SECURE DATABASES: CONSTRAINTS AND INFERENCE CHANNELS

by

Csilla Farkas

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
the Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

____________________ Sushil Jajodia, Dissertation Director

____________________ Alexander Brodsky, Dissertation Director

____________________ Paul Ammann

____________________ Gheorghe Tecuci

____________________ Stephen G. Nash, Associate Dean for Graduate Studies and Research

____________________ Lloyd J. Griffiths, Dean, School of Information Technology and Engineering

Date: ____________________ Summer Semester 2000
George Mason University
Fairfax, Virginia

Secure Databases: Constraints and Inference Channels

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University.

By

Csilla Farkas
Bachelor of Science, Geology, Eotvos Lorand University, Hungary, 1985
Master of Science, Computer Science, George Mason University, Virginia, 1993

Directors: Dr. Sushil Jajodia, Professor
Dr. Alexander Brodsky, Associate Professor
Information and Software Engineering

Summer Semester 2000
George Mason University
Fairfax, Virginia

DEDICATION

To my Parents, Anna Nagy and Ignac Farkas

ACKNOWLEDGMENTS

I am deeply grateful to my advisors Professor Sushil Jajodia and Professor Alexander Brodsky, whose suggestions and advice guided me to important results and helped focus my ideas. Their encouragement aided me through the rough periods of my research. I would like to thank Professor Jajodia for teaching me to appreciate high quality research and motivating me to strive for it. I have always admired Professor Jajodia's enthusiasm and his high personal expectations and hope that I will be able follow in his footsteps. Finally, I would like to thank him for understanding and supporting my obligations towards my family.

I am especially grateful to Professor Brodsky's continuous support throughout my graduate studies. His insightful comments played an important role in shaping the direction of my research while his excellent technical expertise helped me reach my research goals. Moreover, Professor Brodsky encouraged me to trust my abilities and go for my dreams by applying for a position in academia.

During my studies at George Mason University I have met most of the faculty at Information and Software Engineering and Computer Science Departments. I am very grateful to all of them for their positive comments and encouragements. I am thankful to my committee members Professor Gheorghe Tecuci and Professor Paul Ammann for their valuable suggestions, encouragement and support. Special thanks go to professors Sean Wang, Larry Kerschberg, Duminda Wijesekara, Peter Denning and Anne Marchant for aiding me to become a better scholar.

I also grateful to my present and former friends and colleagues as well as visitors at the Center for Secure Information Systems at George Mason University - Indrajit, Indrakshi, Jia-Ling, Liu Peng, Ningning, Ning Peng, Xiantao, Sencun Zhu and Yingjiu.

Finally, I am grateful to my husband Robert Mattick for his patience, encouragement, support and for all his help at home. I thank my children Viki and Edward for their understanding during the times when I was not around and when I was too tired or nervous to be the mother they deserve.

TABLE OF CONTENTS

ABSTRACT

SECURE DATABASES: CONSTRAINTS AND INFERENCE CHANNELS

Csilla Farkas, Ph.D.

George Mason University, 2000

Dissertation Directors: Dr. Sushil Jajodia and Dr. Alexander Brodsky

Information security policies in databases aim to protect the *confidentiality* and the *integrity* of data, while ensuring data *availability*. Direct violations of data confidentiality are usually prevented by mandatory access control mechanisms, such as those based on the Bell-LaPadula model. However, even in the presence of a properly functioning mandatory access control mechanism, illegal data accesses by inference channels may occur when constraints are combined with non-sensitive data to infer sensitive data.

This research investigates the problem of detecting and removing *inference channels*. Previous approaches to the problem of inference channels were based on over-classifying data, which led to unnecessary restrictions on the availability of data to legitimate users. This work is the first to introduce the idea of characterizing inference disclosure algorithms by the formal logic properties of *soundness* and *completeness*. Intuitively, soundness means that everything that is generated by the algorithm is disclosed, thus providing maximal data availability; completeness means that everything that can be disclosed is produced by the algorithm, thus guaranteeing confidentiality. Without such characterization we cannot formally compare disclosure

inference algorithms and establish their properties with respect to confidentiality and data availability.

The technical core of this dissertation concentrates on the development of sound and complete *disclosure inference algorithms* for a variety of settings (relational, semi-structured and numeric databases) and constraints (functional, multi-valued and join dependencies, Horn-clause constraints, arithmetic constraints). The developed algorithms can be used in either database design or query processing time.

In summary, the primary contribution of this work to the field of database security is two fold: (1) it provides a much needed *representation model* of the inference channel problem that allows a security officer to formally assure protection against illegal inferences by the properties of completeness and soundness; (2) provides sound and complete *disclosure inference algorithms* to generate all the information that can be disclosed by a user.

CHAPTER 1

Introduction

With the increased use of electronic information systems in every level of our life - individual, commercial, governmental - it is important to protect sensitive information against unauthorized accesses. In traditional databases, direct violations of data confidentiality are prevented by mechanisms based on discretionary (DAC) and/or mandatory (MAC) access control policies. However, these policies do not completely guarantee confidentiality. Illegal data accesses via *inference channels* may occur even when properly functioning access control mechanisms are present.[*] An inference channel in a database is a means by which sensitive information can be inferred from non-sensitive data and some meta-data. The inference channel problem is the problem of detecting and removing inference channels.

1.1 Problem Statement and Thesis

Problem Statement

Most of the illegal inferences in databases occur when integrity constraints, such as functional, multi-valued, and join dependencies, are combined with non-sensitive data to disclose sensitive information. Consider the following example of an *inference channel via functional dependency*. Let *(NAME, RANK, SALARY, EX-*

[*]Covert channels are outside the scope of this research.

NAME	RANK	SALARY ($)	EXPERIENCE (years)
Brown	Clerk	34,000	5
Evan	Clerk	34,000	3
Hammer	Director	65,000	3
Joels	Clerk	34,000	3
Smith	Accountant	41,000	6
Small	Secretary	28,000	8

Table 1.1: *Employee* relation

RANK	SALARY
Clerk	34,000
Director	65,000

$\Pi_{RANK,SALARY}\sigma_{EXPERIENCE=3}(Employee)$

NAME	RANK
Brown	Clerk

$\Pi_{NAME,RANK}\sigma_{EXPERIENCE=5}(Employee)$

Table 1.2: Query answers

PERIENCE) be a relation schema, and assume that the *Employee* relation (Table 1.1) over this schema satisfies the functional dependency $RANK \rightarrow SALARY$. That is, for any two tuples of the *Employee* relations, if they have the same value for *RANK*, they must also have a same value for *SALARY*. The security requirement is that only authorized users (e.g., payroll office workers) can access information about the salaries of employees. However, to support data availability, unauthorized users are permitted to *separately* access data values for *NAME* and *SALARY*.

Suppose that an access control mechanism is implemented to enforce the security requirement, thus unauthorized users cannot access partial tuples over attributes *NAME* and *SALARY*. Consider the scenario when an unauthorized user requests the following two queries:

- "List the *RANK* and *SALARY* of all employees with 3 years of experience."
- "List the *NAME* and *RANK* of all employees with 5 years of experience."

Both of these queries are permitted because they do not access partial tuples over attributes $NAME$ and $SALARY$. But, because the *Employee* relation satisfies the functional dependency $RANK \rightarrow SALARY$, their answers (Table 1.2) discloses the fact that Brown's salary is \$ 34,000. In this example an inference channel was raised by combining non-confidential data (answers to the user's queries) with meta-data (functional dependency) to infer confidential information (Brown's salary).

Thesis

In this dissertation we present a novel approach to detect and remove inference channels which are raised by combining non-confidential data with constraints. We *formally define* the inference channel problem using concepts of mathematical logic and introduce measurements of assurance against illegal disclosure based on the *soundness* and *completeness* properties of disclosure inference algorithms. Intuitively, soundness means that everything that is generated by the algorithm is disclosed, thus providing maximal data availability; completeness means that everything that can be disclosed is produced by the algorithm, thus guaranteeing confidentiality.

We define the notions of *data-dependent* and *data-independent* disclosure inferences. Data-dependent disclosure inference is established using actual data items and database constraints. Data-independent disclosure inference is established using only queries and database constraints. Both data-dependent and data-independent disclosure guarantee confidentiality (completeness property). Data-dependent disclosure is computationally expensive but allows maximal data availability (soundness property). Data-independent disclosure is computationally less expensive than data-dependent disclosure, but may result in over-classification of data.

Moreover, we develop sound and complete *disclosure inference algorithms* for a variety of settings (relational, semi-structured and numeric databases), constraints (functional, multi-valued and join dependencies, Horn-clause constraints, arithmetic constraints) and settings (query-time, database design-time).

Detailed descriptions of the main contributions in this dissertation – with regards to inference channels in multilevel secure relational databases, in semi-structured databases and in numeric databases – are given in the following section.

1.2 Research Contributions

The contributions of this research can be grouped into three separate but related groups.

Inference channels in multilevel secure relational databases

The research results of this section are published in [BFJar, BFJ00]. We present an integrated security mechanism, called the *Disclosure Monitor* (DiMon) (see Figure 1.1), which guarantees data confidentiality by extending a mandatory access control mechanism (MAC) with the *Disclosure Inference Engine* (DiIE). After a user's query is received, MAC enforces the standard security policy by allowing the user to read only the data that is dominated by the user's security clearance. If a security violation is detected, i.e. the requested data has higher security classification than the user's, the query is rejected. If direct security violation is not detected, the query is submitted to DiIE for further evaluation. DiIE generates all the information that can be disclosed by the user, based on the user's past and current requests and the database constraints. Then, the disclosed information is returned to MAC to be reevaluated. If no security violation is detected at this point, the query is answered and the user's *history-file* is updated.

The fundamental notions of *data-dependent* and *data-independent* disclosures

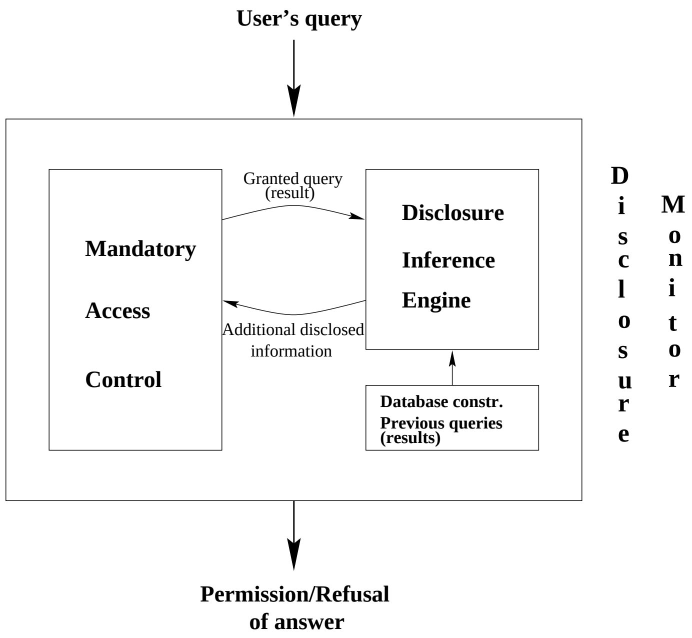

Figure 1.1: Disclosure Monitor

are introduced and studied. While motivated by DiMon architecture, the developed notions of disclosure and their inferences are orthogonal to specific security architectures, and, in fact, are applicable to a wide variety of both design and query-time security frameworks. We develop a Disclosure Inference Engine that operates in two modes: in *data-dependent* mode, the actual answers $P_1, \ldots, P_n$ to the user's past and present queries $Q_1, \ldots, Q_n$ and the constraints $\mathcal{D}$ are considered to generate the disclosed facts; in *data-independent* mode, only the past and present queries $Q_1, \ldots, Q_n$ (without the actual data items) and the constraints $\mathcal{D}$ are used to generate new queries which are disclosed. Data-independent disclosure is computationally less expensive and requires less storage overhead than the data-dependent disclosure, but it may indicate disclosures which are not present in a given database. Without knowing the actual data, however, data-independent disclosure is the best one can achieve.

This work is the first, to the best of our knowledge, that introduces a formal classification of disclosure inference algorithms by the properties of *completeness* and *soundness*. The *completeness* property ensures that the algorithm generates all of the disclosed information, thus no possible inference remains undetected. The algorithm is *sound* if the generated information (actual data in data-dependent mode and queries in data-independent mode) is indeed disclosed. Intuitively, soundness along with completeness, provides the highest data availability without violating security. For the data-dependent disclosure, we develop a disclosure inference algorithm that, we prove, is both sound and complete.

For the data-independent disclosure, we provide a sound and complete disclosure algorithm for the case when the selection conditions in queries and the constraints do not contain constants and a general disclosure inference algorithm that works without these restrictions and is guaranteed to be complete.

Inference channels in semi-structured databases

The research results of this section are published in [BFJed]. We develop a framework and algorithmic solutions for preventing privacy violations via illegal data inferences that occur when domain knowledge is combined with non-private data. We propose a formal model, called *Privacy Information Flow (PIF) Model*, to represent how information is transferred among participants (e.g., people and organizations) as well as to specify the privacy requirements. The privacy information flow model consists of two main layers: a *Privacy Information Flow (PIF) Graph* and a *Privacy Contract*. The PIF graph creates a requirement that no data, beyond what is described in the PIF graph, is allowed to be transferred. A privacy contract specifies the information that is not allowed to be transferred to, or logically entailed by, the participants.

We develop a privacy architecture, called *Privacy Mediator* (PriMe), that enforces the privacy requirements on every data transfer while provides maximal data availability. PriMe extends a *standard access control mechanism* with an *inference engine*. All data transfers are performed via PriMe which guarantees that no data transfer is completed if it violates the privacy requirements described in the PIF graph and the privacy contracts. Even if the disallowed data is not directly transferred, but can be logically entailed from the domain knowledge and the accumulated and transferred data, PriMe would still disallow the transfer.

We develop a *sound* and *complete inference algorithm* used by the inference engine, which is the core of PriMe. The inference engine generates all disclosed information from the current and previously disclosed data, and the domain knowledge. We consider a tree-like semi-structured database model, similar to [BDHS96], which is suitable to describe information from heterogeneous data sources, such as federated databases, the World Wide Web, as well as traditional relational databases.

Inference channels in numeric databases

The research results of this section are published in [BFWW00]. We consider inference channels that arise by combining arithmetic constraints with non-confidential data to infer confidential data. We adopt a simple data model that is a set of variables. A database is an instantiation of values to appropriate variables such that the instantiation satisfies the database (arithmetic) constraints. The *database* can be viewed as a single *point in $n + k$-dimensional space*, where n is the number of data items stored in the database (extensional database) and k is the number of values that can be derived from the stored data and additional knowledge (intentional database). The *database constraints* over extensional and intentional data form an $n + k$-dimensional *constraint object*. When a user requests a data value a range of possible values – represented as two new database constraints – are generated and returned to the user.

The *security requirements* on data (both extensional and intentional) are expressed by the accuracy to which users are allowed to know data values. To our best knowledge , this is the first paper that introduces a security model in which classification of data is based on the precision of numeric answers and thus is *continuous.* We propose two methods to assign security classification to data items:

- The entire *volume* of the constraint object that contains the data item is considered.
- The size of the *interval* that contains the data item is considered.

For both methods we develop algorithms to generate query answers that are *correct* (i.e., the data item is within the returned interval) and *safe* (i.e., the user is unable to infer which point within the interval is the actual value of the data item).

1.3 Limitations of the Related Works

Research on detection and removal inference channels in relational databases can be organized into two categories. The first category includes techniques that detect inference channels *during database design*; any channels are removed by modifying the database design and/or by increasing the classification levels of some of the data items [Buc90, GM84, Hin88, Mea90, TDC95, Mar96, Mor88, Smi90, SO91, DdVS99a, DdVS99b]. These techniques often result in *over-classification of data* and, therefore, reduce the availability of data. Techniques in the second category seek to eliminate inference channel violations *during query processing time* [Den85, MSS88, ST90, Thu87]. If an inference channel is detected, the query is either refused or modified to avoid security violations.

Each category requires either data-dependent or data-independent *inference* *algorithms*. However, none of the above works has the formal notion of soundness and completeness for data-dependent and data-independent disclosure, and thus cannot establish these formal properties of disclosure inference. Also, most authors [Buc90, GM84, Mor88, Smi90, MSS88, ST90, Thu87], with the exception of [Den85, Hin88, SO91, DdVS99a, DdVS99b], do not consider the problem of actual inference for specific families of constraints (and its decidability, soundness, completeness, etc.); rather they develop a framework, assuming that disclosure inference algorithms are readily available. It is our view, that the main technical difficulty of solving the inference channel problem lies in developing (sound and complete) inference algorithms, especially for the data-independent case.

The work of Su and Ozsoyoglu [SO91] is the closest to our research in that they consider inference algorithms for specific constraint families, namely functional and multi-valued dependencies. However, their inference methods are limited in several respects. First, as the source of illegal inferences, they consider only two types of dependencies, functional (FD) and multi-valued (MVD) dependencies; inference algorithms are given for each of the two types separately, but not when they are present together. Whereas, our methods apply to Horn-clause constraints which are more expressive than the combination of FDs and MVDs. Second, they use single-attribute security granularity for FD-compromise and tuple-level security granularity for MVD-compromise. In contrast, we propose a flexible security classification schema that can also express content- and context-based security granularity, including selection-projection queries and arbitrary sets of (partial) tuples. While their inference algorithms seem to be sound and complete for single-attribute level security granularity, it would not be sound even for security granularity based on sets of attributes.

Since Su and Ozsoyoglu's solution for eliminating a detected inference channel is based on increasing the classification level of the data items, it restricts the availability of data. To illustrate this shortcoming, consider the introductory inference channel example. In their similar example, Su and Ozsoyoglu classify the attribute *SALARY* as top-secret. Then, by using the FD $RANK \rightarrow SALARY$ and the assumption that the mapping associated with the FD is known by the users, they detect an inference channel and raise the classification level of attribute *RANK* to top-secret in order to remove the inference channel. Unfortunately, this solution results in rejecting many legitimate queries of users with secret or lower security classifications. For example, queries such as "List the years of experiences of the employees whose rank is *Clerk*" will be refused although they do not violate the security requirement.

Our solution is more flexible than the one recommended by Su and Ozsoyoglu. It classifies only the pair $<NAME,SALARY>$ as top-secret, and evaluates each user's queries to determine if they illegally disclose any of these pairs. The mappings, which are associated with the database dependencies and assumed to be known by all of the users, are represented as database projections released to the users. For example, the mapping associated with $RANK \rightarrow SALARY$ is represented as $\Pi_{RANK\ SALARY}$, where $\Pi_{RANK\ SALARY}$ denotes the projection of the active relation on attributes *RANK SALARY*. Each new query is analyzed to determine if it discloses a $<NAME,SALARY>$ pair based on the answer to the current and the previous queries, the known database projections, and the database constraints. For example, if a user with secret clearance already received the answer to the query $\Pi_{RANK\ SALARY}(Employee)$ and requests the new query $\Pi_{NAME,\ RANK}(Employee)$, it will be refused, thus eliminating illegal data access via the inference channel. However, the query $\Pi_{RANK,EXPERIENCE}(Employee)$ is permitted since the answer to this query, along with the FD and the answers to the first query, does not disclose any top-secret information.

Also, the scheme recommended by Su and Ozsoyoglu is database design-time oriented rather than query-time oriented, as in the case of DiMon. While design-time approach is easier to manage and implement, query-time approach allows more availability of data than design-time approach because more information (i.e., past and current data/queries) can be used for disclosure inference. Our disclosure inference algorithms are readily available to both design- and query-time schemes (of cause, only data-independent disclosure is relevant to the design-time approach.)

During the 1990s, with the rapid development of interconnected databases and the Internet, the problem of *protecting computerized private data* have surfaced (e.g., [Cou97, Mee97, O'L95, RGR97, Ste98, WBD98]). However, most of these works considering privacy protection are limited in two respects: (1) they focus mainly on preventing privacy violations via *direct data accesses*, thus illegal inferences that lead to the disclosure of private information are still possible; (2) the proposed solutions are based on the *relational data model*, which is unsuitable for web-like information or heterogeneous, distributed data sources. Tree-like, semi-structured databases have been proven successful to accommodate these applications. To the best of our knowledge, there is no published report that considers the privacy problem via inference disclosure in semi-structured databases.

Finally, all of the previous research to protect against illegal inferences are based on withholding answers from the user. In certain applications that involve numeric values confidentiality can be achieved by releasing a range of possible data values but not the actual values. The difficulty lies in (1) defining how to assign *security classification* (i.e., accuracy) to the data items and (2) how to generate query answers that are *correct* (i.e., the actual data value is within the released range) and *safe* (i.e., the user does not no which value within the released range is the actual data value). To the best of our knowledge, there is no published report that considers the inference problem at the presence of continues classification schema.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows. In Section 2 we give an overview of the related work. Section 3 contains our model to detect and remove inference channels in multilevel secure relational databases. This section introduces the notions of data-dependent and data-independent disclosures; and the soundness and completeness properties of disclosure inference algorithms. Disclosure inference algorithms to generated data-dependent and data-independent disclosure are given. Inference channels that lead to privacy violations in semi-structured databases are studied in Section 4. We present our Privacy Information Flow graph to represent the privacy requirements and Privacy Mediator architecture to enforce these requirements. The unique problem of defining precision based security classification for numeric data is presented in Section 5. Algorithms to generate correct and safe query answers are developed. Finally, we conclude in Section 6 with a summary of research contributions and future research directions. The Appendix contains the proofs of the theorems and propositions of the dissertation.

CHAPTER 2

Related Work

Inferences in Statistical Databases

The inference channel problem first was considered in statistical databases. The security requirement in statistical databases is to provide access to statistics about groups of individual but restrict access to information about any particular individual. The problem is that a user might obtain confidential information about an individual by correlating different statistics.

Inference control in statistical databases have been considered since the early 70s in both statistical and database community. Denning [Den82] gives a comprehensive overview of the inference attacks and the protection mechanisms developed. She states that *simple inference control mechanisms*, like query-set size control, query overlap control, are *insufficient* because they can be subverted easily. Some of the *mechanisms which provides security with high assurance* (e.g., data swapping, partitioning) are *not practical* in general-purpose databases (i.e., which provides both statistical and general purpose access) and has to be limited to certain applications. Denning mentions the possibility of *introducing noise* as a promising direction to provide security in statistical databases.

One of the earliest security mechanisms to protect data privacy is based on controlling statistics about small query sets. *Query-set-size control* mechanism restricts query set having less than n or more than $N - n$ records (N is the total number of records). Unfortunately, this control is easy to subvert. Denning et. al [DDS79, DS80] and Schlörer [Sch80] describe simple snooping tools, called *trackers* that pad small query sets with enough records to put them into allowable ranges. By subtracting out the effect of the padding, the restricted (small query size) statistics is obtained. They show that trackers can be constructed for most statistical databases and it is usually easy to find a tracker, therefore simple query-set-size control mechanisms do not provide adequate database security.

Kam and Ullman [KU77] propos a model for statistical databases and define compromisability condition at the presence of (s, k) queries, where $s \leq k$, k is a sequence of k bits used as a key of the database, and s is the number of bits specified in the query. Chin [Chi78] extends Kam and Ullman work to account non-existing records in the model for statistics about sums and averages. He defines a necessary and sufficient condition for compromisability as well as recommends a protection mechanism by *limiting the number of related queries.*

Dobkin et. al [DJL79] recommends a security mechanism for average and median statistics by controlling *overlaps between queries* and the number of permitted queries. They calculate the minimum number of queries needed for compromise, based on the total number of data elements in the database, a fixed query set size, the maximum overlap between queries, and the users' preliminary knowledge about the data. The security of the database is guaranteed by limiting the number of queries to be less than the minimum number needed for compromise.

Beck [Bec80] proposes a technique to increase the number of queries needed for compromise at the presence of total, average, count, and percentile statistics, thus achieve "effectively uncompromisable" security. His method is based on *perturbing each data value* then use this perturbed value to compute the desired statistics. Beck also describes a scheme for minimizing the error due to the perturbation in the resulting statistics.

Denning [Den80] introduces the *random sample queries* for on-line statistical databases. The security of data is protected by making it impossible for the user to control the formulation of the query sets. By using large samples of the the total number of the records, it is ensured that the released statistics is accurate.

Schlörer [Sch81] uses *multidimensional transformation* to achieve high assurance security while preserve $k - order$ statistics (i.e., some statistical quantity which can be computed from the values of exactly k attributes). The multi-dimensional transformation can be applied in two ways: (1) physically performed on the database and (2) only the $k - order$ statistics is transformed but not the physical records. However, finding a general transformation is NP-complete.

Chin and Ozsoyoglu [CO81] investigates the security problem at *conceptual data model* level. They (logically) *partition* the individuals represented in the database into population based on their common characteristics. Populations are hierarchically organized. Security constraints are used to determine whether a statistic should be released to the user. In addition, their design also maintains an audit mechanism to keep track of the data retrieved by the users and the change in the security requirements.

Inferences in Relational Databases

During the 1980s researchers started to investigate inference channels that occur in relational databases. Techniques to detect and eliminate inference channels in relational databases fall into one of two general categories [JM95]: (1) techniques which detect inference channels during database design and then modify the database design and/or increase the classification level of the data items to eliminate these inference channels; (2) techniques which evaluate database queries whether they lead to illegal inferences and then reject and/or modify the answers to the queries.

Works of Goguen et. al, Morgenstern et. al, Su et. al, Hinke, Smith, and Buczkowski belongs to the first type of inference control techniques. Goguen and Meseguer [GM84] are among the first researchers who considered the inference problem in relational databases. They formally define a noninterference condition which requires that all effects of a high user are hidden from lower and non-comparable level users.

Morgenstern [Mor88] extends Goguen and Messeguer's model by characterizing the inference problem in terms of information theory. He introduces the function $INFER(x \rightarrow y)$ that determines the amount of information that can be inferred about y if x is given.

$$INFER(x \rightarrow y) = [H(y) - H_x(y)]/H(y) if [H(y) - H_x(y)]/H(y) > \epsilon;$$
$$INFER(x \rightarrow y) = 0 \; otherwise$$

where $H(y)$ is the entropy of y and $H_x(y)$ is the conditional entropy of y if x is given. Morgenstern also defines the notion of *sphere of influence* as all data that can be inferred from a core set of data. The sphere of inference is defined by the $INFER$ function. Inference problem occurs if there exists a data item within the sphere of influence that has higher security classification than the highest classified data in the core. By backward chaining from the data outside the core, which causes the illegal inference, the inference channel can be determined and then eliminated. Morgenstern suggests the notion of *universe of disclosure* which incorporates the data not stored in the database, but can form the basis of illegal data disclosure.

Su and Ozsoyoglu [SO87] consider inferences via functional dependencies (FDs) and a single multi-valued dependency (MVD). They provide algorithms for raising the classification labels of attributes or tuples during database design in order to avoid inferences.

Hinke [Hin88] defines an inference detection tool to detect potential inferences during database design. He describes the construction of a semantic relationship graph where each data item is represented as a node and the edges between the nodes represent the relationship between the data items. He reduces the inference problem to the problem of finding alternative paths between the nodes such that a user is cleared to see all of the edges in one path but not all of the edges in the other path. If such alternative edges are found then the classifications of the edges should be adjusted (if possible) to eliminate the inference channel.

Smith [Smi90] further refines the semantic data model introduced by Hinke by allowing the user to express different types of relationships. The model allows to classify any of the data items or the relationship between them, thus provides a simple mechanism to eliminate a potential inference channel.

Buczkowski [Buc90] presents an expert system tool to detect and correct logical inferences during database design. He introduces the function $PINFER(x, y)$ which is very similar to the $INFER(x \rightarrow y)$ function defined by Morgenstern. The $PINFER$ function gives the probability that y can be inferred from x. A probabilistic inference network is built which gives the logical dependencies of classified data on unclassified aggregates. The expert is asked to determine the probability that y can be inferred from x ($PINFER(x, y)$) for certain pairs of facts x and y; then this probability is propagated through the inference network using fuzzy logic. These propagated probabilities estimate the degree of dependency between additional facts (e.g. what is the probability that z can be inferred from x).

The second type of inference control techniques were developed by Denning, Thuraisingham, Mazumdar, and Stachour. Denning [Den85] focuses on inference channels which arise when the evaluation of the query involves data for which the user does not have the proper authorization. She introduces the concept of *commutative filter* that guarantees that the result of a query is equivalent to one that would have been generated if the query was posed on an authorized view of the database. Denning's work, however, considers only the classifications of data used to evaluate the query; constraints are not considered.

Thuraisingham [Thu87] proposes a framework to modify a user's query if it would lead to illegal inferences. This method not only eliminates the possible inference channel but also supports the user to formulate correct queries. However, this technique also introduces the possibility of additional inferences by observing the way the query is modified.

Mazumdar et al. [MSS88] uses theorem prover to evaluate the security of a transaction with regard to a predefined set of secrets. Their evaluation depends on the integrity constraints of the database, the precondition of the transaction, and the input data type of the transaction. Since the technique does not depend on any specific data, it can be applied on the transaction at compilation time. However, their tool does not detect all of the possible inference channels resulting through transactions and therefore illegal data accesses are not fully protected.

Finally, a comprehensive analysis of the inference problem was incorporated in the design of the Lock Data Views (LDV) MLS database management system [ST90]. LDV is built on top of the LOgical Coprocessor Kernel (LOCK) Trusted Computing Base (TCB) which performs the basic (operating system oriented) security controls. The security policy of LDV extends the basic LOCK provided security for database systems by incorporating the semantics of data into the security policy. LDV defines five types of classification rules, including *inference control rules*, to properly assign classification levels of data within the DBMS context. To ensure secure release of data from the database, the classification level of the answer to a query is upgraded to the level of the information that can be derived from them in the given context. LDV defines the *consistency of the classification constraints* such that no security violation is possible if the constraints are consistent, meaning that all possible inferences should be considered.

Inferences and privacy protection

Recently, with the advent of electronic database systems, new concerns regarding privacy violations emerged. Researchers focusing on privacy violations [gen98, And97, BB89, Bra96, Cou97, DO98, Mee97, Off99, O'L95, Rin97, RGR97, Ste98, Tin98, WBD98, WBSQ96a, WBSQ96b] consider primarily direct privacy violations, such as unauthorized data access, eavesdropping. The recommend solutions, with the exceptions of [WBD98] and [DO98], are based on developing policies to address privacy issues as well as deploying traditional security mechanisms, like intrusion detection/prevention, authentication, access control, and cryptography, etc.

Wiederhold et. al [WBD98] recommends a security mediator to ensure data confidentiality in clinical databases. The purpose of their mechanism it to support information sharing while preserving data confidentiality. The security mediator, developed under the TIHI project, screens the queries and answers to the queries to detect privacy violations. The security requirements are encoded as rules and enforced by the mediator. The paper provides a general framework and the description of a prototype for enforcing domain specific privacy requirements.

Dreyer and Olivier [DO98] propose an information can-flow graph to represent positive (permitted) and negative (not permitted) information flow among the entities. They assume that users are not trustworthy and information flow between users should be taken into account, too. They distinguish between the static (potential) and dynamic (actual) aspects of the information flow. They present algorithms to extend the static can-flow graph to incorporate the dynamic aspect without introducing unauthorized information flow.

CHAPTER 3

Inference in Multilevel Secure Relational Databases

3.1 Problem Description

In Multilevel Secure (MLS) Relational Database Management Systems (RDBMSs) direct violations of data confidentiality are usually prevented by lattice-based (mandatory) access control mechanisms, such as those based on the Bell-LaPadula (BLP) model. Mandatory access control (MAC) is expressed via security classification labels, assigned to *subjects*, i.e., active computer system entities that can initiate requests for information, and to *objects*, i.e., passive computer system repositories that are used to store information. Classification labels, e.g., unclassified, confidential, secret, top-secret, form a mathematical lattice structure with a dominance relation among the labels. MAC policies control read and write operations on the data objects based on the classification labels of the requested data objects and the classification label (also called clearance) of the subject requesting the operation. For example, BLP policy ensures that a subject can read an object only if the subject's classification label dominates the object's classification label (simple-security property) and that a subject can write an object only if the object's classification label dominates the subject's classification label (star-property). However,illegal data accesses via *inference channels* may occur even if a properly functioning mandatory access control mechanism is present.

Database integrity constraints, such as functional, multi-valued and join dependencies, are especially important from the perspective of generating inference channels. Moreover, meta-data that can be expressed by general Horn-clause constraints can also generate inference channels. Consider again the example presented in the Introduction. If an unauthorized user requests the following two authorized queries on the *Employee* relation (Table 1.1)

- "List the *RANK* and *SALARY* of all employees with 3 years of experience."
- "List the *NAME* and *RANK* of all employees with 5 years of experience."

then applies the functional dependency $RANK \rightarrow SALARY$ on the answers, the user can disclose the confidential information that Brown's salary is \$ 34,000.

3.2 Disclosure Monitoring Architecture

3.2.1 Data, Queries, and Security Classification

Our model is built upon lattice-based access control, therefore, it is necessary to define what the objects and the subjects are and how to assign security classification labels to them. For simplicity, we assume that users and *subjects*, executing in behalf of these users, are the same entities, thus we use the terms user and subject interchangeably. To formally define *objects*, we need the definition of projection facts, queries, and their combinations, as given below.

We consider *selection-projection* queries of the form $\Pi_Y \sigma_C$, where Π_Y denotes the projection of a relation on attributes Y, and σ_C denotes the selection condition of the query, where C is a conjunction of equalities of the forms $A = B$ or $A = c$, such that A, B are attribute names and c is a constant. We assume the existence of a single, "universal" relation r with schema R. The symbol R is used to indicate both the relation name and the set of all its attributes.

In our model, it is possible to express complex security requirements dealing not only with content- and context-based security restrictions but different security granularity also, as demonstrated in the following example.

Example 1 Consider the following security requirements on relation *Employee* in Table 1.1:

1. Tuples with value *Smith* for $NAME$ are *top-secret* (element-level granularity).
2. *NAME* and *SALARY* are *top-secret* when retrieved together (context-based restriction).
3. *RANK* is *top-secret* when *SALARY* is $ 65,000 (content- and context-based restriction).

$\square$

Definition 1 (Projection fact)

A *projection fact* (PF) *of type* $A_{i_1}, \ldots, A_{i_k}$, where $A_{i_1}, \ldots, A_{i_k}$ are attributes in R, is a mapping m from $\{A_{i_1}, \ldots, A_{i_k}\}$ to $\bigcup_{j=1}^{k} dom(A_{i_j})$ such that $m(A_{i_j}) \in dom(A_{i_j})$, for all $1 \leq j \leq k$. We will denote a projection fact by an expression of the form

$$R[A_{i_1} = a_1, \ldots, A_{i_k} = a_k]$$

where R is the relation name and $a_1, \ldots, a_k$ are constants in $dom(A_{i_1}), \ldots, dom(A_{i_k})$ respectively. A relation instance r' over $R' \subseteq R$ can be viewed as a set of projection facts of type R'. *

Note, that since a projection fact is a mapping, the order of attributes $A_{i_1}, \ldots, A_{i_k}$ in $R[A_{i_1} = a_1, \ldots, A_{i_k} = a_k]$ is not important.

*Often, the name *partial tuples* is used in literature.

Definition 2 (Query-answer pair)

An *atomic query-answer pair* (QA-pair) is an expression of the form $(PF, \Pi_Y \sigma_C)$, where PF is a projection fact over Y that satisfies C. A query-answer pair is either an atomic QA-pair or an expression of the form $(P, \Pi_Y \sigma_C)$, where P is a set of projection facts $\{PF_1, \ldots, PF_l\}$ such that every projection fact PF_i, $(i = 1, \ldots, l)$ is over Y and satisfies C.

Projection facts and query-answer pairs are fundamental concepts of our model, we use the following example to illustrate them.

Example 2 Let *R=(NAME, RANK, SALARY,EXPERIENCE (years))* be a relation schema and let $\Pi_{NAME,SALARY}\sigma_{SALARY=34{,}000}$ be a selection-projection query. An example **projection fact** is

$$R[NAME = Brow, RANK = Clerk, SALARY = 34,000, EXPERIENCE = 5]$$

atomic QA-pair is

$$(R[NAME = Brow, SALARY = 34,000], \Pi_{NAME,SALARY}\sigma_{SALARY=34{,}000})$$

and **QA-pair** is

$$(\{R[NAME = Brow, SALARY = 34,000], R[NAME = Evan, SALARY = 34,000], R[NAME = Joels, SALARY = 34,000]\}, \Pi_{NAME,SALARY}\sigma_{SALARY=34{,}000})$$

□

QA-pairs are used as security labeled objects. Specifically, an *object* in our security model can have one of the following forms:

1. An atomic QA-pair $(PF, \Pi_Y \sigma_C)$.

2. A query of the form $\Pi_Y\sigma_C$. This includes Π_Y representing $\Pi_Y\sigma_{TRUE}$ and σ_C, representing $\Pi_R\sigma_C$.

3. A set of atomic QA-pairs. As a security labeled object, a QA-pair $(P, \Pi_Y\sigma_C)$ will be interpreted as the set $\{(PF, \Pi_Y\sigma_C) \mid PF \in P\}$ and a set of QA-pairs $\{(P_1, \Pi_{Y_1}\sigma_{C_1}), \ldots, (P_n, \Pi_{Y_n}\sigma_{C_n})\}$ will be interpreted as $\{(PF, \Pi_{Y_i}\sigma_{C_i}) \mid 1 \leq i \leq n,\ PF \in P_i\}$. We will also use the shortcut $\Pi_Y\sigma_C(r)$ for $(P, \Pi_Y\sigma_C)$, where $P = \Pi_Y\sigma_C(r)$, as well as $\Pi_Y(r)$ for $\Pi_Y\sigma_{TRUE}(r)$ and $\sigma_C(r)$ for $\Pi_R\sigma_C(r)$. Furthermore, in data-dependent mode $\Pi_Y\sigma_C$ will be interpreted as $\Pi_Y\sigma_C(r)$, where r is the current database relation; i.e., all QA-pairs of $\Pi_Y\sigma_C(r)$.

Definition 3 (Security classification)

Let SL be a set of security labels, e.g. unclassified, secret, top-secret, etc. A *security classification* is a triple $< \mathcal{O}, \mathcal{U}, \lambda >$, where

$\mathcal{O} = \{o_1, \ldots, o_n\}$ is a set of security objects

$\mathcal{U}$ is a set of subjects (users), and

$\lambda : \mathcal{O} \cup \mathcal{U} \rightarrow SL$ is a security classification mapping.

Since some of the security objects in O may be sets of atomic QA-pairs, we denote by $A(O')$, for a subset O' of $\mathcal{O}$, the set of all atomic objects that appear in O'; i.e.,

$$A(O') = \{o \mid o \in O' \textit{ or there is a } o' \in O' \textit{ such that } o \in o'\}$$

Example 3 The security requirements given in Example 1 can be represented as follows:

1. $\lambda((R[NAME = Smith], \Pi_{NAME})) = \textit{top-secret}$

2. $\lambda(\Pi_{NAME,SALARY}) = \textit{top-secret}$

3. $\lambda(\Pi_{RANK}\sigma_{SALARY=\$65,000}) = \textit{top-secret}$

$\square$

To ensure data confidentiality, the access control mechanism has to guarantee that users cannot access objects unless their security clearance is greater than or equal to the security classification of the objects; i.e., $\lambda(user) \geq \lambda(object)$. Because $\mathcal{O}$ does not necessarily contain all of the objects that may be requested by the user, we need to describe a mechanism that detects whether the objects, that are currently requested or previously received by the user, disclose objects in $\mathcal{O}$ for which the user is not authorized.

3.2.2 Disclosure Monitor

Disclosure Monitor (DiMon), presented next, is an enhancement of the standard mandatory access control mechanism with a Disclosure Inference Engine to protect against inference channels that result from database constraints (see Figure 1.1). Algorithm 1, given in Figure 3.1 describes the steps followed by DiMon. Note that the user's id, given to the algorithm as an input value, may refer to a particular user or to a group of users sharing the same security clearance. We illustrate the functionality of DiMon by the following example.

Example 4 Consider again the relation schema *(NAME, RANK, SALARY, EXPERIENCE)* and the relation *Employee* (see Table 1.1) that satisfies the functional dependency $RANK \rightarrow SALARY$. The security requirement is that only users with *top-secret* security clearances can access the salaries of the employees;

Algorithm 1:	Disclosure Monitor
INPUT	1. User's query (object) Q_i 2. User's id u 3. Security classification $< \mathcal{O}, \mathcal{U}, \lambda >$ 4. User's history-file: objects which were previously retrieved by the user (QA-pairs in data-dependent mode and queries in data-independent mode) 5. $\mathcal{D}$, a set of Horn-clause constraints
OUTPUT	Answer to Q_i and update of the user's history-file or refusal of Q_i
METHOD	MAC evaluates direct security violation: **if** there is a classified object $o \in \mathcal{O}$ such that $\lambda(u) \ngeq \lambda(o)$ **then** reject Q_i (note, direct security violation is detected, DiMon functions as the basic MAC mechanism) **else** (no direct security violation was detected) **begin** 1. Find $O' \subseteq \mathcal{O}$, the set of classified objects that cannot be returned to the user; i.e., $O' = \{o \in \mathcal{O} \mid \lambda(u) \ngeq \lambda(o)\}$. 2. Using *Disclosure Inference Engine* (DiIE), determine whether any of the (atomic) objects in $A(O')$ (i.e., disallowed objects) is disclosed from the user's history, the current query, and the constraints $\mathcal{D}$. 3. **if** disclosure is detected in (2) (i.e., security is violated) **then** reject Q_i **else** (security is not violated) answer Q_i and update the user's history-file. **end**

Figure 3.1: Disclosure Monitor Algorithm

i.e., $\lambda(\Pi_{NAME,SALARY})$ =*top-secret.* In the data-dependent mode, the relation *Employee* is used by DiIE.

Assume that a user u with secret clearance has already received the answer to the query $\Pi_{SALARY}\sigma_{RANK=Clerk}(Employee)$, which is the tuple $< 34,000 >$. When representing the information revealed to the user, in both data-dependent and data-independent mode, besides the actual data values that were returned to the user, we also represent information contained in the selection conditions of the queries. For example, while the previous query only returned the value for the *SALARY*, the user also knows that this is the salary of the employee who is a clerk. Therefore, the user's history file in the data-dependent mode contains the QA-pair

$$(R[RANK = Clerk, SALARY = 34,000], \Pi_{RANK,SALARY})$$

and in the data-independent mode the query

$$\Pi_{RANK,SALARY}\sigma_{RANK=Clerk}$$

Next, suppose that u requests the query

$$\Pi_{NAME,RANK}\sigma_{EXPERIENCE=6}(Employee).$$

According to our algorithm, DiMon first verifies if there is a direct security violation. Because tuples over $NAME$ and $RANK$ are classified as secret, there is no security violation detected. DiMon next checks for possible indirect security violations. First, a set O' is constructed.

- In the *data-dependent mode*

  $$O' = \{(\Pi_{NAME,SALARY}(Employee), \Pi_{NAME,SALARY})\}$$

  Thus the set of all atomic objects (QA-pairs) is

  $$A(O') = (\{R[NAME = Brown, SALARY = 34,000], \Pi_{NAME,SALARY}), \ldots,$$
  $$R[NAME = Small, SALARY = 28,000]\}, \Pi_{NAME,SALARY})$$

  Next, the answers to the current and previous queries and the FD are used to determine if any of the atomic objects in $A(O')$ are disclosed. Table 3.1 summarizes the answers to the first (first tuple) and to the second queries (second

NAME	RANK	SALARY	EXPERIENCE (years)
δ_1	Clerk	34,000	δ_3
Smith	Accountant	δ_4	6

Table 3.1: Answers to $\Pi_{SALARY}\sigma_{RANK=Clerk}$ and $\Pi_{NAME,RANK}\sigma_{EXPERIENCE=6}$

.

NAME	RANK	SALARY	EXPERIENCE (years)
δ_1	Clerk	34,000	δ_3
Smith	Clerk	δ_4	6

Table 3.2: Example of a data-dependent disclosure for the detected data-independent disclosure.

tuple). Note, that the δ_is represent unique null values; i.e., values that the user does not know. Using DiIE, it is determined that no atomic object in $A(O')$ is disclosed. Intuitively, this happens in the example because the two answered tuples have different values for attribute *RANK*, and, therefore they cannot be combined by using the FD.

- In the data-independent mode O' only contains the query $\Pi_{NAME,SALARY}$, thus $A(O') = O'$. Next, the user's previously answered queries, the current query, and the FD are submitted to DiIE. Here, it is determined that the object $\Pi_{NAME,SALARY}$ is disclosed from the queries $\Pi_{SALARY}\sigma_{RANK=Clerk}$, $\Pi_{NAME,RANK}\sigma_{EXPERIENCE=6}$, and the FD. Intuitively, without knowing the actual data, there exists an example (data) which constitutes a data-dependent disclosure. For example, if the employee Smith was a *clerk*, e.g., the fifth tuple of the *Employee* relation is $<$ *Smith, Clerk, 34,000, 6* $>$, then the answers to the two queries would be as given in Table 3.2. From these answers, by using the FD, users could infer that Smith's salary is \$34,000.

□

3.3 Database Constraints and Data Disclosure

In our model *database constraints* form the basis of data disclosures. In this paper we consider Horn-clause constraints which are defined as follows.

Definition 4 (Horn-clause constraint)

A *Horn-clause constraint* is an expression of the form

$$\forall x_1, \ldots, x_m \; (p_1 \wedge \ldots \wedge p_n \rightarrow q)$$

where $n \geq 1$ and $x_1, \ldots, x_m$ are all of the free variables in $p_1 \wedge \ldots \wedge p_n \rightarrow q$.

Each p_i is of the form

$$R[A_{i_1} = b_1, \ldots, A_{i_k} = b_k]$$

where b_i is either a variable or a constant, and each q has one of the following two forms:

1. $R[A_1 = a_1, \ldots, A_n = a_n]$ where $A_1, \ldots, A_n$ are all of the attributes of R (i.e., the constraint is full) and each a_i is either a constant or a variable that must appear in $p_1 \wedge \ldots \wedge p_n$. In this case, the constraint is called *tuple-generating*.

2. Equality of the form $a_i = a_j$, where each a_i is either a constant or a variable that must appear in $p_1 \wedge \ldots \wedge p_n$. In this case the constraint is called *equality-generating*.

We will refer to $p_1 \wedge \ldots \wedge p_n$ as the *body* and to q as the *head* of the constraint. We require, that all the variables of q appear in the body. We will use the shorthand $p_1 \wedge \ldots \wedge p_n \rightarrow q$ for Horn-clause constraints.

Horn-clause constraints extend generalized dependencies [Ull88] with constants, and can express functional, multi-valued, and join dependencies, as well as a wide variety of user defined knowledge.

Example 5 Let $R = (A, B, C, D)$ be a relation schema.

- *Functional dependency* $A \to B$ can be represented as:

$$R[A = a, B = b_1] \wedge R[A = a, B = b_2] \to b_1 = b_2$$

- *Multi-valued dependency* $MVD[A, B]$ can be written as:

$$R[A = a, B = b_1, C = c_1, D = d_1] \wedge R[A = a, B = b_2, C = c_2, D = d_2] \to$$

$$R[A = a, B = b_1, C = c_2, D = d_2]$$

- *Join dependency* $JD[AB, BC, CD]$ can be written as:

$$R[A = a, B = b] \wedge R[B = b, C = c] \wedge R[C = c, D = d] \to R[A = a, B = b, C = c, D = d]$$

□

Definition 5 (Truth value of the projection fact)
We say that a projection fact PF of type $A_{i_1}, \ldots A_{i_k}$ is *TRUE* with respect to a set of projection facts $\mathcal{P}$ if $\mathcal{P}$ contains a projection fact PF' with the same predicate symbol and the same values for attributes $A_{i_1}, \ldots, A_{i_k}$. (PF' might contains other attributes as well.)

Definition 6 (Constraint satisfaction)
A database constraint d is *satisfied* by a set $\mathcal{P}$ of projection facts if its formula is TRUE for $\mathcal{P}$. (The truth value is defined in the standard way.)

Since the relation r over R can be viewed as a set of projection facts of type R, we can also speak of d being satisfied by r. In the following, we restrict ourselves to constraints which can be represented in Horn-clause forms with equalities. We can now formally define the notion of data-dependent disclosure.

Definition 7 (Data-dependent disclosure)
Let $\mathcal{D}$ be a set of database constraints, $P_1, \ldots, P_n$ be sets of projection facts over attribute sets $X_1, \ldots, X_n$, and PF be a projection fact over Y. We say that the set of QA-pairs

$$\mathcal{P} = \{(P_1, \mathit{\Pi}_{X_1}\sigma_{C_1}), \ldots, (P_n, \mathit{\Pi}_{X_n}\sigma_{C_n})\}$$

discloses $(PF, \Pi_Y\sigma_C)$ in *data-dependent mode*, denoted as $\mathcal{P} \models_{\mathcal{D}} (PF, \mathit{\Pi}_Y\sigma_C)$, if for every r over R that satisfies $\mathcal{D}$,

$$P_i \subseteq \Pi_{X_i}\sigma_{C_i}(r) \;\; for \;\; all \;\; i = 1, \ldots, n$$

implies

$$PF \in \Pi_Y\sigma_C(r)$$

Given a set S of atomic QA-pairs, we say that $\mathcal{P} \models_{\mathcal{D}} S$ if for every (atomic) QA-pair $(PF, \Pi_Y\sigma_C) \in S$, $\mathcal{P} \models_{\mathcal{D}} (PF, \mathit{\Pi}_Y\sigma_C)$. Finally, we denote by $(PF, \Pi_Y\sigma_C) \models (PF', \Pi_{Y'}\sigma_{C'})$ the disclosure $\{(PF, \Pi_Y\sigma_C)\} \models_{\mathcal{D}} (PF', \mathit{\Pi}_{Y'}\sigma_{C'})$, where $\mathcal{D}$ is empty. In this case, we say that $(PF, \Pi_Y\sigma_C)$ *data-dependently dominates* or simply *dominates* $(PF', \Pi_{Y'}\sigma_{C'})$.

Note that throughout the paper we use $\models$ to denote both data-dependent disclosure when $\mathcal{D}$ is empty as well as to denote logical entailment. The meaning of $\models$ will be clear from its context.

By the Definition 7, if $\mathcal{D}$ is inconsistent (i.e., no relation satisfies it) then everything is disclosed.

The construction of data-dependent disclosure is computationally expensive and requires large storage overhead. Therefore, we introduce the notion of data-independent disclosure, where only the queries are considered to detect whether an example of data-dependent disclosure can be found, that corresponds to the queries.

Definition 8 (Data-independent existential disclosure)
Let $\mathcal{D}$ be a set of database constraints and $\Pi_{X_1}\sigma_{C_1}, \ldots, \Pi_{X_n}\sigma_{C_n}$ queries over R. We say that the set of queries

$$\mathcal{P} = \{\mathit{\Pi}_{X_1}\sigma_{C_1}, \ldots, \mathit{\Pi}_{X_n}\sigma_{C_n}\}$$

data-independently (or existentially) discloses the query $\Pi_Y\sigma_C$ under $\mathcal{D}$, denoted as $\mathcal{P} \leadsto_{\exists\mathcal{D}} \mathit{\Pi}_Y\sigma_C$, if there exist

1. r over R that satisfies $\mathcal{D}$,

2. sets $P_1 \subseteq \Pi_{X_1}\sigma_{C_1}(r), \ldots, P_n \subseteq \Pi_{X_n}\sigma_{C_n}(r)$, and

3. $PF \in \Pi_Y\sigma_C(r)$

such that $\{(P_1, \Pi_{X_1}\sigma_{C_1}), \ldots, (P_n, \Pi_{X_n}\sigma_{C_n})\} \models_{\mathcal{D}} (PF, \mathit{\Pi}_Y\sigma_C)$

We say, that $\mathcal{P} \leadsto_{\exists\mathcal{D}} \mathcal{P}'$ if for every $\Pi_Y\sigma_C$ in $\mathcal{P}'$, $\mathcal{P} \leadsto_{\exists\mathcal{D}} \mathit{\Pi}_Y\sigma_C$. Further, $\mathcal{P} \leadsto_{\exists\mathcal{D}} \mathit{\Pi}_Y$ will denote $\mathcal{P} \leadsto_{\exists\mathcal{D}} \mathit{\Pi}_Y\sigma_{TRUE}$, and $\mathcal{P} \leadsto_{\exists\mathcal{D}} \sigma_C$ will denote $\mathcal{P} \leadsto_{\exists\mathcal{D}} \mathit{\Pi}_R\sigma_C$, where R is the set of all attributes of R. Also, we say that $\mathcal{P} \leadsto_{\exists\mathcal{D}} (PF, \Pi_Y\sigma_C)$ if $\mathcal{P} \leadsto_{\exists\mathcal{D}} \mathit{\Pi}_Y\sigma_{C'}$, where C' is constructed as follows: if $PF = R[A_{i_1} = a_1, \ldots, A_{i_k} = a_k]$, where $A_{i_1}, \ldots, A_{i_k} = Y$, then C' is C conjuncted with all equalities of the form $A_{i_j} = a_j$, $(j = 1, \ldots, k)$. Finally, we denote by $\Pi_{Y'}\sigma_{C'} \leadsto_{\exists} \Pi_Y\sigma_C$ the case where $\{\Pi_{Y'}\sigma_{C'}\} \leadsto_{\exists\mathcal{D}} \mathit{\Pi}_Y\sigma_C$ and $\mathcal{D}$ is empty. In this case we say that $\Pi_{Y'}\sigma_{C'}$ *existentially dominates* $\Pi_Y\sigma_C$.

To solve the problem of existential disclosure, we also need the notation of universal disclosure, defined as follows.

Definition 9 (Universal disclosure)
Let $\mathcal{D}$ be a set of database constraints and $\Pi_{X_1}\sigma_{C_1}, \ldots, \Pi_{X_n}\sigma_{C_n}$ queries over R. We say, that the set of queries

$$\mathcal{P} = \{\mathit{\Pi}_{X_1}\sigma_{C_1}, \ldots, \mathit{\Pi}_{X_n}\sigma_{C_n}\}$$

universally discloses the query $\Pi_Y\sigma_C$ under $\mathcal{D}$, denotes as $\mathcal{P} \rightsquigarrow_{\forall\mathcal{D}} \mathit{\Pi}_Y\sigma_C$, if for every PF over $\Pi_Y\sigma_C$ there exist

1. r over R that satisfies $\mathcal{D}$,

2. sets $P_1 \subseteq \Pi_{X_1}\sigma_{C_1}(r), \ldots, P_n \subseteq \Pi_{X_n}\sigma_{C_n}(r)$

such that $\{(P_1, \Pi_{X_1}\sigma_{C_1}), \ldots, (P_n, \Pi_{X_n}\sigma_{C_n})\} \models_{\mathcal{D}} (PF, \mathit{\Pi}_Y\sigma_C)$.

Similarly to the existential disclosure, we say that $\mathcal{P} \rightsquigarrow_{\forall\mathcal{D}} \mathcal{P}'$ if for every $\Pi_Y\sigma_C$ in $\mathcal{P}'$, $\mathcal{P} \rightsquigarrow_{\forall\mathcal{D}} \mathit{\Pi}_Y\sigma_C$. Further, $\mathcal{P} \rightsquigarrow_{\forall\mathcal{D}} \mathit{\Pi}_Y$ will denote $\mathcal{P} \rightsquigarrow_{\forall\mathcal{D}} \mathit{\Pi}_Y\sigma_{TRUE}$, $\mathcal{P} \rightsquigarrow_{\forall\mathcal{D}} \sigma_C$ will denote $\mathcal{P} \rightsquigarrow_{\forall\mathcal{D}} \mathit{\Pi}_R\sigma_C$, where R is the set of all attributes of R, and $\mathcal{P} \rightsquigarrow_{\forall\mathcal{D}} (PF, \Pi_Y\sigma_C)$ if $\mathcal{P} \rightsquigarrow_{\forall\mathcal{D}} \mathit{\Pi}_Y\sigma_{C'}$ and C' is constructed as follows: if $PF = R[A_{i_1} = a_1, \ldots, A_{i_k} = a_k]$, where $A_{i_1}, \ldots, A_{i_k} = Y$, then C' is C conjuncted with all equalities of the form $A_{i_j} = a_j$, $(j = 1, \ldots, k)$. Finally, we denote by $\Pi_{Y'}\sigma_{C'} \rightsquigarrow_{\forall} \Pi_Y\sigma_C$ the case where $\{\Pi_{Y'}\sigma_{C'}\} \rightsquigarrow_{\forall\mathcal{D}} \mathit{\Pi}_Y\sigma_C$ and $\mathcal{D}$ is empty. In this case we say that $\Pi_{Y'}\sigma_{C'}$ *universally dominates* $\Pi_Y\sigma_C$.

Note that, by Definitions 8 and 9, if $\mathcal{D}$ is inconsistent then nothing is disclosed. We study the interconnection between the existential and the universal disclosure in Section 3.5. The following is an example of both existential and universal dominance (i.e., simple case of disclosure).

Example 6 Consider queries Π_{ABC} and $\Pi_{ABC}\sigma_{A=2}$. According to our definitions, we have

$$\Pi_{ABC} \rightsquigarrow_{\forall} \Pi_{ABC}\sigma_{A=2} \quad \mathit{and} \quad \Pi_{ABC} \rightsquigarrow_{\exists} \Pi_{ABC}\sigma_{A=2}$$

However, while

$$\Pi_{ABC}\sigma_{A=2} \rightsquigarrow_{\exists} \Pi_{ABC}$$

we do not have universal disclosure; i.e.,

$$\Pi_{ABC}\sigma_{A=2} \not\rightsquigarrow_{\forall} \Pi_{ABC}$$

Also, note that

$$\Pi_{ABC}\sigma_{A=2} \not\rightsquigarrow_{\exists} \Pi_{ABC}\sigma_{A=3}$$

□

3.4 Data-Dependent Disclosure

In this section we develop an algorithm to construct a cover of all disclosed query-answer pairs under database constraints $\mathcal{D}$ from the set $\mathcal{P}$ of query-answer pairs. First we state the decidability result for the data-dependent disclosure.

Theorem 1 The following problem is decidable:
Given a set $\mathcal{D}$ of database constraints and a set $\mathcal{P}$ of QA-pairs,

1. whether $\mathcal{P} \models_{\mathcal{D}} (PF, \mathit{\Pi}_Y\sigma_C)$ for a given atomic QA-pair $(PF, \Pi_Y\sigma_C)$

2. whether $\mathcal{P} \models_{\mathcal{D}} S$ for a given set S of atomic QA-pairs.

Theorem 1 will be a corollary to Theorem 2 that states correctness (i.e., soundness and completeness) of Algorithm 3 that computes a data-dependent disclosure cover which is defined in this section. To define disclosure cover, first we need the following propositions.

Proposition 1 Let $(PF, \Pi_Y \sigma_C)$ and $(PF', \Pi_{Y'} \sigma_{C'})$ two QA-pairs, and let C_e, C'_e be the selection conditions generated from C and C', respectively, by extending them with the equalities explicitly given in PF and PF'. Then,

1. $(PF, \Pi_Y \sigma_C) \models (PF', \Pi_{Y'} \sigma_{C'}) \Longleftrightarrow C_e \models C'_e$

2. data-dependent dominance is transitive, that is $(PF, \Pi_Y \sigma_C) \models (PF', \Pi_{Y'} \sigma_{C'}) \wedge (PF', \Pi_{Y'} \sigma_{C'}) \models (PF'', \Pi_{Y''} \sigma_{C''}) \Rightarrow (PF, \Pi_Y \sigma_C) \models (PF'', \Pi_{Y''} \sigma_{C''})$

Proof: See Appendix 7.1.1. □

Note that data-dependent dominance can be extended to QA-pairs that contain sets of projection facts. We say, that $(P, \Pi_Y \sigma_C)$ dominates $(P', \Pi_{Y'} \sigma_{C'})$ if for every $PF' \in P'$, there exists a $PF \in P$ such that $(PF, \Pi_Y \sigma_C) \models (PF', \Pi_{Y'} \sigma_{C'})$.

Clearly, there may be different QA-pairs Q_1 and Q_2 that are equivalent in the sense that each dominates the other (i.e., $Q_1 \models Q_2$ and $Q_2 \models Q_1$). For example, the QA-pairs $(R[A = 1, B = 2, L = 3, D = 4], \Pi_{ABLD} \sigma_{E=F})$, $(R[A = 1, B = 2], \Pi_{AB} \sigma_{L=3 \wedge D=4 \wedge E=F})$, $(R[A = 1], \Pi_A \sigma_{B=2 \wedge L=3 \wedge D=4 \wedge E=F})$, $(R[A = 1, L = 3], \Pi_{AL} \sigma_{B=2 \wedge D=4 \wedge E=F})$ are all equivalent. The reason is that all of the above QA-pairs result in the same extended selection condition: $A = 1 \wedge B = 2 \wedge L = 3 \wedge D = 4 \wedge E = F$. To provide a uniform representation of equivalent QA-pairs, we define the notion of normal forms of QA-pairs as follows:

Definition 10 (Normal form of a QA-pair)

Given a QA-pair $(PF, \Pi_Y \sigma_C)$, its *normal form*, $(PF', \Pi_{Y'} \sigma_{C'})$ is constructed as follows:

Let C_e be the selection condition constructed from C by extending it with all the equalities explicitly given in PF. Initially, $PF' = PF, Y' = Y$, and $C' = C$.

Step 1: For every attribute $A_i \notin Y'$ such that $C_e \models A_i = c$, where c is a constant, do:

1. Extend PF' over A_i by adding $A_i = c$, and
2. Add A_i to Y'.

Step 2: For every equality $A_i = B \in C'$, where B is either a constant or an attribute, such that $A_i \in Y'$ remove $A_i = B$ from C' (i.e., C' contains only equalities not given explicitly in PF').

Finally, we say, that a QA-pair $(PF, \Pi_Y \sigma_C)$ is in normal form if it is identical to its normal form. Clearly, every QA-pair is equivalent to its normal form.

Example 7 Consider the QA-pair $(R[A = 2, B = 3], \Pi_{AB}\sigma_{L=B \wedge D=L \wedge E=F})$. The extended selection condition C_e is $A = 2 \wedge B = 3 \wedge L = B \wedge D = L \wedge E = F$. Then, $C_e \models L = 3$ and $C_e \models D = 3$, therefore the normal form of $(R[A = 2, B = 3], \Pi_{AB}\sigma_{L=B \wedge D=L \wedge E=F})$ is $(R[A = 2, B = 3, L = 3, D = 3], \Pi_{ABLD}\sigma_{E=F})$. □

Proposition 2 Data-dependent disclosure is transitive, that is for any sets $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ of QA-pairs

$$\mathcal{P}_1 \models_{\mathcal{D}} \mathcal{P}_2 \wedge \mathcal{P}_2 \models_{\mathcal{D}} \mathcal{P}_3 \Rightarrow \mathcal{P}_1 \models_{\mathcal{D}} \mathcal{P}_3$$

Proof: See Appendix 7.1.2. □

Definition 11 (Data-dependent disclosure cover of $\mathcal{P}$ under $\mathcal{D}$ ($DDC_{\mathcal{D}}(\mathcal{P})$))
Given a set of dependencies $\mathcal{D}$ and a set of QA-pairs

$$\mathcal{P} = \{(P_1, \Pi_{X_1}\sigma_{C_1}), \ldots, (P_n, \Pi_{X_n}\sigma_{C_n})\}$$

data-dependent disclosure cover under $\mathcal{D}$, denoted as $DDC_{\mathcal{D}}(\mathcal{P})$, is defined as a set $\mathcal{S}$ of QA-pairs $(P, \Pi_Y \sigma_C)$ that is

1. *sound*; i.e., for every $(P, \Pi_Y \sigma_C) \in \mathcal{S}$ and $(P', \Pi_{Y'} \sigma_{C'})$

$$(P, \Pi_Y \sigma_C) \models (P', \Pi_{Y'} \sigma_{C'}) \Rightarrow \mathcal{P} \models_{\mathcal{D}} (P', \mathit{\Pi}_{Y'} \sigma_{C'})$$

2. *complete*; i.e., for every $(PF', \Pi_{Y'} \sigma_{C'})$, $\mathcal{P} \models_{\mathcal{D}} (PF', \mathit{\Pi}_{Y'} \sigma_{C'})$ implies that there exists $(P, \Pi_Y \sigma_C)$ in $\mathcal{S}$ such that $(P, \Pi_Y \sigma_C) \models (PF', \Pi_{Y'} \sigma_{C'})$

3. *compact*, for every $(P, \Pi_Y \sigma_C) \in \mathcal{S}$

   (a) it is in normal form

   (b) C is non-redundant, that is, no strict subset of C is equivalent to C

   (c) there does not exist $(P', \Pi_{Y'} \sigma_{C'})$ in $\mathcal{S}$ that is different from $(P, \Pi_Y \sigma_C)$ and that dominates $(P, \Pi_Y \sigma_C)$.

Before we can describe the algorithm that is based on the *chase* process [Ull88], we need to define first the notions of atom mapping and application of dependencies.

Definition 12 (Atom mapping of dependencies)
Given a Horn-clause constraint $B_1, \ldots, B_n \rightarrow H$ and a relation r, we define an *atom mapping* as a function

$$h : \{B_1, \ldots, B_n\} \rightarrow r$$

such that

1. h preserves constants; i.e., if $h(R[\ldots, A_i = c, \ldots]) = (c_1, \ldots, c_i, \ldots, c_n) \in r$ and c is a constant, then $c = c_i$

2. h preserves equalities; i.e., if $B_i = R[\ldots, A_k = a, \ldots]$, $B_j = R[\ldots, A_l = a, \ldots]$ and $h(B_i) = (c_1, \ldots, c_k, \ldots, c_n)$, $h(B_j) = (c'_1, \ldots, c'_l, \ldots, c'_n)$, then $c_k = c'_l$.

Clearly, h can be extended to work from the set S_d of the symbols (i.e., variables and constants) of the constraint to the symbols of r. We will use the symbol h for this extended mapping as well.

Definition 13 (Application of dependencies)
A dependency d is applied on a relation r by using an atom mapping h as follows:

1. If d is an equality-generating dependency of the form

$$B_1, \ldots, B_n \rightarrow a = b$$

then *equate* $h(a)$ and $h(b)$ as follows:

   (a) If both $h(a)$ and $h(b)$ are null-values then replace all occurrences of one of them in r with the other.

   (b) If one of them, say $h(a)$, is not a null-value, then replace all occurrence of $h(b)$ in r with $h(a)$.

   (c) If both are not null-values (i.e., constants), do nothing. If $h(a) \neq h(b)$, we say that *inconsistency* occurred.

2. If d is a tuple-generating dependency of the form

$$B_1, \ldots, B_n \rightarrow R[A_1 = a_1, \ldots, A_n = a_n]$$

and the tuple $(h(a_1), \ldots, h(a_n))$ is not in r, then add it to r.

Now that the application of the dependencies is defined, we can describe the chase process, which is a variation of the *Chase* defined in [Ull88]. Algorithm 2 for the chase process is given in Figure 3.2.

Algorithm 2:	Chase process
INPUT	1. Set of Horn-clause constraints $\mathcal{D}$
	2. Relation r, which may contain null-values
OUTPUT	Updated relation r.
METHOD	**begin**
	Apply dependencies in $\mathcal{D}$ on r until no more changes to r occur.
	end

Figure 3.2: Chase process

In order to preserve the information encapsulated in the selection conditions of the queries, we generate dependencies from them. These dependencies are then applied to the answers of the corresponding queries. Let C be a conjunction of conditions of the forms $A_i = A_j$ or $A_i = c$, where A_i and A_j are attribute names and c is a constant. We *generate dependencies* from each condition in C as follows:

1. For each condition $A_i = A_j$ we create a dependency $R[A_i = a_i, A_j = a_j] \rightarrow a_i = a_j$
2. For each condition $A_i = c$ we create a dependency $R[A_i = a_i] \rightarrow a_i = c$

Figure 3.3 shows Algorithm 3 that generates the data-dependent disclosure cover of a set $\mathcal{P}$ of query-answer pairs under dependencies $\mathcal{D}$.

Theorem 2 (data-dependent disclosure) Algorithm 3 effectively computes the data-dependent disclosure cover $DDC_{\mathcal{D}}(\mathcal{P})$; i.e., the algorithm terminates and the computed $\mathcal{S}$ is (1) *sound*, (2) *complete* and (3) *compact*.

Proof: See Appendix 7.1.3. $\square$

The complexity of Algorithm 3 is dominated by the *chase* process in Step 2. In it, an iteration (application of a dependency) is bounded by $O(I^T)$, where I is the

Algorithm 3:	Data-Dependent Disclosure Cover
INPUT	1. Set $\mathcal{P} = \{(P_1, \mathit{\Pi}_{X_1}\sigma_{C_1}), \ldots, (P_m, \mathit{\Pi}_{X_m}\sigma_{C_m})\}$ of QA-pairs 2. Set of Horn-clause constraints $\mathcal{D}$
OUTPUT	Data-dependent disclosure cover $DDC_{\mathcal{D}}(\mathcal{P})$ (The set $\mathcal{S}$ at the conclusion of the algorithm.)

METHOD

1. **for** every $(P_i, \Pi_{X_i}\sigma_{C_i})$ in $\mathcal{P}$ **do**

   (a) *Augment* each $PF \in P_i$ with new unique null values for attributes in $R - X_i$ to create r_i

   (b) *Generate dependencies* D_i from C_i

   (c) *Chase* r_i with D_i to construct r'_i

2. Construct $r' = \bigcup_{i=1}^{m} r'_i$ and *chase* r' with dependencies $\mathcal{D}$ to get r''

3. **if** during Step 1 or 2 *inconsistency* occurred, **then** return $\mathcal{S} = \{(ALL, \Pi_R\sigma_{TRUE})\}$ as output, where ALL is the domain of R (i.e., everything is disclosed)
   else let $\mathcal{S}$ to be the set of all $(PF, \Pi_Y\sigma_C)$ such that there exists $t \in r''$ and

   (a) Y contains all attributes in which t is not null

   (b) $PF = \Pi_Y(t)$

   (c) C encodes only the equalities among the attributes which do not appear explicitly in PF.

4. (a) Continue until no more changes occur:
   if two different QA-pairs $(PF, \Pi_Y\sigma_C)$ and $(PF', \Pi_{Y'}\sigma_{C'})$ are in $\mathcal{S}$, and $(PF, \Pi_Y\sigma_C) \models (PF', \Pi_{Y'}\sigma_{C'})$ then remove $(PF', \Pi_{Y'}\sigma_{C'})$ from $\mathcal{S}$.

   (b) Continue until no more changes occur:
   if two different QA-pairs $(P, \Pi_Y\sigma_C)$ and $(P', \Pi_Y\sigma_{C'})$ are in $\mathcal{S}$ such that $C \equiv C'$ do:

      i. Add $(P \cup P', \Pi_Y\sigma_C)$ to $\mathcal{S}$, and

      ii. Remove $(P, \Pi_Y\sigma_C)$ and $(P', \Pi_Y\sigma_{C'})$ from $\mathcal{S}$.

   (c) **for** each $(P, \Pi_Y\sigma_C)$ in $\mathcal{S}$ replace C by an equivalent minimal subset of C.

5. Return $\mathcal{S}$ as the output.

Figure 3.3: Data-Dependent Disclosure Cover

number of symbols in r' and T is the total number of attributes of r'. Each iteration, if done naively, may check $O(I^{nT})$ of potential atom mappings, where n is the number of formulas in the body of the dependency, spending $O(nT)$ time for a check. Thus, the work in each iteration is bounded by $O(nT \cdot I^{nT})$. Therefore, the overall time of *chase* is bounded by $O(nT \cdot I^{nT} \cdot I^{T}) = O(nT \cdot I^{(n+1)T})$. Since the number of symbols in r' is bounded by qT, where q is the overall number of atomic QA-pairs in $\mathcal{P}$, the complexity of Algorithm 3 is bounded by $O(nT \cdot (qT)^{(n+1)T})$, i.e. polynomial in the size of the input, if the number of attributes is bounded by a constant, and exponential in the number of attributes.

Example 8 This is a detailed illustration of generating the data-dependent disclosure cover for the two queries presented earlier. The input for Algorithm 3 is the QA-pairs

$$\mathcal{P} = \{(R[RANK = Clerk, SALARY = 34,000], \mathit{\Pi}_{RANK,SALARY}),$$

$$(R[NAME = Smith, RANK = Accountant], \Pi_{NAME,RANK}\sigma_{EXPERIENCE=6})\}$$

and the Horn-clause constraint

$$R[RANK = r, SALARY = s_1] \wedge R[RANK = r, SALARY = s_2] \rightarrow s_1 = s_2$$

that represents the functional dependency $RANK \rightarrow SALARY$. Table 3.1 shows the relation r' (Step 2 of Algorithm 3) that is chased by the FD. Since there does not exist any atom mapping from the FD to the tuples of r' the FD cannot be applied (see Definition 13) and, therefore, r' is also the output relation (r'') of chase process.

In Step 3 r'' is used to construct $\mathcal{S}$ that is the data-dependent disclosure cover of $\mathcal{P}$ under FD.

$$\mathcal{S} = \{(R[RANK = Clerk, SALARY = 34,000], \mathit{\Pi}_{RANK,SALARY}),$$

$$(R[NAME = Smith, RANK = Accountant, EXPERIENCE = 6],$$

$$\Pi_{NAME,RANK,EXPERIENCE})\}$$

Since $\mathcal{S}$ does not disclose any top-secret object the query is answered to the user. □

Example 9 Now, consider the previous example with the difference that the employee Smith is a clerk, that is the fifth tuple in the *Employee* relation is <Smith, Clerk, 34,000, 6>. Then the input for Algorithm 3 is the QA-pairs

$$\mathcal{P} = \{(R[RANK = Clerk, SALARY = 34,000], \Pi_{RANK,SALARY}),$$

$$(R[NAME = Smith, RANK = Clerk], \Pi_{NAME,RANK}\sigma_{EXPERIENCE=6})\}$$

In this case, there exists an atom mapping h from the FD to the tuples generated from $\mathcal{P}$ (See Table 3.2), that is $h(r) = Clerk$, $h(s_1) = 34,000$, and $h(s_2) = \delta_4$. The result of the application of the functional dependency with mapping h is that δ_4 is replaced with $34,000$. Since the FD cannot be applied again, DiIE generates the set

$$\{(R[RANK = Clerk, SALARY = 34,000], \Pi_{RANK,SALARY}),$$

$$(R[NAME = Smith, RANK = Clerk, SALARY = 34,000,$$

$$EXPERIENCE = 6], \Pi_{NAME,RANK,SALARY,EXPERIENCE})\}$$

After the redundant QA-pair $(R[RANK = Clerk, SALARY = 34,000], \Pi_{RANK,SALARY})$ is removed, the final output of Algorithm 3 is

$$\mathcal{S} = \{(R[NAME = Smith, RANK = Clerk, SALARY = 34,000,$$

$$EXPERIENCE = 6], \Pi_{NAME,RANK,SALARY,EXPERIENCE})\}$$

Since $\mathcal{S}$ discloses a top-secret object, the second query is rejected. □

3.5 Data-Independent Disclosure

We start the section with the decidability result of data-independent disclosure.

Theorem 3 If queries and constraints *do not involve constants* then the following problem is decidable:
Given a set of queries $\mathcal{P} = \{\Pi_{X_1}\sigma_{C_1}, \ldots, \Pi_{X_n}\sigma_{C_n}\}$, a set of Horn-clause constraints $\mathcal{D}$, and a query $\Pi_Y\sigma_C$, determine whether $\mathcal{P} \leadsto_{\exists\mathcal{D}} \Pi_Y\sigma_C$.

The proof of Theorem 3 is based on the correctness of the algorithm that computes existential disclosure cover that we present in this section. As in the data-dependent case, we need the notion of normal forms.

Definition 14 (Normal form of $\Pi_Y\sigma_C$)
Given a query $\Pi_Y\sigma_C$, its *normal form* is $\Pi_{Y'}\sigma_C$, where Y' is the minimal set of attributes that is closed under the following properties:

1. $Y \subseteq Y'$
2. If $C \models A_i = A_j$ and $A_j \in Y$, then $A_i \in Y'$
3. If $C \models A_i = c$, where c is a constant, then $A_i \in Y'$

If $Y' = Y$, we say that $\Pi_Y\sigma_C$ is in normal form.

The following proposition establishes syntactic conditions (that can be effectively tested) for universal and existential dominance which are used in this section.

Proposition 3 Let $\Pi_Y\sigma_C$ and $\Pi_{Y'}\sigma_{C'}$ be two queries in normal forms. Then,

1. $\Pi_Y\sigma_C \leadsto_{\exists} \Pi_{Y'}\sigma_{C'} \iff$

(a) $Y' \subseteq Y$,

(b) $C' \wedge C$ is consistent, and

(c) $C \models C'_{\neg Y}$, where $C'_{\neg Y}$ is the conjunction of equalities $A_i = A_j$ such that $C' \models A_i = A_j$ and $A_i \notin Y$ or $A_j \notin Y$.

2. $\Pi_Y \sigma_C \rightsquigarrow_{\forall} \Pi_{Y'} \sigma_{C'} \iff$

(a) $\Pi_Y \sigma_C \rightsquigarrow_{\exists} \Pi_{Y'} \sigma_{C'}$ and

(b) $C' \models (C \wedge C')_{Y'}$, where $(C \wedge C')_{Y'}$ is the conjunction of equalities

i. $A_i = c$ such that $C \wedge C' \models A_i = c$, where c is a constant and $A_i \in Y'$, or

ii. $A_i = A_j$, such that $C \wedge C' \models A_i = A_j$, where A_i and A_j are in Y'

Proof: See Appendix 7.2.1. $\square$

Next, we establish the relationship between the notions of universal and existential dominance.

Proposition 4 For every $\Pi_Y \sigma_C$, $\mathcal{P}$, and $\mathcal{D}$

1. If C is consistent, then

$$\mathcal{P} \rightsquigarrow_{\forall \mathcal{D}} \mathit{\Pi}_Y \sigma_C \Rightarrow \mathcal{P} \rightsquigarrow_{\exists \mathcal{D}} \mathit{\Pi}_Y \sigma_C$$

2. If $\mathcal{P} \rightsquigarrow_{\exists \mathcal{D}} \mathit{\Pi}_Y \sigma_C$, then there exists a consistent set C' such that $(C' \models C)$, $\mathcal{P} \rightsquigarrow_{\forall \mathcal{D}} \mathit{\Pi}_Y \sigma_{C'}$ and $\Pi_Y \sigma_{C'} \rightsquigarrow_{\exists} \Pi_Y \sigma_C$

Proof: See Appendix 7.2.2. $\square$

The following two propositions deal with the transitivity of universal and existential disclosures.

Proposition 5 Universal disclosure is transitive. Let $\mathcal{P}_1, \mathcal{P}_2$, and $\mathcal{P}_3$ be sets of queries.

$$\mathcal{P}_1 \rightsquigarrow_{\forall\mathcal{D}} \mathcal{P}_2 \ \wedge\ \mathcal{P}_2 \rightsquigarrow_{\forall\mathcal{D}} \mathcal{P}_3 \Rightarrow \mathcal{P}_1 \rightsquigarrow_{\forall\mathcal{D}} \mathcal{P}_3$$

In particular, universal dominance is transitive; i.e.,

$$\Pi_{Y_1}\sigma_{C_1} \rightsquigarrow_{\forall} \Pi_{Y_2}\sigma_{C_2} \ \wedge\ \Pi_{Y_2}\sigma_{C_2} \rightsquigarrow_{\forall} \Pi_{Y_3}\sigma_{C_3} \Rightarrow \Pi_{Y_1}\sigma_{C_1} \rightsquigarrow_{\forall} \Pi_{Y_3}\sigma_{C_3}$$

Proof: See Appendix 7.2.3. □

Proposition 6 Existential disclosure ($\rightsquigarrow_{\exists}$) is not transitive

Proof: See Appendix 7.2.4. □

Similarly to the data-dependent case, we need the notion of compact cover.

Definition 15 (Data-independent disclosure cover of $\mathcal{P}$ under $\mathcal{D}$)

Given $\mathcal{P} = \{\mathit{\Pi}_{X_1}\sigma_{C_1}, \ldots, \mathit{\Pi}_{X_n}\sigma_{C_n}\}$ a set of queries, the *data-independent disclosure cover* of $\mathcal{P}$ under $\mathcal{D}$, denoted as $IDC_{\exists\mathcal{D}}(\mathcal{P})$, is a set $\mathcal{S}$ of queries $\Pi_Y\sigma_C$ that is

1. *sound*; i.e., for every $\Pi_Y\sigma_C$ in $\mathcal{S}$ and query $\Pi_{Y'}\sigma_{C'}$,

$$\Pi_Y\sigma_C \rightsquigarrow_{\exists} \Pi_{Y'}\sigma_{C'} \Rightarrow \mathcal{P} \rightsquigarrow_{\exists\mathcal{D}} \mathit{\Pi}_{Y'}\sigma_{C'}$$

2. *complete*; i.e., for every query $\Pi_{Y'}\sigma_{C'}$, $\mathcal{P} \rightsquigarrow_{\exists\mathcal{D}} \mathit{\Pi}_{Y'}\sigma_{C'}$ implies that there exists $\Pi_Y\sigma_C$ in $\mathcal{S}$ such that $\Pi_Y\sigma_C \rightsquigarrow_{\exists} \Pi_{Y'}\sigma_{C'}$

3. *compact*; i.e., for every query $\Pi_Y\sigma_C \in \mathcal{S}$

   (a) C is non-redundant, that is, no strict subset of C is equivalent to C

   (b) there does not exist $\Pi_{Y'}\sigma_{C'}$ in $\mathcal{S}$ that is different from $\Pi_Y\sigma_C$ and universally dominates $\Pi_Y\sigma_C$

Next, we present an algorithm that generates data-independent disclosure cover when neither the selection conditions of the queries nor the database dependencies contain constants. The algorithm is shown in Figure 3.4.

Algorithm 4:	Constant-free Data-Independent Disclosure Cover
INPUT	1. $\mathcal{P} = \{\mathit{\Pi}_{X_1}\sigma_{C_1}, \ldots, \mathit{\Pi}_{X_m}\sigma_{C_m}\}$ (C_is are conjunctions of equalities between attributes) 2. $\mathcal{D}$ (set of Horn-clause constraints without constants)
OUTPUT	$\mathcal{S}$ (existential disclosure cover of $\mathcal{P}$ under $\mathcal{D}$)
METHOD	1. (*Initialization*) **for** every $\Pi_{X_i}\sigma_{C_i} \in \mathcal{P}$ *generate* a tuple t_i as follows: (a) **for** each attribute $A_j \in X_i$ assign the variable x. (b) **if** C_i contains equality among two attributes A_i and A_j such that at least one of them is in X_i then assign variable x to both A_i and A_j. (c) Assign unique null values to attributes that were not considered previously, while preserving equalities in C_i; i.e., if $A_i = A_j \in C_i$ and $A_i, A_j \notin X_i$ then assign the same null value to both A_i and A_j. 2. *Construct* r to be $\{t_1, \ldots, t_m\}$ 3. *Chase* r with $\mathcal{D}$ to generate r' 4. Compute $\mathcal{S}$ as follows: **for** each tuple $t \in r'$ generate a query $\Pi_Y\sigma_C$, where (a) Y contains all the attributes A_i where $t[A_i] = x$ (b) C has all the equalities among null values. 5. (a) Continue until no more changes occur: **for** every two different queries $\Pi_Y\sigma_C$ and $\Pi_{Y'}\sigma_{C'}$ in $\mathcal{S}$, if $\Pi_Y\sigma_C \rightsquigarrow_{\forall} \Pi_{Y'}\sigma_{C'}$ then remove $\Pi_{Y'}\sigma_{C'}$ from $\mathcal{S}$. (b) **for** every query $\Pi_Y\sigma_C$ in $\mathcal{S}$, replace C by an equivalent minimal subset of C. 6. Return $\mathcal{S}$ as output.

Figure 3.4: Constant-free Data-Independent Disclosure Cover

Theorem 4 (Constant-free Data-Independent Disclosure Cover)
Algorithm 4 effectively computes $IDC_{\exists\mathcal{D}}(\mathcal{P})$; i.e., the algorithm terminates and the computed $\mathcal{S}$ is (1) *sound*, (2) *complete*, and (3) *compact*.

Proof: See Appendix 7.2.5. $\square$

Similarly to Algorithm 3, the complexity of Algorithm 4 is $O(nT \cdot I^{(n+1)T})$, where I is the number of symbols in r (see Step 2 of Algorithm 4). Since I is bounded by qT, where q is the number of queries is $\mathcal{P}$, the complexity is bounded by $O(nT \cdot (qT)^{(n+1)T})$. Of course, a typical input size of Algorithm 4 is much smaller than the input size of Algorithm 3, since Algorithm 4 only considers queries not the actual data. Next we give an example of applying Algorithm 4.

Example 10 This is a detailed illustration of generating a data-independent disclosure cover for the queries, $\Pi_{SALARY,RANK}$ and $\Pi_{NAME,RANK}$. The input to Algorithm 4 is:

$$\mathcal{P} = \{\mathit{\Pi}_{SALARY,RANK}, \mathit{\Pi}_{NAME,RANK}\}$$

and

$$\mathcal{D} = \{R[RANK = r, SALARY = s_1] \wedge R[RANK = r, SALARY = s_2] \rightarrow s_1 = s_2\}$$

Table 3.3.a shows that tuples generated in Step 1 of Algorithm 5 from $\mathcal{P}$. Next, the functional dependency in $\mathcal{D}$ is applied with mapping h, where $h(r) = x$, $h(s_1) = x$, and $h(s_2) = \delta_3$. The result of the application is that δ_3 is replaced with x (3.3.b). Since the functional dependency cannot be applied again, $\mathcal{S}$ that is generated in Step 4 of Algorithm 4 is:

$$\mathcal{S} = \{\mathit{\Pi}_{RANK,SALARY}, \mathit{\Pi}_{NAME,RANK,SALARY}\}$$

NAME	RANK	SALARY	EXPERIENCE (years)
δ_1	x	x	δ_2
x	x	δ_3	δ_4

(a)

NAME	RANK	SALARY	EXPERIENCE (years)
δ_1	x	x	δ_2
x	x	x	δ_4

(b)

Table 3.3: Relations generated in Algorithm 4 from the queries $\Pi_{SALARY,RANK}$ and $\Pi_{NAME,RANK}$

Since $\Pi_{NAME,RANK,SALARY} \rightsquigarrow_\forall \Pi_{RANK,SALARY}$, the final output of Algorithm 1 is

$$\mathcal{S} = \{\mathit{\Pi}_{NAME,RANK,SALARY}\}$$

Since $\mathcal{S}$ discloses a *top-secret* object, the second query is rejected. $\square$

For the case with constants we need additional notions, such as weak-transitivity and generalized atom mapping, which we define next.

Definition 16 (Weak-transitivity)

Let E be a binary relation over the set V of variable names, null-values, and constants. E is *weakly transitive* if for every a, b, and c in V, where a and c are not two different constants, $(a, b) \in E$, $(b, c) \in E$ imply that $(a, c) \in E$. From now on, we denote $(a, b) \in E$ by $a \sim b$.

Definition 17 (Generalized atom mapping)

Let d be a dependency of the form $B_1, \ldots, B_n \rightarrow H$, r be a relation over the set V of variable names, null-values, and constant, and E be a weakly transitive binary relation over V. A *generalized atom mapping* h (with respect to d, r, and E) is a function

$$h : \{B_1, \ldots, B_n\} \to r$$

such that

1. h preserves constants; i.e., if $h(R[\ldots, A_i = c, \ldots]) = (c_1, \ldots, c_i, \ldots, c_n)$ and c is a constant, then $c \sim c_i$

2. h preserves equalities; i.e., if $B_i = R[\ldots, A_k = a, \ldots]$, $B_j = R[\ldots, A_l = a, \ldots]$ and $h(B_i) = (c_1, \ldots, c_k, \ldots, c_n)$, $h(B_j) = (c'_1, \ldots, c'_l, \ldots, c'_n)$ then $c_k \sim c'_l$.

Note, that h extends to a mapping from the symbols of d to sets of symbols of r. The reason is that, by definition of atom mapping h, two equal symbols of d may be mapped to two (possibly different) symbols of r that are related by E; i.e., if $s_1, s_2 \in h(a)$ then $s_1 \sim s_2$.

The application of dependencies is defined as follows:

Definition 18 (Application of dependencies)
An *E-Application* of a dependency d on (r, E) with respect to atom mapping h is defined as follows:

1. If d is an equality-generating dependency of the form $B_1, \ldots B_n \to a = b$ then for every pair s_1, s_2 such that $s_1 \in h(a)$, $s_2 \in h(b)$, s_1, s_2 are not two different constants, and $s_1 \sim s_2$ is not in E do:

   - Add $s_1 \sim s_2$ to E, and
   - Close the result under weak transitivity

2. If d is a tuple-generating dependency of the form $B_1, \ldots B_n \to R[A_1 = a_1, \ldots, A_n = a_n]$ then select representatives $s_1, \ldots, s_n$ from the sets $h(a_1), \ldots, h(a_n)$ as follows:

- If there exists a constant $c \in h(a_i)$ then $s_i = c$,
- Else, if there exists a variable $x \in h(a_i)$ then $s_i = x$,
- Otherwise, $s_i = \delta_i$, where δ_i is any null-value in $h(a_i)$.

If tuple $(s_1, \ldots, s_n)$ is not in r then add it to r.

Algorithm 5, presented in Figure 3.5, computes all the queries that are existentially disclosed from a set $\mathcal{P}$ of queries by database constraints $\mathcal{D}$.

Theorem 5 (Existential disclosure)
Algorithm 5 terminates and its output is a *complete* and *compact* set of existentially disclosed queries.

Proof: See Appendix 7.2.6. □

The complexity analysis of Algorithm 5 is similar to the one of Algorithm 4 but requires and additional argument. The difference is that not every iteration of the *E-Chase* in Step 4 of Algorithm 5 creates a new tuple. Each application of a tuple-generating dependency is bounded by I^T, where I is the number of symbols in r and T is the number of attributes of r. Every equality-generating dependency application must add a pair to the binary relation E. Since the size of E is bounded by I^2, the number of such iterations is bounded by I^2. In total, the number of iterations is bounded by $I^T + I^2$, which is $O(I^T)$, the same as the complexity of the *chase* in Algorithm 4. Therefore, the complexity of Algorithm 5 is the same as the complexity of Algorithm 4, that is $O(nT \cdot (qT)^{(n+1)T})$, where q is the number of queries in $\mathcal{P}$. Next we give an example to show how data-independent disclosure cover can be computed.

Example 11 This is the detailed illustration of generating the data-independent disclosure cover for the queries, $\Pi_{SALARY}\sigma_{RANK=Clerk}$ and $\Pi_{NAME,RANK}\sigma_{EXPERIENCE=6}$.

NAME	RANK	SALARY	EXPERIENCE (years)
δ_1	Clerk	x	δ_2
x	x	δ_3	6

Table 3.4: Tuples generated in Step 1 of Algorithm 5 from the queries Π_{SALARY} $\sigma_{RANK=Clerk}$ and $\Pi_{NAME,RANK}\sigma_{EXPERIENCE=6}$

The input of Algorithm 5 is:

$$\mathcal{P} = \{\mathit{\Pi}_{SALARY}\sigma_{RANK=Clerk}, \mathit{\Pi}_{NAME,RANK}\sigma_{EXPERIENCE=6}\}$$

and

$$\mathcal{D} = \{R[RANK = r, SALARY = s_1] \wedge R[RANK = r, SALARY = s_2] \rightarrow s_1 = s_2\}$$

Table 3.4 shows that tuples generated in Step 1 of Algorithm 5 from $\mathcal{P}$. The weakly transitive relation E, contains all reflexive pairs of the form $s \sim s$ over the symbols s of Table 3.4, and the pairs $x \sim Clerk$ and $x \sim 6$. Next, the functional dependency in $\mathcal{D}$ is applied with generalized atom mapping h, where $h(r) = Clerk \sim x$, $h(s_1) = x$, and $h(s_2) = \delta_3$. The result of the application is the extension of E with the pair $x \sim \delta_3$. After this, E is closed under weak transitivity, to derive new relations $Clerk \sim x \sim \delta_3$ and $6 \sim x \sim \delta_3$. Since the functional dependency cannot be applied again, the final output of Algorithm 5 is

$$\mathcal{S} = \{\mathit{\Pi}_{NAME,RANK,SALARY,EXPERIENCE}\sigma_{EXPERIENCE=6}\}$$

Since $\mathcal{S}$ discloses a *top-secret* object, the second query is rejected. □

Example 12 This is an example of Algorithm 5 when both functional and join dependencies are present. Let the following $\mathcal{P}$ and $\mathcal{D}$ be the input of Algorithm 5.

$$\mathcal{P} = \{\mathit{\Pi}_{RANK,SALARY}, \mathit{\Pi}_{RANK,EXPERIENCE}, \mathit{\Pi}_{NAME,EXPERIENCE}\sigma_{NAME=Smith}\}$$

NAME	RANK	SALARY	EXPERIENCE (years)
δ_1	x	x	δ_2
δ_3	x	δ_4	x
Smith	δ_5	δ_6	x

(a)

NAME	RANK	SALARY	EXPERIENCE (years)
δ_1	x	x	δ_2
δ_3	x	δ_4	x
Smith	δ_5	δ_6	x
δ_3	δ_5	δ_6	x
Smith	x	δ_4	x

(b)

Table 3.5: Example of Algorithm 5 with functional and join dependencies

$$\mathcal{D} = \{R[RANK = r, SALARY = s_1] \wedge R[RANK = r, SALARY = s_2] \rightarrow s_1 = s_2,$$

$$R[RANK = r, SALARY = s, EXPERIENCE = e] \wedge R[NAME = n, EXPERI-$$

$$ENCE = e] \rightarrow R[NAME = n, RANK = r, SALARY = s, EXPERIENCE = e]\}$$

Table 3.5(a) shows that tuples generated in Step 1 of Algorithm 5 from the two queries of $\mathcal{P}$. The weakly transitive relation E, contains all reflexive pairs of the form $s \sim s$ over the symbols s of Table 3.5, and the pair $x \sim Smith$.

Let the functional dependency be applied first, that generates the pair $x \sim \delta_4$, and after transitively closed, the pair $x \sim Smith \sim \delta_4$. Next, the join dependency can be applied twice, that generates the 4th and 5th tuples in the relation given in Table 3.5(b). Since neither the FD nor the JD can be applied any more, the set $\mathcal{S}$, generated in Step 5 of Algorithm 5 is

$$\mathcal{S} = \{\Pi_{RANK,SALARY}, \Pi_{RANK,SALARY,EXPERIENCE}, \Pi_{NAME,EXPERIENCE}\sigma_{NAME=}$$

$$_{Smith}, \Pi_{EXPERIENCE}, \Pi_{NAME,RANK,SALARY,EXPERIENCE}\sigma_{NAME=Smith}$$

Since $\Pi_{NAME,RANK,SALARY,EXPERIENCE}\sigma_{NAME=Smith}$ universally dominated all of the other queries in $\mathcal{S}$, they are removed in Step 6 of Algorithm 5, thus the final output of Algorithm 5 is $\mathcal{S} = \{\mathit{\Pi}_{NAME,RANK,SALARY,EXPERIENCE}\sigma_{NAME=Smith}\}$. Since $\mathcal{S}$ discloses a top-secret information, the second query is rejected.

$\square$

Algorithm 5:	General Data-Independent Disclosure Algorithm
INPUT	1. $\mathcal{P} = \{\Pi_{X_1}\sigma_{C_1}, \ldots, \Pi_{X_m}\sigma_{C_m}\}$ (C_is are conjunctions of equalities of the forms $A_i = A_j$ or $A_i = c$, where A_i, A_j are attributes and c is a constant..) 2. $\mathcal{D}$ (set of Horn-clause constraints)
OUTPUT	$\mathcal{S}$ (Complete and compact set of existentially disclosed queries)

METHOD

1. (*Initialization*) **for** every $\Pi_{X_i}\sigma_{C_i} \in \mathcal{P}$, $(i = 1, \ldots, m)$ *generate* a tuple t_i by assigning symbols to each attribute A_j, $j = 1, \ldots, n$ as follows:
   (a) **if** $C_i \models A_j = c$, where c is a constant, then assign c to A_j.
   (b) **else**, **if**
   - $A_j \in X_i$ or
   - there exists $A_k \in X_i$ such that $C_i \models A_j = A_k$

   then assign variable x to A_j,
   (c) **otherwise**, assign null-values to A_j as follows:
   - **if** $C_i \models A_j = A_k$, $(k = 1, \ldots, j$ and A_k was assigned δ_k then assign δ_k to A_j.
   - **else**, assign a unique null-value δ_j to A_j.
2. *Construct* r to be $\{t_1, \ldots, t_m\}$
3. *Construct* a weakly transitive binary relation E over the set V of symbols of r and the constants of $\mathcal{D}$ as follows:
   (a) E contains all *reflexive pairs* of the form $x \sim x$, $\delta_i \sim \delta_i$, and $c \sim c$.
   (b) **for** every constant c that appears in r or in $\mathcal{D}$, E contains equalities of the form $x \sim c$.
4. *E-Chase* (r, E) with $\mathcal{D}$; that is, E-Apply $\mathcal{D}$ on (r, E) until no more changes occur. The result is (r', E').
5. Compute $\mathcal{S}$ as follows:
   for each tuple $t \in r'$ generate a query $\Pi_Y\sigma_C$, where
   (a) Y contains all the attributes A_i such that
   - $t[A_i] = x$,
   - $t[A_i] = c$, where c is a constant, or
   - $t[A_i] = \delta_i$ and E' contains either $\delta_i \sim c$ or $\delta_i \sim x$

   (b) C contains the following equalities
   - $A_i = c$ if $t[A_i] = c$ and
   - $A_i = A_j$ if $A_i, A_j \notin Y$, and $t[A_i] = \delta_i$, $t[A_j] = \delta_j$, and E' contains $\delta_i \sim \delta_j$.
6. (a) Continue until no more changes occur:
   for every two different queries $\Pi_Y\sigma_C$ and $\Pi_{Y'}\sigma_{C'}$ in $\mathcal{S}$, if $\Pi_Y\sigma_C \leadsto_\forall \Pi_{Y'}\sigma_{C'}$ then remove $\Pi_{Y'}\sigma_{C'}$ from $\mathcal{S}$.
   (b) **for** every query $\Pi_Y\sigma_C$ in $\mathcal{S}$, replace C by an equivalent minimal subset of C.
7. Return $\mathcal{S}$ as output.

Figure 3.5: General Data-Independent Disclosure Algorithm

CHAPTER 4

Inference in Semi-Structured Databases

4.1 Problem Description

Privacy is defined as an individual's control over disclosure and use of personal information. At the age of globally interconnected information systems, the protection of privacy has become increasingly difficult. Besides direct attacks, such as intrusions into databases and eavesdropping, seemingly safe actions, like sharing non-private information with related organizations, may also lead to undesired disclosure of private data.

To illustrate an indirect privacy violation consider the following scenario from the medical domain. Jane Smith tells her physician that some of her ancestors had suffered from breast cancer and, therefore, she would like to be tested to determine whether she inherited the mutated BRCA1 and BRCA2 genes which would increase her risk of developing breast cancer. Since she is afraid that her health insurance company would discriminate against her if the test result is positive, she pays for the testing and asks the physician's office and the medical laboratory not to give any information about her test to the health insurance company. The test results are positive, meaning that Smith inherited the mutated genes and, therefore, she is at high risk to develop breast cancer. Smith's physician prescribes her the drugs Raloxifene and Hercepin which reduce her risk of developing the disease. Smith gets the medications from her pharmacy, which then sends the prescription and the billing information to her insurance company. Unfortunately, this unique combination of drugs is used only to treat people who are at high risk of developing breast cancer. Consequently, the insurance company assumes that Smith has breast cancer and cancels her insurance.

Smith's privacy has been violated. How could it be prevented? Smith, like most people, wants to protect her private data and at the same time obtain services like insurance coverage. Clearly, there is a trade off between privacy and data sharing needed to obtain these services. However, we believe that it is possible to achieve the "right balance" between privacy and data sharing so that individuals will be able to make informed decisions on privacy matters. Since the insurance company used the prescription information to infer Smith's illness, a naive solution to protect her privacy is to keep the prescription information private. However, while this solution would guarantee her privacy, it would also force her to pay for all of her medications. A better solution is to withhold just enough information about the prescriptions (e.g., submit only one of the prescriptions) so that the insurance company will not be able to infer Smith's illness. Clearly, the second solution is superior to the first one, because not only it protects Smith's privacy but it also allows her to get reimbursed for some of her expenses.

To address privacy problems in various domains, such as the medical domain, appropriate legislative and law-enforcement mechanisms are required. It is also necessary to develop technical solutions which not only enforce the privacy law and organizational (e.g., HMO's) rules but also address the privacy versus information sharing problem as presented in the example above. This is exactly the focus of this paper.

4.2 Data Model, Queries, and Disclosure Entailment

In this section we consider a tree-like, semi-structured data model and select-project queries over semi-structured data. Our data model is similar to the model proposed by Buneman et.al [BDHS96]; and the notation is adopted from [BDHS96]. However, our model has been modified to make it more convenient for reasoning on disclosure entailment, as described in this section. We also suggest a Horn-clause based representation of domain knowledge and define the concept of disclosure inference.

4.2.1 Data Model

In our model a *database* is represented as an edge-labeled tree. We start with an example of a simple *Medical Database* (MED-DB), shown in Figure 4.1. The database contains data of pharmacies, health insurance companies, and a medical office. Note that the database is not constrained by a schema, each data unit may have its own structure. For example, different pharmacies may store information about customers differently.

We capture trees with the symbolic *tree-expressions* (TE). For example, the TE representation of MED-DB in Figure 4.1

$\{Pharmacies \Rightarrow$

$\qquad\{Potomac\ Pharmacy \Rightarrow$

$\qquad\qquad\{Address \Rightarrow \{\ldots\},$

$\qquad\qquad Customers \Rightarrow$

$\qquad\qquad\qquad\{KenLyen \Rightarrow \{\ldots\}\}\}$

$\qquad Robinson\ Pharmacy \Rightarrow$

$\qquad\qquad\{Pharmacy\ info \Rightarrow \{\ldots\},$

$\qquad\qquad Customers \Rightarrow$

$\qquad\qquad\qquad\{Jane\ Smith \Rightarrow$

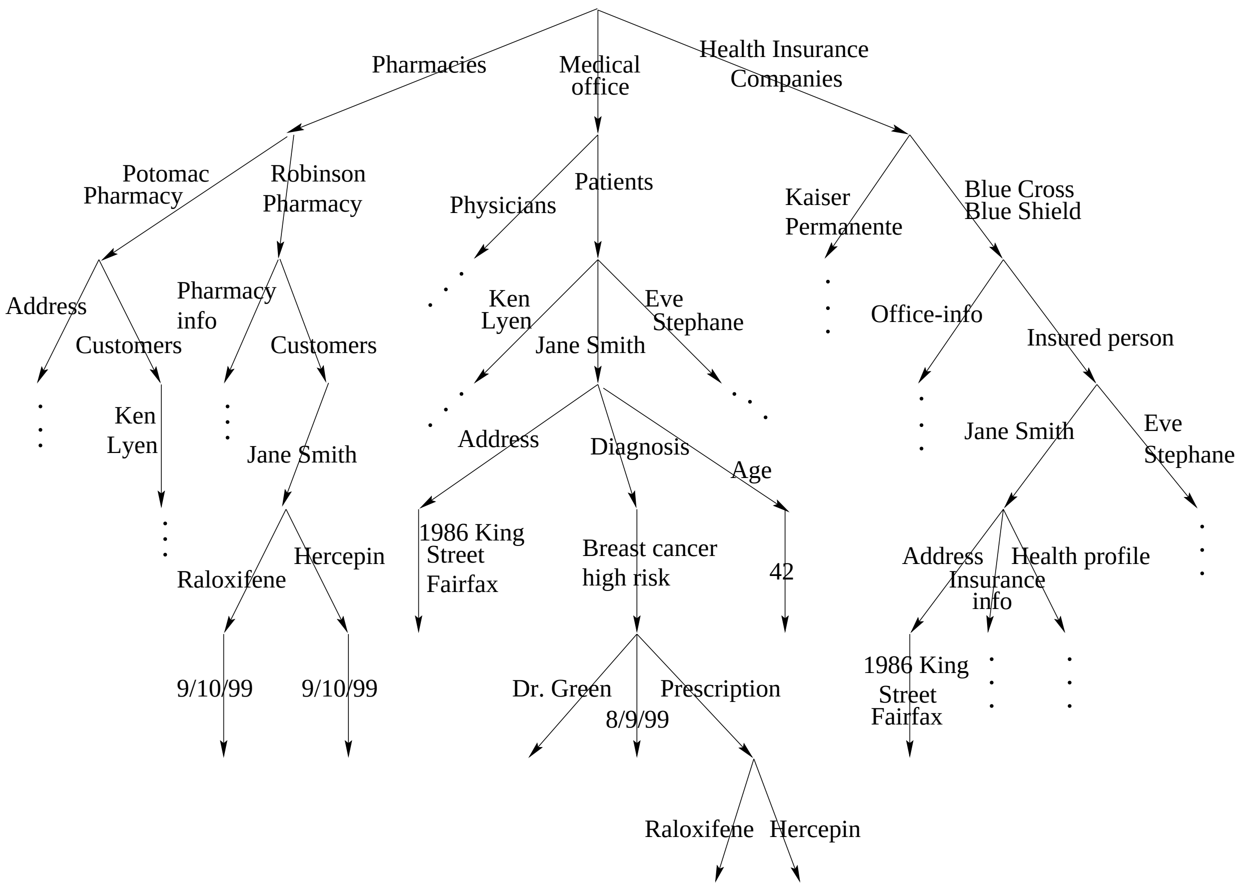

Figure 4.1: Interconnected Medical Database (MED-DB)

$\{Raloxifene \Rightarrow \{9/10/99\},$

$Herceptin \Rightarrow \{9/10/99\}\}\}\}$

$Medical\ Office \Rightarrow$

$Patients \Rightarrow$

$\{Jane\ Smith \Rightarrow \{\ldots\}\ldots\}$

More formally, tree-expressions are defined as follows:

Definition 19 (Tree-Expression, TE)

Let L be a set of labels.

1. The empty set $\{\}$ is a tree-expression (TE), called the empty tree.

2. A *tree-variable* t is a TE.

3. If $t_1, t_2, \ldots, t_n$ are TEs and $l_1, l_2, \ldots, l_n$ are in L and are all different from each other, then the set $\{l_1 \Rightarrow t_1, l_2 \Rightarrow t_2, \ldots, l_n \Rightarrow t_n\}$ is a TE. In this case we say that $\{l_1 \Rightarrow t_1, l_2 \Rightarrow t_2, \ldots, l_n \Rightarrow t_n\}$ represents the tree whose root has the outgoing edges labeled $l_1, l_2, \ldots, l_n$ to which the trees $t_1, t_2, \ldots, t_n$ are attached, respectively.

Note that in our model for every node of a TE each outgoing edge label is unique, i.e., there are no two outgoing edges with the same label. Intuitively, this limitation of our model is very similar to the hierarchical structure of file directories (folders), where each subdirectory has its unique name. From now on, we use the terms *tree-expression* (TE) and *tree* interchangeably.

Labels in TEs may be constants, null-values, or label-variables. We say that a TE is *ground* if it has no tree-variables and all of its labels are constants. A database is a ground TE. We use TEs in queries and to represent domain knowledge, query answers and disallowed information. The formal interpretation of the edge labels and their use are described in the following sections.

4.2.2 Queries

We start with an example. The following query can be imposed on the MED-DB database (Figure 4.1) to find pharmacy prescription information of the customer Jane Smith:

select $\{x \Rightarrow t\}$

from $\{\ Pharmacies \Rightarrow \{y \Rightarrow \{Customers \Rightarrow \{x \Rightarrow t\}\}\}\} \leftarrow MED-DB$

$$\textbf{where } x = \textit{Jane Smith} \tag{Q}$$

where x, y are label-variables, t is a tree-variable, and *Pharmacies, Customers* are constants (used as labels).

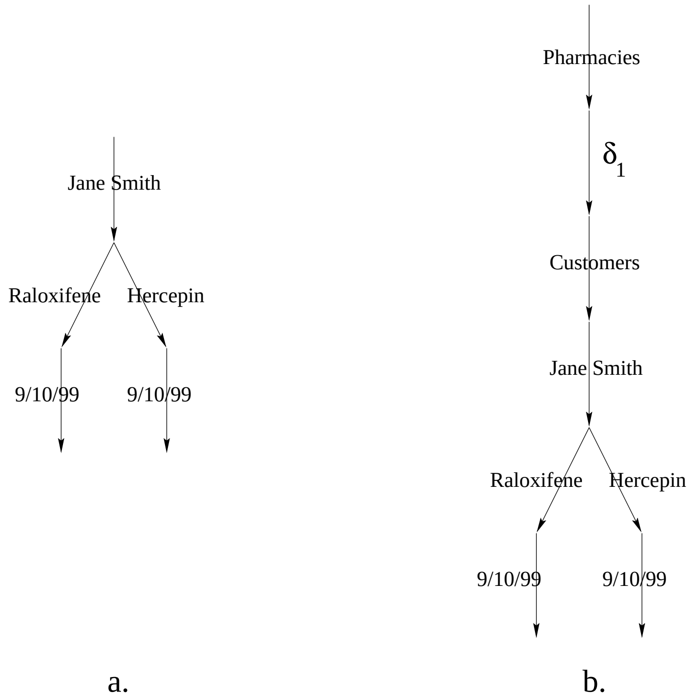

Figure 4.2: Answer (a) and Disclosed Query Data (b) of **Query** Q

Intuitively, the answer to this query is computed by finding all instantiations to the variables x, y and t that makes the TE $\{$ $\textit{Pharmacies} \Rightarrow \{y \Rightarrow \{Customers \Rightarrow \{x \Rightarrow t\}\}\}\}$ "match" the MED-DB tree in Figure 4.1. Then, only those instantiations that satisfy the **where** clause are taken (i.e., those in which x is instantiated to *Jane Smith*). For each such instantiation the structure $x \Rightarrow t$ is retrieved to construct the combined answer tree. For our example, the answer is shown in Figure 4.2.a.

More formally, we consider *select-project* queries of the following form:

$$\begin{aligned}&\textbf{select } s\\&\textbf{from } P \leftarrow DB\\&\textbf{where } c_1, \ldots, c_m\end{aligned} \tag{1}$$

where

- P and s are TEs with no null-values, i.e., they only involve constants and label- and tree-variables. It is required that all variables of s also appear in P.

- $c_1, \ldots, c_m$ are equalities among constants and label-variables that appear in P, like $x = y$, $x = 3$, where x, y are variables, and 3 are constants.

To define the semantics of queries, we need the notions of query pattern, tree-union and tree-mapping. Intuitively, a query pattern encodes P and the equalities $c_1, \ldots, c_m$ just in one TE.

Definition 20 (Query Pattern)
Given a query Q of the form (1), the *query pattern* q of Q is a TE defined by construction, as follows:

1. Initially, let q be P

2. If the conjunction of the equalities $c_1, \ldots, c_m$ in the **where** clause implies that

   (a) $x = l$ where x is a variable and l is a constant, then

   - replace all occurrences of x in q with l

   (b) $x = y$ where both x and y are variables, then

   - if only one of the variables, say x, appears in s, then replace all occurrences of y in q with x.

   - otherwise, replace one of the variables, say x, with y in q

For example, the query pattern of Q is generated from

$\{$ $Pharmacies \Rightarrow$

$\{y \Rightarrow$

$\{Customers \Rightarrow$

$\{x \Rightarrow t\}\}\}\}$

by replacing x with the constant *Jane Smith*, making it:

$\{$ $Pharmacies \Rightarrow$

$\{y \Rightarrow$

$\{Customers \Rightarrow$

$\{JaneSmith \Rightarrow t\}\}\}\}$

Next, we define the *tree-union* of two TEs.

Definition 21 (Tree-union)

Let T, T_1, T_2 be TEs. The *tree-union*, denoted by $\cup$, is defined recursively as follows*:

1. $\{\} \cup T = T \cup \{\} \stackrel{def}{=} T$

2. If $T_1 = \{l_1 \Rightarrow t_1, \ldots, l_k \Rightarrow t_k, l_{k+1} \Rightarrow t_{k+1}, \ldots, l_m \Rightarrow t_m\}$
   and $T_2 = \{l_{k+1} \Rightarrow t'_{k+1}, \ldots, l_m \Rightarrow t'_m, l_{m+1} \Rightarrow t'_{m+1}, \ldots, l_n \Rightarrow t'_n\}$
   (i.e., $l_{k+1}, \ldots, l_m$ are the common labels), then
   $T_1 \cup T_2 \stackrel{def}{=}$

$$\{l_1 \Rightarrow t_1, \ldots, l_k \Rightarrow t_k,$$

$$l_{k+1} \Rightarrow t_{k+1} \cup t'_{k+1}, \ldots, l_m \Rightarrow t_m \cup t'_m$$

$$l_{m+1} \Rightarrow t'_{m+1}, \ldots, l_n \Rightarrow t'_n\}$$

*Of course, $\cup$ is not the set union, although we use the same symbol.

Definition 22 (Tree-mapping)

Let λ be a mapping from a set LS of label-variables to constants, and from a set TS of tree-variables to ground TEs. The *tree-mapping* Λ, originating from λ is defined as a mapping from TEs with no null-values to ground TEs as follows: First, λ is extended to all symbols of T to work as the identity function on all symbols of T which are not in $LS \cup TS$. Then,

1. $\Lambda(\{\}) = \{\}$
2. $\Lambda(t) = \lambda(t)$, where t is a tree-variable
3. $\Lambda(\{l_1 \Rightarrow t_1, \ldots, l_k \Rightarrow t_n\}) = \{\lambda(l_1) \Rightarrow \Lambda(t_1), \ldots, \lambda(l_k) \Rightarrow \Lambda(t_n)\}$

Next, we define the notion of TE satisfaction by a ground TE.

Definition 23 (Ground TE satisfies a TE)

Let G be a ground TE and g be a TE with no null-values. We say that G *satisfies* g if there exists a variable mapping λ that maps all label-variables of g to constant labels of G and all tree-variables of g to subtrees of G such that $\Lambda(g)$ is a subtree of G, where Λ is a tree-mapping that originates from λ. In this case, we say that G satisfies g with respect to Λ.

Similarly, we say that a ground TE G *satisfies* a set $G' = \{g'_1, \ldots, g'_k\}$ of arbitrary TEs if there exists a *satisfying tree-mapping* Λ from $g'_1, \ldots, g'_k$ to G.

Now, we can formally define the semantics of a select-project query.

Definition 24 (Query semantics)

Let Q be a select-project query of the form given in (1), and DB be a ground TE (i.e., only contains constants). The *answer* to Q from DB is a ground TE, defined as follows:

Let $\Lambda_1, \ldots, \Lambda_k$ be all tree-mappings that satisfy q with respect to DB, where q is the query pattern of Q (Definition 20). The answer to the query is defined as follows:

$$\Lambda_1(s) \cup \ldots \cup \Lambda_k(s)$$

where $\cup$ is the tree-union.

Consider the query Q again. When the query is evaluated all *tree-mappings* that satisfy q with respect to MED-DB are generated. In this example, only one such mapping Λ exists that maps

$\{$ $Pharmacies \Rightarrow$

$\{y \Rightarrow$

$\{Customers \Rightarrow$

$\{JaneSmith \Rightarrow t\}\}\}\}$

to $\{$ $Pharmacies \Rightarrow$

$\{RobinsonPharmacy \Rightarrow$

$\{Customers \Rightarrow$

$\{JaneSmith \Rightarrow$

$\{Raloxifene \Rightarrow 9/10/99,$

$Hercepin \Rightarrow 9/10/99\}\}\}\}\}$

The tree-mapping Λ originates from the symbol-mapping λ that is:

$$\lambda(y) = Robinson\ Pharmacy$$

$$\lambda(t) = \{Raloxifene \Rightarrow 9/10/99, Hercepin \Rightarrow 9/10/99\}$$

Therefore, the *answer* to Q is:

$\{JaneSmith \Rightarrow$

$\{Raloxifene \Rightarrow 9/10/99,$

$Hercepin \Rightarrow 9/10/99\}\}$

The corresponding tree is depicted in Figure 4.2.a.

4.2.3 Constraints and Logical Entailment of Disclosure

From the perspective of privacy and security we are only interested in information about the original database that is disclosed if the query is answered; and not interested in additional structural information in the **select** clause. We define the notion of *disclosed query data* to represent the disclosed information explicitly.

Definition 25 (Disclosed Query Data)

Let Q be a select-project query of the form given in (1), and DB be a ground TE (i.e., only contains constants). The *disclosed query data* by Q from DB is a TE, defined as follows:

1. Let q be the query pattern of Q (Definition 20).

2. Let $q_1, \ldots, q_k$ be copies of q, where each copy is generated from q by replacing the variables within a copy that do not appear in s with new null values, while preserving equalities among variables (i.e., appearances of the same variable are replaced with the same new null-value).

3. Let $\Lambda_1, \ldots, \Lambda_k$ be all tree-mappings that satisfy q with respect to DB.

The disclosed query data is defined as:

$$\Lambda_1(q_1) \cup \ldots \cup \Lambda_k(q_k)$$

where $\cup$ is the tree-union and null-values in $q_1, \ldots, q_k$ are treated as if they were constants.

Similar to the computation of the answer of query (Q), the *disclosed query data* of Q is:

$\{Pharmacies \Rightarrow$

$\{\delta_1 \Rightarrow$

$\{Customers \Rightarrow$

$\{JaneSmith \Rightarrow$

$\{Raloxifene \Rightarrow 9/10/99,$

$Hercepin \Rightarrow 9/10/99\}\}\}\}\}$

The corresponding tree is depicted in Figure 4.2.b.

Definition 26 (Horn-clause constraints)

A *Horn-clause constraint* is an expression of the form

$$\forall x_1, \ldots, x_m(B_1 \wedge \ldots \wedge B_n \rightarrow H)$$

where $n \geq 1$, $B_1, \ldots, B_n$ are *tree expression* that may contain variables and constants and $x_1, \ldots, x_n$ are all the label and tree-variables in $B_1 \wedge \ldots \wedge B_n \rightarrow H$. Each H has one of the following forms:

1. H is a TE with no null-values. We require that all label- and tree-variables of H must appear in $B_1, \ldots, B_n$. In this case we say that the constraint is *tree-generating.*

2. H is an equality of the form $a_i = a_j$, there each a_i is either a constant or a label-variable that appears in $B_1, \ldots, B_n$. In this case we say that the constraint is *equality-generating.*

Consider the medical database (MED-DB) given earlier, and the domain knowledge that the drugs *Raloxifene* and *Hercepin* are used to treat patients who are likely to develop breast cancer. The combination of these drugs reduces the patient's risk to develop the disease. Individually, Raloxifene and Hercepin are also used to treat other illnesses, e.g. Raloxifene is used to treat osteoporosis too. This domain knowledge of MED-DB can be represented as Horn-clause constraint of the following from:

$$\{Pharmacies \Rightarrow \{p_1 \Rightarrow \{Customer \Rightarrow \{x \Rightarrow \mathit{Raloxifene}\}\}\}\} \wedge$$

$$\{Pharmacies \Rightarrow \{p_2 \Rightarrow \{Customer \Rightarrow \{x \Rightarrow \mathit{Hercepin}\}\}\}\} \rightarrow$$

$$\{\mathit{Medical\ Office} \Rightarrow \{Patients \Rightarrow \{x \Rightarrow \{Diagnosis \Rightarrow \mathit{Breast\ cancer\ high\ risk}\}\}\}\}$$

This constraint means that if there exists a satisfying tree-mapping Λ from the TEs

$$\{Pharmacies \Rightarrow \{p_1 \Rightarrow \{Customers \Rightarrow \{x \Rightarrow \mathit{Raloxifene}\}\}\}\}$$

and

$$\{Pharmacies \Rightarrow \{p_2 \Rightarrow \{Customers \Rightarrow \{x \Rightarrow \mathit{Hercepin}\}\}\}\}$$

to a database, then the database must contain † the subtree

$$\Lambda(\{\mathit{Medical\ Office} \Rightarrow \{Patients \Rightarrow \{x \Rightarrow \{Diagnosis \Rightarrow \mathit{Breast\ cancer\ high\ risk}\}\}\}\})$$

Definition 27 (Constraint satisfaction)

Let d be a Horn-clause constraint of the form $B_1 \wedge \ldots \wedge B_n \rightarrow H$. We say that a ground tree T satisfies d if for every tree-mapping Λ that satisfies $\{B_1, \ldots, B_n\}$ with respect to T the following holds:

1. If d is a tree-generating constraint, then Λ is a tree-mapping that satisfies H with respect to T

†The actual physical database may not contain these values, i.e., some values may be stored implicitly.

2. If d is a equality-generating constraint where H is of the form $a = b$, then $\lambda(a) = \lambda(b)$, where λ is the symbol-mapping for Λ

Consider again the previous Horn-clause constraint. Let Λ be a tree-mapping that maps
$\{Pharmacies \Rightarrow \{p_1 \Rightarrow \{Customer \Rightarrow \{x \Rightarrow Raloxifene\}\}\}\}$
to $\{Pharmacies \Rightarrow \{Robinson\ Pharmacy \Rightarrow \{Customer \Rightarrow \{Jane\ Smith \Rightarrow Raloxifene\}\}\}\}$
and $\{Pharmacies \Rightarrow \{p_2 \Rightarrow \{Customer \Rightarrow \{x \Rightarrow Hercepin\}\}\}\}$
to $\{Pharmacies \Rightarrow \{Robinson\ Pharmacy \Rightarrow \{Customer \Rightarrow \{Jane\ Smith \Rightarrow Hercepin\}\}\}\}$
The associate symbol-mapping λ is:

$$\lambda(p_1) = \lambda(p_2) = Robinson\ Pharmacy$$

$$\lambda(x) = Jane\ Smith$$

Λ also satisfies

$\{Medical\ Office \Rightarrow \{Patients \Rightarrow \{x \Rightarrow \{Diagnosis \Rightarrow Breast\ cancer\ high\ risk\}\}\}\}$

with respect to MED-DB, i.e., MED-DB contains the subtree

$\{Medical\ Office \Rightarrow \{Patients \Rightarrow \{Jane\ Smith \Rightarrow \{Diagnosis \Rightarrow Breast\ cancer\ high\ risk\}\}\}\}$

Therefore, MED-DB satisfies the above Horn-clause constraint.

Definition 28 (Logical Entailment of Disclosure)
Let $\mathcal{K}$ be a set of Horn-clause constraints, T and T' TEs with no variables (but possibly with null-values). We say that T *discloses* T' under $\mathcal{K}$, denoted as $T \models_{\mathcal{K}} T'$, if for every database DB that satisfies $\mathcal{K}$ the following holds:

If DB satisfies T (when null-values in T are treated as variables) then DB also satisfies T' (when null-values in T' are treated as variables).

4.3 Privacy Information Flow Model and Privacy Mediator

4.3.1 The Model

In this section we present a *privacy information flow (PIF) model* to represent allowed information flow within a domain and define the privacy requirements. The PIF model has two main components: 1) a *privacy information flow (PIF) graph* and 2) *individual privacy requirements* (IPR). Before giving formal definitions, we first illustrate the purpose and functionality of these components using the medical example of the Introduction. Figure 4.3 shows the PIF graph built for this example. Each node in the graph represents the type of a participant. In our example the participants are *Patient, Physician, Pharmacy, Medical lab* and *Health Insurance Company.* In general, participant-types may correspond to individuals, organizations, or their

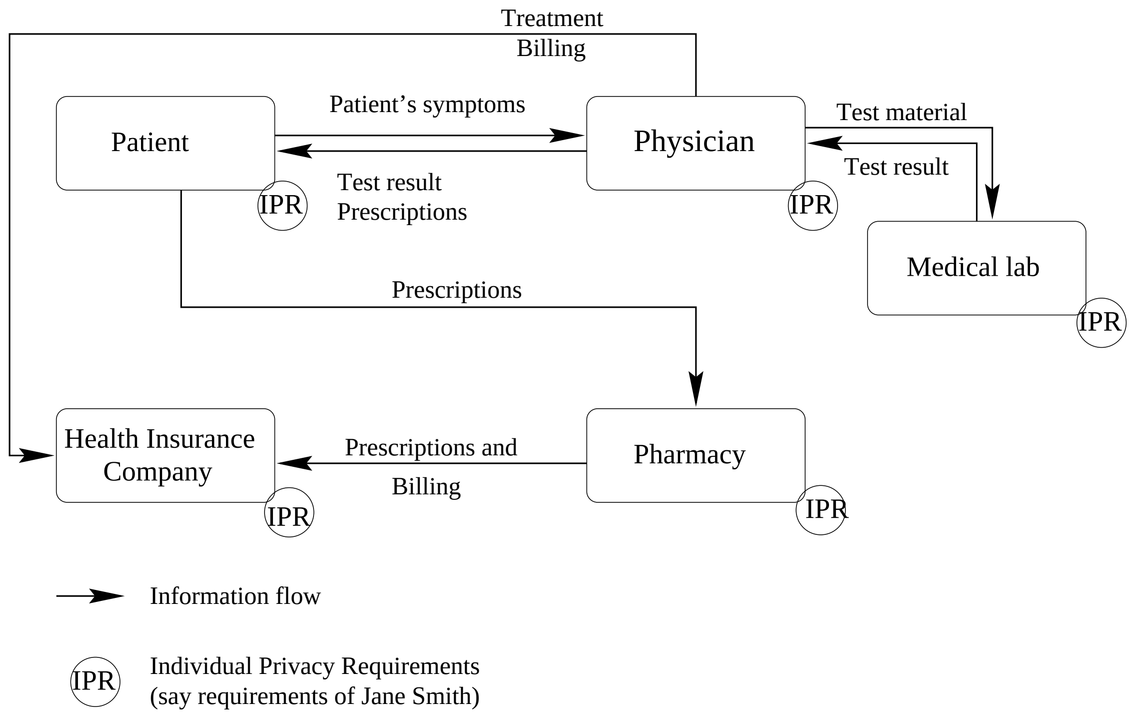

Figure 4.3: Medical Information Flow: Patient's perspective

groups that share the same properties and roles. Each arrow in the PIF graph indicates the direction of the information flow and its label T the type of the transferred information . For example, a physician may transfer *treatment* and *billing* information to the health insurance company, *test material* to the medical lab, and *test results* and *prescriptions* to the patient. Intuitively, the label T indicates that nothing but T is allowed to be transferred between the two participants. However, T may be disallowed by certain participants, e.g., individual patients, based on their individual privacy requirements (IPR's). For example, Jane Smith may disallow her physician to transfer any treatment information about her genetic and chronic diseases to the health insurance company.

Definition 29 (Privacy Information Flow (PIF) Graph)
A *PIF graph* is a directed, labeled graph $G = (V, E, \delta)$, where

1. V is a nonempty set of *participant-types* (nodes of the graph),
2. E is a set of directed edges, also called information flow channels, which are ordered pairs of nodes, i.e., $E \subseteq V \times V$.
3. δ is a mapping that associates with every edge (s, r) in E a *transferred information specification* T. Intuitively, T indicates that nothing but this information is allowed to be transferred from participant-type s to participant-type r.

Consider the PIF graph given in Figure 4.3. The set of participant-types V is $\{$ Patient, Physician, Medical lab, Pharmacy, Health Insurance Company $\}$. Some of the information flow channels are *(Patient,Physician), (Physician,Patient), (Patient, Pharmacy), (Pharmacy,Health Insurance Company), (Physician, Health Insurance Company)*, etc. The mapping δ associates *Patient's symptoms* with the directed edge

$(Patient, Physician)$, i.e., δ *(Patient,Physician)=Patient's symptoms, Prescriptions* with $(Patient, Pharmacy)$, i.e., δ *(Patient, Pharmacy)=Prescriptions*, etc.

Note that information flow among participants happens either when a participant requests some information from other participants (e.g., health insurance company submits a query to a physician to obtain information about a patient's treatments) or when a participant supplies some information to other participants without any request (e.g., a pharmacy sending billing information to a health insurance company). In both cases, the type of the transferred information is described by means of a query. For example, *Prescription and Billing* data type that corresponds to the edge $(Pharmacy, HealthInsuranceCompany)$ is represented by the query:

select $\{Pharmacy \Rightarrow y, Customer \Rightarrow x, Drugs \Rightarrow w, Date \Rightarrow z\}$

from $\{Pharmacies \Rightarrow \{y \Rightarrow \{Customer \Rightarrow \{x \Rightarrow \{w \Rightarrow z \Rightarrow \{\}\}\}\}\}\} \leftarrow MED - DB$

This means that only information of the form $\{Pharmacy \Rightarrow y, Customer \Rightarrow x, Drugs \Rightarrow w, Date \Rightarrow z\}$ is allowed to be transferred with the appropriate instantiations to x, y, z, w that satisfies q with respect to the database.

Typically, a PIF graph is designed for each domain with respect to information regarding a participant-type, e.g. patient in our example. In addition to the PIF graph, individual participants may have their special requirements.

Definition 30 (Individual Privacy Requirements (IPR))
Given a PIF graph $G = (V, E, \delta)$, an *individual privacy requirements* for an individual participant is described as a mapping m that associates with every participant-type in V a *Disallowed Inferred Information D*. D is expressed as a set of queries.

For example, the patient Jane Smith disallows that her DNA test result and high risk breast cancer diagnosis be known by a health insurance provider. The direct transfer of the test results and diagnosis is prevented by the PIF graph that does not specify them in the transferred information specifications T between medical labs and health insurance companies or between physicians and health insurance companies. Similarly to T, we use queries to represent the specific requirement about keeping breast cancer information private. The mapping *m(Health-Insurance-Company)* contains the query :

$$\begin{array}{l}\textbf{select } \{Patient \Rightarrow Jane\ Smith, Diagnosis \Rightarrow High\ risk\ breast\ cancer\} \\ \textbf{from } \{Medical\ Office \Rightarrow \{Patients \Rightarrow \{Jane\ Smith \Rightarrow \{Diagnosis \Rightarrow \\ \qquad\qquad Breast\ cancer\ high\ risk\}\}\}\} \leftarrow MED - DB\end{array}$$

4.3.2 Privacy Contract, Policy and Mediator

A *privacy contract* is composed of

1. A PIF graph $G = (V, E, \delta)$,
2. Horn-clause constraint rules $\mathcal{K}$, and
3. Individual Privacy Requirements, m

The *privacy policy* for the above privacy contract is defined as follows:

1. Every individual *participant* must be of a single participant-type $p \in V$. For example, Jane Smith is of the type *Patient*, Blue Cross Blues Shield is of the type *Health Insurance Company.*
2. For every individual participant the accumulated information A is stored, i.e., information that has been received from other entities up to the current moment.
3. A participant-type s is allowed to send a data item d to a participant r if and only if the following conditions hold:

(a) d is the form Q, which is $\delta(s, r)$ as indicated in the PIF graph, and

(b) No disallowed data item in $m(r)$ is logically entailed from $\mathcal{K}$, the prior knowledge of r and data item d.

The Privacy Mediator (PriMe) architecture, given in Figure 4.4, can be used to enforce the privacy policy on all data transfers. The *transfer control module* of PriMe is responsible for enforcing the third requirement of the privacy policy. Specifically, requirement 3.a is enforced using the information in the PIF graph; requirement 3.b is enforced using the inference engine, which uses the domain knowledge and the accumulated data to generate all logically entailed information.

To represent the functionality of PriMe, consider the scenario when Jane Smith's health insurance company requests information about her prescriptions from a pharmacy. The disclosed query data for this request is:

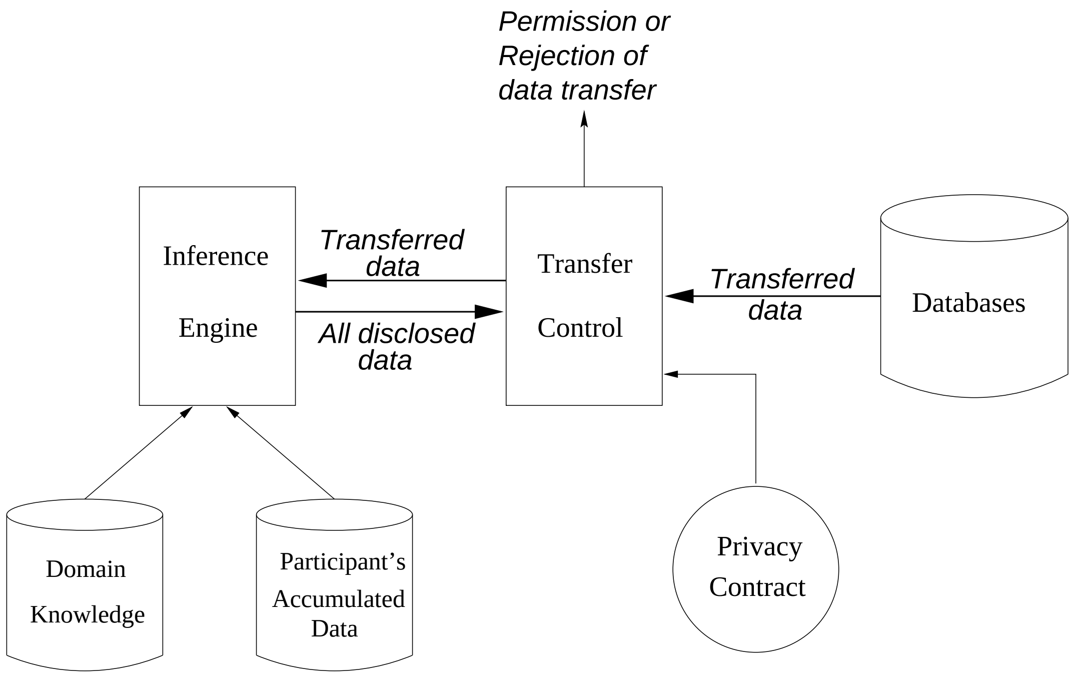

Figure 4.4: Privacy Mediator (PriMe) Architecture

$\{Pharmacies \Rightarrow$

$\{\delta_1 \Rightarrow$

$\{Customers \Rightarrow$

$\{Jane\ Smith \Rightarrow$

$\{Raloxifene \Rightarrow 9/10/99,$

$Hercepin \Rightarrow 9/10/99\}\}\}\}\}$

Next, the inference engine applies the Horn-clause constraint

$$\{Pharmacies \Rightarrow \{p_1 \Rightarrow \{Customer \Rightarrow \{x \Rightarrow Raloxifene\}\}\}\}\wedge$$

$$\{Pharmacies \Rightarrow \{p_2 \Rightarrow \{Customer \Rightarrow \{x \Rightarrow Hercepin\}\}\}\} \rightarrow$$

$$\{Medical\ Office \Rightarrow \{Patients \Rightarrow \{x \Rightarrow \{Diagnosis \Rightarrow Breast\ cancer\ high\ risk\}\}\}\}$$

on the disclosed data, and derives the TE

$$\{Medical\ Office \Rightarrow \{Patients \Rightarrow \{Jane\ Smith \Rightarrow \{Diagnosis \Rightarrow Breast\ cancer\ high\ risk\}\}\}\}$$

which is indeed in the disallowed information $m(HealthInsuranceCompany)$; therefore the transfer control mechanism of PriMe refuses the requested data transfer.

4.4 Disclosure Inference

In this section we develop an algorithm to generate a complete tree that is logically entailed from a tree under Horn-clause constraints $\mathcal{K}$. First, we state the decidability result of the inference disclosure, which is a corollary of Theorem 7 that states the correctness of the developed inference algorithm.

Theorem 6 The following problem is decidable:
Given a set $\mathcal{K}$ of Horn-clause constraints and TEs T and T', determine whether $T \models_{\mathcal{K}} T'$.

Theorem 6 will be a corollary to Theorem 7.

Definition 31 (Disclosure Cover of T under $\mathcal{K}$ ($DC_{\mathcal{K}}(T)$))
Let $\mathcal{K}$ be a set of Horn-clause constraints and T be a TE with no variables (but possibly with null-values). A TE $\mathcal{S}$ (with no variables) is called a *disclosure cover* of T under $\mathcal{K}$, denoted as $DC_{\mathcal{K}}(T)$, if $\mathcal{S}$ satisfies the following properties:

1. $\mathcal{S}$ is *sound*; i.e., $T \models_{\mathcal{K}} \mathcal{S}$
2. $\mathcal{S}$ is *complete*; i.e., for every TE T' with no variables, if $T \models_{\mathcal{K}} T'$ then $\mathcal{S}$ satisfies T', assuming that null-values of T' are treated as variables, and null-values of $\mathcal{S}$ are treated as constants.

The algorithm (Algorithm 6: Disclosure Cover) to generate the disclosure cover of a TE T under Horn-clause constraints $\mathcal{K}$ is given in Figure 4.5.

Theorem 7 (Disclosure Cover)
Algorithm 6: Disclosure Cover effectively computes a disclosure cover of T under $\mathcal{K}$, $DC_{\mathcal{K}}(T)$; i.e., the algorithm terminates and the computed $\mathcal{S}$ is (1) *sound* and (2) *complete*.

Proof: See Appendix 7.3.1 $\square$

Algorithm 6:	Disclosure Cover
INPUT	1. Ground TE T 2. Set $\mathcal{K}$ of Horn-clause constraints
OUTPUT	Disclosure cover $DC_{\mathcal{K}}(T)$ (The set $\mathcal{S}$ at the conclusion of the algorithm.)
METHOD	see below

METHOD

1. Initially, set $\mathcal{S}$ to T.

2. Apply constraints in $\mathcal{K}$ on $\mathcal{S}$, until no more changes occur or inconsistency occurred, as follows: If there exists a constraint

$$d = B_1 \wedge \ldots \wedge B_n \rightarrow H$$

and a tree-mapping Λ from $\{B_1, \ldots, B_n\}$ to $\mathcal{S}$ (when null-values of $\mathcal{S}$ are treated as constants), apply (d, Λ) on $\mathcal{S}$ as follows:

   (a) If d is an equality-generating constraint, i.e., of the form $B_1, \ldots, B_n \rightarrow a = b$ then

   - If $\lambda(a) = \lambda(b)$, where λ is the symbol-mapping from which Λ originates, then construct a symbol-mapping δ that maps $\lambda(a)$ to $\lambda(b)$, i.e., $\delta(\lambda(a)) = \lambda(b)$. Set $\mathcal{S}$ to be $\Delta(\mathcal{S})$, where Δ is a tree-mapping that originates from δ.
   - Otherwise, is one of $\lambda(a), \lambda(b)$, say $\lambda(a)$, is not a null-value then construct δ that maps $\lambda(b)$ to $\lambda(a)$, i.e., $\delta(\lambda(b)) = \lambda(a)$. Set $\mathcal{S}$ to be $\Delta(\mathcal{S})$, where Δ is a tree-mapping that originates from δ.
   - Otherwise, i.e., if both $\lambda(a), \lambda(b)$ are constants, *do nothing*. In this case, if $\lambda(a) \neq \lambda(b)$, we say that *inconsistency* occurred.

   (b) If d is an tree-generating constraint, i.e., of the form $B_1, \ldots, B_n \rightarrow H$, where H is a TE, then

   - Set $\mathcal{S}$ to $\mathcal{S} \cup \mathit{\Lambda}(H)$

3. If inconsistency occurred during Step 2 then set $\mathcal{S}$ to a special value ALL that indicates that any TE (with no variables) is disclosed.

4. Return $\mathcal{S}$ as output.

Figure 4.5: Semi-Structured Data: Disclosure Cover

CHAPTER 5

Inference in Numeric Databases

5.1 Problem Description

Multilevel security mechanisms may unnecessarily limit data availability by refusing data accesses that do not satisfy the strict security requirements. However, in a number of applications it is desirable to provide a range of valid answers that would still allow the users with limited authority to complete their jobs without revealing actual data values. For example, military support services should be able to perform scheduled maintenance work without knowing the actual mission (e.g., destination) of the serviced equipment. This paper provides a framework to define *flexible confidentiality requirements* of the data items and presents algorithms that compute *correct* and *safe* answers to queries requesting sensitive data. Intuitively, a correct query answer means that the actual data item is contained in the returned interval, and safety means that the user is not able to infer anything beyond what is released.

To illustrate an illegal inference in this setting, consider a military database that contains information about different military units, types and specifications of the vehicles, and additional information like amount of fuel used. Assume that the military plan is to use unit **A** to launch a surprise attack on the enemy. Meanwhile, units **B** and **C** create diversion by seemingly preparing to attack. The secrecy requirement is that only authorized officers of the command center and of the units **A**, **B** and **C** may know the plan. Consider the situation where an enemy spy is able to gain information about the amount of fuel used by the different units. Since usually the mileage of the vehicles and the number and types of the vehicles of the units are known, the distance and, therefore the potential destination of unit **A** can be predicted.

How can we protect such secret information? One possible solution is to keep fuel volume secret. However, this would severely limit data availability and require that all users accessing this information must be authorized to access information about the military plan as well. In this paper we present a different approach where users are allowed to access a range of possible data values but not the precise value. This flexible solution allows to preserve confidentiality and provides improved data availability.

5.2 Problem Formulation

The work reported in this paper formalizes a security model applicable to numerical databases, where a database instance is a vector of numerical values. Users (subjects) making queries are given a security classification such as secret, top secret etc. Allowed queries request numerical answers, and instead of refusing to provide an answer, for each such query and each user making a query, there is *precision* with which the answer can be released. This precision can be either a range for the numeric value or a volume representing the entity that has associated numeric values as attributes. The database may also have some publically disclosed constraints that will allow the user to derive some other consequences about the data item that is not directly released. The system presented in this paper ensures that the answers provided in terms of intervals for queries does not allow the user to infer any information that he is not authorized to obtain. Consequently, the problem is formulated in terms of the following components: (1) A data model of which the (numeric) instances consist of objects of discourse, given in terms of vectors of numerical values, (2) Arithmetic constraints imposed among their object component (numeric) values. (3) Security classification of users and data items. (4) Queries requesting numeric values.

The next section provides an example that will be used throughout this paper for illustrative purposes.

5.2.1 The Running Example

Consider the military database containing information about units **A, B** and **C**, mentioned in the Introduction. For example, the database contains information about the *number of vehicles, number of tanks, tons of food, amount of fuel, movable distance, rounds of artillery, number of soldiers* etc, as shown in Table 5.1. Some of these attributes are such as *amount of fuel* are stored, while others such as the *travel range* is derivable. In addition, the database is expected to satisfy the following constraints.

$$\begin{aligned} \text{(number of soldiers)} &\leq 10,000 \\ \text{amount of Fuel} &\leq 20,00000 \\ 5 \times \text{amount of fuel} &\leq \text{Travel range} \end{aligned}$$

Users of the database are generals, military planners, field commanders, field soldiers, supply supervisors, supply personnel, press and general public. Based on their rank and tasks, users have different access rights to data items. In general, users with their access rights form a security model, and is elaborated in Section 5.3.2.

The queries in our model request numerical values, such as *amount of fuel* used by the military unit **A**. Based on the data confidentiality (security classification) we return a range. In calculating the range, we ensure that the user cannot infer other attribute ranges such as the *movable distance* of the military unit **A** more precisely than allowed by the security model. However, we assume that all constraints are known to all users, and therefore they can make some estimates of the ranges without making queries against the database.

5.3 Data Model, Queries and Security Requirements

5.3.1 Data Model

The data model consists of two sets of attributes of numerical type, and a set of constraints among them.

Database

In our data model, an entity consists of a set of n attributes that are directly stored in the data base. Furthermore, there is a set of k derivable attributes, not stored in the database. Consequently, every entity (data item) is a $n + k$ dimensional vector $(v_1, \ldots, v_n, v_{n+1} \ldots v_{n+k})$ of numerical type. We say that n is the dimension of the data model. We use set of variables $x_1, \ldots, x_n, x_{n+1} \ldots x_{n+k}$ to represent attributes, and consequently, instantiations of variables to numeric values such as $x_1 \mid a_1, \ldots, x_n \mid a_n, x_{n+1} \mid a_{n+1}, \ldots, x_{n+k} \mid a_{n+k}$ as an instance. In the running example we have with military regiments, a potential database is given in Table 5.1. For each regiment the stored attributes are the number of soldiers, food at hand (in tons), fuel in movable storage (in million gallons) and rounds of artillery. The derivable attributes are the travel range of the regiment and the number of days it could survive without any extra supplies. As stated, the derivable attributes are not stored explicitly in the database, but can be computed from publically available statistics, and the exact values of stored attributes. Henceforth, we use both derivable and explicitly stored attributes without distinction.

Stored Attributes					Derived Attributes	
Regiment Name	Soldiers (hundreds)	Food (tons)	Fuel (mil. gals)	Artillery (mil. rounds)	Travel Range (miles)	Survivability (days)
A	10	500	10	25	400	10
B	5	600	7	35	700	15
C	1	400	7	40	1500	30

Table 5.1: A Database of Military Operations

Arithmetic Constraints

We consider conjunctions of arithmetic constraints of the following forms:

$$f(x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+k}) \geq b$$

$$f(x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+k}) > b$$

where f is an arithmetical function, $x_1, \ldots, x_{n+k}$ are variables, and b is a real number. Conjunction of a finite number of arithmetic constraints of the above form can express equalities as well. For example, the linear equality $f(x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+k}) = b$ can be expressed as a conjunction of constraints $f(x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+k}) \geq b$ and $f(x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+k}) \leq b$.

Let C be the conjunction of arithmetical constraints over variables $x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+k}$. A set of constraints define an n+k-dimensional *constraint object*, that is all possible instances of database entities that satisfy these given constraints.

Consider our previous example. Suppose that it is known that for military regiments on the move the average fuel consumption is 500 gallons per mile, and that as a policy the military maintains at least one million gallons of fuel with any regiment on the move, and as a safety precaution they do not store more than 20 million gallons of fuel either. Furthermore assume that due to logistical support problems no military regiment is moved more than 15,000 miles. Figure 5.1 shows the database $A(10, 4)$

in the constraint object (lightly shaded area). By adding constrains corresponding to other attributes, we can create a 7 dimensional region that satisfy all of them.

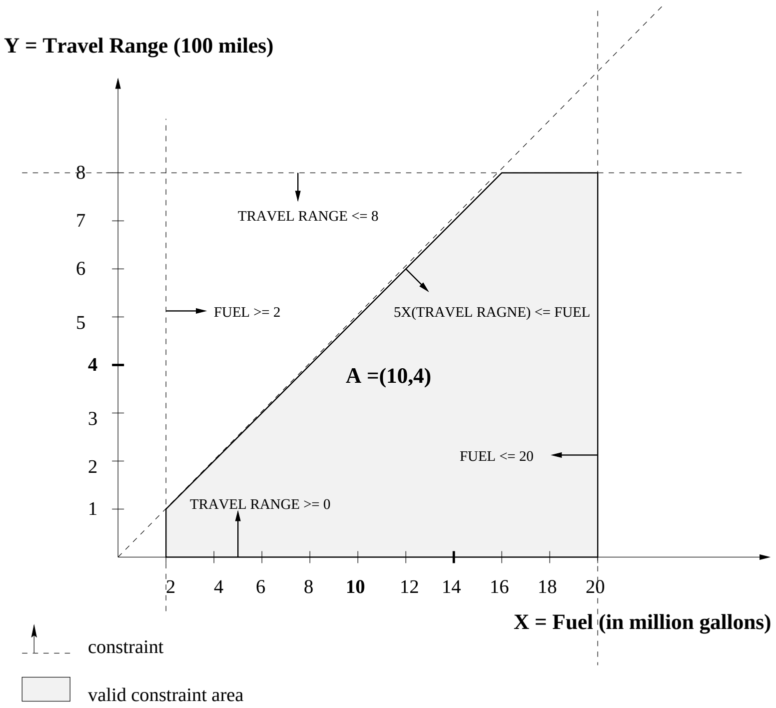

Figure 5.1: Database and constraints

Queries and Answers

We consider **queries** where a user requests a particular data value of x_i, i.e., the coordinate (a_i) of the database along the x_i axis. The *answer to the query* is an interval $I = [a_{i_1}, a_{i_2}]$ of data values, such that $a_{i_1} \leq a_i$ and $a_i \leq a_{i_2}$. The query answer can be viewed as two new constraints of the form $a_{i_1} \leq x_i$ and $x_i \leq a_{i_2}$.

The *data revealed* by the answer of the query is the new constraint object C' defined by $C \wedge (a_{i_1} \leq x_i) \wedge (x_i \leq a_{i_2})$. C' is usually a restriction of the original constraint object, where C is the original conjunction of constraints.

5.3.2 Security Model

Multi-level secure relational databases contain data classified at different security levels, providing a lattice of such levels. Users of the database have security clearances assigned to them from these levels. Access policies and enforcing mechanisms of such databases permit users access to data at or below their level of clearances. Security labels which are used to express security classifications and clearances form a lattice object with the partial order relation, called dominance relation ($\leq$), among the labels. A user is permitted to access a data item only if the user's security clearance dominates the security classification of the requested data item. Figure 5.2 shows the security lattice corresponding to our example about military battalions, which consists of the following information.

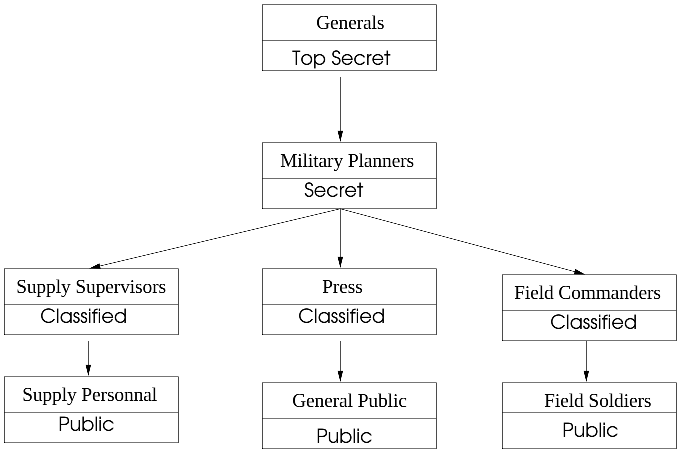

Figure 5.2: Lattice of Users

- Generals are at the top of the lattice. I.e. they are higher than every other kind of subjects, and have security level of *top secret*.
- Military planners (with security level *secret*) $\geq$ press, supply supervisors, allied commanders, all with security level *classified*.

- Press $\geq$ general public (with security level *public*).
- Supply supervisors $\geq$ supply personnel with security level *public*.

Subjects are active entities that requests operations (e.g., read value) on the passive entities (objects). *Subjects* of our model are the *users*, and each subject has a security label, referred to as its *security clearance*.

To define the security objects of our model, it is necessary to understand what information the user gains if a query is answered. Let us give an intuitive explanations before formally defining the objects and their security classifications of our model.

Originally a user knows:

- The (n+k) -dimensional constraint object C.
- That the data instances satisfy C.
- If C is bounded for a data value of x_i, then the value of x_i must be in the interval $[min(a_i), max(a_i)]$, where $min(a_i)$ and $max(a_i)$ are the smallest and largest possible values of x_i defined by C.

After Answering a Query that is an interval $[a_{i_1}, a_{i_2}]$ containing x the user knows:

- C', the restriction of C by the two new constraints $a_{i_1} \leq x_i$ and $x_i \leq a_{i_2}$.
- that the database is a point within C'.
- If C' is bounded for a data item x', than the value of x' must be in the interval $[a', a'')]$, where a' and a'' are the smallest and largest possible values of x' defined by C'.

We assume that all instances of an entity within a constraint space to have the same probability to be in the database, i.e., the database is *uniformly distributed* within the constraint object. Clearly, the smaller a constraint object is the more accurate information the user has about the actual values of the data items. Since the constraint object represents the user's knowledge about the database we use its properties, such as volume, projection, to define the security requirements on the data items.

Definition 32 (Volume and Projection of a constraint Object)
Let C be an n+k -dimensional constraint object. The *volume* of C is defined as the $(n+k)$ dimensional volume of the n-dimensional geometrical shape. The *projection* of C on axis x is an interval $[a_1, a_2]$, where a_{i_1} and a_{i_2} are the minimal and maximal values along the x axis defined by C.

In this papers we present two approaches to define the *security* of objects in our database. First, we use the *volume* of C as the security object. While this method is simple and can be easily implemented, it does not allow to assign different sensitivity levels to the different data items. [*] Second, we propose a security classification based on the *projection* of C to represent the security object. Moreover, we attach probability measurement to incorporate the shape of the constraint object, which is not reflected in the projection. The second approach enables us to assign different security classifications to the different data items as well as represent partial disclosures. Following sections have the details.

5.4 Security by Volume

As mentioned previously, the volume of the constraint object represent how accurately a user knows the database. In this section we present a security model in which we use absolute volumes of the constraint objects to represent the sensitivity of the database.

[*] Same as database level of security granularity in relational databases.

Definition 33 (Volume Objects)

Let $L = l_1, \ldots, l_k$ be a set of security labels and C a constraint object with volume V. For each security label $l_i,\ i = 1, \ldots, k$ we create a *volume object*, denoted by V_{l_i}, as follows:

1. $V_{l_i} \leq V$, i.e., the volume object must be smaller than the volume of the constraint object.

2. If $l_i \leq l_j$ then $V_{l_i} \geq V_{l_j}$, i.e., volume objects at lower security levels can not be smaller than volume objects at higher security level.

The secrecy of the database is violated if a user with security clearance l_i is able to disclose a constraint object with volume less than V_{l_i}. We propose a security mechanism based on controlling the *volume* of the constraint object disclosed by the user. For this approach it is necessary that the original constraint object C is bounded from every direction.

To visualize the security classifications, consider Figure 5.1. The shaded area define valid data values. Since knowing that the database must satisfy the four constraints, the user knows that the value of X (fuel in million gallons) must be within $[2, 20]$, and consequently, Y (the travel range) must be within $[0, 8]$ and that any data instance of an entity (corresponding to a military regiment) in the database is in the constraint object. Originally the volume V of the constraint area is 81.

Assume, that in addition to the constraints of the database, the user knows that X, the amount of fuel in million gallons must be in $[8, 14]$. Clearly, this user's knowledge about X is more accurate now than before. The darkly shaded area of Figure 5.3 shows the new constraint object C' after the addition of $8 \leq X \leq 14$. The volume of C' is 33. The increase of the user's knowledge about the data items

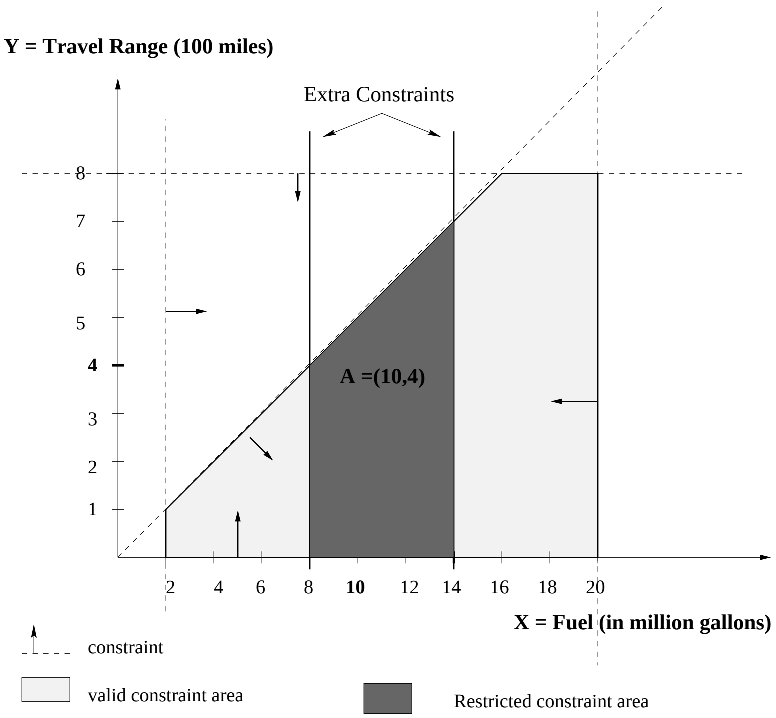

Figure 5.3: Restricted constraint object

can be represented by the decrease of the size (volume) of constraint object, that is $81 - 33$. Observation of the reduced constraint object indicates that the user also gained information about Y, the possible travel range of the regiment. C' indicates that the value of Y, the possible travel range must be within $[4, 7]$. Notice that for some adverserial activity, possible travel range of a an enemy regiment to this accuracy may be sufficient.

Definition 34 (Correctness, Safety and Efficiency of Query Answer by Volume)
Let DB be the database, x the data item in DB requested by the user, l the user's security clearance, C the constraint object known by the user to contain the database, and V_l the volume object at security level l. A query answer $[a_1, a_2]$ is correct and safe if

1. *Correct*: $a \in [a_1, a_2]$, i.e., the actual data value of x is within the returned interval.

2. *Safe:* the query answer does not violate the security requirements, i.e., if C' is the new constraint object $C \wedge (a_1 \leq x) \wedge (x \leq a_2)$ with volume V' then:

   (a) $V' \geq V_l$ and

   (b) DB is uniformly distributed in C', i.e., the user knows only that the data item satisfies C'.

3. *Efficient:* Let C be the constraint object known by the user u, l_2 the security clearance of u, and V_{l_1} and V_{l_2} be the volume objects at security levels l_1 and l_2, respectively, where $l_1 \leq l_2$. An *efficient query answer* for u requesting data item x is one that satisfies $C' = C \wedge (a_1 \leq x) \wedge (x \leq a_2)$ then $V_{l_2} \leq V' \leq V_{l_1}$, where V' is the volume of the restricted constraint object.

Algorithm 7 given in Figure 5.4 generates a correct, safe and efficient query answer. It uses a random number generator assumed to be uniform.

Theorem 8 Algorithm 7 (1) *terminates* and returns a (2) *correct*, (3) *safe* and (4) *efficient* according to Definition 34.

Proof: See Appendix 7.4.1. □

Returning back to our running example, assume that the answer given as a result of a query made by the *press* for the amount of fuel used by regiment **A** is $[8, 14]$, as given by Figure 5.3. Furthermore assume that the volume associated with this security level is 30, and by levels lower than that of the *press* (say the general public) is 35, and the volume of the restricted constraint object as given is 33. Hence the answer is correct, safe, and efficient (but not optimal).

Algorithm 7:	Query Answer by Volume
INPUT	• User's query Q requesting data item x • User's previous query answers $q_1, \ldots, q_m$ • User's security clearance l • Constraints object C known by the user to contain the DB • Database point DB • Volume objects $V_{l_1}, \ldots, V_{l_h}$
OUTPUT	• Answer to Q • Restricted constraint object C'
METHOD	1. **If** $q_1, \ldots, q_m$ contains an answer $[a'_1, a'_2]$ for data item x, **then** (a) Return $[a'_1, a'_2]$ as the answer. (b) *Generate* C' as $C' = C$. 2. *Count* attempts to generate answer **If** *count(r)* $>$ 1000 **then** Return $[a', a'']$ as default answer. **Else** *Generate* two points a_1 and a_2 as follows: (a) Let a be the actual value of data item x (b) Let a' be the smallest and a'' the largest possible value of x defined by C. (c) *Randomly* pick a_1 from the interval $[a', a]$ (d) *Randomly* pick a_2 from the interval $[a, a'']$ 3. *Generate* two linear constraints: (a) $a_1 \leq x$ (b) $x \leq a_2$ 4. *Generate* the new constraint object $C' = C \wedge (a_1 \leq x) \wedge (x \leq a_2)$ 5. *Calculate* volume V' of C' 6. *Verify* security requirements: I. **If** V' smaller than V_l (security requirement is violated) **then** (a) Return to 2 II. **Else** (security is not violated) **then** (a) **If** $V_{l'} \leq V'$, where $V_{l'}$ is the volume object at security level l' such that $l' \leq l$ **then** (more accurate answer is possible) i. Return to 2 (b) **Else** i. *Return* $[a_1, a_2]$ as the answer to the query ii. *Store* C' as the new constraint object known by the user.

Figure 5.4: Query Answer by Volume

5.5 Security by Interval

Similarly to the previous section, the information disclosed by a user is reflected in the *constraint object* by this users. However, in this section we explore a flexible security classification that is based on the projection of the constraint object on the axis. The proposed method allows the security officer to individually classify each data item.

First, let us give an intuitive example. Consider the database and constraint space we presented previously. Assume, that data item x_1 corresponding to the amount of fuel is more sensitive than data item X_2, say the amount of food; for example, only top-secret users are allowed to know the exact value of x_1, while x_2 may be known by users with secret or top-secret security clearances. The security requirement is that users with secret clearances may only know that the value of x_1 is within an interval of size 3. For example, since the actual value of x_1 is 10, a secret user may know that know that x_1 is in the interval $[8.5, 11.5]$. However, if the user knows that x_1 is within $[9.5, 10.5]$ then the security is violated, as that precision may be sufficient for him to derive unauthorized facts. The situation is shown in Figure 5.5.

Moreover, it is possible, that the security officer would like to consider disclosures, where the data item is not fully disclosed within a given interval size but has high probability to be there. Note, that we assumed that originally the database is uniformly distributed within the constraint object C. Because of this uniform distribution, the probability that the database is within a sub-structure C' of C is proportion to the volumes of C and C'. For example, if the the volume of C is V and the volume of C' is V', the probability that the database is contained in C' is V'/V. This observation allows the security officer to the data items not only from certain but from disclosures with high probabilities. For example, assume that the

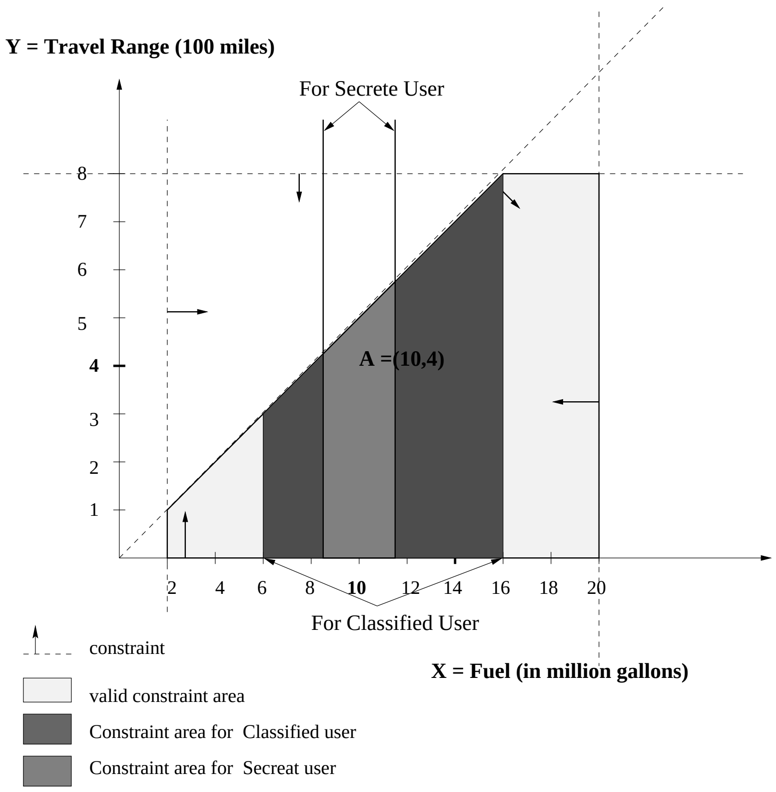

Figure 5.5: Interval Based constraints

security officer would like to ensure that the users can't guess the value of x_1 with 85 accuracy within an interval of size 2. Consequently, the user can not disclose a constraint object C, with volume V, such that C has a sub-object C', with volume V' and projection size 2 on the x_1 axis, such that $V'/V > 0.85$. To represent this type of security requirements we propose a new security object, called *interval object*.

Definition 35 (Interval objects)

Let $x_1, \ldots, x_{n+k}$ be a database, $l_1, \ldots, l_h$ a set of security labels and C a constraint object with volume V. For each data item x_i $i = 1, \ldots, n{+}k$ and for each security label l_j, $j = 1, \ldots, h$ we create an *interval object*, denoted by (I_{i_j}, p_{i_j}) $i = 1, \ldots, n + k, j = 1, \ldots, h$, as follows:

1. I_{i_j} is an interval size, that defines the minimal interval size allowed to be known by the users to contain x_i with probability p_{i_j}.

2. If $l_s \leq l_r$ and $p_{i_s} = p_{i_r}$ then $I_{i_s} \geq I_{i_r}$, i.e., interval size of an objects at a low security level can not be smaller than the interval size of the objects at high security level.

3. If $l_s \leq l_r$ and $I_{i_s} = I_{i_r}$ then $p_{i_s} \leq p_{i_r}$, i.e., if two interval sizes of a data item are the same at different security levels, then the high security level can not have lower probability assigned to than the low security level.

Given a data item x and its interval object (I, p) at security level l, the secrecy of x is violated if a user with security clearance l is able to disclose a constraint object C such that C contains a sub-object C', with projection size on x smaller than or equal to I, that contains the database with probability higher than p.

Definition 36 (Correct and Safe Query Answer - Interval)

Let $x_1, \ldots, x_{n+k}$ be the database, x_i the data item in DB requested by the user, l the user's security clearance, C the constraint object known by the user to contain the database, and $(I_{1_l}, p_{1_l}), \ldots, (I_{n_l}, p_{n_l})$ the interval objects of data items $x_1, \ldots, x_i, \ldots, x_{n+k}$ at security level l. A query answer $[a_1, a_2]$ is correct and safe if

1. *Correct*: $a \in [a_1, a_2]$, i.e., the actual data value f x_i is within the returned interval.

2. *Safe*: the query answer does not violate the security requirements, i.e., if C' is the new constraint object $C \wedge (a_1 \leq x_i) \wedge (x_i \leq a_2)$ with volume V' then

   (a) C' does not contain a sub-object C'_j with projection size smaller than of equal to I_{j_l} $j = 1, \ldots, n+k$ and volume V'_j, such that V'_j/V' greater than p_{j_l} and

(b) DB is uniformly distributed in C', i.e., the user knows only that the database is any point within C'.

We propose a security mechanism to control the *size of the interval* returned to the user as the answer to the query; thus ensuring that the security requirements are not violated. This approach does not require that the original constraint object is bounded.

Coming back to our running example about military convoys, lets assume that that due to the sensitive nature of possible targets, the security officer has determined that the minimum disclosure length of the travel range allowable for the *classified* security level is 500 miles. Consequently, the data disclosed in Figure 5.3 is in violation of this new requirement. Consequently, an interval of $[6, 16]$ could be disclosed for the Fuel so that the travel range could be protected by the appropriate amount of 500, as shown in Figure 5.5.

Theorem 9 Algorithm 8 (1) *terminates* and returns a (2) *correct* and (3) *safe* query answer according to Definition 36.

Proof: See Appendix 7.4.2 □

Note, that since we do not enforce any restriction on the correlation of interval size and probability we can't define efficient answer similarly to the volume based model.

Algorithm 8:	Query Answer by Interval
INPUT	1. User's query requesting data item x 2. User's previous query answers $q_1, \ldots, q_m$ 3. User's security clearance l 4. Constraints object C known by the user 5. Database point DB 6. Set S of Interval objects of the form (I_l, p_l)
OUTPUT	1. Answer to Q 2. Restricted constraint object C'

METHOD

1. **If** $q_1, \ldots, q_m$ contains an answer $[a'_1, a'_2]$ for x **then**
   (a) Return $[a'_1, a'_2]$ as the answer
   (b) Generate C' as $C' = C$
2. Let a' and a'' be the smallest and largest possible values of x defined by C.
3. *Count* attempts to generate random length r
   If $count(r) > 1000$ **then** Return $[a', a'']$ as the default answer.
   Else *Generate* a random length r
4. Randomly *pick a point* a_1 in the interval $[a-r, a]$ (starting point of the answer), where a is the actual value of x in the database.
5. **If** $a_1 < a'$ **then** (security is violated) Return to Point 3.
6. *Count* attempts to generate random length r'
   If $count(r') > 1000$ **then** Return $[a', a'']$ as the default answer.
   Else *Generate* a random length r'
7. **If** $r' \leq r$ (incorrect answer) or $r' < I_l$ (security is violated) **then** Return to Point 6 (i.e., incorrect answer)
   Else generate $a_2 = a_1 + r'$ (end point of the answer)
8. **If** $a_2 > a''$ **then** (security is violated) Return to Point 6.
9. *Generate* two linear constraints:
   (a) $a_1 \leq x$
   (b) $x \leq a_2$
10. *Generate* the new constraint object $C' = C \wedge (a_1 \leq x) \wedge (x \leq a_2)$
11. *Calculate* volume V' of C'
12. *Verify* security requirements for every data item y in the DB
    If there exist a sub-object C'_i of C' such that
    (a) $\mid C'_i \mid_y \leq I_{y_l}$, where $\mid C'_i \mid_y$ is the projection of C'_i on y and I_{y_l} is from $(I_{y_l}, p_{y_l}$, assigned to y at security level l, and
    (b) $V'_i / V' > p_{y_l}$
    then (security is violated) Return to Point 3
13. **Else** (security is not violated)
    (a) *Return* $[a_1, a_2]$ as the answer to the query
    (b) *Store* C' as the new constraint object known by the user.

Figure 5.6: Query Answer by Interval

CHAPTER 6

Discussions and Conclusions

This dissertation is the first to give a systematic characterization of the inference problem in terms of

1. Data-dependent versus data-independent disclosure
2. Assurance of disclosure inference via soundness and completeness algorithms
3. Expressiveness of data models and constraints
4. Design-time versus query-time approaches

Formal characterization of disclosure inference algorithms is necessary to establish their properties with respect to confidentiality and data availability. Without these characterizations, disclosure inference algorithms may only be compared via empirical results. The main technical difficulty of the research was the development of sound and complete inference algorithms for a variety of

- settings: relational, semi-structured and numeric databases
- constraints: functional, multi-valued, and join dependencies, domain knowledge and arithmetic constraints
- operation modes: design-time versus query-time

The final research contributions of the dissertation surpass the results of related research works regarding expressiveness of settings and constraints. We conclude by listing some suggestions for continuing research:

Relational databases

In this dissertation we assumed that the database is static. However, most real databases change over time. Clearly, updates affect the possible inferences the user is able to perform. To assure confidentiality and allow maximal data availability, we need to keep track of these updates and incorporate them in the answers the users receive. Our main focus is to prevent the confidentiality of data. Note that the case when users are able derive "wrong" inferences are outside the focus of our interest. For example, consider the following scenario, where t_1, t_2 and t_3 represent time and $t_1 < t_2 < t_3$:

t_1 User receives tuple: $<$ Brown, Clerk $>$

t_2 Brown got promoted to "Manager"

t_3 User receives tuple: $<$ Clerk, 34,000 $>$

Since Brown got promoted before the user received the second answer, no information about Brown's salary is disclosed. On the other hand, if the user is unaware of the update, s/he may wrongly infer that Brown's salary is $ 34,000.

Currently we focus on multiple attacks of a single user. However, in a real life environment it is possible that malicious users share their information to obtain data for which they do not have the proper authorization. With little modification, DiMon can protect against collaborating users as well. From the perspective of users' interaction, we distinguish among the following types of attacks:

1. Single user attacks, where each user is considered individually.

2. Attacks of collaborating users where all of the users are assumed to collaborate with each other.

3. Attack of collaborating groups of users, where only certain groups of users, with the same security clearances, are assumed to share their information. Two types of these attacks are considered:

   (a) Attacks of a predetermined number of users.

   (b) Attacks of a predefined groups of users.

Another future direction is to incorporate the problem of data aggregation into the model [MMJ96]. The aggregation problem occurs if a certain number (threshold) of data items can be released safely while, if the size of the aggregate exceeds this number, the security is violated. Some of the aggregation control mechanisms ensure security by monitoring the number of tuples released to users. If the number of released tuples exceeds the threshold value by answering a new query, then the query is rejected. We believe that database dependencies, especially tuple generating dependencies, should be incorporated to provide secure aggregation control. Note that the inference problem can be viewed as a special type of aggregation problem with zero threshold value.

Semi-structured databases

Currently we consider a tree-like data structure. This can be extended to accommodate graph structures with cycles. Also, in our model the structure of the tree is preserved through explicit label values (constants), thus in the presence of label-variables and null-values, we cannot tell whether two outgoing edges from the same node are different or they correspond to the same edge.

Next, variables that are used in the TE in the **from** clause of a select-project query, may function as wild card symbols, i.e., they correspond to any edge label of the database. However, they are limited to match a single edge only and not an arbitrary path. While it is possible, with the knowledge of the data structure, to use a set of select-project queries to mimic a query with variables corresponding to an arbitrary path, it is cumbersome. A future extension of our model would be to incorporate path-variables that matches any path of arbitrary length.

Finally, Algorithm 6 generates all disclosed data using actual data items (trees). Since it might be computationally expensive, in some applications it might be desirable to establish possible inferences using only the past and present queries and the database constraints.

Numeric databases

Algorithms 7 and 8 provide an effectively accurate answer that is not optimal, i.e., the released answer in not the most accurate value the user is allowed to obtain. While we believe it is impossible to provide a general algorithm that generates safe, correct, and optimal query answer, such a method may exist for a restricted family of arithmetic constraints.

Also, in this work we considered queries that request a single data value. This work could be extended to incorporate queries requesting several data items simultaneously and incorporate selection conditions in queries. Finally, the interval based accuracy can handle sets of sensitive intervals and probabilities. For example, a data item x may be accessed by a user only if none of the classifications $(i_{x_1}, p_{x_1}), \ldots, (i_{x_k}, p_{x_k})$ is violated.

CHAPTER 7

Appendix

7.1 Proofs of Theorems and Propositions in Section 3.4

7.1.1 Proof of Proposition 1

1. ($\Leftarrow$) Assume that $C_e \models C'_e$. We want to show, that $(PF, \Pi_Y \sigma_C) \models (PF', \Pi_{Y'} \sigma_{C'})$. Let $PF \in \Pi_Y \sigma_C(r)$ for a relation r. Then, there must exist a tuple $t \in r$ such that $PF = \Pi_Y \sigma_C(t)$, and therefore, t satisfies C_e. Because $C_e \models C'_e$, t must also satisfies C'_e, that includes all of the equalities explicitly given in PF'. Thus, $PF' = \Pi_{Y'} \sigma_{C'}(t)$ and $PF' \in \Pi_{Y'} \sigma_{C'}(r)$, which completes the proof of $\Leftarrow$.

($\Rightarrow$) Assume, that $(PF, \Pi_Y \sigma_C) \models (PF', \Pi_{Y'} \sigma_{C'})$. We want to show that $C_e \models C'_e$; i.e., for every tuple t, if t satisfies C_e then t must satisfy C'_e. Consider an arbitrary tuple t that satisfies C_e. Let r be a relation that contains the single tuple t, such that $PF = \Pi_Y \sigma_C(t)$. Then, by Definition 7, we know that $PF' \in \Pi_{Y'} \sigma_{C'}(r)$. Since r contains only one tuple, t, $PF' = \Pi_{Y'} \sigma_{C'}(t) = \Pi_{Y'} \sigma_{C'_e}(t)$, and t must satisfy C'_e. This completes the proof of $\Rightarrow$.

2. The proof of (2) immediately follows from (1) because the logical entailment $\models$ on constraints is transitive.

7.1.2 Proof of Proposition 2

Assume that $\mathcal{P}_1 \models_{\mathcal{D}} \mathcal{P}_2 \wedge \mathcal{P}_2 \models_{\mathcal{D}} \mathcal{P}_3$. Let r be a relation that satisfies $\mathcal{D}$. Assume also that for every $(P_i, \Pi_{Y_i}\sigma_{C_i}) \in \mathcal{P}_1$ $(i = 1, \ldots, n)$, $P_i \subseteq \Pi_{X_i}\sigma_{C_i}(r)$. Then, by Definition 7, every $(P_j, \Pi_{Y_j}\sigma_{C_j}) \in \mathcal{P}_2$ $(j = 1, \ldots, m)$, $P_j \subseteq \Pi_{X_j}\sigma_{C_j}(r)$. Moreover, because $P_j \subseteq \Pi_{X_j}\sigma_{C_j}(r)$ holds for every $(P_j, \Pi_{Y_j}\sigma_{C_j}) \in \mathcal{P}_2$ $(j = 1, \ldots, m)$, by Definition 7, $P_l \subseteq \Pi_{X_l}\sigma_{C_l}(r)$ must hold for every $(P_l, \Pi_{Y_l}\sigma_{C_l}) \in \mathcal{P}_3$,$(l = 1, \ldots, k)$. This completes the proof of Proposition 2.

7.1.3 Proof of Theorem 2

First, Algorithm 3 must terminate since the applications of dependencies do not introduce new constants or symbols, and, therefore, there is only a finite number of tuples that can be generated. We need to show that the set $\mathcal{S}$ computed by the algorithm is (1) sound, (2) complete and (3) compact.

(1) Proof of soundness: Let $\mathcal{S}$ be the set produced by Algorithm 3. Because data-dependent disclosure is transitive (Proposition 2), to prove *soundness* of $\mathcal{S}$ it is sufficient to prove that every QA-pair $(P, \Pi_Y\sigma_C)$ in $\mathcal{S}$ is disclosed; i.e., $\mathcal{P} \models_{\mathcal{D}} (P, \mathit{\Pi}_Y\sigma_C)$. This is based on the following claim:

Claim 1: For every relation r that satisfies $\mathcal{D}$ there exist symbol mappings (preserving equalities and constants) from the tuples of the relations r_i, r'_i, and r'', which were generated by the algorithm in Step 1a, 1c, and 2 respectively, to the tuples of r.

Before proving Claim 1, we complete the proof of soundness. If *inconsistency* occurred in Step 2 then, by Claim 1, there does not exist r that satisfies $\mathcal{D}$ such that $P_i \subseteq$

$\Pi_{X_i}\sigma_{C_i}(r)$ for all $i = 1, \ldots, m$. Therefore, $\mathcal{S} = \{(ALL, \mathit{\Pi}_R\sigma_{TRUE})\}$ is vacuously disclosed. If *no inconsistency* occurred then, by using Claim 1, every $(PF, \Pi_Y\sigma_C)$ generated in Step 3 of the algorithm is indeed disclosed. Finally, Step 4 of Algorithm 3 clearly leaves the set $\mathcal{S}$ sound.

The only remaining part is the *proof of Claim 1*, for which we will use the following claim:

Claim 2: Let r_1 be a relation, that possibly contains null values, and r_2 be a regular relation, such that r_1 and r_2 are over the same schema and r_2 satisfies a set of dependencies DD. If there exists a symbol mapping (preserving equalities and constants) form r_1 to r_2, then there also exists a symbol mapping from $Chase_{DD}(r_1)$ to r_2, where $Chase_{DD}(r_1)$ is the relation obtained by applying the dependencies DD on r_1.

Before proving Claim 2, we complete the proof of Claim 1. Clearly, there is a mapping from r_i to $\Pi_{X_i}\sigma_{C_i}(r)$ and $\Pi_{X_i}\sigma_{C_i}(r)$ satisfies D_i. From this and Claim 2 it results that there is a symbol mapping from $r'_i = Chase_{D_i}(r_i)$ to $\Pi_{X_i}\sigma_{C_i}(r)$ and therefore to r. But then, there are symbol mappings $p_1, \ldots, p_m$ from every $r'_i \;\; i = 1, \ldots, m$ to r. Since two different relations r'_i and r'_j cannot share a common null value, because of the way r_is are constructed in Step 1a and the fact that $Chase_{D_i}$ only operates on single relations, the union of the symbol mappings $p_1 \cup \ldots \cup p_m$ constitutes a symbol mapping from $r' = \bigcup_{i=1}^{m} r'_i$ to r. But then, by Claim 2, there exist a symbol mapping from $r'' = Chase_{\mathcal{D}}(r')$ to r.

The only remaining part is *Claim 2*, which we prove by induction on the number of application of dependencies on r_1. We know that originally, there is a symbol mapping from r_1 to r_2 and that r_2 satisfies the dependencies DD. Assume, that dependencies $d_1, \ldots, d_k$ of DD were applied on r_1 and that, by induction hypothesis, there exists a symbol mapping p^k from $r_1^k = Chase_{d_1,\ldots,d_k}(r_1)$ to r_2. We want to show, that after the application of the next dependency d_{k+1} on r_1^k there still exists a symbol mapping p^{k+1} from r_1^{k+1} to r_2.

Let tuples $t_{1_1}, \ldots, t_{1_l}$ in r_1^k satisfy the body of d_{k+1}; i.e., there exists an atom mapping h from the body of d_{k+1} to the tuples $t_{1_1}, \ldots, t_{1_l}$, and let tuples $t_{2_1}, \ldots, t_{2_l}$ be their corresponding tuples in r_2; i.e., $p^k(t_{1_i}) = t_{2_i}\ for\ all\ i = 1, \ldots, l$. If d_{k+1} is a(n)

1. *equality-generating dependency* of the form $B_1, \ldots, B_n \to a = b$ then $h(a)$ and $h(b)$ are the two attribute values of r_1^k which are equated, say all occurrences of $h(a)$ are replaced by $h(b)$. We define p^{k+1} as follows:

   - If tuple t in r_1^{k+1} was created from tuple t' in r_1^k, then $p^{k+1}(t) = p^k(t')$. (Note, this includes the case when $t = t'$.)

   We claim, that p^{k+1} is indeed a symbol mapping; i.e., it preserves equalities and constants. For this, let $p^k(h(a)) = c_1$ and $p^k(h(b)) = c_2$, where c_1 and c_2 are constant values of r_2. Since r_2 satisfies d_{k+1} and p^k preserves equalities and constants, c_1 must be equal to c_2 and, therefore, p^{k+1} is a symbol mapping from r_1^{k+1} to r_2.

2. *tuple-generating dependency* of the form $B_1, \ldots, B_n \to R[A_1 = a_1, \ldots, A_n = a_n]$ and a new tuple $(h(a_1), \ldots, h(a_n))$ is added to r_1^k. We define p^{k+1} as follows:

   - For the tuples in r_1^k, p^{k+1} coincides with p^k.
   - For the new tuple, $p^{k+1}((h(a_1), \ldots, h(a_n))) = (p^k(h(a)), \ldots, p^k(h(a_n))) = t$

   We claim, that t is indeed in r_2. This follows from the fact, that r_2 satisfies d_{k+1} and that p^k preserves equalities and constants.

This completes the proof of the soundness of the algorithm.

(2) Proof of completeness: If *inconsistency* occurred then completeness trivially follows since $(ALL, \Pi_R\sigma_{TRUE})$ dominates every QA-pair.

If *no inconsistency* occurred then let us suppose that a QA-pair $(PF, \Pi_Y\sigma_C)$ is disclosed from $\mathcal{P} = \{(P_1, \mathit{\Pi}_{X_1}\sigma_{C_1}), \ldots, (P_m, \mathit{\Pi}_{X_m}\sigma_{C_m})\}$ by $\mathcal{D}$. We construct a relation r from r'' at the end of Step 2 by replacing each unique null value δ_i with a new unique constant while preserving equalities. It is clear, that r satisfies $\mathcal{D}$ and $P_i \subseteq \Pi_{X_i}\sigma_{C_i}(r)$ *for all* $i = 1, \ldots, m$. But then, by the definition of data-dependent disclosure $PF \in \Pi_Y\sigma_C(r)$, meaning that in Step 3 the algorithm should have produced a QA-pair q which dominates $(PF, \Pi_Y\sigma_C)$. Since, by construction of $\mathcal{S}$ in Step 4 of the algorithm, a created QA-pair $q' \in \mathcal{S}$ must dominate q, therefore, q' must also dominate $(PF, \Pi_Y\sigma_C)$. This completes the proof of completeness.

(3) Proof of compactness: is given by contradiction. Assume that $\mathcal{S}$ is not compact. Clearly, by construction of $(PF, \Pi_Y\sigma_C)$ in Steps 3.a,b, and c, it is in normal form. Also, by step 4c, C is non-redundant. But then, there is a QA-pair $(P, \Pi_Y\sigma_C) \in \mathcal{S}$ for which there exists a $(P', \Pi_{Y'}\sigma_{C'})$ in $\mathcal{S}$ such that $(P', \Pi_{Y'}\sigma_{C'}) \neq (P, \Pi_Y\sigma_C)$ and $(P', \Pi_{Y'}\sigma_{C'}) \models (P, \Pi_Y\sigma_C)$.

By construction in Step 4b, P, P' are non-empty and therefore, there are $PF \in P$ and $PF' \in P'$ such that $PF = \Pi_Y(PF')$. But then, we must have had QA-pairs $(PF', \Pi_{Y'}\sigma_{C'_e})$ and $(PF, \Pi_Y\sigma_{C_e})$, where $C'_e \equiv C'$ and $C_e \equiv C$ after step 4a of the algorithm, such that $(PF', \Pi_{Y'}\sigma_{C'_e}) \models (PF, \Pi_Y\sigma_{C_e})$. This is contradiction, since at least one of them should have been eliminated in Step 4a. This completes the proof that the set $\mathcal{S}$ at the end of the algorithm is compact.

7.2 Proofs of Theorems and Proposition in Section 3.5

7.2.1 Proof of Proposition 3

1. ($\Rightarrow$) Assume that $\Pi_Y\sigma_C \rightsquigarrow_\exists \Pi_{Y'}\sigma_{C'}$. Let r be a relation, $P \subseteq \Pi_Y\sigma_C(r)$ be a set of projection facts, and $PF' \in \Pi_{Y'}\sigma_{C'}(r)$ be a projection fact , such that $\{(P, \Pi_Y\sigma_C)\} \models (PF', \Pi_{Y'}\sigma_{C'})$. Then, by Theorem 2, if $(P, \Pi_Y\sigma_C)$ is given as an input to Algorithm 3, with $\mathcal{D} = \emptyset$, then the algorithm must generate a QA-pair, $(PF'', \Pi_{Y''}\sigma_{C''})$, which dominates $(PF', \Pi_{Y'}\sigma_{C'})$. From here on, an argument, similar to the one presented for $\Rightarrow$ of the universal dominance can be repeated to show that $\Rightarrow$ holds.

($\Leftarrow$) Construct a t that satisfies $C \wedge C'$. Note, that such tuple exists because $C \wedge C'$ is consistent. Let r be a one tuple relation that contains t, and let PF be $\Pi_Y\sigma_C(r) = \Pi_Y(t)$, and PF' be $\Pi_{Y'}\sigma_{C'}(r) = \Pi_{Y'}(t)$. Consider an arbitrary relation r' and assume that $PF \subseteq \Pi_Y\sigma_C(r')$. Let t' be a tuple of r' such that $PF = \Pi_Y\sigma_C(t')$. By $Y' \subseteq Y$ and the construction of PF and PF', $PF' = \Pi_{Y'}(t')$. But then, it is sufficient to show that the tuple $t' \in r'$ that corresponds to PF also satisfies C'. PF satisfies C and equalities of C' between attributes which are in Y. Further, $C \models C'_{\neg Y}$ ensures that t' also satisfies equalities of C' which are not directly controlled by PF. Therefore, t satisfies C', $PF' = \Pi_{Y'}\sigma_{C'}(t)$, and $PF' \in \Pi_{Y'}\sigma_{C'}(r')$.

2. ($\Rightarrow$) Assume that $\Pi_Y\sigma_C \rightsquigarrow_\forall \Pi_{Y'}\sigma_{C'}$. Let PF' be any projection fact over $\Pi_{Y'}\sigma_{C'}$. Then, there must exist a relation r and $P \subseteq \Pi_Y\sigma_C(r)$ such that $(P, \Pi_Y\sigma_C) \models (PF', \Pi_{Y'}\sigma_{C'})$. Then, by Theorem 2, if $(P, \Pi_Y\sigma_C)$ is given as an input to Algorithm 3, with $\mathcal{D} = \emptyset$, then the algorithm must generate a QA-pair, $(PF'', \Pi_{Y''}\sigma_{C''})$, which dominates $(PF', \Pi_{Y'}\sigma_{C'})$.

Let r_d denote the relation that was generated from $(P, \Pi_Y\sigma_C)$ in Step 1.a of Algorithm 3. To complete the proof, we use the following two straightforward claims:

Claim 3: The only dependencies that can be applied in Algorithm 3 on r_d are the equality-generating dependencies, which were derived from C, because $\mathcal{D}$ is empty.

Claim 4: The application of the equality-generating dependencies which were derived from C can equate only two different null-values, never a null value with a constant, because $\Pi_Y \sigma_C$ is in normal form.

Let t denote the tuple that was used by Algorithm 3 in Step 3 to construct $(PF'', \Pi_{Y''}\sigma_{C''})$. That $Y'' = Y$ follows directly from Claim 4. Further, t must satisfy both C'' and C''_e, where C''_e is the extension of C'' with the equalities explicitly given in PF''. It follows from Claim 3 that t must have been originated from one of the tuples of r_d. By Claim 4, and because all of the tuples of r_d satisfy C, t must satisfy C. Since t satisfies C, C'', and C''_e, both $C \wedge C''$ and $C \wedge C''_e$ must be consistent.

By Theorem 2, $(PF'', \Pi_{Y''}\sigma_{C''})$ is in normal form. Therefore, C'' contains equalities only between those attributes which are not in Y'', that is equalities between null-values of t. We know that when Algorithm 3 generated r_d in Step 1.a, unique null-values were assigned to all of the attributes that were not in Y. But then, t can contain the same null-value for different attributes only if they were equated by one of the dependencies which were created from C. Therefore, C entails all the equalities of t among null-values, and therefore $C \models C''_{\neg Y}$. Also, $C'' \equiv C''_{\neg Y}$ because $Y = Y''$, and thus $C \models C''$.

We showed that $Y'' = Y$, $C''_e \wedge C$ is consistent, $C \models C''_{\neg Y}$, and we also know from $(PF'', \Pi_{Y''}\sigma_{C''}) \models (PF', \Pi_{Y'}\sigma_{C'})$ that $C''_e \models C'_e$. To complete the proof of $\Rightarrow$, we need to show that

1. $Y' \subseteq Y$,

2. $C' \wedge C$ is consistent,

3. $C \models C'_{\neg Y}$, and

4. $C' \models (C \wedge C')_{Y'}$

To show (1) we use the fact, that because $(PF'', \Pi_{Y''}\sigma_{C''})$ is in normal form and $C''_e \models C'_e$, $Y' \subseteq Y''$ must hold. From this and $Y = Y''$, we have $Y' \subseteq Y$. For (2) assume, by contradiction, that $C \wedge C'$ is inconsistent. Then, $C \wedge C' \models A = a$ and $C \wedge C' \models A = b$ for two different constants a and b. We know that $C''_e \models C'_e \models C'$. But then, $C''_e \wedge C \models A = a$ and $C''_e \wedge C \models A = b$, where $a \neq b$. But this is a contradiction to the fact that $C \wedge C''_e$ is consistent.

For (3) notice that by construction of C'' in Step 3 of Algorithm 3, $C''_e = C''_{Y''} \wedge C''_{\neg Y''}$, where

1. $C''_{Y''}$ is the conjunction of equalities $A_i = c$, where c is a constant and $A_i \in Y''$,

2. $C''_{\neg Y''}$ is the conjunction of equalities $A_i = A_j$, where $A_i \notin Y''$ and $A_j \notin Y''$.

Thus, the attribute sets of $C''_{Y''}$ and $C''_{\neg Y''}$, denoted as $Attr(C''_{Y''})$ and $Attr(C''_{\neg Y''})$ respectively, are disjoint; i.e., $Attr(C''_{Y''}) \cap Attr(C''_{\neg Y''}) = \emptyset$.

Since $C''_e \models C'_e \models C'$, if $C' \models A_i = A_j$ where A_i or A_j are not in Y'', then $C''_e \models A_i = A_j$. Since at least one of A_i or A_j is not in Y'' and because $Attr(C''_{Y''}) \cap Attr(C''_{\neg Y''}) = \emptyset$ we know that neither A_i nor A_j is in Y''. Therefore, only equalities in $C''_{\neg Y''}$ could be used to derive $C''_e \models A_i = A_j$; hence $C''_{\neg Y''} \models A_i = A_j$. Since $A_i = A_j$ is an arbitrary equality entailed by $C'_{\neg Y}$ and $Y = Y''$, we get $C''_{\neg Y} \models C'_{\neg Y}$. Then, by using $C \models C''_{\neg Y}$ and the transitivity of logical entailment, $C \models C'_{\neg Y}$.

Finally, we show that (4) holds by using the fact that the tuple t used to generate $(PF'', \Pi_{Y''}\sigma_{C''})$ satisfies $C''_e \wedge C$. Since $C''_e \models C'_e$, if $C'_e \wedge C$ is inconsistent, then does not exist t that satisfies $C''_e \wedge C$, and therefore $C \wedge C'_e$ must be consistent. Assume, by contradiction, that (4) does not hold, and that $C \wedge C' \models A_i = c_1$, where $A_i \in Y'$ and c_1 is a constant, but $C' \not\models A_i = c_1$. Since $C \wedge C'$ is consistent, C' cannot entail any equality between A_i and a constant different from c_1. Let PF' be a projection fact over $\Pi_{Y'}\sigma_{C'}$ such that $A_i = c_2$ in PF' and $c_1 \neq c_2$. Such PF' exists because $C' \not\models A_i = c_1$. Since $C'_e \models C'$, we have $C \wedge C'_e \models A_i = c \wedge A_i = c_1$, and therefore $C \wedge C'_e$ is inconsistent.

Next, assume that $C \wedge C' \models A_i = A_j$, where A_i and A_j are in Y' but $C' \not\models A_i = A_j$. Since $C \wedge C'$ is consistent, there does not exist two different constants c_1 and c_2, such that $C' \models A_i = c_1$ and $C' \models A_j = c_2$, further, since $C' \not\models A_i = A_j$, there there does not exist a constant c such that $C' \models A_i = c \wedge A_j = c$. Let PF' be a projection fact over $\Pi_{Y'}\sigma_{C'}$, and assign two different constants to A_i and A_j. Then, since $C'_e \models C'$, we have $C \wedge C'_e \models A_i = c_1 \wedge A_j = c_2 \wedge A_i = A_j$, where $c_1 \neq c_2$. Thus $C \wedge C'_e$ is inconsistent. This completes the proof of ($\Rightarrow$).

($\Leftarrow$) Let PF' be any projection fact over $\Pi_{Y'}\sigma_{C'}$. Construct a tuple t' from PF' such that t satisfies $C \wedge C'_e$. In order to be able to construct t we have to show that $C \wedge C'_e$ is consistent. For this, we use the following observations:

Observation 1. Let

1. C_Y be the conjunction of equalities
   (a) $A_i = c$, such that $C \models A_i = c$, where c is a constant and $A_i \in Y$, or
   (b) $A_i = A_j$, such that $C \models A_i = A_j$, and A_i or A_j is in Y.
2. $C_{\neg Y}$ be the conjunction of equalities $A_i = A_j$, such that $C \models A_i = A_j$ and A_i or A_j are not in Y.

Then, C_Y and $C_{\neg Y}$ form a partition of C; i.e.,

1. $C = C_Y \wedge C_{\neg Y}$, and
2. $Attr(C_Y) \cap Attr(C_{\neg Y}) = \emptyset$, where $Attr(C_Y)$ and $Attr(C_{\neg Y})$ denote the attributes of C_Y and $C_{\neg Y}$, respectively.

Observation 2. Let

1. $C'_{Y'}$ be the conjunction of equalities
   (a) $A_i = c$, such that $C'_e \models A_i = c$, where c is a constant and $A_i \in Y'$, or
   (b) $A_i = A_j$, such that $C'_e \models A_i = A_j$, and A_i or A_j is in Y'.
2. $C'_{\neg Y'}$ be the conjunction of equalities $A_i = A_j$, such that $C'_e \models A_i = A_j$ and A_i or A_j are not in Y'.

Then, $C'_{Y'}$ and $C'_{\neg Y'}$ form a partition of C'_e; i.e.,

1. $C'_e = C'_{Y'} \wedge C'_{\neg Y'}$, and
2. $Attr(C'_{Y'}) \cap Attr(C'_{\neg Y'}) = \emptyset$, where $Attr(C'_{Y'})$ and $Attr(C'_{\neg Y'})$ denote the attributes of $C'_{Y'}$ and $C'_{\neg Y'}$, respectively.

Observation 3. Let $C'_{\neg Y'} = C'_{Y-Y'} \wedge C'_{\neg Y}$, where

1. $C'_{Y-Y'}$ be the conjunction of equalities of $C'_{\neg Y'}$, where A_i or A_j are in Y.
2. $C'_{\neg Y}$ be the conjunction of equalities of C'_e, where A_i or A_j are not in Y.

Then, $C'_{Y-Y'}$ and $C'_{\neg Y}$ form a partition of $C'_{\neg Y'}$; i.e.,

1. $C'_{\neg Y'} = C'_{Y-Y'} \wedge C'_{\neg Y}$, and
2. $Attr(C'_{Y-Y'}) \cap Attr(C'_{\neg Y}) = \emptyset$, where $Attr(C'_{Y-Y'})$ and $Attr(C'_{\neg Y})$ denote the attributes of $C'_{Y-Y'}$ and $C'_{\neg Y}$, respectively.

Proof: Assume, by contradiction, that $C'_e \models A_i = A_j$ such that $A_i \in Y$ and $A_j \notin Y$. Then, $C \models A_i = A_j$ because $C \models C'_{\neg Y}$. However, since $\Pi_Y \sigma_C$ is in normal form, if $A_i \in Y$ and $C \models A_i = A_j$ then A_j must be in Y, too. But this contradicts our assumption that $A_j \notin Y$.

Let $C^*_{Y'}$ denote the selection conditions that is equivalent to $C'_{Y'}$; i.e., $C^*_{Y'} \equiv C'_{Y'}$, but all equalities of the form $A_i = A_j$ are removed. (Note, that these equalities of $C'_{\neg Y'}$ are redundant, since they are implicitly enforced by the assignment of constants to the attributes in Y'.) Let C^*_e be $C^*_e = C^*_{Y'} \wedge C'_{Y'}$, and therefore $C^*_e \equiv C'_e$. Construct a graph G such that its nodes are the all the attribute names and constants of C and C^*_e. There is an edge between two nodes A and B if and only if $A = B$ is in $C \wedge C^*_e$. Assume, by contradiction, that $C \wedge C^*_e$ is inconsistent. Then, G contains a path between two different constants. Let P be such a path of minimal length, say from a to b, such that $a \neq b$. The first edge e_1 of P must be of the form (a, A_1); i.e., resulting from the equation $A_1 = a$, where a is a constant and A_1 is an attribute name. From Observations 1 and 2, e_1 is either in C_Y or in $C^*_{Y'}$.

Case 1: If e_1 is in $C^*_{Y'}$, then $C^*_{Y'} \models A_1 = a$ and $A_1 \in Y'$. If the next edge is of the form $e_2 = (A_1, c)$, where c is a constant, then it is either the reverse of e_1, and therefore $a = c$, or it must be in C_Y. In the later case, we have $C \models A_1 = c$, and by requirement 1.d we must have $C' \models A_1 = c$. But then, in either case $a = c$, because each of C and C'_e is consistent. Then, the node a appears twice in P, which contradicts the minimality of P.

If the second edge e_2 is the form of (A_1, A_2), where A_2 is an attribute name, then e_2 must be in C_Y. Let $(A_1, A_2), \ldots, (A_{k-1}, A_k), (A_k, c)$ denote the sub-path of P from the node A_1 to the first constant node c. Thus, edges $(A_1, A_2), \ldots, (A_{k-1}, A_k)$ must come from $C'_{Y-Y'}$ or C_Y by Observations 1, 2, and 3. But since $C' \models C'_{Y-Y'}$

and $C \models C_Y$ we have $C \wedge C' \models A_1 = A_2 \wedge \ldots \wedge A_1 = A_k$. Further, if (A_k, c) is in C_Y, then $C \models A_k = c$ and, therefore, $C \wedge C' \models A_1 = c$. Then, by requirement 1.d, $C^*_{Y'} \models A_1 = c$. Since $C^*_{Y'}$ is consistent, we get $a = c$, and the the node a appears twice in P, which contradicts the minimality of P. Finally, if (A_k, c) is in $C^*_{Y'}$ then $A_k = c$, and, because $C \wedge C' \models A_1 = A_k$, $C' \models A_1 = a \wedge A_1 = A_k \wedge A_k = c$. Since C' is consistent, we get $a = c$, and the the node a appears twice in P, which contradicts the minimality of P.

Case 2: If e_1 is in C_Y, then if $A_1 \in Y'$ then we can construct an argument similar to Case 1 to derive contradiction to the minimality of P. Otherwise, if A_1 must be in $Y - Y'$ and in equalities of $C'_{Y-Y'}$ and C_Y. If the second edge from A_1 is the form $e_2 = (A_1, c)$ then it must be in C_Y, by Observation 2, and therefore $C \models A_1 = a \wedge A_1 = c$. Since C is consistent, we get $a = b$, and the the node a appears twice in P, which contradicts the minimality of P.

Otherwise, let $(A_1, A_2), \ldots, (A_{k-1}, A_k), (A_k, c)$ denote the sub-path of P from the node A_1 to the first constant node c. Again, we know that $C \wedge C'$ entails the edges $(A_1, A_2), \ldots, (A_{k-1}, A_k)$. If the last edge (A_k, c) is in C_Y, then we have $C \models A_k = c$, and therefore $C \wedge C' \models A_1 = a \wedge A_1 = A_k \wedge A_k = c$. Since, by 1.b, $C \wedge C'$ is consistent, we get $a = c$ and the the node a appears twice in P, which contradicts the minimality of P.

Finally, if the last edge is in $C^*_{Y'}$; i.e., $C^*_{Y'} \models A_k = c$ then $A_k \in Y'$. Since $C \wedge C' \models A_1 = a \wedge A_1 = A_k$ and therefore $C \wedge C' \models A_k = a$, we know that, by requirement 1.d, $C^*_{Y'} \models A_k = a$. But then, $a = c$ and the the node a appears twice in P, which contradicts the minimality of P.

We showed that does not exists a path in G between two different constants, and therefore $C \wedge C'_e$ is consistent. Now, let $PF = \Pi_Y(t')$. By construction of PF

and because $Y' \subseteq Y$, we have $PF' = \Pi_{Y'}(PF)$.

Let r be any relation that contains PF; i.e., $PF \subseteq \Pi_Y \sigma_C(r)$. Then, there exists a tuple t in r that satisfies C and $PF = \Pi_Y(t)$. Since $PF' = \Pi_{Y'}(PF)$, $PF' = \Pi_{Y'}(t)$. The only remaining part is to show that t also satisfies C'.

1. Because $Y' \subseteq Y$ and the way of constructing PF, we know that equalities $C' \models A_i = c$, where c is a constant, must be satisfied.

2. Further, since $C \wedge C'$ is consistent and the way of constructing PF, we know that equalities $C' \models A_i = A_j$ where A_i and A_j are in Y must be satisfied.

3. Finally, we need to show that equalities in $C'_{\neg Y}$ are satisfied by t. Since t satisfies C and $C \models C'_{\neg Y}$, t must also satisfy $C'_{\neg Y}$.

This completes the proof that t satisfies C', and therefore $PF' = \Pi_{Y'}\sigma_{C'}(t)$. We showed that for any PF' over $\Pi_{Y'}\sigma_{C'}$ there exists a PF such that $(PF, \Pi_Y\sigma_C) \models (PF', \Pi_{Y'}\sigma_{C'})$. But this is exactly the requirement for universal dominance and therefore $\Pi_Y\sigma_C \rightsquigarrow_\forall \Pi_{Y'}\sigma_{C'}$. This completes the proof of Proposition 3.

7.2.2 Proof of Proposition 4

Let $\mathcal{P} = \{\mathit{\Pi}_{X_1}\sigma_{C_1}, \ldots, \mathit{\Pi}_{X_n}\sigma_{C_n}\}$. For the *first part* of the claim, if C is consistent, then there must exists a PF over $\Pi_Y\sigma_C$. But then, by definition of *universal disclosure* we know that for PF there exist r that satisfies $\mathcal{D}$, $P_i \subseteq \Pi_{X_i}\sigma_{C_i}(r)$ $(i = 1, \ldots, n)$ such that

$$\{(P_1, \Pi_{X_1}\sigma_{C_1}), \ldots, (P_n, \Pi_{X_n}\sigma_{C_n})\} \models (PF, \Pi_Y\sigma_C)$$

But this is exactly the requirement for existential disclosure, thus part 1 of the claim is true.

To prove the *second part* of the claim, we use that there must exist

1. r that satisfies $\mathcal{D}$,

2. $P_1 \subseteq \Pi_{X_1}\sigma_{C_1}(r), \ldots, P_n \subseteq \Pi_{X_n}\sigma_{C_n}(r)$, and

3. $PF \in \Pi_Y\sigma_C(r)$

such that

$$\{(P_1, \Pi_{X_1}\sigma_{C_1}), \ldots, (P_n, \Pi_{X_n}\sigma_{C_n})\} \models_{\mathcal{D}} (PF, \mathit{\Pi}_Y\sigma_C)$$

since $\mathcal{P} \rightsquigarrow_{\exists\mathcal{D}} \mathit{\Pi}_Y\sigma_C$. That is, for every relation r' that satisfies $\mathcal{D}$, $P_1 \subseteq \Pi_{X_1}\sigma_{C_1}(r'), \ldots, P_n \subseteq \Pi_{X_n}\sigma_{C_n}(r')$ implies $PF \in \Pi_Y\sigma_C(r')$.

Let PF be the projection fact as defined above. We know that PF satisfies C. Construct C' by extending C with all of the equalities of PF. By construction, PF satisfies C'. Consider $\Pi_Y\sigma_{C'}$. We need to show that (1) $\mathcal{P} \rightsquigarrow_{\forall} \mathit{\Pi}_Y\sigma_{C'}$ and (2) $\Pi_Y\sigma_{C'} \rightsquigarrow_{\exists} \Pi_Y\sigma_C$. To show (1) we use that since $\mathcal{P} \rightsquigarrow_{\exists\mathcal{D}} \mathit{\Pi}_Y\sigma_C$ we know that for every relation r' that satisfies $\mathcal{D}$, $P_1 \subseteq \Pi_{X_1}\sigma_{C_1}(r'), \ldots, P_n \subseteq \Pi_{X_n}\sigma_{C_n}(r')$ implies $PF \in \Pi_Y\sigma_C(r')$. But then, it also implies $PF \in \Pi_Y\sigma_{C'}(r')$, therefore

$$\{(P_1, \Pi_{X_1}\sigma_{C_1}), \ldots, (P_1, \Pi_{X_1}\sigma_{C_1})\} \models (PF, \Pi_Y\sigma_{C'})$$

Finally, since PF is the only projection fact over $\Pi_Y\sigma_{C'}$, the above implication is true for all of the projection facts over $\Pi_Y\sigma_{C'}$, therefore $\mathcal{P} \rightsquigarrow_{\forall\mathcal{D}} \mathit{\Pi}_Y\sigma_C$

(2) follows from the fact that $Y \subseteq Y$ and $C' \wedge C$ is consistent. Therefore the claim is true.

7.2.3 Proof of Proposition 5

We prove that universal disclosure is transitive by showing that for every query $\Pi_{Y''}\sigma_{C''}$ in $\mathcal{P}_3$ and for every PF'' over $\Pi_{Y''}\sigma_{C''}$ there exist r that satisfies $\mathcal{D}$, $P \subseteq$

$\Pi_Y \sigma_C(r)$ such that

$$\{(P, \Pi_Y \sigma_C)\} \models (PF'', \Pi_{Y''} \sigma_{C''})$$

Let PF'' be a projection fact over $\Pi_{Y''} \sigma_{C''}$. By $\Pi_{Y'} \sigma_{C'} \leadsto_\forall \Pi_{Y''} \sigma_{C''}$, we know that for PF'' there must exist r, $P' \subseteq \Pi_{Y'} \sigma_{C'}(r)$ such that

$$\{(P', \Pi_{Y'} \sigma_{C'})\} \models (PF'', \Pi_{Y''} \sigma_{C''})$$

i.e., for every r' $P' \subseteq \Pi_{Y'} \sigma_{C'}(r')$ implies $PF'' \subseteq \Pi_{Y''} \sigma_{C''}(r')$.

Let projection facts $PF'_1, \ldots, PF'_k \in P'$. By $\Pi_Y \sigma_C \leadsto_\forall \Pi_{Y'} \sigma_{C'}$, we know that for every PF'_i, $i = 1, \ldots, k$ there exist r_i, $P_i \subseteq \Pi_Y \sigma_C(r_i)$ such that

$$\{(P_i, \Pi_Y \sigma_C)\} \models (PF'_i, \Pi_{Y'} \sigma_{C'})$$

i.e., for every relation r' $P_i \subseteq \Pi_Y \sigma_C(r')$ implies $PF'_i \subseteq \Pi_{Y'} \sigma_{C'}(r')$.

Let r_{all} be a relation containing $P_1, \ldots, P_k$ and let P be the union of P_is ($i = 1, \ldots, k$). By construction, $P \subseteq \Pi_Y \sigma_C(r_{all})$, by $\Pi_Y \sigma_C \leadsto_\forall \Pi_{Y'} \sigma_{C'}$, $P_2 \subseteq \Pi_{Y'} \sigma_{C'}(r_{all})$. But then, by $\Pi_{Y'} \sigma_{C'} \leadsto_\forall \Pi_{Y''} \sigma_{C''}$ $PF'' \in \Pi_{Y''} \sigma_{C''}(r_{all})$, that is exactly the requirement of universal disclosure. Therefore, $\Pi_Y \sigma_C \leadsto_\forall \Pi_{Y''} \sigma_{C''}$ must hold.

7.2.4 Proof of Proposition 6

We prove it by giving a counter example: By definition of data-independent disclosure, we know that

$$\Pi_{ABC} \sigma_{A=2 \wedge B=3} \leadsto_\exists \Pi_{ABC} \sigma_{A=2}$$

Also,

$$\Pi_{ABC} \sigma_{A=2} \leadsto_\exists \Pi_{ABC} \sigma_{A=2 \wedge B=1}$$

But,

$$\Pi_{ABC} \sigma_{A=2 \wedge B=3} \not\leadsto_\exists \Pi_{ABC} \sigma_{A=2 \wedge B=1}$$

This completes the proof of Proposition 6.

7.2.5 Proof of Theorem 4

Clearly, since the application of dependencies do not generate new symbols, there is only a finite number of distinct tuples that can be generated; therefore the algorithm must *terminate*. We need to show that the set $\mathcal{S}$ computed by the algorithm is (1) sound, (2) complete and (3) compact.

(1) Proof of soundness: It is sufficient to show that $\mathcal{S}$ constructed in Step 4 (i.e., before Step 5) of Algorithm 4 is *sound*. This follows from the following two claims:

Claim 5: Every $\Pi_Y \sigma_C \in \mathcal{S}$ is existentially disclosed from $\mathcal{P}$ under $\mathcal{D}$, and

Claim 6: If $\mathcal{P} \rightsquigarrow_{\exists \mathcal{D}} \mathit{\Pi}_Y \sigma_C$ and $\Pi_Y \sigma_C \rightsquigarrow_{\exists} \Pi_{Y'} \sigma_{C'}$ then $\mathcal{P} \rightsquigarrow_{\exists \mathcal{D}} \mathit{\Pi}_{Y'} \sigma_{C'}$

To prove Claim 5 we construct an example relation r_{ex} that satisfies $\mathcal{D}$, sets $P_i \subseteq \Pi_{X_i} \sigma_{C_i}(r_{ex})$, $(i = 1, \ldots, m)$, $PF \in \Pi_Y \sigma_C(r_{ex})$, and show that $\{(P_1, \Pi_{X_1} \sigma_{C_1}), \ldots, (P_m, \Pi_{X_m} \sigma_{C_m})\} \models_{\mathcal{D}} (PF, \Pi_Y \sigma_C)$.

Let r_{ex} be a relation over R that contains only one tuple t_{ex} such that for each attribute $A_i \in R$, $t_{ex}[A_i] = c$, where c is a constant. Clearly, r_{ex} satisfies $\mathcal{D}$. Let $P_i = \Pi_{X_i} \sigma_{C_i}(r_{ex})$ for $i = 1, \ldots, m$. Note that since C_i contains only equalities among attributes, t_{ex} satisfies all of the queries and the answer to the query $\Pi_{X_i} \sigma_{C_i}$ is the projection of t_{ex} on attributes X_i. Also, $PF = \Pi_Y \sigma_C(r_{ex})$.

We use Algorithm 3 (Data-dependent disclosure cover algorithm) to show $\{(P_1, \Pi_{X_1} \sigma_{C_1}), \ldots, (P_m, \Pi_{X_m} \sigma_{C_m})\} \models_{\mathcal{D}} (PF, \mathit{\Pi}_Y \sigma_C)$. Let $\bar{r}$ be the relation generated from $P_1, \ldots, P_m$ at the beginning of Step 2 on Algorithm 3. Note, that at this stage each P_i, $i = 1, \ldots, m$ has already been *chased* by the dependencies which were generated by the selection conditions of the queries. This application of dependencies is equivalent to the initialization steps 1.a and 1.b of Algorithm 4.

We want to show that if a sequence of dependency applications by Algorithm 4 created a tuple t that was used to generate $\Pi_Y\sigma_C$ in $\mathcal{S}$ then the same sequence of dependency applications will generate a tuple $\bar{t} \in \bar{r}$ by Algorithm 3 such that PF over $\Pi_Y\sigma_C$ can be generated from $\bar{t}$. From this and Theorem 2 it follows that every $\Pi_Y\sigma_C \in \mathcal{S}$ is existentially disclosed from $\mathcal{P}$ under $\mathcal{D}$.

To show that the application of dependencies on $\bar{r}$ will indeed generate the tuple $\bar{t}$ notice that $\bar{r}$ and r, the input relation in Step 2 of Algorithm 4, are *isomorfic* with the only difference that in places where r has a variable, $\bar{r}$ has the constant c. But then, the *chase* process executes exactly the same way on both r and $\bar{r}$.

To prove Claim 6 note, that because $\mathcal{P} \rightsquigarrow_{\exists\mathcal{D}} \mathit{\Pi}_Y\sigma_C$ there exists r over R that satisfies $\mathcal{D}$, $P_i \subseteq \Pi_{Y_i}\sigma_{C_i}(r)$ for all $i = 1, \ldots, n$, and $PF \in \Pi_Y\sigma_C(r)$ such that $\{(P_1, \Pi_{Y_1}\sigma_{C_1}), \ldots, (P_n, \Pi_{Y_n}\sigma_{C_n})\} \models_{\mathcal{D}} (PF, \mathit{\Pi}_Y\sigma_C)$. Also, from $\Pi_Y\sigma_C \rightsquigarrow_{\exists} \Pi_{Y'}\sigma_{C'}$ and Proposition 3, we know that $Y' \subseteq Y$, $C' \wedge C$ is consistent, and $C \models C'_{\neg Y}$. To show that $\mathcal{P} \rightsquigarrow_{\exists\mathcal{D}} \mathit{\Pi}_{Y'}\sigma_{C'}$, we construct an example projection fact PF' over $\Pi_{Y'}\sigma_{C'}$ such that, we can prove, that there exists a relation r' that satisfies $\mathcal{D}$, sets of projection facts P_i, $i = 1, \ldots, n$, such that $P_i \subseteq \Pi_{Y_i}\sigma_{C_i}(r')$, $PF' \in \Pi_{Y'}\sigma_{C'}(r')$ such that $\{(P_1, \Pi_{Y_1}\sigma_{C_1}), \ldots, (P_n, \Pi_{Y_n}\sigma_{C_n})\} \models_{\mathcal{D}} (PF', \mathit{\Pi}_{Y'}\sigma_{C'})$.

For this, construct a tuple t that satisfies $C \wedge C'$ out of PF. This tuple exists, since $C \wedge C'$ is consistent. Then, let PF' be $\Pi_{Y'}(t)$. By construction, and the fact that $Y' \subseteq Y$, we have $PF' = \Pi_{Y'}(PF)$. From $\{(P_1, \Pi_{Y_1}\sigma_{C_1}), \ldots, (P_n, \Pi_{Y_n}\sigma_{C_n})\} \models_{\mathcal{D}} (PF, \mathit{\Pi}_Y\sigma_C)$ $P_i \subseteq \Pi_{Y_i}\sigma_{C_i}(r)$ for all $i = 1, \ldots, n$ implies $PF \in \Pi_Y\sigma_C(r)$. Then, there must exists a tuple t' in r' such that $PF = \Pi_Y(t')$, and because $PF' = \Pi_{Y'}(PF)$, we have $PF' = \Pi_{Y'}(t')$. Also, because t' satisfies C and $C \models C'_{\neg Y}$ t' also satisfies

C'. But then, we have $PF' = \Pi_{Y'}\sigma_{C'}(t')$, and, therefore, $PF' \in \Pi_{Y'}\sigma_{C'}(r')$. But this is exactly the requirement for existential disclosure, thus $\mathcal{P} \leadsto_{\exists\mathcal{D}} \mathit{\Pi}_{Y'}\sigma_{C'}$. This completes the proof of Claim 6. This completes the proof of soundness.

(2) Proof of completeness: First we show that for $\mathcal{S}$ generated in Step 4 (i.e., before Step 5) in Algorithm 4 and for every query $\Pi_{Y'}\sigma_{C'}$, $\mathcal{P} \leadsto_{\exists\mathcal{D}} \mathit{\Pi}_{Y'}\sigma_{C'}$ implies that there exists $\Pi_Y\sigma_C \in \mathcal{S}$ such that $\Pi_Y\sigma_C \leadsto_{\exists} \Pi_{Y'}\sigma_{C'}$; that is Step 4 of Algorithm 4 generates a tuple over Y and equality among null values such that $C \models A_i = A_j \in C'$ where $A_i or A_j \notin Y$.

Since $\mathcal{P} \leadsto_{\exists\mathcal{D}} \mathit{\Pi}_{Y'}\sigma_{C'}$, there must exist (1) a relation r that satisfies $\mathcal{D}$, (2) $P_i \subseteq \Pi_{X_i}\sigma_{C_i}(r),\ i = 1, \ldots, m$, and (3) $PF \in \Pi_{Y'}\sigma_{C'}(r)$ such that

$$\{(P_1, \Pi_{X_1}\sigma_{C_1}), \ldots, (P_1, \Pi_{X_m}\sigma_{C_m})\} \models_{\mathcal{D}} (PF, \mathit{\Pi}_{Y'}\sigma_{C'})$$

But then, by Theorem 2, Algorithm 3 will generate a projection fact that contains PF if $P_1, \ldots, P_m$ are given as input.

Let $\bar{r}$ be the relation generated from $P_1, \ldots, P_m$ at the beginning of Step 2 of Algorithm 3; i.e., each P_i is already *chased* by dependencies generated from the C_is. Let $D_1, \ldots, D_l$ the sequence of dependency applications that generated the tuple $\bar{t}$ which was used to extract $(PF, \Pi_{Y'}\sigma_{C'})$. We want to show that

1. $D_1, \ldots, D_l$ can be applied on r, the relation generated in Step 2 of Algorithm 4, and
2. the application of $D_1, \ldots, D_l$ by Algorithm 4 will generate a tuple that corresponds to query $\Pi_Y\sigma_C$, where $\Pi_Y\sigma_C \leadsto_{\exists} \Pi_{Y'}\sigma_{C'}$.

For this, we use the notion of generally corresponding tuples. We say, that the tuple $t \in r$ generally corresponds to tuples $\bar{t_1}, \ldots, \bar{t_k}$ if there exist mappings

$\nu_1, \ldots, \nu_k$, which preserve equalities, such that $t = \nu_1(\bar{t_1}), \ldots, t = \nu_k(\bar{t_k})$, where ν_i maps all constants into x and is the identity mapping for the null values.

Clearly, at the beginning there exist such mappings from all of the tuples of $\bar{r}$ to the tuples of r. Assume that D_i is applied on tuples $\bar{t_{i_1}}, \ldots, \bar{t_{i_j}} \in \bar{r}$ that created the tuple $\bar{t}$. Since ν_is preserve equalities, D_i can be applied on the generally corresponding tuples of $\bar{t_{i_1}}, \ldots, \bar{t_{i_j}}$, and the generated tuple t is defined as follows:

1. If D_i is an equality generating dependency which replaces all occurrences of a null value in $\bar{r}$ and because ν_is are the identity mappings for the null values, the application of D_i on r will replace all occurrences of the same null value in the generally corresponding tuples.

2. If D_i is a tuple generating dependency that generated the new tuple $\bar{t} = (h(a_1), \ldots, h(a_n))$ in $\bar{r}$, when applied on the generally corresponding tuples will generate a tuple
$(\nu(h(a_1)), \ldots, \nu(h(a_n))) = \nu(h(a_1), \ldots, h(a_n))$ in r.

But then, after the application of dependencies on both $\bar{r}$ and r, there is a generally corresponding tuple in r for every tuple $\bar{t} \in \bar{r}$. To complete the proof of Completeness, we need to show that eliminating queries from $\mathcal{S}$ in Step 5 of Algorithm 4 still leaves the set $\mathcal{S}$ complete. Specifically, we need to show that if $\Pi_{Y'}\sigma_{C'}$ is eliminated from $\mathcal{S}$ in Step 5 because $\Pi_Y\sigma_C \rightsquigarrow_\forall \Pi_{Y'}\sigma_{C'}$ for a query $\Pi_Y\sigma_C$ in $\mathcal{S}$, then for every $\Pi_{Y''}\sigma_{C''}$ such that $\Pi_{Y'}\sigma_{C'} \rightsquigarrow_\exists \Pi_{Y''}\sigma_{C''} \Rightarrow \Pi_Y\sigma_C \rightsquigarrow_\exists \Pi_{Y''}\sigma_{C''}$. First, since C must be consistent, by Proposition 4, universal dominance implies existential dominance, and thus $\Pi_Y\sigma_C \rightsquigarrow_\exists \Pi_{Y'}\sigma_{C'}$. Then, since $\Pi_{Y'}\sigma_{C'} \rightsquigarrow_\exists \Pi_{Y''}\sigma_{C''}$, by transitivity of existential dominance (Claim 6), $\Pi_Y\sigma_C \rightsquigarrow_\exists \Pi_{Y''}\sigma_{C''}$, which completes the proof of completeness.

(3) Proof of compactness: The compactness property follows directly from the way $\mathcal{S}$ is constructed in Step 5 of Algorithm 4.

7.2.6 Proof of Theorem 5

Algorithm 5 must *terminate* because the application of dependencies do not generate new symbols and, therefore, only a finite number of distinct tuples and relations of E' can be created. We need to show that the set $\mathcal{S}$ computed by the algorithm is (1) complete and (3) compact.

(1) Proof of completeness: Let $\mathcal{S}$ be the output of Algorithm 5. To show completeness of $\mathcal{S}$, we have to prove that for every query $\Pi_{Y'}\sigma_{C'}$, $\mathcal{P} \rightsquigarrow_{\exists\mathcal{D}} \mathit{\Pi}_{Y'}\sigma_{C'}$ implies that there exists $\Pi_Y\sigma_C \in \mathcal{S}$ such that $\Pi_Y\sigma_C \rightsquigarrow_{\exists} \Pi_{Y'}\sigma_{C'}$. Assume that $\mathcal{P} \rightsquigarrow_{\exists\mathcal{D}} \mathit{\Pi}_{Y'}\sigma_{C'}$, that is, by Definition 8, there exist

1. a relation r that satisfies $\mathcal{D}$,
2. sets of projection facts $P_i \subseteq \Pi_{X_i}\sigma_{C_i}(r)$, $i = 1, \ldots, m$, and
3. $PF' \in \Pi_{Y'}\sigma_{C'}(r)$

such that $\{(P_1, \Pi_{X_1}\sigma_{C_1}), \ldots, (P_m, \Pi_{X_m}\sigma_{C_m})\} \models_{\mathcal{D}} (PF', \mathit{\Pi}_{Y'}\sigma_{C'})$. But then, by Theorem 2, if $\{(P_1, \Pi_{X_1}\sigma_{C_1}), \ldots, (P_m, \Pi_{X_m}\sigma_{C_m})\}$ and $\mathcal{D}$ are given as the input for Algorithm 3, it must generate a QA-pair $(PF'', \Pi_{Y''}\sigma_{C''})$ that dominates $(PF', \Pi_{Y'}\sigma_{C'})$.

The proof of completeness is based on the following claim:

Claim 7: The output $\mathcal{S}$ of Algorithm 5 contains a query $\Pi_Y\sigma_C$ such that $\Pi_Y\sigma_C \rightsquigarrow_{\exists} \Pi_{Y''}\sigma_{C''_e}$, where C''_e is the extension of C'' with all the equalities explicitly given in PF''.

We first complete the proof of completeness and then prove Claim 7. By construction of $\Pi_Y\sigma_C$ in Step 5 of Algorithm 5, $\Pi_Y\sigma_C$ is in normal form. Let $\Pi_{Y^*}\sigma_{C'}$

the the normal form of the query $\Pi_{Y'}\sigma_{C'}$. To prove $\Pi_Y\sigma_C \rightsquigarrow_\exists \Pi_{Y'}\sigma_{C'}$, it is sufficient to show that, $\Pi_Y\sigma_C \rightsquigarrow_\exists \Pi_{Y^*}\sigma_{C'}$, that is by Proposition 3, (1) $Y^* \subseteq Y$, (2) $C \wedge C'$ is consistent, and (3) $C \models C'_{\neg Y}$. In the proof we use the following facts:

Fact 1. By Proposition 1, $(PF'', \Pi_{Y''}\sigma_{C''}) \models (PF', \Pi_{Y'}\sigma_{C'})$ implies $C''_e \models C'_e$.

Fact 2. By Proposition 3, $\Pi_Y\sigma_C \rightsquigarrow_\exists \Pi_{Y''}\sigma_{C''_e}$ implies

1. $Y'' \subseteq Y$,
2. $C''_e \wedge C$ is consistent,
3. $C \models C''_{\neg Y}$

By construction of $(PF'', \Pi_{Y''}\sigma_{C''})$ in Algorithm 3, Y'' contains all the attributes that have constants assigned to, and C'' contains only equalities among attributes that are not in Y''. By Fact 1, $C''_e \models C'_e$, and therefore, C''_e must entail all of the equalities of the form $A_i = c$ which are explicitly given in PF'. Therefore, $Y^* \subseteq Y''$ must hold. This, and $Y'' \subseteq Y$ (Fact 2.1) gives $Y^* \subseteq Y$ follows.

From $C''_e \models C'_e$ (Fact 1) and $C''_e \wedge C$ is consistent (Fact 2.2), it follows that $C \wedge C'$ is consistent. The only remaining part is to show that $C \models C'_{\neg Y}$. Since $(PF'', \Pi_{Y''}\sigma_{C''})$ is in normal form (Theorem 2), C''_e contains two disjoint groups of equalities: (a) equalities among attributes that are not in Y'' (equalities of C'') and (b) equalities among attributes of Y'' (equalities of PF''). Therefore, if $C''_e \models A_i = A_j$ where A_i and A_j are not in Y'' then only equalities of group (a) can be used to derive the entailment. We know that $C''_e \models C'_e$, more specifically, $C''_e \models C'_{\neg Y}$. Since $Y = Y''$ and by the above argument, we have $C''_{\neg Y} \models C'_{\neg Y}$. Then, by using $C \models C''_{\neg Y}$ (Fact 2.3) and the transitivity of logical entailment, we have $C \models C'_{\neg Y}$. This completes the proof of completeness.

To prove Claim 7, we use the notion of *generally corresponding tuples.* Initially we define general corresponding tuples as follows. Let r_1 denote the relation that was generated from $\{(P_1, \Pi_{X_1}\sigma_{C_1}), \ldots, (P_m, \Pi_{X_m}\sigma_{C_m})\}$ by Algorithm 3 in Step 2 as r', and r_2 denote the relation generated from $\{\Pi_{X_1}\sigma_{C_1}, \ldots, \Pi_{X_m}\sigma_{C_m}\}$ by Algorithm 5 in Step 2 as r. Also, let $t_{i_1}, \ldots, t_{i_l}$ in r_1 denote the tuples that were generated in Step 1 of Algorithm 3 from the QA-pair $(P_i, \Pi_{X_i}\sigma_{C_i})$ $(i = 1, \ldots, m)$, and $\bar{t}_i$ in r_2 denote the tuple that was generated from the query $\Pi_{X_i}\sigma_{C_i}$ $(i = 1, \ldots, m)$ in Step 1 of Algorithm 5. Initially, we say, that $\bar{t}_i$ *generally corresponds* to the tuples $t_{i_1}, \ldots, t_{i_l}$. Let ν be the mapping that maps each tuple of r_1 to its generally corresponding tuple in r_2.

Consider the applications $a_1 = (d_1, h_1), \ldots, a_k = (d_k, h_k)$ of dependencies in Algorithm 3 on r_1, where d_i $(i = 1, \ldots, k)$ is the dependency being applied and h_i $(i = 1, \ldots, k)$ is the mapping being used. Let r_1^i, $(i = 1, \ldots, k)$ denote the relation that was produced from $r_1 = r_1^0$ after applications of $a_1, \ldots, a_i$. In the following claim, we recursively define the corresponding applications $\bar{a}_1 = (d_1, \bar{h}_1), \ldots, \bar{a}_k = (d_k, \bar{h}_k)$ on $r_2 = r_2^0$ (to be used in Algorithm 5) that will produce $r_2^1, \ldots, r_2^k$, and relations $E = E_0, E_1, \ldots, E_m$. Further, we define the mappings $\nu = \nu_0, \nu_1, \ldots, \nu_k$, each ν_i $(i = 0, \ldots, k)$ from r_1^i to r_2^i.

Claim 8: $\nu = \nu_0$ from $r_1 = r_1^0$ to $r_2 = r_2^0$ preserves constants and equalities; i.e., for tuples t_l and t_k in r_1,

1. if $t_l[A_i] = c$, where c is a constant, then

   (a) $\nu(t_l)[A_i] \neq c_1$, where c_1 is a constant different from c (i.e., $c_1 \neq c$), and

   (b) $\nu(t_l)[A_i] \sim x$

2. if $t_l[A_i] = t_k[A_j]$ then $\nu(t_l)[A_i] \sim \nu(t_k)[A_j]$,

where $\sim$ denotes the relation E defined in Step 3 of Algorithm 5.

Further, for all i, $i = 1, \ldots, k$

1. *Definition:* Given the application $a_i = (d_i, h_i)$, where d_i is of the form $\{B_1, \ldots, B_n\} \rightarrow H$, $\bar{h}_i$ is defined as $\bar{h}_i(B_j) \stackrel{def}{=} \nu_{i-1}(h_i(B_j))$, for all $j = 1, \ldots, n$.

2. *Claim:* $\bar{h}_i$ is indeed a generalized atom mapping with respect to d_i, r_2^{i-1}, and E_{i-1}; i.e., $\bar{h}_i$ preserves constants and equalities.

3. *Definition:* ν_i is defined as follows:

   (a) If d_i is an equality-generating dependency and let $t' \in r_1^i$ be the tuple that was originated from $t \in r_1^{i-1}$, then if $\nu_{i-1}(t) = \bar{t}$, then $\nu_i(t') = \bar{t}$.

   (b) If d_i is a tuple-generating dependency such that a_i generated the tuple $t' \in r_1^i$ and $\bar{a}_i$ generated $\bar{t}' \in r_2^i$ then $\nu_i(t) = \nu_{i-1}(t)$ for all tuples t in r_1^{i-1} and $\nu_i(t') = \bar{t}'$.

4. *Claim:* ν_i from r_1^i to r_2^i preserves constants and equalities.

Before proving Claim 8, we complete the proof of Claim 7. Let $a_1, \ldots, a_m$ be the dependency applications on r_1 by Algorithm 3 that generated the tuple t that was used in Step 3 to construct the QA-pair $(PF'', \Pi_{Y''}\sigma_{C''})$. Let $\bar{a}_1, \ldots, \bar{a}_m$ be the corresponding dependency applications, where each $\bar{h}_i$ $(i = 1, \ldots, m)$ is constructed as defined in Claim 8.2. We order the dependency applications performed by Algorithm 5 on r_2, such that $\bar{a}_1, \ldots, \bar{a}_m, \ldots, \bar{a}_n$; i.e., $\bar{a}_1, \ldots, \bar{a}_m$ are completed before any other dependency is applied by Algorithm 5. We can reorder the dependency applications of Algorithm 5, by the following Claim 9, that will be proved after we complete the proof of Claim 7.

Claim 9: The output of Algorithm 5 is independent of the order of the dependency applications.

Next, we apply $\bar{a}_1, \ldots, \bar{a}_m$ on r_2 in Step 4 of Algorithm 5. By items 2 and 4 of Claim 8. we know that for every tuple t in r_1^m there is a tuple $\bar{t}$ in r_2^m, such that $\nu_m(t) = \bar{t}$, and ν_m preserves equalities and constants. Note, that it is possible that Algorithm 5 will apply additional dependencies after $\bar{a}_1, \ldots, \bar{a}_m$ are completed. However, since a dependency application can only add tuples to r_2^m or extend E_m the final relation r' at the end of Step 4 of Algorithm 5 must contain $\bar{t}$, and E' must contain E_m.

Let $\Pi_Y \sigma_C$ be the query that was generated in Step 5 of Algorithm 5 from $\bar{t}$ and E'. To complete the proof of Claim 7, we need to show that $\Pi_Y \sigma_C$ existentially discloses $\Pi_{Y''} \sigma_{C_e''}$; i.e., by Proposition 3(2), that $Y'' \subseteq Y$, $C \wedge C_e''$ is consistent, and $C \models C''_{\neg Y}$. $Y'' \subseteq Y$ follows from that fact that Y'' are exactly all attributes in which t has a constant, the fact that ν_m (recall $\bar{t} = \nu_m(t)$) preserves constants, and the way of construction Y in Step 5 of Algorithm 5.

To show that $C \wedge C_e''$ is consistent, we use the following observations:

Observation 1. $C = C_Y \wedge C_{\neg Y}$, where

1. C_Y is the conjunction of equalities of C of the form $A_i = c$, where A_i is an attribute name and c is a constant, and $A_i \in Y$; i.e., $\bar{t}[A_i] = c$, and
2. $C_{\neg Y}$ is the conjunction of equalities of C of the form $A_i = A_j$, where A_i or A_j is not in Y, and $\bar{t}[A_i] \sim \bar{t}[A_j]$.

We claim, that the attribute set of C_Y, denoted as $Attr(C_Y)$, and the attribute set of $C_{\neg Y}$, denoted as $Attr(C_{\neg Y})$, are disjoint; i.e., $Attr(C_Y) \cap Attr(C_{\neg Y}) = \emptyset$

Proof: Assume, by contradiction, that $A_i \in Attr(C_Y) \cap Attr(C_{\neg Y})$. But then, $A_i = c$ is in C, where c is a constant, and $A_i = A_j$ is in C, where $A_i \in Y$ and $A_j \notin Y$. Then, $\bar{t}[A_i] = c$, and by construction of E, $c \sim x$. Also, $\bar{t}[A_j] = s$, where s is not a constant and $s \not\sim x$ because $A_j \notin Y$. However, from $\bar{t}[A_i] \sim \bar{t}[A_j]$ we have $s \sim x$ which constitutes a contradiction, thus $Attr(C_Y) \cap Attr(C_{\neg Y}) = \emptyset$ holds.

Observation 2. $C''_e = C''_{Y''} \wedge C''_{\neg Y''}$, where

1. $C''_{Y''}$ is the conjunction of equalities of C''_e of the form $A_i = c$, where A_i is an attribute name and c is a constant, and $A_i \in Y''$; i.e., $t[A_i] = c$, and
2. $C''_{\neg Y''}$ is the conjunction of equalities of C''_e of the form $A_i = A_j$, where A_i or A_j is not in Y'' and $t[A_i] = t[A_j]$.

We claim that $Attr(C''_{Y''})$ and $Attr(C''_{\neg Y''})$ are disjoint; i.e., $Attr(C''_{Y''}) \cap Attr(C''_{\neg Y''}) = \emptyset$.

Proof: Similar to the proof of 1.

Observation 3. $C''_{\neg Y''} = C''_{Y-Y''} \wedge C''_{\neg Y}$, where

1. $C''_{Y-Y''}$ is the conjunction of equalities of $C''_{\neg Y''}$, where A_i or A_j are in Y.
2. $C''_{\neg Y}$ is the conjunction of equalities of C''_e, where A_i or A_j are not in Y.

We claim, that $Attr(C''_{Y-Y''})$ and $Attr(C''_{\neg Y})$ are disjoint; i.e., $Attr(C''_{Y-Y''}) \cap Attr(C''_{\neg Y}) = \emptyset$.

Proof: Assume, by contradiction, that $A_i = A_j$ in $C''_{\neg Y''}$, where $A_i \in Y - Y''$ and $A_j \in \neg Y$. Then, $t[A_i] = t[A_j] = \delta$, where δ is some null-value. Because ν_m preserves equalities, we know that $\nu_m(t)[A_i] = \bar{t}[A_i] \sim \nu_m(t)[A_j] = \bar{t}[A_j]$. Further, because $A_i \in Y$, $\bar{t}[A_i] \sim x$. However, because $A_j \notin Y$, $\bar{t}[A_j]$ not a constant and $\bar{t}[A_j] \not\sim x$. But this is a contradiction.

To complete the proof that $C \wedge C''_e$ is consistent, consider a graph G, in which the nodes are all the attribute names, null-values, and constants of C and C''_e. There is an edge between two nodes A and B if and only if C or C''_e contains the equality $A = B$. By contradiction, assume that $C \wedge C''_e$ is inconsistent. Then, there must exist a path in G from a constant to a different constant. Moreover, there must exist such a path, say P, of *minimal length*; i.e., every path that is shorter than P cannot be inconsistent. Let a and b ($a \neq b$) be the two end points of P, and e_1 be the first edge in P from a. The edge must be of the form (a, A); i.e., corresponding to the equation $a = A$, where A is an attribute name. Therefore, by 1 and 2, e_1 is either in $C''_{Y''}$ or in C_Y.

Case 1: If e_1 is in $C''_{Y''}$ then $t[A] = a$. Consider the second edge e_2. It must be of the form (A, c), where c is a constant, because $Y'' \subseteq Y$ and from Observations 1. and 2. we know that A cannot appear in equalities of $C''_{\neg Y''}$ or $C_{\neg Y}$. So e_2 must appear in $C''_{Y''}$ or in C_Y that have equalities of the form (A, c). If e_2 is in $C''_{Y''}$ then $c = a$, because $C''_{Y''}$ is consistent. If e_2 is in C_Y then $\bar{t}[A] = \nu_m(t)[A] = c$. Since ν_m preserves constants, c must be the same constant as a. But then, the node a appears twice in P, which contradicts the minimality of P.

Case 2: If e_1 is in C_Y then $\bar{t}[A] = a$. Consider the second edge e_2. If $A \in Y''$ then an argument similar to Case 1. can be repeated to derive contradiction to the minimality of P. Otherwise, if $A \in Y - Y''$ then, by Observations 1, 2, and 3, e_2 must be either in C_Y or in $C''_{Y-Y''}$. If e_2 is in C_Y then it must be of the form $A = c$ and, because C_Y is consistent, $a = c$ must hold, contradicting the minimality of P. If e_2 is in $C''_{Y-Y''}$ then it must be of the form (A, A_1), where A_1 is an attribute name. Let $(a, A), (A, A_1), \ldots, (A_{k-1}, A_k), (A_k, c)$ denote a sub-path of P from a to the first constant node c. By 1, 2, and 3, all edges $(A, A_1), \ldots, (A_{k-1}, A_k)$

must come from $C''_{Y-Y''}$, and $t[A] = t[A_1] = \ldots, = t[A_k] = \delta$, where δ is some null-value. The edge (A_k, c) must be in C_Y and therefore $\bar{t}[A_k] = c$. Because ν_m preserves equalities, we know that $\nu_m(t)[A] = a \sim \nu_m(t)[A_1] \sim \ldots \sim \nu_m(t)[A_k] = c$. Then, $a \sim c$ is in E. Since E cannot contain $a \sim c$ for two different constant, c must be the same constant as a. But then, the node a appears twice in P, which contradicts the minimality of P.

This completes the proof that there does not exist a path in G that connects two different constants, and, therefore $C \wedge C''_e$ must be consistent.

Finally, if t has the same null-value for attributes A_i and A_j then, because ν_m preserves equalities, attributes A_i and A_j of $\bar{t}$ must have symbols that are related by E. If A_i and A_j are not in Y then both $\bar{t}[A_i]$ and $\bar{t}[A_j]$ are two null-values which are related by E, and therefore C contains the equality $A_i = A_j$. Therefore, $C \models C''_{\neg Y}$. But then, by Proposition 3, we proved that for any $(PF'', \Pi_{Y''}\sigma_{C''})$ Algorithm 5 will generate a query $\Pi_Y\sigma_C$ such that $\Pi_Y\sigma_C \rightsquigarrow_\exists \Pi_{Y''}\sigma_{C''_e}$.

To complete Claim 7, we have to show that after the query $\Pi_Y\sigma_C$ is removed from $\mathcal{S}$ in Step 6.a of Algorithm 5 because there exist a query $\Pi_{\bar{Y}}\sigma_{\bar{C}} \in \mathcal{S}$ such that $\Pi_{\bar{Y}}\sigma_{\bar{C}} \rightsquigarrow_\forall \Pi_Y\sigma_C$ we still have $\Pi_{\bar{Y}}\sigma_{\bar{C}} \rightsquigarrow_\exists \Pi_{Y''}\sigma_{C''_e}$.

1. From $\Pi_Y\sigma_C \rightsquigarrow_\exists \Pi_{Y''}\sigma_{C''_e}$ we know that

   (a) $Y'' \subseteq Y$,

   (b) $C''_e \wedge C$ is consistent, and

   (c) $C \models C''_{\neg Y}$

2. From $\Pi_{\bar{Y}}\sigma_{\bar{C}} \rightsquigarrow_\forall \Pi_Y\sigma_C$ we know that

(a) $Y \subseteq \bar{Y}$,

(b) $C \wedge \bar{C}$ is consistent,

(c) $\bar{C} \models C_{\neg\bar{Y}}$, and

(d) $C \models \bar{C}_Y$

From 1.a and 2.a it follows that $Y'' \subseteq \bar{Y}$. To show that $\bar{C} \wedge C''_e$ is consistent, assume, by contradiction, that $\bar{C} \wedge C''_e$ is inconsistent; i.e., $C''_e \models A_i = a$ and $\bar{C} \models A_i = b$, where a and b are two different constants. By construction, both $\Pi_{\bar{Y}}\sigma_{\bar{C}}$ and $\Pi_{Y''}\sigma_{C''_e}$ are in normal form. Then, if $C''_e \models A_i = a$, $A_i \in Y''$ and since $Y'' \subseteq Y$, $A_i \in Y$. From 2.4, we know that $C \models \bar{C}_Y$, and therefore $C \models A_i = b$. Moreover, from 1.b, we know that $C''_e \wedge C \models A_i = a \wedge A_i = b$ is consistent. But this is a contradiction, therefore $\bar{C} \wedge C''_e$ is consistent. Finally, from 1.c and 2.c and and the facts that $Y'' \subseteq Y \subseteq \bar{Y}$ it follows that $\bar{C} \models C''_{\neg\bar{Y}}$. Thus, by Proposition 3, $\Pi_{\bar{Y}}\sigma_{\bar{C}} \rightsquigarrow_{\exists} \Pi_{Y''}\sigma_{C''_e}$. This completes the proof of Claim 7.

We **prove Claim 8** by induction on the number of dependency applications in Algorithm 3 and the corresponding applications in Algorithm 5. First, we show that ν_0 preserves constants and equalities. Consider tuples $t_{i_1}, \ldots, t_{i_l}$ in r_i^0 and $\bar{t}_i$ in r_2^0, as they were defined Initially for generally corresponding tuples. By construction of t_{i_j} $(j = 1, \ldots, l)$ in Step 1 of Algorithm 3, t_{i_j} contains a constant for an attribute A_k, if and only if either

1. $C_i \models A_k = c$,

2. $A_k \in X_i$, or

3. there exists $A_j \in X_i$ such that $C_i \models A_k = A_j$

In addition, by construction of $\bar{t}_i$ in Step 1 of Algorithm 5, we know that

1. if $C_i \models A_k = c$ then $\bar{t}_i[A_k] = c$
2. else if
   (a) $A_k \in X_i$ or
   (b) there exists $A_j \in X_i$ such that $C_i \models A_k = A_j$

   then $\bar{t}_i[A_k] = x$.

Then, for all attributes A_k that t_{i_j} $(j = 1, \ldots, l)$ contains a constant, $\bar{t}_i$ contains either a constant or the symbol x. If $\bar{t}_i[A_k] = c$ $(C_i \models A_k = c)$ then all tuples t_{i_j} $(j = 1, \ldots, l)$ must have $t_{i_j}[A_k] = c$ and $\nu_0(t_{i_j})[A_k] = c$; Property 1.(a) must hold. Because E, that was constructed in Step 3 of Algorithm 5, contains all reflexive pairs $x \sim x$ and $x \sim c$ for all constants c that appear in r_2 or in the dependencies, Property 1.(b) must hold, and therefore Property 1. holds. Moreover, equalities among the constants of t_{i_j} $(j = 1, \ldots, l)$ are preserved. To show that ν_0 also preserves equalities among the null-values of t_{i_j}, notice that Initially we assign unique null-values for all attributes of t_{i_j} $(j = 1, \ldots, l)$ that do not already have a constant assigned to. In Step 1.c, Algorithm 3 will equate null-values of t_{i_j} only if C_i entails their equalities. But then, by construction of $\bar{t}_i$ in Step 1.c of Algorithm 5, $\bar{t}_i$ must contain the same null-value for these attributes, and therefore ν_0 preserves equalities. This completes the proof that ν_0 preserves constants and equalities.

Assume, that the claim holds for $i-1$ application; i.e., that ν_{i-1} preserves constants and equalities. Note, that h_i is an atom mapping, thus it also preserves constants and equalities. Then, if $h_i(R[\ldots, A_j = c, \ldots]) = (c_1, \ldots, c_j, \ldots, c_n) \in r_1^{i-1}$ then $c_j = c$. Moreover, if $\nu_{i-1}(c_1, \ldots, c_j, \ldots, c_n) = (s_1, \ldots, s_j, \ldots, s_n) \in r_2^{i-1}$, then s_j

is not a different constant from c_j, and $x \sim s_j$. By construction of E we know that $x \sim c$, and therefore $s_j \sim c$, thus $\bar{h}_i(R[\ldots, A_j = c, \ldots]) = (s_1, \ldots, s_j, \ldots, s_n)$, where $s_j \sim c \sim x$, thus $\bar{h}_i$ preserves constants.

Further, if $h_i(R[\ldots, A_k = a, \ldots]) = (c_1, \ldots, c_k, \ldots, c_n) \in r_1^{i-1}$ and $h_i(R[\ldots, A_l = a, \ldots]) = (c'_1, \ldots, c'_l, \ldots, c'_n) \in r_1^{i-1}$, then $c_k = c'_l$. Moreover, if $\nu_{i-1}(c_1, \ldots, c_k, \ldots, c_n) = (s_1, \ldots, s_k, \ldots, s_n) \in r_2^{i-1}$ and $\nu_{i-1}(c'_1, \ldots, c'_l, \ldots, c'_n) = (s'_1, \ldots, s'_l, \ldots, s'_n) \in r_2^{i-1}$ then $s_k \sim s'_l$. Therefore, $\bar{h}_i(R[\ldots, A_k = a, \ldots]) = (s_1, \ldots, s_k, \ldots, s_n)$, $\bar{h}_i(R[\ldots, A_l = a, \ldots]) = (s'_1, \ldots, s'_l, \ldots, s'_n)$, and $s_k \sim s'_l$, thus $\bar{h}_i$ preserves equalities.

The only remaining part is to show, that ν_i from r_1^i to r_2^i preserves constants and equalities. If d_i is an

Equality-generating dependency of the form $B_1, \ldots, B_n \to a = b$ then a_i equates two symbols $h_i(a)$ and $h_i(b)$ of r_1^{i-1} by replacing all occurrences of one of them with the other, say all occurrences of $h_i(b)$ are replaced with $h_i(a)$. We know that all occurrence of $h_i(b)$ are mapped by ν_{i-1} into symbols of r_2^{i-1} which are related to each other by E_{i-1}. The application $\bar{a}_i$ extends E_{i-1} by adding relations $s_1 \sim s_2$ for every pair s_1, s_2 such that $s_1 \in \bar{h}_i(a) = \nu_{i-1}(h_i(a))$ and $s_2 \in \bar{h}_i(b) = \nu_{i-1}(h_i(b))$ and closes it under weak transitivity.

Since a_i equated $h_i(a)$ and $h_i(b)$ of r_1^{i-1}, and that ν_{i-1} preserves constants and equalities, we know that $\bar{h}_i(a)$ and $\bar{h}_i(b)$ cannot be different constants. Moreover, by extending E_{i-1} for all possible pairs $s_1 \sim s_2$, it is ensured that all occurrences of $h(a)$ in r_1^i are mapped into symbols of r_2^i that are related by E_i. For example, if $t_1[A_l] = h_i(a)$, $t_2[A_k] = h_i(b)$, $\nu_{i-1}(t_1)[A_l] = s_1$, and $\nu_{i-1}(t_2)[A_k] = s_2$ then, after the applications a_i and $\bar{a}_i$, we have $t_2[A_k] = h_i(a)$, $\nu_{i-1}(t_2)[A_k] = s_2$,where $s_1 \sim s_2$ is in E_i. Therefore, after the applications (d_i, h_i) on r_1^{i-1} and $(d_i, \bar{h}_i)$ on

r_2^{i-1}, where d_i is an equality-generating dependency ν_i from r_1^i to r_2^i preserves constants and equalities.

Tuple-generating dependency , and a_i generated the tuple $t = (h_i(a_1), \ldots, h_i(a_n))$ in r_1^i then $\bar{a}_i$ will generate a tuple $\bar{t}$ in r_2^i where

$$\bar{t} = (\nu_{i-1}(h_i(a_1)), \ldots, \nu_{i-1}(h_i(a_n))) = \nu_{i-1}(h_i(a_1), \ldots, h_i(a_n))$$

Then $\nu_{i-1}(t) = \bar{t}$. Clearly, ν_i from r_1^i to r_2^i coincides with ν_{i-1} on the tuples of r_1^{i-1} and r_2^{i-1}. Further, ν_i maps the new tuple $t \in r_1^i$ to the new tuple $\bar{t} \in r_2^i$, where $\nu_{i-1}(t) = \bar{t}$. But then, since ν_{i-1} preserved constants and equalities, ν_i does too.

This completes the proof of Claim 8.

The only remaining part is the proof of Claim 9. Let (r_2', E') and (r_2'', E'') denote the results of two different sequences of dependency applications, $\bar{a}_1', \ldots, \bar{a}_l'$ and $\bar{a}_1'', \ldots, \bar{a}_k''$, at the end of Step 4 of Algorithm 5. Assume, by contradiction, that (r_2', E') and (r_2'', E'') are not the same (up to the renaming of the null-values); i.e., one of them contains either a tuple or a pair $s_1 \sim s_2$ that is not present in the other. Assume, that (r_2', E') contains a tuple or a pair that is not in (r_2'', E''). Since r_2'' must contain all the tuples of r_2, and E'' must contain all of the relations of E, we can apply $\bar{a}_1', \ldots, \bar{a}_l'$ on (r_2'', E''); i.e., using r_2'' instead of r_2 and E'' instead of E. We know, that the applications $\bar{a}_1', \ldots, \bar{a}_l'$ must produce the tuple or the relation that was not in (r_2'', E''). But this is contradiction, because originally, when (r_2'', E'') was generated, the *E-Chase* process could terminated only if no more dependency could be applied that resulted in a change. Therefore, (r_2', E') and (r_2'', E'') must be equal, and this completes the proof of Claim 9.

(2) Proof of compactness: The compactness property follows directly from the way $\mathcal{S}$ is constructed in Step 6 of Algorithm 5.

7.3 Proofs of Theorems in Section 4

7.3.1 Proof of Theorem 7

Proof Sketch: Applications of constraints on $\mathcal{S}$ do not create new symbols; and the height of $\mathcal{S}$ is bounded. Therefore, there are only a finite number of paths that can be created, thus the algorithm must *terminate.*

For *soundness*, let DB be a ground tree that satisfies $\mathcal{K}$ and also satisfies T. The proof of soundness is based on the inductive statement that after each application of a constraint on $\mathcal{S}$ in the algorithm, the resulting tree $\mathcal{S}$ is still satisfied by DB. To show this, we construct a tree-mapping from $\mathcal{S}$ to DB that is based on the definition of constraint satisfaction by DB.

For *completeness*, assume by contradiction that for some T', $T \models_{\mathcal{K}} T'$ but $\mathcal{S}$ does not satisfy T' (where null-values of $\mathcal{S}$ are treated as constants). Consider a DB, constructed from $\mathcal{S}$ by replacing each null-value with a unique constant. In the proof we show that the constructed DB satisfies the constraint $\mathcal{K}$, satisfies T, but does not satisfies T'. This is a contradiction to the fact that $T \models_{\mathcal{K}} T'$.

□

7.4 Proofs of Theorems in Section 5

7.4.1 Proof of Theorem 8

Proof Sketch:

1. *Terminates:* straightforward by Step 2.

2. *Correctness:* By the construction of points a_1 and a_2 in Steps 2.c and 2.d, respectively, Algorithm 7 ensures that the data value a is within the generated interval.

3. *Safety:* (a) By comparing the volume of the restricted constraint object V' with the volume object V_l in Step 6, Algorithm 7 ensures that V' is not smaller than V_l. (b) By using a random number generator based on uniform distribution, any point within the restricted constraint structure C' has the same probability to be the database, i.e., the database is uniformly distributed within C'. Note that any new information about a data item is fully contained in the previous answers thus no possibly hazardous overlap can occur.

4. *Efficiency:* Follows from step 6.II.a.

$\square$

7.4.2 Proof of Theorem 9

Proof Sketch:

1. *Termination:* straightforward by Step 3.

2. *Correctness:* By the construction of lengths r (Step 3), r' (Step 6) and the points a_1 (Steps 4,5) and a_2 (Steps 7,8), Algorithm 8 ensures that the data value a is within the generated interval.

3. *Safety:* (a) By verifying for every data item that no disallowed *interval object* is disclosed (Step 12), Algorithm 8 ensures that the security requirements are not violated. Further, (b) by using a random number generator based on uniform distribution, any point within the restricted constraint structure C' has the

same probability to be the database, i.e., the database is uniformly distributed within C'. Similarly to the security classifications via volume objects, note that any new information about a data item is fully contained in the previous answers thus no possibly hazardous overlap can occur.

$\square$

BIBLIOGRAPHY

BIBLIOGRAPHY

[And97] J. G. Anderson. Clearing the way for physicians' use of clinical information systems. *Communications of ACM*, 40(8):83–90, August 1997.

[BB89] J. Biskup and H. H. Bruggemann. The personal model of data - towards a privacy oriented information system (extended abstract). In *Proc. of the Fifth International Conference of Data Engineering, February 6–10, 1989, Loas Angeles, California, USA*, pages 348–355. IEEE Computer Society, 1989.

[BDHS96] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 505–516, 1996.

[Bec80] L.L. Beck. A security mechanism for statistical databases. *ACM Trans. on Database Systems*, 5(3):316–338, 1980.

[BFJ00] A. Brodsky, C. Farkas, and S. Jajodia. Data disclosure and inference channels. Technical report, George Mason University, 2000.

[BFJar] A. Brodsky, C. Farkas, and S. Jajodia. Secure databases: Constraints, inference channels, and monitoring disclosure. *IEEE Trans. Knowledge and Data Eng.*, To Appear.

[BFJed] A. Brodsky, C. Farkas, and S. Jajodia. Information privacy and the inference problem. *IEEE Trans. Knowledge and Data Eng.*, To be submitted.

[BFWW00] A. Brodsky, C. Farkas, D. Wijesekera, and S. X. Wang. Constraints, inference channels and secure databases. In *Sixth International Conference on Principles and Practice of Constraint Programming, September 18 – 22, 2000.* Springer Verlag, LNCS, 2000.

[Bra96] B. Braithwaite. National health information privacy bill generates heat at scamc. *Journal of Americal Informatic Association*, 3(1):95–96, 1996.

[Buc90] L.J. Buczkowski. Database inference controller. In D.L. Spooner and C. Landwehr, editors, *Database Security III: Status and Prospects*, pages 311–322. North-Holland, Amsterdam, 1990.

[Chi78] F.Y. Chin. Security in statistical databases for queries with small counts. *ACM Trans. on Database Systems*, 3(1):92–104, 1978.

[CO81] F.Y. Chin and G. Ozsoyoglu. Statistical database design. *ACM Trans. on Database Systems*, 6(1):113–139, 1981.

[Cou97] National Research Council. For the record: Protecting electronic health information. Technical report, National Academy of Sciences, 1997.

[DDS79] D.E. Dennning, P.J. Denning, and M.D. Schwartz. The tracker. *ACM Trans. on Database Systems*, 4(1):75–95, 1979.

[DdVS99a] S. Dawson, S. De Capitani di Vimercati, and P. Samarati. Minimal data upgrating to prevent inference and association attacks. In *Proc. of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 114–125, 1999.

[DdVS99b] S. Dawson, S. De Capitani di Vimercati, and P. Samarati. Specification and enforcement of classification and inference constraints. In *Proc. IEEE Symp. on Security and Privacy*, 1999.

[Den80] D.E. Dennning. Secure statistical database with random sample queries. *ACM Trans. on Database Systems*, 5(3):291–315, 1980.

[Den82] D.E. Denning. *Cryptography and Data Security.* Addison-Wesley, Mass., 1982.

[Den85] D.E. Denning. Commutative filters for reducing inference threats in multilevel database systems. In *Proc. IEEE Symp. on Security and Privacy*, pages 134–146, 1985.

[DJL79] D. Dobkin, A.K. Jones, and R.J. Lipton. Secure databases: Protection against user influence. *ACM Trans. on Database Systems*, 4(1):97–106, March 1979.

[DO98] L. C. J. Dreyer and M. S. Olivier. Dynamic aspect of the infopriv model. In *Proc. 9th Database and Expert Systems Applications DEXA 98*, pages 340–345. IEEE Computer Society, Los Alamitos, 1998.

[DS80] D.E. Dennning and J. Schlorer. A fast procedure for finding a tracker in a statistical database. *ACM Trans. on Database Systems*, 5(1):88–102, 1980.

[gen98] Genetic information and the workplace. Technical report, Department of Labor,Department of Health and Human Services, Equal Employment Opportunity Commission Department of Justice, 1998.

[GM84] J.A. Goguen and J. Meseguer. Unwinding and inference control. In *Proc. IEEE Symp. on Security and Privacy*, pages 75–86, 1984.

[Hin88] T.H. Hinke. Inference aggregation detection in database management systems. In *Proc. IEEE Symp. on Security and Privacy*, pages 96–106, 1988.

[JM95] S. Jajodia and C. Meadows. Inference problems in multilevel secure database management systems. In M.D. Abrams, S. Jajodia, and H. Podell, editors, *Information Security: An integrated collection of essays*, pages 570–584. IEEE Computer Society Press, Los Alamitos, Calif., 1995.

[KU77] J.B. Kam and J. Ullman. A model of statistical databases and their security. *ACM Trans. on Database Systems*, 2(1):1–10, March 1977.

[Mar96] D.G. Marks. Inference in MLS database systems. *IEEE Trans. Knowledge and Data Eng.*, 8(1):46–55, February 1996.

[Mea90] C. Meadows. Extending the brewer-nash model to a multilevel context. In *Proc. IEEE Symp. on Security and Privacy*, pages 95–102, 1990.

[Mee97] B. N. Meeks. Privacy lost, anytime, anywhere. In *Communications of ACM*, volume 40/8, pages 11–13, 1997.

[MMJ96] D.G. Marks, A. Motro, and S. Jajodia. Enhancing the controlled disclosure of sensitive information. In *Proc. European Symp. on Research in Computer Security,* Springer-Verlag Lecture Notes in Computer Science, Vol. 1146, pages 290–303, 1996.

[Mor88] M. Morgenstern. Controlling logical inference in multilevel database systems. In *Proc. IEEE Symp. on Security and Privacy*, pages 245–255, 1988.

[MSS88] S. Mazumdar, D. Stemple, and T. Sheard. Resolving the tension between integrity and security using a theorem prover. In *Proc. ACM Int'l Conf. Management of Data*, pages 233–242, 1988.

[Off99] United States General Accounting Office. Medical records privacy, access needed for health research, but oversight of privacy protections is limited. Technical report, United States General Accounting Office, Report to Congressional Requesters GAO/HEHS-99-55, 1999.

[O'L95] D. E. O'Leary. Some privacy issues in knowledge discovery: Oecd personal privacy guidelines. *IEEE Expert/Intelligent Systems and Their Applications*, 10(2), April 1995.

[RGR97] A. D. Rubin, D. Geer, and M. J. Ranum. *WEB Security Sourcebook*. John Wiley and Sons, Inc., 1997.

[Rin97] T. C. Rindfleisch. Privacy, information technology, and health care. *Communications of ACM*, 40(8):93–100, August 1997.

[Sch80] J. Schlorer. Disclosure form statistical databases: Quantitative aspect of trackers. *ACM Trans. on Database Systems*, 5(4):467–492, 1980.

[Sch81] J. Schlorer. Security of statistical databases: Multidimensional transformation. *ACM Trans. on Database Systems*, 6(1):95–112, 1981.

[Smi90] G.W. Smith. Modeling security-relevant data semantics. In *Proc. IEEE Symp. Research in Security and Privacy*, pages 384–391, 1990.

[SO87] T. Su and G. Ozsoyoglu. Data dependencies and inference control in multilevel relational database systems. In *Proc. IEEE Symp. Security and Privacy*, pages 202–211, 1987.

[SO91] T. Su and G. Ozsoyoglu. Inference in MLS database systems. *IEEE Trans. Knowledge and Data Eng.*, 3(4):474–485, December 1991.

[ST90] P.D. Stachour and B. Thuraisingham. Design of LDV: A multilevel secure relational database management system. *IEEE Trans. Knowledge and Data Eng.*, 2(2):190–209, June 1990.

[Ste98] L. D. Stein. *Web Security - A Step-by-Step Reference Guide*. Addison-Wesley Longman, inc., 1998.

[TDC95] T.H.Hinke, Harry S. Delugach, and Asha Chandrasekhar. A fast algorithm for detecting second paths in database inference analysis. *Jour. of Computer Security*, 3(2,3):147–168, 1995.

[Thu87] B.M. Thuraisingham. Security checking in relational database management systems augmented with inference engines. *Computers and Security*, 6:479–492, 1987.

[Tin98] T. C. Ting. Privacy and confidentiality in healthcare delivery information systems. In *Proc. of the 12th IEEE Symposium on Computer-Based Medical Systems*, 1998.

[Ull88] J.D. Ullman. *Principles of Database and Knowledge-base Systems, Volumes 1,2.* Computer Science Press, Rockville, MD, 1988.

[WBD98] G. Wiederhold, M. Bilello, and C. Donahue. Web implementation of a security mediator for medical databases. In T. Y. Lin and S. Qian, editors, *Database Security XI Status and Prospent*, pages 60–67. Chapman and Hall, 1998.

[WBSQ96a] G. Wiederhold, M. Bilello, V. Sarathy, and X. L. Qian. Protecting collaboration. In *Proceedings of the NISSC 1996 National Information Systems Security Conference*, pages 561–569, 1996.

[WBSQ96b] G. Wiederhold, M. Bilello, V. Sarathy, and X. L. Qian. A security mediator for health care information. In *Proceedings of the 1996 AMIA Conference*, pages 120–124, 1996.

CURRICULUM VITAE

Csilla Farkas is a PhD candidate at George Mason University, Virginia. Her research interests include database security, data mining, and scientific databases. She received a BS degree in geology from Eotvos Lorand University, Hungary, in 1985, a BS in computer science from SZAMALK, Hungary, in 1989, and an MS in computer science from George Mason University, Virginia, in 1993.

This dissertation was typeset with LaTeX[‡] by the author.

[‡]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.