Access Control Policies Verification Over Distributed Queries

Adel Jebali

UNIVERSITÉ DE TUNIS EL MANAR
FACULTÉ DES SCIENCES MATHÉMATIQUES, PHYSIQUES ET NATURELLLES DE TUNIS
École Doctorale de Mathématiques, Informatique, Sciences et Technologies de la Matière

PHD THESIS

for the degree of

Doctor of Philosophy in Computer Science

Defended by
Adel Jebali

Access Control Policies Verification Over Distributed Queries

Publicly defended on December, 30th ,2021

Committee :

President:

Mr. Samir Ben Ahmed — Professor FST, Université Tunis El manar

Reviewers:

Mr. Sami Faiz — Professor ISAMM, Mannouba Universirty

Mr. Belhassen Zouari — Professor Sup'Com, Carthage University

Examiner:

Mr. Kamel KAROUI — Associate Professor INSAT, Carthage University

Advisor:

Mr. Abderrazak Jemai — Professor INSAT, Carthage Universirty

Realized within SERCOM laboratory, Polytechnic school of Tunisia

Dedication

I dedicate this thesis to

My beloved parents and siblings

My sweetheart Manel

My precious friends

And to anyone who believed in me and contributed directly or indirectly to the accomplishment of this Thesis

Acknowledgments

First of all, a very special gratitude goes to my supervisor Prof. Abderrazak Jemai who had supported me for the last four years through my ups and downs and who had sacrificed time and energy to elaborate this Thesis. I am also grateful to my co-supervisors Prof. Richard Chbeir and Dr. Salma Sassi for their dedication to achieve this thesis' goals. Special thanks to Prof. Belhassen Zouari and Prof. Sami Faiez for accepting to review my thesis manuscript and for their kind efforts to enhance its quality. My thanks go as well to Prof. Samir Ben Ahmed and Dr. Kamel Karoui for having accepted to act as examiner of my thesis. Special thanks are also directed to Dr. Mokhtar Sellami for his valuable advices.

I would also like to thank my father, mother, brother, sisters and all my family. They have always been supporting and encouraging me with their best wishes. Their faith in me allowed me to be as ambitious as I wanted and helped me a lot through the past years.

And finally, last but by no means least, I am very grateful to my fiancee Manel Sansa who had always supported my decisions and had encouraged me. I am also grateful to all my friends who have supported me along the way.

Thank you all for your encouragement!

تجري الرياح كما تجري سفينتنا

نحن الرياح و نحن البحر و السفن

إن الذي يرتجي شيئاً بهمّتهِ

يلقاهُ لو حاربَتْهُ الانسُ والجنُّ

فاقصد إلى قمم الاشياءِ تدركها

تجري الرياح كما رادت لها السفنُ

عبد الرزاق عبد الواحد

Abstract

In this thesis, we address the problem of data outsourcing in presence of access control policies. Due to the emergence of Database-as-a-Service paradigm, secure data outsourcing has become one of the crucial challenges which strongly imposes itself. Indeed, data owners place their data among Cloud Storage Service Providers (CSSP) in order to increase flexibility, optimize storage, enhance data manipulation and decrease processing time. In spite of that, access control is considered as a major barrier to cloud computing and data outsourcing arrangements. Hence, the central challenge identified in this context is: *How access control policies of data owner are preserved when data is moved to the cloud?*

From a security perspective, preserving access control policies means that if an access was prohibited initially by the owner's access control policies, it should be also prohibited when data is externalized to Cloud Storage Service Providers. Also, the policy in the Cloud Storage Service Providers level should protect data against indirect access via inference channels. This inference channel is derived from the combination of legitimate answers received from the system with semantic constraints. Furthermore, to maintain data utility, an optimal data placement decision should be considered when this latter is moved to the cloud.

In this manuscript, on the basis of vertical partitioning, we propose a graph-based approach to preserve owner's access control policies efficiently when data is externalized to the Cloud Storage Service Providers. To do that, our proposed approach runs through the following steps: Firstly, it relies on semantic relatedness measure between users roles and schema attributes to derive an optimal vertical partitioning. Optimal partitioning is the minimization of the number of distributed queries issued from a user role in order to provide higher performance and speedup. Secondly, by reasoning about functional dependencies as source of

inference, we propose a set of algorithms to detect inference leakage and control them. Thirdly, on the basis of hypergraph theory, we propose a hypergraph partitioning algorithm to compute the set of secure partitions stored in Cloud Storage Service Providers level. The proposed partitioning algorithm takes advantage of the distributed system to enforce access control policies. Then, by considering access control rules as a set of queries to be revoked, we infer the implicit combinations of queries that could lead to the violation of owner's access control policies. Finally, we propose a monitoring module based on Role-Based Access Control and History-Based Access Control to monitor users queries at run time and block suspicious ones.

Keywords: Access control, Inference control, Data dependencies, Distributed databases, Semantic relatedness, Hypergraph theory, security and privacy.

Resumé

Dans cette thèse, nous abordons le problème de l'externalisation des données en présence de politiques de contrôle d'accès. En raison de l'émergence du paradigme Database-as-a-Service, l'externalisation des données est devenue l'un des défis cruciaux qui s'impose fortement. En effet, les propriétaires de données placent leurs données auprès des fournisseurs de services de stockage cloud afin d'augmenter la flexibilité, d'optimiser le stockage, d'améliorer la manipulation des données et de réduire le temps de traitement. De ce fait, le contrôle d'accès est considéré comme un problème très potentiel par rapport aux accords de cloud computing et d'externalisation des données. Par conséquent, le défi majeur identifié dans ce contexte est: *Comment les politiques de contrôle d'accès du propriétaire des données sont préservées lorsque les données sont déplacées vers le cloud?*

Du point de vue sécurité des données, la préservation des politiques de contrôle d'accès signifie que si un accès a été initialement interdit par les politiques de contrôle d'accès du propriétaire, il devrait également être interdit lorsque les données sont externalisées vers les fournisseurs de services Cloud. En outre, la politique de sécurité au niveau des fournisseurs de services cloud doit protéger les données contre l'accès indirect à travers les fuites d'inférence. Une fuite d'inférence est déduite par la combinaison de réponses légitimes reçues du système avec des contraintes sémantiques. De plus, pour maintenir l'utilité des données, une décision de placement optimale des données doit être envisagée lorsque ces dernières sont déplacées vers le cloud.

Dans ce manuscrit, en s'appuyant sur le partitionnement vertical, nous proposons une approche basée sur la théorie des graphes pour préserver efficacement

les politiques de contrôle d'accès du propriétaire lorsque les données sont externalisées vers les fournisseurs de services cloud. Pour ce faire, notre approche proposée comporte les étapes suivantes: Premièrement, elle s'appuie sur la mesure de corrélation sémantique entre les rôles des utilisateurs et les attributs de schéma de la base de données pour dériver un partitionnement vertical optimal. Le partitionnement optimal consiste à minimiser le nombre de requêtes distribuées émises à partir d'un rôle utilisateur afin de fournir des performances élevées et accélérer le temps du traitement. Deuxièmement, en raisonnant sur les dépendances fonctionnelles comme source d'inférence, nous proposons un ensemble d'algorithmes pour détecter les fuites d'inférence et les contrôler. Troisièmement, en se basant sur la théorie de l'hypergraphe, nous proposons un algorithme de partitionnement hypergraphique pour calculer l'ensemble des partitions sécurisées stockées au niveau des fournisseurs de services cloud. L'algorithme de partitionnement proposé tire parti du système distribué pour enforcer les politiques de contrôle d'accès. Par la suite, en considérant les règles de contrôle d'accès comme un ensemble de requêtes à révoquer, nous déduisons les combinaisons implicites de requêtes qui pourraient conduire à la violation des politiques de contrôle d'accès du propriétaire. Enfin, nous proposons un module de surveillance basé sur le contrôle d'accès à base des rôles et le contrôle d'accès à base de l'historique pour surveiller les requêtes des utilisateurs au moment de leurs exécution et bloquer les requêtes suspectes.

Mots-clés: Contrôle d'accès, Contrôle d'inférence, Dépendances de données, Bases de données distribuées, Corrélation sémantique, théorie de l'hypergraphe, sécurité et confidentialité.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Problem Statement . . . 5
- 1.3 Objectives . . . 6
- 1.4 Contributions . . . 8
  - 1.4.1 Constraints Generation . . . 8
  - 1.4.2 Schema Partitioning . . . 9
  - 1.4.3 Query Evaluation Model . . . 9
- 1.5 Thesis Outline . . . 9
- 1.6 Publication . . . 10

2 State of the Art **12**
- 2.1 Introduction . . . 13
- 2.2 Data Outsourcing . . . 14
- 2.3 Security of Database Systems . . . 17
  - 2.3.1 Access Control Models . . . 17
    - 2.3.1.1 Discretionary Access Control Model . . . 17
    - 2.3.1.2 Mandatory Access Control Model . . . 18
    - 2.3.1.3 Role-Based Access Control Model . . . 18
    - 2.3.1.4 Attribute-Based Access Control Model . . . 19
    - 2.3.1.5 History-Based Access Control Model . . . 20
    - 2.3.1.6 Advanced Access Control Models . . . 20
  - 2.3.2 Access Control for Relational Database Systems . . . 21
    - 2.3.2.1 The System R Access Control Model . . . 22
    - 2.3.2.2 Oracle Virtual Private Database . . . 22
    - 2.3.2.3 Oracle Label . . . 23
    - 2.3.2.4 Content-based Access Control Model with Authorization View . . . 23
  - 2.3.3 Access Control Verification in Distributed Environment . . . 25

2.3.3.1 Distributed Access Control Policies 25
2.3.3.2 Role-Based Access Control for Distributed Database Systems 25
2.3.3.3 SQL for Distributed Database Security 26
2.3.3.4 Access Control Through Confidentiality Constraints 26
2.3.3.5 Auditing a Distributed Database System 27
2.3.3.6 Access Control Enforcement in Data Integration Systems 27
2.4 Data Protection from Insider Threat 30
2.4.1 Access Control vs Inference Control 32
2.4.2 Inference Control in Centralized Database Systems 33
2.4.2.1 Inference Attacks 34
2.4.2.2 Inference Prevention Methods 35
2.4.2.3 Discussion of the Inference Prevention Methods 37
2.4.3 Inference Control in Distributed Environment 38
2.4.3.1 Inference Control in Distributed Data Sources . 38
2.4.3.2 Inference Control in Data Integration Systems . 40
2.5 Data Outsourcing in Presence of Access Control Policies 44
2.5.1 Secure Data Outsourcing with Non-Communicating Servers 45
2.5.2 Secure Data Outsourcing: The Case of Communicating Servers 47
2.5.3 Data Outsourcing and the Inference Problem 49
2.6 Optimal Database Schema Partitioning 51
2.7 Discussion 52
2.7.1 Choice of the Access Control Model 52
2.7.2 Discussion of the Proposed Approaches 53
2.8 Conclusion 54

3 Preliminaries and Problem Statement **56**
3.1 Introduction 57
3.2 Preliminaries and Basic Concepts 57
3.2.1 Definitions Related to Access Control and Inference Problem 57
3.2.2 Definitions Related to Graph Theory 58
3.2.3 Definitions Related to User Role 58
3.3 Problem Statement 59
3.4 Motivating Scenario 60
3.5 Discussion of Studied Problem with their Relevant Fields 62
3.5.1 Data Outsourcing 62
3.5.2 Access Control Model 63
3.5.3 Inference Control 63

3.5.4 Ontology-Based Vertical Database Schema Partitioning . . 64
3.6 Overview of the Proposed Approach 64
3.7 Conclusion 66

4 Constraints Generation 67
4.1 Introduction 68
4.2 Visibility Constraints Generation Based on Semantic Relatedness . 68
4.3 Inference Control 74
4.3.1 Step 1: Building the Functional Dependency Graph G(V,E) 74
4.3.2 Step 2: Generating Join Chain Set 74
4.3.3 Step 3 : Detecting relaxed_cut 78
4.3.4 Step 4 : Constraints-based inference control generation . . 79
4.4 Conclusion 81

5 Schema Partitioning and Query Evaluation Model 83
5.1 Introduction 84
5.2 Schema Partitioning 84
5.2.1 Hypergraphs and constraint satisfaction problems 85
5.2.2 Computing K-balanced partitions 87
5.3 Query Evaluation Model 89
5.3.1 Violating Transactions Detection 90
5.3.2 Query lock 92
5.4 Conclusion 93

6 Experimental Study 95
6.1 Introduction 96
6.2 Experimental Design 96
6.3 Evaluation 98
6.3.1 Functional Dependencies Impact on Constraints-Based Inference Control Generation 98
6.3.2 Impact of the Number of Attributes on the Partitioning Algorithm 99
6.3.3 Impact of the Variation of Confidentiality Constraints and Visibility Constraints on the Partitioning Algorithm 100
6.3.4 Comparison of Query Execution Time Between MySQL and SparkSQL 101
6.3.5 Time Required to Lock a Suspicious Query 102
6.4 Complexity Study 103
6.5 Conclusion 105

Conclusion 106

Appendix **110**

List of Figures

1.1 Secure data outsourcing 6

2.1 Literature review process 14
2.2 Cloud computing scenario 15
2.3 Full data outsourcing 16
2.4 Keep a few 16
2.5 Bypass access control with inference channels 31
2.6 Secure Data Integration System 41

3.1 Hospital-db-schema 60
3.2 Overview of the studied problem with their relevant fields 62
3.3 The proposed methodology to generate secure partitions and lock suspicious queries 65

4.1 An example of a partitioning w.r.t visibility constraints v_1, v_2 and v_3 72
4.2 The functional dependency graph 76

5.1 Hypergraph representation of the partitioning problem 86
5.2 A refined partitioning of the Hospital-db schema w.r.t C, V, $BC = 2$ and $K = 4$ 89
5.3 A query evaluation strategy augmented with a monitoring module 91

6.1 Deployment of the proposed approach on a cloud service 98
6.2 Impact of functional dependencies on the required timings to generate the constraint-based inference control set 98
6.3 Impact of the number of attributes on the partitioning algorithm . 99
6.4 Computational time varying the number of confidentiality constraints and visibility constraints 100

6.5 Comparison of query execution time between MySQL and Spark-SQL 101
6.6 Time required to lock a suspicious query 102

List of Tables

2.1 Keywords used in review search 13
2.2 Access control vs inference control 32
2.3 A comparison between existing works and the proposed model . . 54

5.1 Attribute to vertex mapping 86

6.1 number of confidentiality constraints C and visibility constraints V for each run 101

List of Algorithms

1 Generation of visibility constraints 71
2 Building Functional Dependency Graph G(V,E) 75
3 Join Chain Detection 77
4 Detecting Relaxed_cut 80
5 Constraints-based inference control generation 81
6 Computing K-balanced partitions 88
7 VT_Track 92
8 Query Lock 94

Chapter 1

Introduction

1.1 Motivation

In light of the fast development of new information technology (e.g Cloud Computing, Big Data, Internet of Things and so on) during the last decades, data engineering systems have been developed and deployed in several sectors. Those systems design, manage and optimize the flow of data coming from day-to-day transactions (healthcare systems, social insurance systems, social networks), or an on-line analytical processing (OLAP systems) to store and analyze everything -vital or not- to an organization in order to enhance the decision support. Among these technologies, database system is considered as an elegant and robust part of data engineering systems with its own identity. This system enables the creation, maintenance, and use of large amounts of data for modern data application systems as they are used as a data repository behind an interface (78).

Historically, Early database management systems were based on the network and hierarchical models. The logical organization of data for those two models is based respectively on graph and network. However, these representations closely mirror the physical storage of the data (2). Furthermore, those models focus primarily on navigation through the physically stored data. In the 1970s, Codd's relational model revolutionized the field. This proposal succeeded the hierarchical models and networks, with the aim to define a model that is easy to understand, based on logic mathematics and set theory. The basic idea of this model is simple:

put the data ”flat” in tables, so that every value appearing in the boxes of the table is atomic. This simplification is the main explanation for the performance of the relational model, because it avoids recursive processing (cross complex graphs and trees). The relational databases transactions respect the *ACID* properties (Atomicity, Consistency, Isolation, and Durability). These properties are used for maintaining the integrity of database during transaction processing where:

- Atomicity: A transaction is a single unit of operation.
- Consistency: The content must be consistent at the start and at the end of a transaction (e.g. with respect to integrity constraints).
- Isolation: Transaction should be executed in isolation from other transactions
- Durability: Once the transaction is successfully completed, the state of the database is permanent.

In the last decade, due to the huge volume and heterogeneity of data such as documents, e-mail, multimedia and social media, a growing number of companies have adopted various types of non-relational databases, commonly referred to as NoSQL databases (77). NOSQL encompasses a wide range of technologies and architectures, in order to solve the problems of performance that relational databases were not designed to cope. NoSQL have adopted other properties grouped by the acronym BASE: Basically Available, Soft-state and Eventually consistent.

Recently, with the increasing volume and variety of data collected from diverse sources, costs of in-house data storage and highly significant data processing cost. The database community has been devoted a huge amount of research efforts to provide new paradigms to manage voluminous and heterogeneous data. Each research direction has been designed to meet the requirements of new scenarios and applications. The main directions that have been tackled are:

- Data integration (70): This architecture aims at providing a unique entry point to distributed and heterogeneous data sources.

- Data exchange (48): It takes data structured under a source schema and creating an instance of a target schema that reflects the source data as accurately as possible.
- Data fusion (24): It aims at resolving conflicts (e.g., semantic heterogeneity, duplicate detection) arising in data integration scenarios.
- Data outsourcing (84): It distributes data among Cloud Storage Service Providers in order to increase flexibility, optimize storage, enhance data manipulation and decrease processing time.

Nonetheless, protecting database systems is a challenging task because security is relatively a diversified concept. This wide use of such database systems involving that security breaches and unauthorized disclosures threat those systems and increase the exposure of data and made security more difficult (14). It has been shown conclusively that threats to databases security can cause the loss or degradation of some or all of the following (7):

- Loss of integrity.
- Loss of confidentiality.
- Loss of availability.

There is a consensus among security researchers that ensuring the security of a database system is not straightforward. In fact, several attempts have been made to do this but no one of them can be considered as the universal solution. This due to the fact that database security involves several security mechanisms e.g. access control mechanisms, inference control, privacy preserving methods and so on. Several studies investigating database security have been carried out since 1970 (93). Based on the survey presented in (93), we present in the following the impact of database systems' evolution on the security of such systems since 1970.

- Around 1970s : Development of The system R and INGRES.
- Around the late 1980s : Emergence of multi-level secure relational database systems including seaview and lock dataview.

- Around the late 1980s : Investigation of the inference problem around MSD.
- Early 1990s : Multi-level secure distributed database and secure object database as well as security for heterogeneous database systems.
- In the late 1990s : Investigation of security for e-commerce systems and secure web database systems were investigated.
- Early 2000s : Investigation of privacy preserving data mining.
- Mid 2000s : Development of privacy models : K-Anonymity, L-Diversity and Differential privacy. Development of data mining tools to handle intrusion detection and malware analysis.
- Late 2000 and early 2010 : Securing of the emerging systems such as semantic web, web services and social media systems.
- 2010 until now : Examination of data security in cloud computing, big data security systems and privacy violation arising due big data analytic.

In this manuscript, we address the security challenges that may arise in data outsourcing scenarios. In fact, we focus on access control as security mechanism to guarantee *CIA (Confidentiality, Integrity, Availability)* properties. Indeed, access control is considered as a major barrier to cloud computing and data outsourcing arrangements. Access control policies of the data owner must be preserved when data is moved to the cloud. Preserving access control policies means that a if an access was prohibited initially by the owner's access control policies, it should be also prohibited when data is externalized to Cloud Storage Service Providers. Also, the policy in the Cloud Storage Service Providers level should protect data against indirect access via inference channels by which one can infer sensitive data from non-sensitive ones. Furthermore, an optimal data placement decision should be considered when this data is moved to the cloud while retaining access control policies of data owner.

1.2 Problem Statement

In this thesis, we will focus on the security challenge that mainly arises in data outsourcing scenarios. Our goal is to develop an approach to enforce the owner's access control policies when data is externalized to the cloud (Figure 1.1). In such type of architecture, data is distributed in separate partitions among CSSP. From a security perspective, access control is considered as a challenging task. The first challenge is to specify what access model to adopt in this situation. Next, we should guarantee the enforcement of the owner's access control policies when data is moved to CSSP level. Also, we should protect the system from insider threats through inference leakages. Furthermore, the distribution of the database among CSSP should be done while taking some optimality criteria into consideration. In this context, we are interested in the following issues:

- **Security and privacy concerns:** We consider owner's access control policies which enforces security and privacy regulations. The idea is to preserve those policies when the database is distributed among CSSP. In other words, the security policies in the CSSP level should comply with security policies of the data owner. Complying with owner's access control policies means that if an access is prohibited initially by the owner, it should be also prohibited in the CSSP (here we consider CSSP as untrusted parties).

- **Protection from insider threat:** It is important to point out that using only the access control may not be sufficient to ensure data protection against insider threat. This latter occurs when a malicious user combines the legitimate response received from the system with metadata to produce an inference channel. So, it is crucial and vital to ensure whether a user, even though is authorized by the access control mechanism, is trying to derive an inference channel.

- **Optimal distribution:** From optimality perspective, it would be preferable to establish an optimal data placement decision in the CSSP level. We believe that this will provide higher performance and speed-up when data is processed in the cloud.

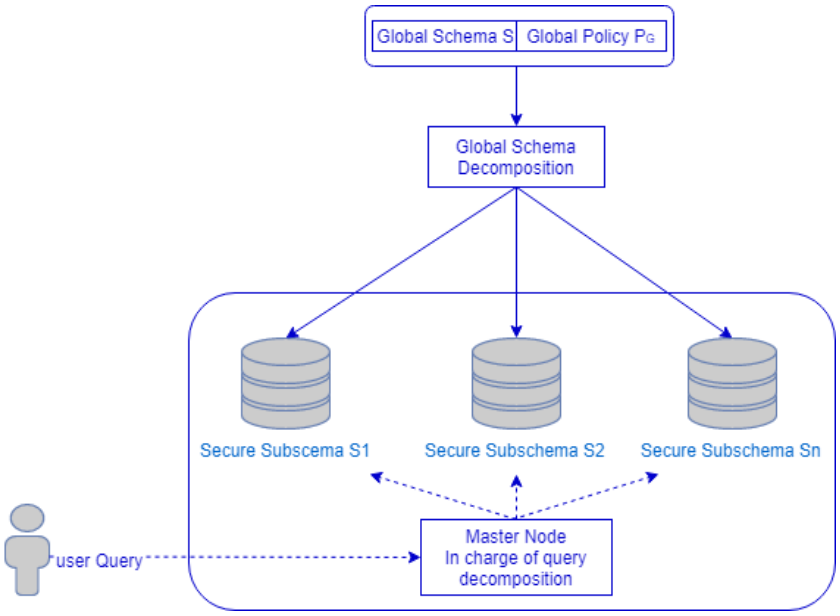

Figure 1.1: Secure data outsourcing

1.3 Objectives

In order to be in a position to fully respond to the security requirements arising in data outsourcing, our primary goal is to develop a methodology, together with a set of algorithms that can help the database administrator to design a secure and optimal distributed database schema in the cloud. The proposed approach allows to preserve owners' access control policies when the distributed database is deployed in CSSP level. Also, it helps detect and block insider threat by reasoning about semantic constraints as source of inference. As well, the proposed approach relies to an optimal data placement decision to improve throughput and decrease the degradation of the system.

In order to reach this objective, we have performed a deep and wide-ranging literature review to investigate different research fields that could provide the fundamental concepts to tackle our research project. Then, we have conducted an in-depth analysis of the existing approaches in each field to efficiently extract the correlation between those approaches. Hence, we have formally defined the issues to be addressed in order to design the proposed approach. This approach

runs through three main phases. Each phase is built on top of the previous one.

Most of the proposed approaches have focused on data management and query processing in data outsourcing scenarios. However, less attention has been devoted to security and privacy concerns. Thence, information security is still considered as a challenging task to be investigated in such system. We have identified four relevant fields in relation with our research problem. these fields are:

- Data outsourcing (84): Different ways of data outsourcing (Full outsourcing (3) or Keep a few (35)), induce different access control issues. In this manuscript, we have focused on security and privacy concerns that happened when data is fully externalized to CSSP.

- Access Control (16): Protecting data from direct access in the CSSP level is the first step toward preserving owner's access control policies. Therefrom, we have reviewed different access control models to protect database systems from direct access and we have designated the most relevant model in relation with our problem.

- Inference Control (49): It refers to the ability of a malicious user to exploit the correlation between non sensitive information to derive sensitive information. Typically, this is the most important field in our approach since this type of security breaches bypass access control mechanism without leaving a trace to the database administrator. In our work, we will specify how to capture inference leakage and how to control them.

- Data locality (38): In this field, we will study how to maximize data locality to increase system throughput. Existing approaches did not treat the case of maximizing data locality when the query workload. [1] is not available or the database schema is not loaded with data. Therefore, we propose a workload-independent approach to maximize data locality in CSSP level.

[1] Information regarding queries, their frequencies, involved attributes, arrival patterns, and so on

1.4 Contributions

In this section, we describe the different phases of the proposed methodology (66). Our approach relies on vertical partitioning (In this manuscript, we are interested to the relational model), it aims to produce a set of secure sub-schemes, each sub-schema represents a partition and each partition is stored exactly in one server in the CSSP. In addition, it introduces a secure distributed query evaluation strategy to efficiently request data from distributed partitions while retaining access control policies. The proposed approach is centered around three phases as follows:

1.4.1 Constraints Generation

This phase aims to generate two types of constraints that in addition to the confidentiality constraints, will guide the process of vertical schema partitioning. This will be done through the following steps:

- *Visibility constraints generation*: These constraints will be enforced as *soft constraints* in the partitioning phase and their severity is less than confidentiality constraints. To generate them, we perform a semantic analysis of the relational schema in order to measure semantic relatedness between attributes and users roles. These constraints will be preserved (stay visible) when the relational schema is fragmented. In other terms, we aim to maximize intra-dependency between attributes that seem to be frequently accessed by the same role while minimizing the inter-dependency between attributes in separate partitions.

- *Constraints-based inference control generation*: These constraints are enforced (like confidentiality constraints) as *hard constraints*. In this step, we resort to the method proposed in (99) to build a functional dependencies graph and generate a set of join chains. Then, we use a relaxed technique to cut the join chain only at a single point in order to minimize dependencies loss. We mean by cutting a join chain at a single point, the enforcement of the attributes in the *LHS* and *RHS* of the functional dependency representing

the cut point as a confidentiality constraint. By consequence, we guarantee that the join chain will be broken.

1.4.2 Schema Partitioning

In this phase, we resort to hypergraph theory to represent the partitioning problem as a hypergraph constraint satisfaction problem. Then, we reformulate the problem as a multi-objective function F to be optimized. Therefore, we propose a greedy algorithm to partition the constrained hypergraph into k partitions while minimizing the multi-objective function F.

1.4.3 Query Evaluation Model

In this phase, we propose a monitor module to mediate every query issued from users against data stored in distributed partitions. The monitor module contains two mechanisms : a Role-Based Access Control mechanism and History-Based Access Control mechanism. The first mechanism checks the user role who issued the query and if this latter is not granted to execute distributed queries, then his query will be forwarded directly to the desired partition. Otherwise, the query is forwarded to the History Access Control mechanism which takes as input a set of violating transactions to be prohibited and checks if the cumulative of user past queries and current query could complete a violating transaction. If it is the case, the query is revoked.

1.5 Thesis Outline

The structure of this thesis follows the sequence of reported contributions and it is organized as follows:

- In Chapter 2, we will investigate the state of the art. We start by giving an overview about important architectures in data outsourcing. Next, we discuss different access control models in relation with database security, we highlight research efforts related to secure data outsourcing and the inference problem. We conclude this chapter with a discussion of these fields.

- In Chapter 3, we will introduce a set of relevant concepts and preliminaries. We will also introduce the studied problem, describe the motivating scenario and give an overview about the proposed methodology.
- In Chapter 4, we will introduce the first phase of our three-step approach. We will start by generating the set of visibility constraints and then we will introduce our technique to detect inference channels caused by functional dependencies and how to control them.
- In Chapter 5, we will present the second and the third phases of our approach. We will introduce the schema partitioning phase and how to compute secure and optimal partitions. Next, We will describe our query evaluation model and how to lock suspicious queries.
- In Chapter 6, we will provide an experimental evaluation of our proposed approach and an analysis of the proposed algorithms.
- In Chapter 7, we will summarize our research contributions and the lessons learned while developing secure data outsourcing. Finally, we will highlight the limitations and we will propose potential future extension of our work.

1.6 Publication

The outcomes of this thesis are published in the hereafter list of publications:
International Journals (published)

- Jebali, A., Sassi, S., Jemai, A.,and Chbeir, R. Secure data outsourcing in presence of the inference problem: A graph-based approach. Journal of Parallel and Distributed Computing 160(2022), 1–15, **Q1, Impact factor:3.7**
- A. Jebali, S. Sassi, A. Jemai, Secure data outsourcing in presence of the inference problem: issues and directions, Journal of Information and Telecommunication (2020) 1-19, **Indexed Taylor & Francis.**

International Conferences (published)

- A. Jebali, A. Jemai, S. Sassi, A survey study on the inference problem in distributed environment (s)., in: SEKE, 2019, pp. 113-152, **Class B**
- A. Jebali, S. Sassi, A. Jemai, Inference control in distributed environment: A comparison study, in: International Conference on Risks and Security of Internet and Systems, Springer, 2019, pp. 69-83, **Class C**

Chapter 2

State of the Art

2.1 Introduction

The purpose of this chapter is to introduce the main related topics and research areas covered in this thesis investigation. The fundamental issue that we address is the verification of access control in data outsourcing. From the wide array of previous research, we examine in this chapter the most representative fields which directly impact the research of this dissertation, namely data outsourcing ,access control models, their adaptation to data outsourcing scenarios and protection from insider threats through inference channels. The last section of this chapter is dedicated to the discussion of the proposed works in each of those fields, to motivate our research challenges.

The selection process of the literature review is done as follows: Firstly, our advance keyword research was conducted on Google Scholar search engine with a time filter from 1 January 1970 to 31 December 2019. Table 2.1 lists the used keywords in different queries search in Google Scholar. The logical operator used between keywords during our search was the "AND" operator. begincenter

Table 2.1: Keywords used in review search

Keyword	**Number of viewed papers**
Access control, Data outsourcing	43
Cloud computing , Authorization policies	48
Database, inference leakage	33
Confidentiality constraints, Cloud database	41
Secure data integration	11
Big data, Distributed query processing	39
Privacy, data publishing	24

The methodology of reviewing adopted in this manuscript includes as shown in Figure. 2.1 three steps: *Input literature, processing steps and review output*.

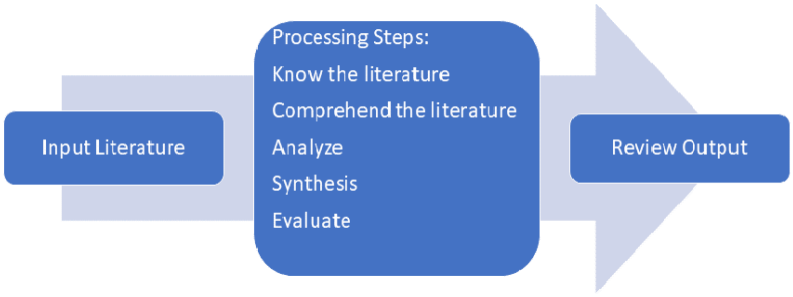

Figure 2.1: Literature review process

2.2 Data Outsourcing

Recently, the emergence of cloud computing technology has facilitated numerous configurable resources in which the data is stored and managed in a decentralized architecture. According to NIST (74) standard, cloud computing is defined as ”a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”. Cloud computing enables data manipulation and storage to a third-party and gives the ability to the data owner to access this data from anywhere at any time (Figure 2.2) (8).

It is from this perspective that modern database systems have been moved from a central repository for record-keeping of internal enterprise data to a distributed repository. This is due to the fact that in-house storage and processing of large collections of data has becoming very cost. Hence, this has given emergence to the Database-as-a–Service (DBaaS) paradigm. Such a service is considered attractive to data owners for two reasons (39):

- Economic concerns: The hardware and energy costs incurred by data owner are likely to be much lower when they are paying for a share of a service rather than running everything locally (in-house data processing).
- The costs adopted in a well-designed Database-as-a-Service will be proportional to actual usage (“pay-as-you-go”).

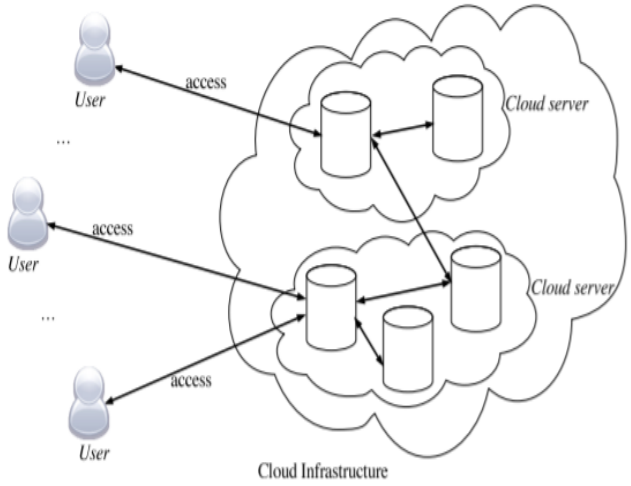

Figure 2.2: Cloud computing scenario

On the other hand, data outsourcing opens the door to possible security breaches to the data and introduces therefore new security and privacy concerns to be addressed. Cloud service providers are considered honest-but-curious: The database servers answer user queries correctly and do not manage stored data, but they attempt to analyze intelligently data and queries in order to learn as much information as possible from them. Two powerful techniques have been proposed to enforce access control in cloud databases: The first one is by exploiting vertical database fragmentation to keep some sensitive data separated from each other. The second one is by resorting to encryption to make single attribute invisible to unauthorized users. These two techniques can be implemented using the following approaches:

- **Full outsourcing** (3): The hole in-house database is moved to the cloud. It considers vertical database fragmentation to enforce confidentiality constraints with more than two attributes by keeping them separated from each other among distributed servers. Moreover, it resorts to encryption in order to hide confidentiality constraints with single attribute (Figure 2.3).

- **Keep a few** (35): This approach departs from encryption by involving owner side. The attributes to be encrypted are stored in plain text in the

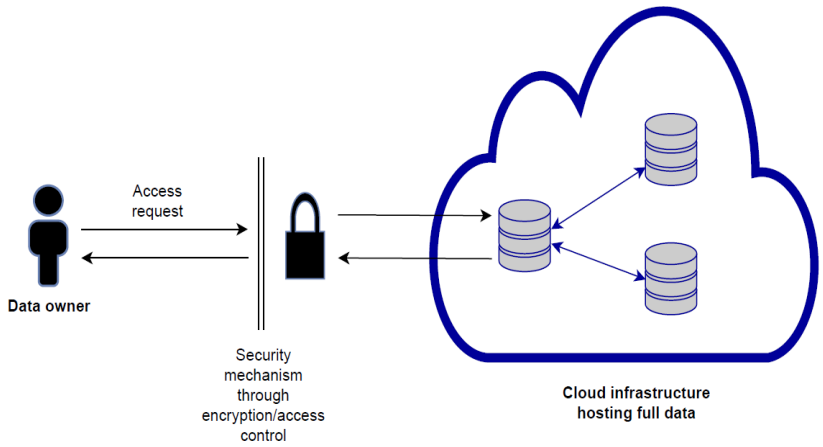

Figure 2.3: Full data outsourcing

owner side since this latter is considered as a trusted part. The rest of the database is distributed among servers while maintaining data confidentiality through vertical fragmentation (Figure 2.4).

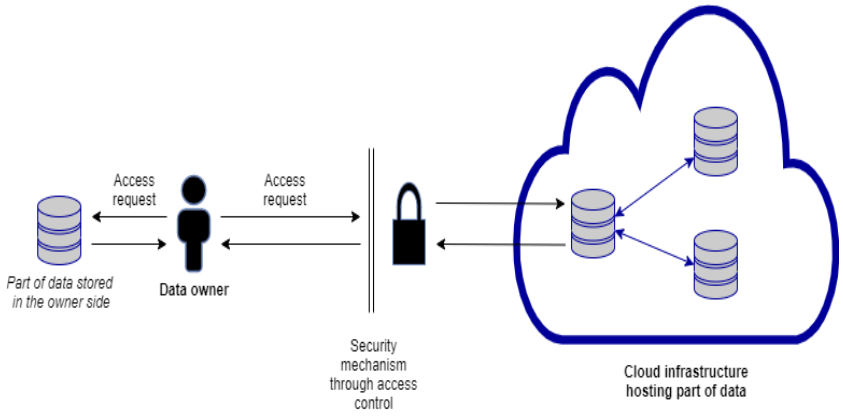

Figure 2.4: Keep a few

In this thesis, we are interested to security issues, particularly, we focus on access control challenges that arise when data is fully externalized to the cloud. Next, we discuss the existing access control models, their adaptation to data outsourcing scenarios and protection from insider threats through inference channels.

2.3 Security of Database Systems

Since 1970, a lot of research efforts have been devoted from the database security research community to develop access control mechanisms and guarantee CIA properties. Based on the surveys in (16) and (83), we discuss the most known models and approaches proposed for specifying and enforcing access control policies in database management systems. We investigate also how those models were adapted to access control verification in distributed environment.

2.3.1 Access Control Models

Access control consists in mediating every access request issued from the user against the system, and determining whether this access should be allowed or denied. Access control system is composed of three main components: *Access control policy, access control model and access control mechanism*. A policy defines a set of rules specifying whether an access request is allowed or denied. A policy is then formalized through an access control model enforced through an access control mechanism. Here, we describe the main approaches that have been proposed.

2.3.1.1 Discretionary Access Control Model

According to (83), Discretionary Access Control Model (DAC) is a model based on the identity of the subject, identities of object and permissions. The policy is presented as an authorization rule (S,O,A) where S is the subject, O is the object and A is the access right. Hence, it is called discretionary because of the delegation principle. Furthermore, DAC has been developed considerably based on the access matrix (it is considered as the most known framework for describing the discretionary access control). (87). DAC expands authorization where the condition can be associated with time, location or the content of object. Also, it supports the principle of user groups and object groups. Moreover, DAC is enriched with positive (closed policy) and negative authorization (open policy) and they can be also combined into one policy. This combination must satisfy incompleteness

and inconsistency (conflict resolution). In fact, conflict resolution can be resolved according to the following principle: Denial-take-precedence, most-specific-take-precedence, strong/weak, priority level,positional, grantor-dependent and time-dependent.

Nonetheless, it should be noted that DAC model ignores the users originate process that execute on behalf and accordingly submit request to the system. This aspect makes DAC vulnerable and can be bypassed by trojan horses embedded in programs (16).

2.3.1.2 Mandatory Access Control Model

Mandatory Access Control Model (MAC) enforces access control policies on the basis of regulations mandated by a central authority (83). The most common form of MAC is the multilevel security policy based on the classification of subject and object in the system. Furthermore, MAC is based on two principles: The Bell-LaPadula model and Integrity-based mandatory model. The former model protects only the confidentiality of the information; no control is enforced on its integrity, through the No-read-up and No-write-down principles. However, the second model prevents subjects from enforcing improper indirect modifications to object they cannot write through the No-read-down and No-write-up principles.

Despite its pedantism, MAC model is vulnerable to covert channel and aggregation problem (90) (29). Another limit for MAC, is the complication called Polyinstantiation. In such situation, subjects at different levels have different views on a relation, which is the view composed only of elements they are cleared to see (83), this may affect the integrity constraints (primary key constraint) of the database in the update and insert operation.

2.3.1.3 Role-Based Access Control Model

The conventional access control model which is based on the identity of users, and the mandatory access control model in which users have security clearances and objects have security classifications are not a good fit (83). Therefrom, Role-Based Access Control Model (RBAC) tries to fill in this gap by merging the

flexibility of explicit authorizations with additionally imposed organizational constraints. In such a model, a set privileges are grouped into a role, then a user is assigned to this role by activating his session. Furthermore, the policies regulate the access of users to the information on the basis of organizational activities and responsibility that users have in an organization. A role can be a user's job (eg.,manager) or it can be more specific matching a task that user need to perform (separation of duty by distinguishing between users roles and users process). Moreover, roles can be hierarchically organized to exploit the propagation of access control privileges along the hierarchy, a user may be allowed to simultaneously play more than one role and more users may simultaneously play the same role (52).

RBAC is a better suits to commercial environment, it reduces the administration costs through the role assignment. However, despite this features, RBAC quite weak. Indeed, it is a time consuming to build a model instance and it lacks flexibility to efficiently adapt to changing users, objects, and security policies (60).

2.3.1.4 Attribute-Based Access Control Model

Attribute-Based Access Control (ABAC) is based on the concept of constraint satisfaction (57). In this model, a user is represented by a set of features characterizing her/his profile, and a constraint is associated with a resource. The role of ABAC is to mediate every request executed by a user against an object in the system, and to verify if the profile of the user requesting access satisfies the object constraints. If so, the access is granted. Otherwise, the access is denied. ABAC is considered as being more flexible than RBAC since it could express different types of access control policies. It is referred as logic-based access control in some works (101), while in other works, it is referred as rule-based (6). The features of RBAC and ABAC are considered to be complimentary, this is what motivate researchers to show an increasing interest in combining RBAC and ABAC models like the work proposed in (60).

2.3.1.5 History-Based Access Control Model

In order to ensure maximal availability of data while guaranteeing the non disclosure of sensitive information, History-Based Access Control Model (HBAC) consists in monitoring the access requests and revokes those that could lead to the violation of access control policies (1). The main idea of HBAC module is the following: When a user launches an access request to the system HBAC computes the cumulative of user past access and current access. If the cumulative can complete a violation of an access control policy, then the access request is revoked. Otherwise this latter is allowed.

2.3.1.6 Advanced Access Control Models

In recent years, the security breaches' rate is increasing worldwide. This is due to the fact that traditional access control models could not keep up with the continuous improvement of bypass techniques. As a consequence, security researchers were interested in enhancing the existing access control models in order to mitigate security breaches. In this section, we will discuss some of these works.

In (72), the authors developed a Cyberspace-oriented access control model. The objective was to ensure security of users accessing sensitive objects in the internet via their own devices, with temporal and spatial limitations. The model deals with seven atomic operations (read, write, store, execute, publish, forward, and select) that cover most operations in cyberspace. Then, the authors reformulated a suite of security policies for each atomic operation based on the following criteria : Confidentiality classification, integrity classification, object-related attribute, confidentiality clearance, subject trustworthiness, subject-related attribute, security range and security risk. Although this access control model gave the policy for only read and write operations, it seemed to be suitable for cyberspace especially for social networks.

A Policy Machine Access Control model was developed in (51). This model is a redefinition of many access control models aiming to provide a unified framework to support a wide range of policies under a single framework. It is composed of a policy machine server containing the policy machine database, the

policy decision point,the event processing, and the policy administration point. Furthermore, the policy machine contains a policy machine client in charge of hosting the OS, the applications, the API and the policy enforcement point. The policy machine is composed of relations (divided in three types: assignment, prohibitions, obligations and administrative commands) and the reference mediation function responsible of enforcing the access state. Also, the policy machine considers inter-process communication by preventing illegal information flow. In this access control model, the authors also highlighted the capability of the policy machine to express different access control models (DAC, MAC, RBAC and Chinese Wall Policy), and its ability to preserve security policies. From a commercial perspective, policy machine looks like a suitable model for client and vendors. Indeed, policy machine is decoupled from application code and OS, small portion of the code need to be integrated with the application. This will reduce the amount of code that needs to be trusted. On the other hand, the clients look like to be satisfied with this model since it implements their precise policy requirements.

The authors in (83) introduced the access control model used in large scale networks where remotely located parties may know little about each others (e.g, world wide web). In such huge networks, traditional access control models do not hold anymore. So, the authors resorted to Certificate-based access control model which is based on the use of digital certificates or credential. In such case, every part (client or server) possesses its own digital certificates and the access control decision requires a negotiation between these parties. It is important to note that the server cannot simply return ”allow/deny” decision, but it should inform the requester with the information of what he should do to get access.

2.3.2 Access Control for Relational Database Systems

In the previous section, we have investigated different access control models existing in the literature. Those models can be adopted to heterogeneous data structure, such as relational databases, XML documents, object databases and so on. Since we are interested in the relational model, we will discuss how access control models are applied and extended for use to relational databases.

2.3.2.1 The System R Access Control Model

System R access control model was proposed by *IBM* for relational databases. It is based on the *GRANT/REVOKE* approach to delegate and revoke access rights from users. Objects to be protected under this system are tables and views. The possible access modes that subjects can exercise on tables corresponding to the known SQL operations (SELECT, INSERT, DELETE and UPDATE). System R pursues DAC model, this is what pushes it to follow the ownership approach. In other words, the user who created an object became the owner of that object, and he can execute all access modes on this object. Moreover, the owner has the right to delegate access rights to other users through GRANT option.

Numerous studies have been developed on the basis of System R model. For example, in (102), the authors introduced a trigger mechanism by resorting to specific access control mode allowing a subject to implement a trigger on a table. Similarly, the introduction of mechanisms for referential integrity, through the use of foreign keys, has required the introduction of a related access mode allowing a subject to refer a table from another table (16).

2.3.2.2 Oracle Virtual Private Database

Oracle virtual private database implemented a Fine-grained access control model where the access can be controlled at table, row and even cell level (16). Oracle VPD policy is implemented in two steps. In the first step, a function that defines the restrictions to be enforced is defined. The second step is the creation of actual policy that associates the function with a table or a view. Oracle VPD policy is enforced as the following : When a user executes a query against a table or a view, a where clause containing the Oracle VPD policy is then attached dynamically to the SQL statement issued by the user. It have been showed that Oracle VPD provides the following benefits :

- Scalability: Reusability of functions.
- Security: Oracle VPD is enforced at server-level.

- Simplicity: The policy is added once at the database objects level than to be embedded repeatedly in the applications using the same database objects.

2.3.2.3 Oracle Label

Oracle Label security is an access control product developed by Oracle (16). It supports a multilevel relational model with granularity of tuple level. Hence, in this model, different tuples in the same multilevel relation may have different access classes called labels. The tuple-level access control is based on the VPD mechanism. Oracle Label policy is enforced as the following: When a subject issues a query against the content of a tuple, the Oracle Label compares the tuple's label with the subject's label and privileges. Thereby, the access control in Labeled Oracle is based on these factors:

- Tuple's label to which user requests access.
- User's label requesting access.
- The authorization that a user possesses.

It is important to note that Oracle Label enforces mandatory access control with discretionary access control, based on authorizations users received on the data and fine-grained restrictions, based on the VPD mechanism.

2.3.2.4 Content-based Access Control Model with Authorization View

Content-Based Access Control Model with Authorization View, also called View-Based Access Control Model is considered as the most suitable model for relational database systems regarding its flexibility in specifying access control policies (62). Essentially, Content-Based Access Control requires that access control decisions rely on data contents (16). Declarative languages like SQL, facilitates the development of this model. According to (53), the most common mechanism to support Content-Based Access Control is based on views. The view defines a query called *view definition query* through the database relations. This latter can be seen as a dynamic window that specifies which part of data the user is authorized to access, such view is called authorization view. An authorization view

can be a traditional relational view or a parameterized view. A parameterized authorization view is an SQL view definition which makes use of parameters like user-id, time, user-location and so on (81). In the following example, we want the user Bob to access only employees whose salary is lower than 10000 and their job is programmer, the authorization view to be defined is:

$CREATE\ AUTHORIZATION\ VIEW\ Vemp\ AS\ SELECT\ \star\ FROM\ Employees\ WHERE\ Salary\ <\ 1000\ and\ Job\ =\ programmer$;

Content-Based Access Control Model with Authorization View can enforce fine-grained access control since the portion of data to be accessed specified in the query can be a row , column or even a cell. Furthermore, to relieve the problem concerning the truthfulness of results returned to users, the authors in (81) designated the following models :

- *Truman-Model:* Applies the query rewriting technique to the user query to provide only the answers that are authorized to user. This approach maximizes the result query returned to the user. The main idea is that when a user executes a query Q against a database schema, if there exists a query Q' expressed in terms of the authorization views and Q' is included in Q then the result of Q' is returned instead of the result of Q.

- *Non-Truman-Model:* In such model, a query rewriting technique is also executed, but the difference according to the previous model is in the interpretation of the query. Thereby, when a query Q' is executed, if there exists a query Q' expressed in terms of the authorization views and Q is equivalent to Q' then Q' is mentioned as valid and the access to the data is granted. Otherwise, the access request is denied.

However, the view mechanism have some weaknesses (16). In fact, the access control policies are often different for different users, the number of views would further increases. Furthermore, application programs would have to code different interfaces for each user, or group of users to comply with the correct views. In addition, the modification of access control policy requires the definition of new views with consequent modifications to application programs.

In the next section, we will highlight how access control models were adopted to distributed environment.

2.3.3 Access Control Verification in Distributed Environment

In this section, we discuss security in distributed database systems. A distributed database system includes a Distributed Database Management System (DDBMS), a distributed database, and a network for interconnection (92). Access control in a distributed database system has been previously investigated.

2.3.3.1 Distributed Access Control Policies

According to (92), the access control enforcement in distributed environment could be centralized, distributed or replicated. In the case where the access control policy is centralized, a central server is in charge of granting/denying access to local sources. The central server stores all the access control rules of all users. If the authenticator is distributed, then the particular rule is located and enforced for particular access. In some cases, access rules attached to a particular database are stored in the same site with the database. In the case of replicated rules, each node of the distributed database system can enforce its proper rules for the data that it manages.

There is a consensus between security community that the distribution of the data exacerbates the management of access control. For example, enforcing rules where the access decision is based on the content of data is a complicated task. Another point is to ensure the consistency of the rules. When a rule is updated, this update should be propagated to all nodes in the distributed system.

2.3.3.2 Role-Based Access Control for Distributed Database Systems

Role-based access control model has been investigated for distributed database systems (92). For example, in (15), the authors developed a model for controlling access in workflow systems. Traditional role-based access control model can be extended to distributed environment. The idea is that a user can play different

roles in different distributed data sources. At this point, user role in different data sources need to be verified before granting access.

2.3.3.3 SQL for Distributed Database Security

SQL extension can be used to specify access control policies in distributed database systems. We have seen in the previous section the use of System R (GRANT/REVOKE) and authorization view to enforce access control policies in centralized database systems. In a distributed system, the access control specified in SQL for a centralized system remains the same (92). Nonetheless, some works have been going behind this. In (82), the authors have tackled the problem of how to automatically derive access control rules for the data warehouse from those of the sources. The inference of data warehouse permission has been done by extending the standard SQL GRANT/REVOKE model. The notion of access permission in this work was split in two types of permissions: permission information (for views) and physical permission (physical tables). This theory makes the system easier to administer and its application more robust.

2.3.3.4 Access Control Through Confidentiality Constraints

Access control through confidentiality constraints is an access control model adopted to guarantee data security and privacy in distributed relational database systems. According to (3), this access control model could define the privacy requirements and achieve it. The privacy requirements are specified as a set of confidentiality constraints C, expressed on the relational database schema. This model is based on *Open Policy* (83). This means that a user is allowed to access a resource if there is no confidentiality constraint denying this access. Each confidentiality constraint is represented by a subset, denoted c, of the attributes of the relational schema R. Semantically, this means that the user cannot see the combination between attributes in c at the same time. However, any proper subset of c may be revealed. For example, let's consider a database schema S with a user *nurse* and a confidentiality constraint c = {Patient_SSN, Patient_diagnosis}. According to this access control policy, the nurse is denied from the combination of attributes

Patient_SSN and Patient_diagnosis at the same time. Hence, the privacy is ensured by decomposing the database schema vertically, in such a way that the two attributes Patient_SSN and Patient_diagnosis are not visible together at the same partition in the distributed system.

2.3.3.5 Auditing a Distributed Database System

Database auditing can be performed by the database administrator to check the queries posed, the updates made, and other activities that may be performed. These activities can be extended to distributed database systems with two approaches (92). The first one is the central audit manager where all information are gathered and analyzed centrally. The second one is the distributing auditing where different sites communicating with each others to form a big picture. According to (13), the audit can be done at a fine-grained level through data mining techniques. However, this must be done by taking into consideration the privacy of the users of database system. This can be achieved by reconciling audit and user privacy (by using privacy-preserving data mining techniques).

2.3.3.6 Access Control Enforcement in Data Integration Systems

In this section, we investigate the main approaches aiming to enforce access control for data integration systems (DIS) through the mediator/wrapper architecture.

Despite the interesting researches developed for composing access control policies of distributed parties to ensure their interoperability, these approaches look like inadequate for data integration systems. This comes from the fact that the global synthesized policy cannot stand up to aggregation and combination problems such as second inference channel and privacy attacks. Those approaches are only interested in establishing connections between the systems to facilitate the user of one system accessing to another system through these links (87). This is what motivated researchers in the last two decades to give more interest to this field (5; 4; 56; 11; 76; 57; 85; 87; 82).

In (5), the authors analysed data leakage threats in DIS architecture. The threats mentioned were relevant to data level only and treated the following data

security proprieties : Confidentiality, privacy and trust. The performed analysis is based on the location threat. Hence, the possible data leakage location identified by authors are : Data leakage between data sources, data leakage between the integration location and other entities and data leakage from the data consumer side. The same authors developed a formal modeling of DIS security policies in (4) to help DIS designers in mitigating the threats of data leakage. This were achieved through the minimization of data exposure due to incorrect specification of the security policies. Their proposed framework applied a formal approach to model DIS security policies by a process having three steps of refinement :

- Modeling confidentiality
- Modeling privacy
- Modeling trust

Other works were interested in materialized views' security. In (40), authors have proposed a framework to effectively and efficiently select fine-grained access control rules from the physical tables of relational database to the set of materialized views derived from this database. The authors pointed out two main contributions: Firstly, they introduced a Datalog-based syntax and relating semantics to model and express access control rules over relational databases, and to benefice from the flexibility and expressiveness of such modeling formalism. Secondly, by resorting to the *VSP-Bucket* algorithm, which is based on the view-based query rewriting technique, authors were able to effectively and efficiently derive the set of access control rules to be attached to the materialized views. Based on this work, the authors in (11) provided a framework able to infer access control rules to be attached to materialized views from those attached to physical tables. To do this, the authors resorted to authorization views as access control rules and extended the *MiniCon* algorithm based on query rewriting to *HminiCon+*. Hence, they provided a sequence of query rewriting to derive the authorization views to be attached to materialized views. Nevertheless, the framework was considered too restrictive because it deals only with authorization views based on selection rules and conjunctive queries.

The inference problem and privacy concerns of views have been investigated through the works of (57; 86; 85). Besides to generating a global security policy at the mediator level that preserves the security policy of distributed sources, the authors of these works dealt with the inference problem that exploits semantic constraints to get access to unauthorized data. We will discuss those works with more details in section 2.4. Privacy-preserving data integration was discussed in (67). In this work, authors laid out a privacy-preserving framework for DIS. Firstly, they introduced a privacy framework which contains the following components: Privacy view, privacy policies and purpose statements. Then, they presented a schema matching algorithm that do not expose the source data and schema. Indeed, this match preserved privacy in two steps: Find matches and elaborating matches into semantic mappings. This schema matching solution was followed by human verification of matches and mapping creation. Moreover, a query across source strategies was elaborated to ensure query results without violating privacy policy, neither the leakage of information from query answer. These strategies were summarized through the following query techniques: Statistical databases, privacy-preserving join computation and privacy-preserving top K-query.

In the last two decades, security issues in data warehouses also received much interest from researchers. In (50), the authors introduced a framework that specifies security measures from the earliest stages of the data warehouse design in the multidimensional modeling process and enforces them. To the best of our knowledge, this is the first work that considers security issues in conceptual level during data warehouse design process. Hence, the authors resorted to an extension of the Unified Modeling Language for specifying security constraints in the conceptual multidimensional model. The advantage of this approach, is its independence from the target platform where the data warehouse has to be implemented. Furthermore, the work of (88) considered also security constraints in designing data warehouse from the earliest stage of the development process. The methodology introduced by the authors aimed to establish a framework for the design of secure data warehouse based on Model Driven Architecture (MDA) and Query/View /Transformation (QVT). The authors backed up their approach by the

fact that MDA and QVT covered all the design phases: Conceptual, logical and physical, and specified security measures in all of them. Thus, making the security rules to be closer to the end user.

It is clear to deduce from this review that the objective of access control models discussed along this section is to protect data from security breaches by intercepting every access request to data. The access decision is then established based on the profile of the user requesting access. An important point to take into consideration is the reconciliation between data security and privacy. As already mentioned, assuring data security requires among other measures creating user activity profiles for anomaly detection, collecting data provenance, and context information such as user location (14). In this situation, users may feel uncomfortable since data misuses by administrators may lead to privacy breaches. However, this is not always the case, many techniques have been developed in order to reconcile data security and privacy such as :

- Privacy-preserving data mining.
- Privacy-preserving data publishing.
- Privacy-preserving attribute-based fine-grained access control for data on a cloud.
- Privacy-preserving location-based role-based access control.

Next, we will discuss approaches aiming to protect information systems from insider threats, and how to detect and control security breaches caused by inference channels.

2.4 Data Protection from Insider Threat

Traditional access control models aim to prevent data leakage via direct accesses. A direct access occurs when a requester performs his query directly into the desired object. However, these models fail to protect sensitive data from being accessed with indirect access (49). Indirect accesses via inference channels occur

when a malicious user combines the legitimate response that he received from the system with metadata (Figure 2.5).

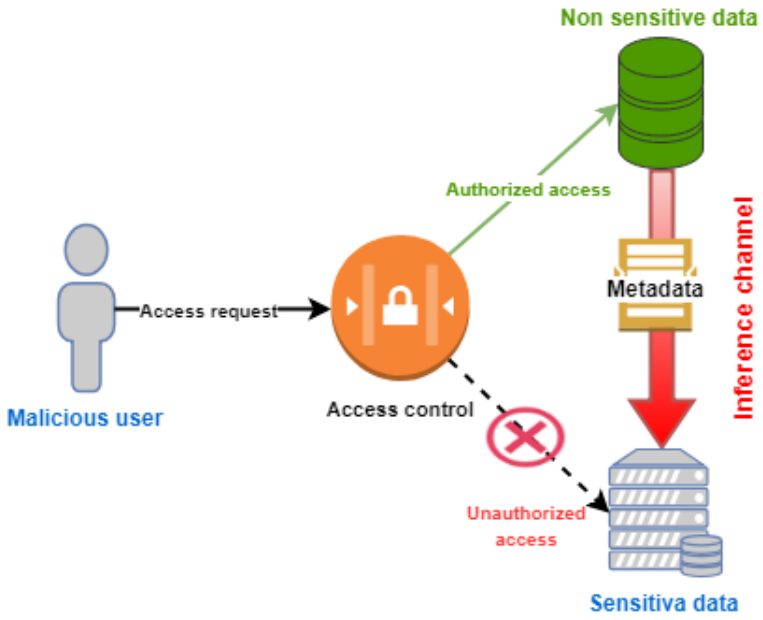

Figure 2.5: Bypass access control with inference channels

According to (55), external information to be combined with data in order to produce an inference channel could be database schema, system's semantics, statistical information, exceptions, error messages, user-defined functions and data dependencies. Detecting and removing inference in database systems guarantee a high quality design in terms of data secrecy and privacy since the latter is considered as a new vision of the inference problem. Absolutely, this diversity of techniques to bypass access control mechanisms with inference channels has attracted considerable attention in recent years. A growing body of literature has examined the inference problem but no one of those proposed solutions seemed to be a viable method to resolve the problem. In reality, for each of the underlying techniques, a specific solution has been proposed for handling each particular attack. There is consensus among security community that data distribution exacerbates inference problem. This is why several attempts have been done in the last two decades to address this problem.

In this section, we investigate current and emerging research efforts about in-

ference control in centralized database systems, then, we highlight inference control in distributed environment.

2.4.1 Access Control vs Inference Control

Traditional access control models aim to prevent data leakage via direct accesses. A direct access occurs when a requester poses his query directly on the desired object. However, these models fail to protect sensitive data from being accessed via indirect accesses (49). To deal with such situation, several works have augmented access control mechanisms with an inference engine in order to control both direct and indirect accesses. Although access control and inference control share the same goal of preventing data from unauthorized disclosure, they differ in several fundamental aspects (68). We give in table 2.2 the major differences between them.

Table 2.2: Access control vs inference control

Access Control	Inference Control
Direct access control	Indirect access control
Deterministic	Related to stochastic channels
Static: through a set of rules	Dynamic: vary through time and influenced by user action and queries
Normal expensive	More expensive then access control
Computational efficiency and high accuracy of security control	Efficiency and accuracy less than access control
Modular: can cover distributed environment	Adaptability to data distribution requires complicated techniques

According to our comparison table 2.2, we can note that access control is more preferable than inference control from a complexity perspective. Consequently, several researchers attempted to replace inference control engines with access control mechanisms (18; 68; 19).

2.4.2 Inference Control in Centralized Database Systems

Classical access control models have not been designed to protect data from inference. These models have been only designed to protect data from direct accesses. An inference problem, also called inference aggregation problem, occurs when a user deduces sensitive information from a sequence of innocuous information in the database. It has been widely investigated in the literature since 1987 with the emergence of multilevel database systems. The first works in this field are presented in (95; 90; 75).

In (95), the authors augmented the relational database with an inference engine, this module acted as an intermediary between the queries and the database. The inference engine resorted to first order logic to represent queries, security constraints, environment information and real world information. Hence, the current query was converted into a first order logic expression by the inference engine and then compared against the database constraints to determine if the query could lead to security constraint violation. If this is the case, and the query is identified as suspicious, then, it will be rejected. Otherwise, it will be converted to relational algebra and forwarded to the DBMS for execution. Results returning from the database were assigned a classification label to ensure the absence of unauthorized disclosure.

Morgenstern (75) presented a framework for inference analysis and detection during design and classification of data in multilevel database. The proposed methodology aimed to anticipate the inference channels in order to classify data. Hence, the authors introduced the classification process as a constraint satisfaction, then, they provided a concept called sphere of influence (SOI) to restrict the scope of possible inferences. In order to prevent these inference channels consisting of a set of objects from being achieved, they presented the following solutions: safety classification level which consisted of increasing the security clearance of an inferred object to an appropriate classification. Besides, a data sanitization process which consists of imposing aggregation classification on the set and classifies the aggregation in a way that the whole aggregate cannot be inferred from the individual values given to a lower level user. Furthermore, a third solution

was introduced aiming to disturb the inference channels and to prevent precise inference from being revealed by adding noise data.

The work in (90) investigated the inference problem due to functional dependencies (FD) and multivalued dependencies (MVD) in multilevel relational databases with attributes and records classification schemes. For inference channels produced by functional dependencies, the authors proposed a level adjustment algorithm to adjust attributes level with minimum information loss so that FD compromises are avoided. In the case of inference channels with MVD, authors introduced a level adjustment algorithm that adjusted security levels of tuples in relation instance to eliminate MVD compromises. However, there were some shortcomings to mention in this work. Firstly, the inference algorithms were given for each type of dependency separately but not when they were present together. Secondly, the authors considered only two types of dependencies, FD and MVD. Thirdly, the access control granularity was restricted to tuple level and single-attribute level. Finally, since their solution for eliminating a detected inference channel is based on increasing the classification level of individual attributes, it restricted the availability of data.

2.4.2.1 Inference Attacks

According to (49), there are three types of inference attacks:

- **Statistical attacks:** It is considered with statistical databases. The security breaches occur when confidential information about an individual is obtained by correlating different statistics while the security requirement is to provide access to statics about groups of entities while protecting the confidentiality of the individual entities (47).

- **Semantic attacks:** They aim to produce inference channels to bypass an access control model. In this type of attack, the semantic relationship between data is used to infer sensitive information. There exists three ways to produce a semantic inference attack including: Inference from constraints on queries, inference resulting from the combination of data with metadata and inference from value constraints (constraints defined over data).

- **Inference due to data mining:** Data mining is considered as one of the most used techniques for knowledge discovery and process. Nevertheless, it is considered as a double-edged weapon. From one hand, there is a consensus for the importance of data mining in data engineering and knowledge discovery. On the other hand, data mining can produce inference channels through its mining algorithms if the privacy of users is not respected.

For each of the mentioned techniques, researchers devoted a lot of efforts to deal with inference problems. For statistical attacks, techniques like *Anonymization* and *Data-perturbation* were developed to protect data from indirect access. For security threats based on data mining, techniques like *privacy-preserving data mining* and *Privacy-preserving data publishing* were carried out. Furthermore, a lot of works have investigated the semantic attacks (90; 31; 29).

2.4.2.2 Inference Prevention Methods

There exists in the literature more than one criterion to classify approaches that deal with inference. One proposed criterion is to classify these approaches according to data level and schema level (107). In such classification, inference constraints are classified into schema constraints level and data constraints level. Another criterion could be according to the time when the inference control techniques are performed. Therefore, the proposed approaches are classified into three categories: design time, query run time and at update time.

Design time approaches are considered as proactive techniques. They argue that since the inference problem is a resources consuming task, it is better to achieve all consuming processing offline before the system is executed. Moreover, since the inference is examined at a conceptual level in these approaches, detecting an inference channel is called second path detection (characterized as multiple paths between two attributes such that the path's security levels are inconsistent). In fact, these approaches exploit the semantic relationship between data and graphs to detect the second path (the inference channel). When an inference channel is detected, either the classification level of the inferred attributes is increased or the database schema is modified to block the inference channel from

being achieved. Approaches that deal with inference at design time are found in (80; 44; 58; 90; 79; 91; 59; 100).

Run time inference control techniques belong to reactive categories. these approaches aim to detect and block inference channels at query run time. For this purpose, the DBMS should be augmented with an inference engine to control the queries executed by the users against the database. The run time approaches are history-based since they combine the query executed by the user with the set of queries previously executed by the same user in addition to a set of semantic constraints to determine if the current query could lead to an inference channel. Several approaches supporting run time were developed in the past (31; 94; 9; 29; 95). It is important to note that users can collaborate to produce an inference channel based on the query sequences of collaborators. This problem was addressed in (32) where the authors developed a query-time inference violation detection model to evaluate collaborative inference based on the query sequences of collaborators and their task-sensitive collaboration levels. Besides, the work in (55) presented a database inference controller in the presence of probabilistic dependencies. Authors of (55) developed a formal language called ATKLOG providing an expressive and concise way to represent attacker's beliefs during their interaction with the system. By leveraging to this formal language, they developed AGERONA a secure inference controller mechanism to prevent the disclosure of sensitive information in presence of probabilistic dependencies.

In addition to design time and query run time categories, inference control can be performed at update time. To the best of our knowledge, this problem was only tackled in (96). In this work, the authors built on (29) to introduce a Dynamic Disclosure Monitor to prevent illegal inferences via database constraints at update time while maximizing data availability. The authors developed a module called *Update Consolidator* which uses the user's history file in addition to database updates and database constraints to produce a new history file that does not contain any outdated data values. Then, by resorting to *Update Consolidator* and *Disclosure Monitor*, the proposed approach guaranteed the completeness and soundness properties of the inference algorithm (data security and data availability) in the presence of updates. Furthermore, the authors proved through the experimental

results how their framework can handle collaborative inference.

2.4.2.3 Discussion of the Inference Prevention Methods

The purpose of inference control at design time is to detect inference channels from earliest stage and eliminate them. These approaches provide a better performance for the system since no monitoring module is needed when the users query the database, by consequence improving query execution time. Nevertheless, design time approaches are too restrictive and may lead to over classification of the data. Besides, it is required that the designer has a good concept of how the system will be utilized. On the other hand, run time approaches provide data availability since they monitor the suspicious queries at run time. However, run time approaches lead to performance degradation of the database server since every query needs to be checked by the inference engine. Furthermore, the inference engine needs to manage a huge number of log files and users. As a result, this could induce slowing down query processing. In addition, run time approaches could induce a non deterministic access control behavior (users with the same privileges may not get the same response).

From this perspective, we can conclude that the main evaluation criteria of these techniques is a trade-off between availability and system performance. Some works have been elaborated to overcome these problems especially for run time approaches. Example in (106) a new paradigm of inference control with trusted computing was developed to push the inference control from server side to client side in order to mitigate the bottleneck on the database server. Furthermore, in (89) the authors developed a run time inference control techniques while retaining fast query processing. The idea behind this work was to make query processing time depends on the length of the inference channel instead of user query history.

We assert that the distribution of the data exacerbates the inference and privacy problems. In the next section, we will investigate the inference problem in distributed environment.

2.4.3 Inference Control in Distributed Environment

Inference control in distributed environment has been investigated from early 2000 until now. This field of study has received the intention from researchers in database security, due to the fact that distribution aggravates inference problems and privacy concerns. In this section, we start by investigating research efforts on inference prevention in distributed data sources, then, we review different works for mitigating inference in data integration systems. We survey inference problem in data integration systems through the *Mediator/Wrapper* architecture for the reason that this is the most suitable design to access distributed, heterogeneous and autonomous data sources. We note that inference attacks and prevention methods destined for centralized database systems remain applicable in distributed environment.

2.4.3.1 Inference Control in Distributed Data Sources

In (30), the authors considered the inference problem where the data is combined from distributed database and released to the final users. In this situation of data dissemination, the problem arised when non-sensitive attributes compromise sensitive attributes. According to presented work, one technique to mitigate inference was by modifying the non-sensitive data in the database. Nevertheless, even with this modification, sensitive attributes still deducible when data from other databases is incorporated. For example, consider a database containing an attribute aids classified as sensitive. In addition, consider another database containing information about drug abuse, one can deduce from the two databases that drug abuser's injections may lead an individual to contract aids. As a result, the history of drug injection (which is considered non-sensitive), allows to infer with a higher chance if a user is contracted with aids or not even the diagnosis has not been released. In order to mitigate the occurrence of inference, the authors proposed a mechanism called *Relational Downgrader* that exploits Bayesian network to reason about probabilistic dependencies relationships among attributes of the distributed database systems. The Downgrader is composed of three components: the GUARD, the Decision Maker and the Parsimonious Downgrader. The

main idea behind this framework is to not release certain non-sensitive information that can lead to probabilistic inference about the sensitive information while minimizing the loss of functionality. Consequently, the outputs of the Downgrade are records that have been modified in order to anonymize sensitive attributes.

The authors of (97) built on (30) to develop a work turning around inference prevention in distributed database systems. They proposed an inference prevention approach that enabled each of the database in a distributed system to keep track of probabilistic dependencies with other databases and by consequence use that information to help preserve the confidentiality of sensitive data. The methodology is called "Agent-based" because every node in the distributed system is augmented with an agent to keep track of other nodes so that single point of failure and communication bottleneck are avoided. The principle of this framework is the following: In the first phase, a *Rule Generator* is developed to reflect the probability dependency relationship among the databases by the construction of a *Bayesian Network* that preserves the individual database dependencies and take into consideration dependencies between autonomous databases. Then, a set of rules are derived from the trained Bayesian Net by analyzing the inference of the sensitive target attributes. In the next phase, an agent which is considered as a downgrader, is attached to each machine in the distributed system. This agent blocks the inference in the local machine and the inference from several distributed machines by keeping track of these machines. However, this approach have some limits. It treats the case where the distributed databases are overlapped (similar or have common attributes). Moreover, it assumes that the records in the distributed databases share the same keys constraints.

Inference problems have been also investigated in Peer-to-Peer environment through the work in (43). The authors pinpointed the inference that occurred in homogeneous peer agent through distributed data mining. This process was called peer-to-peer agent-based data mining systems. They asserted that performing Distributed Data Mining (DDM) in such extremely open distributed systems exacerbates data privacy and security issues. As a matter of fact, inference occurs in DDM when one or more peer sites learn any confidential information (model, patterns, or data themselves) about the dataset owned by other peers during a data

mining session. The authors firstly classified inference attacks in DDM in two categories:

- *Inside Attack Scenario:* It occurs when a peer try to infer sensitive information from other peers in the same mining group. Depending on the number of attackers, the authors distinguished single attack (when one peer behaves maliciously) and coalition attack (when many sites collude ta attack one site). Moreover, a probe attack was introduced by the authors, which is independent of the number of peers participating in the attack.

- *Outside Attack Scenario:* It takes place when a set of malicious peer tries to infer useful information from other peers in a different mining group. In this case, eavesdropping channel attack is performed by malicious peers to steal information from other peers.

After identifying DDM inference attacks, the authors proposed an algorithm to control potential attacks (inside and outside attacks) to particular schema for homogeneous distributed clustering, known as *KDEC*. The main idea behind this algorithm is to reconstruct the data from the kernel density estimates since a malicious peer can use the reconstruction algorithm to infer non-local data. However, the proposed algorithm needs to be improved from an accuracy point to expose further possible weakness of the KDEC schema.

2.4.3.2 Inference Control in Data Integration Systems

Inference control in data integration systems were investigated in the last decade through the work of (57; 87; 85). In such system, a mediator is defined as a unique entry point to the distributed data sources. It provides to the user a unique view of the distributed data. From a security point of view, access control is a major challenge in this situation. two major security issues have been studied in this field : access control policies integration (5; 4; 11; 76; 56; 40; 86) and inference problems (57; 87; 85). In the former, researchers aim to enforce a global security policy which is deduced from the back-end data sources in addition to possibly enforcing additional security properties. In this situation, the global policy must

comply with the source policies. Figure 2.6 reports that complying with source policies means that a prohibited access at the source level should be also prohibited at the global level.

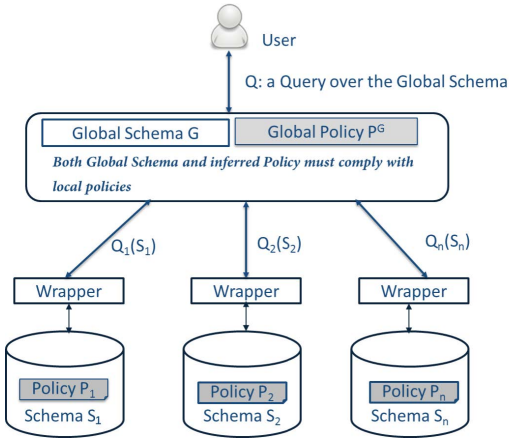

Figure 2.6: Secure Data Integration System

In (57; 87; 85), the authors demonstrated that despite the generation of a global policy at the mediator level that synthesizes and enforces the back-end data sources policies, security breaches were still possible via inference channel produced by semantic constraints (data dependencies). The problem is that the system (or the designer of the system) cannot anticipate the inference channels that arise due to the dependencies that appear at the mediator level. To the best of our knowledge, all the works that deal with inference problem in data integration systems are performed at query run time or at the global policy design time.

The first work attempting to control inference in data integration systems was introduced in (57). The authors proposed an incremental approach to prevent inference with functional dependencies. The proposed methodology included three steps:

- *Synthesizing global policies:* Derives the authorization rule of each virtual relation individually by the way that it preserves the local authorization of the local relations involved in the virtual relation.

- *Detection phase:* Identifies all the violations that could occur using functional dependencies by resorting to a graph based approach. Such violation

is called violating transaction that consists of a series of innocuous queries when it is achieved leads to violation of authorization rule.

- *Reconfiguration phase:* In this phase, the author proposes two methods to forbid the completion of each transaction violation. The first one uses a historic-based access control by keeping track of previous queries to evaluate the current query. This method is considered as a run-time approach. The second one proposes to reconfigure the global authorization policies at the mediator level in a way that no authorization violation will occur. This method is considered at design-time of the global security policy.

In this work, the authors discussed only semantic constraints due to functional dependencies. Neither inclusion nor multi-valued dependencies was investigated. Besides, other mapping approaches need to be discussed such as LAV and GLAV approaches.

The authors of (87) were inspired from (57) to propose an approach aiming to control inference in data integration systems. The proposed methodology resorted to formal concept analysis as a formal framework to reason about authorization rules and functional dependencies as a source of inference. The authors adopted an access control model with authorization views and proposed an incremental approach with three steps:

- *Generation of the global policy, global schema and global FD:* This step takes as input a set of source schema together with their access control policies and starts by translating the schema and policies to formal contexts. Then, the global policy is generated in a way that the source rules are preserved at the global level. Next, the schema of the mediator (i.e. virtual relations) is generated from the global policy to avoid useless attributes combination (every attributes in the mediator schema is controlled by the global policy). Finally, a global FD is considered from the source FD as a formal context.

- *Identifying disclosure transactions:* By resorting to formal concept analysis as a framework to reason about the global policy, the authors identified

the profiles to be denied from accessing sensitive attributes at the mediator level. Then, they extracted the violating transaction by reasoning about the Global functional dependencies.

- *Reconfiguration phase:* This step is achieved by two ways. At design time with a policy healing consisting to complete the global policy with additional rules in order that no violating transaction is achieved. At query run time with a monitoring engine to prohibit suspicious queries.

In (85), the authors examined the inference that arised in the web through RDF store. They proposed a fine-grained framework for RDF data, then they exploited close graph to verify the consistency propriety of an access control policy when inference rules and authorization rules interact. Without accessing the data at policy design-time, the authors proposed an algorithm to verify if an information leakage will arise given a policy P and a set of inference rules R. Furthermore, the authors demonstrated the applicability of the access control model using a conflict resolution strategy (most specific takes precedence).

Since the discussed works are recent, there are a number of concepts associated to security policies, privacy, data distribution and semantic constraints which could be considered to ensure better security and prevent inference from occurring in distributed environment:

- Absence of modularity in data integration systems. In the case where a new source joins the system, it is necessary to revise the global schema and the global policy. This is not suitable for distributed environment where the source joins and leaves the system continuously (e.g. Mobile environment).

- Authors dealt only with semantic constraints represented by functional dependencies and probabilistic dependencies as a source of inference. However, other semantic constraints, example inclusion dependencies, join dependencies and multi-valued dependencies should be considered as sources of inference.

- In data integration scenario, all approaches aim to handle inference at query run time by keeping track of the history of user queries and the current

query. In the case where the system deals with a large volume of data and users number, run time approaches will lead to performance degradation by slowing down query processing, consequently, this may push the server (mediator) to bottleneck. Hence, design time approach should be adopted to overcome these problems since it is performed offline.

2.5 Data Outsourcing in Presence of Access Control Policies

As we discussed in section 2.2, data owners place their data among CSSP in order to increase flexibility, optimize storage, enhance data manipulation and decrease processing time. Nonetheless, security concerns are widely recognized as a major barrier to cloud computing and other data outsourcing or Database-as-a-Service arrangements. Users are reluctant to place their sensitive data in the cloud due to concerns about data disclosure to potentially untrusted external parties and other malicious parts (105). Being processed and stored externally, owners cannot take control of their sensitive data anymore. Consequently, users privacy will be at risk. Two main approaches were adopted in secure data outsourcing: *Secure Data Outsourcing with Non-Communicating Servers and Secure Data Outsourcing with Communicating Servers*. Previous studies concentrated primarily on non communicating cloud servers (3; 34; 45; 35; 41). In this situation, servers were unaware of each other and did not exchange any information. When a master node receives a query, it decomposes it and processes it locally without the need to perform join query. In recent years, researchers studied the effect of communication between servers on query execution and secure query evaluation strategies have been elaborated (23; 22; 42; 46). We give in this paper a review study of current and emerging researches on privacy and confidentiality concerns in each of those approaches and discuss data outsourcing in relation with the inference problem.

2.5.1 Secure Data Outsourcing with Non-Communicating Servers

The first work attempting to enforce access control in database outsourcing using vertical fragmentation was presented in (3). Under the assumption that servers do not communicate, the work was aiming to split the database on two untrusted servers while preserving data privacy with some of the attributes possibly encrypted. Hence, a secure fragmentation of a relation R is a triple (F_1, F_2, E) where F_1, F_2 contain attributes in plain text stored in different servers and E is the set of encrypted attributes. The tuple identifier and the encrypted attributes are replicated with each fragment. The protection measures was also augmented by a query evaluation technique defining how queries on the original table can be transformed into queries on the fragmented table.

The work in (61) introduced an approach to enforce confidentiality and privacy while outsourcing data to CSSP. The proposed technique relied on vertical fragmentation and applied only minimal encryption to preserve data exposure to malicious part. However, the fragmentation algorithm enforced the database logic schema to be in third normal form to produce a good fragmentation design. Also, the query execution cost was not proven to be minimal. Authors of (33), addressed the problem of privacy preserving data outsourcing by resorting to the combination of fragmentation and encryption. The former was exploited to break sensitive associations between attributes, while the latter enforced privacy of singleton confidentiality constraints. Furthermore, the authors defined a formal model of minimal fragmentation and proposed a heuristic minimal fragmentation algorithm to efficiently execute queries over fragments while preserving security proprieties. Meanwhile, when a query is executed over a fragment involving an attribute that is encrypted, further query will be executed to evaluate conditions of the attributes and this will lead to a performance degradation by slowing down query processing.

In (36), researchers treated the concept of secure data publishing in presence of confidentiality and visibility constraints. By modeling these constraints as boolean formulas and fragment as complete truth assignment, the authors relied on Ordered Binary Decision Diagrams (OBDD) technique to check whether

a fragmentation satisfied confidentiality and visibility constraints. The proposed algorithm ran through OBDD and returned a fragmentation that guaranteed correctness and minimality. Nonetheless, query execution cost was not investigated in this paper and the algorithm ran only on database schema with single relation. Authors of (105) studied the problem of finding secure fragmentation with minimum cost for query support. Firstly, they defined the cost of a fragmentation F as the sum of the cost of each query Qi executed on F multiplied by the execution frequency of Qi. Secondly, they resorted to heuristic local search graph-based approach to obtain near optimal fragmentation. The search space was modeled as a fragmentation graph and transformation between fragmentation as a set of edges E. Then, two search strategies where proposed: a static search strategy which is invariant with the number of steps in a solution path, in addition to a dynamic search strategy based on guided local search which guaranteed the safeness of the final solution while avoiding dead-ends. However, this work did not investigate visibility constraints which is an important concept for data utility. Moreover, other heuristic search techniques could be addressed (i.e. Tabu search or simulated annealing).

The work in (35) has put forward a new paradigm to securely publishing data in the cloud while completely departing from encryption, since this latter is considered sometimes as a very rigid tool, delicate in its configuration, and may slow down query processing. The idea behind this work was to engage owner side (assumed to be a trusted part) to store a limited portion of data (supposed to be encrypted) in the clear and use vertical fragmentation to break sensitive associations among data to be stored in the cloud. The proposed algorithm computed a fragmentation solution that minimized the load for the data owner while guaranteeing privacy concerns. Moreover, authors highlighted other metrics that can be used to characterize the quality of a fragmentation and decide which attribute will be affected to client side and which attribute will be externalized. Even though, engaging the client to enforce access control requires the fact to mediate every query in the system which could lead to bottleneck and impacting performances.

Researchers in (26) proposed a separation of duties' technique based on vertical fragmentation to address the problem of confidentiality preserving when

outsourcing data to CSSP. To capture privacy requirement, confidentiality constraints and data dependencies where introduced in this work. The separation of duties problem was treated as an optimization problem to maximize the utility of the fragmented database and to enhance the query execution over the distributed servers. However, the optimization problem was addressed only from the point of minimizing the number of distributed servers. Besides, when collaboration between servers is established, the separation of duties approach is no longer efficient to preserve confidentiality constraints. The NP-Hardness proofness of the separation of duties problem discussed in (26) was proved in (27). The separation of duties problem was addressed as an optimization problem by the combination of the two famous NP-Hard problems: Bin packing and Vertex coloring. The bin packing problem was introduced to take into consideration capacity constraint of the servers in view that fragments should be placed in a minimum number of servers without exceeding the maximum capacity. Meanwhile, vertex coloring was introduced to enforce confidentiality constraints seeing that the association of certain attributes in the same server violated the confidentiality propriety. We would like to mention that this work tackled the separation of duties problem for single relation database and to make the theory applicable in practical scenarios many relations database should be treated.

Parting from the fact that communication between distributed servers in data outsourcing scenarios exacerbates privacy concerns, secure query evaluation strategies should be adopted. In the next subsection, we will investigate works turning around secure data outsourcing with communicating servers.

2.5.2 Secure Data Outsourcing: The Case of Communicating Servers

In the last years, few works investigated the problem of data outsourcing with communicating servers (23; 22; 42; 46). Besides to guarantee confidentiality and privacy preserving when distributing databases in the cloud, these works implemented secure query evaluation strategies to retain access control policy when servers communicate with each others. It is clear when servers (containing sensi-

tive attributes whose association is forbidden) interact through join queries, user's privacy will be at risk. Therefore, secure query evaluation strategies aim to prevent linkability between sensitive attributes when a malicious user tends to establish it.

Authors in (23) built on (22) to propose an approach that securely outsourcing data based on fragmentation and encryption. It also enforced access control when querying data by resorting to query privacy technique. The approach treated the case of multi-relations database and a new inter-table confidentiality constraints was introduced. It assumed that the distributed servers could collude to break data confidentiality so the connection between servers is supposed to be based on primary-key/foreign-key. In addition, the query evaluation model which is based on private information retrieval ensured data unlinkability performed by malicious user using semi join query. Yet, the proposed technique enforced database schema to be normalized, and generated a huge number of confidentiality constraints due to the transformation of inter-table constraints to singleton and association constraints which could affect the quality of the fragmentation algorithm. In addition, more generic queries should be considered.

Join query integrity check was tackled through the work in (42). In this work, researchers were inspired from (46) to propose a new technique in order to verify the integrity of join queries computed by potentially untrusted cloud providers. The authors aimed also in their approach to prevent servers to learn from answered queries which could lead to breach user's privacy. To do so, authors showed firstly how markers, twins, salts and buckets can be adopted to preserve integrity when a join query is executed as a semi-join. Then, they introduced two strategies to minimize the size of the verification: limit the adoption of buckets and salts to twins and markers only and representing twins and markers through slim tuples. Besides, authors demonstrated through their experiments how the computational and communication overhead can be limited due to integrity check.

To summarize, we can classify the discussed approaches according to the following criteria: confidentiality constraints support, optimal distribution support and secure query evaluation strategy support. We would like to mention that optimal distribution was treated through secure distributions that guarantee minimum query execution costs over fragments. From this point, it is clear

that all mentioned approaches supported access control verification through confidentiality constraints. However, query evaluation was not treated in all works (26; 33; 35; 41). Those approaches differ from the fact that some of them ensured minimum query execution costs and data utility for the database application, but other ones addressed the problem of data outsourcing with confidentiality constraints only. Besides, among the secure database distribution with query evaluation strategy, we see that the work in (23) provided an integral framework ensuring secure database fragmentation and communication between distributed servers. Also, it showed a reasonable query execution cost.

Nevertheless, those works assumed that the threat comes from the cloud service providers that try to collude to break sensitive association between attributes, it does not treat the case of internal threat where a malicious user aims to bypass access control with an inference channel. This is why we will present in the next section an insightful discussion about data outsourcing in presence of the inference problem.

2.5.3 Data Outsourcing and the Inference Problem

Data outsourcing and the inference problem is a research field that researchers begun to investigate few years ago. Inference leakage is recognized as a major barrier to cloud computing and other data outsourcing or Database-As-a-Service arrangements. The problem is that the designer of the system cannot anticipate the inference channels that arise in cloud level and could lead to security breaches.

In (21), authors resorted to a Controlled Query Evaluation strategy (CQE) to detect inference based on the knowledge of non-confidential information. Those information were supposed to be contained in the outsourced fragments and prior knowledge that a malicious user might have. Regarding that CQE relied on logic-oriented view on database systems, the main idea of this approach was to model fragmentation logic-oriented to allow for inference proofness to be proved formally even the semantic database constraints that an attacker may hold. Besides, vertical database fragmentation technique was considered by authors in (45) to ensure data confidentiality in presence of data dependencies among attributes. Those

dependencies allow unauthorized users to deduce information about sensitive attributes. In this work, three types of confidentiality violations that can be caused by data dependencies were defined: Firstly, when a sensitive attribute or association is exposed by the attributes in a fragment. Secondly, if An attribute appearing in a fragment is also derivable from some attributes in another fragment, thus enabling linkability among such fragments. Thirdly, when an attribute is derivable (independently) from attributes appearing in different fragments, thus enabling linkability among these fragments. To tackle these issues, authors reformulated the problem graphically through a hypergraph representation and then computed the closure of a fragmentation by deducing all information derivable from its fragments via dependencies to identify indirect access. Nevertheless, the major limit of this approach is that it explored the problem only in single relational database.

Despite that data outsourcing was not explicitly mentioned in (98; 99), these two works aimed to control inference problem caused by functional dependencies and meaningful join proactively by decomposing the relational logical schema into a set of views (to be queried by the user) where inference channels cannot appear. In (98), authors proposed a proactive and decomposition based inference control strategy for relational databases to prevent access to forbidden set via direct or indirect access. The proposed decomposition algorithm controlled both functional and probabilistic dependencies by breaking down those leading to infer a forbidden attribute set. However, this approach was considered too rigid by the fact that if the ways of associate forbidden sets attributes are defined as a chain of functional dependencies, the algorithm broke these chains from both sides for both attributes. Parting from this limit, the same researchers proposed a graph-based approach in (99) consisting of proactive decomposition of the external schema, in order to satisfy both the forbidden and required associations of attributes. In this work, functional dependencies are represented as a graph in which vertices are attributes and edges are functional dependencies. Inference channel is then defined as a process of searching a sub-tree in the graph containing the attributes that need to be related. Compared to the approach (98), in this one, the cut of the inference channel is relaxed by cutting the chains only at a single point, consequently minimizing dependencies loss. Nevertheless, like the

previous technique, it leads to semantic loss and need query rewriting techniques to query decomposed views.

2.6 Optimal Database Schema Partitioning

In recent years, some researchers devoted their research efforts to address the problem of vertical database schema partitioning in the CSSP level taking into consideration optimality constraints. These optimality constraints consist in minimizing the total resource consumption in the CSSP. In (38), the authors elaborated an experimental study to prove the relation between resource consumption and the queries distributed over multiple servers. In reality, minimizing the number of servers invoked in the users queries, enhances throughput and decreases latency which consequently will decrease resource consumption in CSSP level. Literature works dealing with this field of study could be classified in two categories :

- *Workload-aware approaches* (38; 69; 34; 105): These approaches focused on the workload (information regarding queries, their frequencies, involved attributes, arrival patterns, and so on) to derive an optimal schema distribution in the CSSP level. For example in (38; 69), authors relied on a hypergraph representation of the database where the tuples are represented by vertices and transactions (queries) are hyperedges that relate them. Hence, the partitioning algorithm minimized the number of cross-partition transaction by minimizing cuts of hyperedges in the hypergraph. However, these approaches are not always applicable since the query workload is not available in all scenario or it is useless due to errors in physical parameter estimations. For this reason, some workload-independent techniques have been proposed.

- *Workload-independent approaches* (104; 25; 103): These approaches attempted to exploit database semantics without queries in order to derive optimal vertical schema partitioning. Nonetheless, these approaches were not in line with our methodology for the following reasons: The work in (25) attempted to exploit the logical database schema by extracting the functional

dependencies from database tables and use them to perform partitioning. This could produce as discussed in (64; 63; 65) inference leakage by malicious users and then violate access control policies. Moreover, the work in (104) used an ontology-driven partitioning method to horizontally partition the data loaded in the relational schema according to their semantic similarity. Nevertheless, the adopted Ontology in this work heavily relies on structure designed primarily for is-a relation and it cannot detects similarity between concepts connected by other kinds of relations. Authors in (103) proposed a clustering based fragmentation and use replication technique to derive semantic fragmentation of data in the database for flexible query answering. They used medical taxonomy to measure semantic similarity between medical expressions. But, like the previous discussed approaches, this methodology requires data to be loaded in the database schema which is not the case in our situation.

2.7 Discussion

2.7.1 Choice of the Access Control Model

In order to choose the most suitable access control model to our context, we need to be aware of their adaptability to data outsourcing in order to ensure high level of data security and utility. Among the discussed models, we need to choose a model supporting the fallowing properties:

- ***Relational model support***: Since in thus thesis we are interested to the relational model, we need an access control applicable to this model.
- ***Can cover relational distributed database systems***: This is the most important property, since the access control model will be enforced in CSSP level where the database schema is stored in separate partitions.
- ***Truthfulness***: The model should ensure the correctness and Truthfulness of the results delivered to users.

- ***Inference control***: The adopted model should be able to break inference channels by exploiting the distribution of the database. This will enable us to control inference leakage without building an inference control engine.

Among the discussed access control models, we see that access control model through confidentiality constraints discussed in 2.3.3.4 seems to be a most suite model for the following reasons: Indeed, this model enforces access control policies through confidentiality constraint which is a set of attributes of the relational schema R where their visibility in the same partition is forbidden. So, we can deduce that this model can be easily adopted to the relational model. Furthermore, it can exploit the database distribution to enforce confidentiality constraints be keeping attributes belonging to the same constraint in separate partitions. In addition, since in this thesis we are interested in inference channels produced by functional dependencies. So, an inference channel is a sequence of attributes that functionally determine a confidentiality constraint. To block an inference channel, this access control model can exploit the database distribution to separate those attributes from each other in CSSP level. More over, access control model through confidentiality constraints has the lowest computational expensive compared to other models since it exploit the distributed database schema to enforce access control policies in CSSP level.

2.7.2 Discussion of the Proposed Approaches

The most relevant related works are classified in Table 2.3. The classification process is based on the following criteria: Confidentiality constraints support (enforced as hard constraints and guarantee access control rules), visibility constraints support (enforced as soft constraints and used to maximize data locality in CSSP level), inference control support and query evaluation strategy support.

The literature review provided above, highlights research efforts devoted to ensure data confidentiality when this later is externalized to the cloud. As we can deduce from Table 2.3, none of the proposed approaches investigated the problem considering simultaneously the following criteria: Access control support, inference leakage through semantic constraints, reducing distributed transactions

Table 2.3: A comparison between existing works and the proposed model

	Confidentiality constraints	***Inference control***	***Visibility constraints***	***Query evaluation***
(23)	yes	no	yes	yes
(27)	yes	no	yes	no
(105)	yes	no	yes	yes
(20)	yes	yes	no	no
(36)	yes	no	yes	yes
(33)	yes	no	yes	yes
(34)	yes	no	yes	yes
(46)	yes	no	no	yes
(45)	yes	yes	no	no
(99; 98)	yes	yes	yes	no
(26)	yes	no	yes	yes
(3)	yes	no	no	yes
(61)	yes	no	no	no
Required model	yes	yes	yes	yes

through visibility constraints and maintaining a secure query evaluation strategy. In fact, there is no significant contribution on data outsourcing guaranteeing the control of inference leakage caused by functional dependencies. Also, none of the cited works established a secure query evaluation strategy on a big data-based system. Through this research paper, we will address these limits. In reality, our contribution is seen through the adoption of vertical partitioning taking into consideration the most extreme situation faced by the database designer. Indeed, we suppose that neither the query workload is available, nor the database schema is loaded with data. Only the database schema with its semantic constraints and users privileges are available to the database designer. This further confirms to us the novelty of the research we propose.

2.8 Conclusion

In this chapter, we have investigated different works in relation with our research. First, we have highlighted the approaches of data outsourcing. Next, we discussed

access control models and their adaptation to distributed environment. Then, we have outlined existing works which have paid an attention to the inference problem and its outcomes on data outsourcing. Finally, a discussion was presented to pick the most suitable access control model and highlight the main issues in existing approaches. In the next chapter, we will introduce our preliminary concepts and problem statement.

Chapter 3

Preliminaries and Problem Statement

3.1 Introduction

In order to more explain our research problem, we will start by giving a set of definitions and concepts that will be needed throughout the next parts of this manuscript. Second, we will motivate our work through an example. Third, we will discuss the correlations between our studied problem and the relevant related fields. Finally, we will describe at an abstract level, our proposed methodology.

3.2 Preliminaries and Basic Concepts

Prior detailing our approach, we would like to introduce basic definitions in relation with our research.

3.2.1 Definitions Related to Access Control and Inference Problem

Definition 1. ***(Confidentiality constraint)*** *(27). Let R(A) be a relation schema over the set of attributes A. A confidentiality constraint on R(A) is defined by a subset of attributes* $c \subseteq A$ *with* $c \neq \emptyset$ *stating that a partition is not allowed to store the combination of attributes contained in c. However, any proper subset of c may be revealed. Without loss of generality we consider confidentiality constraints with cardinality greater or equal to 2 (* $|c| \geq 2$*).*

Definition 2. ***(Functional dependency)*** *(2). A functional dependency (FD) over a schema R is a statement of the form:* $R : I \longrightarrow J$ *where I, J two sets of attributes* $\subseteq$ *schema R stating that each I value in R is associated with precisely one Y value in R. We refer to I as the left hand side (LHS) and J as the right hand side (RHS) of the functional dependency* $I \longrightarrow J$*.*

Definition 3. ***(Meaningful join)*** *(99). Meaningful join is briefly an equi-join operation between a foreign and primary keys. The inference problem is defined on inhibiting all possible meaningful joins, among the attributes of confidentiality constraints.*

Definition 4. ***(Violating Transaction)*** *(57). A violating transaction* $T = \{Q_1, ..., Q_i, ...\}$ *is a set of select queries such that if they are executed and their results combined, they will lead to disclosure of sensitive information and thus violating the confidentiality constraint.*

3.2.2 Definitions Related to Graph Theory

Definition 5. ***(Graph)*** *(54). A graph G = (V, E) is a mathematical structure consisting of two finite sets V and E. The elements of V are called vertices (or nodes) , and the elements of E are called edges . Each edge has a set of one or two vertices associated to it , which are called its end points.*

Definition 6. ***(Hypergraph)*** *(28). It is a generalization of a graph in which an edge can connect any number of vertices. Formally, a hypergraph H is a pair* $H = (X, E)$ *where X is a set of elements called nodes or vertices, and E is a set of non-empty subsets of X called hyperedges.*

Example:
Let M be a computer science meeting with k ≥ 1 sessions : $S_1, S_2, S_3, ..., S_k$. Let V be the set of people at this meeting. Assume that each session is attended by one person at least. We can build a hypergraph in the following way:

- The set of vertices is the set of people who attend the meeting:
- the family of hyperedges $(e_i), i \in \{1, 2, ...k\}$ is built in the following way: $e_i,\ i \in 1, 2, ...k$ is the subset of people who attend the meeting S_i.

3.2.3 Definitions Related to User Role

Definition 7. ***(User-role)*** *A role is a database object grouping a set of privileges and assigned to one or more user. A role is allowed to access a resource if there is no confidentiality constraint denying this access. To each role* r_i *we associate a head called* $head_i$ *containing a set of terms* $head_i = \{t_1, ..., t_l\}$ *which semantically describe the role* r_i.

Definition 8. ***(Role label)***
Let $r = \{r_1, ..., r_m\}$ be the set of all users roles in the database, and let L_{rd} and L_{rl} be two labels:

- *$r_i(L_{rd})$ states that the role r_i is allowed to execute distributed queries over K-partitions.*
- *$r_j(L_{rl})$ states that the role r_i is allowed to execute only local queries (every query issued from this role cross exactly one partition).*

.

Definition 9. ***(Visibility constraint)*** *(36). Let R(A) be a relation schema over the set of attributes A. A visibility constraint is a monotonic Boolean formula over attributes in R(A). A Visibility constraint $v = \{a_1, ..., a_m\}$ states that attributes $a_1, ..., a_m$ must be jointly visible in the same partition.*

3.3 Problem Statement

In this section, we define the tackled problem as follows: *"Given a relational database schema to which is attached a set of access control polices, a set of functional dependencies and a set of users roles, our approach should answer the following questions:"*

- *How database roles can be exploited to reduce the amount of servers that a user query has to span in CSSP level?*
- *How to capture inference leakage caused by functional dependencies and how to control them?*
- *What theory should be adopted to generate a partitioning algorithm that preserves access control polices when data is externalized to CSSP and reduces the amount of distributed queries?*
- *How to evaluate distributed queries which are derived from the aggregation of partial results from distributed partitions while retaining access control policies?*

3.4 Motivating Scenario

To illustrate the ideas behind our proposed methodology, a hospital database schema is considered as shown in Figure. 3.1. The schema is designed to store patient's medical records about addiction cure. Hence, it is clear that exposing such sensitive information to untrusted parties will bring security breaches.

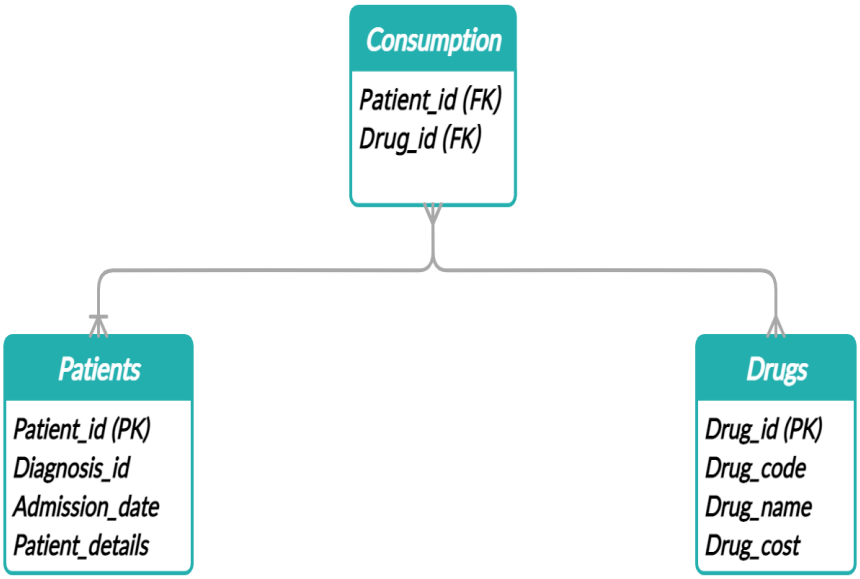

Figure 3.1: Hospital-db-schema

We consider the roles: R_1*: doctor and* R_2*: chemical-drug-agent*
Relation: Patients(Patient_id, Diagnosis_id, Admission_date, Patient_details)
Functional dependencies: FD_1 : Patient_id $\longrightarrow$ Diagnosis_id; FD_2 : Patient_id $\longrightarrow$ Admission_date; FD_3 : Patient_id $\longrightarrow$ Patient_details;
FD_4 : Diagnosis_id $\longrightarrow$ Patient_id; FD_5 : Diagnosis_id $\longrightarrow$ Patient_details.

Relation: Drugs(Drug_id, Drug_code, Drug_name, Drug_costs)
Functional dependencies: FD_6 : Drug_id $\longrightarrow$ Drug_code; FD_7 : Drug_id $\longrightarrow$ Drug_name; FD_8 : Drug_id $\longrightarrow$ Drug_cost.

Confidentiality constraint: c_1 = {Drug_code, Patient_details}

We would like to mention that unlike previous works which impose that the attributes belonging to the same confidentiality constraint must be in the same relation, we authorize in our work the use of inter-relation confidentiality constraints like constraint c_1.

Relation: Consumption(Patient_id, Drug_id)
Functional dependencies: FD_9 : Patient_id,Drug_id $\longrightarrow$ Patient_id;
FD_{10} : Patient_id,Drug_id $\longrightarrow$ Drug_id

It is clear that the association of sensitive attributes is the main cause of exposing data in cloud to unauthorized disclosure. According to c_1, neither the doctor nor the chemical-drug-agent are allowed to see at the same time the visibility of attributes Drug_code and Patient_details. This confidentiality constraint is specified according to open policy where the access control rules specify all requests that are to be denied. Under such a policy, any request that is not denied by the access control rule is allowed by default. Let's see now how functional dependencies could be used by a malicious user to produce an inference channel and violate the confidentiality constraint c_1. Let us assume the following queries issued by a doctor: $Q_1\{Patient\_id,\ Drug\_id\}$, $Q_2\{Patient\_id,\ Patient\_details\}$ and $Q_3\{Drug\_id,\ Drug\_code\}$. Combining the results of these queries and using functional dependencies $\{FD_9,\ FD_3,\ FD_{10},\ FD_6\}$, the doctor can derive meaningful joins to obtain Patient_details and Drug_code simultaneously which induces the violation of the confidentiality constraint c_1. These observations encourage the idea of partitioning vertically the relations into fragments in CSSP level to preserve confidentiality constraints and break inference channels while maintaining some attributes visible together to ensure data utility. Next, we propose an approach to solve this problem.

3.5 Discussion of Studied Problem with their Relevant Fields

Security-aware data outsourcing, as shown in Figure 3.2 , is a complex and challenging task because different fields are involved in such an issue. In this section, we discuss the different fields that we identified to be relevant to our problem. The selected fields include data outsourcing and cloud computing, access control models, inference problem and domain ontology.

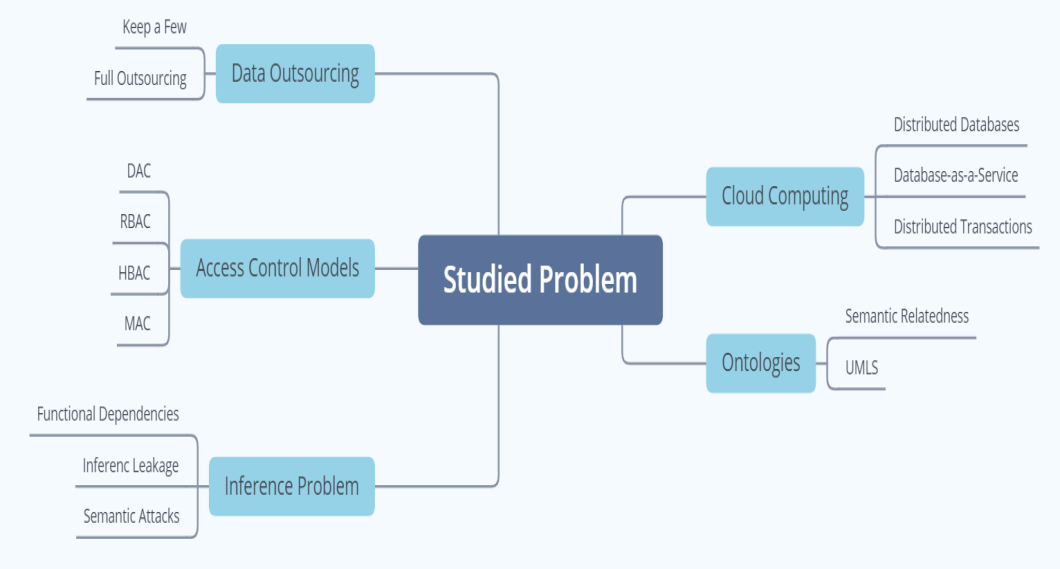

Figure 3.2: Overview of the studied problem with their relevant fields

3.5.1 Data Outsourcing

As we discussed previously, data owners place their data among CSSP in order to increase flexibility, optimize storage, enhance data manipulation and decrease processing time. In chapter 2, we mentioned that there exists two approaches to externalize data to CSSP: Keep a few (35) and full outsourcing (3). The former consists to store portion of the data in the owner side since this later is considered as a trusted part. The rest of the database is distributed among servers while maintaining data confidentiality through vertical fragmentation. The latter con-

sists to externalize the hole in-house database to the cloud. It considers vertical database fragmentation to enforce confidentiality constraints with more than two attributes by keeping them separated from each other among distributed servers. In this thesis, we consider the case where the data owner is outsourcing all his data to CSSP. We make this choice because the data placed in owner side don't need any security mechanism to be protected. Indeed, owner's access request will be forwarded directly to these data which are under the owner control. Furthermore, maintaining a portion of the data in the owner side and involving this latter as a trusted party will increase the load requested from the owner to query these data.

3.5.2 Access Control Model

Securing data in CSSP level is a challenging task. The main objective is to preserve owner's access control policies when data is externalized to the cloud. Preserving access control policies means that if an access is initially prohibited by owner's access control policies, it should be also prohibited in CSSP level. As discussed in the previous chapter, several models (DAC, MAC, ABAC, VBAC, RBAC, HBAC) have been proposed in literature. We saw that access control model through confidentiality constraints is the most flexible model when we consider data outsourcing scenario. Indeed, this model exploits the distributed architecture of the system to enforce access control policies by keeping sensitive attributes separated from each others.

3.5.3 Inference Control

Access control models protect sensitive data from direct disclosure via direct accesses, however, they fail to prevent indirect accesses (49). Indirect accesses via inference channels occur when a malicious user combines the legitimate response that he received from the system with metadata. In this manuscript, we consider semantic attacks which happen at CSSP level. In particular, we treat the case where a malicious user is issuing a set of well chosen queries and taking advantage of functional dependencies to infer sensitive information.

3.5.4 Ontology-Based Vertical Database Schema Partitioning

Since we place ourselves in the most extreme situation faced by the database designer, we suppose that neither the query workload is available, nor the database schema is loaded with data. Hence, deriving an optimal distributed database schema in this situation is a challenging task. In our approach, we propose an ontology-based technique to generate the set of visibility constraints which will guide the process of vertical schema partitioning. Our objective is to reduce the amount of servers that a user query has to span, so it reduces the communication cost over sites, avoids excessive network and delays and improves data locality (38).

3.6 Overview of the Proposed Approach

To achieve the goal of a security-aware data outsourcing, we propose an approach that relies on the relational model and aims, as shown in Figure 3.3, to produce a set of secure subschemes. Each subschema is represented by a partition P_i and each partition is stored exactly in one server in the CSSP. Furthermore, it introduces a secure distributed query evaluation strategy to efficiently request data from distributed partitions while retaining access control policies. To do that, our methodology takes as input a relational schema R to which is attached a set of confidentiality constraints C, a set of functional dependencies FD and a set of users roles $r_{i,i\in\{1,\ldots,m\}}$, where m is the number of roles in the database, then it runs through the following phases:

1. *Constraints generation:* This phase aims to generate two types of constraints: Constraints-based inference control and visibility constraints. These two types of constraints that in addition to the confidentiality constraints specified initially by data owner, will guide the process of vertical schema partitioning.

2. *Schema partitioning:* In this phase, we rely on hypergraph theory to represent the partitioning problem as a hypergraph constraint satisfaction prob-

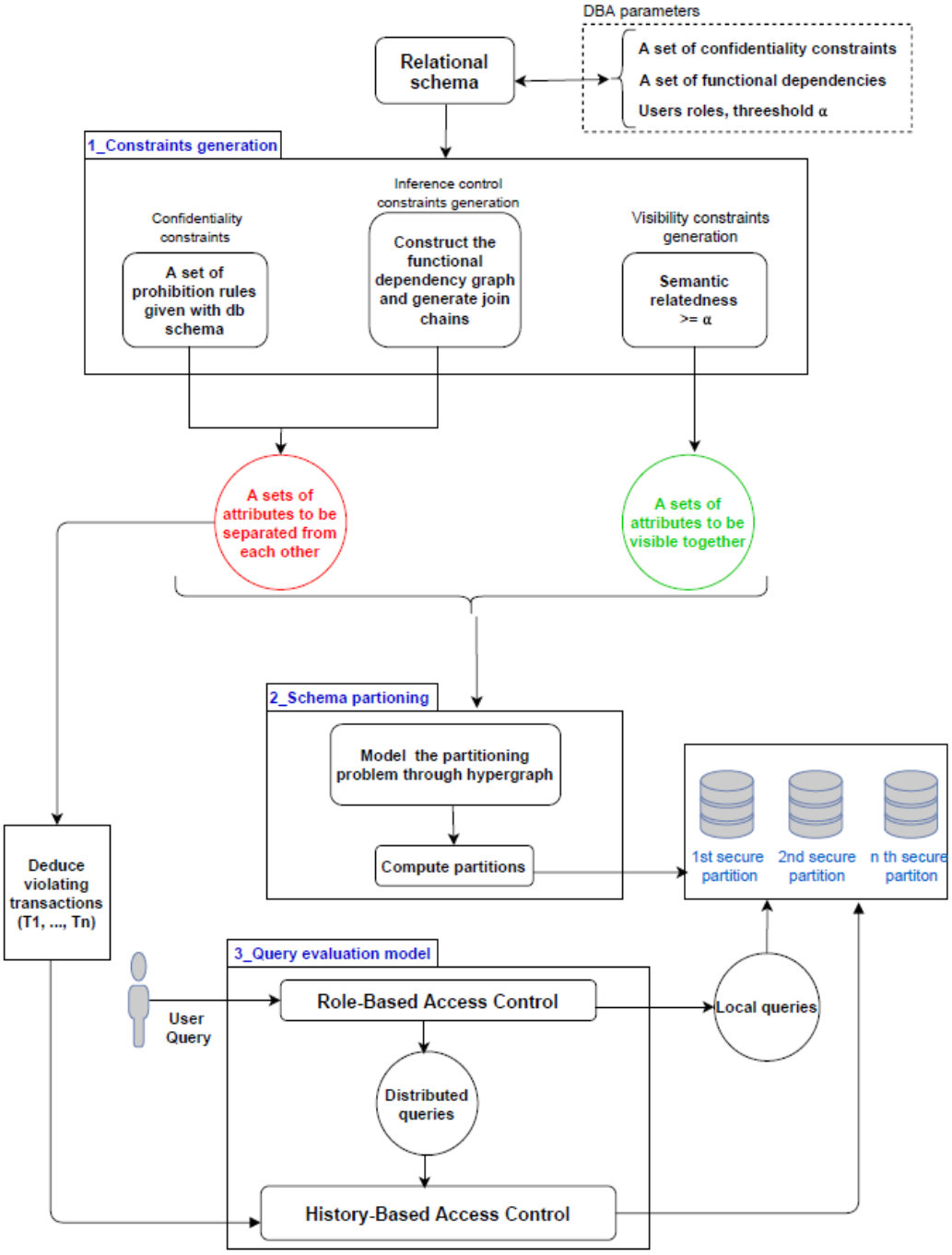

Figure 3.3: The proposed methodology to generate secure partitions and lock suspicious queries

lem. Then, we reformulate the partitioning problem as a multi-objective function F to be optimized. After that, we propose a greedy algorithm to partition the constrained-hypergraph into k partitions while minimizing the multi-objective function F.

3. *Query evaluation model:* In this phase, we propose a monitoring module to mediate every access request issued from users against data stored in distributed partitions and decide whether the access should be allowed or denied. The monitor module contains two mechanisms: Role-Based Access Control mechanism and History-Based Access Control mechanism.

3.7 Conclusion

In this chapter, firstly, we presented the definitions and basic concepts related to our work. Secondly, we introduced the motivating scenario. Thirdly, we discussed the studied problem in relation with relevant fields. Then, we gave an overview of our proposed methodology. The next chapter 4 introduces the first phase of our approach. It draws up how to generate the visibility constraints and constraint-based inference control constraints that guide the process of schema partitioning. Chapter 5 presents the second and the third phases. It presents how to vertically partition the database schema in order derive the optimal and secure partitions. It highlights also the strategy of evaluating access requests and block suspicious ones. In chapter 6, we provide an experimental evaluation of our proposed approach and analysis of the proposed algorithms. Finally, we conclude our work while also opening new research directions.

Chapter 4

Constraints Generation

4.1 Introduction

In this chapter, we will introduce the first phase of our three-step approach. This phase aims to generate a set of constraints that will guide the process of schema partitioning and distribution in the cloud. First, in order to maximize data locality, we generate a set of visibility constraints based on semantic relatedness measure between users' roles and attributes in the relational schema. Those constraints will stay visible during the partitioning process. Second, to deal with inference channels produced by functional dependencies, we generate a set of constraint-based inference control to capture inference leakage and control them. Those constraint that in addition to confidentiality constraints specified initially by the data owner, will be distributed among partitions to enforce owner's access control policies and break inference channels.

To more illustrate our approach, We also apply in this chapter the proposed algorithms to our motivating scenario.

4.2 Visibility Constraints Generation Based on Semantic Relatedness

Our objective in this section is to demonstrate how to reduce the amount of servers that a user query has to span, so it reduces the communication cost over sites, avoids excessive network and improves data locality. To do that, we propose a technique that relies on semantic relatedness to derive an optimal vertical partitioning schema while preserving access control policies. Preserving access control policies means that a prohibited access on specified initially by the data owner should be also prohibited when the schema is fragmented and mapped to the distributed partitions in the cloud. We adopt vertical partitioning for the reason that we place ourselves in the most extreme situation that the database designer can face. In fact, we suppose that neither the query workload is available, neither the database schema is loaded with data so one can perform horizontal distribution like the work in (104). Our technique performs a semantic analysis of

the database logical schema in order to compute semantic relatedness between database schema attributes and users roles. Hence, attributes having a high score of relatedness to a given role are supposed to be frequently accessed by this role. Consequently these attributes will be assigned to the same visibility constraint that will be preserved when the database schema is fragmented (provided that this constraint is not in conflict with a confidentiality constraint).

Let's try now to apply the ontology-driven approach proposed in (104) to our scenario. We would like to clarify firstly that despite the ability of ontology to evaluate the closeness between two concepts from the whole set of their semantic links (including meronymy, antonomy and so on) by defining a topological similarity, the adopted ontology in (104) measures only semantic similarity between terms designed primarily only for the is-a relation. So, if our database schema contains the two concepts *doctor* and *gastrologist*, the ontology's measure will assign a high score of similarity between these two concepts, since a *gastrologist* is a *doctor*. But, for the role *doctor* and attribute *Patient_details*, the ontology's measure cannot detect the semantic relatedness between them, because in this work the adopted ontology is designed for is-a hierarchy and cannot detect relation between concepts connected by other kinds of relations. Moreover, according to this work, if we consider the three roles *doctor*, *gastrologist* and *cardiologist*, the similarity score between *doctor* and *gastrologist* is the same between *doctor* and *cardiologist* because the distance between them is the same. Having much equality in the similarity score will influence the quality of our algorithms. Moreover, the work in (25) attempted to exploit the logical database schema by extracting functional dependencies from the database schema and using them to perform partitioning. We would like to clarify that this functional dependency based approach is not applicable in our scenario, since this technique could produce as discussed in (64; 63; 65) inference leakage by malicious users and then violate access control policies which is in contradiction with our proposal. We mention that we place ourselves in the most extreme case where neither the query workload is available, neither the database schema is loaded with data. The database designer has only the logical schema and users roles as input.

This example highlights the limitation of the proposed systems in answering

users queries. Next, we propose an intuitive approach for solving this problem while.

Definition 10. ***(Semantic relatedness)*** *It is a set of semantic relationship names describing relationships between concepts, i.e.,* $R = \{Is - A,\ Has - A,\ Part - Of,\ Has - Part,\ etc.\}$. *Hence, the semantic relatedness between a role* r_i *and an attribute* a_i *is calculated as the average of semantic relatedness between this attribute and all term* $t_k \in head_i$. *Consequently it holds that :*

$$relatedness(r_i,\ a_j) = \frac{\sum relatedness((t_k \in head_i),\ a_j)}{|head_i|} \tag{4.1}$$

where $|head_i|$ *denotes the set cardinality*

We suppose that each user is assigned at least to one role r_i in the database. Our work aims to maximize intra-dependency between attributes that seem to be frequently accessed by the same role by assigning them to the same partition, while minimizing the inter-dependency between attributes in separate partitions. The main idea is to detect semantic relatedness between user role and each attribute in the logical schema. If the semantic relatedness is greater than a threshold α, then this attribute will be added to a visibility constraint v_i. This constraint represents attributes that will be frequently accessed by the role r_i. We produce the set of visibility constraints as specified in the following definition:

Definition 11. ***(Visibility constraints generation)*** *Let R(A) be a relation schema over the set of attributes A and* $a_{j\in\{1,\dots,n\}} \in A$. *Let* $r_{i\in\{1,\dots,m\}}$ *be the set of users roles described respectively by* $head_{i\in\{1,\dots,l\}}$, *and let* α *be a threshold such that* $0 \leq \alpha \leq 1$. *Then, a set of visibility constraints* $v_{i,i\in\{1,\dots,m\}}$ *holds if:*

- *Vertical partitioning: for every visibility constraint* $v_{i,i\in\{1,\dots,m\}}$, $v_i \subseteq A$
- *Completeness: for every attribute* a_j *there is a visibility constraint* v_i *in which* a_j *is contained*
- *Reconstructability:* $A = v_1 \cup ... \cup v_m$
- *Threshold: for every attribute* $a_j \in v_i$, $\exists\ !\ r_i$ *such that* $relatedness(r_i, a_j) \geq \alpha$

Algorithm 1 produces the set of visibility constraints based on the semantic relatedness measure.

Algorithm 1 Generation of visibility constraints

Input: Attribute set A of the relation schema R(A); Database roles $r_{i,i\in\{1,\ldots,m\}}$; $head_{i,i\in\{1,\ldots,m\}}$; A threshold $\alpha, 0 \leq \alpha \leq 1$
Output: A set of visibility constraints V

Begin
$V \leftarrow \varnothing$
for each role r_i **do**
  for each attribute $a_j \in A$ **do**
    Compute relatedness(r_i, a_j)
    if relatedness(r_i, a_j) $\geq \alpha$ **then**
      $v_i \leftarrow v_i \cup \{a_j\}$
      $A \leftarrow A \setminus \{a_j\}$
    end if
  end for
  $V \leftarrow V \cup v_i$
end for
Return V
End

Consider our running example in Figure 3.1 with roles *doctor and chemical-drug-agent*. To simplify for database designer the specification of the threshold α in compliance with his requirements, let's consider the following ranges:

- Weak relatedness: $0 \leq \alpha \leq 0.3$
- Medium relatedness: $0.3 < \alpha < 0.7$
- Strong relatedness: $0.7 \leq \alpha \leq 1$

The semantic relatedness measure considered in this example is derived from the Unified Medical Language System (UMLS) as an ontology and the similarity interface on top of it [1]. By considering a threshold α belonging to the medium re-

[1] nlm.nih.gov/research/umls/index.html

latedness range, the set of generated visibility constraints according to algorithm 1 is represented as the following:

v_1 = {Patient_id, Diagnosis_id, Admission_date, Patient_details}

v_2 = {Drug_id, Drug_code, Drug_name, Drug_cost}

v_3 = {Patient_id, Drug_id}

Where v_1 is the visibility constraint assigned to role *doctor*, v_2 is the visibility constraint assigned to role *chemical-drug-agent* and v_3 is used to ensure data lossless after the partitioning phase. Hence, under the assumption that the distributed servers could communicate between each other, a conceivable partitioning of the logical schema according to constraints v_1, v_2 and v_3 is shown in figure 4.1 . Server 1 can be used to process queries issued from doctors and server 2 can process queries related to chemical-drug-agents. We mention that this partitioning will be refined all along the rest of this paper by considering confidentiality constraints.

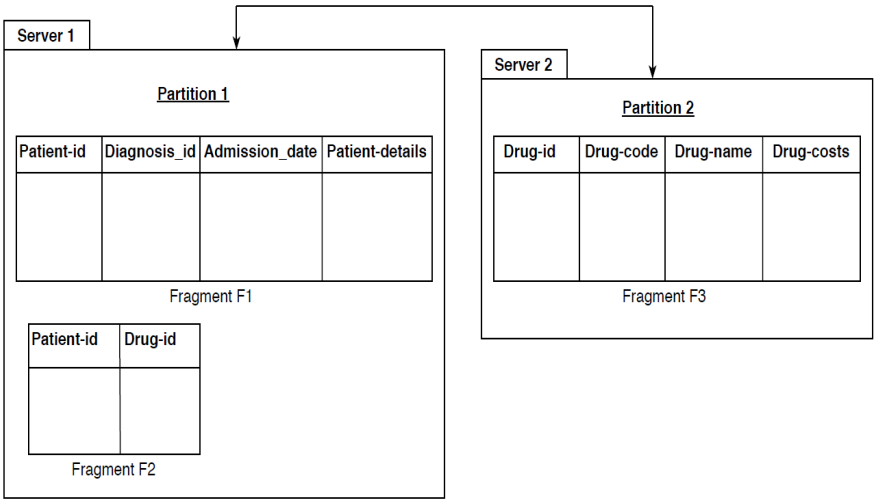

Figure 4.1: An example of a partitioning w.r.t visibility constraints v_1, v_2 and v_3

The Unified Medical Language System is a data warehouse including over 100 different biomedical and clinical data resources. One of the largest individual sources is the Systematized Nomenclature of MedicineClinical Terms (SNOMED CT), a comprehensive terminology created for the electronic exchange of clinical health information. It also implements various measures of similarity and

relatedness for WordNet [2]. The semantic relatedness measure is derived from Second order co-occurrence vectors computed from biomedical corpora, UMLS and WordNet. This method takes advantage of large corpora of medical texts, the UMLS Metathesaurus and Semantic Network, and concept definitions from WordNet.

To compute semantic relatedness, we could use the measure proposed in (71). This measure determines the relatedness between two terms/concepts by counting the number of overlaps between their two definitions. But, when implementing this measure in Wordnet, authors found that the definitions were short, and did not contain enough overlaps to distinguish between multiple concepts. Therefrom, authors in (73) have extended this measure using second-order co-occurrence vectors. In this approach, a vector is created for every word in the concept's definition containing terms that co-occur with it in a corpus. These word vectors are averaged to create a single co-occurrence vector for the concept. The relatedness between the concepts is calculated by taking the cosine between the concepts' second-order vectors.

UMLS compute the strength of semantic relatedness between any pair of concepts c_1 and c_2 by calculating the cosine of the angle between second-order vectors $\overrightarrow{c_1}$ and $\overrightarrow{c_2}$, (73) :

$$relatedness_{vector}(c_1,\ c_2) = \frac{\overrightarrow{c_1}\ .\overrightarrow{c_2}}{(c_1)\ .\ (c_2)} \tag{4.2}$$

This formula is applied to our proposal by considering each attribute in *R(A)* and each term belonging to role-head as a concept. Therefore, each concept c_i describes a term t_i. Hence, to measure semantic relatedness between a role r and an attribute a, this formula is computed n times between every term t in the role-head and the attribute a.

[2]https://wordnet.princeton.edu/

4.3 Inference Control

In this phase, we were inspired from (99) to propose a set of algorithms which will help us to identify and block inference channels performed with meaningful join (joins on key). The proposed technique performs a *relaxed_cut* on the join chains which will consequently minimize dependency loss.

4.3.1 Step 1: Building the Functional Dependency Graph G(V,E)

The aim of the G(V,E) is to list all the join chains that could be derived from a confidentiality constraints using functional dependencies. To build G, we resort to Algorithm 2 as follows :

1. Decompose all functional dependencies in F, such that each functional dependency will have a single attribute in the RHS.
2. Create an individual vertex for all attributes in schema and add to V .
3. Create vertices for the attribute sets with more than one element, which exist in the LHS side of any functional dependency and does not exist in V .
4. Create for each relation R_i an individual node in V .
5. For each $(A_i \longrightarrow A_j)$ in F, add an edge to E if A_i and A_j are different vertices in V.

Now let us apply Algorithm 2 to our previous example with the set of functional dependencies given in section 3.4. Then, the functional dependency graph constructed for this schema is given in Figure 4.2.

4.3.2 Step 2: Generating Join Chain Set

Definition 12. ***(Common Ancestor of a confidentiality constraint)*** *(99)*
It is a vertex in the Functional Dependency Graph, from which there exists simple paths to each attribute of the confidentiality constraint.

Algorithm 2 Building Functional Dependency Graph G(V,E)

Input: Attribute set A of the relation schema R(A); The set of functional dependencies F
Output: Functional dependency graph G(V,E)

Begin

V $\leftarrow$ {}
$E \leftarrow$ {}
▹ Step 1

for each $F_i(A_i \longrightarrow A_j)$ in F **do**

if |A_j| > 1 **then**

remove F_i from F

for each $A_k \in A_j$ **do**

Add $A_i \longrightarrow A_k$ to F

end for

end if

end for
▹ Step 2

for each relation $R_i \in A$ **do**

for each $A_j \in R_i$ **do**

Add A_j to V

end for

end for
▹ Step 3

for each $F_i(A_i \longrightarrow A_j)$ in F **do**

if |A_i| > 1 *and* $A_i \notin V$ **then**

Add A_i to V

end if

end for
▹ Step 4

for each relation $R_i \in A$ **do**

if $R_i \notin V$ **then**

Add R_i to V

end if

end for
▹ Step 5

for each $F_i(A_i \longrightarrow A_j)$ in F **do**

if $A_i \neq A_j$ and $A_i \in V$ and $A_j \in V$ **then**

Add $A_i \longrightarrow A_j$ to E

end if

end for

End

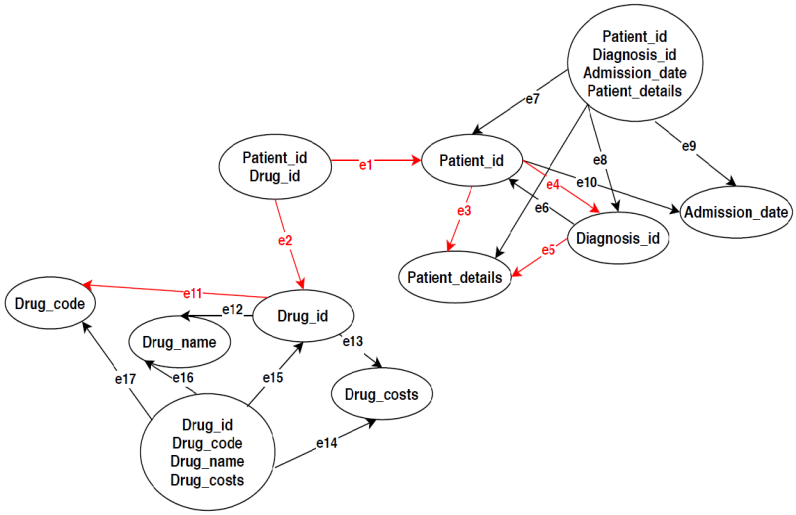

Figure 4.2: The functional dependency graph

In Figure 4.2 (Patient_id, Drug_id) is the common ancestor for the confidentiality constraint c_1.

Definition 13. ***(Join Chain of a confidentiality constraint)*** *(99) (Denoted as JC hereafter) is the set of edges of simple paths in the Functional dependency Graph, from a common ancestor to the confidentiality constraint.*

This step aims at identifying the set of join chain that can be used from a malicious user to derive a confidentiality constraint using meaningful join. The steps of the join chain detection algorithm are as follows:

1. All edges are reversed.
2. Taking each element of attribute set (A) as starting vertex, apply DFS up to all connected vertices and all possible simple paths are determined for each end vertex.
3. If there exists simple paths to the same end vertex, which are common (common ancestors in original FDG) for all set attributes (assumed to be starting vertices), all combinations of constructed simple paths, starting from different set attribute and ending in the same vertex is a join chain.
4. If a chain composes another, it is discarded.

Algorithm 3 Join Chain Detection

Input: Functional dependency graph G(V,E) ; Attribute set : A
Output: JC: join chain set

Begin
▹ Step 1

$JC \leftarrow \{\}$

for each $(A_i \longrightarrow A_j) \in E$ **do**

remove $A_i \longrightarrow A_j$ from E

Add $A_j \longrightarrow A_i$ to E

end for
▹ Step 2

Initialize $TargetArr$ as array of array of vertices

Initialize $PathArr$ as array of array of set of edges

for each $A_i \in A$ **do**

$C_{Ai} \leftarrow$ *empty set of connected vertices*

$P_{Ai} \leftarrow$ *emptysetofpathedgesets*

C_{Ai}, $P_{Ai} \leftarrow$ *apply DFS to FDG with starting vertex* A_i

j ← 0

for each C_j in C_{Ai} **do**

TargetArr[i][j] ← C_j

PathArr[i][j] ← *simple path to* C_j *in* P_{Ai}

j ← $j + 1$

end for

end for
▹ Step 3

$SharedArr \leftarrow$ array of shared vertices by all Target arrows

for each $S_i \in SharedArr$ **do**

Add $\cup(PathArr[k][l]\ as\ TargetArr[k][l] = S_i)$ to JC

end for
▹ Step 4

for each $JC_i \in JC$ **do**

for each $JC_j \in JC$ **do**

if $JC_j \subseteq JC_i$ **then**

remove JC_i from JC

end if

end for

end for

End

After applying algorithm 3 to our example with the confidentiality constraint $c_1 = \{Drug\_code,\ Patient\_details\}$ and common ancestor $\{Patient\_id, Drug\_id\}$, the following set of join chain is generated (the join chains are emphasized with red color in Figure 4.2).
$JC_1 = \{Patient\_id, Drug\_id \longrightarrow Drug\_id; Drug\_id \longrightarrow Drug\_code;$
$Patient\_id, Drug\_id \longrightarrow Patient\_id; Patient\_id \longrightarrow Patient\_details\}$
$JC_2 = \{Patient\_id, Drug\_id \longrightarrow Drug\_id; Drug\_id \longrightarrow Drug\_code;$
$Patient\_id, Drug\_id \longrightarrow Patient\_id; Patient\_id$
$\longrightarrow Diagnosis\_id; Diagnosis\_id \longrightarrow Patient\_details\}$

4.3.3 Step 3 : Detecting relaxed_cut

Definition 14. ***(relaxed_cut detection)***
It consists in detecting the minimum number of functional dependencies in the FDG in order to cut the join chains of the confidentiality constraints to satisfy all security requirements (not allowing a malicious user to derive confidentiality constraint attributes using join chains).

It has been shown in (12) that the relaxed_cut problem is equivalent to Minimum Hitting Set Problem and thus it is NP-Complete. Hence, we give here algorithm 4 inspired from (99) to greedily resolve this problem. The steps of the algorithm are as follows:

1. Compute all join chains (JC_i) for each confidentiality constraint (Algorithm 3).

2. For each edge in the FDG, determine the number of times (SecurityCount) it appears in join chains.

3. Sort the edges first according to their SecurityCount in descending order, then, the number of attributes on the nodes at both sides of the edge, in ascending order (in order to detect lower chains first to cut).

4. Cross the sorted list and mark each join chain as cut, if the edge is contained. These edges are selected ones and to be a selected one, an edge should be

an element of at least one unmarked join chain. A set of selected edges will be enforced in the next algorithm (phase 4) as new generated confidentiality constraints.

Now, let's execute algorithm 4 on the previous example by considering FDG in Figure 4.2 and the set of join chain $\{JC_1, JC_2\}$. A naive approach could choose to select the edge e_3 as a cut point (since the dependency $Patient\_id \longrightarrow Patient\_details$ is already enforced as a confidentiality constraint and this will minimize dependency loss). However, this will break only JC_1 and it will not break the join chain JC_2. According to algorithm 4 edge e_{11} ($Drug\_id \longrightarrow Drug\_code$) is the preferred cut point since this will break JC_1 and JC_2 simultaneously while minimising dependency loss (we mean by minimizing dependency loss breaking JC_1 and JC_2 by placing the LHS and RHS of the selected edge in different partitions when processing the partitioning step).

4.3.4 Step 4 : Constraints-based inference control generation

Unlike the technique proposed in (99) to break the join chains which leads to data loss and needs query rewriting techniques to query decomposed views, we propose here an algorithm to enforce each cut point selected by algorithm 4 as a confidentiality constraint. This will help the database designer to break join chains by placing the LHS and RHS of the selected edge in different parts when processing the partitioning step. Hence, a malicious user will need to perform a distributed query against the distributed partitions to derive a confidentiality constraint with a join chain. Indeed, this malicious technique will be treated in chapter 5 when we will elaborate a secure distributed query evaluation strategy. We would like to mention also that in order to guarantee data lossless when distributing fragments over partitions we proceed as the following : When a functional dependency of the form $A \longrightarrow B$ is cut a tuple identifier t_{id} is added to both fragments containing attributes A and B. Algorithm 5 proceeds as the following :

Applying algorithm 5 to the set of edges generated from algorithm 4 will extend the set of confidentiality constraints as the following :
$C = \{c_1, c_2\}$; $c_1 = \{Drug\_code, Patient\_details\}$, $c_2 = \{Drug\_id, Drug\_code\}$

Algorithm 4 Detecting Relaxed_cut

Input: The relational schema R(A); The set of confidentiality constraints C
Output: A minimum set of edges to be cut

Begin

$JC \leftarrow$ *empty array of join chain sets*
▹ Step 1

for each $c_i \in C$ **do**

$JC_i \leftarrow$ *join chain set for* c_i (*Algorithm* 3)

Add JC_i to JC

end for

FDG(V,E) $\leftarrow$ *FDG of* $R(A)$ (*Algorithm* 2)

SecurityCount $\leftarrow$ *array of integers initialized to* 0, *size* $|E|$
▹ Step 2

for each $E_i \in E$ **do**

for each $JC_k \in JC$ **do**

for each $JC_{ki} \in JC_k$ **do**

if $E_i \in JC_{ki}$ **then**

$SecurityCount[i]++$

end if

end for

end for

end for
▹ Step 3

$E \leftarrow$ *sort edges in descending order* (*uses SecurityCount*)

Selection $\leftarrow$ *empty set of edges*
▹ Step 4

for each $E_i \in E$ **do**

for each $JC_k \in JC$ **do**

for each $JC_{ki} \in JC_k$ **do**

if $E_i \in JC_{ki}$ and JC_{ki} is unmarked **then**

mark JC_{ki}

add E_i to Selection

end if

end for

end for

end for

End

Algorithm 5 Constraints-based inference control generation

Input: The set of selected edges to be cut (named Selection)
Output: An extended set of confidentiality constraints

Begin

$C \leftarrow$ *set of confidentiality constraints*

for each $e_i \in Selection$ **do**

$c_i \leftarrow \{LHS, RHS\}$ *of the edge* e_i

$C \leftarrow C \cup c_i$

end for

End

Until now, we have produced from the constraints' generation step of our proposed methodology three types of constraints : *confidentiality constraints, visibility constraints and constraints-based inference control* which will be considered as confidentiality constraints in the rest of this manuscript. Next, we will prove how these two constrains' types will guide the process of schema partitioning to produce a secure and optimal distribution.

4.4 Conclusion

In this Chapter, we have detailed the first phase of our proposed model, aiming to generate the constraints which will guide the database schema partitioning. First, to maximize data locality, we have generated a set of visibility constraints based on semantic relatedness measure between users roles and schema attributes. The proposed technique is workload-independent and don't rely neither on the query workload nor on the loaded data. Next, to detect inference channels caused by functional dependencies, we have built a functional dependency graph and we have proposed a set of algorithms to detect join chains aiming to derive confidentiality constraints through functional dependencies. Furthermore, we have proposed an algorithm to identify the relaxed cut point in order to break join chains while minimizing dependencies loss.

How to derive a partitioning technique which preserve those constraints, and

how to securely evaluate users queries is the topic of the next chapter.

Chapter 5

Schema Partitioning and Query Evaluation Model

5.1 Introduction

The objective of this chapter is to introduce the phase 2 and 3 of our approach. In phase 2, we will detail our partitioning technique to generate a set of secure partitions to be hosted in the distributed system in CSSP level. We will reformulate the problem as a multi-objective function F to be optimized. Then, we will introduce a greedy algorithm to partition the constrained hypergraph into k partitions while minimizing the multi-objective function F. In phase 3, we will propose a monitor module to mediate every query issued from users against data stored in distributed partitions and lock suspicious ones.

5.2 Schema Partitioning

In this phase of our approach, we resort to hypergraph theory to produce an optimal attribute placement decision while preserving confidentiality constraints. We first use hypergraph to model our partitioning problem graphically then we propose a greedy algorithm to produce an optimal partitioning while retaining access control constraints.

Hypergraph partitioning have been previously used in database partitioning (69; 38), In these approaches, authors have used workload based approach to database partitioning, by relying on a hypergraph representation of the database where the tuples are represented by vertices (horizontal distribution) and transactions (queries) are hyperedges that relate them. Then, the partitioning algorithm minimizes the number of cross-partition transaction by minimizing cuts of hyperedges in the hypergraph.

Our choice of hypergraph theory was inspired from the comparison study provided in (38), where authors have assessed the quality of partitioning in terms of the number of distributed transactions which is the same criterion of optimality adopted in this thesis. The comparison study has evaluated the number of distributed transactions produced by hypergraph partitioning algorithm, the manual partitioning (manual), replication of all tables (replication) and hash partitioning on the primary key or tuple id (hashing). The study has been evaluated on

9 datasets where hypergraph partitioning technique has demonstrated its performance compared to other algorithms.

5.2.1 Hypergraphs and constraint satisfaction problems

According to (28), a constraint satisfaction problem, P is defined as a tuple:

$$P = (W, D, D_1(S_1), ..., D_k(S_k))$$

where W is a finite set of variables, D is a finite set of values which is called the domain of P, $D_i(S_i)$ is a constraint, S_i is an ordered list of n_i variables, called the constraint scope, and D_i is a relation over D of arity n_i, called the constraint relation.
To a constraint satisfaction problem, we can associate a hypergraph in the following way:

- The vertices of the hypergraph are the variables of the problem.
- There is a hyperedge containing the vertices $x_1, x_2, ..., x_t$ when there is some constraint $D_i(S_i)$ with a scope $S_i = \{x_1, x_2, ..., x_t\}$.

By considering our running example, vertices represent the attribute set A and hyperedges represent both visibility and confidentiality constraints. Figure. 5.2 illustrates our partitioning problem with respect to visibility constraints v_i and confidentiality constraints c_i. Hyperedges e_1, e_2, e_3 represent the set of visibility constraint v_1, v_2, v_3 while hyperedges e_4, e_5 represent the set of confidentiality constraint c_1 *and*c_2. For the sake of simplicity, the following table illustrates the mapping between the attribute set A and the set of vertices x_i in the hypergraph.

To each hyperedge e_i, we assign a weight w_i. (We will explain the utility of this parameter when we introduce the multiobjective function). The dotted line in figure. 5.1 states that the two hyperedges e_4 and e_5 should be cut through the partitioning algorithm since these hyperedges represent the confidentiality constraints. However, hyperedges e_1, e_2 and e_3 which represent visibility constraints will be

Table 5.1: Attribute to vertex mapping

Vertex	Attribute
a	*Patient_id*
b	*Diagnosis_id*
c	*Admission_date*
d	*Patient_details*
e	*Drug_id*
f	*Drug_code*
g	*Drug_name*
h	*Drug_costs*

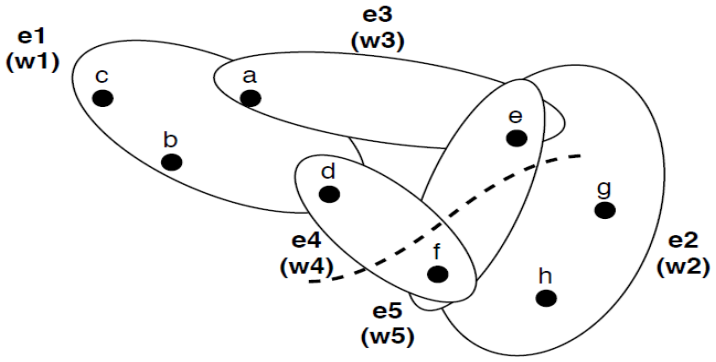

Figure 5.1: Hypergraph representation of the partitioning problem

preserved. Therefrom, we see that our objective function should maximize the hyperedges cut that represent confidentiality constraints and minimizes hyperedges cut representing the set of visibility constraints.

The traditional hypergraph partitioning techniques allow to optimize one of the objective functions : Minimize the communication cost between parts by minimizing hyperedge cut (min-cut), or declustering vertices in the same hyperedge by maximizing hyperedge cut (max-cut). Thus, we propose a multiobjective function that joins the two objectives max-cut and min-cut together. We use a priority-based formulation in which objectives are handled separately, our greedy algorithm computes a k partitions such that it simultaneously optimizes all objectives, giving preference to the objectives with higher priorities. In our algorithm, we try to minimize the following function:

$$F = \sum_{x_i \in e_c} W(e_c)\, \psi[x_i \notin P_k] + \sum_{x_j \in e_v} W(e_v)\, \psi[x_j \notin P_l] \tag{5.1}$$

- e_c and e_v are hyperedges representing confidentiality constraints and visibility constraints respectively.
- $Pi(1 \leqslant i \leqslant k)$: K-partitions to be computed.
- $\psi[true] = 1\ and\ \psi[false] = 0$.
- $W(e_c)$ and $W(e_v)$ are the weights of the hyperedge representing confidentiality constraint and visibility constraint respectively. The first term is the cost of violating a confidentiality constraint and the second one is the cost of violating a visibility constraint.

Please note that if a constraint is certain (the case of confidentiality constraints), its corresponding hyperedge in H(X,E) should have a large weight. In such a case, the algorithm will prioritize solutions that do not violate this constraint. If a constraint is uncertain (the case of visibility constraints), then its corresponding hyperedge in H(X,E) should be given a small weight, so the algorithm might give priority to solutions with a smaller first term.

5.2.2 Computing K-balanced partitions

We propose a greedy algorithm to compute k balanced partitions while minimizing the objective function F. Our algorithm is an attribute-to-partition mapping between individual vertex in H(X,E) and partition labels. We observed later in our experiments that this greedy approach mostly determines the optimal solution compared to other state of the art approaches. First, we would like to highlight the balance constraint parameter. This latter enforces the partitioning algorithm to generate partitions where the difference in the number of vertices between these partitions must not exceed a threshold fixed according to the imbalance tolerance. We tuned our partitioning algorithm to generate balanced partitions according to the number of vertices in H(X,E) as long as we place ourselves in the most extreme case where the workload is not available.

Let $\varepsilon \in [0, 1]$ be the imbalance tolerance. The balance constraint BC is defined as:

$$BC : \overline{W}.(1 - \varepsilon) \leqslant W(P_i) \leqslant \overline{W}.(1 + \varepsilon) \forall\ P_i \in P \text{ and } \overline{W} = \frac{(\sum_{x_i \in H} X)}{K} \tag{5.2}$$

- $\overline{W}$ is the total number of vertices in the hypergraph divided by the number of partition K.
- $W(P_i)$ is the number of vertices assigned to partition P_i.

Algorithm 6 Computing K-balanced partitions

Input: Hypergraph H(X;E); number of partitions K; the multiobjective function F; the balance constraint BC; the set of confidentiality constraints C
Output: K-balanced partitions

Begin

$Cutset \leftarrow \varnothing$
▷ Force the algorithm to prioritize confidentiality constraints by assigning large weights to these latters

Reweight $H(X; E)$ *such that* $Max(W(e_v)) \lll Min(W(e_c))$

while $Cutset \neq C$ **do**

Compute K-partitions by allocating vertices one by one while minimizing F and satisfying BC

if $\exists\ (x_i, x_j) \in e_c$ *such that* $x_i \in P_l$ *and* $x_j \in P_k$ **then**

Mark e_c *as Cut*

$Cutset \leftarrow Cutset \cup \{e_c\}$

end if

end while

if BC is not satisfied **then**

loop back

end if

End

Our algorithm minimizes F by prioritising the cut of hyperedges representing confidentiality constraints over preserving hyperedges that represent visibility

constraints. A cut of a confidentiality constraint is ensured by assigning vertices of the corresponding hyperedge to different partitions. We follow a greedy method by assigning vertices one by one while retaining the balance constraint. Considering our previous example, Figure 5.2 illustrates a possible partitioning according to algorithm 6 with the confidentiality constraints $\{c_1, c_2\}$, the visibility constraints $\{v_1, v_2, v_3\}$, a balance constraint $BC = 2$ and a number of partition $K = 4$.. The tuple identifier t_{id} is an artificial attribute added to ensure data lossless when a functional dependency involving a primary key was cut during the partitioning process.

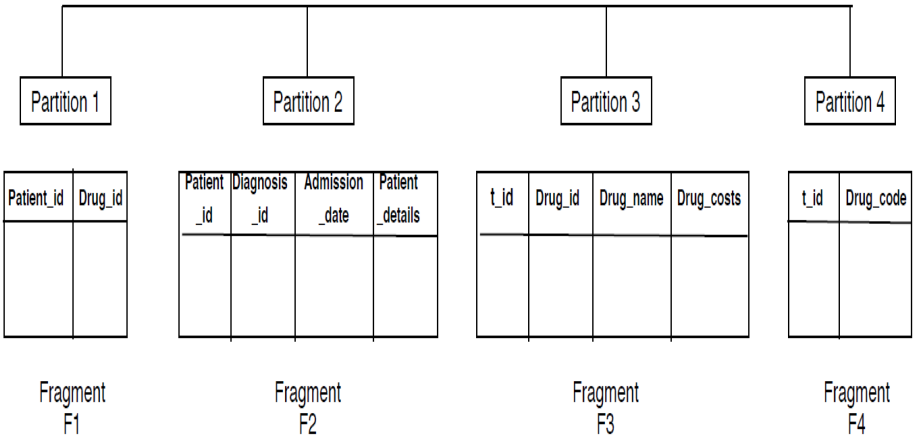

Figure 5.2: A refined partitioning of the Hospital-db schema w.r.t C, V, $BC = 2$ and $K = 4$

5.3 Query Evaluation Model

In this phase, we propose a monitor module to mediate every query issued from users against data stored in distributed partitions. The monitor module contains two mechanisms : a Role-Based Access Control mechanism and History-Based Access Control mechanism. The first mechanism checks the user role who issued the query and if this latter is not granted to execute distributed queries, then his query will be forwarded directly to the desired partition. Otherwise, the query is forwarded to the History Access Control mechanism which takes as input a set of violating transactions to be prohibited and checks if the cumulative of user past

queries and current query could complete a violating transaction. If it is the case, the query is revoked.

We have developed our querying model, as shown in figure 5.3, based on a Spark architecture where data is manipulated by users through DataFrames since this latter support all common relational operators. We use Spark SQL for relational processing because it provides high performance using established DBMS techniques and enables extension with advanced analytics algorithms such as graph processing and machine learning (10). Our querying model is composed of the following components:

- Monitoring module: It is composed of two sub modules: A Role-Based Access Control module (RBAC) implemented using Apache Sentry, and a History-Based Access control module (HBAC). The monitoring module mediates all user queries and enforces a runtime approach which consists in monitoring the execution of queries and revokes those queries that could lead to the violation of access control policies.

- Master node: It contains the cluster manager and it is in charge of decomposing and forwarding user queries from the monitoring module to the workers.

- Worker nodes: They contain confidential data and they are responsible for relational queries processing.

Next, we will introduce our query evaluation model which is composed of the two following steps: *Violating transactions detection and query lock.*

5.3.1 Violating Transactions Detection

By exploiting the functional dependency graph from Figure 4.2, we introduce an algorithm for automatically enumerating the set of queries having the following property: When all the queries of a given set are combined then the combination of their results will produce a violating transaction T. These violating transactions occur using functional dependencies and considering the confidentiality constraints as queries that need to be forbidden. To cope with this problem, we firstly propose

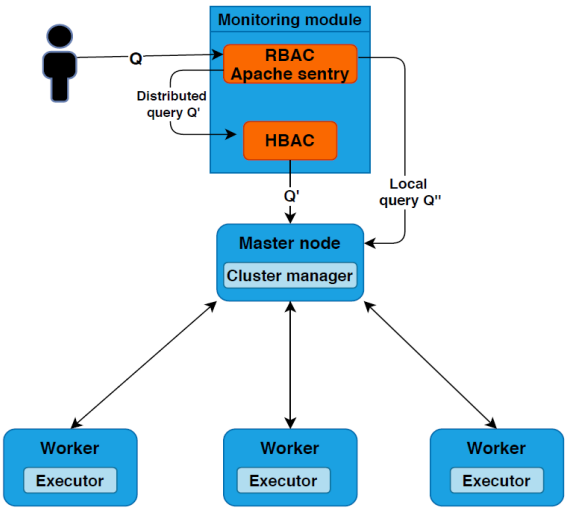

Figure 5.3: A query evaluation strategy augmented with a monitoring module

an algorithm to enumerate all violating transactions. Then, we propose a runtime approach to prohibit violating transaction completion.

It is important to note that to produce a violating transaction, we push the malicious user to perform a distributed query since we guarantee through our partitioning technique that the confidentiality constraint cannot be violated from a single partition.

Considering our example with confidentiality constraint $c_1 = \{Drug\_code, Patient\_details\}$. By running algorithm 7 on the functional dependency graph G(V,E) with join chains $\{JC_1, JC_2\}$, the following set of violating transactions is generated: $VT_s = \{T_1 = \{Q_r : \{Patient\_id, Drug\_id\}, Q_1 : \{Patient\_id, Patient\_details\}, Q_2 : \{Drug\_id, Drug\_code\}\}, T_2 = \{Q_r : \{Patient\_id, Drug\_id\}, Q_1 : \{Patient\_id, Diagnosis\_id, Patient\_details\}, Q_3 = \{Drug\_id, Drug\_code\}\}\}$.
Hence, it is clear that a malicious user could combine authorized queries with functional dependencies to evaluate all the queries of a violating transaction and by consequence violates the confidentiality constraint c_1.

Algorithm 7 VT_Track

Input: Functional dependency graph G(V,E); list of join chains JC_i; the set of confidentiality constraints C

Output: The set of violating transactions VT_s

Begin

$VT_s \leftarrow \varnothing$

for each join chain JC_i of c_j **do**

$T_n \leftarrow \varnothing$

Initialise the VT_Tracker to the root node R_t

▷ R_t is the common ancestor node of the attributes in G(V,E) representing the confidentiality constraint c_j

$Q_r \leftarrow R_t$ ▷ Q_r represents the query of the common ancestor node

$T_n = T_n \cup Q_r$

for each attribute $a_m \in JC_i \cap c_j$ **do**

$Q_m \leftarrow \varnothing$

Move the VT_Tracker to the next node until VT_Tracker $\leftarrow a_m$

At each move add VT_tracker to Q_m

$T_n = T_n \cup Q_m$

end for

$VT_s = VT_s \cup T_n$

end for

Return VT_s

End

5.3.2 Query lock

Our query lock technique heavily relies on Role-Based Access Control Model (RBAC) and History-Based Access Control Model (HBAC) (83). In RBAC, a set privileges are grouped into a role. Roles are created for the various job functions in an organization and users are assigned roles based on their responsibilities and qualifications. Hence, a user is assigned to this role by activating his session. HBAC consists in monitoring the access requests and revokes those that could lead to the violation of access control policies. The main idea of HBAC module

is the following: When a user launches an access request to the system, HBAC computes the cumulation of user past access and current access. If the cumulation can complete a violation of an access control policy, then the access request is revoked. Otherwise, the access is allowed.

In order to ensure maximal availability of data while ensuring the non disclosure of sensitive information, we propose a runtime approach which consists in monitoring the execution of queries and revokes those queries that could lead to the violation of policies. Under the assumption that malicious users cannot collude to produce a violating transaction, the main idea of the monitoring module is the following: When a user launches a query to the system, this query is automatically mediated by a Role Based Access Control mechanism. Then, the user role label is verified, if the label indicates that the user is not granted to execute distributed queries (the user label is equal to L_{rl}), the query will be forwarded to the master node for relational processing. Otherwise, if the user label is equal to L_{rd} then the query is forwarded from the Role Based Access Control mechanism to the History Based Access Control mechanism which takes as input the set of violating transactions to be prohibited and computes the cumulative of user past queries and current query. If the cumulative can complete a violating transaction then the query is locked, elseways, the query is forwarded to the master node. Algorithm 8 describes how the monitoring model processes user queries.

5.4 Conclusion

In this chapter, we have introduced our partitioning technique based on hypergraph theory to compute the set of secure partitions while retaining access control policies. Also, by resorting to RBAC and HBAC, we were able to build a monitoring module aiming to survey user query at runtime and deny those aiming to derive a violating transaction. The next chapter will be dedicated to experimental study.

Algorithm 8 Query Lock

Input: Current user query Q_i, Current user role r, the set of violating transactions VT_s and user query cumuli log UQ_{log}
Output: Query_state; locked or allowed

Begin
Query_state ← *locked*
for each issued query Q_i **do**
  if Label $r(Q_i) = L_{rl}$ **then**
    Query_state ← *allowed*
  else
    for each $T_n \in VT_s$ **do**
      if $Q_i \cup UQ_{log} < T_n$ **then**
        Query_state ← *allowed*
        Update UQ_{log}
      else
        Query_state ← *locked*
      end if
    end for
  end if
end for
Return Query_state
End

Chapter 6

Experimental Study

6.1 Introduction

In this chapter, we will demonstrate the applicability of our proposed methodology through a set of experiments. In order to demonstrate the performance and the scalability of our approach as well as the impact of the access control rules, optimality constraints, and the query evaluation model, we conducted an evaluation with different configurations.

6.2 Experimental Design

We provide an experimental study of our proposed approach and analysis of the proposed algorithms. We have developed a prototype of our methodology written in Java : An inference control module, a partitioning module and a module for the access control based on user history. Apache sentry 2.1.0 was used to enforce the access control based on user role. Apache Hive was used to manage relational data in distributed storage using SQL. To evaluate the queries execution on a big data framework, we have used Cloudera Entreprise VM 5.16.1 with Apache Spark 2. This evaluation was compared to another running MySQL DBMS. Experiments were conducted on an eight core PC (Inter(R) Core(TM) i5-8CPUs≈1.8GHZ) running CentOs 6.7 with 12 GB of RAM.

Spark SQL

Spark SQL is a Spark module for structured data processing. Unlike the basic Spark RDD API, the interfaces provided by Spark SQL provide Spark with more information about the structure of both the data and the computation being performed. uses this extra information to perform extra optimizations. There are several ways to interact with Spark SQL including SQL and the Dataset API. One use of Spark SQL is to execute SQL queries. Spark SQL can also be used to read data from an existing Hive installation.

Apache Hive

The Apache Hive data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Structure can be projected into data already in storage. A command line tool and JDBC driver are

provided to connect users to Hive.

Apache Sentry

Apache Sentry is a system for enforcing fine grained role based authorization to data and metadata stored on a Hadoop cluster. Particularly, Apache Sentry can be easily integrated to Cloudera and can manage fine grained role based authorization to data stored in Hive. Apache Sentry is now a Top-Level Apache project.

For our benchmarking, we have used the relational database schema *retail_db* (downloaded from the following link : [RT]). The relational dataset was converted to csv and produced the following files:

- customers.csv: containing 12435 records
- order_items: containing 172198 records
- orders: containing 68884 records
- products.csv: containing 1345 records
- categories.csv: containing 58 records
- departments.csv: containing 6 records

Figure 6.1 depicts how the proposed approach could be deployed on a cloud based system. We can see from this figure that a part of our approach is deployed on organization side and the other part is deployed on cloud level. In the organization side we found the user terminal and the monitoring module. Every query posed by the user is mediated by the monitoring module. Then, based on the user role and history the monitoring module decides whether the query is revoked or forwarded through the public network. In cloud level, we found the big data cluster responsible for relational query processing. The main advantage of our approach is that there is no need of an access control mechanism in the cloud. In fact, owner's access control polices are already enforced through the generation of secure subschemes hosted in different partitions.

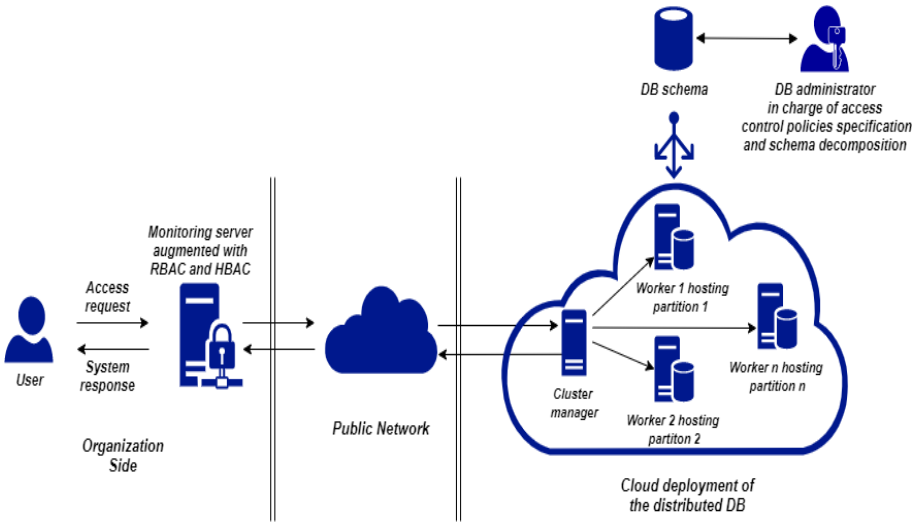

Figure 6.1: Deployment of the proposed approach on a cloud service

6.3 Evaluation

6.3.1 Functional Dependencies Impact on Constraints-Based Inference Control Generation

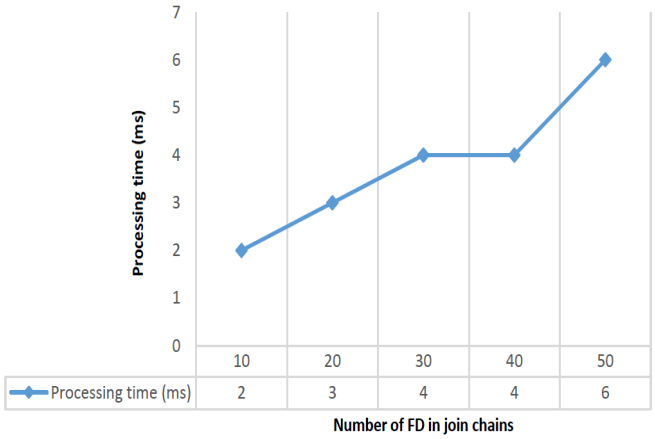

Figure 6.2: Impact of functional dependencies on the required timings to generate the constraint-based inference control set

Figure 6.2 shows a number of trials we performed so that for each run we considered a random number (ranging from 10 to 50) of functional dependencies of

the form $X \longrightarrow Y$. We observed that the running time is proportional to the number of considered functional dependencies. This can be explained by the fact that the expansion of the join chain path in the functional dependency graph will push the algorithm to take more time to identify the relaxed_cut. For a scenario where we could have 50 functional dependencies per join chain, the algorithm will take 6 ms to generate the constraint-based inference control set which we considered a reasonable timing. Despite that the algorithm shows promising results, we saw in some tests that, when the number of functional dependencies is greater than 100 (which is probably hard to find in real word scenarios), the algorithm takes an exponential behavior. This will be investigated in our future works.

6.3.2 Impact of the Number of Attributes on the Partitioning Algorithm

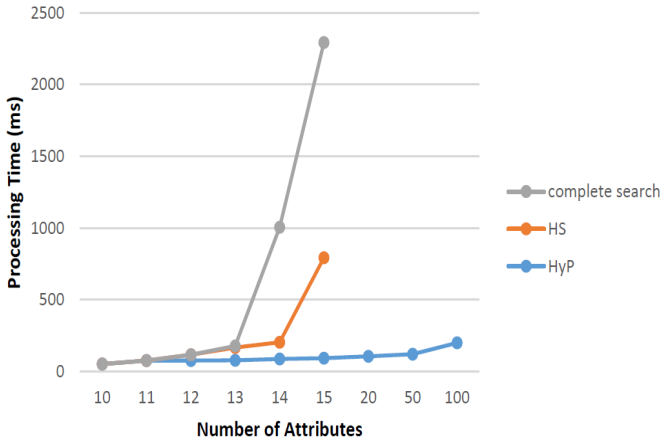

Figure 6.3: Impact of the number of attributes on the partitioning algorithm

In figure 6.3, we depict the impact of the variation of the number of attributes on the partitioning algorithms. We compare our algorithm *(HyP)* to other state of the art algorithms (Heuristic search HS (34) and complete search (37)). We observed that when the number of attributes exceeds a threshold (≈15 attributes), HS and complete search algorithms take an exponential behavior until they became unfeasible. By contrast, the execution time of our algorithm always remains low, for 100 attributes it takes 200ms and this guarantees its applicability to large

relational schemas since our algorithm will be performed offline only one time by the database designer to generate the distributed database schema.

6.3.3 Impact of the Variation of Confidentiality Constraints and Visibility Constraints on the Partitioning Algorithm

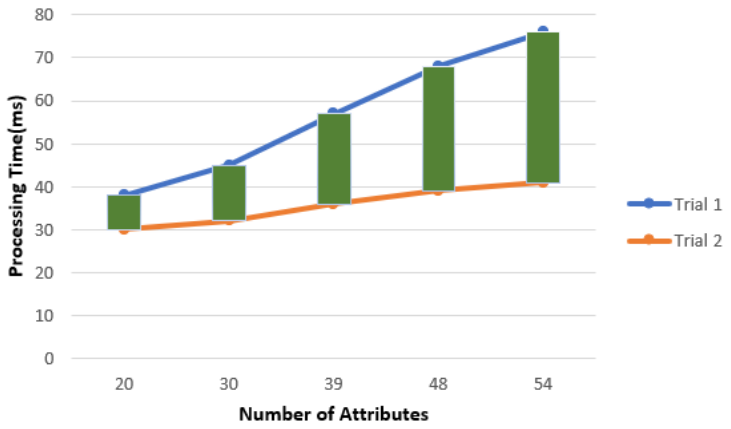

Figure 6.4: Computational time varying the number of confidentiality constraints and visibility constraints

In Figure 6.4, we fixed the number of attributes at each run and we varied the number of confidentiality constraints in Trial 1, and the number of visibility constraints in Trial 2 as shown in table 6.1. After examining the execution time of our partitioning algorithm, we saw that the increase of the number of confidentiality constraints for the same number of attributes and the same number of visibility constraints has importantly increased the execution time due to the fact that a large number of confidentiality constraints will likely result in highly fragmented schema. However, increasing the number of visibility constraints while fixing the number of confidentiality constraints has slightly affected the execution time of the partitioning algorithm.

Table 6.1: number of confidentiality constraints C and visibility constraints V for each run

Trial 1	C=2,V=6	C=4,V=6	C=7,V=6	C=10,V=6	C=12,V=6
Trial 2	C=6,V=2	C=6,V=4	C=6,V=7	C=6,V=10	C=6,V=12

6.3.4 Comparison of Query Execution Time Between MySQL and SparkSQL

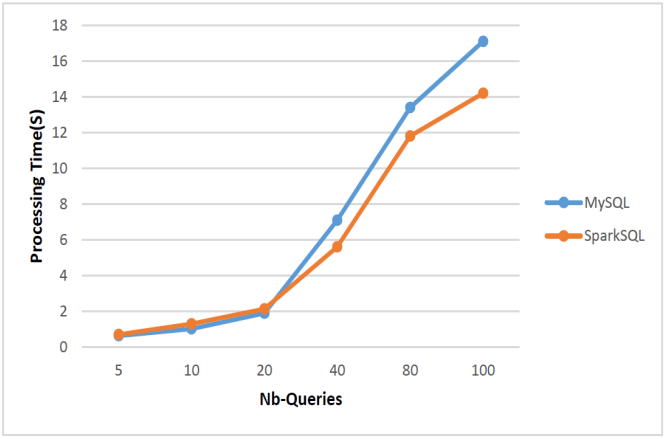

Figure 6.5: Comparison of query execution time between MySQL and SparkSQL

We compared in Figure 6.5 the execution time between MySQL database and SparkSQL for local and join queries . The database was firstly setup in a centralized server running MySQL, and a first trial was performed in which we had computed execution time. The number of queries was ranged from 5 to 100 queries. In the second trial, we had partitioned the database vertically into 6 partitions according to our proposed algorithms (we considered 6 visibility constraints and 12 confidentiality constraints) and we deployed it on a Spark cluster running 5 worker nodes. As illustrated in Figure 6.5, when the number of executed queries is less than 40 the MySQL database shows good performance than SparkSQL, but when the number of distributed queries becomes large SparkSQL shows less execution time than MySQL DBMS. The optimization of local queries was reached through the maximization of data locality adopted in our partitioning technique while the optimization of join queries was reached through the parallelization of

query execution.

6.3.5 Time Required to Lock a Suspicious Query

In Figure 6.6, we performed several runs. For each run, we fixed the number of issued queries from the user and we evaluated the required time to block a suspicious query. We note that the time required to block a suspicious query increase with the number of queries issued from the user. Observed results confirmed that our monitoring module took a reasonable timing to enforce both access control mechanisms RBAC and HBAC. However, dealing with a very large number of queries is an open issue which we will investigate in the future.

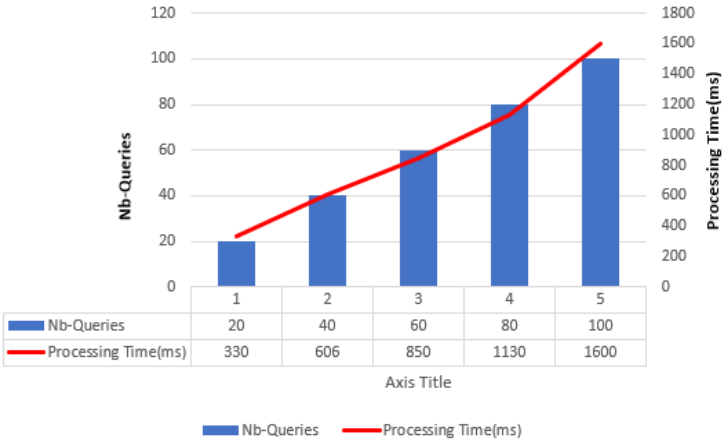

Figure 6.6: Time required to lock a suspicious query

The conducted experiments have showed the practicability of our methodology. Nevertheless, to ensure the usability of the approach, the following parameters should be taken into consideration:

- Number of functional dependencies: The proposed methodology showed some limitation only when it deals with a large number of functional dependencies. We believe that finding such typical scenario with large number of functional dependencies is probably hard.

- Value of the threshold α: Indeed, it is very important to choose an adequate value of α for the simple reason that this parameter direct affects the visi-

bility constraints which in turn affect the partitioning process. Some studies have been performed to determine optimal threshold according to the research domain. For example, in bioinformatics, the work in (17) attempted to derive an optimal threshold for interpreting semantic similarity and particularity.

- Since we store every generated partition in exactly one server, the number of allocated workers in the cloud should be equal to the number of generated partitions K.
- The relational schema given as an input should be static in all different steps involved in our methodology. Any change brought to the database schema will lead to execute the approach from scratch.

6.4 Complexity Study

In this section, we will discuss the complexity of the proposed algorithms used to achieve the goals of our approach. First we begin algorithm 1. To study the complexity of algorithm 1 let's consider the following notations used in this algorithm:

- $|R|$: The number of roles in the database.
- $|A|$: The cadinality of attribute set of the relational schema *R(A)*.

Hence the complexity of algorithm 1 "Generation of visibility constraints" would be $O(|R| * |A|)$ which gives us a linear complexity.

In algorithm 3 we consider the following parameters:

- $|A|$: The cadinality of attribute set of the relational schema *R(A)*.
- $|E|$: The number of edges generated in the functional dependency graph.
- $|SharedArr|$: size of the array of shared vertices by all Target arrows.
- $|C|$: The cardinality of the confidentiality constraints set.

- $|JC|$: The number of generated join chains.

we identify in this algorithm two instruction blocks: block(9-19) and block(24-30).

- The study of block(9-19) gives that this block is performed in $O(|A| * |C|)$.
- The complexity of the instruction block(24-30) is evaluated to $O(|JC^2|)$

Hence, the complexity of algorithm 3 "Join chain detection" would be: $O((|A| * |C|) + (|E|) + (|JC^2|) + (|SharedArr|))$.

To study the complexity of algorithm 4 "Detecting relaxed cut" we consider the following notations:

- $|C|$: The cardinality of the confidentiality constraints set.
- $|E|$: The number of edges generated in the functional dependency graph.
- $|JC|$: The set of all join chains sets
- $|JC_k|$: The set of join chains of c_k

we identify in this algorithm three instruction blocks: block(3-6), block(9-17) and block(20-29). The complexity of the block(3-6) is evaluated to $O(|C|)$. The block(9-17) is performed in $O((|E|) * (|JC|) * (|JC_K|))$ and the block(20-29) is performed also in $O((|E|) * (|JC|) * (|JC_K|))$. Finally, the complexity of algorithm 4 "Detecting relaxed cut" is $O((|C|) + 2 * ((|E|) * (|JC|) * (|JC_K|)))$. Algorithms 3 and 4 have a polynomial complexity which heavly depends from the number of generated edges in the functional dependency graph. we observed that when the number of edges is large the algorithms take an exponential behavior, but we think that finding such typical scenarios in real cases is probably hard.

The complexity of algorithm 6 "Computing K-balanced partitions" heavly depends on the number of considered confidentiality constraints. It shows a reasonable execution time compared to Heuristic search HS (34) and complete search (37). However, it shows an exponential behavior and becomes unfeasible only when it deals with a large number of confidentiality constraints. For algorithm 8 we define the parameter $|Q_i|$ which represents the number of issued queries from

the system user, and $|VT_s|$ the number of violating transactions. Thus, the complexity of algorithm 8 "Query lock" is evaluated to $O((|Q_i|) * (|VT_s|))$ which gives a linear complexity. However, the parameter $|Q_i|$ is dynamic and in the worst cases it becomes very large which will directly affects the algorithm complexity. This is why we considered it as an open issue to be investigated in the future.

6.5 Conclusion

In this chapter, we have analyzed the performance of our approach by conducting a set of experiments using different configurations. We have implemented our approach based on a Big Data system. The experimental evaluation of our proposed approach has presented a prominent results that could be more enhanced in future work.

Conclusion and Perspectives

Summary

In this Thesis, we have considered the problem of secure data outsourcing in presence of the inference problem. We have focused on the illicit inferences, which result from combining semantic constraints with, authorized information to infer sensitive data. We have started by investigating different research fields relevant to our problem, namely, access control models, secure data outsourcing, the inference problem and optimal database schema distribution. Based on this study, we have formally defined our problem. We have selected the full data outsourcing approach as a strategy of data outsourcing. Then, we have considered access control model through confidentiality constraints to specify access control rules. We have focused on considering semantic relatedness between user role and schema attributes to maximize data locality. We have also pinpointed the role of functional dependencies in allowing malicious users to infer sensitive information. Finally, we have developed a monitoring module to lock suspicious queries. In this context, our aim was to partition the database schema and generate a set of partitions to be hosted in CSSP level. Those partitions have minimized the number of distributed transactions by maximizing data locality, and preserving owners access control policies from being bypassed with direct access or indirect access via inference channels. Also, we have been able to securely evaluate user queries while retaining access control policies.

To achieve this goal, we have proposed a three-phase approach that can be

highlighted through the following contributions:

- We have first generated a set of visibility constraints that have been enforced as *soft constraints* in the partitioning process. Those constraints have aimed to maximise data locality by minimizing the number of distributed queries in the distributed database hosted in CSSP level. To generate them, we have performed a semantic analysis of the relational schema in order to measure semantic relatedness between attributes and users roles. Attributes that have a semantic relatedness to a role in the database greater than a threshold α were assigned to the visibility constraint attached to this role. These constraints were preserved when the relational schema was fragmented.

- To deal with inference leakage, we have generated a set of constraints-based inference control: These constraints are enforced as *hard constraints*. In this step, we have resorted to graph theory in order to build the functional dependencies graph and generate a set of join chains. A join chain was considered as an inference channel that enabled a malicious user to combine functional dependencies with the legitimate response received from the system to produce security breaches. Then, We have used a relaxed technique to cut the join chain only at a single point in order to minimize dependencies loss.

- By relying to hypergraph theory, we have firstly presented the partitioning problem as a hypergraph constraint satisfaction problem. Then, based on this hypergraph, we have reformulated the problem as a multi-objective function F to be minimized. Therefore, we have proposed a greedy algorithm to partition the constrained hypergraph into k partitions while minimizing the multi-objective function F. Our partitioning algorithm has demonstrated its performance compared to other state of the art algorithms.

- Regarding that our security model could be broken when a malicious user performed a join query between distributed partitions to regroup a confidentiality constraint, we have described the way to securely evaluate those queries at runtime. We have proposed a monitor module to mediate every query issued from users against data stored in distributed partitions. The

monitor module contained two access control mechanisms: A Role-Based Access Control mechanism and History-Based Access Control mechanism. The former mechanism was dedicated to check the user role who issued the query and if this latter was not granted to execute distributed queries, then his query was forwarded directly to the desired partition. Otherwise, the query was forwarded to the History Access Control mechanism which has taken as input a set of violating transactions to be prohibited and checked if the cumulative of user past queries and current query could complete a violating transaction. If it is the case, the query was denied.

Future Work

Since our approach is recent, there are a number of concepts associated to security policies, privacy, query workload and data dependencies which could be considered to ensure better security level and to enhance query processing in the cloud. Hence, there are many research directions to pursue:

- When the workload becomes available after the database is set up in the cloud, the challenge is how to dynamically reallocate the distributed database fragments among distributed partitions while retaining access control policies? Indeed, developing a dynamic reallocating algorithm by taking into consideration the query workload changes and confidentiality constraints will be a challenging task. We believe that machine learning could be a suitable technique in this situation towards an intelligent cloud database system.

- Another interesting issue is the collaborative inference. Indeed, we proposed to block a violating transaction from being achieved to prevent inference leakage, but what if these violating transactions results from a combination of a set of queries from more than one user?

- Furthermore, we aim in our future work to consider other semantic constraints as a source of inference leakage. So far, we have demonstrated

that functional dependencies are considered as the main sources of inference attacks. But other semantic constraints should be also considered: For example, inclusion dependencies, join dependencies and multivalued dependencies.

- Another interesting topic of discussion among security issues in data outsourcing is to preserve access control policies of the data owner when data is externalized to the cloud based on the mapping rules. In this situation, we cannot expect total conformity between two policies, and conflicts may occur. Indeed, we believe that the adoption of an access control model with authorization view and query rewriting technique seems to be adequate and prominent to deal with such issue.

- It is obvious that maintaining the record of user history in the HBAC mechanism without an optimal archiving policy has operational drawbacks and hight costs. Hence, a further interesting research challenge is the establishment of an optimal archiving policy to efficiently record user history in the HBAC mechanism. Indeed, we believe this will enhance the query runtime monitoring technique adopted in this thesis.

Bibliography

[1] Abadi, M., and Fournet, C. Access control based on execution history. In *NDSS* (2003), vol. 3, pp. 107–121.

[2] Abiteboul, S., Hull, R., and Vianu, V. *Foundations of databases*, vol. 8. Addison-Wesley Reading, 1995.

[3] Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., and Xu, Y. Two can keep a secret: A distributed architecture for secure database services. *CIDR 2005* (2005).

[4] Akeel, F., Fathabadi, A. S., Paci, F., Gravell, A., and Wills, G. Formal modelling of data integration systems security policies. *Data Science and Engineering 1*, 3 (2016), 139–148.

[5] Akeel, F. Y., Wills, G. B., and Gravell, A. M. Exposing data leakage in data integration systems. In *The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014)* (2014), IEEE, pp. 420–425.

[6] Al-Kahtani, M. A., and Sandhu, R. Rule-based rbac with negative authorization. In *20th Annual Computer Security Applications Conference* (2004), IEEE, pp. 405–415.

[7] Al-Sayid, N. A., and Aldlaeen, D. Database security threats: A survey study. In *2013 5th International Conference on Computer Science and Information Technology* (2013), IEEE, pp. 60–64.

[8] Alsirhani, A., Bodorik, P., and Sampalli, S. Improving database security in cloud computing by fragmentation of data. In *2017 International Conference on Computer and Applications (ICCA)* (2017), IEEE, pp. 43–49.

[9] An, X., Jutla, D., and Cercone, N. Dynamic inference control in privacy preference enforcement. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services* (2006), pp. 1–10.

[10] Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A., et al. Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data* (2015), pp. 1383–1394.

[11] Bahloul, S. N., Coquery, E., and Hacid, M.-S. Securing materialized views: a rewriting-based approach. *29emes Journées BDA* (2013), 1–25.

[12] Bailey, J., and Stuckey, P. J. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *International Workshop on Practical Aspects of Declarative Languages* (2005), Springer, pp. 174–186.

[13] Bertino, E. Data protection from insider threats. *Synthesis Lectures on Data Management 4*, 4 (2012), 1–91.

[14] Bertino, E. Data security–challenges and research opportunities. In *Workshop on Secure Data Management* (2013), Springer, pp. 9–13.

[15] Bertino, E., Ferrari, E., and Atluri, V. The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security (TISSEC) 2*, 1 (1999), 65–104.

[16] Bertino, E., Ghinita, G., and Kamra, A. *Access control for databases: Concepts and systems*. Now Publishers Inc, 2011.

[17] Bettembourg, C., Diot, C., and Dameron, O. Optimal threshold determination for interpreting semantic similarity and particularity: application to the comparison of gene sets and metabolic pathways using go and chebi. *PloS one 10*, 7 (2015), e0133579.

[18] Biskup, J., Embley, D. W., and Lochner, J.-H. Reducing inference control to access control for normalized database schemas. *Information Processing Letters 106*, 1 (2008), 8–12.

[19] Biskup, J., Hartmann, S., Link, S., and Lochner, J.-H. Efficient inference control for open relational queries. In *IFIP Annual Conference on Data and Applications Security and Privacy* (2010), Springer, pp. 162–176.

[20] Biskup, J., and Preuss, M. Database fragmentation with encryption: Under which semantic constraints and a priori knowledge can two keep a secret? In *IFIP Annual Conference on Data and Applications Security and Privacy* (2013), Springer, pp. 17–32.

[21] Biskup, J., Preuss, M., and Wiese, L. On the inference-proofness of database fragmentation satisfying confidentiality constraints. In *International Conference on Information Security* (2011), Springer, pp. 246–261.

[22] Bkakria, A., Cuppens, F., Cuppens-Boulahia, N., and Fernandez, J. M. Confidentiality-preserving query execution of fragmented outsourced data. In *Information and Communication Technology-EurAsia Conference* (2013), Springer, pp. 426–440.

[23] Bkakria, A., Cuppens, F., Cuppens-Boulahia, N., Fernandez, J. M., and Gross-Amblard, D. Preserving multi-relational outsourced databases confidentiality using fragmentation and encryption. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl. 4*, 2 (2013), 39–62.

[24] Bleiholder, J., and Naumann, F. Data fusion. *ACM computing surveys (CSUR) 41*, 1 (2009), 1–41.

[25] Bobrov, N., Chernishev, G., and Novikov, B. Workload-independent data-driven vertical partitioning. In *European Conference on Advances in Databases and Information Systems* (2017), Springer, pp. 275–284.

[26] Bollwein, F., and Wiese, L. Separation of duties for multiple relations in cloud databases as an optimization problem. In *Proceedings of the 21st International Database Engineering & Applications Symposium* (2017), pp. 98–107.

[27] Bollwein, F., and Wiese, L. On the hardness of separation of duties problems for cloud databases. In *International Conference on Trust and Privacy in Digital Business* (2018), Springer, pp. 23–38.

[28] Bretto, A. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer* (2013).

[29] Brodsky, A., Farkas, C., and Jajodia, S. Secure databases: Constraints, inference channels, and monitoring disclosures. *IEEE Transactions on Knowledge and Data Engineering 12*, 6 (2000), 900–919.

[30] Chang, L., and Moskowitz, I. A study of inference problems in distributed databases. In *Research Directions in Data and Applications Security*. Springer, 2003, pp. 191–204.

[31] Chen, Y., and Chu, W. W. Database security protection via inference detection. In *International Conference on Intelligence and Security Informatics* (2006), Springer, pp. 452–458.

[32] Chen, Y., and Chu, W. W. Protection of database security via collaborative inference detection. In *Intelligence and security informatics*. Springer, 2008, pp. 275–303.

[33] Ciriani, V., Di Vimercati, S. D. C., Foresti, S., Jajodia, S., Paraboschi, S., and Samarati, P. Fragmentation and encryption to enforce privacy in data

storage. In *European symposium on research in computer security* (2007), Springer, pp. 171–186.

[34] Ciriani, V., di Vimercati, S. D. C., Foresti, S., Jajodia, S., Paraboschi, S., and Samarati, P. Fragmentation design for efficient query execution over sensitive distributed databases. In *2009 29th IEEE International Conference on Distributed Computing Systems* (2009), IEEE, pp. 32–39.

[35] Ciriani, V., Di Vimercati, S. D. C., Foresti, S., Jajodia, S., Paraboschi, S., and Samarati, P. Keep a few: Outsourcing data while maintaining confidentiality. In *European Symposium on Research in Computer Security* (2009), Springer, pp. 440–455.

[36] Ciriani, V., di Vimercati, S. D. C., Foresti, S., Livraga, G., and Samarati, P. Enforcing confidentiality and data visibility constraints: An obdd approach. In *IFIP Annual Conference on Data and Applications Security and Privacy* (2011), Springer, pp. 44–59.

[37] Ciriani, V., Vimercati, S. D. C. D., Foresti, S., Jajodia, S., Paraboschi, S., and Samarati, P. Combining fragmentation and encryption to protect privacy in data storage. *ACM Transactions on Information and System Security (TISSEC) 13*, 3 (2010), 1–33.

[38] Curino, C., Jones, E., Zhang, Y., and Madden, S. Schism: A workload-driven approach to database replication and partitioning. *Proc. VLDB Endow. 3*, 1–2 (Sept. 2010), 48–57.

[39] Curino, C., Jones, E. P., Popa, R. A., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., and Zeldovich, N. Relational cloud: A database-as-a-service for the cloud.

[40] Cuzzocrea, A., Hacid, M.-S., and Grillo, N. Effectively and efficiently selecting access control rules on materialized views over relational databases. In *Proceedings of the Fourteenth International Database Engineering & Applications Symposium* (2010), pp. 225–235.

[41] De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., and Samarati, P. Fragments and loose associations: Respecting privacy in data publishing. *Proceedings of the VLDB Endowment 3*, 1-2 (2010), 1370–1381.

[42] De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., and Samarati, P. Efficient integrity checks for join queries in the cloud 1. *Journal of Computer Security 24*, 3 (2016), 347–378.

[43] de Mantaras, R. L., and Saina, L. Inference attacks in peer-to-peer homogeneous distributed data mining. In *ECAI 2004: 16th European Conference on Artificial Intelligence, August 22-27, 2004, Valencia, Spain: Including Prestigious Applicants [sic] of Intelligent Systems (PAIS 2004): Proceedings* (2004), vol. 110, IOS Press, p. 450.

[44] Delugach, H. S., and Hinke, T. H. Wizard: A database inference analysis and detection system. *IEEE Transactions on Knowledge and Data Engineering 8*, 1 (1996), 56–66.

[45] di Vimercati, S. D. C., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., and Samarati, P. Fragmentation in presence of data dependencies. *IEEE Transactions on Dependable and Secure Computing 11*, 6 (2014), 510–523.

[46] di Vimercati, S. D. C., Foresti, S., Jajodia, S., Paraboschi, S., and Samarati, P. Integrity for join queries in the cloud. *IEEE Transactions on Cloud Computing 1*, 2 (2013), 187–200.

[47] Domingo-Ferrer, J. Advances in inference control in statistical databases: An overview. *Inference Control in Statistical Databases* (2002), 1–7.

[48] Fagin, R., Kolaitis, P. G., and Popa, L. Data exchange: getting to the core. *ACM Transactions on Database Systems (TODS) 30*, 1 (2005), 174–210.

[49] Farkas, C., and Jajodia, S. The inference problem: a survey. *ACM SIGKDD Explorations Newsletter 4*, 2 (2002), 6–11.

[50] Fernández-Medina, E., Trujillo, J., Villarroel, R., and Piattini, M. Developing secure data warehouses with a uml extension. *Information Systems 32*, 6 (2007), 826–856.

[51] Ferraiolo, D., Atluri, V., and Gavrila, S. The policy machine: A novel architecture and framework for access control policy specification and enforcement. *Journal of Systems Architecture 57*, 4 (2011), 412–424.

[52] Gertz, M., and Jajodia, S. *Handbook of database security: applications and trends*. Springer Science & Business Media, 2007.

[53] Griffiths, P. P., and Wade, B. W. An authorization mechanism for a relational database system. *ACM Transactions on Database Systems (TODS) 1*, 3 (1976), 242–255.

[54] Gross, J. L., and Yellen, J. *Graph theory and its applications*. CRC press, 2005.

[55] Guarnieri, M., Marinovic, S., and Basin, D. Securing databases from probabilistic inference. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)* (2017), IEEE, pp. 343–359.

[56] Haddad, M., Hacid, M.-S., and Laurini, R. Data integration in presence of authorization policies. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications* (2012), IEEE, pp. 92–99.

[57] Haddad, M., Stevovic, J., Chiasera, A., Velegrakis, Y., and Hacid, M.-S. Access control for data integration in presence of data dependencies. In *International Conference on Database Systems for Advanced Applications* (2014), Springer, pp. 203–217.

[58] Hinke, T. H., and Delugach, H. S. Aerie: An inference modeling and detection approach for databases. In *Sixth Working Conference on DATABASE SECURITY* (1992), p. 187.

[59] Hinke, T. H., Delugach, H. S., and Wolf, R. P. Protecting databases from inference attacks. *Computers & Security 16*, 8 (1997), 687–708.

[60] Huang, J., Nicol, D. M., Bobba, R., and Huh, J. H. A framework integrating attribute-based policies into role-based access control. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies* (2012), pp. 187–196.

[61] Hudic, A., Islam, S., Kieseberg, P., and Weippl, E. R. Data confidentiality using fragmentation in cloud computing. *Int. J. Communication Networks and Distributed Systems 1*, 3/4 (2012), 1.

[62] Jbali, A., and Sassi, S. Access control policies for relational databases in data exchange process. In *International Conference on Database and Expert Systems Applications* (2017), Springer, pp. 264–271.

[63] Jebali, A., Jemai, A., and Sassi, S. A survey study on the inference problem in distributed environment (s). In *SEKE* (2019), pp. 113–152.

[64] Jebali, A., Sassi, S., and Jemai, A. Inference control in distributed environment: A comparison study. In *International Conference on Risks and Security of Internet and Systems* (2019), Springer, pp. 69–83.

[65] Jebali, A., Sassi, S., and Jemai, A. Secure data outsourcing in presence of the inference problem: issues and directions. *Journal of Information and Telecommunication* (2020), 1–19.

[66] Jebali, A., Sassi, S., Jemai, A., and Chbeir, R. Secure data outsourcing in presence of the inference problem: A graph-based approach. *Journal of Parallel and Distributed Computing 160* (2022), 1–15.

[67] Kantarcioglu, M., and Clifton, C. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE transactions on knowledge and data engineering 16*, 9 (2004), 1026–1037.

[68] Katos, V., Vrakas, D., and Katsaros, P. A framework for access control with inference constraints. In *2011 IEEE 35th Annual Computer Software and Applications Conference* (2011), IEEE, pp. 289–297.

[69] Kumar, K. A., Quamar, A., Deshpande, A., and Khuller, S. Sword: workload-aware data placement and replica selection for cloud data management systems. *The VLDB Journal 23*, 6 (2014), 845–870.

[70] Lenzerini, M. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (2002), pp. 233–246.

[71] Lesk, M. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation* (1986), pp. 24–26.

[72] Li, F., Li, Z., Han, W., Wu, T., Chen, L., and Guo, Y. Cyberspace-oriented access control: Model and policies. In *2017 IEEE Second International Conference on Data Science in Cyberspace (DSC)* (2017), IEEE, pp. 261–266.

[73] Liu, Y., McInnes, B. T., Pedersen, T., Melton-Meaux, G., and Pakhomov, S. Semantic relatedness study using second order co-occurrence vectors computed from biomedical corpora, umls and wordnet. In *Proceedings of the 2nd ACM SIGHIT international health informatics symposium* (2012), pp. 363–372.

[74] Mell, P., Grance, T., et al. The nist definition of cloud computing.

[75] Morgenstern, M. Controlling logical inference in multilevel database systems. In *Proceedings. 1988 IEEE Symposium on Security and Privacy* (1988), IEEE Computer Society, pp. 245–245.

[76] Nait-Bahloul, S., Coquery, E., and Hacid, M.-S. Authorization policies for materialized views. In *IFIP International Information Security Conference* (2012), Springer, pp. 525–530.

[77] Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E., and Abramov, J. Security issues in nosql databases. In *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications* (2011), IEEE, pp. 541–547.

[78] Özsu, M. T., and Valduriez, P. *Principles of distributed database systems*, vol. 2. Springer, 1999.

[79] Qian, X., Stickel, M. E., Karp, P. D., Lunt, T. F., and Garvey, T. D. Detection and elimination of inference channels in multilevel relational database systems. In *Proceedings 1993 IEEE Computer Society Symposium on Research in Security and Privacy* (1993), IEEE, pp. 196–205.

[80] Rath, S., Jones, D., Hale, J., and Shenoi, S. A tool for inference detection and knowledge discovery in databases. In *Database security IX*. Springer, 1996, pp. 317–332.

[81] Rizvi, S., Mendelzon, A., Sudarshan, S., and Roy, P. Extending query rewriting techniques for fine-grained access control. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data* (2004), pp. 551–562.

[82] Rosenthal, A., and Sciore, E. View security as the basis for data warehouse security. In *DMDW* (2000), p. 8.

[83] Samarati, P., and de Vimercati, S. C. Access control: Policies, models, and mechanisms. In *International School on Foundations of Security Analysis and Design* (2000), Springer, pp. 137–196.

[84] Samarati, P., and Di Vimercati, S. D. C. Data protection in outsourcing scenarios: Issues and directions. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security* (2010), pp. 1–14.

[85] Sayah, T., Coquery, E., Thion, R., and Hacid, M.-S. Inference leakage detection for authorization policies over rdf data. In *IFIP Annual Conference*

on Data and Applications Security and Privacy (2015), Springer, pp. 346–361.

[86] Sellami, M., Gammoudi, M. M., and Hacid, M. S. Secure data integration: a formal concept analysis based approach. In *International Conference on Database and Expert Systems Applications* (2014), Springer, pp. 326–333.

[87] Sellami, M., Hacid, M.-S., and Gammoudi, M. M. Inference control in data integration systems. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (2015), Springer, pp. 285–302.

[88] Soler, E., Trujillo, J., Blanco Bueno, C., Fernández-Medina Patón, E., et al. Designing secure data warehouses by using mda and qvt.

[89] Staddon, J. Dynamic inference control. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery* (2003), pp. 94–100.

[90] Su, T.-A., and Ozsoyoglu, G. Controlling fd and mvd inferences in multilevel relational database systems. *IEEE Transactions on Knowledge and Data Engineering 3*, 4 (1991), 474–485.

[91] Thuraisingham, B. Handling security constraints during multilevel database design. *Burns, R.(ed.) Research Directions zn Database Securt (v, IV, Mitre Technical report, M92B0000 118, Mitre Corp., McLean, Va* (1992).

[92] Thuraisingham, B. *Database and applications security: Integrating information security and data management*. CRC Press, 2005.

[93] Thuraisingham, B. Database security: Past, present, and future. In *2015 IEEE International Congress on Big Data* (2015), IEEE, pp. 772–774.

[94] Thuraisingham, B., Ford, W., Collins, M., and O'Keeffe, J. Design and implementation of a database inference controller. *Data & knowledge engineering 11*, 3 (1993), 271–297.

[95] Thuraisingham, M. Security checking in relational database management systems augmented with inference engines. *Computers & Security 6*, 6 (1987), 479–492.

[96] Toland, T. S., Farkas, C., and Eastman, C. M. The inference problem: Maintaining maximal availability in the presence of database updates. *Computers & Security 29*, 1 (2010), 88–103.

[97] Tracy, J., Chang, L., and Moskowitz, I. S. An agent-based approach to inference prevention in distributed database systems. *International Journal on Artificial Intelligence Tools 12*, 03 (2003), 297–313.

[98] Turan, U., Toroslu, İ. H., and Kantarcioğlu, M. Secure logical schema and decomposition algorithm for proactive context dependent attribute based inference control. *Data & Knowledge Engineering 111* (2017), 1–21.

[99] Turan, U., Toroslu, I. H., and Kantarcioglu, M. Graph based proactive secure decomposition algorithm for context dependent attribute based inference control problem. *arXiv preprint arXiv:1803.00497* (2018).

[100] Wang, J., Yang, J., Guo, F., and Min, H. Resist the database intrusion caused by functional dependency. In *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)* (2017), IEEE, pp. 54–57.

[101] Wang, L., Wijesekera, D., and Jajodia, S. A logic-based framework for attribute based access control. In *Proceedings of the 2004 ACM workshop on Formal methods in security engineering* (2004), pp. 45–55.

[102] Widom, J., and Ceri, S. *Active database systems: Triggers and rules for advanced database processing*. Morgan Kaufmann, 1996.

[103] Wiese, L. Clustering-based fragmentation and data replication for flexible query answering in distributed databases. *Journal of Cloud Computing 3*, 1 (2014), 1–15.

[104] Wiese, L. Ontology-driven data partitioning and recovery for flexible query answering. In *Database and Expert Systems Applications* (2015), Springer, pp. 177–191.

[105] Xu, X., Xiong, L., and Liu, J. Database fragmentation with confidentiality constraints: A graph search approach. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy* (2015), pp. 263–270.

[106] Yang, Y., Li, Y., and Deng, R. H. New paradigm of inference control with trusted computing. In *IFIP Annual Conference on Data and Applications Security and Privacy* (2007), Springer, pp. 243–258.

[107] Yip, R. W., and Levitt, E. Data level inference detection in database systems. In *Proceedings. 11th IEEE Computer Security Foundations Workshop (Cat. No. 98TB100238)* (1998), IEEE, pp. 179–189.