authorization table:** This method constructs a three columns table. The columns correspond to subjects, object and actions. Each line of this table expresses that a user could perform a set of actions on an object.
- **Access control list (ACL):** In this method, a list is constructed for each object. This list enumerates all the subjects and their actions that could be performed on the object.
- **Capability:** For each user a capability list is defined. The list contains all actions that the subject could perform on objects.

[1] The name of the HRU model stands for Harrison, Ruzzo, Ullman model [Harrison 1976].

2.3.2.2 MAC (Mandatory Access Control)

In MAC policies a central authority regulates the actions that are performed by subjects on objects. The most popular way to enforce policies in this model by defining a set of levels organized through a partial order [Denning 1976]. Then, subjects are assigned levels and objects are assigned levels as well. If the subject's level is higher than the object's level then the user could access that object. This kind of policies is called multilevel security policies. This model also aimed at controlling information flows between different security levels. An information flow occurs when information is moved from one level to another. These flows could induce sensitive information leakage. To deal with these flows some additional rules are defined. Two popular rules are the following:

- No read up: This rule states that a user could only read objects having a lower level than her/his own level. This rule is quite intuitive and aims at preserving data against unauthorized access.
- No write down: This rule states that a user could only write on object having an equal or higher level. This rule has been introduced to ensure that data could not be declassified.

So far, we described the general mandatory model. More specific models aiming at providing a particular property have been introduced. Next, we discuss two of those models, namely, the Bell-LaPadula model and the Biba model.

- Bell-LaPadula Model [Bell 1973]: This model has been designed to multilevel policies with the aim of guaranteeing data confidentiality. This model enforces the no read up property.
- Biba model [Biba 1977]: This model has been introduced to multilevel policies in order to ensure data integrity. This model enforces the no write down property.

First concerns about information flows and indirect access to prohibited information have been identified in mandatory policies (*e.g.,* the no read up and the no write down rules). Indeed, both Bell-Lapadula and Biba models

aim at preventing information from reaching insecure levels. For instance, this could be done by a user reading an information at a secret level and then writing it at a public level. This transfer of information from secret to public level is called a flow. Such flows represent one of the main limitations of mandatory policies, which led to the emergence of new access control issues which will be further discussed in the inference problem section of this chapter.

2.3.2.3 Role-Based Access Control (RBAC)

RBAC [Ferraiolo 1995, Sandhu 1998] introduces the concept of role that lies between subjects and objects. A role is a set of privileges on objects. Intuitively, if a user possesses a role having privileges on an object then she/he could access such an object. Basically, each user is assigned a set of roles and each role has privileges on a set of objects. RBAC is a very popular model and has been deeply investigated [Ahn 1999, Fernandez 1994, Jaeger 1996, Sandhu 2000, Sandhu 1996b]. This model offers more flexibility than DAC and MAC models. Indeed, according to [Osborn 2000] RBAC could simulate both DAC and MAC models. Different concepts have been introduced to enrich RBAC with new functionalities. The most popular concepts are the following:

- Hierarchical roles: This feature allows the definition of a hierarchy between roles. This hierarchy captures inheritance relationship between roles. This means that a role could inherit all privileges of another role. This feature aims at facilitating role administration as well as role assignment to users.

- Least privilege: This feature aims at providing users only privileges that are needed to perform their tasks. In other words, depending on how roles are defined, RBAC could offer fine access granularity. This fine granularity allows minimizing the risk of data misuse.

- Separation of duties: This feature aims at preventing users from getting too much authorizations. Indeed, some policies could state that critical tasks need to be launched by at least two persons. Even though a user possesses both roles for launching the task, she/he will not be allowed to perform sensitive tasks alone. Separation of duties could be

enforced either at design time by defining conflicting roles or at runtime by checking activated roles when queries are evaluated.

- Constraint enforcement: This feature allows adding some restriction on roles. For instance, a policy could state that a particular role could only be activated by some selected users.

We note that these features are not required to define an RBAC model. Various RBAC models have been defined to enforce only some of these four features.

2.3.2.4 Attribute-Based Access Control (ABAC)

The attribute-based access control is a model that aims at offering more flexibility than previous models. In such a model a user is represented by a set of features characterizing her/his profile. A constraint is associated with a resource. The access control is enforced as follows: if the user profile satisfies the access constraints associated with a resource then the access is allowed. Otherwise, the access is denied. Different names were used to describe such an access control in the literature. It is referred to as attribute-based access control in [Wang 2004] where a logical framework is used to enforce such a model. In [Al-Kahtani 2004] it is referred to as rule-based where rules are used to enforce constraints in an RBAC model. The same concept is referred to as credential-based access control in [Bonatti 2000]. The flexibility of this model made it popular in open environments [Yuan 2005] and distributed systems [Hur 2011]. We note that this model could simulate any of the classical access control models. Indeed, it could be viewed as introducing an additional layer between the user and the access control entities. For instance, it could simulate RBAC if a set of attributes allows a user acquiring a particular role. The same reasoning applies also to MAC. Indeed, a set of attributes could allow a user acquiring a security level.

2.3.3 View-Based Access Control (VBAC)

View-based approaches aim at providing a general framework to enforce different kinds of policies. In view-based access control, the idea is to define a

view representing some piece of data. Then, a user is allowed or prohibited from accessing this view. Depending on how views are defined this approach could enforce DAC, MAC, RBAC and even ABAC policies. This approach has been conceived for the relational model but its rational could be applied to other models (*e.g.,* XML [Damiani 2002]). In this approach, a policy is defined by a set of views. We refer to such views as authorizations views. These authorization views enforce the abstract rules defined by the policy. View-based access control could be classified into two categories. The first one [Motro 1989, Rosenthal 2000] rewrites the user query to provide only the answers that are authorized to the user. The second one [Rizvi 2004] answers the query if the user has privileges on the whole results. Otherwise, no results are returned. The motivation in the first approach is maximizing the results returned to the user. The authors of the second approach argue that rewriting the query could induce some mistakes in the user exploitation of the results. Indeed, the user could believe the answer contains all results while only parts of them have actually been returned as an answer. We note that these two approaches express different point of views on how a system should evaluate queries. Both of these approaches are interesting and relevant for some specific applications. Indeed, application willing to sacrifice some accuracy while gaining availability will choose the rewriting approach. Applications that do not support any answer alteration would rather choose the approach without rewriting the results.

2.3.3.1 View-based access control with rewriting of the results

This approach, first introduced by [Motro 1989], aims at maximizing the query results. The idea is to rewrite a query using a set of authorization views. This step could be viewed as a filtering step. Indeed, it returns only the accessible part of the results. In this model, when a user issues a query Q against the database schema, if there exists a query Q' expressed in terms of the authorization views and Q' is included in Q (and Q' is maximal, *i.e.,* there is no Q" such that Q' is included in Q" and Q" is included in Q), then, Q' is evaluated and its results are returned instead of the results of Q.

2.3.3.2 View-based access control without rewriting of the results

In this approach, introduced by [Rizvi 2004], a query rewriting is also performed. The main difference with respect to the previous approach is the interpretation of the rewriting. In this model, when a user issues a query Q against the database schema, if there exists a query Q' expressed in terms of the authorization views and Q $\equiv$ Q' then Q is said to be valid and it is evaluated. Otherwise, the query Q is not valid and it is denied.

2.3.4 Access control and associations of attributes

A concept that we consider of interest is the one of controlling the access to an association of attributes. Intuitively, this problem states that two pieces of information could be non sensitive when considered separately, but they could become sensitive when they are considered together (*i.e.,* as a whole). To illustrate this concept, we consider the following example. Assume a relation *Patient* having two attributes. The first one is *SSN* (Social Security Number) and the second one is *Disease*. Each attribute considered separately is not sensitive. Indeed, having the list of all *SSN* could be unclassified. Having the list of all diseases is not sensitive as well and is unclassified. The main problem here is that, when put together, *SSN* and *Disease* should not be unclassified. The association of the two attributes should be kept secret. In this example we discuss a multilevel policy, but the reasoning applies for any arbitrary model. This example highlights the difficulty of composing policies. Indeed, even if we have a policy for one attribute and we have the same policy for another attribute, it does not mean that the same policy could be attached to the association resulting from the joint use of both attributes.

In [Clifton 2004], the importance of access controls over associations of attributes has been established. This work discusses at an abstract level issues that could arise when pieces of different information are put together. The authors propose view-based approach, defined in SQL notation, to capture such associations.

Another interesting interpretation of controlling an association is the one of inferences. Indeed, if the results of two queries are not sensitive when taken separately, this does not mean that their combination is not sensitive as well.

Although some works on access control of an association have been proposed, we believe that many access control approaches overlook such a problem.

In our work, we pay particular attention to this problem and discuss its relevance with respect to the inference problem in a data integration scenario.

2.3.5 Synthesis on access control models

After presenting the main access control models and mechanisms, we shall discuss their strengths and their limitations. Indeed, these models (or the models based on them) are widely used and they are considered important for database systems. In particular, from the time access control has been identified as a crucial component to achieve security.

2.3.5.1 What did these approaches achieve?

Access control has always been an essential part of information systems. Different approaches have been designed to protect data from unauthorized access. These approaches started by simple ones that have a coarse granularity and offering basic mechanisms. Then, new approaches have been introduced to reach finer granularity and offering better guarantees. Indeed, theses classical approaches (DAC, MAC and RBAC) have provided strong security guarantees by being able to prove some properties. From an enforcement point of view these models include ad-hoc mechanisms that targeted a specific model. Later on, more general mechanisms have been introduced. The most popular one is the view-based access control. Indeed, view-based approach could simulate any classical access control model. Also, it targets one of the widely used data model, namely, the relational model.

2.3.5.2 What are their limitations?

These approaches reach their objectives by allowing definition and rigorous enforcement of different kinds of policies. Nevertheless, they show some limitations. Some of these limitations are specific to a particular model and thus one could deal with them by choosing a more suitable model. The main limitation which is shared by all these models is the management of indirect

access. Indeed, these models have been designed to protect data from direct access. A direct access occurs when a query is issued to directly access some piece of data. An indirect access occurs when a set of queries are combined to disclose sensitive information. The problem of detecting and dealing with indirect access is referred to as the inference problem. In the next section we introduce the inference problem and describe the approaches that have been proposed to tackle this problem.

2.4 Inference problem

The inference problem (see [Farkas 2002] for a survey), in an access control context, refers to the ability of a malicious user to synthesize sensitive information from a combination of non sensitive information.

The inference problem is motivated by some limitations inherent to classical access control mechanisms. Indeed, as discussed in the previous section, these models do not consider indirect access to information. An indirect access occurs when a malicious user is able, by combining a set of non sensitive accesses, to infer sensitive information. Classical access control models have not been designed to protect data from inferences. Even though they achieve good guarantees for direct access, they fail to deal with such inferences.

We have introduced the inference problem in an abstract manner. We will discuss different types of inferences that have been studied in the literature. Indeed, malicious users could use different techniques to infer prohibited information. For each of these techniques specific approaches have been proposed to deal with. Usually, each solution deals with a particular attack. An attack is the method used to infer some prohibited information. This is due to the fact that depending on the configuration of a system different attacks could occur. A simple example of such a configuration is the one of the query language provided to users. Indeed, allowing users to issue aggregate queries and not only conjunctive queries could widen the space of attacks. Another parameter is the kind of policies that is established. Depending on the nature of the attack, proper solutions have been proposed.

Historically, first inference attacks appeared on statistical databases [Adam 1989]. In these systems, users are allowed to obtain information about

data trends on populations but are not allowed to access data on individuals. The idea of statistical attack is to combine a set of statistical information to infer information about a particular individual.

The second kind of attack that appears is the one that rests on the exploitation of semantic constraints [Su 1987, Thuraisingham 1987]. In these attacks, malicious users take advantage of semantic relations between datasets to infer prohibited information. There has been an extensive amount of work devoted to this issue from different perspectives.

With the emergence of data mining approaches new inference threats have been identified. This led to the design of methods that balance the utility of data mining approaches with data confidentiality.

More recently, de-identification attacks have been recognized [Samarati 1998]. Indeed, with the emergence of data publishing approaches new threats of data confidentiality appeared. In data publishing approaches, data is anonymized before it is released. The issue that arises is that classical anonymization techniques could have some limitations. Indeed, classical approaches remove identifying attributes before releasing the data. It has been shown [Sweeney 2002a] that inferring individual identity could be achieved even when identifying attributes are removed.

Next, we discuss each of these attacks separately and the solutions that have been proposed to deal with them.

2.4.1 Statistical attacks

Statistical attacks have been extensively studied (see [Adam 1989] for a survey). These attacks occur when a policy is designed to protect data about individuals (*e.g.,* salary of a particular employee) while providing access to aggregate information (*e.g.,* average salary within a company). This kind of attack mainly results from overlapping between query results. A number of methods have been proposed to deal with statistical attacks. These methods could be classified into three categories: query restriction [Denning 1983], data perturbation [Schlörer 1981] and output perturbation [Fellegi 1974].

Query restriction [Denning 1983, Fellegi 1972, Friedman 1980, Hoffman 1970] could be performed by constraining the number of tu-

ples that are used to construct query results. This induces the definition of an arbitrary threshold for the number of tuples involved in the query. Another way to achieve query restriction is to check for overlapping between queries. Indeed, if two queries share a large amount of tuples, their results could induce prohibited information disclosure. Another approach to achieve query restriction is cell suppression. This approach removes some sensitive cells from the database before query evaluation. A last method is query auditing. This approach needs to record all user queries. When a new query is issued, the combination between the new query and past queries is checked before evaluating the new query.

Data perturbation [Reiss 1984, Schlörer 1981, Liew 1985, Lefons 1983, Warner 1965, Traub 1984] modifies data in such a way that it limits data inferences. There are two methods devoted to data perturbation. The first one, called probability distribution, aims at changing data while keeping the same distribution. The goal of this approach is to preserve the results of aggregate queries even though the data is changed. In the second approach of data perturbation, data is modified by adding noise to it. For example, numerical values could be modified by adding to it a constant parameter. These approaches induce a bias problem which means that data provided to users could induce some errors. Indeed, means or frequencies of modified data could differ from the real values.

Output perturbation [Denning 1980, Beck 1980, Achugbue 1979, Fellegi 1974, Ozsoyoglu 1985] modifies query results in order to prevent from sensitive information disclosure. An example of output perturbation is the use of sample queries where a subset of data is selected randomly and queries are evaluated against this sample. Accuracy of the results is the main issue in such approaches.

2.4.2 Semantic attacks

Approaches dealing with semantic attacks tackle some issues that arise in classical access control mechanisms. In particular, they consider indirect accesses in addition to direct access. The idea in such approaches is to detect indirect access that could induce prohibited information disclosure. Such indirect access is referred to as an inference channel. These approaches aim at achieving

two goals. The first one is inference channel detection. The second one is to provide mechanisms able to deal with these inference channels. Semantic attacks have been identified as a serious threat to data security at a time where multilevel security policies were much popular [Goguen 1984, Su 1987]. This induces that much of the work and approaches have been designed to this kind of policies. We note that the rationale behind these approaches applies to other models. In semantic attacks, relation between data is used by malicious users to infer sensitive information. These relationships exist regardless of access control model that is used.

First attempts to study the inference induced by semantic relationship have been reported in [Goguen 1984, Morgenstern 1988]. For instance, [Morgenstern 1988] defines a function based on information theory to characterize the inference between two objects X and Y. This function describes the amount of information that one would know about Y given the knowledge of X. This approach displays a (major) limitation. Indeed, it has been shown that such a function is impossible to define in most cases.

Other approaches have been introduced to tackle specific attacks. We could classify such semantic attacks into three categories: inference by applying constraints on queries, inference using metadata and inference using value constraints.

2.4.2.1 Inference from constraints on queries

This kind of inference occurs when a malicious user could infer some prohibited information using specific constraints on a query. This attack assumes that data labeling is performed at the attribute level. This means that each attribute could have a specific security level. To illustrate this attack we present the example of [Meadows 1987]. In this example there are two relations: an unclassified one, EP, and a secret one, PT. EP contains the Employee-Name and Project-Name attributes while PT contains Project-Name and Project-Type attributes. Assume a non authorized user issues the following query:
SELECT EP.Employee-Name
FROM EP, PT
WHERE EP. Project-Name= PT. Project-Name
AND PT.Project-Type = 'SDI'

Although this query only returns results about unclassified information, it does induce prohibited information leakage. Indeed, information about the secret relation have been indirectly accessed. This kind of inference is quite easy to identify and to deal with. Indeed, one only needs to rewrite the user query into another query using only authorized attributes. Nevertheless, this example pointed out the importance of paying particular attention to inferences in order to preserve data confidentiality.

2.4.2.2 Inference resulting from the combination of data with metadata

This kind of inference occurs when malicious users take advantage from metadata to infer prohibited information. Early work on such attacks considered the role played by key integrity[2] to disclose sensitive information. It has been generalized to consider functional dependencies [Su 1987]. To illustrate this kind of inference we present the example given in [Su 1987]. In this example, a relation *Employee* is defined. This relation has three attributes: *Name*, *Rank* and *Salary*. The attributes *Name* and *Rank* are considered unclassified while *Salary* is considered secret. In this example it is assumed that employees having the same rank have the same salary. In other words, a functional dependency exists between rank and salary where rank identifies salary. Now, assume a user issues a query on *Name* and *Rank*. This query does not access any secret information and thus it is evaluated. The malicious user could then infer the salary of each employee. This is an interesting example that shows that functional dependencies could be used to infer sensitive information. Even though in this case functional dependencies could be used to infer prohibited information, it is only the case because there is a function that for each rank computes its corresponding salary.

The authors of [Su 1987] provide algorithms to propagate the security level of attributes that could be inferred from other attributes. In the previous example the idea is to associate the level secret to the attribute rank.

Please note that in the general case accessing the left hand side of a functional dependency does not allow to have access to the value of the right hand

[2]Key integrity [Chen 1976] is a constraint stating that a set of attributes constitutes a key. A key value should be unique in a given relation.

side. To illustrate this, we consider the following example:

Assume we have a relation that stores the social security number (*SSN*), the *AdmissionDate* and the *Disease* of patients. We further assume that *SSN* and *AdmissionDate* determine the attribute *Disease*. In other words there is a functional dependency between *SSN* and *AdmissionDate* in one hand and *Disease* on the other hand. Although this functional dependency exists, a user accessing to the attributes *SSN* and *AdmissionDate* could not infer anything about the *Disease*. This is due to the fact that there is no function that takes *SSN* and *AdmissionDate* as input and returns the corresponding *Disease*.

2.4.2.3 Value constraints

A value constraint attack occurs when a user could infer information about the domain of a sensitive attribute. The authors of [Meadows 1988] illustrate such an attack with the following example. Assume there are two attributes A and B. A is unclassified while B is secret. Assume that the database has the following constraint: $A+B < 20$. This constraint could be unclassified since it concerns A. Although this constraint does not provide any value of B, it could disclose information on the value that could occur on B. In particular, the malicious user knows the values taken by A (since A is unclassified). Taking advantage of the previous constraint, the same user could infer the values that could be taken by B.

Now, we discuss how these inference channels are dealt with.

2.4.3 Methods to deal with semantic inferences

There has been a lot of work devoted to detect and deal with inference channels. The proposed approaches could be classified into two categories. In the first category, all processing is achieved at design time. In this category, the security policy is modified in order to avoid inferences. In the second category, the processing is achieved at runtime. In this approach, when a query is issued, the system checks if it leads to some inference. If so, the query is denied. Some of the runtime approaches maintain a log of all past queries and checks for inferences by combining current and past queries.

2.4.3.1 Design time approaches

Design time approaches argue that since the inference problem is a complex one, it is better to achieve all consuming processing offline before the system is launched. This modification usually is performed on the policy in such a way that inferences could not occur. Modifications could also target the database schema. Indeed, if the granularity of the access control is enforced at the table level then modifying the schema could prevent from some inferences.

Now, we discuss two major examples of design time approaches, namely, the work of [Delugach 1996] and the one of [Su 1987]. In [Delugach 1996], a conceptual graph based approach has been used to capture semantic relationships between entities. The authors show that this kind of relationships could lead to inference violations. The main drawback of such an approach is that it could induce some false detection. As discussed previously, the authors of [Su 1987] consider the inference problem using functional dependencies. In this work, the authors assume that the user knows the mapping between the attributes of any functional dependency.

2.4.3.2 Runtime approaches

Runtime approaches aim at detecting inference when evaluating queries. This approach is motivated by the objective of increasing data availability. Indeed, it denies a query only when it is going to induce an inference. Most of these approaches are history-based and require managing huge log files.

Now we discuss two important runtime approaches: [Brodsky 2000] and [Thuraisingham 1987]. In [Thuraisingham 1987], both queries and authorization rules are specified using first order logic. While the inference engine considers the past queries, the functional dependencies are not taken into account. In [Brodsky 2000], a history-based approach has been considered for the inference problem. The authors have considered two settings: the first one is related to the particular instance of the database. The second one is only related to the schema of both relations and queries.

2.4.3.3 Comparison of the two methods

By using the design time approach one improves query execution time with respect to run time approaches. Indeed, no extra processing is needed to evaluate the query. Runtime approaches in the other hand induce checking for inference when evaluating the query. This could induce slowing down query processing. In particular, when inference are checked with respect to past queries. Indeed, in this case a huge log file needs to be considered for each new query.

Also, runtime time approaches could induce a non deterministic access control behavior. Indeed, depending on past queries the current one is evaluated or denied. This means that the system could take different decisions for two users having the same credentials. Consider a first user having issued past queries that could be combined with the current query and leading to deny the current query. Now consider a second user, having the same credentials, but the current query could not be combined with her/his past queries. In this case the current query is evaluated. We observe that issues related to non deterministic access control do not appear in design time approaches.

One can note that, runtime approaches maximize data availability. Indeed, a query is denied only if it could induce an inference. Design time approaches are too restrictive. Since queries are not known in advance, these approaches take the appropriate decision of denying the access if there is any doubt. As discussed, these two methods differ on how to deal with inference. But they agree on the importance of detecting inferences. To this end, they aim at deploying efficient mechanisms against inference channels. We also observe that the two methods could be combined. This could be achieved by dealing with some inference at design time and with other at runtime. This combination increases the availability of data while reducing the size of the log file.

2.4.3.4 Synthesis on semantic attacks

The approaches dealing with semantic attacks show the relevance of the inference problem to data protection. Indeed, several alternatives allowing malicious users getting prohibited information have been identified. Also, these approaches highlight the limits of classical access control models. Indeed, these models do not provide any mechanism to deal with inferences. When

it comes to the solutions we discussed so far, we could clearly state that the inference problem using semantic constraints is not completely solved. All the proposed approaches have their drawbacks. Even when we overlook problems arising from external knowledge, these approaches are either too restrictive (*e.g.,* [Delugach 1996]). We mean by too restrictive approaches that detect false inference channels which induces over classification of objects. Other approaches are not realistic in practice (*e.g.,* [Morgenstern 1988]). These approaches require information about data that is not easy if not impossible to determine. Finally, some approaches are specific to particular cases (*e.g.,* [Su 1987, Thuraisingham 1987]).

2.4.4 Data mining based attacks

Data mining algorithms allow the discovery of interesting new information from data. Usually, this new information is useful and relevant for many real world applications. Although the utility of data mining approaches have been established, these approaches could help malicious users infer and discover sensitive information. In [Clifton 1996], the authors pointed out the threat that data mining could represent for sensitive information. An interesting discussion of different data mining approaches from privacy point of view could be found in [Verykios 2004]. Different approaches have been developed to protect data against data mining algorithms. Some of these approaches share common rationale with approaches used in statistical inference.

Data mining approaches for preserving data confidentiality highlight, even more than statistical and semantic approaches, the trade off that exists between data confidentiality and data utility. Finding an appropriate balance between these two measures represents one of the main challenges of the inference problem.

2.4.5 De-identification attacks

A de-identification attack occurs when data which is supposed to be anonymized could still be used to infer information about one particular individual. The first work on this issue is the one of *k-anonymity* [Samarati 1998, Samarati 2001a, Sweeney 2002a, Sweeney 2002b] where the

problem is identified and solutions are provided. An important concept in the case of this problem is the one of quasi identifier. A quasi identifier is a set of attributes that could almost identify an individual. The idea of k-anonymity is to make sure that each value of a quasi identifier corresponds to at least k individuals. The two main methods to achieve k-anonymity proceed either by generalization or by suppression.

In the generalization method, values of quasi identifier attributes are transformed into less specific values. Depending on the nature of the attribute, different generalization methods could be considered. For example, a Zip code could be generalized by removing one or two digits. A numerical attribute (*e.g.,* age) could be generalized by replacing the age value by an interval containing this value. Category value could be generalized using a hierarchy. In this case, the idea is to replace the category value by its ancestor according to the hierarchy. Generalization allows grouping tuples by quasi identifier and make sure that there are at least k members in each group.

In the suppression method, values of attributes are removed. As pointed out by [Sweeney 2002a] both methods could be combined. Indeed, suppression could be useful to decrease the amount of generalization used. Indeed, when a sensitive value is removed one does not need to generalize the corresponding quasi identifiers. In some cases combining suppression and generalization could preserve a better data utility than generalization alone.

Other approaches have been derived from k-anonymity like *t-closeness* [Li 2007] , *p-sensitive k-anonymity* [Truta 2006] and *l-diversity* [Machanavajjhala 2007]. These approaches are built on the same intuition behind k-anonymity and have been designed to remedy its shortcomings.

As pointed out by [Domingo-Ferrer 2008], all these approaches (k-anonymity, t-closeness, p-sensitive k-anonymity and l-diversity) are considered to be non convincing. Indeed, the authors argue that the balance between data protection and data utility is not achieved. Some approaches (*e.g.,* k-anonymity) are vulnerable to attack, whereas some others (*e.g.,* t-closeness) greatly damage data utility.

2.4.5.1 Synthesis on de-identification approaches

De-identification techniques are very popular nowadays due to their relevance to data publishing on the web. Nevertheless, these techniques show some limitations. The main limitation is the loss of data utility. Indeed, all proposed solutions to achieve k-anonymity or its derivations induce a loss of data accuracy (*e.g.,* generalization) or a loss of part of the data (*e.g.,* suppression) or even both. Nonetheless, these approaches could be relevant to some applications. They also show that inferences are still an important problem for data protection.

2.4.6 Synthesis on inference approaches

Now, we summarize the four main approaches that have been proposed over the last four decades to deal with different ways of achieving inference. Historically, these four categories appear in the following order: statistical attacks, semantic attacks, data mining based attacks and de-identification attacks. Although these solutions have been designed independently from each other and targeting different purposes, some global trend could be observed.

2.4.6.1 What do they agree on?

These approaches mainly agree on how to detect potential inferences. Indeed, most of these approaches identify that sets of queries and metadata are the main concepts inducing inferences.

- By considering a set of queries: Most of these approaches recognize the importance of considering a set of queries as the main threat for data confidentiality by means of inferences. Indeed, classical access control models provide strong guarantees when dealing with a single query (or when query combination is overlooked).
- By considering metadata: Most of these approaches identify metadata and external knowledge as a relevant way for malicious users to access prohibited information. Thus, they propose methods able to detect if metadata could be used to access some particular prohibited information.

2.4.6.2 What do they disagree on?

These approaches mainly disagree on the way to deal with detected inferences. Some approaches modify data or returned results. Other approaches modify policy attached to data. Another disagreement is concerned with the enforcement. Indeed, some approaches perform enforcement at design time while others are runtime-based enforcement.

Data modification or policy modification

Some of the solutions are performed at the data level. Indeed, they either modify the data or modify the query results. Other solutions modify the policy attached to data. Data modification is efficient to provide some guarantees regardless of the policy, in particular if the policy is defined by other systems (*e.g.,* data publishing). Policy modification is efficient for applications that handle their own policy. Also, some useful information could be lost during data modification. Thus, policy modification is more relevant for applications that do not tolerate any data accuracy loss.

Design time or run time

Some of the solutions are achieved at design time either by modifying data or the policy protecting this data in such a way that no inference could occur. Other solutions are achieved at runtime. Usually, these approaches are history-based and consider combination of current and past queries to decide if the current query should be evaluated.

In this thesis, we focus on semantic attacks. In particular, we focus on those caused by the exploitation of functional dependencies. We take the side of policy modification and propose two solutions: design time based approach and runtime based approach. We then compare these solutions in our context.

2.5 Policy composition

Policy composition approaches aim at combining a set of policies defined independently. Their goal is to produce a global policy from a set of local policies. In the following, we discuss some approaches dealing with integration of security policies. We could organize these approaches into two main categories.

The first one is general purpose approaches which usually define an abstract language to perform the composition. The second one includes model specific approaches which are specifically designed for one particular access control model. Next, we elaborate on both categories.

2.5.1 General purpose approaches

In [Bonatti 2002], an algebra is defined to model how policy composition could be performed. In this work all policies to be combined are translated into the algebra. The authors define an authorization as a triple (*subject, object, action*). This definition aims at preventing from being specific to any classical access control model. Then, different methods to perform the composition are discussed. Another abstract way to perform policy composition is the one presented in [Jajodia 2001]. In this approach policy combination is also considered. The authors consider both positive and negative authorizations. This setting induces conflict management during policy composition. Next, we present some popular methods introduced by [Bonatti 2002] and [Jajodia 2001] for composing two policies.

- Addition: This method constructs a global policy which corresponds to the union of the two policies. In other words, if one local policy allows the access then the global policy also allows the access.

- Conjunction: This method constructs a global policy from the intersection of two policies. In this method, an access is allowed at global policy only if all policies allow the same access.

- Subtraction: This method constructs a global policy from the first policy while removing all authorizations that appear in the second policy.

These methods could be used together to combine a set of policies. Indeed, each method constructs a policy which could be combined with another policy and so on until all policies are combined.

2.5.2 Model specific approaches

These approaches target a specific access control model and assume all policies to be combined use the same model. Then, a method is used to perform policy composition within this model. In [Shafiq 2005] and [Oliva 2001] a mapping between the access control entities (RBAC and MAC respectively) is performed. This mapping rests on another mapping at the data level. A method dealing with the issues that arise from the previous mapping (*e.g.,* cycle introduction) is given. In [Shehab 2005] and [Bonatti 1997] links between the access control entities (RABC and MAC respectively) are manually introduced. The integration of the security policies rests on these links.

2.5.3 Synthesis on policy composition approaches

General purpose approaches provide a framework that allows combining policies. Also, this abstraction allows providing some properties on the resulting composition. Model specific approaches are more ad-hoc and target one model. In the other hand, the specific approaches take advantage of semantic relationships between data to compose policies.

These approaches for composing policies were not designed to be used in the context of data integration and the inference problem. These approaches aim at providing users of one system with access to data of another system, but do not consider how the access to combined data provided by different systems should be enforced. As discussed in the previous sections, combining data could induce prohibited information leakage. So, these approaches are not suitable for dealing with inferences. Nevertheless, these approaches could provide a preliminary global policy that needs to be enriched to avoid inferences.

In this thesis, we investigate the inference problem in data integration scenarios. We consider composition of local policies as a first phase, which needs to be completed, towards the definition of a global policy that avoids inferences.

2.6 Relation between the different (studied) dimensions

In this chapter, we discussed several concepts that we believe should be considered together when one aims at securing data integration systems. Indeed, a first dimension is the data one. In particular, how the schema of the mediator is constructed. As discussed in the policy composition section the way data is combined could have an impact on the way policies are composed as well. The second is the one of access control model. In this part, we observed that these traditional models have been designed to deal with direct access but fail to preserve data from indirect access. We also pointed out the importance of choosing an appropriate access control model to be enforced. Indeed, some recent models (*e.g.,* ABAC) are more flexible and could simulate previous models. Another criterion for choosing an access control model is granularity. Granularity is an important aspect in access control. It refers to the size of an object that can be controlled. The more the granularity is finer the more the access control is precise. We noted that some enforcement mechanisms (*e.g.,* VBAC) offer finer granularity which thus allow more precise access control. The third dimension is inference. In this part, we discussed different methods we identified in the literature. The methods allow malicious users to synthesize prohibited information. We observed that dealing with inferences induces some loss of data utility. We also characterized the fact that many concrete methods of inference involve the combination of a set of queries and consider the benefit in using the semantic constraints (metadata) that hold on data. Finally, we considered policy composition approaches. We noted that these approaches were not designed to deal with inferences. Even though we could expect more inferences in a composed system since malicious users are provided with more data that could be composed.

Chapter 3

Problem statement

Contents

3.1 Introduction

In this chapter, we start by motivating our work and we provide an abstract statement of the problem. In order to be able to precisely define the problem,

we give a set of definitions and concepts that will be needed throughout the next parts of this manuscript. We mainly focus on definitions related to relational databases and data integration, access control and authorization, and some specific definitions on functional dependencies and graphs. We discuss our approach with respect to the four dimensions we described in the previous chapter. Finally, we describe, at an abstract level, the methodology we propose.

3.2 Motivation

Our main objective is to secure data integration systems at the access level. This is motivated by the fact that data integration approaches offer convenient ways to combine data provided by heterogeneous sources. Nevertheless, these approaches do not clearly and simply consider how access control should be defined and enforced in such systems. Enforcing access control at the mediator level is a challenging task. Indeed, one needs to preserve source policies as well as preventing from inferences. We aim at defining a representative global policy to be attached to the mediator. This is mainly motivated by the following arguments:

1. **Access control:** When enforcing access control at mediator level one should make sure that the source policies are preserved. This means that if a user is not allowed to access some piece of information at a source level, she/he should not be able access such an information at the mediator level. Although this property seems natural and intuitive, it is essential to guarantee it. Indeed, sources will be reluctant to participate in a system that discloses their sensitive information. Ensuring this property requires particular care when combining policies provided by different sources. Indeed, this property requires to consider, at the mediator level, the most restrictive source policies. This is highlighted by the following example. Consider two sources having common information. If one source forbids the access for some users and the other source does not, the mediator needs to forbid the access in order to preserve both policies. In doing so, access control could encourage sources participating in data integration systems. This constitutes one of our motivations: promoting data integration systems that have shown great

utility for users by gathering distributed data into one entry point. We believe that the more access control guarantees are provided the more data integration could be considered for systems managing sensitive information.

2. **Information disclosure:** As discussed in the related work chapter, combining access control policy is not sufficient for securing systems. Indeed, combination of information could allow malicious user to infer prohibited information. Such inferences are more likely to occur in data integration systems. This is due to the fact that data integration systems combine information from different sources. Each source being designed independently from the others, administrators could not anticipate data combination with other sources. This setting constitutes our second motivation, namely, preventing inferences at the mediator level.

3. **Query evaluation:** Taking access control decision about query evaluation could decrease mediator load as well as network communication and sources load. Indeed, if no policy is defined at the mediator level one needs to rewrite user queries (which could be costly) then send them to the sources (which increases network communications). After some processing the sources might reject the query because of insufficient credentials. All these issues could be avoided by attaching a policy to the mediator. In doing so, access control decision is taken at the mediator level and only queries that are expected to be evaluated are rewritten and sent to the sources. If a query is denied at the mediator level there is no need to rewrite it nor to send it to sources.

4. **Assist the administrator defining the mediator's policy:** As previously discussed, defining a global policy for the mediator could improve both access control enforcement and data management (query evaluation). Unfortunately, to our knowledge, there is no automatic approach in the literature that achieves this goal. We believe that defining a global policy manually is a difficult task that could be tedious, time consuming and error prone. This issue represents another motivation of our work, namely, elaborate an approach that allows to state and derive the global policy that should be attached to the mediator while preserving the policies of the sources and preventing from malicious inferences.

3.3 Problem statement

Now, we can define the problem we consider as follows:

"Given a set of sources, each one having its own authorization policy, a defined global schema together with a mapping between the global schema and the local (source) schemas, what policy should be attached to the mediator while preserving the local (source) policies and preventing from inferences?"

3.4 Preliminaries

The previous definition refers to a few concepts. Those concepts need to be formally defined in order to be able to formally define the problem.

3.4.1 Definitions related to relation, queries and integration

Definition 1 (Clause) *[Abiteboul 1995]*

A clause has the form $L_0 : -L_1, ..., L_n$ *where each Li is a literal, of the form* $p_i(t_{i1}, ...t_{ik})$ *such that* p_i *is a predicate and each* t_{ij} *is either a variable or a constant.*

Definition 2 (Local and virtual relations) *A local relation is a relation defined at the source level. A virtual relation is a relation defined at the mediator level.*

Definition 3 (Global schema) *A global schema is a schema of the mediator. It is a specification defined by a set of clauses. Each clause defines a virtual relation. The left hand side of a clause denotes a virtual relation whereas the right hand side is a set of local and/or virtual relations.*

Definition 4 (Conjunctive query) *[Abiteboul 1995]*

A conjunctive query is an expression of the form $R_1(u_1) :- R_2(u_2), ..., R_n(u_n)$, *where* $n \geq 1$, $R_1, ..., R_n$ *are relation names and* $u_1, ..., u_n$ *are free tuples of appropriate arities. Each variable occurring in* u_1 *must also occur in at least one of* $u_2, ..., u_n$.

Here the notation $R_i(u_i)$ *where* $u_i = <A_1, A_2, ..., A_m>$ *is a shorthand for* $R_i(A_1, A_2, ..., A_m)$

Definition 5 (GAV integration approach) *In GAV (Global As View) integration approach, each virtual relation of the mediator is defined using a conjunctive query over the local relations.*

3.4.2 Definitions related to functional dependencies

Definition 6 (Functional Dependency) *[Levene 1999]*

A functional dependency over a schema R (or simply an FD) is a statement of the form:

$R : X \rightarrow Y$

(or simply $X \rightarrow Y$ *whenever R is understood from the context), where* X, $Y \subseteq$ *schema(R). We refer to* X *as the left hand side (LHS) and* Y *as the right hand side (RHS) of the functional dependency* $X \rightarrow Y$.

Definition 7 (Satisfaction of a functional dependency) *[Levene 1999]*

A functional dependency $R : X \rightarrow Y$ *is satisfied in a relation* r *over* R, *denoted by* $r \models R : X \rightarrow Y$, *iff* $\forall\, t_1, t_2 \in r$ *if* $t_1[X] = t_2[X]$, *then* $t_1[Y] = t_2[Y]$.

Definition 8 (Closure of a set of functional dependencies) *[Levene 1999]*

The closure F^+ *of a set of function dependencies* F *is the set of all FDs that could be derived from* F *using Armstrong rules [Armstrong 1974].*

Definition 9 (Pseudo transitivity rule) *[Levene 1999]*

The pseudo transitivity rule is an inference rule that could be derived from Armstrong rules [Armstrong 1974]. This rule states that if $X \rightarrow Y$ *and* $YW \rightarrow Z$ *then* $XW \rightarrow Z$.

Without loss of generality we consider functional dependencies having only one attribute in their RHS. A functional dependency of the form $X \rightarrow YZ$ could always be replaced by $X \rightarrow Y$ and $X \rightarrow Z$ by using the decomposition rule [Levene 1999] which is defined as follows: *if* $F \vdash X \rightarrow YZ$, *then* $F \vdash X \rightarrow Y$ *and* $F \vdash X \rightarrow Z$.

3.4.3 Definitions related to graph

Let $G = (V, E)$ be a graph such that V is a set of nodes and E is a set of links.

Definition 10 (Path) *A path on the graph* G *is a sequence of distinct nodes* $v_1, v_2, ..., v_k$ *of* V *such that the links* $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$ *belong to* E.

3.4.4 Definitions related to access control

Definition 11 (Malicious user) *A malicious user is a user of the system that aims at obtaining sensitive information. In other words, a malicious user is a legitimate (i.e., authenticated by the system) and curious user. This concept is introduced to model insider attacks.*

Definition 12 (Inference) *In the context of access control, an inference is the process that a malicious user could perform in order to obtain a prohibited information.*

Definition 13 (User profile) *A user profile is a set of predicates that characterize the user.*

Examples are: $Role = Doctor$, $Age = 20$.

Definition 14 (Constraint) *A constraint is a formula over a set of predicates.*

An example is: $Role = Doctor \wedge Age > 40$.

Definition 15 (Datalog rule) *[Abiteboul 1995]*

A (datalog) rule is an expression of the form $R_1(u_1) :- R_2(u_2), ..., R_n(u_n)$, *where* $n \geq 1$, $R_1, ..., R_n$ *are relation names and* $u_1, ..., u_n$ *are free tuples of appropriate arities. Each variable occurring in* u_1 *must also occur in at least one of* $u_2, ..., u_n$.

Definition 16 (Authorization policy) *An authorization policy is a set of authorization rules. An authorization rule is a view that describes the part of data that is either allowed or prohibited to the user.*

Definition 17 (Authorization rule) *An authorization rule will be expressed using an augmented datalog rule. This augmentation consists in adding a set of predicates characterizing the users to whom the authorization rule applies. An authorization rule is defined by a view composed of two parts : a query part and a constraint part. The query part describes the piece of information that the view protects. The constraint part states which constraint should be satisfied to access the query part.*

An authorization rule defined by a view allows to control the access to an association of attributes. This concept is of interest since combining information (*e.g.,* by associating attributes) is common in a data integration context. Also, an association of attributes could be more sensitive than when each attribute is considered alone. For example, consider the association of SSN and $Diagnosis$. Each attribute, taken individually, is not sensitive. However, their association is sensitive.

Definition 18 (Positive authorization) *A positive authorization is an authorization where all predicates involved in the constraint part are positive.*

Example: $V_1(SSN, Diagnosis) : -Patient(SSN, Diagnosis, Doctor)$,
$\$Role = Doctor$.

This authorization rule allows doctors to access the association between SSN and $Diagnosis$.

Definition 19 (Negative authorization) *A negative authorization is an authorization where all predicates involved in the constraint part are negative.*

Example: $V_2(SSN, Diagnosis) : -Patient(SSN, Diagnosis, Doctor)$, $\neg(\$Role = Nurse)$.
This authorization rule forbids nurses from accessing the association between SSN and $Diagnosis$. Please note that $\neg(\$Role = Nurse)$ could be written as $(\$Role \neq Nurse)$. In our examples, we will use the latter notation.

Definition 20 (Open and closed policies) *[Bertino 1999]*

An open policy allows specifying only positive authorizations. In this case, all explicitly allowed accesses are authorized.

A closed policy allows specifying only negative authorizations. In this case, only non explicitly denied accesses are allowed.

In general, open and closed policies are equivalent from an expressivity point of view. In other words, one could translate an open policy into a closed one and the other way around. In our previous example $V_1(SSN, Diagnosis)$ and $V_2(SSN, Diagnosis)$ could express the same policy if the set of roles in the system is $\{Doctor, Nurse\}$. Indeed, in this case both V_1 and V_2 allow the access for doctors and denies it for nurses.

Definition 21 (Violating Transaction) *A violating transaction T is a set of queries such that if they are executed and their results combined, they will lead to disclosure of sensitive information and thus violating one or more authorization rules.*

3.5 Motivating example

In order to illustrate how inferences could occur in the data integration scenario we consider a healthcare scenario. We start by describing the local

(source) schemas and the (mediator) global schema. Then, we present a source authorization view which is propagated to the mediator. Finally, we show how users could make an inference leading to the violation of the previous authorization view.

Global as View Integration. We consider a data integration scenario where a GAV [Lenzerini 2002] approach is used to define a mediator over three sources. Particularly, we consider the sources S_1, S_2 and S_3 with the following local schemas: $S_1(SSN, Diagnosis, Doctor)$ contains the patient social security number (SSN) together with the diagnosis and the doctor in charge of her/him, $S_2(SSN, AdmissionT)$ provides the patient admission timestamp, $S_3(SSN, Service)$ provides the service (*i.e.,* the department) to which a patient has been assigned.

The mediator virtual relation, according to the GAV integration approach, is defined by using relations of the sources. To simplify the scenario, we consider a single virtual relation, but the same reasoning applies for a global schema composed by a set of virtual relations. In our example, the mediator will combine the data of the sources joined over the SSN attribute as shown by the following rule (3.1).

$$M(SSN, Diagnosis, Doctor, AdmissionT, Service) :- S_1(SSN, Diagnosis, Doctor), S_2(SSN, AdmissionT), S_3(SSN, Service). \quad (3.1)$$

Authorization Policies are specified by each source on its local schema and propagated to the mediator. In our example, we assume two categories of users: doctors and nurses. For S_1, doctors can access SSN and $Diagnosis$ while nurses can access either SSN or $Diagnosis$ but not their association (*i.e.,* simultaneously). The rule (3.2) expresses this policy in the form of a prohibition.

$$R_1(SSN, Diagnosis) :- S_1(SSN, Diagnosis), \$role \neq nurse. \quad (3.2)$$

The other sources allow accessing to their content without restrictions both for doctors and nurses, therefore there are no more authorization rules to specify.

At the Mediator, authorization rules are propagated by the sources aiming at preserving their policies. The propagation could lead to policy inconsistencies and conflicts [di Vimercati 1997]. These issues are out of the scope of this work. In our example there is only one rule defined by S_1 to be propagated at the mediator.

We then assume that in the mediator the following functional dependencies are identified, either manually during the schema definition or by analyzing the data with algorithms such as TANE [Huhtala 1999]:

$$(AdmissionT, Service) \rightarrow SSN \qquad (F_1)$$
$$(AdmissionT, Doctor) \rightarrow Diagnosis \qquad (F_2)$$

F_1 holds because in each service there is only one patient that is admitted at a given time $AdmissionT$. Note that $AdmissionT$ represents the admission timestamp including hours, minutes and seconds. F_2 holds because at a given timestamp, a doctor could perform only one diagnosis.

Now, we will show how functional dependencies, could be used by a malicious user to violate the rule (3.2). Let us assume the following queries are issued by a nurse: $Q_1(SSN, AdmissionT, Service)$ and then $Q_2(Diagnosis, AdmissionT, Service)$. Combining the results of the two queries and using the functional dependency F_1, the nurse can obtain SSN and $Diagnosis$ simultaneously, which leads to the violation of the authorization rule (3.2). To do so, the nurse could proceed as follows: (a) join the result of Q_1 with those of Q_2 on the attributes $AdmissionT$ and $Service$; (b) take advantage of F_1 to obtain the association between SSN and $Diagnosis$.

This simple example shows how inferences could occur at the mediator level. Also, it highlights how joins and functional dependencies could be used in order to carry out an inference. This helps us to characterize and to define the problem in the next sections.

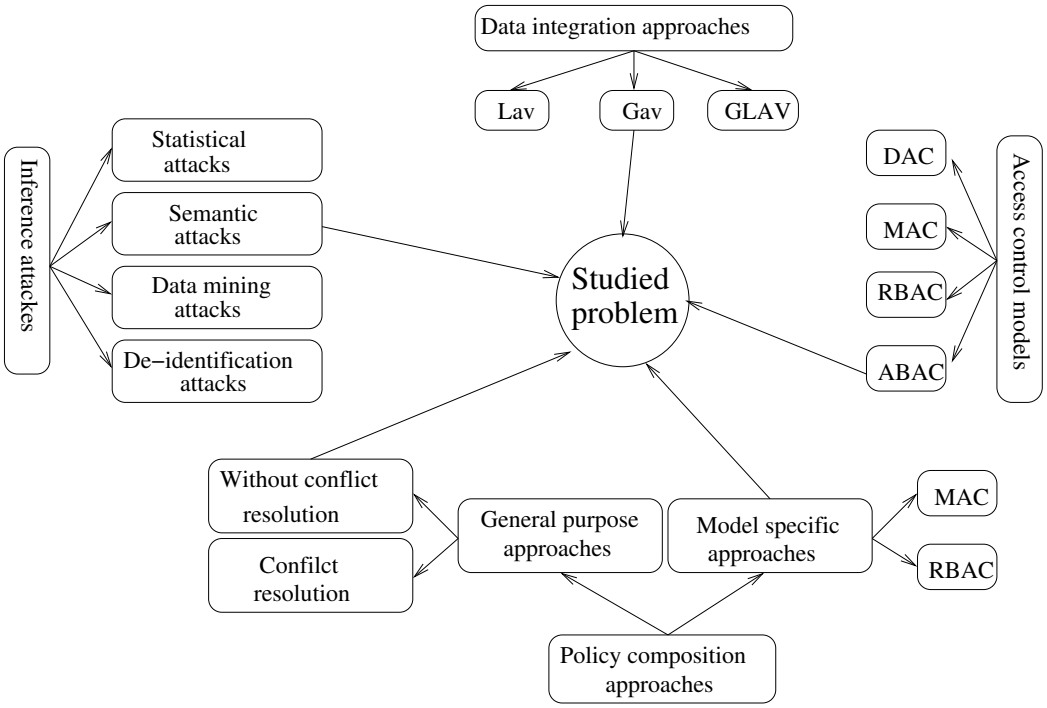

Figure 3.1: The studied problem with respect to related work

3.6 Problem discussion with respect to related work

In this section we discuss the different dimensions that we identified to be relevant to our problem. These dimensions include data integration, access control, policy combination and inference attacks. Figure 3.1 describes how our problem could fit the different dimensions mentioned in the related work chapter. Indeed, for each of the four dimensions we consider a particular instance that we believe is the most relevant. Next, we discuss the problem with respect to each dimension.

3.6.1 Data integration

We consider mediators constructed using GAV data integration approach. This means that each virtual relation of the mediator is defined as a conjunctive query over the local (source) relations. We also consider the case of conjunctive queries. We make this choice because conjunctive queries con-

stitute the basic building block for further extensions. Indeed, this class of queries represents a core class with respect to other classes of queries. We investigate this problem in the setting of conjunctive queries in order to design a first and extensible approach for securing data integration systems.

3.6.2 Access control model

We use Attribute-based access control which could simulate the main traditional access control models (as discussed in Chapter 2). In order to avoid terminological ambiguity we will refer to this model as Constraint-Based Access Control (CBAC). To enforce this model we resort to view-based access control. In particular, we consider the model of parameterized views. The idea is to illustrate how to define an authorization policy at the mediator level using the most flexible access control model. Indeed, other models could be simulated using the constraint-based access control model. So, one only needs to translate a policy expressed in a traditional model into CBAC to use our approach.

Another key component of access control is the one of controlling access to an association. Indeed, as we already discussed this issue in Chapter 2, information of an association of attributes could be more sensitive than each attribute considered separately. Since we consider a distributed system where sources are autonomous and combination of information is frequent, we believe that we need to pay particular attention to controlling access to associations of attributes.

3.6.3 Policy combination

Policy combination is used to generate a preliminary policy for the mediator. The idea is to preserve the local constraints in the mediator. To do so, we consider the conjunction of constraints protecting the same information. In other words, if the same information (*e.g.,* an attribute) is provided by different sources using different constraints to protect it, we consider the conjunction of such constraints. The idea is to preserve all source constraints and thus preserving the local (source) policies.

3.6.4 Inference based attacks

We consider semantic attacks which happen at the mediator level. In particular, we study the problem of a malicious user issuing a set of well chosen queries and taking advantage of functional dependencies to infer some prohibited information.

3.7 Problem definition

By considering the previous definitions and the different highlighted dimensions, we now can define our problem. To do so, we describe the different inputs that are required. Namely, the local (source) policies, the global (mediator) schema and the set of functional dependencies that hold at the mediator level. Then, we describe the expected output which is the mediator policy together with the properties that this policy should guarantee. We present both inputs and output according to the notation used in the preliminaries section.

To characterize inferences we introduce the notion of lossless join.

Definition 22 (Lossless join) *[Aho 1979]*

Let $\{R_1, R_2, ..., R_n\}$ *be a set of relation schemas and let* $R = \bigcup_{i=1}^{n} R_i$. *We say that the set* $\{R_1, R_2, ..., R_n\}$ *has a lossless join property if for any instance* I*:*

$\pi_R(I) = \bowtie_{i=1}^{n} \pi_{R_i}(I)$.

This definition describes how a set of queries could be joined to reconstruct another relation. When the reconstructed relation is forbidden, an inference could occur. Indeed, each query, taken separately is not sensitive. Nevertheless, this set of queries could lead to reconstruct a prohibited information, and hence violating the authorization policy.

- **Inputs**

  1. A set of sources $S_1, S_2, ..., S_n$, where each source S_i has a set of local authorization views.
  2. A set of virtual relations, each one built using the GAV approach (*i.e.,* by using a conjunctive query over the local relations).
  3. A set of functional dependencies that hold at the mediator level.

- **Output**

  A set of authorization views to be attached to the mediator, hence corresponding to its policy.

- **Properties**

  When a malicious user issues a batch of queries $Q_1, Q_2, ..., Q_m$ to the mediator, the following properties must hold:

  1. **Preserving sources' policies:** This property states that for each Q_i that is denied at source level, the authorization views of the mediator should ensure that Q_i is also denied at the mediator level.
  2. **Preventing from inference:** This property states that if the lossless join of $\{Q_1, Q_2, ..., Q_m\}$ is denied at a source level, it should also be denied at the mediator level.

3.8 Methodology

The proposed methodology, as shown in Figure 3.2, consists of a sequence of phases and steps involving appropriate algorithms. It takes as input a set of functional dependencies that hold in the mediator, the policies of the sources and the schema of the mediator. The methodology applies the following phases:

- **Propagation phase:** It aims at propagating the local (source) policies to the mediator. This is achieved by combining the local constraints of the mediator. This phase considers each virtual relation of the mediator separately and computes the authorization views of the virtual relation depending on how this relation has been constructed.

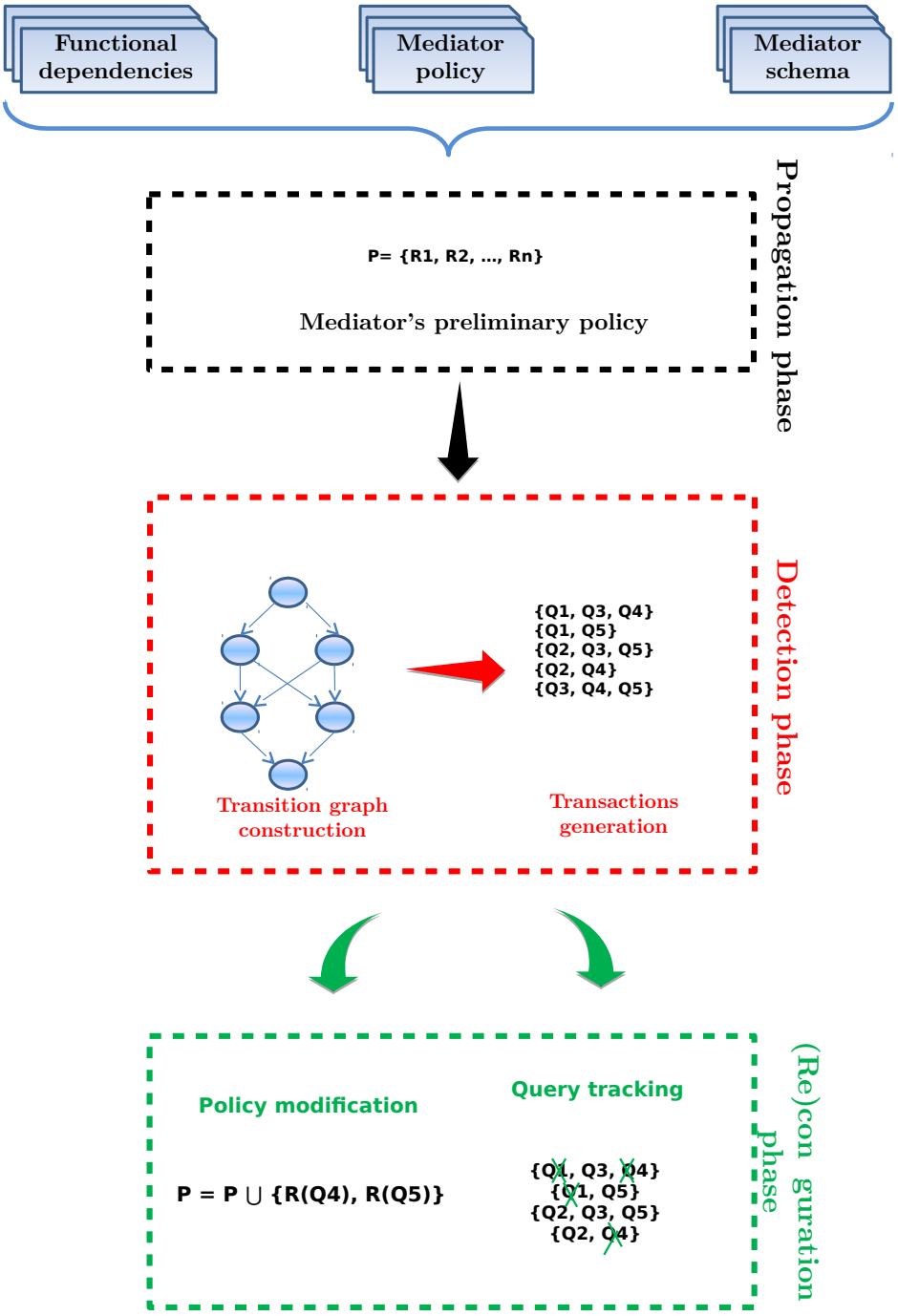

Figure 3.2: The proposed methodology to define an inference free policy to the mediator

- **Detection phase:** It aims at identifying all the violations that could occur if functional dependencies are used. Each of the resulting transactions represents a potential violation. Indeed, as shown previously, the combination of all the queries of a single transaction leads to a violation of an authorization. This phase includes the following steps:
  - *Construction of a transition graph (G):* This is done for each authorization rule by using the set of provided functional dependencies.
  - *Identification of the set Violating Transactions (VT):* It consists in identifying all the different paths between nodes in G to generate the set violating transactions.
- **Reconfiguration phase:** It proposes an approach to forbid the completion of each violating transaction. By completion of a transaction we mean issuing and evaluating all the queries of that transaction. A rule is violated only if the entire transaction is completed. This phase proposes two different solutions to achieve this goal.
  - *Design time solution:* It modifies/repairs the authorization policy in such a way that no violating transaction could be completed. This solution aims at generating a minimal set of authorization rules that forbid malicious users from accessing any violating transaction.
  - *Runtime solution:* It keeps track of the user queries and make sure that no violating transaction is completed. This solution is history-based and allows users accessing part of violating transaction. When a user query (combined with past queries) is about to provide a user access to a whole violating transaction, then this query is denied.

Next chapters elaborate on these phases. Chapter 4 discusses the propagation and the detection phase which are achieved at the design time. Chapter 5 discusses the proposed solutions: policy revision which is achieved at the design time and query tracking which is achieved at runtime.

Chapter 4

Detection Phase

Contents

4.1 Introduction

In this chapter we discuss how to detect inferences induced by semantic constraints in data integration systems. We first start by describing how policies should be propagated from sources to the mediator. In other words, how to

combine the policies of the sources in order to generate a mediator's policy. This phase is designed to ensure that constraints defined at source level will be preserved at the mediator level. We then discuss that such policy combination could fail to protect data against inferences. To deal with inferences, we shall first detect them. In this chapter, we propose an intuitive approach that aims at detecting all inferences that could occur at the mediator level by using functional dependencies.

We also introduce in this chapter a comprehensive scenario to illustrate both of these phases. We finally describe some experiments that we have conducted.

4.2 Propagating and combining the policies of the sources

In this section we discuss how to combine the policies of the sources in order to generate a representative policy at the mediator level. Please recall that we consider sources enforcing constraint-based access control. As discussed in Chapter 2, this model could simulate most classical access control models [Wang 2004, Bonatti 2000]. We also consider view-based access control to enforce constraint-based access control.

In order to generate a mediator's policy from the sources' policies, we can identify two cases:

- *Propagation:* We refer to propagation in the case where one source provides a particular information to the mediator. In other words, a virtual relation of the mediator is defined using only one relation from one source. This case is pretty simple and one needs only to propagate the source's constraints to the mediator with no change. The mediator enforces the same constraints as the source that provides the information and thus preserves the source policy.

- *Combination:* We refer to combination when the same information is provided by different sources. In this case, each source enforces its own constraints. The mediator needs to comply with all sources' constraints.

A mean to achieve this objective is to consider the conjunction of the source polices. Indeed, attaching the conjunction of the constraints to the mediator ensures that the mediator's policy preserves each constraint of the sources. The conjunction of constraints is always stronger than each constraint taken separately. This means that if a user's profile complies with the conjunction of the constraints, it also complies with each constraint taken separately.

We could observe that propagation is a particular case of combination when there is only one source. So, we describe how combination of policies should be performed in a data integration context.

In GAV data integration context, each virtual relation of the mediator is defined using a conjunctive query over the sources' relations. In order to generate the mediator's policy, we need to generate a set of appropriate authorization rules for each virtual relation.

To achieve this goal, this phase considers each virtual relation of the mediator separately and computes the authorization views of the virtual relation depending on how this relation has been constructed. Now, we discuss how to build the set of authorization views of the mediator (algorithm 1). For a given virtual relation it considers all the local relations used to build the virtual relation. Every local relation could have a set of authorization views. For each local authorization view, if all of its attributes appear in a virtual relation, then a global authorization view with the same schema as the local view is created. This allows the association between attributes that were forbidden at the source level to remain forbidden at the mediator level. It also allows to take into consideration only the local authorizations that are relevant to the mediator. Now, we describe how the constraints to be attached to the global authorization view are computed:

- If the schema of the local authorization view is unique with respect to other local authorization views, then the global authorization view will have the same constraint as the local authorization view.
- If there are at least two local authorization views having the same schema, then the authorization view constraint will be defined as the conjunction of the local constraints.

This way of computing the constraints of global views ensures that each local authorization view is preserved at the mediator level. Indeed, each constraint forbidding an access at source level also applies at mediator level.

Algorithm 1: Authorization views generation

input : R: a virtual relation of the mediator.
$V = \{V_1, V_2, ..., V_k\}$: the set of local authorization views of the local relations used to construct R.

output: A set of authorization views for R.

```
1 foreach V_l ∈ V do
2     if ∀Attibute ∈ V_l : Attribute ∈ R then
3         Create a global authorization view for R with the same schema
          as V_l
4         Compute the constraint of the global authorization view
```

4.3 Scenario

To illustrate how source policies need to be propagated to the mediator, we refer to our health care scenario where four sources are used to create a mediator.

- Source 1: $S_1(SSN, Diagnosis, Doctor)$. This source provides the social security number (SSN) of a patient, the diagnosis that has been made and the doctor who made it.
- Source 2: $S_2(SSN, AdmissionTime)$. This source provides SSN and the admission time which is a timestamp. This timestamp corresponds to the date of the admission including hours, minutes and seconds.
- Source 3: $S_3(SSN, Service)$. This source provides SSN together with the service (department) where the patient had been admitted.
- Source 4: $S_4(Diagnosis, Doctor, AdmissionTime, Service)$. This source provides SSN, the Doctor in charge of the patient, AdmissionTime which is a timestamp and the Service where the patient has been admitted.

Now, we consider a data integration system built upon these sources. This system is designed by defining a mediator over the sources. In this scenario, we consider a mediator that includes three virtual relations: $M_1(SSN, Diagnosis, Doctor)$, $M_2(SSN, AdmissionTime, Service)$ and $M_3(SSN, AdmissionTime, Service)$. M_1 is defined by importing all the information from S_1. M_2 is defined by joining the information of S_2 and S_3. M_3 is defined by taking all the information of S_4.

The mapping between the mediator's virtual relations and the sources is defined as follows:

- $M_1(SSN, Diagnosis, Doctor) : -S_1(SSN, Diagnosis, Doctor).$
- $M_2(SSN, AdmissionTime, Service) : -S_2(SSN, AmissionTime),$
  $S_3(SSN, Service).$
- $M_3(Diagnosis, Doctor, AdmissionTime, Service) : -$
  $S_4(Diagnosis, Doctor, AdmissionTime, Service).$

We further assume that the following functional dependencies hold at the mediator level:
$F_1 : (AdmissionT, Service) \rightarrow SSN$
$F_2 : (AdmissionT, Doctor) \rightarrow Diagnosis$

F_1 holds because at each service there is only one patient that is admitted at a given time $AdmissionT$. Note that $AdmissionT$ represents the admission timestamp including hours, minutes and seconds. F_2 holds because at a given timestamp, a doctor could make only one diagnosis.

We have described how the mediator is defined at data level. Next, we discuss how to generate its authorization policy. But first we need to introduce the local (source) policies.

4.3.1 Local (source) policies

Each source defines its own policy independently from the other sources. In this section, we describe how each source protects its data. We recall that we consider authorization views expressing prohibitions. The idea is thus to

enumerate for each source a set of authorization rules expressing its policy. In order to simplify such a scenario we assume there are only four roles considered at each source, namely, doctors, nurses, administratives and receptionists. We also assume that access control constraints used in the authorization views are related to roles. Our reasoning applies also if there are more roles or when constraints are related to other entities than roles. Such assumptions are made to simplify the presentation.

4.3.1.1 Policy of source 1

The first authorization rule of source 1 states that the association between SSN and $Diagnosis$ is allowed to users having a role different from administrative, receptionist and nurse. In other words, this authorization rule allows only doctors to access the association of SSN and $Diagnosis$.

$$V_{1S1}(SSN, Diagnosis) : -S_1(SSN, Diagnosis, Doctor),$$
$$\$role \neq Administrative \wedge \$role \neq receptionist \wedge \$role \neq nurse.$$

The second authorization rule states that SSN and $Doctor$ should not be disclosed to administratives and receptionists. This rule allows only doctors and nurses to have access to such an association.

$$V_{2S1}(SSN, Doctor) : -S_1(SSN, Diagnosis, Doctor),$$
$$\$role \neq Administrative \wedge \$role \neq receptionist.$$

4.3.1.2 Policy of source 2

Source 2 has two authorization rules. The first one forbids administratives and receptionists from accessing SSN and $AdmissionTime$. The second one prohibits receptionists from accessing SSN.

$$V_{1S2}(SSN, AmissionTime) : -S_2(SSN, AmissionTime),$$
$$\$role \neq Administrative \wedge \$role \neq receptionist.$$

$$V_{2S2}(SSN) : -S_2(SSN, AmissionTime),$$
$$\$role \neq receptionist.$$

4.3.1.3 Policy of source 3

The first authorization rule of source 3 forbids the access to SSN and $Service$ for administratives.

$V_{1S3}(SSN, Service) : -S_3(SSN, Service),$
$\$role \neq Administrative.$

The second authorization rule of source 3 forbids the access to SSN for administratives and receptionists.

$V_{2S3}(SSN) : -S_3(SSN, Service),$
$\$role \neq Administrative \wedge \$role \neq receptionist.$

4.3.1.4 Policy of source 4

The authorization rule of source 4 states that administratives are not allowed to access all attributes of source 4. This means that doctors and nurses are allowed to access all attributes of source 4.

$V_{1S4}(Diagnosis, Doctor, AdmissionTime, Service) : -$
$S_4(Diagnosis, Doctor, AdmissionTime, Service),$
$\$role \neq Administrative \wedge \$role \neq receptionist.$

4.3.2 Policy of the mediator

For each virtual relation of the mediator, we propagate the underlying authorizations of the sources.

$V_{1M1}(SSN, Diagnosis) : -M_1(SSN, Diagnosis, Doctor),$
$\$role \neq Administrative \wedge \$role \neq receptionist \wedge \$role \neq nurse.$

This authorization rule is propagated from source 1 (V_{1S1}).

$V_{2M1}(SSN, Doctor) : -M_1(SSN, Diagnosis, Doctor),$
$\$role \neq Administrative \wedge \$role \neq receptionist.$

This authorization rule results from propagating V_{2S1} to the mediator

$V_{1M2}(SSN, AmissionTime) : -M_2(SSN, AdmissionTime, Service),$
$\$role \neq Administrative \wedge \$role \neq receptionist.$

This authorization rule is propagated from source 2 (V_{1S2}).

$V_{2M2}(SSN, Service) : -M_2(SSN, AdmissionTime, Service),$
$\$role \neq Administrative \wedge \$role \neq receptionist.$

This authorization rule is propagated from source 3 (V_{1S3}).

$V_{3M2}(SSN) : -M_2(SSN, AdmissionTime, Service),$
$\$role \neq Administrative \wedge \$role \neq receptionist.$

This authorization rule results from the combination of V_{2S2} and V_{2S3}. Indeed both of these rules protect the same information but use different constraints. Source 2 denies the access to receptionists while source 3 denies it to both administratives and receptionists. At the mediator level, the conjunction of the two constraints is used to build V_{3M2}. This leads to deny the access to administratives and receptionists.

$V_{1M3}(Diagnosis, Doctor, AdmissionTime, Service) : -$
$M_3(Diagnosis, Doctor, AdmissionTime, Service),$
$\$role \neq Administrative \wedge \$role \neq receptionist.$

This authorization rule is propagated from source 4 (V_{1S4}).

4.3.3 Example of inference

Although propagating policies of the sources to the mediator is needed to preserve the sources' constraints, this propagation could suffer from some incompleteness. Indeed, each authorization rule generated for the mediator is relevant. This is due to the fact that each mediator's authorization rule translates a requirement from one or many sources.

The aim of the previous scenario was to illustrate how local (source) policies are propagated to the mediator. It also serves the purpose of illustrating inferences that could be made at the mediator level.

To highlight such inferences, we consider a nurse issuing a batch of queries that will be combined to obtain sensitive information. In this scenario, we consider a nurse issuing the following queries:

- $Q_1(SSN, AdmissionTime, Service) : -M_2(SSN, AdmissionTime, Service).$

- $Q_2(Diagnosis, AdmissionTime, Service) : -$
  $M_3(Diagnosis, Doctor, AdmissionTime, Service)$.

We observe that both queries are allowed for a nurse. Indeed, V_{1M2} and V_{2M2} allow rewriting Q_1 using authorization views. Thus, Q_1 is evaluated. On the other hand V_{1M3} allows Q_2. Thus, both queries are evaluated. We recall that access control is enforced as follows: if a query could be rewritten (*i.e.,* expressed) using authorization views then the query is evaluated. Otherwise, it is denied.

Now, let us show that the policy generated to the mediator (the set of authorization rules) in the propagation phase is not sufficient. Indeed, some inferences could be made by malicious users in order to obtain sensitive information.

The problem in such a scenario is that some functional dependencies could hold and be used to infer sensitive information. In this case, F_1 : $(AdmissionT, Service) \rightarrow SSN$ holds at the mediator level.

Combining the results of the two queries and using the functional dependency F_1, the nurse can obtain SSN and *Diagnosis* simultaneously, which leads to the violation of the authorization rule V_{1M1} of the mediator. To do so, the nurse could proceed as follows:

- Join the result of Q_1 with those of Q_2 on the attributes $AdmissionT$ and $Service$.

- Take advantage of F_1 to obtain the association between SSN and $Diagnosis$.

From now on, we refer to a query set like $\{Q_1, Q_2\}$ as a *violating transaction.* This scenario highlights the limitation of the naive propagation of the policies of the sources to the mediator. In the next sections, we discuss why such an inference could occur. We then propose an intuitive approach for solving this problem.

4.4 Why inference could occur?

In this section, we discuss why inference could occur in data integration context. Indeed, understanding what caused the inferences is the first step in order to deal with them. We have identified two main concepts that allow malicious users inferring sensitive information:

- **Issuing a batch of queries** that seem non sensitive when taken separately. Indeed, a malicious user could issue a set of queries. Each query taken separately does not violate the authorization policy. The malicious user stores each query results and combine them in order to obtain a prohibited information. We note that the malicious user has to carefully choose an appropriate set of queries that are not sensitive when considered individually and that could be combined afterwards. In our previous scenario, Q_1 and Q_2 represent such a batch of queries that we refer to as a violating transaction.
- **Taking advantage of functional dependencies** to combine the queries aiming at synthesizing sensitive information. In particular this becomes possible when new semantic constraints appear in the mediator. Indeed, combining attributes from different sources could lead to new functional dependencies that do not exist in any source. For instance, in the previous scenario, F_1 does not hold in any source. This is due to the fact that this functional dependency uses attributes provided by different sources. Thus, the semantic constraints expressed by such functional dependencies could not be considered by any source while defining its policy.

4.5 Detecting inferences

The first step towards detecting inferences is to be able to characterize such inferences. Here, we use the concept of lossless join to characterize how functional dependencies could lead to infer sensitive information. To this end, we start by considering an authorization rule $V(X, Y)$ prohibiting the access to the association of two attributes X and Y. The idea is to find, at design time, how one could simulate V without accessing X and Y in the same query.

In order to simulate V using a batch of queries, one could use functional dependencies. This simulation is achieved by constructing a lossless join of V. Next, we recall the definition of a lossless join that has been introduced in chapter 3.

Definition 23 *(Lossless join)[Aho 1979]*

Let $\{R_1, R_2, ..., R_n\}$ *be a set of relation schemas and let* $R = \bigcup_{i=1}^{n} R_i$. *We say that* $\{R_1, R_2, ..., R_n\}$ *has a lossless join property if for any instance* I*:*

$\pi_R(I) = \bowtie_{i=1}^{n} \pi_{R_i}(I)$.

This definition provides the intuition behind the lossless join property. When this property holds, a relation (R) could be reconstructed using a set of relations (here $\{R_1, R_2, ..., R_n\}$).

An interesting result on lossless join is the one of [Rissanen 1977].

Definition 24 *(Lossless join and functional dependencies) [Rissanen 1977]*

$R \bowtie S$ *is lossless join if and only if* $R \cap S \rightarrow^* R$ *or* $R \cap S \rightarrow^* S$.

This definition states that if $T = R \bowtie S$, one could reconstruct T using R and S only if:

$R \cap S \rightarrow^* R$ or $R \cap S \rightarrow^* S$

where $\rightarrow^*$ denotes the transitivity property of functional dependencies (see Chapter 3).

This result is interesting for our investigation. Indeed, we are looking for the different ways a malicious user could proceed to simulate (*i.e.,* reconstruct) an authorization view $V(X, Y)$s using a batch of queries. The idea is to enumerate all the sets of queries that represent a lossless join of V.

In order to construct a lossless join of V, we start by constructing two queries $Q_1(A, X)$ and $Q_2(A, Y)$. The intersection of Q_1 and Q_2 is the attribute A. So, Q_1 and Q_2 could simulate V only if $A \rightarrow X$ or $A \rightarrow Y$. We could look at this from another perspective. We look for functional dependencies that

have as right hand side either X or Y. For each such a functional dependency, we know that we could generate two queries to simulate V.

For example, if $(A \rightarrow X)$ holds then one could simulate $V(X, Y)$ using two queries : $Q_1(X, A)$ and $Q_2(A, Y)$. Note that neither Q_1 nor Q_2 access both attributes X and Y. So, both queries will be considered as legitimate queries. An important issue to note is that Q_1 itself could be simulated by a batch of queries, say Q_3 and Q_4. We note that this recursive process will simulate the transitive functional dependency in the definition of [Rissanen 1977].

In this example there are two different ways to simulate V:

- using Q_1 and Q_2; or
- using Q_3, Q_4 and Q_2.

To deal with this problem we introduce an intuitive process that allows to enumerate the alternatives that could lead to simulate all ways of simulating a particular authorization rule. This process is divided into two steps.

1. **Building the transition graph:** In this step, we build a graph departing from an authorization view. From this authorization we derive, by using functional dependencies, the queries that could be used to simulate the authorization view. Each derived query will generate a new node in the graph. The same process will be applied to this new node. Once again this recursive process aims at deriving all possible queries.
2. **Identifying violating transactions:** This step aims at enumerating all sets of queries that could simulate an authorization view. To this end, it uses the previous constructed graph and considers the different paths of this graph. This step shows that each path departing from the authorization view node represents a violating transaction.

Next, we elaborate on both of these steps.

4.5.1 Building the Transition Graph

The aim of the transition graph is to list all the queries that could be derived from an authorization rule by using functional dependencies. For each

authorization rule we use the set of functional dependencies ($\mathcal{FD}$) to derive a transition graph ($\mathcal{G}$). To build $\mathcal{G}$ we resort to Algorithm 2 as follows:

1. Consider the set of attributes of an authorization rule as the initial node.
2. For each FD in $\mathcal{FD}$ that has the RHS attribute inside the current node (starting from the root):
   (a) Create a new node by replacing the RHS attribute of the node by the set of attributes of the LHS of FD.
   (b) Create an edge between the initial node and the new node. Then, label this edge with the corresponding FD that has been used.
3. Apply the same process for the new node.

Algorithm 2: BuildTransitionGraph (BuildG)

input : r_i an authorization rule of the policy P.
$\mathcal{FD}$, the set of functional dependencies.
output: $\mathcal{G}(V, E)$, the transition graph.

```
1  V := {v(r_i)}; // create the root v with the attributes of r_i
2  W := {v(r_i)}; // add v to a set W of vertices to visit
3  foreach w ∈ W do
4      W := W − {w};
5      foreach FD(LHS → RHS) ∈ FD do
6          if RHS ∈ w then                    // RHS has one attribute
7              x := w − {RHS} + LHS; // create a new vertex
8              if x ∉ V then
9                  V := V + {x};
10                 W := W + {x};
11             e := (w, x, LHS + {RHS}); // e is a new edge from w
                   to x //with as transition the attributes
                   LHS + {RHS}
12             if e ∉ E then              // if not already in E add it
13                 E := E + {e};
14 return G(V, E) ;
```

4.5.1.1 Bounding of the Order[1] of the Graph

Here we give an upper bound of the graph order. Consider a privacy rule having n attributes $A_1, \ldots, A_n$. Let $\mathcal{FD}$ be the set of functional dependencies that hold. We denote by $\mathcal{FD}_t$ the set of all functional dependencies that could be derived from $\mathcal{FD}$ by using the pseudo-transitivity rule (see Definition 9). Let m be the number of functional dependencies in $\mathcal{FD}_t$ that have in their RHS an attribute of the privacy rule. The number of nodes V of the constructed graph is at most $(\frac{m}{n}+1)^n$.

Let m_i be the number of functional dependencies having A_i as the RHS. We have $\Sigma m_i = m$. For each functional dependency, we create a new node by replacing the RHS by the LHS. Then, for each A_i we have m_i possible replacements. Thus, the number of nodes obtained by replacing one or more attributes is $\prod_{i=1}^{n}(m_i+1)$ which is bounded by $(\frac{m}{n}+1)^n$.

4.5.2 Transition graph of the scenario

Now, we describe how this process could be applied to our scenario. We focus on a single authorization view, namely, the authorization view V_{1M1}:

$$V_{1M1}(SSN, Diagnosis) :- M_1(SSN, Diagnosis, Doctor),$$
$$\$role \neq Administrative \wedge \$role \neq nurse \wedge \$role \neq receptionist.$$

In particular we will focus on the nurse role. As we have discussed in section 4.3.3 nurses could perform some inference that lead to disclose sensitive information. Indeed, we showed that nurses could access the association of SSN and $Diagnosis$. This access violates the authorization view V_{1M1}.

Figure 4.1 illustrates the generation of the transition graph. The process of constructing this graph is the following:

- Construct the initial node $(SSN, Diagnosis)$ that corresponds to the schema of V_{1M1}. This schema represents the forbidden association.
- Select a functional dependency that has as right hand side either SSN or $Diagnosis$. These two attributes are those of V_{1M1}.

[1] The number of nodes that could be generated.

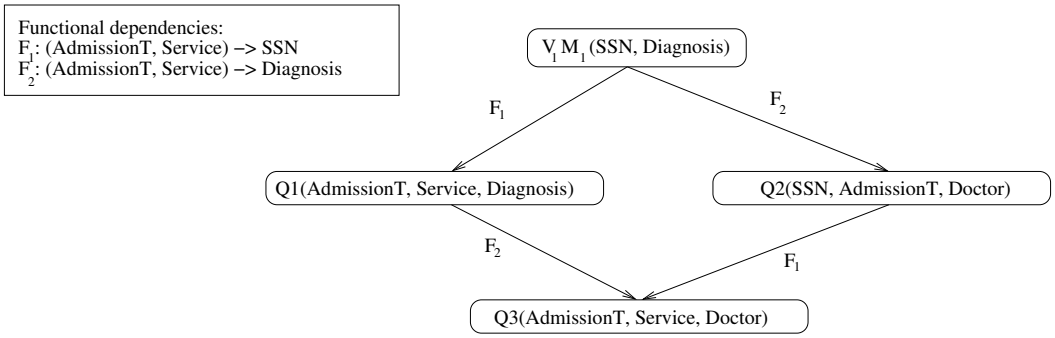

Figure 4.1: Transition graph construction

- First case: F_1 is selected.
  - Create a new node by replacing the right hand side of F_1 by its left hand side. Replace SSN by $AdmissionTime$ and $Service$. This leads to the creation of the node $Q1(AdmissionTime, Service, Diagnosis)$.
  - Create a link between the initial node $(SSN, Diagnosis)$ and the new node $Q1(AdmissionTime, Service, Diagnosis)$.
  - Label this link with the functional dependency that has been used. In this case, it is F_1.
  - Apply the same process to the node $Q_1(AdmissionTime, Service, Diagnosis)$ using F_2 which will lead to the creation of the node $Q_3(AdmissionTime, Service, Doctor)$. For the node $Q_3(AdmissionTime, Service, Doctor)$ there are no more functional dependencies that could be applied (*i.e.,* a functional dependency having as right hand side either $AdmissionTime$, $Servive$ or $Doctor$).
- Second case: F_2 is selected.
  - Create a new node by replacing the right hand side of F_2 by its left hand side. Replace $Diagnosis$ by $AdmissionTime$ and $Doctor$. This leads to the creation of the node $Q_2(SSN, AdmissionTime, Doctor)$.
  - Create a link between the initial node $(SSN, Diagnosis)$ and the new node $Q_2(SSN, AdmissionTime, Doctor)$.

  - Label this link with the functional dependency that has been used. In this case, it is F_2.
  - Apply the same process to the node $Q_2(SSN, AdmissionTime, Doctor)$ using F_1 which leads to the creation of the node $Q_3(AdmissionTime, Service, Doctor)$. For the node $Q_3(AdmissionTime, Service, Doctor)$ there are no more functional dependencies that could be applied (*i.e.,* a functional dependency having as right hand side either $AdmissionTime$, $Servive$ or $Doctor$).

- There is no more node that could be created in either branch. Thus, the process terminates and the graph is fully constructed.

4.5.3 Identifying Violating Transactions

In the previous step, we have generated a graph that stores all the derivations, from an authorization view, that could be made using functional dependencies.

As introduced in definition 23 with the lossless join property, if $(A \rightarrow X)$ then $Q_1(A, X)$ and $Q_2(A, Y)$ could simulate $V(X, Y)$. This means that $\{Q_1, Q_2\}$ is a violating transaction (*i.e.,* a set of queries that could lead to infer a sensitive information).

The constructed transition graph has been designed to capture the notion of violating transaction. Indeed, each violating transaction is represented by a path of the graph. So, the violating transaction is a set of queries constructed as follows:

- Each functional dependency on the path is translated into a query. This query asks for all attributes appearing in the functional dependency either in right or left hand side. In this case, $(A \rightarrow X)$ is translated into $Q_1(A, X)$.
- The final node belongs to the violating transaction. In this case, the final node is $Q_2(A, Y)$.

The set of minimal violating transactions ($\mathcal{VT}$) is constructed as follows. First, a path between the initial node (the node representing the authorization

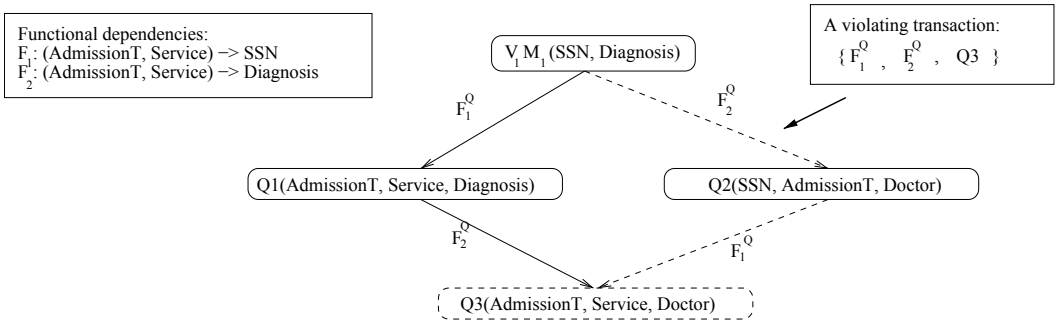

Figure 4.2: Violating transactions identification

rule) and every other node is considered. As shown in Figure 4.2, from this path a transaction (*i.e.,* a set of queries) is constructed. Each query that is used as a label on this path is added to the transaction. Finally, the query of the final node of the path is also added to the transaction. This is done for all nodes and paths in $\mathcal{G}$. Before introducing the algorithm that generates the set of violating transactions, let us introduce the following definitions.

Definition 25 (Building a query from a functional dependency) *Let F be a functional dependency. We define F^Q as the query that projects on all the attributes that appear in F, either in the RHS or in the LHS. For example, let $R(A_1, A_2, A_3, A_4)$ be a relation and let F be the functional dependency $A_1, A_2 \rightarrow A_3$ that holds on R. In this case F^Q is the query that projects on all the attributes that appear in F. F^Q is the query $F^Q(A_1, A_2, A_3) {:-} R(A_1, A_2, A_3, A_4)$.*

Definition 26 (Minimal Query) *A query Q is minimal if all its attributes are relevant, that is $\forall Q' \subset Q : Q'$ cannot be used instead of Q in a violating transaction.*

Definition 27 (Minimal Violating Transaction) *A violating transaction T (see Chapter 3) is minimal if: (a) all its queries are minimal, and (b) all its queries are relevant i.e. $\forall Q \in T : T \setminus \{Q\}$ is not a violating transaction.*

To generate the minimal set of transactions ($\mathcal{VT}$) that is compliant with the definition 27, we use the recursive Algorithm 3. The initial call to the algorithm is:

$$\mathcal{VT} := FindVT(\mathcal{G}, root, \emptyset, \emptyset)$$

Algorithm 3: FindViolatingTransactions (FindVT)

input : $\mathcal{G}(V, E)$, the transition graph, v current vertex, c_t current path, $\mathcal{VT}$ current set of transactions.
output: $\mathcal{VT}$, the set of minimal violating transactions.

```
1 foreach e ∈ outgoing edges of v do
2     t := c_t + e.transition + e.to ;
      // e.transition is the set of attributes of the transition
         while e.to is the destination node
3     if ∄k ∈ VT | k ⊆ t then
4         VT := VT + {t} ;
5         foreach k ∈ VT do
6             if t ⊆ k then
7                 VT := VT − {k} ;
                  // reducing further VT
8     return FindVT(G, e.to, c_t + e.transition, VT);
      // recursive call with the v reached by e (e.to) by
         adding the e.transition to the current VT
```

The example in Figure 4.2 contains three nodes Q_1, Q_2 and Q_3 in addition to the initial node V_{1M1}. If we apply Algorithm 3, it will generate, for each node Q_i, a transaction containing each F^Q on the path between V_{1M1} and Q_i, and Q_i itself. For example, to generate T_3 that represents the path between V_{1M1} and Q_3, we start by adding each F_i on the path from V_{1M1} to Q_3. Here, F_1 and F_2 are translated into F_1^Q and F_2^Q respectively. Finally, we add Q_3. Thus, we obtain $T_3 = \{F_1^Q, F_2^Q, Q_3\}$. In the example the returned $\mathcal{VT}$ is: $\mathcal{VT} = \{T_1 = \{Q_1, F_1^Q\}, T_2 = \{Q_2, F_2^Q\}, T_3 = \{Q_3, F_1^Q, F_2^Q\}\}$.

At this stage we emphasized the fact that functional dependencies could be combined with authorized queries to obtain sensitive information. In our example, this issue is illustrated by the fact that if all the queries of any transaction T_i are issued and evaluated then the authorization rule $V_{1M1}(SSN, Diagnosis)$ is violated.

4.5.4 Summary on the detection phase

In this phase we have shown how to enumerate sets of queries, called violating transactions. When queries of a violating transaction are combined, they lead to the violation of an authorization view. This phase aims at detecting the flaws that could occur after the propagation phase is achieved. We have shown that even for a fairly small scenario three different violating transactions could occur.

The next section describes a set of experiments that we have conducted to evaluate some parameters from a practical point of view.

4.6 Experiments

We have conducted a number of experiments on real and synthetic datasets to validate the detection phase. With synthetic datasets we generated particular configurations (*e.g.,* worst-case scenarios) while with the real datasets (downloaded from the UCI ML Repository [Bache 2013]) we first extracted $\mathcal{FD}$ by using a well-known algorithm called TANE [Huhtala 1999] and then we run our algorithms with sets of rules having different numbers of attributes (from 2 to 10). We also tested the algorithms on specific subsets of $\mathcal{FD}$ (*e.g.,* 100 and 200 extracted from the Bank dataset) that were not present in real datasets (Sub 1 and 2 in Table 4.1). The source code of the algorithms is released under GPL v3 free software license and is available at the following address [Haddad 2013b].

4.6.1 The testing procedure

The characteristics of the real and synthetic datasets are shown in Table 4.1. The whole sequence of steps needed to set up the environment and execute the algorithms is the following:

1. Selection of datasets with different numbers of attributes. We assume a single relation at the mediator level and we define S as the union of attributes.

2. Extraction of the functional dependencies from real datasets by using the TANE algorithm. In case of synthetic datasets we generated FDs by considering a worst-case approach. We generated FDs by varying the number of attributes of the LHS. Each FD in $\mathcal{FD}$ is generated in such a way that the whole set $\mathcal{FD}$ will be considered during the graph construction. To do so, each RHS of a new FD needs to be contained in the policy or one of the LHS of the previously generated FDs. For example, if the policy is $(A_1,\ A_2)$ and if the number of attributes in LHS is equal to 2, the sequence of generated FDs will be $(A_3, A_4) \rightarrow A_1$, $(A_5, A_6) \rightarrow A_2$, $(A_7, A_8) \rightarrow A_3$ and so on.

3. Generation of policy P while varying the number of attributes inside rules. As we will show in the next section the number of attributes of a rule affects the graph size and consequently the time to build $\mathcal{G}$ and identify $\mathcal{VT}$.

4.6.2 Results

Table 4.1 shows the results obtained by running the algorithm for each dataset. The different parameters are:

- FD_l is the average number of attributes that appear in $\mathcal{FD}$.
- $|\mathcal{G}(V)|$ is the number of nodes.
- $|\mathcal{G}(E)|$ is the number of edges of the generated graph.
- $BuildG$ is the time, in ms, to build $\mathcal{G}$.
- $|\mathcal{VT}|$ is the number of generated violating transactions.
- $FindVT$ is the time, in ms, to construct the set of violating transactions.

For each of the tests reported in Table 4.1 we calculated the mean value for 100 different executions.

While Table 4.1 reports on the approach practicability on real datasets, the graphs in Figures 4.3, 4.4 and 4.5 show tests performed on synthetic datasets.

Dataset description		Identified $\mathcal{FD}$		Performed experiments and results				
Name	$\|S\|$	$\|\mathcal{FD}\|$	$\mathcal{FD}_l$	$\|\mathcal{G}(V)\|$	$\|\mathcal{G}(E)\|$	$BuildG$	$\|\mathcal{VT}\|$	$FindVT$
Yeast	8	10	3.88	6	10	5	5	4
Chess	20	22	9.14	21	20	3	20	14
Breast W.	11	37	4.13	41	165	26	37	65
Abalone	8	44	3.79	87	835	60	17	42
Sub 1	17	100	4.41	217	1312	193	130	197
Sub 2	17	200	4.92	453	8152	1502	1737	16596
Bank	17	433	6.47	14788	879241	3826	9137	335607

Table 4.1: The data sets together with results of the experiments

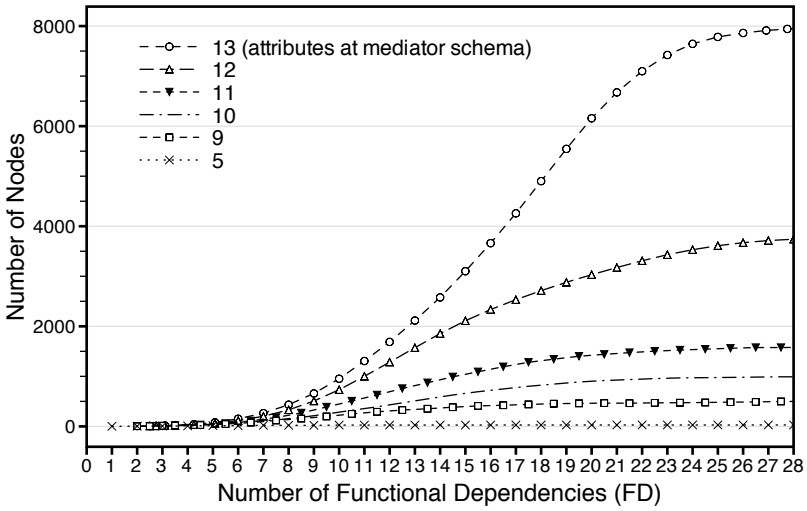

Figure 4.3: Graph construction and violating transactions identification

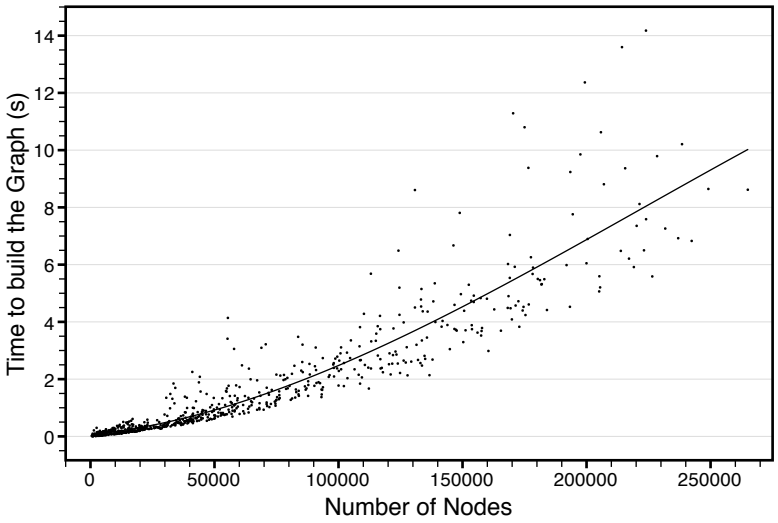

Figure 4.4: Graph construction and violating transactions identification

Also in this case we run multiple tests while varying parameters that are not subject to the evaluation.

In particular, Figure 4.3 shows the relation between the number of nodes and the cardinality of randomly generated $\mathcal{FD}$. We reported different tests while varying the number of attributes of the global (mediator) schema. The tests show that by increasing the cardinality of $\mathcal{FD}$, the number of nodes increases very fast until, at a certain point, it starts slowing and approaching

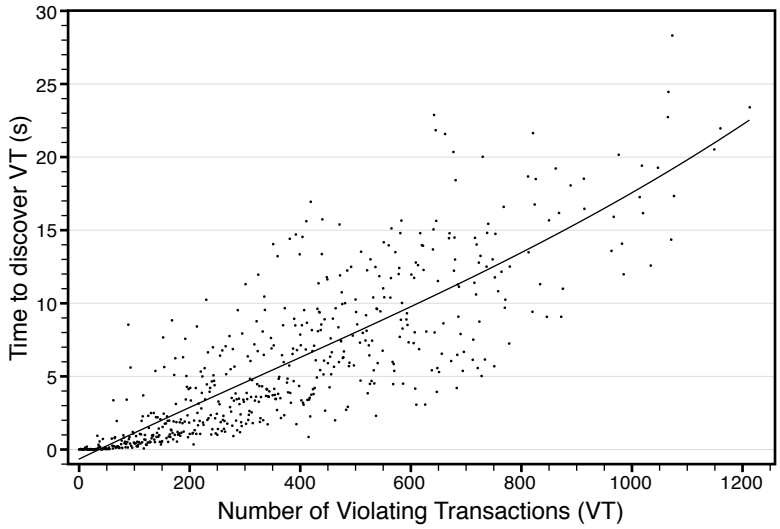

Figure 4.5: Graph construction and violating transactions identification

its upper bound as expected theoretically.

Figure 4.4 shows the relation between the number of nodes and the time needed to build $\mathcal{G}$ with fixed attributes in the global (mediator) schema. As we can see, the time to build $\mathcal{G}$ increases proportionally with respect to the number of nodes. This is mainly because we use binary trees to manage the nodes. The dots in figures represent single executions while the line has been generated using the Spline algorithm [Hastie 1990].

Figure 4.5 reports the performances for identifying $\mathcal{VT}$ from previously built graphs. The time increases proportionally with respect to the number of transactions.

The experiments show the practicability of our approach on different datasets with different characteristics. The approach showed some limitation only when the cardinality of $\mathcal{FD}$ becomes very large (*e.g.*, greater than 1500 for a single relation) and hence not able to discover transactions in an acceptable period of time. We believe that this amount of FDs does not represent a typical scenario. Nevertheless, this issue remains relevant and important to be investigated.

4.7 Discussion

In this chapter we have presented the first two phases of our methodology that aims at defining a global policy of the mediator. This policy should both preserve the sources' policy and prevent from inferences at global level.

The propagation phase ensures that the sources' constraints are preserved at the mediator level. To achieve this goal, this phase considers conjunctions of local constraints at the mediator level.

The detection phase has been designed to enumerate all violating transactions. To this end a transition graph is constructed, then the paths on this graph are used to construct the violating transactions. We have described a set of experiments that we have performed to evaluate this phase. The results of these experiments show the practicability of the proposed approach.

These two phases are mandatory in order to prevent against inferences. How to deal with such inferences and prevent malicious from synthesizing a prohibited information is the topic of the next chapter.

Chapter 5

Reconfiguration phase

Contents

5.1 Introduction

In this chapter, we discuss how to deal with violating transactions. We present the reconfiguration phase of our methodology. This phase aims at preventing

a user from issuing all the queries of a violating transaction. Please remember that queries of a violating transaction can be issued and evaluated as long as they do not lead to a violation of the transaction. If a user could not complete the execution of all the queries of any violating transaction then no violation could occur.

This chapter is organized as follows: we start by stating the problem of the reconfiguration phase, we then describe both the design time and the runtime approaches. Finally, we compare these two approaches and describe some performed experiments over this phase.

5.2 Problem statement

In the reconfiguration phase we tackle the following problem: *"Given a set of violating transactions, each violating transaction being defined by a set of queries, prevent a user from getting answers to all the queries of any single transaction".*

5.3 Approaches

To tackle this problem, there are two different visions. These visions differ on which time to tackle this problem. Indeed, this problem could be resolved either at design time or at runtime. Nevertheless, these visions agree on the need of preventing malicious users from inferring sensitive information. Thus, preventing users from completing any violating transaction (*i.e.,* accessing all queries of a single transaction). We propose two solutions[1] to deal with the problem of the reconfiguration phase. We then discuss and compare both solutions. The two proposed solutions are summarized here:

- **Policy revision:** This solution is performed at design time and aims at repairing the mediator policy. This is achieved by adding an appropriate set of authorization rules to the mediator in such a way that no violating transaction could be completed.

[1] One solution for each vision.

- **Query tracking:** This solution is performed at runtime. It is history-based. This means that each user query that is evaluated is stored into a log file. Then, the combination of past queries and the current query is performed each time a new query is issued. If this combination leads to the access to all queries of a given single transaction then the current query is denied. Otherwise, the current query is evaluated and added to the log file.

5.4 Policy revision

This solution revises the policy by adding new rules such that no violating transaction could be completed. A naive approach could be to deny one query for each transaction. This ensures that the malicious user could not access all queries of any single transaction. To illustrate this principle, we consider the following arbitrary set of violating transactions:

$T_1 = \{Q_1^1, Q_2^1, ..., Q_{m1}^1\}$
$T_2 = \{Q_1^2, Q_2^2, ..., Q_{m2}^2\}$
$\vdots$
$T_n = \{Q_1^n, Q_2^n, ..., Q_{mn}^n\}$

Indeed, if $\forall T_i, \exists Q_j^i$ such that Q_j^i is denied then no transaction could be completed.

Although this naive solution is safe from an access control point of view, it is not desired from an availability point of view. Indeed, taking randomly one query from each transaction could induce denying n different queries to the user. This could be costly from an availability point of view. In particular when a smaller set of m queries ($m < n$) shares at least one query with every transaction.

To achieve a trade off between authorization enforcement and availability, we investigate the problem of finding the minimal set of queries that denies at least one query for each violating transaction. We refer to this problem as *query cancellation problem*. We first recall some definitions related to complexity theory. We then *formalize* and characterize the *complexity* of the query cancellation problem for one rule. Finally, we discuss the case of a policy (*i.e.,* a set of rules).

5.4.1 Complexity: some relevant issues

Complexity theory aims at analyzing the difficulty of solving problems. This difficulty could be measured by quantifying the different resources that are needed to perform the required computations [Garey 2002]. The resources that are usually of interest are the computation time and memory space needed to solve the problem.

An important part of complexity theory deals with decision problems. A decision problem is a problem which output is either "yes" or "no" for any input that is provided. A decision problem could be modeled by a question to which the answer is always either "yes" or "no".

A complexity class is a set of all problems that could be solved using the same amount of resources. A complexity class could either refer to a time complexity or a memory complexity. We focus on time complexity which refers to the number of elementary operations that should be performed by an algorithm in order to solve the problem. Two of the most studied complexity classes are P and NP.

The complexity class P is the set of all decision problems that could be solved in polynomial time by a deterministic machine. In this case, the problem has an efficient algorithmic solution.

The complexity class NP is the set of decision problems that could be solved in polynomial time using a non deterministic machine.

A problem A is reducible to a problem B if there exists an algorithm that translates each instance $a \in A$ into an instance $b \in B$ such that the answer to b is "yes" if and only if the answer to a is "yes" as well. A decision problem A is said to be NP-complete if:

- It belongs to NP; and
- Any NP problem could be reduced to A

When a problem only satisfies the second condition it is said to be NP-hard.

In order to proof that a problem A is NP-complete, we need to proof that a well known problem which was previously proven to be NP-complete could

be reduced to A. Historically, the first problem that have been proven to be NP-complete is the satisfiability problem SAT [Cook 1971]. Since then, many other problems have been proven to be NP-complete as well. Any of these problems could be used to prove that a new problem is also NP-complete.

The concepts introduced in this section will be used to formalize and study the query cancellation problem.

5.4.2 Problem formalization

Let $\mathcal{VT} = \{T_1, \ldots, T_n\}$ be a set of minimal violating transactions and let $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ be a set of queries such that $\forall i \in \{1, \ldots, n\} : T_i \in \mathcal{P}(\mathcal{Q}) \setminus \emptyset$. We define the following *Query Cancellation (QC)* recognition (decision) problem as follows:

- **Instance:** A set $\mathcal{VT}$, a set $\mathcal{Q}$ and a positive integer k.
- **Question:** Is there a subset $Q \subseteq \mathcal{Q}$ with $|Q| \leq k$ such that $\forall i \in \{1, \ldots, n\} : T_i \setminus Q \neq T_i$? Here, $|Q|$ denotes the cardinality of Q.

Algorithm 4: QueryCancellation

input : $\mathcal{VT}$ is the set of minimal violating transactions.
Q is the set of all the queries that appear in $\mathcal{VT}$
output: S is the set of all solutions

```
1 foreach q ∈ Q do
2     if ∀t ∈ VT, t ∩ q ≠ ∅ then
3         S := S ∪ q ;
4 return S ;
```

Thus, the optimization problem, which consists in finding the *minimum number of queries* to be canceled is called *Minimum Query Cancellation (MQC)*.

5.4.3 Problem complexity

In this section, we show the NP-completeness of QC. We propose a reduction from the domination problem in split graphs [Bertossi 1984]. In an undirected graph $\mathcal{G} = (V, E)$, where V is the node (vertex) set and E is the edge set, each node *dominates* all nodes joined to it by an edge (neighbors). Let $D \subseteq V$ be a subset of nodes. D is a dominating set of $\mathcal{G}$ if D dominates all nodes of $V \setminus D$. The usual *Dominating Set (DS)*[Bertossi 1984] decision problem is stated as follows:

- **Instance:** A graph G and a positive integer k.
- **Question:** Does G admit a dominating set of size at most k ?

This problem has been proven to be NP-complete even for split graphs [Bertossi 1984]. Recall that a split graph is a graph whose set of nodes is partitioned into a clique C and an independent set I. In other words, all nodes of C are joined by an edge and there is no edge between nodes of I. Edges between nodes of C and nodes of I could be arbitrary.

Theorem 1 *QC is NP-complete.*

Proof 1 *QC belongs to NP since checking if the deletion of a subset of queries affects all transactions could be performed in polynomial time. Let G be a split graph such that C is the set of nodes forming the clique and I is the set of nodes forming the independent set. We construct an instance QC of query cancellation problem from $\mathcal{G}$ as follows: $\mathcal{Q} = C$, $\mathcal{VT} = I$ and each transaction T_i is the set of queries that are joined to it by an edge in G. We then prove that G admits a dominating set of size at most k if and only if QC admits a subset $Q \subseteq \mathcal{Q}$ of size at most k such that $\forall i \in \{1, \ldots, n\} : T_i \setminus Q \neq T_i$.*

Assume QC admits a subset $Q \subseteq \mathcal{Q}$ of size at most k such that $\forall i \in \{1, \ldots, n\} : T_i \setminus Q \neq T_i$. Q is also a dominating set of $\mathcal{G}$. In fact, all nodes of I are dominated since all the transactions are affected by Q and all remaining nodes in the clique C are also dominated since they all are connected with nodes of Q.

Assume $\mathcal{G}$ admits a dominating set D of size at most k. Observe that D could be transformed into a dominating set D' of same size and having all its nodes in C. To ensure this transformation it is sufficient to replace all nodes of D that are in I by any of their neighbors in C. Note that the obtained set D' is also a dominating set of $\mathcal{G}$. The subset of queries to be canceled is then computed by setting Q to D'.

Corollary 1 *MQC is NP-hard.*

To generate the set of queries that need to be canceled we use Algorithm 4. It returns all the (candidate) sets of queries that have a non-empty intersection with each violating transaction. We can use different metrics to determine which set to choose. The first metric is the cardinality of the smallest set. Other metrics could be defined by the administrator. Indeed, some queries can be identified as more relevant to the application. In this case, the set of queries to be chosen could be the one that does not contain any relevant query. The minimal set of queries MQ is defined using one of the previous metrics. For each query Q in MQ a new authorization rule is added to prevent from the evaluation of Q.

5.4.4 Generalization for a policy

We have considered the problem of query cancellation for a single authorization rule. Now we summarize the process (Algorithm 5) for the whole policy. We denote by P the policy (*i.e.,* the set of rules). We denote by N_R the set of new rules that has been generated. The new policy set ($\mathcal{P}$) will be the union of $\mathcal{P}$ (given as input) and N_R.

- For each authorization rule in the policy P
  - Construct the graph.
  - Generate the violating transactions.
  - For each transaction find the minimum set of queries $MinQ$ using the query cancellation algorithm.
  - For each query in $MinQ$ add a new authorization rule into N_R.

Algorithm 5: GenerelizationForPolicy

input : $\mathcal{P}$ the set of authorization rules.
output: $\mathcal{P}$ augmented with new rules.

```
1  foreach r_i ∈ P do
2      G := BuildG(r_i);
3      VT := FindVT(G, root, ∅, ∅);
4      S := QueryCancellation(VT, Q); // Q is obtained listing
           VT
5      N_R := ∅; // N_R is the set of new rules
6      foreach q ∈ S do
7          N_R := N_R ∪ {r(q)}; // Generate a new authorization rule
               r from q
8      if N_R is not empty then
9          N_R := N_R ∪ GenerelizationForPolicy(N_R);
10 P := P ∪ N_R;
11 return P ;
```

- If N_R is not empty, apply the same process to N_R (instead of P) as input.
- Add the new generated authorization rules to the policy given as input.

5.4.5 Termination of the algorithm

Algorithm 5 deals with query cancellation for the whole policy. We denote by $\mathcal{P}$ the policy (*i.e.,* the set of rules). We denote by N_R the set of new rules that have been generated. The new policy set ($\mathcal{P}$) will be the union of $\mathcal{P}$ and N_R ($\mathcal{P} = \mathcal{P} \cup N_R$). In such settings, we need to make sure that the algorithm terminates.

A new rule could generate other new rules and so on until no rule is added. Let N_S be the set of attributes of the mediator schema. Since N_S is finite, the maximum number N_r of rules that could be defined is also finite. Let $N_{\mathcal{P}}$ be the number of rules in $\mathcal{P}$. Let n be the difference between N_r and $N_{\mathcal{P}}$. At each recursive call of the algorithm either no rule has been generated or n decreases since N_r increases. Thus, the algorithm terminates.

5.5 Query tracking

This solution is history-based, namely, it uses the past queries and the ongoing query to decide if the ongoing query can be evaluated. This solution takes as input the set of violating transactions that have been generated in the detection phase. The idea is to prevent a user from executing all the queries of any single transaction. This is achieved by labeling each query of any transaction that has been accessed by a user. Then, if the current query makes all the queries of any transaction labeled then the query is denied. Otherwise, the query is evaluated. This solution highlights the importance of the labeling process. Indeed, an incorrect labeling could lead to violations. The main issue is how to determine if a query of a violating transaction has been accessed by the combination of a batch of user queries.

We start by considering a simple example that shows the importance of the labeling method.

Let $T = \{Q_1, Q_2, Q_3\}$ be a violating transaction. If a user accesses all the queries of T then an authorization rule is violated. For each user query Q^u, we check if any Q_i is accessed. In other words, if combining some user queries leads to disclose information about the violating query Q_i. Let $Q^u = \{Q_1^u, Q_2^u, Q_3^u, Q_4^u\}$ be a sequence of user queries. To illustrate how the labeling process is achieved we make the following assumptions:

- Q_1^u discloses information about Q_1.
- Q_2^u together with Q_3^u disclose information about Q_2.
- Q_4^u discloses information about Q_3.

The different steps of the query tracking approach on this example are the following:

- When Q_1^u is received, Q_1 **is labeled and** Q_1^u **is evaluated.**
- When Q_2^u is received, Q_2^u does not disclose by its own information about any Q_i and thus **no labeling is performed and** Q_2^u **is evaluated.**
- When Q_3^u is received, Q_2 **is labeled and** Q_3^u **is evaluated.**

- When Q_4^u is received, Q_4^u discloses information about Q_3. Labeling Q_3 makes all the queries of T labeled. Thus, Q_4^u **is rejected.**

We have discussed how the labeling process should be carried out in order to perform the query tracking approach. This description omitted how to determine if a batch of queries induces disclosing information about a violating query.

Now, we describe how to detect such an information disclosure. We propose two approaches to achieve this goal:

- **Attribute-based approach:** This approach is achieved at the attribute level. In this case, a batch of queries disclose information about a violating query when the join of the the set of user queries contains all attributes of the violating query.
- **Audit-based approach:** This approach aims at refining the previous approach by considering inclusion at both attribute and tuple levels. This approach takes advantage of auditing approaches proposed in the literature.

Next, we elaborate on both approaches. We then compare them by discussing their advantages and their drawbacks.

5.5.1 Attribute-based query tracking

This approach considers joining the user queries in order to determine if this set of queries discloses information about the violating query. In this approach, if all attributes of a violating query are included in the attributes of the join (of the user queries), then the violating query is labeled.

We start by considering how user queries could be joined. Two queries could be joined if they share some common attributes. To know how joining a set of user queries could simulate a violating query, we introduce the $\overline{\wedge}$ operator defined as follows:

Definition 28 ($\overline{\wedge}$ operator) *The $\overline{\wedge}$ operator takes two queries as input and returns the join of the two queries if they share common attributes, otherwise it returns a set containing the two queries.*

$$Q_1 \overline{\wedge} Q_2 = \begin{cases} \{Q_1 \bowtie Q_2\} & \text{if } Q_1 \text{ and } Q_2 \text{ share attributes} \\ \{Q_1, Q_2\} & \text{otherwise} \end{cases}$$

The $\overline{\wedge}$ operator could be generalized to a set of queries and a single violating query as follows:

$$\{Q_1, Q_2, ..., Q_n\} \overline{\wedge} Q = (\{Q_1, Q_2, ...Q_i\} \cup \{Q_{i+1}, ...Q_n\}) \overline{\wedge} Q$$

We assume, without loss of generality, that every query $Q_j \in \{Q_1, Q_2, ...Q_i\}$ shares at least one common attribute with Q. We also assume that no $Q_j \in \{Q_{i+1}, ...Q_n\}$ shares any attribute with Q. We consider the following cases:

- $\{Q_1, Q_2, ...Q_i\}$ *is empty:*

$$\{Q_1, Q_2, ..., Q_n\} \overline{\wedge} Q = \{Q, Q_1, Q_2, ..., Q_n\}$$

- $\{Q_1, Q_2, ...Q_i\}$ *is not empty:*

$$\{Q_1, Q_2, ..., Q_n\} \overline{\wedge} Q = \{Q_1 \bowtie Q_2 \bowtie ... \bowtie Q_i \bowtie Q, Q_{i+1}, ...Q_n\}$$

The idea of this labeling method is that a set of user queries could simulate a single query of some transaction. Thus, when a query is submitted we should consider all the possible combinations with past queries. The $\overline{\wedge}$ operator organizes how past queries should be stored. Indeed, if two queries could be combined (joined) then we just need to keep (*i.e.,* we store) the joined query. If two queries could not be joined then we keep both queries. This reasoning is generalized to a set of queries. Then, we check if the new set of queries induces the labeling of all the queries of a single transaction.

Algorithm 6 describes attribute-based query tracking approach. It takes as input a set of violating transactions and a sequence of user queries. For each user query it applies the labeling method described above (*i.e.*, considering combination of the current query with past queries).

Algorithm 6: Attribute-Based Query Tracking

input : $\mathcal{VT} = \{T_1, T_2..., T_n\}$, the set of violating transactions. $PastQueries = \{Q_1^u, Q_2^u..., Q_m^u\}$, the set of already executed queries. Q^u the ongoing query.

output: Decision to accept or reject each Q

```
1  UserQueries := PastQueries ⊼ Q ;
2  foreach T_i ∈ VT do
3      foreach Q_j ∈ T_i do
4          if All attribute of Q_j are included in any query of PastQuries
           then
5              label Q_j ;
6              if all queries of T_i are labeled then
7                  cancel the labeling of Q_j ;
8                  reject Q^u ;
9              else
10                 PastQueries := UserQueries ;
11                 evaluate Q^u ;
12         else
13             PastQueries := UserQueries ;
14             evaluate Q^u ;
```

First, the ongoing user query is combined with past queries using the $\overline{\wedge}$ operator. If this combination leads to access all attributes of a violating query, then the violating query is labeled. When this labeling makes any violating transaction completely labeled the ongoing query is denied.

5.5.2 Audit-based query tracking

In this section, we describe a labeling method that is achieved at both attribute and tuple levels. This labeling method takes advantage of auditing approaches that have been proposed in the literature.

Next, we start by introducing the auditing context and relevant definitions in this field. Then, we discuss how audit techniques could be considered in order to perform query tracking solution.

5.5.2.1 Query auditing

Query auditing approaches aim at investigating unauthorized access a posteriori. This vision is dual to classical access control mechanisms where authorization decision is made before the query is evaluated. Audit systems in the other hand check if an access to a particular information has been performed after query evaluation. This induces to record all user interactions with the system in a so called audit trail. This audit trail could be exploited offline and one is able to determine if a user accessed sensitive information.

Usually, query auditing is performed at tuple level. In particular, it aims at detecting if a user query has accessed a particular tuple. When such a tuple contains sensitive information, this concept allows modeling information that need to be protected.

Definition 29 (Essential tuple) *A tuple t is said to be essential for a query Q if its omission makes a difference in the results of Q.*

Another important definition is the one of audit expression.

Definition 30 (Audit expression) *An audit expression is a query that emphasizes the part of data to be protected (audited).*

The intuition behind these definitions is that an essential tuple is a tuple that belongs to the query results. In other words, the tuple satisfies the schema and the constraints of the query. From an audit point of view, an audit expression describes the part of data which is prohibited. If a user accesses any essential tuple of an audit expression then a violation occurs.

Query auditing techniques fall into two categories: data dependent and data independent. In data dependent techniques [Agrawal 2004, Kaushik 2011] the audit checks if a sensitive information on the current data instance has been accessed. The data independent techniques [Machanavajjhala 2006, Miklau 2004] check if there exists an instance in which the query accessed a prohibited information. This difference is characterized by the way an essential tuple is defined. Next, we provide both definitions of essential tuple.

Definition 31 (Essential tuple - data dependent) *A tuple t is said to be essential for a query Q if its omission from the current instance makes a difference in the results of Q, i.e., $Q(I) \neq Q(I) - \{t\}$ for the current instance I.*

This definition states that an essential tuple in data dependent settings is a tuple that belongs to query results on the current database instance.

Definition 32 (Essential tuple - data independent) *A tuple t is said to be essential for a query Q if there exists an instance for which its omission makes a difference in the results of Q, i.e., $\exists$ an instance I such that: $Q(I) \neq Q(I) - \{t\}$*

This definition states that an essential tuple in data independent settings is a tuple for which there exists a database instance where this tuple belongs to query results.

5.5.2.2 Auditing a set of queries

In [Motwani 2008], the authors tackle the problem of auditing a batch of queries. This problem aims at generalizing the classical auditing problem

where a single query is considered. In this work the problem could be stated as follows: *"Given a set of queries $Q = \{Q_1, Q_2, ..., Q_m\}$, an audit expression A, and a defined notion of suspiciousness S, is Q suspicious with respect to A?"*

The authors of [Motwani 2008] studied this problem for different definitions of suspiciousness and different classes of queries. They assume in all cases that A is a conjunctive query. The main definitions of suspiciousness they consider are the following:

- Semantic suspiciousness: It is a data dependent approach that extends previous data dependent solutions with the ability of considering a set of queries. This approach could fail to identify a batch of queries as being suspicious.
- Strong syntactic suspiciousness: It is a data independent approach that considers a set of queries. This approach aims at detecting suspicious batch of queries as well as limiting false positive detection. Unfortunately, this approach has been proven to be NP-hard even for select-project queries without inequalities.
- Weak syntactic suspiciousness: This approach is also data independent and considers a set of queries. It is a relaxation of the previous approach. This relaxation induces more false positive than the strong syntactic suspiciousness approach. This approach is NP-hard as well but for conjunctive queries.

Next, we present the strong syntactic suspiciousness approach. Indeed, we believe that strong syntactic approach is a good middle ground between the semantic approach that fail in detecting some suspicious batch of queries and the weak syntactic approach which could induce a large amount of false positives.

Definition 33 (Strong syntactic suspiciousness) *A batch of queries Q is strongly syntactically suspicious with respect to an audit expression A, if for any database instance I there is some subset $Q' \subseteq Q$ such that:*

- *A tuple $t \in T$ (the cross-product of tables common to every query in Q' and A) is essential to every query in Q' and to A.*

- *The queries Q' together access all attributes of V.*

The intuition behind this definition is that a batch of queries could access parts of an essential tuple separately. When put together these parts allow to reconstruct the essential tuple. Thus, leading to a violation.

Algorithm 7 describes the general process for achieving the strong syntactic suspiciousness approach proposed by [Motwani 2008]. This process starts by considering each query Q_i belonging to the batch of queries given as input. Then, it considers if the selective constraints of Q_i and A are compatible. The selective constraints are the constraints that a tuple t needs to satisfy in order to belong to query results. Note that the selection constraints are the constraints used in the where clause in an SQL notation.

The selective constraints of Q_i and A are compatible if there exists, in some instance I, a tuple t that belongs to both Q_i and A. In other words, the formula built using the conjunction of the two constraints is satisfiable.

If the constraints of Q_i and A are compatible then all common attributes of Q_i and A are labeled. At the end of this process, when all queries of Q have been considered, the idea is to check if all attributes of A have been labeled. If it is the case, this means that the combination of queries in Q leads to a suspicious situation with respect to A. Otherwise, they are not suspicious.

Algorithm 7: Strong Syntactic Auditor [Motwani 2008]

input : Q a set of queries.
A an audit expression.
output: Decision about the suspiciousness of Q with respect to A

```
1 foreach Q_i ∈ Q do
      if The constraints selecting the tuple of Q_i and A are compatible
      then
2         label all column of A as accessed by Q_i;
  if All column of A are labeled then
3     return suspicious ;
4 else
5     return unsuspicious ;
```

5.5.2.3 Query tracking with auditing

In this section we describe how we could take advantage from auditing techniques in order to achieve the query tracking solution. Recall that we have a set of violating transactions. Each violating transaction is a set of queries.

Algorithm 8: QueryTrackingWithAudit

input : $\mathcal{VT} = \{T_1, T_2..., T_n\}$, the set of violating transactions. $PastQueries = \{Q_1^u, Q_2^u..., Q_m^u\}$, the set of already executed queries. Q^u the ongoing query.

output: Decision to accept or reject each Q

```
1  foreach T_i ∈ VT do
2      foreach Q_j ∈ T_i do
3          if StrongSyntacticAuditor (PastQueries ∪ {Q}) = suspicious
           then
4              label Q_j ;
5              if all queries of T_i are labeled then
6                  cancel the labeling of Q_j;
7                  reject Q^u ;
8              else
9                  add Q^u to PastQueries;
10                 evaluate Q^u ;
11         else
12             add Q^u to PastQueries;
13             evaluate Q^u ;
```

Algorithm 8 describes the general process of the query tracking approach while using audit techniques. Here the audit algorithm that is used could be any of those presented in the previous section. Depending on the guarantees that are desired one could use either semantic or syntactic auditing.

The idea is to consider each query of any violating transaction as a view that we would like to audit with respect to a set of queries. This set of queries is constructed by considering all past queries and the current (ongoing) query. This algorithm determines if the ongoing query should be evaluated or rejected.

To this end, we use the results of the auditing of the set of queries containing past queries as well as the ongoing query to know if this set of queries is suspicious with respect to each query of all violating transactions. If this set induces the access to one of the queries of a violating transaction then this query is labeled. If the set of past and ongoing queries makes a whole transaction labeled then the ongoing query needs to be denied. Otherwise, the current query is evaluated and added to the set of past queries.

5.5.3 Comparison of query tracking approaches

In this section, we describe the advantages and drawbacks of query tracking approaches. The main difference between the attribute-based and the audit-based approaches is how to detect if a batch of queries access a violating query. Indeed, the rest of the reasoning remains the same which is labeling violating queries and denied a user query that makes a whole transaction labeled.

The two approaches behave differently mainly on two metrics. The first one is the complexity of the approach. The second one is the precision of the detection.

- **Complexity:** the attribute-based approach is achieved in polynomial time while the audit-based approach is NP-hard. This means that the attribute-based approach is more efficient computationally wise, in particular, since the query tracking is achieved at runtime. Indeed, each labeling is an extra processing before being able to evaluate the query. So, the more the labeling is costly the more query evaluation is slowed down.

- **Precision:** this metric measures the number of false positive that an approach could generate. A false positive occurs when a legitimate query is denied even though it could not lead to any violation. In this metric, audit-based approach is better. Indeed, this approach considers combination of queries at both column and tuple levels which makes it more precise when considering combination. On the other hand, the attribute-based approach restricts itself to column level. In this setting some queries could be considered as a possible combination while actually they could not be combined. To illustrate this case we consider

the following example: consider two queries $Q_1(A, B)$ and $Q_2(B, C)$. According to the attribute-based approach these queries could be combined (which is true for some case). Now, consider that Q_1 selects only tuples for which $B > 10$ and Q_2 selects only tuples for which $B < 5$. In this case Q_1 and Q_2 could not be combined. This example illustrates that the attribute-based could wrongfully consider that queries could be combined. This induces rejecting some queries which could have been safely evaluated.

5.6 Comparison of the two solutions

In this section we compare the two solutions described in the reconfiguration phase. We consider an abstract query tracking solution regardless if it is attribute-based or audit-based. Here, we discuss the advantages and drawbacks of both policy revision and query tracking.

5.6.1 Policy revision

- Advantage: All the processing is completed at the design time. The mediator policy is modified by adding a set of authorization rules. This modification ensures that no violating transaction could be completed and thus no violations could occur. This ensures that the system performances will not be affected.

- Drawback: This solution could be too restrictive since the decision about denying queries is made at design time. For a particular transaction a query could be denied even if the user did (and will) not perform any other query of this transaction. For example, consider the following transaction $T = \{Q_1, Q_2, Q_3\}$. We assume that Q_1 has been denied because it belongs to the minimal set of queries that covers all transactions. In this case a user could not evaluate Q_1 even though she/he has in mind to issue neither Q_2 nor Q_3.

Table 5.1: Experiments results for the reconfiguration phase

Name	$\|\mathcal{VT}\|$	$\|P'\|$	$QTrack$ (ms)
Yeast	5	7	<1
Chess	20	21	<1
Breast W.	37	20	1
Abalone	17	23	1
Synthetic 1	130	54	4
Synthetic 2	1737	263	8
Bank	9137	513	10

5.6.2 Query tracking

- Advantage: This solution maximizes the information available to the user. Indeed, it denies a query only if it could lead to a violation. A query leads to a violation if it could be combined with past queries and semantic constraints to obtain sensitive information.

- Drawback: This solution requires keeping track of all the queries that a user has issued. In an authorization sensitive context the history should be kept forever. Also, an additional processing is needed before the query execution. Even legitimate queries need to be checked with past queries before being evaluated. The tracking therefore also affects the performances by increasing CPU and memory utilization. Thus, slowing down query evaluation.

To summarize, no solution is clearly better than the other. Depending on the requirement of the targeted application one could either enforce policy revision or query tracking. Indeed, applications where query answering time is important could not afford using query tracking solution. In this case policy revision is the better fit. On the other hand, applications aiming at maximizing data availability will choose the query tracking solution. Indeed, the policy revision could be too restrictive.

5.7 Experiments

Table 5.1 summarizes the experiments that we have conducted regarding the reconfiguration phase. We used the same data sets as for the detection phase. This information reported in table 5.1 are the following:

- $|\mathcal{VT}|$ is number of minimal violating transactions that have been generated for the corresponding dataset. This result is the one of the detection phase.
- For the policy revision approach we have measured $|P'|$ representing the number of rules that need to be added, at design time, to the policy in order to avoid any violation. $|P'|$ is generated using query cancellation (algorithm 4) and generalization for a policy (algorithm 5).
- For the query tracking approach we have measured $QTrack$ the average time, in *ms*, that is needed to check if a query induces the labeling of a whole transaction.

Table 5.1 illustrates the relation between the set of violating transactions and the solutions performed in the reconfiguration phase. For the query tracking we observe that query labeling induces some extra time that will be added to query evaluation time.

For policy revision, figure 5.1 illustrates the relation between the number of transactions and the number of additional rules that are extracted. It shows that the more violating transactions are detected the more authorization rules need to be added to the mediator's policy.

5.8 Discussion

In this chapter we have presented how to deal with previously detected violating transactions. We have proposed two different solutions that are enforced at either design time or runtime. Then, we have compared both solutions. For the design time approach that we refer to as policy revision, we have shown that the underlying problem is NP-complete. This result induces the use of exponential algorithm. Nevertheless, the experiments performed on the

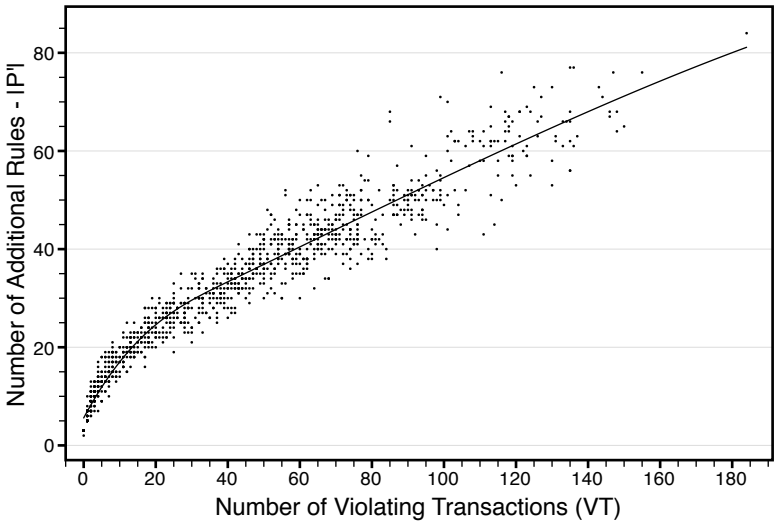

Figure 5.1: Number of violating transactions and number of added rules

policy revision solution demonstrate its practicability, in particular, since all processing of this solution are done offline. The second solution is achieved at runtime. We refer to this solution as query tracking. For this solution the idea is to keep all past queries and check if combination of past queries and the current query induces a violation. To this end, we discussed two different ways of how queries could be combined. The first one at attribute level. The second at both attribute and tuple levels. The second solution is inspired by auditing techniques but induces more processing time to be achieved.

We have discussed how both of these solutions solve the problem of violating transactions. The comparison of the two solutions showed that no solution outperforms the other if one considers all the criteria. In other words, either solution could be better depending on the targeted application.

Chapter 6

Conclusion

Contents

In this thesis, we considered the problem of controlling the access to mediators built using data integration approaches. We started by considering different research fields relevant to our problem, namely, data integration, access control, inference problem and policy composition. Based on this study, we formally defined our problem. We selected the GAV approach to build a mediator. Then, we considered access control policies defined by a set of authorization views that specify prohibitions. Finally, we focused on the role of functional dependencies in allowing malicious users to infer sensitive information. In this context, our aim was to generate an appropriate policy to be attached to the mediator. This policy is required to preserve the sources' policies and prevent against inferences. Preserving the sources means that if an access is forbidden in a source it should also be forbidden at the mediator. Preventing against inferences means that users could not perform an indirect access to sensitive data. An indirect access occurs when several non sensitive pieces of information are combined to obtain sensitive information.

To achieve this goal we proposed an incremental methodology that allows generating the policy of the mediator according to the previous requirements. In our methodology we made the following contributions:

- We described how to combine sources' policies in order to generate a preliminary global policy. We considered constraint-based access control enforced using authorization views. We provided a method that aims at preserving all sources' constraints and thus preserving the sources' policies. We showed that a naive policy combination could be insufficient

to protect sensitive information against inferences by means of indirect access.

- We characterized how malicious users could build a transaction (*i.e.,* a set of queries) that could induce disclosing sensitive information. To this end, we have considered the notion of lossless join to show how queries could be combined to reconstruct some particular view using functional dependencies.

- We provided a graph-based approach able to enumerate all violating transactions. A violating transaction is a set of queries that when combined allow malicious users to synthesize prohibited information.

- We proposed two solutions that are able to prevent users from completing any violating transaction (*i.e.,* accessing all queries of any single transaction). The first one, *policy revision*, could be implemented at design time. The second one, *query tracking*, could be implemented at runtime.

  - The policy revision approach aims at generating a set of minimum queries that covers every violating transaction. The idea is to forbid users from accessing this set of queries. We showed that the corresponding problem to the policy revision approach is NP-complete.
  - The query tracking approach is a history-based approach that records all past queries. This approach checks if the combination of the current query and the past queries induces the completion of a violating transaction. To this end, we proposed two methods: the first is attribute-based and it considers the combination at the attribute level. The second method is based on query auditing techniques. This method is more accurate than the attribute-based method but it is also more complex and thus more time consuming.

6.1 Future work

Our methodology achieves our objectives on a core setting of our problem. Indeed, we made some assumptions on several concepts. For instance we targeted conjunctive queries as a query language, functional dependencies as

semantic constraints and the GAV approach for data integration. For each of these assumptions we selected the core subset of the corresponding concept. Some relevant extensions could be:

- Authorization language: We have focused on negative authorizations and an open world assumption. This means that a policy is defined using a set of rules expressing prohibitions. Also, an access is denied when it is explicitly prohibited. This setting is as expressive as considering positive authorization and a closed world assumption. It could be interesting to allow sources to specify both positive and negative authorizations. The main impact of this would be on the first phase of our methodology, namely, the propagation phase. In this context, propagating the sources' policies could induce conflicts among the global policy. One will need to add a conflict resolution procedure to the propagation phase. This procedure could be inspired by approaches specifically designed to deal with conflict resolution such as those described in [Jajodia 1997, Cuppens 2007].

- Query language: Aggregate queries could be useful in some access control scenarios when disclosing information about a set of individuals is allowed whereas disclosing information about a single individual is forbidden. This extension calls for concepts and techniques from statistical databases [Adam 1989]. Indeed, modifying the query language provides malicious users with new methods to infer sensitive information.

- Integration approach: We considered the case where GAV integration is performed to construct the mediator. Other approaches for data integration could also be used (*e.g.,* LAV and GLAV methods). The idea is to investigate how the mappings between the mediator's elements and the sources relations could induce prohibited inferences.

- Semantic constraints: The use of semantic constraints by malicious users in order to obtain prohibited information was our main objective in this thesis. We restricted our work to functional dependencies because they represent natural semantic constraints in database systems. We could extend our use of functional dependencies as follows:

  - Extend the proposed approach to deal with partial functional dependencies [Kivinen 1995]. A partial functional dependency is a

functional dependency that is not satisfied by all the tuples. Indeed, if a partial functional dependency holds in a large amount of tuples, it could be used to infer sensitive information. In order to achieve this extension, one needs to extend the transition graph by adding, for each functional dependency, the percentage of tuples for which it holds. Then, apply the transitivity rule of partial functional dependencies when generating new nodes. A technical issue is to determine a threshold from which a partial functional dependency is considered to be harmful. In other words, at which percentage a partial functional dependency allows malicious users to infer sensitive information.

- Taking into account of other kinds of semantic constraints such as inclusion dependencies [Casanova 1982], multivalued dependencies [Fagin 1977] and tuple-generating dependencies [Abiteboul 1995]. The idea is to investigate how these kinds of dependencies could allow malicious users to derive sensitive information. This extension impacts the second phase of our methodology, namely, the detection phase. Indeed, new opportunities to infer prohibited information offered by these semantic constraints need to be considered while constructing the violating transaction. The propagation and reconfiguration phases should not be impacted by this extension.

- Existing policy at mediator level: This extension assumes that a policy is already in place at the mediator level. The idea is to use our approach to check if the defined mediator's policy preserves the sources' policies. One way to consider this issue is to adapt compliance techniques [Dougherty 2006, Becker 2009] to check whether the defined mediator policy complies with the inferred policy of our approach. In other words, the defined mediator's policy should not allow an access that is prohibited by the inferred policy. This extension could ensure that a mediator policy defined by an administrator actually preserves the sources' policies.

Bibliography

[Abiteboul 1995] Serge Abiteboul, Richard Hull and Victor Vianu. Foundations of databases, volume 8. Addison-Wesley Reading, 1995. (Cited on pages 2, 46, 47, 49 and 108.)

[Achugbue 1979] James O Achugbue and Francis Y Chin. *The effectiveness of output modification by rounding for protection of statistical databases.* INFOR. Canadian Journal of Operational Research and Information Processing, vol. 17, no. 3, 1979. (Cited on page 29.)

[Acquisti 2005] Alessandro Acquisti and Jens Grossklags. *Privacy and rationality in individual decision making.* IEEE Security & Privacy, vol. 2, pages 24–30, 2005. (Cited on page 1.)

[Adam 1989] Nabil R Adam and John C Worthmann. *Security-control methods for statistical databases: a comparative study.* ACM Computing Surveys (CSUR), vol. 21, no. 4, pages 515–556, 1989. (Cited on pages 27, 28 and 107.)

[Agrawal 2002] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant and Yirong Xu. *Hippocratic databases.* In Proceedings of the 28th international conference on Very Large Data Bases, pages 143–154. VLDB Endowment, 2002. (Cited on page 1.)

[Agrawal 2004] Rakesh Agrawal, Roberto Bayardo, Christos Faloutsos, Jerry Kiernan, Ralf Rantzau and Ramakrishnan Srikant. *Auditing compliance with a hippocratic database.* In Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, pages 516–527. VLDB Endowment, 2004. (Cited on pages 4 and 96.)

[Ahn 1999] Gail-Joon Ahn and Ravi Sandhu. *The RSL99 language for role-based separation of duty constraints.* In Proceedings of the fourth ACM workshop on Role-based access control, pages 43–54. ACM, 1999. (Cited on page 22.)

[Aho 1979] Alfred V Aho, Catriel Beeri and Jeffrey D Ullman. *The theory of joins in relational databases.* ACM Transactions on Database Systems (TODS), vol. 4, no. 3, pages 297–314, 1979. (Cited on pages 55 and 69.)

[Al-Kahtani 2004] Mohammad A Al-Kahtani and Ravi Sandhu. *Rule-based RBAC with negative authorization*. In Computer Security Applications Conference, 2004. 20th Annual, pages 405–415. IEEE, 2004. (Cited on page 23.)

[Allistene 2013] Allistene. *Contribution d'Allistene et des Pôles de compétitivité à la Stratégie Nationale de Recherche Sciences et Technologie du Numérique*, 2013. https://www.allistene.fr/contribution-dallistene-et-des-poles-de-competitivite-a-la-strategie-nationale-de-recherche-sciences-et-technologie-du-numerique/ . (Cited on page 2.)

[Armstrong 1974] William Ward Armstrong. *Dependency Structures of Data Base Relationships*. In IFIP Congress'74, pages 580–583, 1974. (Cited on pages 47 and 48.)

[Ashley 2003] Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers and Matthias Schunter. *Enterprise privacy authorization language (EPAL 1.2)*. Submission to W3C, 2003. (Cited on page 1.)

[Bache 2013] K. Bache and M. Lichman. *UCI Machine Learning Repository*, 2013. http://archive.ics.uci.edu/ml. (Cited on page 77.)

[Beck 1980] Leland L Beck. *A security machanism for statistical database*. ACM Transactions on Database Systems (TODS), vol. 5, no. 3, pages 316–3338, 1980. (Cited on page 29.)

[Becker 2004] Moritz Y Becker and Peter Sewell. *Cassandra: Distributed access control policies with tunable expressiveness*. In Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on, pages 159–168. IEEE, 2004. (Cited on page 4.)

[Becker 2009] Moritz Y Becker. *Specification and analysis of dynamic authorisation policies*. In Computer Security Foundations Symposium, 2009. CSF'09. 22nd IEEE, pages 203–217. IEEE, 2009. (Cited on page 108.)

[Bell 1973] D Elliott Bell and Leonard J LaPadula. *Secure computer systems: Mathematical foundations*. Rapport technique, DTIC Document, 1973. (Cited on page 21.)

[Beneventano 2000] Domenico Beneventano, Sonia Bergamaschi, Silvana Castano, Alberto Corni, R Guidetti, G Malvezzi, Michele Melchiori and Maurizio Vincini. *Information integration: the MOMIS project demonstration*. In VLDB, pages 611–614, 2000. (Cited on page 17.)

[Bertino 1995] Elisa Bertino, Sushil Jajodia and Pierangela Samarati. *Database security: research and practice*. Information systems, vol. 20, no. 7, pages 537–556, 1995. (Cited on page 14.)

[Bertino 1999] Elisa Bertino, Sushil Jajodia and Pierangela Samarati. *A flexible authorization mechanism for relational data management systems*. ACM Transactions on Information Systems (TOIS), vol. 17, no. 2, pages 101–140, 1999. (Cited on page 50.)

[Bertino 2001] Elisa Bertino, Piero Andrea Bonatti and Elena Ferrari. *TRBAC: A temporal role-based access control model*. ACM Transactions on Information and System Security (TISSEC), vol. 4, no. 3, pages 191–233, 2001. (Cited on page 19.)

[Bertino 2005] Elisa Bertino and Ravi Sandhu. *Database security-concepts, approaches, and challenges*. Dependable and Secure Computing, IEEE Transactions on, vol. 2, no. 1, pages 2–19, 2005. (Cited on pages 14 and 19.)

[Bertino 2011] Elisa Bertino, Gabriel Ghinita and Ashish Kamra. Access control for databases: concepts and systems. Now Publishers Inc, 2011. (Cited on page 7.)

[Bertossi 1984] Alan A. Bertossi. *Dominating sets for split and bipartite graphs*. Inf. Process. Lett., vol. 19, no. 1, pages 37–40, September 1984. (Cited on page 88.)

[Biba 1977] Kenneth J Biba. *Integrity considerations for secure computer systems*. Rapport technique, DTIC Document, 1977. (Cited on page 21.)

[Bleiholder 2008] Jens Bleiholder and Felix Naumann. *Data fusion*. ACM Computing Surveys (CSUR), vol. 41, no. 1, page 1, 2008. (Cited on page 3.)

[Bonatti 1997] Piero A Bonatti, Maria Luisa Sapino and VS Subrahmanian. *Merging heterogeneous security orderings*. Journal of Computer Security, vol. 5, no. 1, pages 3–29, 1997. (Cited on page 40.)

[Bonatti 2000] Piero Bonatti and Pierangela Samarati. *Regulating service access and information release on the web*. In Proceedings of the 7th ACM conference on Computer and communications security, pages 134–143. ACM, 2000. (Cited on pages 19, 23 and 60.)

[Bonatti 2002] Piero Bonatti, Sabrina De Capitani di Vimercati and Pierangela Samarati. *An algebra for composing access control policies*. ACM Transactions on Information and System Security (TISSEC), vol. 5, no. 1, pages 1–35, 2002. (Cited on pages 8 and 39.)

[Brodsky 2000] Alexander Brodsky, Csilla Farkas and Sushil Jajodia. *Secure databases: Constraints, inference channels, and monitoring disclosures*. Knowledge and Data Engineering, IEEE Transactions on, vol. 12, no. 6, pages 900–919, 2000. (Cited on page 33.)

[Carey 1995] Michael J Carey, Laura M Haas, Peter M Schwarz, Manish Arya, William F Cody, Ronald Fagin, Myron Flickner, Allen W Luniewski, Wayne Niblack, Dragutin Petkovic*et al. Towards heterogeneous multimedia information systems: The Garlic approach*. In Research Issues in Data Engineering, 1995: Distributed Object Management, Proceedings. RIDE-DOM'95. Fifth International Workshop on, pages 124–131. IEEE, 1995. (Cited on page 17.)

[Casanova 1982] Marco A Casanova, Ronald Fagin and Christos H Papadimitriou. *Inclusion dependencies and their interaction with functional dependencies*. In Proceedings of the 1st ACM SIGACT-SIGMOD symposium on Principles of database systems, pages 171–176. ACM, 1982. (Cited on page 108.)

[Ceri 1983] Stefano Ceri and Giuseppe Pelagatti. *Correctness of query execution strategies in distributed databases*. ACM Transactions on Database Systems (TODS), vol. 8, no. 4, pages 577–607, 1983. (Cited on page 3.)

[Chang 1991] C-C Chang and T-C Wu. *Remote password authentication with smart cards*. Computers and Digital Techniques, IEE Proceedings E, vol. 138, no. 3, pages 165–168, 1991. (Cited on page 4.)

[Chawathe 1994] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman and Jennifer Widom. *The TSIMMIS project: Integration of heterogenous information sources*. 1994. (Cited on pages 3, 7 and 17.)

[Chen 1976] Peter Pin-Shan Chen. *The entity-relationship model - toward a unified view of data.* ACM Transactions on Database Systems (TODS), vol. 1, no. 1, pages 9–36, 1976. (Cited on page 31.)

[Clifton 1996] Chris Clifton and Don Marks. *Security and privacy implications of data mining.* In ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, pages 15–19. Citeseer, 1996. (Cited on page 35.)

[Clifton 2004] Chris Clifton, Murat KantarcioÇ§lu, AnHai Doan, Gunther Schadow, Jaideep Vaidya, Ahmed Elmagarmid and Dan Suciu. *Privacy-preserving data integration and sharing.* In Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, pages 19–26. ACM, 2004. (Cited on page 25.)

[Codd 1970] Edgar F Codd. *A relational model of data for large shared data banks.* Communications of the ACM, vol. 13, no. 6, pages 377–387, 1970. (Cited on page 2.)

[Commission 1998] Federal Trade Commission*et al.* *Children's online privacy protection act of 1998.* Online at: http://www.cdt.org/legislation/105th/privacy/coppa.html, 1998. (Cited on page 5.)

[Commission 2000] Federal Trade Commission*et al.* *FTC Announces Settlement with Bankrupt Website, Toysmart. com, Regarding Alleged Privacy Policy Violations.* FTC Release, July, vol. 21, 2000. (Cited on page 5.)

[Commission 2014] European Commission. *Horizon 2020 Work Programme 2014 - 2015*, 2014. http://ec.europa.eu/research/participants/data/ref/h2020/wp/2014_2015/main/h2020-wp1415-leit-ict_en.pdf. (Cited on page 2.)

[Control 2003] Centers For Disease Control, Prevention*et al.* *HIPAA privacy rule and public health. Guidance from CDC and the US Department of Health and Human Services.* MMWR: Morbidity and Mortality Weekly Report, vol. 52, no. Suppl. 1, pages 1–17, 2003. (Cited on page 5.)

[Cook 1971] Stephen A Cook. *The complexity of theorem-proving procedures*. In Proceedings of the third annual ACM symposium on Theory of computing, pages 151–158. ACM, 1971. (Cited on page 87.)

[Cuppens 2003] Frédéric Cuppens and Alexandre Miège. *Modelling contexts in the Or-BAC model*. In Computer Security Applications Conference, 2003. Proceedings. 19th Annual, pages 416–425. IEEE, 2003. (Cited on page 19.)

[Cuppens 2007] Frédéric Cuppens, Nora Cuppens-Boulahia and Meriam Ben Ghorbel. *High level conflict management strategies in advanced access control models*. Electronic Notes in Theoretical Computer Science, vol. 186, pages 3–26, 2007. (Cited on page 107.)

[Damiani 2002] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi and Pierangela Samarati. *A fine-grained access control system for XML documents*. ACM Transactions on Information and System Security (TISSEC), vol. 5, no. 2, pages 169–202, 2002. (Cited on page 24.)

[Deen 1987] S. Misbah Deen, RR Amin and Malcolm C. Taylor. *Data integration in distributed databases*. Software Engineering, IEEE Transactions on, no. 7, pages 860–864, 1987. (Cited on page 16.)

[Delugach 1996] Harry S. Delugach and Thomas H. Hinke. *Wizard: A database inference analysis and detection system*. Knowledge and Data Engineering, IEEE Transactions on, vol. 8, no. 1, pages 56–66, 1996. (Cited on pages 33 and 35.)

[Denning 1976] Dorothy E Denning. *A lattice model of secure information flow*. Communications of the ACM, vol. 19, no. 5, pages 236–243, 1976. (Cited on pages 19 and 21.)

[Denning 1980] Dorothy E Denning. *Secure statistical databases with random sample queries*. ACM Transactions on Database Systems (TODS), vol. 5, no. 3, pages 291–315, 1980. (Cited on page 29.)

[Denning 1983] Dorothy E. Denning and Jan Schlorer. *Inference controls for statistical databases*. Computer, vol. 16, no. 7, pages 69–82, 1983. (Cited on page 28.)

[di Vimercati 1996] Sabrina De Capitani di Vimercati and Pierangela Samarati. *An authorization model for federated systems.* In Computer Security ESORICS 96, pages 99–117. Springer, 1996. (Cited on page 4.)

[di Vimercati 1997] Sabrina De Capitani di Vimercati and Pierangela Samarati. *Authorization specification and enforcement in federated database systems.* J. Comput. Secur., vol. 5, no. 2, pages 155–188, March 1997. (Cited on page 52.)

[Diffie 1976] Whitfield Diffie and Martin E Hellman. *New directions in cryptography.* Information Theory, IEEE Transactions on, vol. 22, no. 6, pages 644–654, 1976. (Cited on page 4.)

[Domingo-Ferrer 2008] Josep Domingo-Ferrer and Vicenç Torra. *A critique of k-anonymity and some of its enhancements.* In Availability, Reliability and Security, 2008. ARES 08. Third International Conference on, pages 990–993. IEEE, 2008. (Cited on page 36.)

[Dougherty 2006] Daniel J Dougherty, Kathi Fisler and Shriram Krishnamurthi. *Specifying and reasoning about dynamic access-control policies.* In Automated Reasoning, pages 632–646. Springer, 2006. (Cited on page 108.)

[Duschka 2000] Oliver M Duschka, Michael R Genesereth and Alon Y Levy. *Recursive query plans for data integration.* The Journal of Logic Programming, vol. 43, no. 1, pages 49–73, 2000. (Cited on page 18.)

[Fagin 1977] Ronald Fagin. *Multivalued dependencies and a new normal form for relational databases.* ACM Transactions on Database Systems (TODS), vol. 2, no. 3, pages 262–278, 1977. (Cited on page 108.)

[Fagin 1978] Ronald Fagin. *On an authorization mechanism.* ACM Transactions on Database Systems (TODS), vol. 3, no. 3, pages 310–319, 1978. (Cited on page 4.)

[Fagin 2005] Ronald Fagin, Phokion G Kolaitis, Renée J Miller and Lucian Popa. *Data exchange: semantics and query answering.* Theoretical Computer Science, vol. 336, no. 1, pages 89–124, 2005. (Cited on page 3.)

[Farkas 2002] Csilla Farkas and Sushil Jajodia. *The inference problem: a survey.* ACM SIGKDD Explorations Newsletter, vol. 4, no. 2, pages 6–11, 2002. (Cited on pages 5, 8 and 27.)

[Fellegi 1972] Ivan P Fellegi. *On the question of statistical confidentiality.* Journal of the American Statistical Association, vol. 67, no. 337, pages 7–18, 1972. (Cited on page 28.)

[Fellegi 1974] IP Fellegi and JJ Phillips. *Statistical Confidentiality: Some Theory and Application to Data Dissemination.* In Annals of Economic and Social Measurement, Volume 3, number 2, pages 101–112. NBER, 1974. (Cited on pages 28 and 29.)

[Fernandez 1994] Eduardo B. Fernandez, Ehud Gudes and Haiyan Song. *A model for evaluation and administration of security in object-oriented databases.* Knowledge and Data Engineering, IEEE Transactions on, vol. 6, no. 2, pages 275–292, 1994. (Cited on page 22.)

[Ferraiolo 1995] David Ferraiolo, Janet Cugini and D Richard Kuhn. *Role-based access control (RBAC): Features and motivations.* In Proceedings of 11th Annual Computer Security Application Conference, pages 241–48, 1995. (Cited on pages 19 and 22.)

[Friedman 1980] Arthur D Friedman and Lance J Hoffman. *Towards a fail-safe approach to secure databases.* In 1980 IEEE Symposium on Security and Privacy, page 0018. IEEE Computer Society, 1980. (Cited on page 28.)

[Friedman 1999] Marc Friedman, Alon Y Levy, Todd D Millstein*et al. Navigational plans for data integration.* AAAI/IAAI, vol. 1999, pages 67–73, 1999. (Cited on page 18.)

[Garcia-Molina 2002] Hector Garcia-Molina, Jeffrey D. Ullman and Jennifer Widom. Database systems - the complete book (international edition). Pearson Education, 2002. (Cited on page 1.)

[Garey 2002] Michael R Garey and David S Johnson. Computers and intractability, volume 29. wh freeman, 2002. (Cited on page 86.)

[Goguen 1984] Joseph A Goguen and José Meseguer. *Unwinding and inference control.* 1984. (Cited on page 30.)

[Goh 1999] Cheng Hian Goh, Stéphane Bressan, Stuart Madnick and Michael Siegel. *Context interchange: New features and formalisms for the intelligent integration of information*. ACM Transactions on Information Systems (TOIS), vol. 17, no. 3, pages 270–293, 1999. (Cited on page 17.)

[Graham 1972] G Scott Graham and Peter J Denning. *Protection: principles and practice*. In Proceedings of the May 16-18, 1972, spring joint computer conference, pages 417–429. ACM, 1972. (Cited on page 19.)

[Grahne 1999] Gosta Grahne and Alberto O Mendelzon. *Tableau techniques for querying information sources through global schemas*. In Database Theory-ICDT 99, pages 332–347. Springer, 1999. (Cited on page 18.)

[Griffiths 1976] Patricia P Griffiths and Bradford W Wade. *An authorization mechanism for a relational database system*. ACM Transactions on Database Systems (TODS), vol. 1, no. 3, pages 242–255, 1976. (Cited on page 4.)

[Haddad 2012] Mehdi Haddad, Mohand-Saïd Hacid and Robert Laurini. *Data Integration in Presence of Authorization Policies*. In 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2012, Liverpool, United Kingdom, June 25-27, 2012, pages 92–99, 2012. (Cited on page ix.)

[Haddad 2013a] Mehdi Haddad. *Contrôles d'accès pour l'intégration de données*. In 29eme journées Bases de Données Avancées (BDA), young researcher article. Fsrench conference, 2013. (Cited on page ix.)

[Haddad 2013b] Mehdi Haddad, Jovan Stevovic, Annamaria Chiasera, Yannis Velegrakis and Mohand-Said Hacid. *Access Control for Data Integration project Homepagel*, 2013. `http://disi.unitn.it/%7Estevovic/ACforDI.html`. (Cited on page 77.)

[Haddad 2014] Mehdi Haddad, Jovan Stevovic, Annamaria Chiasera, Yannis Velegrakis and Mohand-Saïd Hacid. *Access Control for Data Integration in Presence of Data Dependencies*. In Database Systems for Advanced Applications - 19th International Conference, DASFAA 2014, Bali, Indonesia, April 21-24, 2014. Proceedings, Part II, pages 203–217, 2014. (Cited on page ix.)

[Halevi 1999] Shai Halevi and Hugo Krawczyk. *Public-key cryptography and password protocols*. ACM Transactions on Information and System Security (TISSEC), vol. 2, no. 3, pages 230–268, 1999. (Cited on page 4.)

[Halevy 2001] Alon Y Halevy. *Answering queries using views: A survey*. The VLDB Journal, vol. 10, no. 4, pages 270–294, 2001. (Cited on pages 7, 17 and 18.)

[Halevy 2006] Alon Halevy, Anand Rajaraman and Joann Ordille. *Data integration: the teenage years*. In Proceedings of the 32nd international conference on Very large data bases, pages 9–16. VLDB Endowment, 2006. (Cited on page 3.)

[Harrison 1976] Michael A Harrison, Walter L Ruzzo and Jeffrey D Ullman. *Protection in operating systems*. Communications of the ACM, vol. 19, no. 8, pages 461–471, 1976. (Cited on page 20.)

[Hastie 1990] T. J. Hastie and R. J. Tibshirani. Generalized additive models. London: Chapman & Hall, 1990. (Cited on page 81.)

[Hoffman 1970] Lance J Hoffman and WF Miller. *Getting a personal dossier from a statistical data bank*. Datamation, vol. 16, no. 5, pages 74–75, 1970. (Cited on page 28.)

[Howard 1987] Ralph Howard. *Data encryption standard*. Information age, vol. 9, no. 4, pages 204–210, 1987. (Cited on page 4.)

[Huhtala 1999] Y. Huhtala, J. Kärkkäinen, P. Porkka and H. Toivonen. *TANE: An efficient algorithm for discovering functional and approximate dependencies*. The Computer Journal, vol. 42, no. 2, pages 100–111, 1999. (Cited on pages 52 and 77.)

[Hull 1996] Richard Hull and Gang Zhou. A framework for supporting data integration using the materialized and virtual approaches, volume 25. ACM, 1996. (Cited on pages 16 and 17.)

[Hur 2011] Junbeom Hur and Dong Kun Noh. *Attribute-based access control with efficient revocation in data outsourcing systems*. Parallel and Distributed Systems, IEEE Transactions on, vol. 22, no. 7, pages 1214–1221, 2011. (Cited on page 23.)

[Innovation 2013] Commission Innovation. *Un principe et sept ambitions pour l'innovation*, 2013. http://innovation-2030.entreprises.gouv.fr/pdf/Rapport_Innovation_BDV4.pdf. (Cited on page 2.)

[Jaeger 1996] Trent Jaeger and Atul Prakash. *Requirements of role-based access control for collaborative systems*. In Proceedings of the first ACM Workshop on Role-based access control, page 16. ACM, 1996. (Cited on page 22.)

[Jajodia 1991] Sushil Jajodia and Ravi Sandhu. *Toward a multilevel secure relational data model*. In ACM SIGMOD Record, volume 20, pages 50–59. ACM, 1991. (Cited on page 4.)

[Jajodia 1997] Sushil Jajodia, Pierangela Samarati, VS Subrahmanian and Eliza Bertino. *A unified framework for enforcing multiple access control policies*. In ACM Sigmod Record, volume 26, pages 474–485. ACM, 1997. (Cited on page 107.)

[Jajodia 2001] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino and VS Subrahmanian. *Flexible support for multiple access control policies*. ACM Transactions on Database Systems (TODS), vol. 26, no. 2, pages 214–260, 2001. (Cited on pages 8 and 39.)

[Jonscher 1994] Dirk Jonscher and Klaus R Dittrich. *An approach for building secure database federations*. In Proceedings of the 20th International Conference on Very Large Data Bases, pages 24–35. Morgan Kaufmann Publishers Inc., 1994. (Cited on page 4.)

[Kaushik 2011] Raghav Kaushik and Ravi Ramamurthy. *Efficient auditing for complex SQL queries*. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pages 697–708. ACM, 2011. (Cited on page 96.)

[Kirk 1995] Thomas Kirk, Alon Y Levy, Yehoshua Sagiv, Divesh Srivastava*et al.* *The information manifold*. In Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments, volume 7, pages 85–91, 1995. (Cited on page 17.)

[Kivinen 1995] Jyrki Kivinen and Heikki Mannila. *Approximate inference of functional dependencies from relations.* Theoretical Computer Science, vol. 149, no. 1, pages 129–149, 1995. (Cited on page 107.)

[Lamport 1981] Leslie Lamport. *Password authentication with insecure communication.* Communications of the ACM, vol. 24, no. 11, pages 770–772, 1981. (Cited on page 4.)

[Lampson 1974] Butler W Lampson. *Protection.* ACM SIGOPS Operating Systems Review, vol. 8, no. 1, pages 18–24, 1974. (Cited on page 19.)

[Landwehr 1981] Carl E Landwehr. *Formal models for computer security.* ACM Computing Surveys (CSUR), vol. 13, no. 3, pages 247–278, 1981. (Cited on page 4.)

[LeFevre 2004] Kristen LeFevre, Rakesh Agrawal, Vuk Ercegovac, Raghu Ramakrishnan, Yirong Xu and David DeWitt. *Limiting disclosure in hippocratic databases.* In Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, pages 108–119. VLDB Endowment, 2004. (Cited on page 1.)

[Lefons 1983] Ezio Lefons, Alberto Silvestri and Filippo Tangorra. *An Analytic Approach to Statistical Databases.* In VLDB, pages 260–274, 1983. (Cited on page 29.)

[Lenzerini 2002] Maurizio Lenzerini. *Data integration: A theoretical perspective.* In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 233–246. ACM, 2002. (Cited on pages 3, 7, 17 and 51.)

[Levene 1999] Mark Levene and George Loizou. A guided tour of relational databases and beyond. Springer Verlag, 1999. (Cited on pages 47 and 48.)

[Levy 1996] Alon Levy, Anand Rajaraman and Joann Ordille. *Querying heterogeneous information sources using source descriptions.* 1996. (Cited on page 18.)

[Li 2003] Ninghui Li, William H Winsborough and John C Mitchell. *Distributed credential chain discovery in trust management.* Journal of Computer Security, vol. 11, no. 1, pages 35–86, 2003. (Cited on page 4.)

[Li 2007] Ninghui Li, Tiancheng Li and Suresh Venkatasubramanian. *t-Closeness: Privacy Beyond k-Anonymity and l-Diversity.* In ICDE, volume 7, pages 106–115, 2007. (Cited on page 36.)

[Liew 1985] Chong K Liew, Uinam J Choi and Chung J Liew. *A data distortion by probability distribution.* ACM Transactions on Database Systems (TODS), vol. 10, no. 3, pages 395–411, 1985. (Cited on page 29.)

[Litwin 1990] Witold Litwin, Leo Mark and Nick Roussopoulos. *Interoperability of multiple autonomous databases.* ACM Computing Surveys (CSUR), vol. 22, no. 3, pages 267–293, 1990. (Cited on page 17.)

[Liu 2009] Jianhua Liu, Selnur Erdal, Scott A Silvey, Jing Ding, John D Riedel, Clay B Marsh and Jyoti Kamal. *Toward a fully de-identified biomedical information warehouse.* In AMIA Annual Symposium Proceedings, volume 2009, page 370. American Medical Informatics Association, 2009. (Cited on page 5.)

[Lunt 1990] Teresa F Lunt and Eduardo B Fernandez. *Database security.* ACM SIGMOD Record, vol. 19, no. 4, pages 90–97, 1990. (Cited on page 4.)

[Machanavajjhala 2006] Ashwin Machanavajjhala and Johannes Gehrke. *On the efficiency of checking perfect privacy.* In Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 163–172. ACM, 2006. (Cited on page 96.)

[Machanavajjhala 2007] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke and Muthuramakrishnan Venkitasubramaniam. *l-diversity: Privacy beyond k-anonymity.* ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 1, no. 1, page 3, 2007. (Cited on page 36.)

[Manolescu 2001] Ioana Manolescu, Daniela Florescu and Donald Kossmann. *Answering XML Queries on Heterogeneous Data Sources.* In VLDB, volume 1, pages 241–250, 2001. (Cited on page 17.)

[Meadows 1987] Catherine Meadows and Sushil Jajodia. *Integrity Versus Security in Multi-Level Secure Databases.* In DBSec, pages 89–101, 1987. (Cited on page 30.)

[Meadows 1988] Catherine Meadows and Sushil Jajodia. *Integrity versus security in multi-level secure databases.* In on Database Security: Status and Prospects, pages 89–101. North-Holland Publishing Co., 1988. (Cited on page 32.)

[Miklau 2004] Gerome Miklau and Dan Suciu. *A formal analysis of information disclosure in data exchange.* In Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pages 575–586. ACM, 2004. (Cited on page 96.)

[Miller 2000] Renée J Miller, Laura M Haas and Mauricio A Hernández. *Schema Mapping as Query Discovery.* In VLDB, volume 2000, pages 77–88, 2000. (Cited on page 3.)

[Morgenstern 1988] Matthew Morgenstern. *Controlling logical inference in multilevel database systems.* In Security and Privacy, 1988. Proceedings., 1988 IEEE Symposium on, pages 245–255. IEEE, 1988. (Cited on pages 30 and 35.)

[Motro 1981] Amihai Motro and Peter Buneman. *Constructing superviews.* In Proceedings of the 1981 ACM SIGMOD international conference on Management of data, pages 56–64. ACM, 1981. (Cited on page 16.)

[Motro 1989] Amihai Motro. *An access authorization model for relational databases based on algebraic manipulation of view definitions.* In Data Engineering, 1989. Proceedings. Fifth International Conference on, pages 339–347. IEEE, 1989. (Cited on page 24.)

[Motwani 2008] Rajeev Motwani, Shubha U Nabar and Dilys Thomas. *Auditing sql queries.* In Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on, pages 287–296. IEEE, 2008. (Cited on pages 4, 96, 97 and 98.)

[Narayanan 2008] Arvind Narayanan and Vitaly Shmatikov. *Robust deanonymization of large sparse datasets.* In Security and Privacy, 2008. SP 2008. IEEE Symposium on, pages 111–125. IEEE, 2008. (Cited on page 5.)

[Naumann 2006] Felix Naumann, Alexander Bilke, Jens Bleiholder and Melanie Weis. *Data Fusion in Three Steps: Resolving Schema, Tu-*

ple, and Value Inconsistencies. IEEE Data Eng. Bull., vol. 29, no. 2, pages 21–31, 2006. (Cited on page 3.)

[Oliva 2001] Marta Oliva and Fèlix Saltor. *Integrating multilevel security policies in multilevel federated database systems.* In Data and Application Security, pages 135–147. Springer, 2001. (Cited on page 40.)

[Osborn 2000] Sylvia Osborn, Ravi Sandhu and Qamar Munawer. *Configuring role-based access control to enforce mandatory and discretionary access control policies.* ACM Transactions on Information and System Security (TISSEC), vol. 3, no. 2, pages 85–106, 2000. (Cited on page 22.)

[Ozsoyoglu 1985] Gultekin Ozsoyoglu and Tzong-An Su. *Rounding and Inference Controlin Conceptual Models for Statistical Databases.* 1985. (Cited on page 29.)

[Özsu 2011] M Tamer Özsu and Patrick Valduriez. Principles of distributed database systems. Springer, 2011. (Cited on page 1.)

[Popa 2002] Lucian Popa, Yannis Velegrakis, Mauricio A Hernández, Renée J Miller and Ronald Fagin. *Translating web data.* In Proceedings of the 28th international conference on Very Large Data Bases, pages 598–609. VLDB Endowment, 2002. (Cited on page 3.)

[Pottinger 2000] Rachel Pottinger and Alon Y Levy. *A Scalable Algorithm for Answering Queries Using Views.* In VLDB, pages 484–495, 2000. (Cited on page 18.)

[Pottinger 2001] Rachel Pottinger and Alon Halevy. *MiniCon: A scalable algorithm for answering queries using views.* The VLDB Journal - The International Journal on Very Large Data Bases, vol. 10, no. 2-3, pages 182–198, 2001. (Cited on page 18.)

[Qian 1996] Xiaolei Qian. *Query folding.* In Data Engineering, 1996. Proceedings of the Twelfth International Conference on, pages 48–55. IEEE, 1996. (Cited on page 18.)

[Reiss 1984] Steven P Reiss. *Practical data-swapping: The first steps.* ACM Transactions on Database Systems (TODS), vol. 9, no. 1, pages 20–37, 1984. (Cited on page 29.)

[Rissanen 1977] Jorma Rissanen. *Independent components of relations.* ACM Transactions on Database Systems (TODS), vol. 2, no. 4, pages 317–325, 1977. (Cited on pages 69 and 70.)

[Rivest 1978] Ronald L Rivest, Adi Shamir and Len Adleman. *A method for obtaining digital signatures and public-key cryptosystems.* Communications of the ACM, vol. 21, no. 2, pages 120–126, 1978. (Cited on page 4.)

[Rizvi 2004] Shariq Rizvi, Alberto Mendelzon, Sundararajarao Sudarshan and Prasan Roy. *Extending query rewriting techniques for fine-grained access control.* In Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pages 551–562. ACM, 2004. (Cited on pages 24 and 25.)

[Rosenthal 2000] Arnon Rosenthal and Edward Sciore. *View security as the basis for data warehouse security.* In DMDW, page 8, 2000. (Cited on page 24.)

[Samarati 1998] Pierangela Samarati and Latanya Sweeney. *Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression.* Rapport technique, Technical report, SRI International, 1998. (Cited on pages 28 and 35.)

[Samarati 2001a] Pierangela Samarati. *Protecting respondents identities in microdata release.* Knowledge and Data Engineering, IEEE Transactions on, vol. 13, no. 6, pages 1010–1027, 2001. (Cited on page 35.)

[Samarati 2001b] Pierangela Samarati and Sabrina Capitani de Vimercati. *Access control: Policies, models, and mechanisms.* In Foundations of Security Analysis and Design, pages 137–196. Springer, 2001. (Cited on page 19.)

[Sandhu 1996a] Ravi Sandhu and Pierangela Samarati. *Authentication, access control, and audit.* ACM Computing Surveys (CSUR), vol. 28, no. 1, pages 241–243, 1996. (Cited on page 14.)

[Sandhu 1996b] Ravi S Sandhu, Edward J Coynek, Hal L Feinsteink and Charles E Youmank. *Role-Based Access Control Models yz.* IEEE computer, vol. 29, no. 2, pages 38–47, 1996. (Cited on page 22.)

[Sandhu 1998] Ravi S Sandhu. *Role-based access control*. Advances in computers, vol. 46, pages 237–286, 1998. (Cited on page 22.)

[Sandhu 2000] Ravi Sandhu, David Ferraiolo and Richard Kuhn. *The NIST model for role-based access control: towards a unified standard*. In ACM workshop on Role-based access control, volume 2000, 2000. (Cited on page 22.)

[Schlörer 1981] Jan Schlörer. *Security of statistical databases: multidimensional transformation*. ACM Transactions on Database Systems (TODS), vol. 6, no. 1, pages 95–112, 1981. (Cited on pages 28 and 29.)

[Shafiq 2005] Basit Shafiq, James BD Joshi, Elisa Bertino and Arif Ghafoor. *Secure interoperation in a multidomain environment employing RBAC policies*. Knowledge and Data Engineering, IEEE Transactions on, vol. 17, no. 11, pages 1557–1577, 2005. (Cited on page 40.)

[Shehab 2005] Mohamed Shehab, Elisa Bertino and Arif Ghafoor. *Secure collaboration in mediator-free environments*. In Proceedings of the 12th ACM conference on Computer and communications security, pages 58–67. ACM, 2005. (Cited on page 40.)

[Sheth 1990] Amit P Sheth and James A Larson. *Federated database systems for managing distributed, heterogeneous, and autonomous databases*. ACM Computing Surveys (CSUR), vol. 22, no. 3, pages 183–236, 1990. (Cited on pages 3 and 16.)

[Silberschatz 1996] Avi Silberschatz, Mike Stonebraker and Jeff Ullman. *Database research: achievements and opportunities into the 1st century*. ACM Sigmod Record, vol. 25, no. 1, pages 52–63, 1996. (Cited on page 3.)

[Spitzner 2003] Lance Spitzner. *Honeypots: Catching the insider threat*. In Computer Security Applications Conference, 2003. Proceedings. 19th Annual, pages 170–179. IEEE, 2003. (Cited on page 5.)

[Su 1987] Tzong-An Su and Gultekin Ozsoyoglu. *Data Dependencies and Inference Control*. In IEEE Symposium on Security and Privacy, page 202. IEEE Computer Society Press, 1987. (Cited on pages 5, 28, 30, 31, 33 and 35.)

[Sweeney 2002a] Latanya Sweeney. *Achieving k-anonymity privacy protection using generalization and suppression.* International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10, no. 05, pages 571–588, 2002. (Cited on pages 28, 35 and 36.)

[Sweeney 2002b] Latanya Sweeney. *k-anonymity: A model for protecting privacy.* International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10, no. 05, pages 557–570, 2002. (Cited on page 35.)

[Thuraisingham 1987] MB Thuraisingham. *Security checking in relational database management systems augmented with inference engines.* Computers & Security, vol. 6, no. 6, pages 479–492, 1987. (Cited on pages 5, 28, 33 and 35.)

[Thuraisingham 1992] Bhavani Thuraisingham and Harvey H Rubinovitz. *Multilevel security issues in distributed database management systems III.* Computers & Security, vol. 11, no. 7, pages 661–674, 1992. (Cited on page 4.)

[Traub 1984] Joseph F Traub, Yechiam Yemini and H Woźniakowski. *The statistical security of a statistical database.* ACM Transactions on Database Systems (TODS), vol. 9, no. 4, pages 672–679, 1984. (Cited on page 29.)

[Truta 2006] Traian Marius Truta and Bindu Vinay. *Privacy Protection: p-Sensitive k-Anonymity Property.* In ICDE Workshops, page 94, 2006. (Cited on page 36.)

[Verykios 2004] Vassilios S Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin and Yannis Theodoridis. *State-of-the-art in privacy preserving data mining.* ACM Sigmod Record, vol. 33, no. 1, pages 50–57, 2004. (Cited on page 35.)

[Wang 2004] Lingyu Wang, Duminda Wijesekera and Sushil Jajodia. *A logic-based framework for attribute based access control.* In Proceedings of the 2004 ACM workshop on Formal methods in security engineering, pages 45–55. ACM, 2004. (Cited on pages 19, 23 and 60.)

[Warner 1965] Stanley L Warner. *Randomized response: A survey technique for eliminating evasive answer bias.* Journal of the American Statistical Association, vol. 60, no. 309, pages 63–69, 1965. (Cited on page 29.)

[Wiederhold 1992] Gio Wiederhold. *Mediators in the architecture of future information systems.* Computer, vol. 25, no. 3, pages 38–49, 1992. (Cited on page 16.)

[Yuan 2005] Eric Yuan and Jin Tong. *Attributed based access control (ABAC) for web services.* In Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on. IEEE, 2005. (Cited on page 23.)

[Zhou 1995] Gang Zhou, Richard Hull, Roger King and Jean-Claude Franchitti. *Using Object Matching and Materialization to Integrate Heterogeneous Databases.* In CoopIS, pages 4–18, 1995. (Cited on page 16.)

[Zhuge 1995] Yue Zhuge, Hector Garcia-Molina, Joachim Hammer and Jennifer Widom. *View maintenance in a warehousing environment.* ACM SIGMOD Record, vol. 24, no. 2, pages 316–327, 1995. (Cited on page 16.)