Numéro d'ordre 2013ISAL0067

Thèse

présentée devant
L'Institut National des Sciences Appliquées de Lyon

pour obtenir le grade de
Docteur

spécialité
Informatique

par
Thierno Mahamoudou DIALLO

Discovering Data Quality Rules in a Master Data Management context

Fouille de Règles de Qualité de Données
dans un contexte de Gestion de Données de Référence

Soutenue publiquement le 17 juillet 2013 devant le jury :

Laure Berti-Equille, IRD, Université d'Aix-Marseille	Rapporteur
Bart Goethals, Professeur, Université d'Anvers	Rapporteur
Dominique Laurent, Professeur, Université de Cergy-Pontoise	Examinateur
Martial Doré, Directeur R&D, Orchestra Networks	Invité
Sylvie Servigne, Maitre de Conférences, INSA de Lyon	Co-directrice de thèse
Jean-Marc Petit, Professeur, INSA de Lyon	Co-directeur de thèse

Orchestra Networks R&D. Paris.
Laboratoire d'InfoRmatique en Image et Systèmes d'information. Lyon.
École Doctorale Informatique et Mathématiques. Lyon.

ACKNOWLEDGEMENT

This PhD thesis is a joint work between Orchestra Networks company and LIRIS laboratory in a so called CIFRE funding.

REMERCIEMENTS

Je souhaite dans un premier temps exprimer ma profonde gratitude envers mon directeur de thèse Jean-Marc Petit et ma co-directrice de thèse Sylvie Servigne pour leur encadrement, leur conseil, leur disponibilité pendant ces trois années de thèse. Merci d'avoir partagé vos connaissances, orienté mes interrogations et alimenté de longues heures de discussions. Je remercie Jean-Marc pour la confiance qu'il m'a accordée depuis mon stage de fin d'études d'ingénieur.

Je suis aussi très reconnaissant envers Laure Berti-Equille et Bart Goethals pour le temps consacré à la relecture de mon manuscrit. Je remercie également Dominique Laurent d'avoir accepté de siéger au jury.

Cette thèse est le fruit d'une collaboration entre l'équipe Base de Données (BD) du Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS UMR 5205 CNRS, INSA de Lyon - Université Claude Bernard Lyon 1 - Université Lumières Lyon 2 - Ecole Centrale de Lyon) et l'entreprise Orchestra Networks. La thèse a été financée par Orchestra Networks et par le ministère de l'enseignement supérieur et de la recherche par le biais d'une convention CIFRE de l'ANRT. Merci à Orchestra Networks d'avoir initié ce projet. Merci à Martial Doré et Jean-Baptiste Mestelan et à toute l'équipe R&D : Ghassen, Manuel, Vincent, Ludovic, Laure, David, Yves, Camille. J'ai beaucoup appris en vous côtoyant et je continue dans une ambiance de travail agréable. Merci également à Pierre, Christophe et Eric.

Merci à Noël Novelli du Laboratoire d'Informatique Fondamentale de Marseille (LIF UMR 7279 CNRS, Université de la Méditerranée - Université de Provence) pour sa collaboration. Merci aux membres de l'équipe BD du LIRIS. Je remercie les doctorants et anciens doctorants avec qui j'ai partagé ces trois années de recherche à Lyon: Yann, Brice, Usman, Benjamin, Deming, Simon, Arnaud... Merci également aux membres du personnel LIRIS en particulier Mabrouka et Caroline pour leur aide administrative lors de cette thèse.

Je remercie ma famille pour leur soutien, en particulier mes parents Elhadj Mouhamadou Sarifou et Fatoumata Diouldé, mes frères et soeurs Alpha Oumar, Aissatou, Abdoulaye, Thierno Sadou, Mouhamadou Moukhtar et Houssainatou. Merci à ma belle famille en Belgique. Je remercie également mes amis, particulièrement Moussa et Gérard. Mes pensées vont à Vincent Dominique Diedhiou qui a été présent à mes cotés au début de cette thèse et qui nous a quitté. Merci à toutes celles et tous ceux qui m'ont soutenu tout au long de cette aventure.

Enfin un grand merci à ma superwoman Hadiatou. Sans elle je ne serais pas arrivé au bout. Merci "boborou" de m'avoir encouragé et supporté.

ABSTRACT

Dirty data continues to be an important issue for companies. The datawarehouse institute [Eckerson, 2002], [Rockwell, 2012] stated poor data costs US businesses $611 billion dollars annually and erroneously priced data in retail databases costs US customers $2.5 billion each year. Data quality becomes more and more critical. The database community pays a particular attention to this subject where a variety of integrity constraints like Conditional Functional Dependencies (CFD) have been studied for data cleaning. Repair techniques based on these constraints are precise to catch inconsistencies but are limited on how to exactly correct data.

Master data brings a new alternative for data cleaning with respect to it quality property. Thanks to the growing importance of Master Data Management (MDM), a new class of data quality rule known as Editing Rules (ER) tells how to fix errors, pointing which attributes are wrong and what values they should take. The intuition is to correct dirty data using high quality data from the master. However, finding data quality rules is an expensive process that involves intensive manual efforts. It remains unrealistic to rely on human designers.

In this thesis, we develop pattern mining techniques for discovering ER from existing source relations with respect to master relations. In this setting, we propose a new semantics of ER taking advantage of both source and master data. Thanks to the semantics proposed in term of satisfaction, the discovery problem of ER turns out to be strongly related to the discovery of both CFD and one-to-one correspondences between sources and target attributes.

We first attack the problem of discovering CFD. We concentrate our attention to the particular class of constant CFD known as very expressive to detect inconsistencies. We extend some well known concepts introduced for traditional Functional Dependencies to solve the discovery problem of CFD. Secondly, we propose a method based on INclusion Dependencies to extract one-to-one correspondences from source to master attributes before automatically building ER. Finally we propose some heuristics of applying ER to clean data. We have implemented and evaluated our techniques on both real life and synthetic databases. Experiments show both the feasibility, the scalability and the robustness of our proposal.

CONTENTS

Contents **1**

1 Introduction **3**

1.1 Context 4
1.2 Contributions 6
1.3 Document Organization 8

2 Master Data Management **9**

2.1 Definitions and Issues 10
2.2 Implementation Styles 14
2.3 MDM and Data Quality 17
2.4 Summary 18

3 Preliminaries **19**

3.1 Relational Model 20
3.2 Conditional Functional Dependencies 23
3.3 Editing Rules 29

4 Discovering Conditional Functional Dependencies **31**

4.1 Related Works 32
4.2 New notations for CFD 34
4.3 Discovering constant CFD using conditional agree sets 37
4.4 Frequent constant CFD discovery 43
4.5 Experimentation 49
4.6 Summary 53

5 Discovering Editing Rules **55**

5.1 New Semantic of Editing Rules 57
5.2 Discovering Editing Rules 59
5.3 Related Works 68
5.4 Experimentation 69
5.5 Summary 74

6 Editing Rules for Data Cleaning **76**

6.1 Data Repairing based on CFD 77
6.2 Data Repairing based on ER 78
6.3 Related Works 80
6.4 Experimentation 81

6.5 Summary 83

7 Conclusion **85**
7.1 Summary of Contributions 86
7.2 Discussion and Future works 87

List of Figures **90**

Bibliography **92**

CHAPTER 1

INTRODUCTION

Chapter Outline

1.1 Context **4**

1.1.1 Master Data Management 4

1.1.2 Conditional Functional Dependencies 5

1.1.3 Editing Rules 5

1.2 Contributions **6**

1.2.1 Discovering Conditional Functional Dependencies 6

1.2.2 Discovering Editing Rules 7

1.2.3 Data repair techniques based on Editing Rules 7

1.3 Document Organization **8**

This first Chapter presents the industrial context of our work. Issues of Master Data Management are described. The global challenges of data quality rules in term of data mining and data cleaning are introduced. We present our contributions before giving an overview of the document organization.

1.1 Context

Poor data quality continues to have a serious impact on companies. Data error rates in industry reached 30 % [Redman, 1998] and are more and more critical. For example poor data costs US businesses $611 billion dollars annually and erroneously priced data in retail databases costs US customers $2.5 billion each year stated the datawarehouse institute [Rockwell, 2012], [Berti-Equille, 2006] , [Eckerson, 2002]. PricewaterhouseCoopers concluded in their study [Webster, 2001] that 1/3 of system development projects were forced to delay or cancel due to poor data quality. Worst, 30 to 80% of the development time and budget for data storing and warehousing are for data cleaning according to a study for Merill Lynch [Shilakes and Tylman, 1998]. Therefore, finding efficient techniques to correct dirty data is an important issue. Database community pays a particular attention to this subject and a variety of integrity constraints such as Conditional Functional Dependencies [Bohannon et al., 2007] have been recently introduced in this setting.

In addition, the emergence of Master Data Management (MDM) brings new opportunities for improving the quality of data. According to [Loshin, 2009] master data is a single repository of high-quality data that provides various applications with a synchronized, consistent view of its core business entities. Therefore data quality is seen by companies as a key feature of MDM solutions. Orchestra Networks [Orchestra-Networks, 2012] is one of the leading MDM software vendor [Gartner, 2010] with the multidomain MDM solution called EBX. This thesis address new data mining, data quality and data integration challenges in this particular MDM industrial context.

1.1.1 Master Data Management

The amount of data shared across information systems seems to have exploded and the sources are more and more heterogeneous [Loshin, 2009]. It is necessary to build efficient techniques in place that can integrate data into a consistent view available across the information system. Master data can be considered as the core high quality data that help fulfill that need. MDM is particularly driven by industries [Deloitte and Oracle, 2005], [Russom, 2008], [Loshin, 2009], [Power, 2010] with quite different views. This highlights the need in this thesis to have a closer look on how precisely master data can be defined and also what are the main issues and principles of MDM. Gartner [Eschinger, 2008] expected a significant grow of MDM market segment from $1 billion in 2007 to $2.8 billion in 2012, despite the current economic gloom. Some companies such as Orchestra Networks [Power, 2010] address multidomain master data to cover all type of data, other companies like [Deloitte and Oracle, 2005] are focused on the particular customer or product data.

In this thesis we describe how MDM is related to common database issues such as data integration and data quality. In particular MDM offers efficient solutions, such as Editing Rules [Fan et al., 2010] as a relevant alternative to Conditional Functional Dependencies in a data cleaning setting.

1.1.2 Conditional Functional Dependencies

Initial work on dependency-based data quality methods focused on traditional dependencies, such as Functional Dependencies (FD) [Armstrong and Delobel, 1980] [Beeri et al., 1984a] [Kivinen and Mannila, 1995], that were mainly developed for database design 40 years ago. Their expressiveness often limits the capture of inconsistencies or the specification of dependency rules.

Even if [Kivinen and Mannila, 1995] defined some error measures for FD, none of them can easily capture the frequency (or support) measure popularized by Association Rules (AR) mining [Agrawal et al., 1993]. These limitations highlighted the need for extending either FD to take into account attribute/values or inversely, extending AR to take into account attributes.

In this setting, Conditional Functional Dependencies (CFD) have been recently introduced by [Bohannon et al., 2007] as a compromise to bridge the gap between these two notions. Therefore they can be seen as an unification of FD and AR since they allow to mix attributes and attribute/values in dependencies. The intuition behind CFD is that the constraint holds only on a subset of a relation instead on the entire relation. From a data mining point of view, CFD offer new opportunities to reveal data inconsistencies or new knowledge nuggets from existing tabular datasets.

Data cleaning is one of the main application of CFD. The first step of this data cleaning application is detecting CFD violation. The second and last step is data repair based on CFD, where they allow attribute/value modifications as a repair operation. Clearly, for this process to be effective, it might be useful to automatically discover CFD from a particular database instance. **This is our first data mining challenge**.

In practice, CFD are constraints that help to determine whether errors are present in the data but they fall short of telling us which attributes are exactly concerned by the error. Moreover they are limited on how to efficiently correct errors. Worst, heuristic methods based on CFD may introduce new errors when trying to repair the data [Fan et al., 2010]. These limitations motivate the recent introduction of Editing Rules.

1.1.3 Editing Rules

Editing Rules (ER) are a new class of data quality rules that tells how to efficiently fix errors, i.e. which attributes exactly are wrong and what values they should take [Fan et al., 2010]. ER are boosted by the recent development of MDM and then are defined in term of master data. The global intuition is to help fix and enrich dirty data using the corresponding values from master data.

However, for ER to be effective in practice, it is necessary to have techniques in place that can automatically discover such rules from both source data and their corresponding master data. Indeed it is often unrealistic to rely on human experts to design them by hand via an expensive and long manual process. This practical concern highlights the need for studying the discovery problem of ER. **This is our second data mining challenge**. No contribution has been made for the discovery problem of ER to the best of our knowledge.

Mining data quality rules is challenging, but designing data repair techniques to efficiently apply ER is also important. The latter step has being studied when dealing with CFD [Bohannon et al., 2007], [Cong et al., 2007], [Yakout et al., 2011] and other old class of integrity constraints like FD or INclusion Dependencies [Bohannon et al., 2005]. In this thesis, we tackle this issue using ER.

1.2 Contributions

In this thesis, we address data quality challenges in MDM. We tackle the problem of mining data quality rules and we propose efficient techniques to address these issues. We first attack the discovery problem of CFD in a single relation and then the discovery problem of ER in databases. We also propose heuristics to apply ER. We have implemented the proposed techniques (join work with Noël Novelli). Experiments show both their feasibility and their robustness.

1.2.1 Discovering Conditional Functional Dependencies

In this thesis, we present our proposal on CFD inference which can be seen as a clarification of some simple and basic notions underlying CFD. We address the specific class of constant CFD. The constant class enforces conditions to be only composed by constants. We focus on two types of techniques:

- the first one extends the notion of agree sets initially used for the definition of Armstrong relations [Beeri et al., 1984a] for FD. Our technique based on agree sets helps to easily capture the violation of the CFD in order to solve the discovery problem.

- the second one extends well known techniques used for AR mining [Agrawal and Srikant, 1994] and FD mining [Novelli and Cicchetti, 2001a]. We point out how these data mining techniques can be reused for constant CFD mining.

- finally the techniques have been implemented and experiments on real life data and synthetic data have been carried out. The results show both the feasibility and the scalability of our proposal.

This work has been published to the 10th International French Conference on Knowledge Mining and Management (*Extraction et Gestion des Connaissances, EGC Tunis 2010*) [Diallo and Novelli, 2010] and has also been published to the special issue "Interesting Knowledge Mining" of the 2012 International Journal of Data Mining, Modelling and Management (IJDMMM) [Diallo et al., 2012].

1.2.2 Discovering Editing Rules

We also define and provide solutions for discovering ER in databases provided with source data and master data. The problem turns out to be strongly related to the discovery of both CFD and correspondences from source to master attributes.

In particular we introduce a new semantic of ER allowing to precisely define the discovery problem as a pattern mining one. The early classical semantic of ER in term of edition is extended to a semantic of satisfaction thanks to the introduction of a mapping function. The latter defines the correspondence between source attributes and master attributes as a schema matching problem [Rahm and Bernstein, 2001]. The definition of the mapping function is challenging, specially when dealing with INclusion Dependencies [Lopes et al., 2002].

Thanks to this new semantic, we present an algorithm for discovering ER. As far as we know, no contributions have been made for the discovery problem of ER since their recent introduction. We finally present an experimental evaluation of our techniques demonstrating their usefulness for discovering ER.

This work has been published to the 10th International Workshop on Quality in Databases (QDB) in conjunction with Very Large Databases (VLDB) conference in Istanbul 2012 [Diallo et al., 2012a] and has also been published to the 28th french Advanced Database Days (*journées Bases de Données Avancées, Clermont Ferrand, 2012*) [Diallo et al., 2012b].

1.2.3 Data repair techniques based on Editing Rules

We finally propose an algorithm to apply ER in a data cleaning setting. Different heuristics are provided to improve the efficiency of the algorithm. One strategy of improvement is to repeat the process of applying ER in order to maximize the correction of dirty data. One other strategy is to find an efficient order in which ER have to be applied. We show benefit when ER are sorted with respect to their support before applied. We finally evaluate our heuristics on real life data and show that the proposed optimization improve performance in repair quality without a heavy cost.

1.3 Document Organization

After this current Introduction Chapter, we present a global overview of Master Data Management in **Chapter 2**. A synthesis of master data definitions is addressed before presenting the key features, the issues and the possible architectures of MDM systems. We also present the interaction between MDM and data quality which is relevant for our work.

Preliminary definitions are given in **Chapter 3**. This includes structure and integrity constrains of relational model. We detail important preliminary definitions about CFD and ER.

In **Chapter 4**, the discovery problem of CFD is presented. Related works are discussed. A new notation of CFD is introduced before tackling the search space and the main properties of the latter. Two methods of discovering constant CFD are proposed. Finally, the experimentation is addressed on both synthetic and real life datasets.

In **Chapter 5**, the discovery problem of ER is presented. A new semantic of ER is defined. The matching issue from source schema to master schema is defined. The use of INclusion Dependencies to tackle the issue is detailed. The algorithm for mining ER is described before discussing related works. Finally the experimentation on both synthetic and real life datasets is addressed.

In **Chapter 6**, data repair techniques based on ER are proposed before discussing related works. Finally the experimentation and the optimization techniques are detailed.

A summary of the thesis contribution is presented and discussed in **Chapter 7**. We finally conclude by addressing perspectives about this work.

CHAPTER 2

MASTER DATA MANAGEMENT

Chapter Outline

2.1 Definitions and Issues **10**
2.2 Implementation Styles **14**
2.3 MDM and Data Quality **17**
2.4 Summary **18**

In this Chapter we investigate the definition and issues of Master Data Management. We present the most common implementation styles of a MDM system. We finally address the interaction of MDM and Data Quality management which remains relevant for our work, specially for data cleaning.

2.1 Definitions and Issues

"What is master data?" is the first question we asked at the begining of this thesis. Naturally, we searched for a definition and we found...many. In fact there exists no standard definition of a master data [Bonnet, 2010]. Nevertheless, it is interesting to collect different views about definitions of master data and thus master data management. According to [Bonnet, 2010], a data belongs to the class of master if one of these situation occurs:

- the data is valued by transactions.
- the data is duplicated across many systems.
- the data is exchanged with third parties outside the information system.

For [Deloitte and Oracle, 2005], master data is a set of core data elements such as customer, product, legal entity, chart of accounts, employee, vendor, market channel, geographic location, etc., that span the enterprise IT systems and drive the business.

Additionally, [Morris and Vesset, 2005] consider master data as data that are shared across systems (such as lists or hierarchies of customers, suppliers, accounts, or organizational units) and are used to classify and define transactional data. For example, a company may record the transaction of selling Product A to Customer X on 1/1/06 for $100. Taken as a whole, this is a single piece of transaction data. However, embedded in the transaction are various elements of master data i.e. Product A and Customer X that help define the transaction. This example joins the transaction situation of [Bonnet, 2009].

In the context of business data processing, master data denotes a company essential basic data which remain unchanged over a specific period of time. It is the basis for business processes [Loser et al., 2004].

The classification of [Bonnet, 2010] emphasizes that master data are not transaction data but are involved in transaction and moreover are duplicated and exchanged. These characteristics should highlight the need to have a single view of master data in order to exchange data in an easy and efficient way. *The relevant facts we learn from these definitions suggest that master data are considered critical for businesses, shared across the information system and remain unchanged over a specific time.* Therefore it is important to spend time and money to manage master data. It is easy to understand that an inconsistency in master data may damage all applications that use and share it.

Understanding master data is a first step towards defining and understanding Master Data Management (MDM). For [Russom, 2008] MDM can be seen as the practice of defining and maintaining consistent definitions of business entities, then sharing them via integration techniques across multiple IT systems within an enterprise and sometimes beyond to partnering companies or customers.

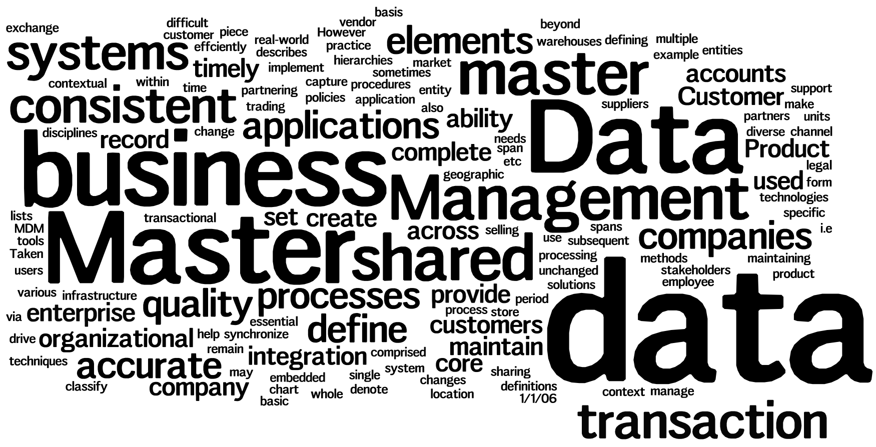
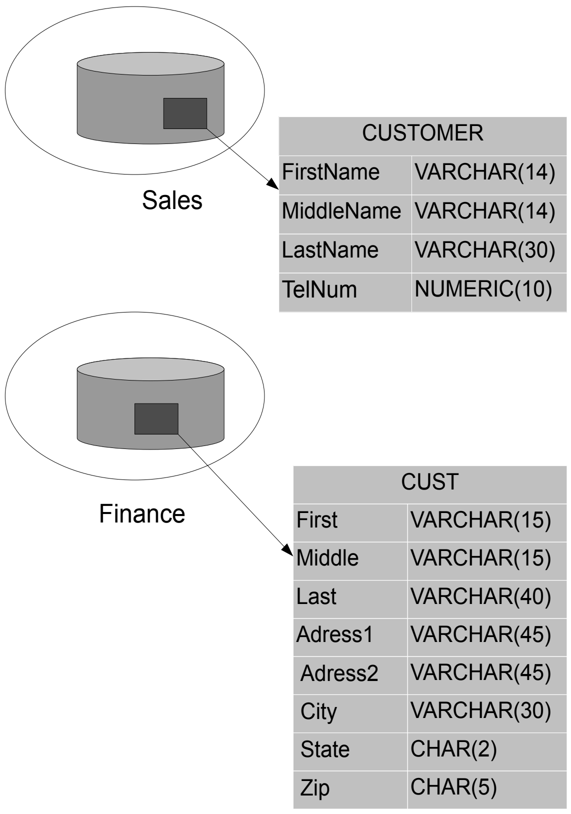


Figure 2.1: A word cloud of MDM survey definitions

Figure 2.1 represents a word cloud of the different definitions of MDM referenced in this thesis [Loshin, 2009], [Russom, 2008], [Bonnet, 2010], [Deloitte and Oracle, 2005], [Business-Intelligent-Network, 2007], [IBM, 2007]. We note some relevant words such as business, shared, consistent, . . . that appear in many definitions. MDM is set for business (objects, users, systems, . . .). It support the share of consistent, high quality data to get in fine a single view of the truth. Some category of data such as customer or product are good candidates for master data. Even if master data does not change continuously over time, it is important that MDM allows to keep track of that evolution through data version management for example.

For [Deloitte and Oracle, 2005] MDM is a process that spans all organizational business processes and application systems. It can provide companies the ability to create, store, maintain, exchange, and synchronize a consistent, accurate, and timely "system of record" for the core master data elements. It also can provide companies with the ability to more efficiently make and manage changes to their master data as the needs of the business change.

The [Business-Intelligent-Network, 2007] consider MDM as comprised of the business applications, methods, and tools that implement the policies, procedures and infrastructure to support the capture, integration, and subsequent shared use of accurate, timely, consistent, and complete master data.



CUSTOMER	
FirstName	VARCHAR(14)
MiddleName	VARCHAR(14)
LastName	VARCHAR(30)
TelNum	NUMERIC(10)

CUST	
First	VARCHAR(15)
Middle	VARCHAR(15)
Last	VARCHAR(40)
Adress1	VARCHAR(45)
Adress2	VARCHAR(45)
City	VARCHAR(30)
State	CHAR(2)
Zip	CHAR(5)

Figure 2.2: Different applications modeling the same customer differently [Loshin, 2009]

MDM describes a set of disciplines, technologies and solutions used to create and maintain consistent, complete, contextual and accurate business data for all stakeholders (users, applications, data warehouses, processes, trading partners) [IBM, 2007].

According to [Russom, 2008], MDM is difficult to define because its real world applications are so diverse. Therefore MDM is driven by companies, and each has it own view. Nevertheless the issues and the principles remain the same.

In many information systems in place, different applications may represent similar business concepts in different ways [Loshin, 2009]. Every business application may value some attributes over others. For example considering information related to a customer, the telemarketer may insist on accuracy of the telephone number to avoid calling the same prospect twice; the sales representative is concerned about duplication; the shipping department craves high-quality location information. Figure 2.2 illustrates this critical aspect when multiple codes exist for the same object. It is just an example among others of the data integration issue involved. Additionaly, similar name is used to refer to object that are completely different. Moreover organizations change quickly and continuously, companies may report data monthly or per region . . . Therefore it is important for an organization to identify critical data, integrate them into a consistent view and share it across the information system, this is one of the main goal of MDM. For [Loshin, 2009] the MDM challenge is envisioning how to organize an enterprise view of the organization's key business information objects and to govern their quality, use and synchronization to optimize the use of information to achieve the organization's operational and strategic business objectives.

One can consider MDM issues can be solved using Data Warehousing. This is not the case. MDM issues go beyond data warehousing challenges. Data warehousing incorporates data stores and conceptual, logical, and physical models to support business goals and end-user information needs [The-Data-Warehouse-Institute, 2012]. Indeed a data warehouse federate data from multiple systems mainly for reporting purpose. But historical reporting may be problematic if the changes to master data over time are not managed as versions. Moreover a data warehouse does not build a synchronization system in contrast to MDM. Wharehouses provide limited data management capabilities. In practice, warehouses are a good complement to a MDM solution, rather than being the solution themselves.

In another comparison line, Extract-Transform-Load (ETL) capabilities can be used for extracting master data from multiple sources and loading the data into the master data repository. ETL is not a management technique but a purely technical process with data integration capabilities. Therefore setting an ETL system is not really managing master data. For example, using ETL to try to manage master data does not allow to take into account business rules. This distinction is important to notice, we were confronted with this question several times during our PhD thesis.

It is also important to note that Enterprise Resource Planning (ERP) solutions are consumers of master data. They are not MDM solution. MDM issues have to be considered as a whole and every solution must be designed with respect to the context and the complexity of the problem. There is no universal MDM implementation which solves all MDM needs but different standard implementation techniques coexist.

2.2 Implementation Styles

Different implementation “styles” of MDM can be considered, applicable in different situations and providing different capabilities. [Gartner, 2010] and other analysts describe four different techniques of MDM deployment, borrowed and presented in the following.

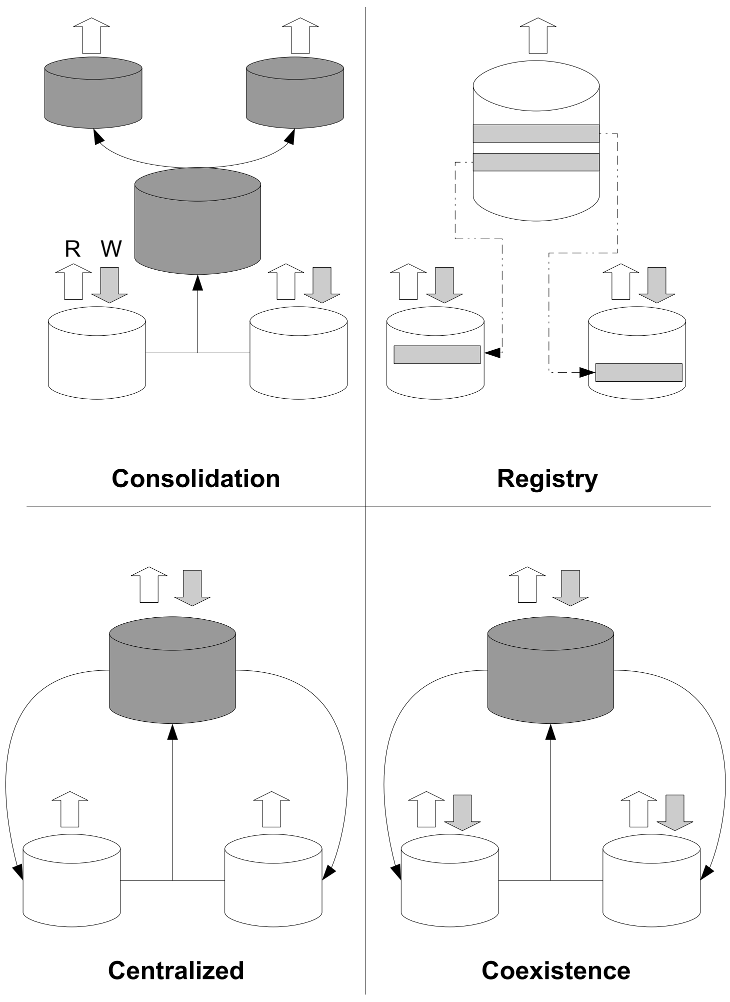


Figure 2.3: Different MDM architecture implementations [Gartner, 2010]

The consolidation style achieves a single version of master data mainly for lookup or Business Intelligence (BI) purposes. Master data is authored in the source systems, then copied to the central "hub" where it undergoes a match-

and-merge process to create a trusted copy. There is no publishing or use for data in any operational systems, only in BI environments. A complication emerges once such a data source is used as a source for new applications that create new data as a result. Therefore, the style shifts from consolidation to one of the other styles where there is an explicit desire to fix source data.

The registry style matches and links master data from source systems to create and maintain a central index into the master data. Different versions of the truth are held in the index and, at runtime, the system assembles a point-in-time composite view. This style is a relatively noninvasive, virtual approach and requires less governance agreement relative to the styles that maintain a physical trusted record.

Data are not duplicated, the index stores only the source record IDs and the key data values needed for mapping techniques similar as one used in data integration issues [Bellahsene et al., 2011], [Rahm and Bernstein, 2001]. There is no consolidation of data due to the use of an index of correspondence, very useful for real time reference. The registry style is fast to build but can be complex to manage. No data is sent back to the system. This architecture avoids the problem of over-writing data in the sources. However, performance can be an issue in this architecture, since the index have to be maintain continuously and since calls are made to each source which can cost a lot when there are many.

The centralized style supports a centralized repository of all the master data for authorship, storage and validation, and is the most invasive style, due to the change in application and information architecture. This is commonly desired when there is a high demand for automated integration between source systems and MDM infrastructure like in ETL techniques.

The master data are accessed directly in read and write. All records must be created and modified using the MDM system which maintains a single version of the truth spread back to the sources. The latter may contain additional information which are only relevant locally. The architecture has the advantage of providing a synchronized master data repository but all applications existing on the source systems require changes because the insertion of data takes place directly in the MDM system. This style is also known as transactional hub. The Orchestra Networks EBX solution uses a centralized architecture style.

The coexistence style recognizes that master data may be authored and stored in different systems across a heterogeneous and distributed environment. It creates greater consistency and data quality across systems, and rapid access to a single version (publishing that view to subscribing systems). This style is much more complex than the other styles because it is not really

Style	Consolidation	Registry	Coexistence	Centralized
What	Aggregate master data into a common repository for reporting	Maintain thin system of record with links to more complete data spread across systems; useful for realtime reference	Manage single view of master data, synchronizing changes with other systems	Manage single view of master data, providing access via services
Benefits	Good for preparing data to feed downstream systems	Complete view is assembled as needed; fast to build	Assumes existing systems unchanged, yet provides read-write management	Support new and existing transactional applications
Drawbacks	Read only; not efficient	Read mostly; may be more complex to manage	Not always consistent with other systems	May require changes to existing systems to exploit

Figure 2.4: An overview of MDM Implementation styles [Dreibelbis et al., 2008]

one style. Some instances represent simple publish/subscribe models (ERP pushes data out to a best-of-breed application), while others (newly emerging) mix and match where individual attributes persist that, combined at runtime (i.e., transaction request), represent the master data.

The coexistence style is a mix between consolidation and registry styles. The MDM system is not just a system of record like in consolidation architecture because master data can be updated in other application. Therefore a particular attention must be paid when synchronizing changes to avoid inconsistencies.

The different architecture are often complementary and also additive. One MDM domain may start a deployment with a given style and terminates by extending the system with another one, most of the time the coexistence style. Figure 2.4 give an overview, benefits and drawbacks of each implementation style.

2.3 MDM and Data Quality

On the one hand and according to [Loshin, 2009], there is a codependence between MDM and Data Quality. Actually, MDM success depends on high-quality data. The Data Quality techniques can be applied to any style seen in section 2.2 and then supports the reliability of master data. On the other hand, Data Quality can be improved through a MDM program. Integration tools driven in a MDM process identify and reduce errors on data. This codependence does not mean managing master data is equivalent to ensuring data quality. MDM requires collaborative processes and decisions through data governance. Data quality procedures can be called in this setting but MDM goes beyond data quality management [Russom, 2008].

The quality of data in general depends on many facts. For example poor data definition, wrong metadata design between operating system, duplicate values... In particular, Figure 2.5 summarizes some dimensions on which master data quality can be measured. These dimensions are commented in details by [Batini and Scannapieco, 2006], [Fan and Geerts, 2012] and must be respected in a MDM system.

Dimension	**Definition**
accuracy	the extent to which data are correctly representing an action or a real world object
completeness	the extent to which values are present in a data collection
timeliness	the extent to which data represents the real world at a given point in time
consistency	the extent to which data knowable in one database correspond to the data in a redundant or distributed database
relevancy	the extent to which data is applicable and helpful for the task at hand
accessibility	the extent to which data is available at a given point in time

Figure 2.5: Data Quality Dimensions [Otto and Ebner, 2010]

Data quality is critical to a MDM system, leading some expert like [Loshin, 2009] to define master data with respect to data quality. In this thesis we rely on this vision of master data to address problems.

We get back to one of the original drivers for data quality tools: "correcting dirty data". Data quality rules are then mined from master data. Based on these rules we build tools in a data cleaning setting to efficiently correct dirty source data.

2.4 Summary

MDM is an on-growing subject and is essentially driven by companies. Identifying and maintaining “golden data” is a dream for most of companies worldwide, MDM fulfills that dream. MDM should not be thought as an application. Rather it is a set of good practice and processes which aim to provide a unique consistent view of data to all business users and services among many other benefits. Although the centralized implementation style is very common in use, the architecture of a MDM system varies with respect to business needs. Moreover, data quality issues, such as data cleaning, addressed by the database community can benefit from MDM.

Chapter 3

Preliminaries

Chapter Outline

3.1 Relational Model **20**
  3.1.1 Structure 20
  3.1.2 Constraints 22
3.2 Conditional Functional Dependencies **23**
  3.2.1 Definition and Semantics 24
  3.2.2 Consistency 27
  3.2.3 The implication problem 27
  3.2.4 Minimal cover 28
3.3 Editing Rules **29**

In this Chapter we address preliminaries of this thesis. Relational model is presented, particularly the structural and the constraints parts. We present the class of CFD recently introduced. Finally we give preliminary notions about ER.

3.1 Relational Model

This section describes a formalization of relational databases [Abiteboul et al., 2000], [Levene and Loizou, 1999]. The relational data model, or simply the relational model, is a combination of the following three components:

- Structural: how the data are organized?
- Integrity constraint: how to prevent the insertion of "non valid" data.
- Languages: how to query the data? tipically relational algebra, relational calculus or Datalog.

The structural and integrity parts are relevant for us, the languages are omitted.

3.1.1 Structure

The relational model has a unique data structure, the *relation* represented by a two dimensions table. The columns give the characteristic of what is represented. The lines are values of what is represented. The first line indicating column names are called *attributes*.

Let U be an countable infinite set of attribute names denoted by *univers*. Let D be a countable infinite set of constant values. The possible values that can constitute a database is D.

Definition 3.1. *For an attribute $A \in U$, we denote $DOM(A)$ the domain of A, i.e. the subset of D in which the values of A are taken: $(DOM(A) \subseteq D)$.*

A relational symbol is a symbol associated to an integer type(R) corresponding to the number of attributes of R. Each symbol R is associated to a function att_R defined from {1,...,type(R)} in the univers U allowing to associate attributes of U to R in a particular order.

We denote $schema(R) = \{att_R(1), \ldots, att_R(type(R))\}$ the set of attributes of R, in other words the schema of R. In the sequel we do not distinguish a relation symbol from its schema when clear from context, i.e. $R = schema(R)$.

Example 3.1. *Let Pers be a relation symbol with $type(Pers) = 4$. The schema of Pers is $schema(Pers)$=$\{att(1), att(2), att(3), att(4)\}$ with $att(1) = id$, $att(2) = name$, $att(3) = firstname$, $att(4) = age$.*

Definition 3.2. *Let R be a relational symbol with $schema(R)$=$\{A_1,\ldots,A_m\}$ and $att_R(i)$=A_i, $i = 1, m$. A **tuple** on R is an element of the Cartesian product $DOM(A_1) \times \ldots \times DOM(A_m)$*

A *relation* over R is a finite set of tuples on R.

Person	id	name	firstname	age
	12	Dupont	Pierre	45
	45	Diallo	Hadi	35

Figure 3.1: The relation Person

Person	id	name	firstname	age
	12	Dupont	Pierre	45
	45	Diallo	Hadi	35

Department	dep	address
	IF	10 rue de la republique
	GC	87 rue de bruxelles

Work	id	dep	activity
	12	IF	Prof
	45	IF	Prof
	45	GC	Dr

Table 3.1: A database d=$\{Person, Department, Work\}$

Example 3.2. *The Person relation illustrated on Figure 3.1 is a relation on the symbol Pers. The tuple <12,Dupont,Pierre,45> belongs to the relation Person.*

Definition 3.3. *An active domain of an attribute $A \in$ schema(R) in r, denoted ADOM(A) is the set of constant values taken by A in r. In other words $ADOM(A,r) = \pi_A(r)$.*

Example 3.3. *Let us consider the relation Person in Figure 3.1, we have:*

$$ADOM(name) = \{Dupont, Diallo\}$$

The active domain of an attribute can be extended to a relation. It is the union of active domains of attributes.

Definition 3.4. *The active domain of a relation r, denoted by ADOM(r) is:*

$$ADOM(r) = \bigcup_{A \in schema(R)} ADOM(A,r)$$

Let $R = \{R_1, \ldots, R_n\}$ be a database schema, we can define a database $d = \{r_1, \ldots, r_n\}$ over R with r_i defined over $R_i (i = 1..n)$.

Example 3.4. *Let Person, Department and work be a set of relations defined respectively on schemas Pers, Depart and Working. One can defined the database* d=$\{Person, Department, work\}$ *in Table 3.1.*

In the sequel, we shall use classical database notions defined by [Levene and Loizou, 1999]. Letters from the beginning of the alphabet $(A, B, C, \ldots)$ shall represent single attribute whereas letters from the end of the alphabet $(X, Y, Z, \ldots)$ attribute sets. For convenience, XY will refer to as $X \cup Y$.

3.1.2 Constraints

Integrity constraints can be viewed as logic statements that restrict the set of allowable relations in database. For example stating that id is the primary key of the relational schema *Pers* is in fact defining an integrity constraint that specifies: no distinct two tuples in a relation over *Pers* have the same id. This notion of primary key is a particular case of the more general class of FD.

3.1.2.1 Functional Dependencies

The concept of FD helps define a relational schema without anomalies. In practice, they have been intensively studied in database design.

Definition 3.5. *Let R be a relation schema. A FD over R is an expression in the form* $R : X \rightarrow Y$*, or simply* $X \rightarrow Y$ *when R is clear from context, where* $X, Y \subseteq schema(R)$*. A FD* $X \rightarrow Y$ *is said :*

- *trivial if* $Y \subseteq X$
- *standard if* $X \neq \emptyset$

cust	CC	AC	phn	type	FN	LN	STR	CT	ZIP	item
t_1 :	01	908	1111111	1	Mike	Brant	Tree Ave.	NYC	01974	CD
t_2 :	01	908	1111111	1	Rick	James	Tree Ave.	NYC	01974	DVD
t_3 :	01	212	2222222	2	Joe	Dalton	Elm Str.	NYC	01202	CD
t_4 :	01	212	2222222	2	Jim	Cook	Elm Str.	NYC	01202	Book
t_5 :	01	215	3333333	2	Ben	Wade	Oak Av.	PHI	01202	CD
t_6 :	44	131	4444444	1	Ian	Duc	High St.	EDI	03560	DVD
t_7 :	44	140	5555555	2	Kim	Lee	High St.	PHI	03560	DVD

Figure 3.2: relation *cust*

Example 3.5. *Let us consider the relation cust described in Figure 3.2. Cust represents a customer with country code (CC), area code (AC), phone number (phn), mobile phone or land line (type), first name (FN), last name (LN), street (STR), city (CT), zip code (ZIP) and item bought by customer (item). For example, we can state the following FD:*

- $f_1 = CC, AC, phn \rightarrow STR, CT, ZIP$
- $f_2 = CC, AC \rightarrow CT, ZIP$
- $f_3 = CC, ZIP \rightarrow STR$

Semantics: a FD is then a constraint defined on a schema. The constraint must be verified or satisfied by any relation defined on this schema.

Definition 3.6. *Let r be a relation over R and $X \rightarrow Y$ a FD over R. $X \rightarrow Y$ is satisfied by r, denoted by $r \models X \rightarrow Y$, if and only if $\forall t_1, t_2 \in r$ if $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$.*

In other words if two tuples are equal on the set of attributes X then they must also be equal on the attributes set Y.

Example 3.6. *We have: $cust \models f_1$, $cust \models f_2$ but $cust \not\models f_3$. Indeed the tuples t_4 and t_5 are equals on CC, ZIP but are different on STR. Formally $t_4[CC, ZIP] = t_5[CC, ZIP]$ but $t_4[STR] \neq t_5[STR]$.*

FD obey Armstrong axioms (or rules) [Armstrong, 1974]:

- **Reflexivity**: if $Y \subseteq X \subseteq schema(R)$, then $F \vdash X \rightarrow Y$
- **Augmentation**: if $F \vdash X \rightarrow Y$ and $W \subseteq schema(R)$, then $F \vdash XW \rightarrow YW$
- **Transitivity**: if $F \vdash X \rightarrow Y$ and if $F \vdash Y \rightarrow Z$, then $F \vdash X \rightarrow Z$

Theorem 3.1. *[Armstrong, 1974] Armstrong axioms are sound and complete.*

Closure operator for FD:

Definition 3.7. *Let $X \subseteq schema(R)$ be a set of attributes over R, and let F be a set of FD over R. The closure of X with respect to F, denoted by X_F^+, is defined as follows:*

$$X_F^+ = \{A \in schema(R) \mid F \models X \rightarrow A\}$$

Property 3.1. *$F \models X \rightarrow Y$ if and only if $Y \subseteq X_F^+$.*

This property helps to check the implication of a dependency by a set using the closure operator. In fact to prove the sentence $F \models X \rightarrow Y$, it is enough to prove that $Y \subseteq X_F^+$.

3.2 Conditional Functional Dependencies

CFD have been recently introduced by [Bohannon et al., 2007] in the context of data cleaning as a compromise to bridge the gap between FD and AR. They can be seen as an unification of these two families since they allow to mix attributes and attribute/values in dependencies.

3.2.1 Definition and Semantics

Consider a relation schema R, the syntax of a CFD is given as follows: a CFD ρ on R is a pair $(X \rightarrow Y, T_p)$ where:

1. $XY \subseteq schema(R)$,
2. $X \rightarrow Y$ a standard FD,
3. and T_p is a *pattern tableau* with attributes in R.

For each attribute $A \in schema(R)$ and for each pattern tuple $t_p \in T_p$, $t_p[A]$ is either

- a constant in $DOM(A)$
- or an *'unnamed variable'* denoted by '_' referring to any value in $DOM(A)$
- or an empty variable denoted by '$*$' which indicates that the corresponding attribute does not contribute to the pattern (i.e. $A \notin XY$)

We recall the relation *cust* from Figure 3.2.

cust	CC	AC	phn	type	FN	LN	STR	CT	ZIP	item
t_1 :	01	908	1111111	1	Mike	Brant	Tree Ave.	NYC	01974	CD
t_2 :	01	908	1111111	1	Rick	James	Tree Ave.	NYC	01974	DVD
t_3 :	01	212	2222222	2	Joe	Dalton	Elm Str.	NYC	01202	CD
t_4 :	01	212	2222222	2	Jim	Cook	Elm Str.	NYC	01202	Book
t_5 :	01	215	3333333	2	Ben	Wade	Oak Av.	PHI	01202	CD
t_6 :	44	131	4444444	1	Ian	Duc	High St.	EDI	03560	DVD
t_7 :	44	140	5555555	2	Kim	Lee	High St.	PHI	03560	DVD

The semantics of a CFD extends the semantics of FD mainly with the notion of matching tuples. Let r be a relation over R, $X \subseteq R$ and T_p a pattern tableau over R. A tuple $t \in r$ *matches* a tuple $t_p \in T_p$ over X, denoted by $t[X] \asymp t_p[X]$, iff for each attribute $A \in X$, either

- $t[A] = t_p[A]$
- or $t_p[A] =$'_'
- or $t_p[A] =$'$*$'

Example 3.7. *Let us consider the three FD previously described:*

- $f_1 = CC, AC, phn \rightarrow STR, CT, ZIP$
- $f_2 = CC, AC \rightarrow CT, ZIP$
- $f_3 = CC, ZIP \rightarrow STR$

CFD are constraints that hold on a subset of tuples rather than on the entire relation. So the basic idea of CFD is to define through a selection formula a subset of a relation on which some FD hold.

For instance the FD $f_3 : CC, ZIP \rightarrow STR$ holds on the set of tuples t_6 and t_7 on cust, selected by the formula $\sigma_{CC=44}(cust)$. Technically this constraint is denoted by the CFD $\phi_0 = (CC, ZIP \rightarrow STR, (44, \_ \parallel \_))$, where the symbol '$\parallel$' is used to separate the left-hand side from the right-hand side of the dependency. This CFD states that in the United Kingdom $(CC = 44)$, the ZIP code determines the street where the customer lives. Which indeed is not available for example in the United States ($CC = 01$) or in France. CFD that hold on cust also include the following (and more):

- $\phi_1 : (CC, AC, phn \rightarrow STR, CT, ZIP(01, 908, \_ \parallel \_, NYC, \_))$
- $\phi_2 : (CC, AC, phn \rightarrow STR, CT, ZIP(01, 212, \_ \parallel \_, PHI, \_))$
- $\phi_3 : (CC, AC \rightarrow CT(01, 215 \parallel PHI))$

One can consider different selection formulas like inferior or superior operator for the embedded condition. For example the CFD $\phi_4 = (CC, ZIP \rightarrow STR, (\_, > 01974 \parallel \_))$ can be defined on the relation *cust* stating that the FD holds on the condition that ZIP must be superior to the value 01974. It may be interesting to enlarge the scope of condition formulas, this interest is out of the scope of our study. In this thesis we only consider selection formula made by equality.

Definition 3.8. *Let r be a relation over R and $\rho = (X \rightarrow Y, T)$ a CFD with $XY \subseteq R$. We say that r satisfies ρ, denoted by $r \models \rho$, iff for all $t_i, t_j \in r$ and for all $t_p \in T$, if $t_i[X] = t_j[X] \asymp t_p[X]$ then $t_i[Y] = t_j[Y] \asymp t_p[Y]$.*

Example 3.8. *The relation cust of the Figure 3.2 satisfies CFD ϕ_0, ϕ_1 and ϕ_3.*

Definition 3.9. *Let Σ_1 and Σ_2 be two sets of CFD defined over the same schema R. We say that Σ_1 is equivalent to Σ_2 denoted by $\Sigma_1 \equiv \Sigma_2$ iff for any relation r over R, $r \models \Sigma_1$ iff $r \models \Sigma_2$.*

In fact, an FD $X \rightarrow Y$ is a special case of CFD $(X \rightarrow Y, t_p)$ where t_p is a single pattern tuple and for each $B \in XY$, $t_p[B] =$ '_ '.

Example 3.9. *The FD f_1 of Example 3.7 can be expressed as the CFD $(CC, AC, PN \rightarrow STR, CT, ZIP(\_, \_, \_ \parallel \_, \_, \_))$. Similarly for f_2.*

Compare to FD, note that a single tuple relation may violate a CFD. It may occur when the pattern tableau has at least one row with at least one constant on the right-hand side. Given a relation, the satisfaction of a CFD has to be checked with both every single tuple and every couple of tuples. As for classical FD, the non-satisfaction is easier to verify: it is enough to exhibit

a counter-example, i.e. either a single tuple or a couple of tuples.
More formally, we have the relation r which violates a CFD $\rho = (X \rightarrow Y, T)$, denoted by $r \nvDash \rho$, iff

- there exists a tuple $t \in r$ and a pattern tuple $t_p \in T$ such that $t[X] \asymp t_p[X]$ and $t[Y] \not\asymp t_p[Y]$
- or there exists $t_i, t_j \in r$ and a pattern tuple $t_p \in T$ such that $t_i[X] = t_j[X] \asymp t_p[X]$ and $t_i[Y] \neq t_j[Y]$

The first condition is new with respect to FD violation. As we will see later, this condition turns out to be "embedded" into the second one.

Example 3.10. *In Figure 3.2 the relation cust does not satisfy these two CFD:*

- $\phi_2 : (CC, AC, PN \rightarrow STR, CT, ZIP(01, 212, \_ \parallel \_, PHI, \_))$.
  Indeed, t_3 violates ϕ_2 : $t_3\,[CC, AC, PN] \asymp (01, 212, \_)$ but $t_3\,[STR, CT, ZIP] \not\asymp (\_, PHI, \_)$.
- $\phi_4 = (CC, CT \rightarrow ZIP(01, \_ \parallel \_))$. *Indeed, t_2 and t_3 violate ϕ_4 since $t_2\,[CC, CT] = t_3\,[CC, CT] \asymp (01, \_)$, but $t_2\,[ZIP] \neq t_3\,[ZIP]$.*

Definition 3.10. *[Fan et al., 2008a] A CFD $(X \rightarrow Y, T_p)$ is in the normal form , when $|Y| = 1$ and $|T_p| = 1$.*

A normalized CFD has then a single attribute on the right-hand side and its pattern tableau has only one single tuple.

Proposition 3.1. *[Fan et al., 2008a] For any set Σ of CFD there exists a set Σ_{nf} of CFD such that each CFD $\rho \in \Sigma_{nf}$ is in the normal form and $\Sigma \equiv \Sigma_{nf}$.*

In the sequel we consider CFD in their normal form, unless stated otherwise. A CFD $(X \rightarrow A, t_p)$ is called:

- a *constant CFD* if $t_p\,[XA]$ consists of constants only, i.e. $t_p\,[A]$ is a constant and $t_p\,[B]$ is also a constant for all $B \in X$.
- a *variable CFD* if the right hand side of its pattern tuple is the unnamed variable '_', i.e. $t_p[A] =$ '_', the left-hand side involving either constants or '_'.

Proposition 3.2. *[Fan et al., 2008a] For any set Σ of CFD over a schema R, there exists a set Σ_c of constant CFD and a set Σ_v of variable CFD over R such that $\Sigma \equiv \Sigma_c \cup \Sigma_v$.*

There always exists a relation that satisfies constraints expressed using FD, without worrying about their consistency. This is no more the case for CFD.

CFD1: if $A \in X$ then $(X \to A, t_p)$.
CFD2: if $(X \to A, t_p)$, $B \in schema(R)$ then $(XB \to A, t'_p)$ where $t'_p[B] = \_$ and $t'_p[C] = t_p[C]\ \forall C \in X \cup \{A\}$.
CFD3: if $(X \to A, t)$, $(A \to B, t_p)$ and $t[A] \asymp t_p[A]$ then $(X \to B, t'_p)$, where $t'_p[X] = t[X]$ and $t'_p[B] = t_p[B]$
CFD4: if $(BX \to A, t_p), t_p[B] = \_$ and $t_p[A] = cte$, then $(X \to A, t'_p)$ where $t'_p[C] = t_p[C]\ \forall C \in X \cup \{A\}$
CFD5: if $(BX \to A, t_p), t_p[B] = \_$ then $(BX \to A, t'_p)$, where $t'_p[C] = t_p[C]\ \forall C \in X \cup \{A\} - \{B\}$, and $t'_p[B] = b$
CFD6:if $(X \to A, t_p)$ and $t_p[A] = cte$ then $(X \to A, t'_p)$ where $t'_p[A] = \_$ and $t'_p[X] = t_p[X]$

Figure 3.3: Inference system for CFD [Bohannon et al., 2007]

3.2.2 Consistency

We begin with an example to illustrate the consistency problem when dealing with CFD.

Example 3.11. *Let $\Sigma = \{\phi_1, \phi_2\}$ be a set of CFD with $\phi_1 = A \to B(\_ \parallel b)$ and $\phi_2 = A \to B(\_ \parallel c)$, $b \neq c$. There exist no relation r over AB that satisfies Σ because $\forall t \in r$ for the same value of $t[A]$, $t[B]$ can not be equal to b and c at the same time.*

This particular case is often observed for attributes with finite domain like Boolean. The latter increases the complexity of the consistency issue. In fact, studying the consistency problem for a set Σ of CFD defined over a schema R is equivalent to check if there exists a relation r over R such that $r \models \Sigma$. The consistency problem is NP-complete [Fan et al., 2008a]. In the sequel we do not detail the problem and we assume the consistency of CFD.

3.2.3 The implication problem

Given a set of CFD and a single CFD, we want to know if the satisfaction of the set of constraints is enough to satisfy the single one. The Armstrong rules [Armstrong, 1974] are fundamental for the implication problem of FD. In Figure 3.3, [Bohannon et al., 2007] proposes a sound and complete inference system similar to the Armstrong inference system for FD to solve the implication problem of CFD.

The following example illustrates the use of the inference system.

Example 3.12. *Let $\phi_1 = (A \to B, (-, b))$, $\phi_2 = (B \to C, (-, c))$ and $\phi_3 = (A \to C, (a, -))$ be CFD. We want to check whether the set $\{\phi_1, \phi_2\}$ implies ϕ_3 ?*

1. *$(A \to B, (-, b))$ by ϕ_1*
2. *$(B \to C, (-, c))$ by ϕ_2*

3. *$(A \to C, (-, c))$ by 1, 2 and the rule CFD3*
4. *$(A \to C, (a, c))$ by 3 and the rule CFD5*
5. *$(A \to C, (a, -))$ by 4 and the rule CFD6*

3.2.4 Minimal cover

Definition 3.11. *[Bohannon et al., 2007] A minimal cover Σ_{cm} of a set Σ of CFD is a set of CFD such that:*

1. *Each CFD in Σ_{cm} is of the form $(X \to A, t_p)$. In other words the CFD are in normal form.*
2. *The set Σ_{cm} is equivalent to Σ. Thus exactness is guaranteed.*
3. *No proper subset of Σ_{cm} implies Σ_{cm}.*
4. *For each $(X \to A, t_p)$ in Σ_{cm}, there exists no CFD $(X' \to A, t_p[X' \cup A])$ in Σ_{cm} such that $X \subset X'$. Thus there is no redundancy.*

The minimal cover Σ_{cm} is equivalent to Σ but does not contain redundancies. The minimal cover is computed by first removing redundant attributes, then redundant CFD as illustrated in Algorithm 1.

Algorithm 1 Minimal cover for CFD

```
Require: a set of CFD $\Sigma$
Ensure: $\Sigma_{cm}$: a minimal cover for $\Sigma$
1:  if $\Sigma$ is not consistent then
2:      $\Sigma_{cm} := \emptyset$
3:  end if
4:  for all $\rho = (X \to A, t_p) \in \Sigma$ do
5:      for all attribute $B \in X$ do
6:          if $\Sigma \models (X - B \to A, (t_p[X - B], t_p[A]))$ then
7:              $\Sigma := \Sigma - \{\rho\} \cup \{(X - B \to A, (t_p[X - B], t_p[A]))\}$
8:          end if
9:      end for
10: end for
11: $\Sigma_{cm} := \Sigma$
12: for all $\rho = (X \to A, t_p) \in \Sigma$ do
13:     if $\Sigma - \{\rho\} \models \rho$ then
14:         $\Sigma_{cm} := \Sigma_{cm} - \{\rho\}$
15:     end if
16: end for
```

FD or CFD are static rules. Their satisfaction is checked in the current state of a single database. There are other classes of rules which are dynamic in the

r	FN	LN	AC	phn	type	str	city	zip	item
t_1	Bob	Brady	020	079172485	2	501 Elm St.	Edi	EH7 4AH	CD
t_2	Robert	Brady	131	6884563	1	null	Lnd	null	CD
t_3	Robert	Brady	020	6884563	1	null	null	EH7 4AH	DVD
t_4	Mary	Burn	029	9978543	1	null	Cad	null	BOOK

s	FN	LN	AC	Hphn	Mphn	str	city	zip	DOB	g
s_1	Robert	Brady	131	6884563	079172485	51 Elm Row	Edi	EH7 4AH	111155	M
s_2	Mark	Smith	020	6884563	075568485	20 Baker St.	Lnd	NW1 6XE	251267	M

u	ItemId	Item	DOF	Price
u_1	701B	Book	04/03/10	45
u_2	017A	CD	11/10/10	11
u_3	904A	DVD	25/08/10	16

Figure 3.4: Source relation r and master database $\{s, u\}$

sense that two databases need to be examined in order to apply them. ER are one of those dynamic constraints.

3.3 Editing Rules

ER are a new class of data quality rules that tells how to efficiently fix errors, i.e. which attributes exactly are wrong and what values they should take [Fan et al., 2010]. ER are boosted by the recent development of MDM [Deloitte and Oracle, 2005], [Russom, 2008], [Loshin, 2009]. The global intuition is to help fix and enrich dirty data using the corresponding values from master data. ER syntax [Fan et al., 2010] is slightly rephrased in the following definition.

Definition 3.12. ***Syntax:*** *Let R be source relation schema and S be a master relation schema. Let X (resp. Y) be a list of distinct attributes from $schema(R)$ (resp. $schema(S)$), $A \in schema(R) \setminus X, B \in schema(S)$ and $Z \subseteq schema(R)$. An ER φ over (R, S) is of the form φ :*
$((X, Y) \rightarrow (A, B), t_p[Z])$ where $t_p[Z]$ is a pattern tuple over R.

Let r and s be respectively a source and a master relation over R and S. The semantics of ER has been defined with respect to the insertion of a tuple t in r [Fan et al., 2010]. The idea is to "correct" r using values of s with respect to pattern selection applying on r.

Definition 3.13. ***Semantics:*** *An ER φ=((X,Y)→(A,B),$t_p[Z]$), with respect to a master tuple $t_m \in s$, is applied to $t \in r$ to obtain t' if: (1) $t[Z]$ matches $t_p[Z]$. (2) $t[X] = t_m[Y]$. (3) t' is obtained by the update $t[A] := t_m[B]$. That is, if t matches t_p and if $t[X]$ equals $t_m[Y]$, then $t_m[B]$ is assigned to $t[A]$.*

Example 3.13. *The relation r in Figure 3.4 represents a customer and is defined over the same schema than relation schema of cust of Figure 3.2. The schema of s and u describes the master database. Let us consider the tuples t_1 and s_1. The tuple t_1 is in the source relation r and the tuple s_1 is in the master relation s. The value $t_1[FN]$ = Bob may be corrected using the right name $s_1[FN]$ = Robert from the master relation with respect to the ER $((phn, Mphn) \rightarrow ((FN),(FN)),t_{p2}[type]=(2))$. That is, if $t[type] = 2$ (indicating mobile phone) and if there is a master tuple s with $s[Mphn]$=$t[phn]$, then $t[FN]$:=$s[FN]$.*

The Example 3.13 illustrates by the way the added value of ER with respect to CFD in a data cleaning context. In next Chapters we address respectively the discovery problem of CFD, the discovery problem of ER and the application of ER in a data cleaning process.

CHAPTER 4

DISCOVERING CONDITIONAL FUNCTIONAL DEPENDENCIES

Chapter Outline

4.1 Related Works **32**
4.2 New notations for CFD **34**
4.2.1 Search space for constant CFD 34
4.3 Discovering constant CFD using conditional agree sets **37**
4.3.1 From agree sets to conditional agree sets 37
4.3.2 Conditional agree set for the discovery problem 39
4.3.3 Implementation strategy 42
4.4 Frequent constant CFD discovery **43**
4.4.1 The Formal concepts of FUN approach 45
4.4.2 From FUN to CFUN 46
4.5 Experimentation **49**
4.6 Summary **53**

In this Chapter, we address the discovery problem of CFD which can be seen as a clarification of some simple and basic notions underlying CFD. In particular we point out how data mining techniques developed for FD and AR can be reused for the mining of constant CFD. We focus on two types of techniques. The first one extends the notion of agree sets. The second one is based on a closure operator to efficiently integrate frequency of constant CFD. The techniques have been implemented and experiments have been carried out showing both the feasibility and the scalability of our proposal.

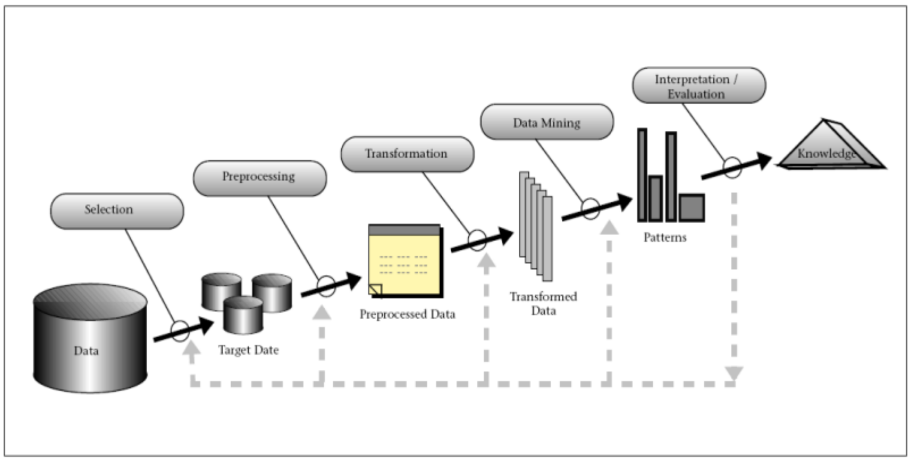


Figure 4.1: An Overview of the Steps That Compose the KDD Process [Fayyad et al., 1996]

The discovery problem of CFD is a data mining problem and more globally a Knowledge Discovery in Databases (KDD) issue. The KDD process is described in Figure 4.1. This iterative process requires the understanding of the knowledge discovery scope. Regarding the goal of the process, data are carefully selected to get target data. The latter may be transformed in order to prepare and enter the mining step in a best way. Data mining is a step in the KDD process that consists of applying data analysis and discovery algorithms that produce a particular enumeration of patterns over the data [Fayyad et al., 1996]. We particularly address this data mining step. In our context patterns mined are CFD.

The Chapter is organized as follows. Related works are discussed in Section 4.1. We introduce news notations in Section 4.2. We attack the particular class of constant CFD and present our first proposition based on the extension of agree sets [Beeri et al., 1984b] in Section 4.3. The second proposition is based on the algorithm Fun [Novelli and Cicchetti, 2001a] for mining FD. The latter technique is extended to take into account the frequency of CFD to discover frequent constant CFD in Section 4.4. The experimental results are presented in Section 4.5 and finally a summary is given.

4.1 Related Works

The idea of extending constraints with condition is not recent. For example [Maher and Srivastava, 1996] early proposed a procedure that generate tuples and constraints that have attached conditions when they exist/hold in order to solve the implication problem for constrained tuple-generating dependencies (CTGDs) [Baudinet et al., 1995].

CFD extend FD with conditions and can be seen as a particular case of CT-GDs. CFD have been recently introduced in a data cleaning purpose by [Bohannon et al., 2007]. In addition, classical problems in data dependencies such as the implication problem or the consistency question have been revisited in a theoritical setting.

[Golab et al., 2008] studied the characterization and the generation of pattern tableaux to realize the full potential of CFD. They formally define what is a good pattern tableau using the support and the confidence. An algorithm evaluated on real life data is proposed to generate optimal pattern tableau for CFD.

A common hierarchy of CFD, FD and AR along with some theoretical results on pattern tableaux equivalence have also been proposed by [Medina and Nourine, 2009]. Authors used the more global work of [Bra and Paredaens, 1983], based on horizontal decomposition of a relation, as a way to represent and reason on CFD.

But most of these works on CFD assume that CFD are already designed and provided or are manually made. In this setting the discovery problem of CFD is a relevant issue and has been addressed in the five last years.

CFD are still in current interest, so the mining problem is not widely addressed. In another hand plenty of contributions have been proposed for FD inference and for AR mining. In the context of this thesis, we can quote for example [Agrawal et al., 1993], [Pasquier et al., 1999], [Huhtala et al., 1999a], [Lopes et al., 2000], [Novelli and Cicchetti, 2001a], [Goethals et al., 2010]. As far as we know, two main contributions have been made by [Chiang and Miller, 2008] and [Fan et al., 2011] for CFD mining.

[Chiang and Miller, 2008] propose a new data driven tool for data quality management which suggests possible rules and identify conform and non-conform records. They present effective algorithms for discovering data quality rules and the portion of contextual data where the dependency holds, i.e. the context. Actually the rules discovered can be consider similar to CFD when the dependency and the context are merged and considered together. Authors also propose in the same tool, a set of dirty values in a data instance with respect to rules. But the rules discovered may contain redundant patterns which can reduce the effectiveness of the rules when used in a practical concern of data cleaning for example.

[Fan et al., 2011] propose three methods to discover CFD:

1. The first one called CFDMiner which mines only constant CFD is based on techniques for mining closed itemsets from [Pasquier et al., 1999]. CFDminer finds a canonical cover of k-frequent minimal constant CFD based on AR concepts of free and closed itemsets popularized by [Zaki, 2004].

2. The second one CTANE is developed for discovering general CFD, i.e. whenever they are constant or variable. CTANE is an extension of well

known algorithms TANE, proposed by [Huhtala et al., 1999a] for mining FD. CTANE prevents from discovering inconsistent CFD, ensure a minimal reduction of left hand sides and ensure also that the pattern is the most general possible. Authors first propose the solution for discovering CFD with a minimal support, and then generalize CTANE to deal with k-frequent minimal CFD.

3. The last one is FastCFD, which extends the algorithm FastFD proposed by [Wyss et al., 2001] for the discovery problem of FD. FastCFD is an alternative for CTANE. In contrast of CTANE, FastCFD is a depth first approach more effective. The validity of a CFD is checked on the fly and k-frequent CFD are directly addressed instead of doing a first pass with 1-frequent CFD.

4.2 New notations for CFD

We now introduce new notations which will turn out to be convenient to represent CFD in our context. Since CFD allow to mix attributes and values in the same semantics, we enrich initial CFD notation gave in section 3.2 of preliminaries Chapter in order to carry a couple "attribute-value" in the search space of CFD. We also characterize the order between CFD using elements of the new search space.

4.2.1 Search space for constant CFD

Usually, the search space for classical FD and AR is a powerset of the set of attributes (or items). With CFD, attributes have to be considered together with their possible values from their domain. This is formally defined in the following for constant CFD.

Definition 4.1. *Let R be a relation symbol. The search space of CFD over R, denoted by $SP_{CFD}(R)$, is defined as follows:*

$$SP_{CFD}(R) = \{(A, a) \mid A \in R, a \in DOM(A)\}$$

The search space is a set of couples composed by attributes of the schema and the respective possible values from their domain. The notation SP_{CFD} refers to "Search sPace of CFD". $SP_{CFD}(R)$ is infinite if at least one of the attributes of R has an infinite domain. Based on this search space, any constant CFD can be for example rewritten as follows:

Let $\rho = (A_1 \dots A_n \rightarrow A, t_p[A_1 \dots A_n A])$ be a constant CFD over R. This constant CFD ρ can be seen as syntactically equivalent to the CFD $\overline{X} \rightarrow \overline{A}$, with $\overline{X} = \{(A_1, t_p[A_1]), \dots, (A_n, t_p[A_n])\} \subseteq SP_{CFD}(R)$ and $\overline{A} = (A, t_p[A]) \in SP_{CFD}(R)$.

In the sequel, CFD will be represented using this new notation unless stated otherwise, with elements of the search space $SP_{CFD}(R)$ only. If necessary we will use the following additional notations.

Given an element $\overline{A}$=$(A,v) \in SP_{CFD}(R)$, we note $\overline{A}.att$ the attribute A and $\overline{A}.val$ the value v. By extension, a set of elements $\overline{X} \subseteq SP_{CFD}(R)$, $\overline{X}.att$ represents the union of attributes belonging to $\overline{X}$, i.e. $\overline{X}.att$=$\bigcup_{\overline{A} \in \overline{X}} \overline{A}.att$ and $\overline{X}.val$=$\bigcup_{\overline{A} \in \overline{X}} \overline{A}.v$.

The previous Definition 4.1 of the search space of CFD $SP_{CFD}(R)$ was defined just over a schema and for any class. This search space of constant CFD is now extended to take into account a relation r over R. The building process is described in Algorithm 2. The A of ASP_{CFD} refers to the word "Active" like in active domain ADOM.

Definition 4.2. *Let R be a relation symbol and r a relation over R. The search space of constant CFD for r, denoted by $ASP_{CFD}(R,r)$, is defined as:*

$$ASP_{CFD}(R,r) = \{(A,a) \mid A \in R, a \in ADOM(A,r)\}$$

Algorithm 2 The space search of constant CFD

```
Require: a relation $r$ over a schema $R$
Ensure: $ASP_{CFD}(R,r)$ the search space of constant CFD for $r$
1: $ASP_{CFD}(R,r) = \emptyset$;
2: for all tuple $t \in r$ do
3:   for all attribute $A \in R$ do
4:     if $(A,t[A]) \notin ASP_{CFD}(R,r)$ then
5:       $ASP_{CFD}(R,r) = ASP_{CFD}(R,r) \bigcup (A,t[A])$;
6:     end if
7:   end for
8: end for
9: return $ASP_{CFD}(R,r)$
```

The set $ASP_{CFD}(R,r)$ is finite since $ADOM(r)$ is finite.

Example 4.1. *Let r be a toy illustration relation over schema $R = \{A,B,C,D\}$ in Figure 4.2. For sake of clearness we denote the couple (A_i,v) by A_iv. We have: $ASP_{CFD}(ABCD,r) = \{A0,A2,B0,B1,B2,C0,C3,D1,D2\}$.*

Definition 4.3. *Let $\overline{X}$,$\overline{Y} \subseteq ASP_{CFD}(R,r)$. We say that $\overline{Y}$ generalizes $\overline{X}$ (or $\overline{X}$ specializes $\overline{Y}$), denoted by $\overline{X} \preceq \overline{Y}$, iff $\overline{Y}.att \subseteq \overline{X}.att$ and for all $\overline{A} \in \overline{Y}$, $\exists \overline{B} \in \overline{X}$ such that $\overline{B} = \overline{A}$.*

Example 4.2. *Let r be the relation in Figure 4.2. $ASPUV_{CFD}(ABCD,r) = \{A0,A2,B0,B1,B2,C0,C3,D1,D2\}$ is the search space of all CFD that hold in r. Then for example: $\{A0,B1\} \preceq \{B1\}$ and $\{A0,B1\} \npreceq \{B2\}$.*

r	A	B	C	D
t_1 :	0	1	0	2
t_2 :	0	1	3	2
t_3 :	0	0	0	1
t_4 :	2	2	0	1
t_5 :	2	1	0	1

Figure 4.2: A toy relation r over $R = \{A, B, C, D\}$

A consequence of the partial order is the characterization of a monotonic predicate.

Property 4.1. *Let r be a relation over R, $\overline{X}, \overline{Y} \subseteq ASP_{CFD}(R, r)$ such that $\overline{X} \preceq \overline{Y}$ and $\overline{A} \in ASP_{CFD}(R, r)$. We have:*
$r \models \overline{X} \to \overline{A} \Rightarrow r \models \overline{Y} \to \overline{A}$ (or equivalently $r \not\models \overline{Y} \to \overline{A} \Rightarrow r \not\models \overline{X} \to \overline{A}$)

Proof. Let us assume that $r \models \overline{X} \to \overline{A}$. Then
$\forall t_i, t_j \in r$, if $t_i[\overline{X}.att] = t_j[\overline{X}.att] = \overline{X}.val$ then $t_i[\overline{A}.att] = t_j[\overline{A}.att] = \overline{A}.val$.
In addition $\overline{X} \preceq \overline{Y}$. So $\overline{Y}.att \subseteq \overline{X}.att$ and therefore $t_i[\overline{Y}.att] = t_j[\overline{Y}.att] = \overline{Y}.val$.
Finally $r \models \overline{Y} \to \overline{A}$ □

Example 4.3. *Let r be the relation in Figure 4.2 and let $\{A0, B1\} \preceq \{B1\}$. We have $r \models (A0B1 \to D2) \Rightarrow r \models (B1 \to D2)$.*

Definition 4.4. *A constant CFD $\overline{X} \to \overline{A}$ is said to be left-reduced on r if for any $\overline{Y}.att \subset \overline{X}.att$, $r \not\models \overline{Y} \to \overline{A}$.*

Proposition 4.1. *For any relation r, there exists a set Σ_c of constant CFD such that $\Sigma_r \equiv \Sigma_c$, Σ_r being the set of satisfied CFD in r.*

According to [Fan et al., 2011] constant CFD are instant-level FD that are particularly useful in object identification, an essential issue to both data quality and data integration. The object identification is very relevant and very expressive for detecting the violation of CFD in the process of repairing data. The proposition 4.1 states that discovering the set of constant CFD satisfied by a given relation is equivalent to the discovery of satisfied CFD even if the latter contains variable CFD. Therefore, in this thesis we concentrate attention to the particular class of constant CFD for our discovery problem.

Given a relation r the problem statement is to discover a minimal cover (Section 3, Definition 3.11) of constant CFD satisfied by r.

4.3 Discovering constant CFD using conditional agree sets

In this section we show how to extend agree set with condition to obtain conditional agree set. We also study how conditional agree sets are used for the discovery problem of CFD.

4.3.1 From agree sets to conditional agree sets

Traditional agree sets were mainly developed in the context of classical FD.

Definition 4.5. *[Beeri et al., 1984b] Let t_1 and t_2 be tuples and let X be a set of attributes. We say that t_1 and t_2 agree exactly on X if $t_1[X] = t_2[X]$, and if $t_1[A] \neq t_2[A]$ for each attribute A not in X.*

This definition guarantees that tuples are similar only in the set of attributes X and nowhere else. We extend this notion to conditional agree sets as follows:

We first define the conditional agree set between a single tuple and a pattern.

Definition 4.6. *Let r be a relation over R, $t \in r$ a single tuple, t_p a pattern tuple over R. A conditional agree set between a single tuple t and a pattern t_p, denoted by $ag(t, t_p)$, is defined by:*

$$ag(t, t_p) = \{(A, t[A]) \mid t[A] \asymp t_p[A], A \in R\}$$

.

The set $ag(t, t_p)$ is the set of couples "attribute/value" that verify the projection of the attribute in the pattern tuple. Algorithm 3 computes this set given a relation r and a pattern tuple t_p.

Algorithm 3 The conditional agree sets between each tuple of a relation and a pattern

```
Require: a relation $r$ over a schema $R$, a pattern tuple $t_p$
Ensure: $result$
1: $result = \emptyset$;
2: for all tuple $t \in r$ do
3:     for all attribute $A \in R$ do
4:         if $t[A] \asymp t_p[A]$ then
5:             $result = result \bigcup (A, t[A])$;
6:         end if
7:     end for
8: end for
9: return $result$
```

Example 4.4. *Let r be the relation over $R = ABCD$ (cf. Figure 4.2).*
$t_p = (0, 1, 3, \_)$ *a pattern tuple.*
$ag(t_1, t_p) = \{A0, B1, D2\}$.
$ag(t_2, t_p) = \{A0, B1, C3, D2\}$.

Next, we introduce the conditional agree set between two tuples and a pattern.

Definition 4.7. *Let r be a relation over R, t_1, t_2 two tuples of r and t_p a pattern tuple over R. A conditional agree set between two tuples t_1, t_2 and t_p, denoted by $ag(t_1, t_2, t_p)$, is defined by: $ag(t_1, t_2, t_p) = \{(A, t_1[A]) \mid t_1[A] = t_2[A] \asymp t_p[A], A \in R\}$.*

Example 4.5. *Let r be the relation over $R = ABCD$ (cf. Figure 4.2).*
$t_p = (0, 1, 3, \_)$ *a pattern tuple.*
$ag(t_1, t_2, t_p) = \{A0, B1, D2\}$.

In practice, the main information we have from $ag(t_1, t_2, t_p)$ is a counter-example when $A = 0$, $B = 1$ and $D = 2$. For instance, $\{t_1, t_2\} \not\models A0, B1, D2 \rightarrow C0$ or $\{t_1, t_2\} \not\models A0, B1, D2 \rightarrow C3$. That is the power of $ag(t_1, t_2, t_p)$, and it will be very useful for capturing CFD violation for the discovery problem.

In the following we define the conditional agree set between a relation and a pattern. The build process of such a set is described in Algorithm 4.

Definition 4.8. *Let r be a relation over R and t_p a pattern tuple over R. The conditional agree set between r and t_p, denoted by $ag(r, t_p)$ is defined by: $ag(r, t_p) = \{ag(t_i, t_j, t_p) \mid t_i, t_j \in r, t_i \neq t_j\}$.*

Algorithm 4 The conditional agree sets between a relation and a pattern

```
Require: a relation $r$ over a schema $R$, a pattern tuple $t_p$
Ensure: $result$
1:  $result = \emptyset$;
2:  for all tuple $t_1 \in r$ do
3:    for all tuple $t_2 \in r$ do
4:      $agt = \emptyset$;
5:      for all attribute $A \in R$ do
6:        if $t_1[A] = t_2[A] \asymp t_p[A]$ then
7:          $agt = agt \bigcup (A, t_1[A])$;
8:        end if
9:        $result = result \bigcup agt$;
10:     end for
11:   end for
12: end for
13: return $result$
```

Lastly we define the *conditional agree set* with all possible pattern tuples of a given relation r denoted by $cag(r)$.

Definition 4.9. *$cag(r) = \bigcup_{t_p} ag(r, t_p)$ for all pattern tuples t_p.*

Example 4.6. *Continuing the Example 4.5, we get:*
$cag(r) = \{\{A0B1D2\}, \{A0C0\}, \{C0\}, \{B1C0\}, \{A0\}, \{B1\}, \{C0D1\}, \{A2C0D1\}\}$

4.3.2 Conditional agree set for the discovery problem

We recall the problem statement. Given a relation r over a schema R, the problem is to discover a minimal cover of constant CFD satisfied in r. More formally, we are interested in enumerating the left-hand sides denoted by lhs of all minimal CFD satisfied in r whose right-hand side is reduced to a single couple (attribute, value) that belongs to the search space of constant CFD.

Definition 4.10. *The left-hand side of left-reduced constant CFD for $\overline{A}$, denoted by $lhs(\overline{A}, r)$, is defined by:*

$$lhs(\overline{A}, r) = \{\overline{X} \subseteq ASP_{CFD}(R - \overline{A}.att, r'(\overline{A})) \mid r \models \overline{X} \to \overline{A} \text{ and } \forall \overline{Y} \subset \overline{X}, r \not\models \overline{Y} \to \overline{A}\}$$

with $r'(\overline{A}) = \pi_{R-\overline{A}.att}(\sigma_{\overline{A}.att=\overline{A}.val}(r))$ defined over $R'(= R - A)$.

In other words, the left hand side of $(\overline{A})$ in the relation r is the smallest set $\overline{X}$ of the search space such that r satisfies the constant CFD $\overline{X} \to \overline{A}$. Many works exist for the characterization of left hand side in this pattern mining context. For example, to characterize $lhs(\overline{A}, r)$, we can borrow the same principles used by [Mannila and Räihä, 1994] for inferring FD. We first define the maximal sets of attribute/values in the search space that do not satisfy the constant CFD.

Definition 4.11. *The maximal sets of not satisfied constant CFD for $\overline{A}$ in r, denoted by $max(\overline{A}, r)$, is defined by:*

$$max(\overline{A}, r) = max_{\subseteq}\{\overline{X} \subseteq ASP_{CFD}(R - \overline{A}.att, r'(\overline{A})) \mid r \not\models \overline{X} \to \overline{A}\}$$

From the monotonic Property 4.1 applied to constant CFD, i.e. the inclusion set is the partial order, maximal sets are enough to capture invalid constant CFD. So we can bridge the gap between conditional agree sets and these maximal sets. Intuitively, we need to consider elements $\overline{X}$ of the conditional agree sets $cag(r)$ such that $\overline{A}$ does not belong to $\overline{X}$. The property 4.2 characterizes this link.

Property 4.2. $max(\overline{A}, r) = max_{\subseteq}\{\overline{X} \in cag(r) \mid \overline{A} \notin \overline{X}\}$

Continuing the Example 4.6, we have the following illustration regarding maximal sets:

Example 4.7. $max(A0,r) = \{\{B1C0\},\{C0D1\}\}$
$max(A2,r) = \{\{B1C0\},\{C0D1\}\}$
$max(B0,r) = \{\{A0C0\},\{C0D1\}\}$
$max(B1,r) = \{\{A0C0\},\{A2C0D1\}\}$
$max(B2,r) = \{\{A2C0D1\}\}$
$max(C0,r) = \{\{A0B1D2\}\}$
$max(C3,r) = \{\{A0B1D2\}\}$
$max(D1,r) = \{\{A0C0\},\{B1C0\}\}$
$max(D2,r) = \{\{A0C0\},\{B1C0\}\}$

Now, from the maximal sets of elements that do not satisfy the CFD, we have to identify the minimal sets of elements that satisfy the CFD. This kind of relationship has been heavily studied in many pattern enumeration problems like FD [Mannila and Räihä, 1994] and has been early formalized. In our context, constant CFD can be seen as interesting sentence, and the task of finding them can be described as follows according to [Mannila and Toivonen, 1997]. Let us consider a relation r, a language L for expressing properties or defining subgroups of the data, and a selection predicate q are given. The predicate q is used for evaluating whether a sentence $\phi \in L$ defines a potentially interesting subclass of r. The task is to find the theory of r with respect to L and q, i.e. the set $Th(L,r,q) = \{\phi \in L \mid q(r,\phi)$ is true$\}$. Our problem is no more than a new instance of this pattern enumeration formal problem said to be representable as sets. Such formal concepts helped us to define the minimal set element of the search space that do satisfy the constant CFD from the maximal set that do not satisfy it.

So we define the complement of elements of $max(\overline{A},r)$ in $ASP_{CFD}(R - \overline{A}.att, r'(\overline{A}))$. For a given $\overline{A}$, the search space is the set of all possible couples of the form (attribute, value) from $r'(\overline{A})$.

Definition 4.12. $cmax(\overline{A},r) = \{ASP_{CFD}(R - \overline{A}.att, r'(\overline{A})) - \overline{X} \mid \overline{X} \in max(\overline{A},r)\}$.

Continuing the previous Example 4.7. Let us consider $cmax(A0,r)$. For $A0$, we have $ASP_{CFD}(R - \overline{A}.att, r'(A0))$=$\{B0,C3,D2,B1,D1\}$. So the complement of $max(A0,r)$ with respect to $ASP_{CFD}(R - \overline{A}.att, r'(A0))$ is:
$cmax(A0,r)$=$\{\{B0C3D1D2\},\{B0B1C3D2\}\}$

Similarly, for the other attribute-value couples, we have:
$cmax(A2,r) = \{\{B2D1\},\{B1B2\}\}$
$cmax(B0,r) = \{\{D1\},\{A0\}\}$
$cmax(B1,r) = \{\{A2C3D1D2\},\{A0C3D2\}\}$
$cmax(B2,r) = \{\}$
$cmax(C0,r) = \{\{A2B0B2D1\}\}$
$cmax(C3,r) = \{\}$
$cmax(D1,r) = \{\{A2B0B1B2\},\{A0A2B0B2\}\}$

$cmax(D2, r) = \{\{B1C3\}, \{A0C3\}\}$

The main result of this process can be stated. We reused the well-known connection between positive and negative borders of interesting pattern enumeration problems representable as sets described by [Mannila and Toivonen, 1997]. This connection relies on minimal transversal of a hypergraph. Let us recall some definitions borrowed from [Berge, 1987].

- A collection $\mathcal{H}$ of subsets of a finite set is a simple hypergraph if $\forall X \in \mathcal{H}, X \neq \emptyset$ and $(X, Y \in \mathcal{H}$ and $X \subseteq Y \rightarrow X = Y)$.
- A transversal T of $\mathcal{H}$ is a subset of R intersecting all the edges of $\mathcal{H}$, i.e. $T \cap E \neq \emptyset, \forall E \in \mathcal{H}$.
- A minimal transversal of $\mathcal{H}$ is a transversal T such that it does not exist a transversal T' of H, $T' \subset T$. The collection of minimal transversals of $\mathcal{H}$ is denoted by Tr($\mathcal{H}$).

The important Theorem 4.1 that follows, characterize the link between left hand sides and minimal transfersal regarding single couple attribute-value of the search space of constant CFD.

Theorem 4.1. *[Mannila and Toivonen, 1997] Let r be a relation over R and $\overline{A} \in ASP_{CFD}(R, r)$.*

$$lhs(\overline{A}, r) = TrMin(cmax(\overline{A}, r))$$

where $TrMin(H)$ is the set of minimal transversal of the hypergraph H.

Continuing the previous examples, we have the following left-hand sides:

Example 4.8.

lhs
$lhs(A0, r) = \{\{B0\}, \{C3\}, \{D2\}, \{B1D1\}\}$
$lhs(A2, r) = \{\{B2\}, \{B1D1\}\}$
$lhs(B0, r) = \{\{A0D1\}\}$
$lhs(B1, r) = \{\{C3\}, \{D2\}, \{A0A2\}, \{A0D1\}\}$
$lhs(B2, r) = \{\}$
$lhs(C0, r) = \{\{A2\}, \{B0\}, \{B2\}, \{D1\}\}$
$lhs(C3, r) = \{\}$
$lhs(D1, r) = \{\{A2\}, \{B0\}, \{B2\}, \{A0B1\}\}$
$lhs(D2, r) = \{\{C3\}, \{A0B1\}\}$

Now we can define the minimal cover of satisfied constant CFD with respect to left hand sides of the search space.

Definition 4.13. *Let r be a relation over R. The minimal cover of satisfied constant CFD in r, denoted by $\Sigma_{cm}(R,r)$, is defined by:*

$$\Sigma_{cm}(R,r) = \bigcup_{\overline{A} \in ASP_{CFD}(R,r)} lhs(\overline{A},r) \rightarrow \overline{A}$$

Continuing the previous example, we obtain the minimal cover of constant CFD satisfied in the relation r.

Example 4.9. $\Sigma_{cm}(R,r) = \{\ B0 \rightarrow A0;\ C3 \rightarrow A0;\ D2 \rightarrow A0;\ B1D1 \rightarrow A0;\ B2 \rightarrow A2;\ B1D1 \rightarrow A2;\ A0D1 \rightarrow B0;\ C3 \rightarrow B1;\ D2 \rightarrow B1;\ A0A2 \rightarrow B1;\ A0D1 \rightarrow B1;\ A2 \rightarrow C0;\ B0 \rightarrow C0;\ B2 \rightarrow C0;\ D1 \rightarrow C0;\ A2 \rightarrow D1;\ B0 \rightarrow D1;\ B2 \rightarrow D1;\ A0B1 \rightarrow D1;\ C3 \rightarrow D2;\ A0B1 \rightarrow D2\ \}$.

The element $D2 \rightarrow A0$ of the minimal cover $\Sigma_{cm}(R,r)$ carries for example the constant CFD $((D \rightarrow A),(2,0))$ that is satisfied by the relation r. By the way we note that the CFD $A0A2 \rightarrow B1$ does not make sense since the attribute A can only be involved with one value in the same dependency. This can be avoided and have to be pruned in a post-processing stage. Such kind of CFD can be seen as a side effect of the transversal minimal computation.

4.3.3 Implementation strategy

Regarding to implementation, as we have shown, inferring constant CFD using conditional agree sets is an instance of the class of interesting pattern enumeration problem formalized by [Mannila and Toivonen, 1997]. Therefore, existing implementations can be used to get constant CFD in this context. For instance, we use the `iZi` library [Flouvat et al., 2009], an adjustable tool for pattern mining problem representable as set, by considering the following:

- the set of sentences $\{\overline{X} \mid \overline{X} \subseteq ASP_{CFD}(R,r)\}$ involved in the constant CFD $\overline{X} \rightarrow \overline{A}$.
- the selection predicate q such that $q(r,\overline{X})$ is true if and only if $\overline{X} \rightarrow \overline{A}$ holds in the relation r.
- for $\overline{Y} \subseteq \overline{X}$, if $q(r,\overline{X})$ is true then $q(r,\overline{Y})$ is also true.

Concretely, given a relation r on R, we first compute the search space of constant CFD $ASP_{CFD}(R,r)$. Then with `iZi` library, we initialize the algorithm with elements of $ASP_{CFD}(R,r)$ having only one attribute and one value. Then during the execution of the algorithm, the predicate *being a satisfied constant CFD* is used to test each pattern against the data. This strategy is repeated for each element $\overline{A} \in ASP_{CFD}(R,r)$ to get all left-hand sides of $\overline{A}$ which compose constant CFD as described in the previous example 4.9. Experimentation are addressed in section 4.5.

Nevertheless, let us consider the CFD $A0, D1 \rightarrow B1$. There is no tuple that matches $(0, 1, \_, 1)$ in the relation of Figure 4.2. This highlights that the mining method based on conditional agree sets proposed in this section may produce "useless" CFD, i.e. CFD that do not match any tuple of r. This could be addressed by taking into account the `frequency` of the CFD, i.e. the number of tuples that match a given CFD. As a post-treatment, it requires a full scan of the database. But this kind of "useless" discovered constant CFD carries some information, i.e. no proper subset of $\{A0D1\}$ implies $B1$.

It also turns out that the notion of frequency or support which capture the strength of a dependency cannot be taken into account easily using conditional agree sets. The dualisation of the minimal transversal does not allow to take into account frequency during its computation. Therefore a better approach is necessary to be able to integrate the frequency of the constraint and address the discovery of **frequent** constant CFD. We overcome this limitation by integrating the support (or frequency) from the beginning, i.e. in the problem definition.

4.4 Frequent constant CFD discovery

In this section we address the discovery problem of frequent constant CFD in a given relation. Intuitively, the frequency of a CFD in a relation is the number of tuples that matches its pattern, i.e. the size of the corresponding selection query when we consider using a selection query. We reuse on efficient technique used for the discovery of frequent FD. Precisely we adapt the principles of the Fun approach [Novelli and Cicchetti, 2001a], [Novelli and Cicchetti, 2001b] for FD inference. We revisit the notion of free sets of FD, also called generators sets and quasi-closure in [Pasquier et al., 1999]. The algorithm Fun provides level-wise techniques well suited to take into account the frequency. Since we attack the problem of discovering frequent constant CFD, we first define the notion of frequency.

Definition 4.14. *Let $\theta = (\overline{X} \rightarrow \overline{Y})$ be a constant CFD over R and r a relation over R. The frequency of θ in r, denoted by $freq(\theta, r)$, is defined as follows:*

$$freq(\theta, r) = |\sigma_{\wedge_{(A,v)\in\overline{X}\cup\overline{Y}}(A=v)}(r)|$$

Let ϵ be an integer threshold value. A CFD θ is said to be frequent in r, if $freq(\theta, r) \geq \epsilon$.

The important property carried by the notion of frequency for the discovery problem is a monotonic predicate as for the support of classical AR mining problem.

Property 4.3. *Let r be a relation over R and $\overline{X}, \overline{Y} \subseteq ASP_{CFD}(R, r)$ such that $\overline{X} \subseteq \overline{Y}$ and ϵ a threshold.*

We have:
$freq(\overline{Y},r) \geq \epsilon \Rightarrow freq(\overline{X},r) \geq \epsilon$ *(or* $freq(\overline{X},r) < \epsilon \Rightarrow freq(\overline{Y},r) < \epsilon$*)*

Definition 4.15. *[Fan et al., 2011] A canonical cover of CFD on a relation r consists of non-redundant frequent CFD on r, from which all frequent CFD that hold on r can be infered.*

The previous problem statement of discovering minimal cover of satisfied CFD using conditional agree sets did not take into account the frequency. The problem is now stated when frequency is consedered. Given a relation r, our problem statement is to discover a conical cover of constant CFD on r.

We are particularly interested in defining a test to check whether or not a given CFD holds in a relation, using projection on attributes embeded in the dependency. Such a property exists for testing the satisfaction of an FD in a relation. The following property is used for FD: $r \vDash X \rightarrow Y$ iff $|\pi_X(r)| = |\pi_{XY}(r)|$. Regarding the search space of constant CFD previously described, we can state the following property.

Property 4.4. *Let R is a relation symbol, r is a relation over R,* $\overline{X}, \overline{Y} \subseteq ASP_{CFD}(R,r)$ *and* $C_{\overline{X}}, C_{\overline{Y}}$ *are two selection formulas over* $\overline{X}$ *and* $\overline{Y}$ *respectively.*

$r \vDash \overline{X} \rightarrow \overline{Y}$ *iff* $|\sigma_{C_{\overline{X}}}(r)| = |\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)|$ *where* $C_{\overline{X}} = \wedge_{(A,v) \in \overline{X}}(A = v)$ *and* $C_{\overline{Y}} = \wedge_{(A,v) \in \overline{Y}}(A = v)$.

Proof. $\sigma_{C_{\overline{X}}}(r)$ and $\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)$ respectively represent the equivalence classes of $\overline{X}.att$ and $\{\overline{X}.att \cup \overline{Y}.att\}$ according to r. They verify conditions $C_{\overline{X}}$ and $C_{\overline{X}} \wedge C_{\overline{Y}}$ in r. $|\sigma_{C_{\overline{X}}}(r)|$ and $|\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)|$ denote the cardinality of each equivelance classes.

- $r \vDash \overline{X} \rightarrow \overline{Y} \Rightarrow |\sigma_{C_{\overline{X}}}(r)| = |\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)|$:
  We suppose that $r \vDash \overline{X} \rightarrow \overline{Y} \not\Rightarrow |\sigma_{C_{\overline{X}}}(r)| = |\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)|$. There are two cases:

  1. $|\sigma_{C_{\overline{X}}}(r)| > |\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)|$. There exists $t \in \sigma_{C_{\overline{X}}}(r)$ such that $t \notin \sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)$, however $r \vDash \overline{X} \rightarrow \overline{Y}$, this is contradictious.
  2. $|\sigma_{C_{\overline{X}}}(r)| < |\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)|$. This means that there exists at least one tuple t that satisfies $C_{\overline{X}} \wedge C_{\overline{Y}}$ and does not satisfy $C_{\overline{X}}$. This is obviously impossible because all conditions are conjunctions

  Thus, $r \vDash \overline{X} \rightarrow \overline{Y} \Rightarrow |\sigma_{C_{\overline{X}}}(r)| = |\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)|$.

- $|\sigma_{C_{\overline{X}}}(r)| = |\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)| \Rightarrow r \vDash \overline{X} \rightarrow \overline{Y}$:

  1. $|\sigma_{C_{\overline{X}}}(r)| < |\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)|$: there exists at least one tuple in r that satisfies $C_{\overline{X}} \wedge C_{\overline{Y}}$ and does not satisfy $C_{\overline{X}}$. Impossible.

2. $|\sigma_{C_{\overline{X}}}(r)| > |\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)|$: there exists at least one tuple t in r that satisfies $C_{\overline{X}}$ and does not satisfies $C_{\overline{X}} \wedge C_{\overline{Y}}$. Thence, t does not satisfy $C_{\overline{Y}}$ and $r \nvDash \overline{X} \to \overline{Y}$.
3. $|\sigma_{C_{\overline{X}}}(r)| = |\sigma_{C_{\overline{X}} \wedge C_{\overline{Y}}}(r)|$: this assertion means that it does not exist any tuples in r that satisfy $C_{\overline{X}}$ and do not satisfy $C_{\overline{X}} \wedge C_{\overline{Y}}$. Thus, $r \vDash \overline{X} \to \overline{Y}$.

□

4.4.1 The Formal concepts of FUN approach

Before the presentation of the adaptation, called CFUN, of the FUN approach [Novelli and Cicchetti, 2000], [Novelli and Cicchetti, 2001a], [Novelli and Cicchetti, 2001b], we recall some notations and definitions provided by FUN.
The approach called FUN proposes a characterization of minimal FD which provides a simpler formal framework. The approach is based on the following concepts: the concept of free set for capturing source of FD and those of closure and quasi-closure of attribute sets from which targets of such dependencies can be captured.

Definition 4.16. *Free Set [Novelli and Cicchetti, 2001a]*
Let $X \subseteq R$ be a set of attributes. X is a free set in r if and only if: $\nexists X' \subset X, |X'|_r = |X|_r$.

The set of all free sets in r is denoted by $\mathcal{FS}_r$. Any combination of attributes not included in $\mathcal{FS}_r$ is called a non free set.

Lemme 4.1. *$\forall X \subseteq R, \forall X' \subset X, |X'|_r = |X|_r \Leftrightarrow X' \to X$.*

The following properties help to characterize free sets.

- Any subset of a free set is a free set itself: $\forall X \in \mathcal{FS}_r, \forall X' \subset X, X' \in \mathcal{FS}_r$.
- Any superset of a non free set is non free: $\forall X \notin \mathcal{FS}_r, \forall Y$ such that $X \subset Y, Y \notin \mathcal{FS}_r$.

The definition of attribute set closure in a given relation is recalled below.

Definition 4.17. *Attribute set closure in a relation*
Let X be a set of attributes, $X \subseteq R$. Its closure in r is defined as follows:
$X_r^+ = X \cup \{A \in R - X \,/\, |X|_r = |X \cup A|_r\}$.

The attribute set quasi-closure in a relation is derived from attribute set closure as described in the following definition.

Definition 4.18. *Attribute set quasi-closure in a relation [Novelli and Cicchetti, 2001a]*
The quasi-closure of an attribute set X in r, denoted by $X_r^\diamond$, is: $X_r^\diamond = X \cup \bigcup_{A \in X}(X - A)^+$.

According to the monotony property of the closure [Birkhoff, 1967], [Gottlob and Libkin, 1990], [Ganter and Wille, 1999], in the context of classical FD, we have: $X \subseteq X_r^\diamond \subseteq X_r^+$. Thus Definition 4.17 of attribute set closure in a relation can be rewritten as follows.

Definition 4.19. *Attribute set closure in a relation*
Let X be a set of attributes, $X \subseteq R$. Its closure in r is defined by:
$X_r^+ = X_r^\diamond \cup \{A \in R - X_r^\diamond \,/\, |X|_r = |X \cup A|_r\}$.

These concepts are key futures of FUN approach for the discovery of classical FD [Novelli and Cicchetti, 2001a]. In the sequel the concept of closure and quasi-closure are adapted for the discovery problem of CFD. Thus the approach is extended to CFUN (Conditional FUN).

4.4.2 From FUN to CFUN

In the FD inference context, the search space is the power set of R. In the constant CFD inference problem, the search space is the power set of $SP_{CFD}(R)$ given in Definition 4.1. The notions of FUN are adapted as follows, with respect to this new search space. We first define conditional free sets known also as conditional non-redundant sets.

Definition 4.20. *Conditional non-redundant sets*
Let $\overline{X} \subseteq ASP_{CFD}(R,r)$ is a set of conditional attributes.
$\overline{X}$ is a conditional non-redundant set (or a conditional free set) in r if and only if $\nexists\ \overline{X'} \subseteq \overline{X}$ such that $|\sigma_{C_{\overline{X'}}}(r)| = |\sigma_{C_{\overline{X}}}(r)|$.

The set of all conditional non-redundant sets (free sets) in r is denoted by $\mathcal{NRS}_r$, respectively $\mathcal{CFS}_r$. Any set of conditional attributes not included in $\mathcal{NRS}_r$ is called a *conditional redundant set*. The following properties help characterize them.

- Any subset of a conditional free set is a conditional free set itself: $\forall\ \overline{X} \in \mathcal{CFS}_r$, $\forall\ \overline{X'} \subset \overline{X}$, $\overline{X'} \in \mathcal{CFS}_r$.
- Any superset of a non conditional free set is non a conditional free set: $\forall\ \overline{X} \notin \mathcal{CFS}_r$, $\forall\ \overline{Y}$ such that $\overline{X} \subset \overline{Y}$, $\overline{Y} \notin \mathcal{CFS}_r$.

The characterization of the monotonicity of CFD described in Property 4.1 can be adapted in the context of non-redundant sets.

Property 4.5. *Let r be a relation over R and $\overline{X}, \overline{Y} \subseteq ASP_{CFD}(R,r)$ such that $\overline{X} \preceq \overline{Y}$. We have:*
$\overline{Y} \in \mathcal{NRS}_r \Rightarrow \overline{X} \in \mathcal{NRS}_r$ *(or equivalently $\overline{X} \notin \mathcal{NRS}_r \Rightarrow \overline{Y} \notin \mathcal{NRS}_r$)*

Clearly, APRIORI-like algorithms [Agrawal et al., 1993], [Agrawal and Srikant, 1994], [Borgelt and Kruse, 2002] can be used to discover frequent non-redundant sets thanks to this monotonic property. From the non-redundant sets, the results given in [Novelli and Cicchetti, 2001a], [Novelli and Cicchetti, 2001b] for FD inference are extended to propose a new characterization of the canonical cover of CFD in which a frequency threshold is extended. It is based on non-redundant sets, frequency, closure and quasi-closure of CFD. The definitions follow.

Definition 4.21. *Conditional attribute set closure in a relation*
Let $\overline{X}$ be a set of conditional attributes, $\overline{X} \subseteq ASP_{CFD}(R,r)$. Its closure in r is defined as follows:

$$\overline{X}^{*}_{\Sigma_r} = \overline{X} \cup \{\overline{A}/\overline{A}.att \in R - \overline{X}.att \wedge |\sigma_{C_{\overline{X}}}(r)| = |\sigma_{C_{\overline{X}} \wedge C_{\overline{A}}}(r)|\}.$$

The concept of quasi-closure allows now to accumulate the knowledge extracted from the subsets of the considered conditional attribute set.

Definition 4.22. *Conditional attribute set quasi-closure in a relation*
The quasi-closure of a conditional attribute set $\overline{X}$ in $ASP_{CFD}(R,r)$, denoted by $\overline{X}^{\diamond}_{\Sigma_r}$, is defined by:

$$\overline{X}^{\diamond}_{\Sigma_r} = \overline{X} \cup \bigcup_{\overline{A} \in \overline{X}} (\overline{X} - \overline{A})^{*}_{\Sigma_r}$$

According to the monotony property of the closure operator, we have: $\overline{X} \subseteq \overline{X}^{\diamond}_{\Sigma_r} \subseteq \overline{X}^{*}_{\Sigma_r}$. Through the following theorem, we prove that the set of constant CFD characterized using the introduced concepts of non-redundant sets, closure and quasi-closure is the canonical cover of constant CFD for the relation r.

Theorem 4.2.

$$\Sigma_{cc_\epsilon}(R,r) = \{\overline{X} \rightarrow \overline{A} \mid \overline{X} \in \mathcal{NRS}_r, freq(\overline{X},r) \geq \epsilon \text{ and } \overline{A} \in \overline{X}^{*}_{\Sigma_r} - \overline{X}^{\diamond}_{\Sigma_r}\}$$

The theoretical framework proposed is well adapted to implement a level-wise approach for discovering CFD from a relation. The CFUN algorithm is based on the concepts of APRIORI to find all conditional non-redundant sets. Once the conditional non-redundant sets discovered for each level and the corresponding frequency (count), quasi-closure and closure, discovering CFD follows with respect to Theorem 4.2. This philosophy is the same as the one used for the FD inference FUN approach [Novelli and Cicchetti, 2001a], [Novelli and Cicchetti, 2001b]. The pruning rule is provided by the Proposition 4.5 to extract only non-redundant sets.

Each level contains a collection of quadruplets $< \overline{X}, |\overline{X}|, \overline{X}^{\diamond}_{\Sigma}, \overline{X}^{*}_{\Sigma} >$ that respectively represents the candidate, its frequency, quasi-closure, and closure as shown in the Figure 4.3. The algorithm starts by initializing the first two levels 0 and 1 then it follows by a loop through levels (line 3-9). Each loop computes the closure (line 4) of non-redundant sets left in the previous level then the quasi-closure (line 5) of candidates in the current level according to the Definition 4.21. The CFD that hold are displayed (line 6) according to the Theorem 4.2. The redundant sets are removed from the current level (line 7) with respect to the Proposition 4.5) then the next level can be generated (line 8) following the well-known APRIORI technique. The loop is over when the new level is empty. The algorithm completes by displaying the CFD discovered at the last valid level.

Algorithm 5 CFUN

```
Require: a relation $r$ over a schema $R$
Ensure: a set of constant CFD satisfied by $r$
 1: $L_0 := < \overline{\emptyset}, 1, \overline{\emptyset}, \overline{\emptyset} >$
 2: $L_1 := \{ < \overline{A}, |\overline{A}|, \overline{A}, \overline{A} > \mid \overline{A} \in ASP_{CFD}(R,r) \wedge |\overline{A}.att| = 1 \}$
 3: for all $k := 1$; $L_k \neq \emptyset$; $k := k + 1$ do
 4:     ComputeClosures( $L_{k-1}$, $L_k$ )
 5:     ComputeQuasiClosures( $L_k$, $L_{k-1}$ )
 6:     DisplayCFD( $L_{k-1}$ )
 7:     PruneRedundantSets( $L_k$, $L_{k-1}$ )
 8:     $L_{k+1}$ := GenerateCandidates( $L_k$ )
 9: end for
10: ComputeClosures( $L_{k-1}$, $L_k$ )
11: DisplayCFD( $L_{k-1}$ )
12: return
```

Example 4.10. *The process is illustrated in Figure 4.3 using the relation already described in Figure 4.2. The first column of the following table is the $\overline{X}$ candidate which can be or not a conditional redundant set. The candidate prefixed by '*' is a conditional redundant set. The second column corresponds to the cardinality of $\overline{X}$ and the two last columns represent the conditional quasi-closure and conditional closure of $\overline{X}$. On the right, the CFD discovered are displayed.*

The CFUN algorithm uses the partitions representation introduced by [Cosmadakis et al., 1986], [Spyratos, 1987] and often used for the FD inference problem [Huhtala et al., 1998], [Huhtala et al., 1999b], [Lopes et al., 2000], [Novelli and Cicchetti, 2001a], [Novelli and Cicchetti, 2001b]. Indeed, one can compute quite efficiently the frequencies and generate only valid combinations of candidates. Thus, for instance, in our context $\overline{AA}$ is invalid and will not be generated as candidate unlike the approach based on conditional agree sets.

$\overline{X}$	$\lvert\overline{X}\rvert$	$\overline{X}^{\diamond}_{\Sigma}$	$\overline{X}^{*}_{\Sigma}$	
A0	3	A0	A0	
A2	2	A2	A2 C0 D1	$A2 \rightarrow C0D1$
B1	3	B1	B1	
B0	1	B0	A0 B0 C0 D1	$B0 \rightarrow A0C0D1$
B2	1	B2	A2 B2 C0 D1	$B2 \rightarrow A2C0D1$
C0	4	C0	C0	
C3	1	C3	A0 B1 C3 D2	$C3 \rightarrow A0B1D2$
D2	2	D2	A0 B1 D2	$D2 \rightarrow A0B1$
D1	3	D1	C0 D1	$D1 \rightarrow C0$
A0 B1	2	A0 B1	A0 B1 D2	$A0B1 \rightarrow D2$
*A0 B0	1	A0 B0 C0 D1	A0 B0 C0 D1	
A2 B1	1	A2 B1 C0 D1	A2 B1 C0 D1	
*A2 B2	1	A2 B2 C0 D1	A2 B2 C0 D1	
A0 C0	2	A0 C0	A0 C0	
*A0 C3	1	A0 B1 C3 D2	AO B1 C3 D2	
*A2 C0	2	A2 C0 D1	A2 C0 D1	
*A0 D2	2	A0 B1 D2	A0 B1 D2	
A0 D1	1	A0 C0 D1	A0 B0 C0 D1	$A0D1 \rightarrow B0$
*A2 D1	2	A2 C0 D1	A2 C0 D1	
...	...	...	...	
*A0 B1 C0	1	A0 B1 C0 D2	A0 B1 C0 D2	
...	...	...	...	

Figure 4.3: Illustration of the proposed characterisation

Example 4.11. *In the following the use of partitions is illustrated. The toy relation in Figure 4.2 is recalled. The threshold is set to 1. The partitions following the attributes A and C are* $\pi_A = \{(1,2,3),(4,5)\}$ *and* $\pi_C = \{(1,3,4,5),(2)\}$. *The values corresponding to equivalence classes are* 0, 2 *for* A *and* 0, 3 *for* C. *The product of* π_A *and* π_C *is* $\pi_{AC} = \{(1,3),(2),(4,5)\}$. *The values corresponding are* $(0,0)$, $(0,3)$ *and* $(2,0)$. *It directly provides the conditional attributes with their frequency:*
$freq(< A0 >) = 3$, $freq(< A2 >) = 2$, $freq(< C0 >) = 4$, $freq(< C3 >) = 1$, $freq(< A0, C0 >) = 2$, $freq(< A0, C3 >) = 1$, $freq(< A2, C0 >) = 2$.
Hence the CFD $A2 \rightarrow C0$ *is held since* $freq(< A2 >) = freq(< A2, C0) >) = 2$. *Moreover, no impossible combinations have been generated.*

4.5 Experimentation

The approach for discovering frequent constant CFD described in Section 4.4 and based on CFun has been implemented in C++ in order to assess performances. The source code [1] is available for free. An executable file can be generated with Visual C++ 9.0 or GNU g++ compilers. We also evaluates the approach using conditional agree set described in Section 4.3 and based on the `iZi` library [Flouvat et al., 2009]. We compare the two techniques. Exper-

[1]http://pageperso.lif.univ-mrs.fr/ noel.novelli/CFDProject

iments are run on both real life and synthetic datasets with various parameters.

Real Datasets The experiments used real datasets from the UCI machine learning repository [2]. In particular, the Winsconsin Breast Cancer (WBC) and Chess datasets. These datasets are also used by [Fan et al., 2011] in their experimentation for the discovery problem of CFD. The following table summarises the characteristics of the real life datasets.

Datasets	#Attributes	#Tuples	Size (Ko)
Wirsconsin Breast Cancer	11	699	19
Chess	7	28 056	519

Synthetic Datasets Synthetic data are also generated with a random data generator. It is a generator of uniform data for each column independently of each other. The synthetic datasets are automatically generated using the following parameters:

- the cardinality of the relation, $|r|$.
- the number of attributes, $|R|$.
- the correlation rate c between attribute values. The more it increases, the more values are similar.

Parameters Our experimentations have been performed on an Intel Pentium Centrino 2 GHz with 2 GB of main memory, running on Linux operating system. The parameters are the number of attributes, the number of tuples, the response time, the number of discovered CFD, the frequency and the data correlation rate.

The Figure 4.4 shows the behavior of CFUN approach applied on real life datasets above described when the support of frequent CFD varies. When the support is minimal, all CFD are considered. The graphs illustrate the execution time in seconds, the total number of CFD discovered and the memory usage in Mo when the minimal support varies. As expected, when the minimal support increases, the execution time and memory usage decrease with respect to less number of candidates to be mined. For a minimal support of 10, the approach discovered 250 CFD in 0.05 secondes, consuming 1 Mo of memory. Nevertheless this measure can be optimized and the scalability of the approach can be increased using various implementation optimisation techniques like pruning. In fact, the results obtained during the experiments are the worse possible in time and memory usage. These results appear to be of the same order in response time w.r.t [Fan et al., 2011] approaches. We did not

[2]http://archive.ics.uci.edu/ml

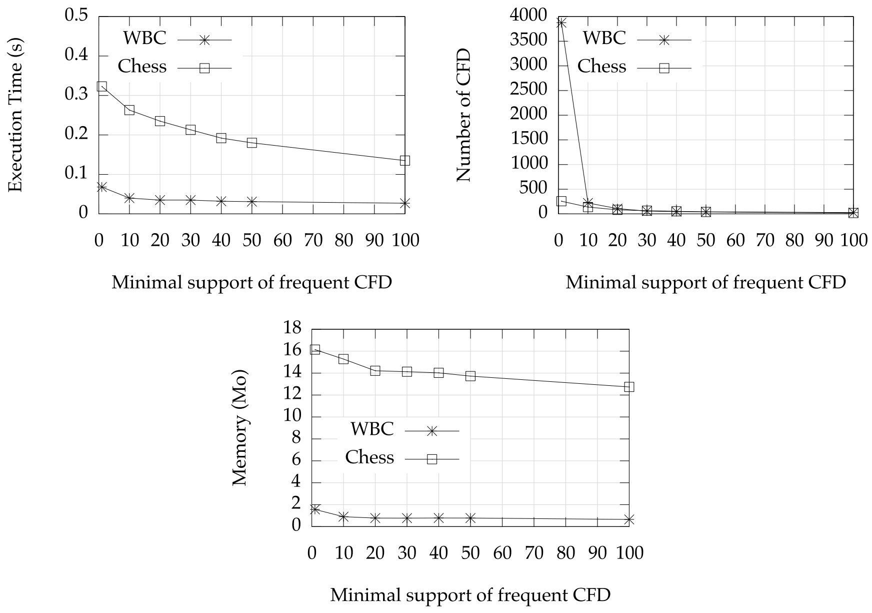


Figure 4.4: Execution time , number of CFD, and memory usage for the *Wisconcin Breast Cancer* and *Chess* real life datasets

experimented the approach based on conditional agree sets when the minimal support varies. We have not implemented the predicate being a satisfied CFD and being frequent using `iZi` library. Indeed as stated in the implementation strategy of Section 4.3, the dualization of the minimal transversal does not allow to take into account frequency during its computation.

The Figure 4.5 shows the behavior of CFun approach and conditional agree set (Cag) approach when the number of tuples goes from 5 000 to 50 000 on different generated synthetic datasets. The data correlation rate is set to 30% allowing us to fix the number of satisfied CFD indepently of the number of tuples. The support is minimal, i.e. all CFD are mined such that there is no notion of frequency, enabling us to experiment the approach based on conditional agree set (Cag) and to compare it to CFun. The memory usage and the execution time are linear according to the number of tuples for both strategy. The two approaches discover the same number of constant CFD while the number of tuples increase. This result help us to check the exactness of the two approaches. The number of CFD does not vary because the data correlation rate is fixed. The number of CFD discovered is independent of the number of tuples.

The figure 4.6 shows the behavior when the number of attributes goes from 5 to 15 on different generated synthetic datasets. The data correlation rate is fixed to 30%. Considering all constant CFD to be mined, the two ap-

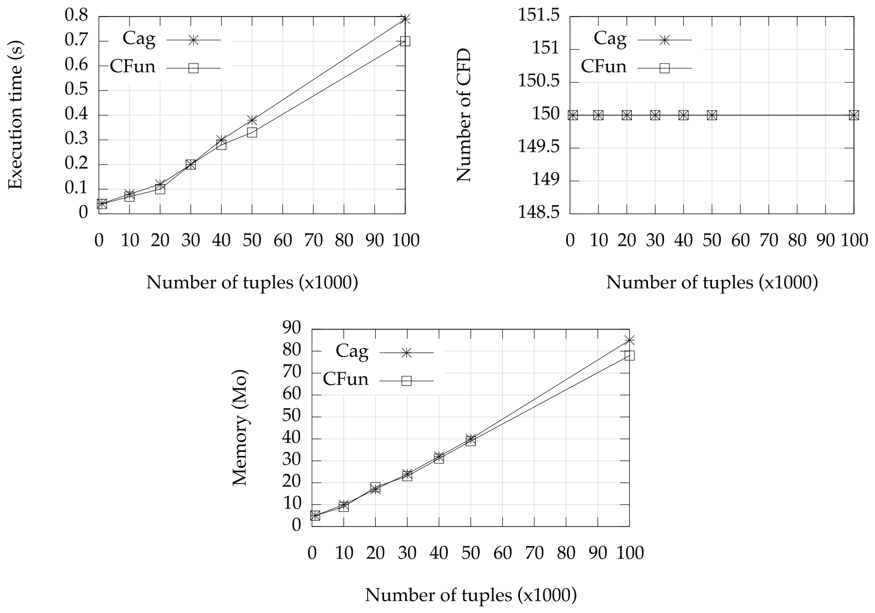


Figure 4.5: Execution time, number of CFD, and memory usage for various number of tuples

proaches does not scale very well with the arity as expected. The memory usage and the execution time are exponential according to the number of attributes, and the number of CFD discovered seems to follow the same tendancy. Indeed when the arity increases, the space search of constant CFD varies exponentially.

The Figure 4.7 shows the behavior when the data correlation rates goes from 30% to 70% on different synthetic datasets. The total number of tuples is set to 5 000 and the number of attributes is set to 7. There is no frequency, all CFD are mined. For `Cag` approach when data corelation rate dicreases the algorithm has to examine more candidates at each level which leads to a performence degradation in term of memory usage and execution time. The higher the correlation the fewer the distinct value appears in each attribute column. The lenght of the partition increases and the calculation at each stage become time and memory consuming. In contrast the performance of CFun only degrades slightly according to the data correlation rates despite the inherent exponential complexity. This behaviour of CFun is also verified when the support is considered. The main reason comes from the efficient implementation of CFun based on partitions of attribute values and generate only valid combinations of candidates. The number of discovered CFD is the same for both strategies with respect to data correlation rate.

The different experiments show the behavior of the CFun approach and

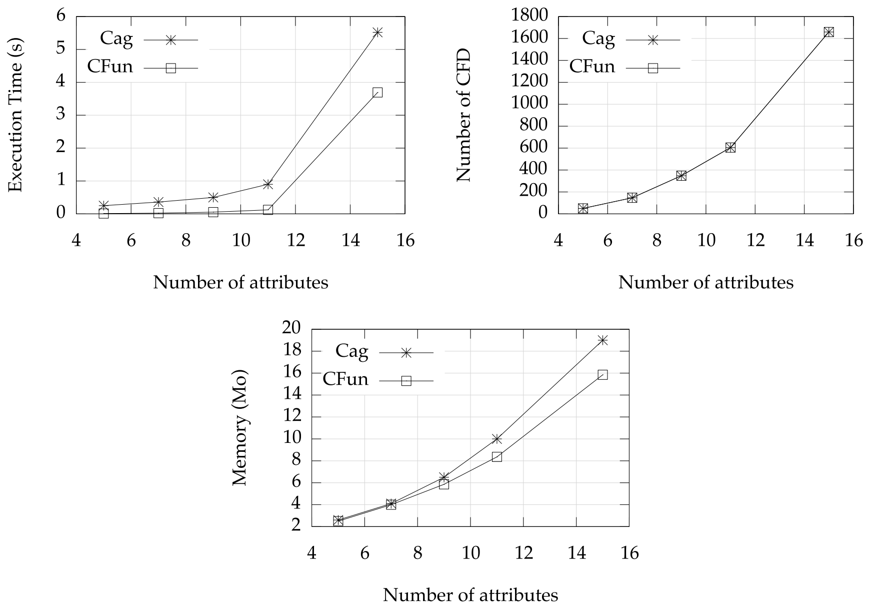


Figure 4.6: Execution time, number of CFD, and memory usage for various number of attributes

`Cag` approach for the discovery problem of constant CFD. The two approaches are exponential according to the number of attributes in time and space but are linear according to the number of tuples. When data correlation dicrease CFun outperforms the `Cag` approach.

4.6 Summary

In this Chapter, we studied the discovery of constant CFD in an existing relation. Two approaches have been proposed. The first one is conditional agree set, a proposition that extends well known agree set [Fagin and Vardi, 1983] to extract CFD. Nevertheless the proposal is suitable only for mining all possible constant CFD without any threshold. The second contribution is an extension of the Fun approach [Novelli and Cicchetti, 2001b] whose advantage is to easily deal with the frequency of CFD which was difficult to integrate using the approach based on conditional agree sets. The two approaches have been tested against different synthetic and real-life datasets showing a linear scalability in the size of the relation, but an exponential behaviour with respect to the arity. The two approaches are less time and memory consuming when

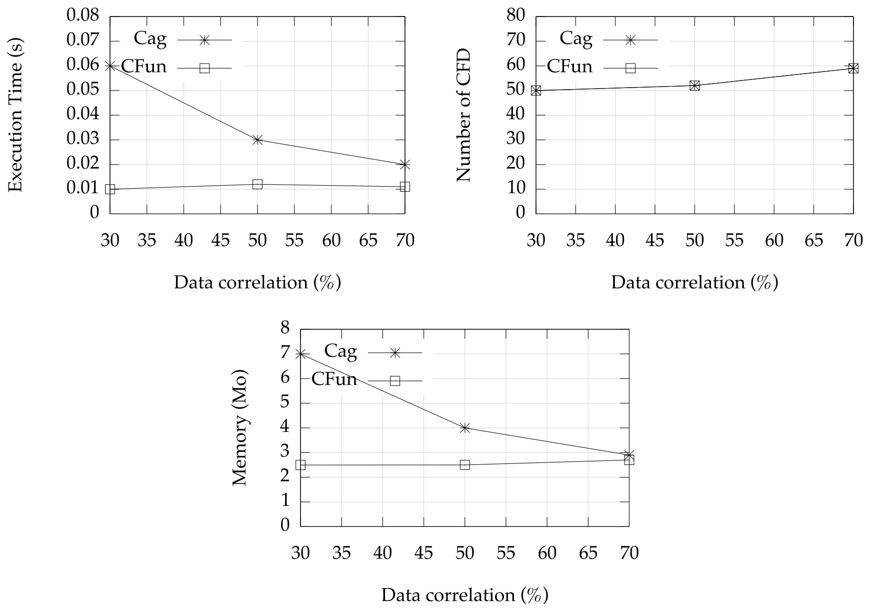


Figure 4.7: Execution time, number of CFD, and memory usage for various data correlation rates

the support of CFD increase, but CFun outperforms the approach based on conditional agree set when the correlation rate of attribute values decreases.

Chapter 5

Discovering Editing Rules

Chapter Outline

5.1 New Semantic of Editing Rules **57**
5.2 Discovering Editing Rules **59**
5.2.1 Discussion 59
5.2.2 From Source and Master Relations to IND 60
5.2.3 From Unary Approximative IND to a Mapping Function . . . 65
5.2.4 Mining Editing Rules 67
5.3 Related Works **68**
5.4 Experimentation **69**
5.4.1 Mining one-to-one correspondences 70
5.4.2 Discovering Editing Rules 73
5.5 Summary **74**

Editing Rules are a new class of data quality rules introduced thanks to the recent development of Master Data Management. In practice, designing Editing Rules is an expensive process that involves intensive manual efforts. In this Chapter, we develop automatic pattern mining techniques for discovering Editing Rules from existing source relations (possibly dirty) with respect to master relations (supposed to be clean and accurate). In this setting, we propose a new semantics of Editing Rules taking advantage of both source and master data. The problem turns out to be strongly related to the discovery of both Conditional Functional Dependencies and one-to-one correspondences from sources to master attributes. We finally evaluate our techniques on both synthetic and real-life databases.

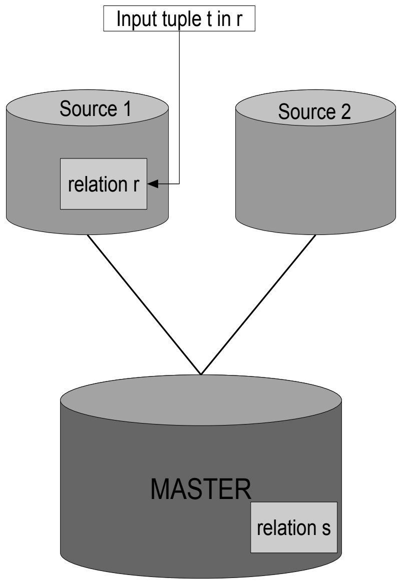


Figure 5.1: A master database with two source databases

ER help enrich and fix dirty source data using the corresponding values from clean and accurate master data. For example in Figure 5.1 the tuple *t* may be corrected with respect the master relation *s* that corresponds to the source *r*.

However, for ER to be effective in practice, it is necessary to have techniques in place that can automatically discover such rules, relaying to human manual rule designers is unrealistic, even more for huge volume of data. This practical concern highlights the need for studying the discovering problem of ER. Given a master database and a source database, it is to mine all possible ER. We assume the existence of both sources and master data. Our approach can be summarized as follows.

1. First, finding one-to-one attribute mapping between source and master relations. This first step is justified by the fact that ER are dynamic rules, i.e. defined using two relations and may also be used to correct input source tuples on the fly. It is necessary to find correspondences between attributes relation involved by the rule in order to apply the correction to the right attribute. This step is equivalent to mine a mapping function from source attributes to master attributes.

2. Inferring rules from master relations. This second and last step is related to the discovery of CFD. Actually, we mine CFD from the master relations.

3. Generating ER using the mapping function from the first step.

This Chapter is organized as follows. In section 5.1, we propose a new semantics suitable for the discovery of ER. We address in detail the process of discovering ER in section 5.2. Finally we conclude the section by giving the process of mining ER when the mapping function is known. Related works are discussed in section 5.3 before we present the experimentation in section 5.4.

5.1 New Semantic of Editing Rules

The problem we are interested in can be stated as follows: given instance r_i from a source database d and an instance s_i from a master database m, it is to find all applicable ER on r_i with respect to s_i. However, the definition 3.13 of ER in Chapter 3 needs to be extended since it just involves tuple modification. This semantics of ER based on tuple modification was introduced for a repair purpose. We need first to define a new semantics independent of any tuple modification and then focus on their mining.

First we introduce a mapping function f from attributes of source relation to attributes of master relation. This function is based on the following requirements:

1. Each attribute of a source relation should have a mapping attribute by f in the master relations, and only one. This assumption is driven by the fact that master data structure are well designed, strongly expressive and should cover the structure of source relations.

2. If two attributes A, B of a source relation have the same mapping attribute by f in a master relation, then they are the same, $A = B$. This assertion avoids having any confusion when the rule is applied for correction.

Definition 5.1. *Let $schema(R)$ be the source schema relation and $schema(S)$ be the master schema relation. We next formally describe the mapping function. A mapping function f from $schema(R)$ to $schema(S)$, is defined as a total and injective function.* $f \; : \; schema(R) \; \rightarrow \; schema(S)$

Definition 5.2. *[Levene and Loizou, 1999] A function f:X → Y is one-to-one or injective if every element of X is mapped to a unique element of Y, namely for all x_1, $x_2 \in X$ if $x_1 \neq x_2$ then $f(x_1) \neq f(x_2)$, or equivalently if $f(x_1) = f(x_2)$ then $x_1 = x_2$.*

r	FN	LN	AC	phn	type	str	city	zip	item
t_1	Bob	Brady	020	079172485	2	501 Elm St.	Edi	EH7 4AH	CD
t_2	Robert	Brady	131	6884563	1	null	Lnd	null	CD
t_3	Robert	Brady	020	6884563	1	null	null	EH7 4AH	DVD
t_4	Mary	Burn	029	9978543	1	null	Cad	null	BOOK

s	FN	LN	AC	Hphn	Mphn	str	city	zip	DOB	g
s_1	Robert	Brady	131	6884563	079172485	51 Elm Row	Edi	EH7 4AH	111155	M
s_2	Mark	Smith	020	6884563	075568485	20 Baker St.	Lnd	NW1 6XE	251267	M

Figure 5.2: Source relation r and master relation s from Figure 3.4

So only a unique attribute is considered. By extension, for a set of attribute $X \subseteq schema(R)$ we note: $f(X) = \bigcup_{A \in X}\{f(A)\}$. The new semantics of ER can be now given.

Definition 5.3. ***New semantics:*** *Let φ=((X,Y) → (A,B), $t_p[Z]$) be an ER over (R,S), r a source relation over R and s a master relation over S. We say that φ is satisfied in (r,s) with respect to f, denoted by $(r,s) \models_f \varphi$, if:*

1. *$f(X) = Y$.*
2. *$f(A) = B$.*
3. *$s \models Y \rightarrow B, t_p[Z']$ with $Z' = f(Z)$.*

Thus the semantics is based on the notion of satisfaction very popular in dependency theory, for example the satisfaction of FD or CFD presented in Chapter 3 Preliminaries.

Example 5.1. *We recall source and master relations of Figure 3.4 describing a customer to illustrate the new semantics in term of satisfaction. Let $\varphi = ((zip, zip) \rightarrow (AC, AC), t_p = (EH7\ 4AH))$ be an ER defined on r, s of Figure 5.2. This ER means the area code must be corrected with respect to the value from the master data whenever there exist an inconsistency driven by the dependency between the zip code and the area code for a subset of tuples captured by the pattern $EH7\ 4AH$. We can say that φ is satisfied in (r,s) with respect to the identity function f. Indeed, attributes zip and AC are defined in each relation and have correspondents: $f(zip) = zip$, $f(AC) = AC$ and finally the last condition of CFD satisfaction by the master relation is verified, i.e. $s \models zip \rightarrow AC, t_p = (EH7\ 4AH)$.*

We can give the problem statement.

> The problem of Discovering ER, denoted by DER, is defined as follows: Given a source relation r, a master relation s and a mapping function f, it is to find all ER satisfied in (r,s) with respect to f.

5.2 Discovering Editing Rules

In order to solve the DER problem, we first consider the case when the source relation and the master relation are defined over the same relation schema, i.e. the mapping function f is the identity function. Then we address the general case when relation schemas are different.

In fact when schemas of the source and the master relation are identical, the DER problem is almost equivalent to the discovery of CFD addressed in Chapter 4 of this thesis. The DER problem can be therefore resolved as follows:

1. Discovering a canonical cover of CFD satisfied in the master relation s.
2. For each discovered constant CFD $(X \to A, t_p[Z])$, generating the ER $((X, f(X)) \to (A, f(A)), t_p[Z])$. Since f is the identity function, the generated ER become $((X, X) \to (A, A), t_p[Z])$.

5.2.1 Discussion

When the schemas of the source relation and the master relation are different, it is necessary to find out correspondences. Different techniques may be used to specify such correspondences, among them we quote:

- User interaction: interacting with we user may be a solution to catch correspondences between attributes. Given a source attribute, the user is asked to propose a corresponding attribute from the master relation. This solution used in many tools is conflicting with our automatic process of discovering ER.
- Attribute naming assumption: in the relational model, the unique name assumption (URSA)) states that the attributes with the same name have same semantics and same meaning [Levene and Loizou, 1999], so represent the same real world entity. This assumption can be used to catch correspondences between attributes. But in practice, two real world entities can have different attribute names and URSA is rarely respected.
- INclusion Dependencies (IND): IND generalize foreign keys in the same way than FD generalize primary keys. Mining unary IND between R and S corresponds to find a correspondence between R and S.

In the sequel we focus on the solution based on IND to retrieve correspondences between attributes. Clearly, due to the presence of errors in source relations and due to the high quality data of the master relation, the definition of such a correspondence should be flexible.

r	A	B	C	D
t_1 :	0	1	0	2
t_2 :	0	1	3	2
t_3 :	0	0	0	1
t_4 :	2	2	0	1
t_5 :	2	1	0	1

The condensed representation of the toy relation r is $CR(r)$:

0	A B C
1	B D
2	A B D
3	C

Figure 5.3: Condensed Representation of the toy illustration relation r

5.2.2 From Source and Master Relations to IND

We define a mapping function f with respect to unary IND from the source relation r to the master relation s. Efficient techniques have been proposed by [Lopes et al., 2002] to discover such unary IND satisfied in a given database. Let A and B be respectively single attributes. The unary inclusion dependency $A \subseteq B$ means all values of attribute A are included in the bag of values of attribute B.

Let d be a database over schema R, where r_1, r_2 are relations over R_1 and R_2 $\in R$. An IND is satisfied in a database d over R, denoted by $d \models R_1[X] \subseteq R_2[Y]$, if $\forall t_1 \in r_1$, $\exists t_2 \in r_2$ such that $t_1[X] = t_2[Y]$. Equivalently $d \models R_1[X] \subseteq R_2[Y]$ whenever $\pi_X(r_1) \subseteq \pi_Y(r_2)$. For example in Figure 5.2, the set of relation $\{r, s\}$ satisfies the unary inclusion dependency $Hphn \subseteq phn$.

We introduce a preprocessing of relations to be as close as possible to AR syntax. The intuition is to use AR techniques through a well known condensed representation of relations. Here is an example.

Example 5.2. *The condensed representation $CR(r)$ of toy illustration relation r from Figure 4.2, is given in Figure 5.2. We can also say that the value 0 is associated to the set of attributes $\{A, B, C\}$, read from the first line.*

Definition 5.4. *Given a common relation r over a schema $schema(R)$, the condensed representation of r, denoted by $CR(r)$, is defined by:*

$$CR(r) = \{(v, X) | v \in ADOM(r), X = \bigcup\{A \in schema(R) | \exists t \in r, t[A] = v\}\}$$

For convenience concerning the condensed representation, we use the following notations:

- $CR(r).val = \{v | (v, X) \in CR(r)\}$

- $CR(r).v.atts = \{X | (v, X) \in CR(r)\}$

The following Algorithm 6 computes the condensed representation of a given relation r.

Algorithm 6 Condensed representation of relation r

```
Require: A relation $r$ over $R$.
Ensure: The condensed representation $CR(r)$ of the relation $r$
 1: $CR = \emptyset$;
 2: for all $t \in r$ do
 3:     for all $A \in R$ do
 4:         if $t[A] \in CR.val$ then
 5:             – Value already seen
 6:             if $A \notin CR.t[A].atts$ then
 7:                 $CR.t[A].atts+ = A$;
 8:             end if
 9:         else
10:             – New value
11:             $CR+ = (t[A], A)$;
12:         end if
13:     end for
14: end for
15: return $CR$.
```

From source and master relations, the goal is to discover IND using the condensed representation. We first rely on an approach based on a closure operator with respect to the condensed representation before taking into account the support of attributes inside the condensed representation.

5.2.2.1 Discovering IND from sets

Given an attribute A and an attribute B, we are interested in to checking if all values of A are included in the set of values of B, i.e. if $ADOM(A) \subseteq ADOM(B)$ using the condensed representation. This verification is done by introducing a closure operator with respect to the condensed representation.

Definition 5.5. *The* closure *of an attribute $A \in schema(R)$ with respect to $CR(r)$, denoted by $A^+_{CR(r)}$, is defined as:*

$$A^+_{CR(r)} = \bigcap_{(v,X) \in CR(r)} \{X | A \in X\}$$

Property 5.1. *Let A and B be attributes from a relation schema. Let A^+ be the closure of A. If $B \in A^+$ then $A \subseteq B$.*

Algorithm 7 outputs the closure of an attribute with respect to the condensed representation.

Algorithm 7 Closure of an attribute with respect to the condensed representation

Require: the condensed representation of r: $CR(r)$, an attribute $A \in R$.
Ensure: The closure $A^+_{CR(r)}$ of A

```
1: $A^+_{CR(r)} = R$
2: for all $(v, X) \in CR(r)$ do
3:   if $A \in X$ then
4:     $A^+_{CR(r)} = A^+_{CR(r)} \bigcap X$;
5:     if $A^+_{CR(r)} = A$ then
6:       return A
7:     end if
8:   end if
9: end for
10: return $A^+_{CR(r)}$
```

Thanks to the condensed representation it is easy to compute the closure.

Example 5.3. *For example let us compute the closure of the attribute A in Figure 5.2. We consider the condensed representation. The attribute A appears in the following sets: ABC and ABD. The intersection $ABC \bigcap ABD$ outputs the set AB. So the closure of the attribute A with respect to the condensed representation $CR(r)$ is the attribute set AB. More formally $A^+_{CR(r)} = AB$.*

In the toy illustration relation, with the property 5.1 and thanks to the condensed representation in Figure 5.2 we can state that $A \subseteq B$ because $A^+_{CR(r)} = AB$. Visually the values $\{0, 2\}$ of attribute A are included in the values $\{0, 1, 2\}$ of attribute B.

5.2.2.2 Towards discovering unary approximative IND

The discovering may miss some IND that almost hold. And it may be also interesting to take into account the support in the discovery process of unary IND.

Definition 5.6. *The* support *of an attribute set $X \subseteq R$ in the condensed representation $CR(r)$, denoted by $sup(X, CR(r))$, is defined by:*

$$sup(X, CR(r)) = |\{(v, Y) \in CR(r) | X \subseteq Y\}|$$

Example 5.4. *For example in Figure 5.2, the support of attribute A with respect to the condensed representation is 2, i.e. $sup(X, CR(r)) = 2$.*

The support can be also used to characterize the satisfaction of an IND. The closure of an attribute, the satisfaction of an IND and the equality of the support of attributes with respect to the condensed representation are equivalent when the following condition is verified:

Property 5.2. *Let r be a relation over R and A, B attributes of R. $r \models A \subseteq B \iff B \in A^+_{CR(r)} \iff sup(\{A, B\}, CR(r)) = sup(\{A\}, CR(r))$.*

Proof. $r \models R[A] \subseteq S[B]$. For all value $v \in \pi_A(r)$, there exists a tuple $t \in r$ such that $v = t[B]$. So the assertion $v \in \pi_A(r)$ is equivalent to the assertion $v \in \pi_B(r)$. Thus $\forall(v, X)$ in the condensed representation $CR(r)$, the assertion $A \in X$ is equivalent to the assertion $B \in X$. So $B \in$ to the intersection set $\bigcap_{(v,X) \in CR(r)} \{X | A \in X\}$.
Finally attribute B belongs to the closure of attribute A with respect to the condensed representation, i.e. $B \in A^+_{CR(r)}$. □

The property 5.2 avoids the heavy process of computing the intersection between attributes when calculating the closure to extract IND.

In practice, it is necessary to introduce an approximation measure to extract unary approximative IND from relation. For example, this can be done using the natural error measure g_3' introduced by [Lopes et al., 2002]. We recall their error measure definition in our context.

Definition 5.7. *[Lopes et al., 2002] Let r be a source over schema R and let s be a master relation over schema S.*

$$g_3'(R[X] \subseteq S[Y], \{r, s\}) = 1 - \frac{max\{|\pi_X(r')| : r' \subseteq r, \{r, s\} \models R[X] \subseteq S[Y]\}}{|\pi_X(r)|}$$

Example 5.5. *In Figure 5.2, we have:*

$$g_3'(R[AC] \subseteq S[AC], \{r, s\}) = 1 - (2/3) = 1/3 = 0.33$$

The intuition is to count the minimal number of tuples to remove to obtain a relation that satisfies the given IND. This error measure helped [Lopes et al., 2002] approximate IND to take into account IND that almost hold in a given database. The use of this particular error measure helps us characterize the correspondence between attributes based on unary approximative IND with respect to an epsilon threshold ϵ. We propose the following condition to obey one-to-one correspondences in our context.

Definition 5.8. *Let $A \in schema(R), B \in schema(S)$ and ϵ a $[0, 1]$-threshold value. We say that A corresponds to B in r, s with respect to ϵ, denoted by $\{r, s\} \models_\epsilon corr(A, B)$, if $g'3(R[A] \subseteq S[B], \{r, s\}) \leq \epsilon$.*

By extension, we say that X corresponds to Y with respect to ϵ if $\forall A \in X, \exists B \in Y$ such that $\{r,s\} \models_\epsilon corr(A,B)$. The natural error measure g_3' related to the correspondence can be also defined using the support of attribute sets in the condensed representation. The error defined using the support of IND is denoted *error*.

Definition 5.9. *Let r be a relation over R and let A and B be attributes from schema of R. The error of the dependency is the ratio of the support,*

$$error(A \subseteq B) = \frac{sup(\{A,B\})}{sup(\{A\})}$$

In practice, errors computed from the support or from the natural measure g_3' are equivalent as stated in the following property:

Property 5.3. *Let r be a relation over R and let A and B be attributes from schema of R.*

$$error(A \subseteq B) = g_3'(A \subseteq B)$$

.

Example 5.6. *In Figure 5.2, we have:*

- $g_3'(B \subseteq A, \{r,s\})$ = *1-(1/3) = 2/3 = 0.66*
- $error(B \subseteq A) = \frac{sup(\{A,B\})}{sup(\{B\})}$ = *2/3 = 0.66*

In fact $B \subseteq A$ does not hold. The tuples t_1, t_2 and t_5 carry for the attribute B the value 1 *that does not belong to $\pi_A(r)$. The error is then equal to 66%. This is the key intuition behind the approximation. Indeed when the tuples t_1, t_2 and t_5 are somehow removed, the IND $B \subseteq A$ become exact.*

Thanks to the Definition 5.9 and Property 5.3, we can now infer a very important result about approximating unary IND using the support of attributes involved in the dependency. More formally we have the following result.

Property 5.4. *Let r be a relation over R and let A and B be attributes from schema of R.*

$$g_3'(A \subseteq B) = \frac{sup(\{A,B\})}{sup(\{A\})}$$

.

Thus to approximate unary IND we just need to compute the support of every single attribute and the support of every couple of attributes. This issue can be addressed using algorithms designed for the discovery of AR between items in transaction databases.

Single attributes are considered as items of size 1 also referenced as F_1 and the generated candidates lead to itemsets of size 2 referenced as F_2. For example, the algorithm APRIORI [Agrawal and Srikant, 1994] for mining AR includes a step for the candidates generation. From the condensed representation of the source and master relation we can generate itemsets F_1 and F_2 thanks to the CandidateGeneration function of [Agrawal and Srikant, 1994] until level 2. Algorithm 8 describes the process.

Algorithm 8 Scalable Closure

```
Require: A condensed representation $CR$ of $r$ and $s$.
Ensure: $F_1, F_2$, itemsets of size 1 and 2 and their supports.
1: $F_1 = \{(A, support(A)) | A \in CR\}$
2: $C_2 = CandidateGeneration(F1)$ [Agrawal and Srikant, 1994]
3: for all $(v, X) \in CR(r)$ do
4:     $F_2 = subset(C_2, X)$ – Return the subset of $C_2$ containing $X$
5:     for all element $e \in F_2$ do
6:         $support(e) + = 1$
7:     end for
8: end for
9: return $F_1, F_2$
```

The set F_1 is computed using the condensed representation. The candidates C_2 are generated from the set F_1 and F_2 is deducted using C_2. The ratio of the support of attribute from F_1 and F_2 directly ensure an approximation of any unary IND build with attributes of the itemsets.

We now concentrate on inferring the mapping function from the unary IND. We remind that the mapping function is the key feature that enable us to retrieve one-to-one correspondences between attributes of the source and the master relation in the process of discovering ER.

5.2.3 From Unary Approximative IND to a Mapping Function

In previous subsection, we concentrate attention on how to discover and approximate unary IND from relations, indeed source and master. We build sets F_1 and F_2 to that end. In the sequel we describe how to finally infer the mapping function that catches the one-to-one correspondences between attributes. Given a source relation, a master relation and a ϵ threshold, the Algorithm 9 (SR2MR) outputs a mapping function between a source relation and a master relation based on approximative unary IND through F_1 and F_2 sets.

Algorithm 9 (SR2MR) Mapping from Source Relation to Master Relation

```
Require: a source relation $r$ over $R$, a master relation $s$ over $S$, a user-defined threshold $\epsilon$.
Ensure: A mapping function $f$ from $R$ to $S$
 1: $CR = Preprocessing(r,s)$;
 2: $(F_1, F_2) = scalableClosure(CR)$;
 3: for all $A \in R$ do
 4:     $((B,\epsilon), F_2) = FindBest(F_1, F_2, A)$;
 5:     while $g_3'(A \subseteq B) \leq \epsilon$ do
 6:         $\epsilon = \epsilon + 0.05$
 7:     end while
 8:     $f(A) = B$
 9: end for
10: return $f$
```

The first step of Algorithm 9 called Preprocessing(r,s) computes for both relations the condensed representation. The itemsets $F1$, $F2$ including the support of each element are generated thanks to Algorithm 8. For each attribute A in the source schema R a corresponding attribute in the source S is mined by *FindBest* procedure with respect to a target result mapping function f. The FindBest procedure is described in details in Algorithm 10. The ϵ threshold is increased until a correspondence is found.

Algorithm 10 FindBest: find the best corresponding attribute

```
Require: $F_1$, $F_2$ and $A \in R$.
Ensure: attribute $B \in S$ that is a mapping attribute for $A$, an approximation
    of the mapping and the remaining itemsets $F_2$ of size 2 with supports.
1: Let $(A, v) \in F_1$;
2: Let $E = \{(A_i A_j, v) \in F_2 | A = A_i$ or $A = A_j\}$
3: if $\exists(AB, v) \in E$ such that for all $(X, v') \in E, v \geq v'$ then
4:     Remove all occurrences of $B$ in $F_2$
5:     return $((B, \frac{v'}{v}), F_2)$
6: else
7:     return $((\perp, 1), F_2)$
8: end if
```

Given itemsets F_1, F_2 derived from the condensed representation and an attribute A from the source schema R, the procedure FindBest outputs a corresponding attribute B, for attribute A. The procedure selects, from F_2, elements that contains the attribute A and such the ratio of the support of the elements and the attribute is maximized. When many approximative unary IND are concerned, the one with the biggest support is chosen. The symbol $\perp$ refers to the default attribute.

5.2.4 Mining Editing Rules

Once it is possible to retrieve one-to-one correspondences between source attributes and master attribute, the building process of ER naturally follows using the discovered constant CFD satisfied by the master relation s. The mapping function f is first computed thanks to Algorithm 9. Then for each attribute A from the source relation (Line 4), a mapping attribute $s.B$ is defined with an error approximation err. The approximation err is the output error measure when computing the corresponding attribute from FindBest procedure of Algorithm 10. We consider all constant CFD (Line 6) from the master relation such that the right hand side of the dependency is the corresponding attribute of A and we finally generate the ER using the mapping function. Following Algorithm 11 describes the process.

Algorithm 11 Discovering ER

```
Require: $r$ a source relation, $s$ a master relation, $\Sigma$ the set of CFD satisfied in
    $s$, $\epsilon$ a user defined threshold.
Ensure: ER for $r$ with respect to $s$
 1: $res = \emptyset$;
 2: $f = SR2MR(r, s)$;
 3: for all $A \in R$ do
 4:   Let $(s.B, err) = f(A)$;
 5:   $CFD = \{cfd \in \Sigma|$ cfd that are defined over s $\}$
 6:   for all $X \to A, t_p[Z] \in CFD$ do
 7:     if $g(A \cup X \cup Z) \in R$ then
 8:       if $\forall B \in (A \cup X \cup Z)$ such that $B.err \leq \epsilon$ then
 9:         $res+ = (f^{-1}(X), X) \to f^{-1}(A), A), t_p[f^{-1}(Z)]$
10:       end if
11:     end if
12:   end for
13: end for
14: return $res$
```

Finally, in this section we detailed the different steps to solve the discovery problem of ER, i.e. given a source relation, a master relation and a mapping function between these relations, it is to find all ER that can be applied to the source to correct inconsistencies with respect to the master relation. We attack in detail the complex case when the schemas of source and master relation are different. In this non trivial case we proposed a method based on approximative unary IND to find a mapping function between attributes to explicit one-to-one correspondences. Once the correspondences are retrieved, the ER are mined based on the discovery of constant CFD.

5.3 Related Works

Since introduction of ER by [Fan et al., 2010], as far as we know, no contribution has been made for their discovery problem. Nevertheless the approach we adopted to resolve the discovery problem of ER is related to a more general problem of mapping function discovery between source and target schemas in a data integration purpose. Indeed, attributes involved in IND correspond each other with respect to a mapping function [Lopes et al., 2002]. Many contributions have been proposed in this setting.

For example, [Bauckmann et al., 2007] proposed an efficient algorithm to find all the IND satisfied by a given relation in a context of schema matching for data integration. The technique focused on unary IND. In this case all pairs of attributes must be tested to check the satisfaction of the dependency, i.e. all the dependent values must be included in the referenced value set. In this setting [Bauckmann et al., 2007] propose the Single Pass Inclusion DEpendency Recognition (SPIDER) algorithm that detect satisfied unary IND in a given database. The SPIDER algorithm first sorts attribute value sets and testes IND candidates in parallel while reading attributes values. The algorithm outputs unary IND even without having information about the schema, all the process is based on values. Even if the algorithm is powerful, it does not detect approximative unary IND.

We early mentioned the solution based on identifying similar column names to extract one-to-one correspondences. But sometimes column names are difficult to interpret. [Kang and Naughton, 2003] proposed a technique that overcomes this limitation and check matching schema in the presence of opaque column names and data values. The matching techniques are not dependent of data interpretation and are based on a new two-step schema matching technique that takes into account the dependency relations among the attributes. In fact [Kang and Naughton, 2003] consider the schema matching problem as a graph matching problem. A labeled graph is built by capturing and structuring dependencies between attributes. Nevertheless all possible cardinality constraints are considered when building the graph for the matching strategy. Therefore this solution is too generic and do not directly index the case of one-to-one correspondences we focus on. Because the technique does not relies on the interpretation of data elements, it can somehow complement existing techniques and can be combined with some schema matching technique like ones based on IND when enlarging the problem to more generic cardinality constraints such as partial mapping.

In general, many techniques have been proposed to identify matching attributes. For example in [Zhang et al., 2010], a robust algorithm for discovering single-column and multi-column foreign keys is proposed. The algorithm may be used as a mapping algorithm for mining correspondences between attributes. The problem has also been addressed in others domain such as ontology alignment by [Euzenat and Shvaiko, 2007] for example.

Moreover, an interesting survey has been done by [Rahm and Bernstein, 2001]. The authors present a taxonomy that covers many existing techniques that have been classified in term of schema, instance, structure, language and constraint level. Different existing approaches are characterized and compared in a very useful way for developing more effective schema matching algorithms.

5.4 Experimentation

We remind the DER problem statement. "Given a source relation r, a master relation s and a mapping function f, it is to find all ER satisfied in (r, s) with respect to f". In this section we first experiment the discovery of correspondences based on approximative IND. We then experiment the discovering process of ER.

The approaches for discovering unary IND and discovering ER have been implemented. Techniques are evaluated and compared in this section on both real life and synthetic datasets with various parameters. In particular we run our techniques for discovering ER on the same real life dataset than [Fan et al., 2010] to compare the discovered ER with the ones their manually designed. The synthetic dataset are exclusively used to evaluate our process of discovering one-to-one correspondences between attribute based on approximative IND.

Real Datasets The experiments for discovering and building ER are set in the real life data Hospital Compare [1] that compares high quality data from different hospital in the United States. The database is composed by several tables, among them:

- the table HOSP records the hospital information including provider number (id), hospital name (hName), phone number (phn), state (ST), zip code (ZIP) and address.
- the table HOSP_MSR_XWLK records the score of each measurement on each hospital in HOSP, e.g. measure name (mName), measure code (mCode) and the score of the measurement for this hospital (Score).
- the table STATE_MSR_AVG records the average score of each measurement on hospitals in all US states, e.g. state (ST), mName and state average (sAvg) the average score of all hospitals in this state.

We merge this three main tables to obtain a single table with the following characteristic.

[1]http://www.hospitalcompare.hhs.gov

Datasets	#Attributes	#Tuples	Size (Mo)
Hospital Compare Table	12	170 000	33

The same synthetic datasets and parameters of discovering constant CFD experimentation are reused.

5.4.1 Mining one-to-one correspondences

We previously described our techniques to identify correspondences between attributes based on unary IND. Given two single attribute A and B, the unary inclusion dependency sentence $A \subseteq B$ is equivalent to the sentence A corresponds to B when the dependency is satisfied. Therefore we exclusively experiment the process of discovering unary IND from relations to evaluate the mining of one-to-one correspondences between attributes through a mapping function. We rely on two techniques:

- the first technique is based on the closure operator (property 5.1). The technique is referred as $IND1$.
- the second technique is based on the support of attribute set to approximate unary IND as stated in property 5.4. The technique may be used to extract both exact unary IND as well as approximative ones. This technique is referenced as $IND2$.

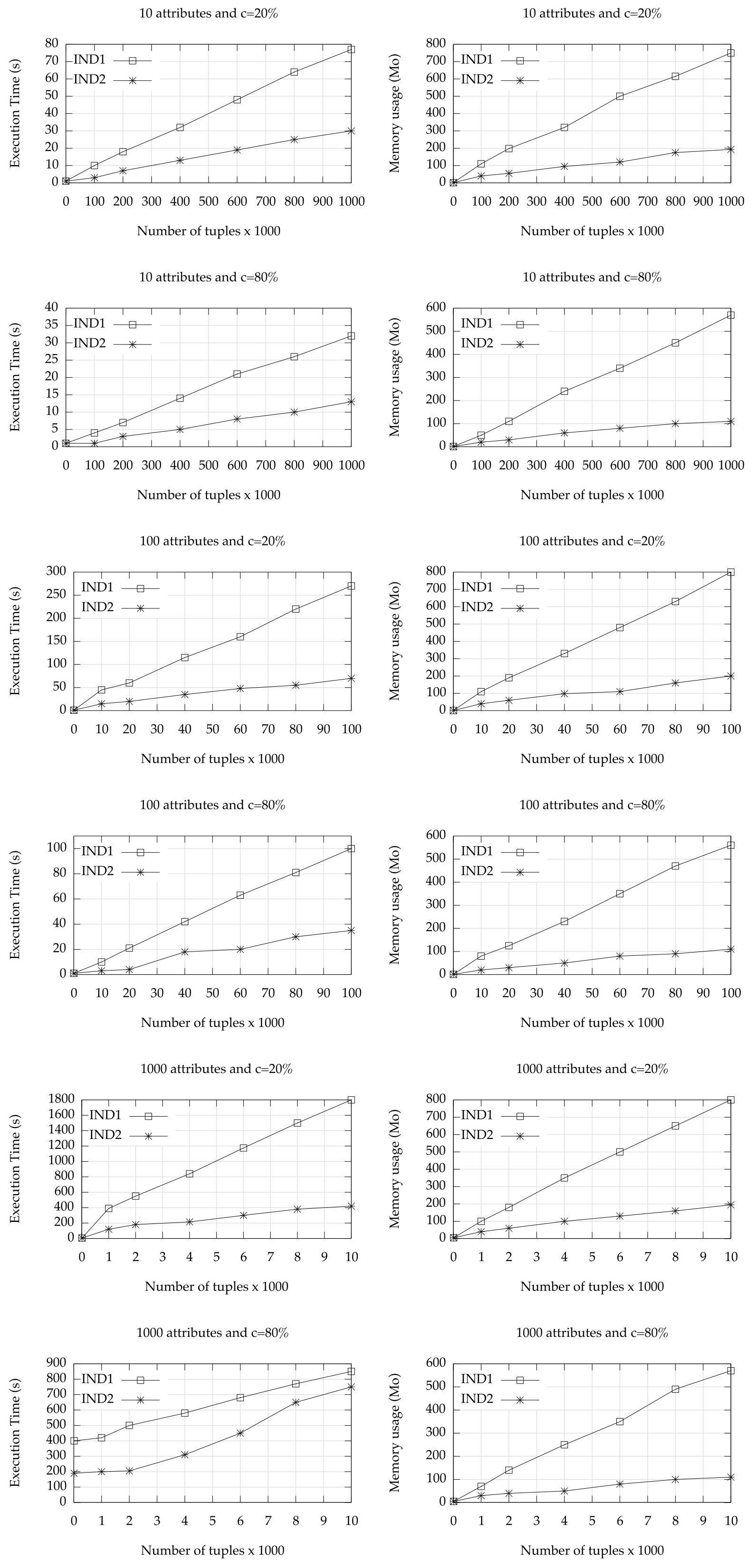


Figure 5.4: Response time and Memory usage with respect to instance size for different attribute size and correlation rate

The first step of computing condensed representation of relations is similar to both of the techniques and is out of our evaluation scope.

For both techniques, we first evaluate response time and memory usage with respect to instance size. We generate synthetic data by fixing the number of attribute to 10, then 100, and finally 1000. For each set we have used two different correlation factors, 20% with less similarities between attribute values and 80% to increase similarities between them.

In Figure 5.4, $IND1$ outperforms $IND2$. Iterations over the search space, attribute search process with a complexity of $O(n)$ and intersection computing between the result of closure are heavy. In contrast the $IND2$ technique has an dichotomous attribute research approach in $O(log(n))$. It is also no longer necessary to compute intersection of closure sets, the ratio of support is enough to extract unary IND. The support computation is not time consuming and is directly obtained from the condensed representation.

We evaluate the behaviour of the techniques when the number of attributes evolves for a given instance of 10000 tuples with a correlation rate of 80%. In Figure 5.5 we note the premise of an exponential behaviour of response time and memory usage for both techniques. When the number of attributes increases, in the one hand computing closure and intersection between sets becomes heavy and time consuming. In the same way the number of candidates generated increases exponentially.

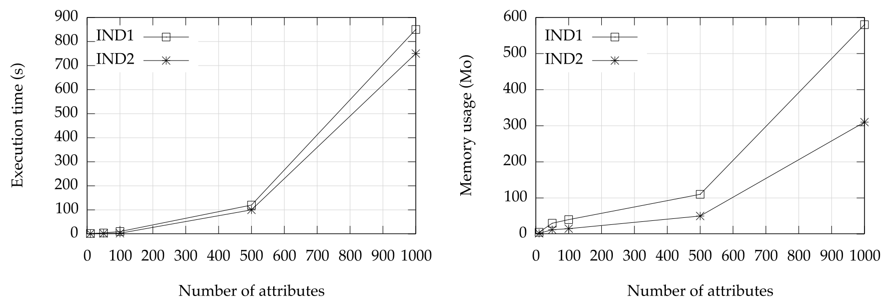


Figure 5.5: Response time and Memory usage with respect to the number of attributes for a 10 000 tuples instance with a correlation rate of 80%

In Figure 5.6, when the correlation rate increases, there are more and more similar values in the instance relation. Therefore the space search is reduced and the storing structure are less heavy and easy to iterate. This reduce the time response and the memory usage. In another hand the number of IND increases because values are more and more similar, so easily included each other. It is clear that whenever the technique is, the number of discovered unary IND is the same. In addition using $IND2$ approach, we can infer approximative unary IND.

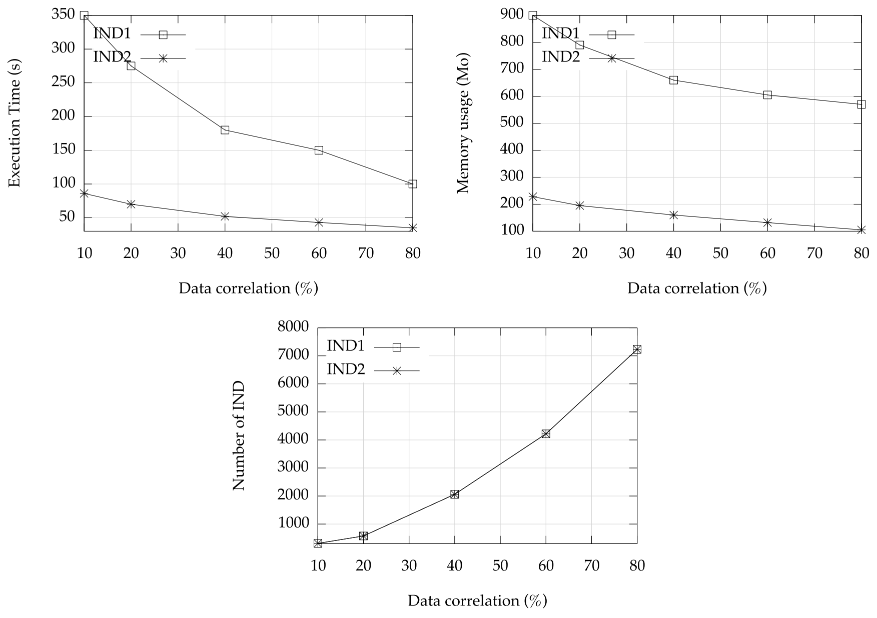


Figure 5.6: Response time, Memory usage, Number of IND for a 100 000 tuples and 100 attributes instance

5.4.2 Discovering Editing Rules

The discovery process of ER follows naturally when correspondences have been extracted. In the sequel we describe and experiment how ER are built once the mapping function is known. The experimental study has been set up in the same condition of [Fan et al., 2010]. We use the same Hospital Compare real life dataset. [Fan et al., 2010] manually designed for HOSP data 37 ER in total, obtained by a careful analysis. Only few of them are publicly available. Indeed five ER cited by [Fan et al., 2010] are:
φ_1=((ZIP,ZIP)→(ST,ST),$t_{p1}[ZIP]$=());
φ_2=((phn,phn)→(ZIP,ZIP),$t_{p2}[phn]$=());
φ_3=((mCode,ST),(mCode,ST))→(sAvg,sAvg),t_{p3}=());
φ_4=((id,mCode),(id,mCode))→(Score,Score),t_{p4}=());
φ_5=(id,id)→(hName,hName),t_{p5}=());

We have been able to recover all ER listed by [Fan et al., 2010] using our technique. We discover a canonical cover of constant CFD from the master relation. For each discovered constant CFD $(X \rightarrow A, t_p[Z])$, we automatically generate the ER $((f^{-1}(X), X) \rightarrow (f^{-1}(A), A), t_p[Z])$ using the mapping function f. For example the ER φ_5=(id,id)→ (hName,hName),t_{p5}=()) is equivalent to the set of constant CFD in the form $id \rightarrow hName$. The Figure 5.7 recalls

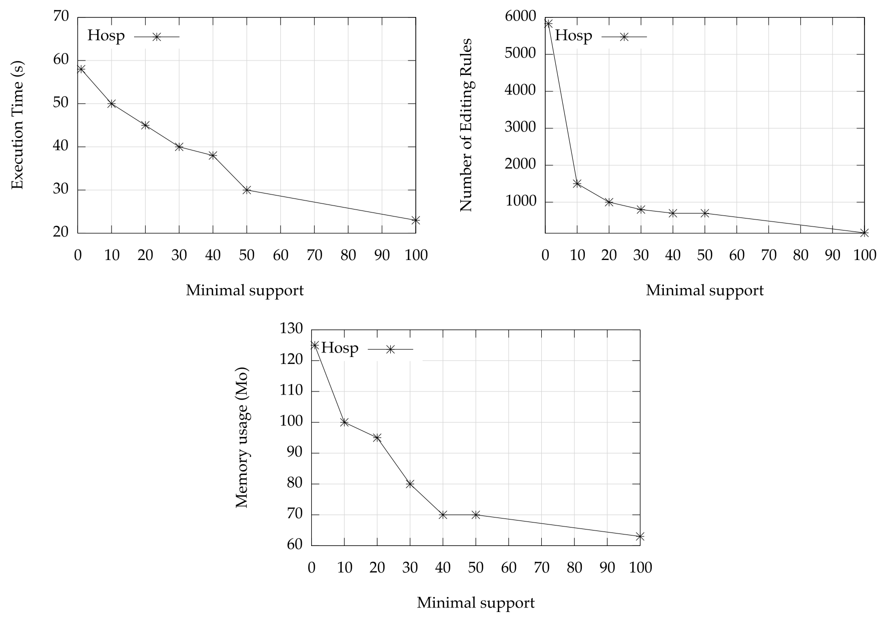


Figure 5.7: Execution time, number of ER, and Memory usage for the *Hosp* real life dataset

our previous experiment for the discovery of constant CFD in section 4.5 of Chapter 4. The experiment is run here with the Hospital Compare real life dataset. The behaviour is similar, i.e. the memory usage and the response time decrease when the support increases even if the approach spend more time and consume more memory to discover constant CFD for the Hospital Compare dataset.

5.5 Summary

ER are a new class of data quality rules boosted by the emergence of master data both in industry [Deloitte and Oracle, 2005, Russom, 2008, Power, 2010] and academia [Fan et al., 2010]. In this Chapter we attack the discovering problem of ER from master data and source data. We proposed a new semantics of ER in order to be able to infer them from existing source relation and a corresponding master relation. Based on this new semantics, we have proposed a mining process in 3 steps:

1. Eliciting one-to-one correspondences between attributes of a source relation and attributes of the master database using unary IND. We have presented a first approach based on intersection of closure sets. The latter technique is improved by proposing a second approach taking

into account the support of attribute set and the error measure defined by [Lopes et al., 2002] to efficiently compute and approximate unary IND.

2. Once correspondences between master attribute and source attributes identified, we mined a cover of constant CFD satisfied in the master relation as described in Chapter 4.

3. Finally we presented how to build ER using the mapping function extracted from step one and CFD mined from step two. As a result, we were able to discover and automatically build all ER manually designed and listed by [Fan et al., 2010], which is a good result.

In a data cleaning setting, the process of discovering ER is just a step. We next attack the question: how to efficiently apply ER in order to clean data sources?

Chapter 6

Editing Rules for Data Cleaning

Chapter Outline

6.1 Data Repairing based on CFD **77**
6.2 Data Repairing based on ER **78**
6.3 Related Works **80**
6.4 Experimentation **81**
6.5 Summary **83**

This Chapter addresses the repair process based on ER. We give a quick overview of main issues of techniques based on CFD before using ER in a data cleaning setting. We propose heuristics to efficiently apply ER. We experiment the techniques in a real life database to show their feasibility and their scalability.

Data quality problems occurs in databases mainly due to invalid data or missing information. Data cleaning, also called data cleansing or scrubbing, deals with detecting and removing errors and inconsistencies from data in order to improve the quality of data [Rahm and Do, 2000]. More, data cleaning is very relevant when data sources are multiple such as in MDM context. In the sequel we point out the limitation of data cleaning approach based on CFD and then we propose heuristic repairs using ER.

6.1 Data Repairing based on CFD

CFD outperform traditional FD when coming to detect inconsistencies. Therefore they are more suitable in practice when developing constraint based method for improving data quality through data cleaning. In this setting, the detection of constraint violation is the first step for data cleaning before removing inconsistencies from the data.

r_1	A	B	C	D	E
t_1 :	0	0	1	2	3
t_2 :	0	0	1	4	5
t_3 :	1	0	2	2	0

$$\varrho_1 = (DC \to E, (*,*,2,2 \parallel 5))$$
$$\varrho_2 = (AB \to C, (0,0 \parallel 2,*,*))$$
$$\varrho_3 = (CD \to E, (*,*,1,2 \parallel 5))$$
$$\varrho_4 = (CD \to E, (*,*,1,4 \parallel 5))$$

Figure 6.1: Relation r_1 over $ABCDE$ and a set of CFD

Actually removing or repairing inconsistencies using CFD is very challenging. For example, the repair of dirty tuples may cause some constraints rules no more applicable [Bohannon et al., 2007]. In this setting the order in which they are applied is important to maximize the number of rules used in practice to correct data.

We illustrate this issue in Figure 6.1. When the set of CFD is applied in the order they appear on Figure 6.1, only ϱ_2 is applied. There are no tuples that match the pattern of ϱ_3 and ϱ_4 because ϱ_2 changed the value of $t[C]$ from 1 to 2. On the other hand, if we apply the following order ϱ_4, ϱ_3, ϱ_2 and finally ϱ_1, all of them are actually considered, ϱ_2 and ϱ_1 are applied. Figure 6.2 outputs two different relations obtained as a result.

This example highlights the importance of the order in which constraint rules are applied. Therefore it is necessary to develop heuristics that maximize the number of rules applied in practice, even for ER.

It is also possible to introduce new inconsistencies when trying to repair data using CFD. Precisely, the semantic of a CFD does not guarantee a certain

r_2	A	B	C	D	E
t_1 :	0	0	2	2	3
t_2 :	0	0	2	4	5
t_3 :	1	0	2	2	0

r_3	A	B	C	D	E
t_1 :	0	0	2	2	5
t_2 :	0	0	2	4	5
t_3 :	1	0	2	2	5

Figure 6.2: Relation r_2 after applying $\varrho_1, \varrho_2, \varrho_3, \varrho_4$ and relation r_3 after applying $\varrho_4, \varrho_3, \varrho_2, \varrho_1$

fix. CFD as traditional FD do not carry the information about which value to choose when correcting the inconsistencies. For example, let us consider a dependency $X \rightarrow Y$ defined in a schema R of a relation r. If there exits two different tuples t_1 and t_2 in r such that $t_1[X] = t_2[X]$ and $t_1[Y] \neq t_2[Y]$, the dependency is violated. But there does not exist a unique way to correct the inconsistency. One can change the value of $t_1[Y]$ by $t_2[Y]$ to have the equality or vice versa. This may lead to create new inconsistencies when the good value is not set even if the dependency is satisfied. ER overcome this limitation with respect to the good value taken from master data.

6.2 Data Repairing based on ER

Once ER are discovered, it is necessary to have heuristics to apply the rules in some given order to correct data.

Heuristic H_0: Baseline Let Σ be the set of discovered ER and r the relation to be cleaned. The use of ER in a basic and natural data cleaning process is described in Algorithm 12, which corresponds to a first naive approach and referred as baseline technique. The rules are applied randomly, i.e. in the order they are produced during the discovery.

Algorithm 12 Heuristic H_0: Baseline

```
Require: $r, \Sigma$
Ensure: A new relation $r' \models \Sigma$ and cpt: the number of corrected tuples of $r$.
 1: $cpt = 0$;
 2: $r' = r$;
 3: for all $t \in r'$ do
 4:     for all $(X,Y) \to (A,B), t_p[Z] \in \Sigma$ such that $t \asymp t_p[Z]$ do
 5:         if $t[X] \asymp t_p[Z]$ then
 6:             $t[A] := t[B]$
 7:             $cpt++$
 8:         end if
 9:     end for
10: end for
11: return $r', cpt$
```

The number of corrected tuples *cpt* into the relation is kept to evaluate the accuracy of the process. The others heuristics are compared to the baseline approach.

Heuristic H_0^*: Baseline-Recall The heuristic H_0^* called Baseline-Recall extends the baseline approach by iterating over rules. Therefore a rule not applied at a given iteration step i can be finally applied at the next iteration step $i+1$. For example, on Figure 6.1 at a first step we apply the set of rules ϱ_1, ϱ_2, ϱ_3 and ϱ_4 using baseline heuristic H_0. In this case, only ϱ_2 is applied. As a consequence, $t[C]$ is changed from 1 to 2. The use of Baseline-Recall heuristic H_0^* ensure a second iteration and then ϱ_1 can be now applied by editing $t_1[E] = t_3[E] = 5$ to obtain the relation r_3. Since we have no guarantee of termination in some noisy case due to rule conflict (see example 3.11 of Chapter 3) we have set a maximum value of iteration to 100.

Heuristic H_1: Left-Hand-Side-Length-Sorted We now investigate sorting the rules with respect to the size of their left hand side (the largest first). In fact the application of a rule may invalidate some others when the attributes involved in rules are similar. When rules are sorted with respect to their left hand sides, the largest are applied first because they carries a repair process on many attributes. By the way this strategy is more suitable for rules build from FD.

Heuristic H_2: Left-Hand-Side-Length-and-Support-Sorted In this strategy, when two or many sets of left hand side attributes have the same size, we take into account the support of rules. In this case, the rule with the biggest support is applied first. In fact, the rules with largest support impact more tuples when applied and thus are more effective.

The Recall strategy of iteration can be also applied to H_1 and H_2 respectively to obtain H_1^* and H_2^*.

6.3 Related Works

This work finds similarities with data repair techniques based on constraints proposed this last decade. First data repairing techniques concentrate on constraints. The main goal of repairing data based on constraints is to apply changes to data so they can satisfy the constraints early violated due to the presence of errors. Early works such as [Chomicki and Marcinkowski, 2005] evaluated the cost of changes in term of tuples insertions and deletions. In contrast [Bohannon et al., 2005] propose a cost based model stated in term of value modification instead of tuple insertions and deletions. In this case, the repair problem becomes much more difficult and complex when tuples modifications is considered instead of tuples insertions and deletions. The approach used is carried by FD and IND and is built using equivalence classes. This technique suggests repair for dirty data but the repair when applied may introduce new inconsistencies.

In the same context, the cost based model of [Bohannon et al., 2005] is extended by [Cong et al., 2007] with respect to CFD to improve the consistency and accuracy of data. [Cong et al., 2007] first characterize how dirty the data is by formalizing the violation of CFD and then better resolve the violation. The repair operation is at attribute value modifications level with respect to the cost model enriched using the levenstein metric [Galhardas et al., 2001] in order to reduce repair cost and then improve accuracy. The levenstein distance guarantees a minimum of modifications to correct dirty values to obtain good ones. But repairs still may introduce some errors. When dealing with ER this model is unnecessary, good values are directly taken from master data. Nevertheless repairs can be improved by proposing only repairs that are above of predefined user threshold with a high confidence. The user feedback is fully explored and integrated in the dirty data repair process by [Yakout et al., 2011]. The repair process is then improved but the framework only relies on CFD.

In a dynamic environment where data and constraints evolve, [Chiang and Miller, 2011] propose a novel unified cost model for data and constraint repair. The co-dependence between constraint and data may be relevant when a MDM solution is integrated to an existing information system for example. New data sources have to be integrated in the master data repository and constraint may evolve over time. Even if the approach is just limited to FD, an equivalence can be found with the work of [Fan et al., 2008b]. In a schema integration context such as MDM, this alternative requires a mapping between old schema and new evolved schema. The goal is quite similar since with CFD conditions are carried by values and when values are modified then the constraints may be modified.

6.4 Experimentation

In this section we experiment the data repair based on ER. We use the same real life dataset as the one for the discovery problem. Therefore we set an experimental protocol as follows, to be as close as possible to a real master data context:

1. We reuse the relation of Hospital Compare dataset (Chapter 5, section 5.4).
2. The relation is duplicated to obtain a second relation r'.
3. Null values are introduced into r' at attribute level to simulate noise.
4. ER are discovered on Hospital Compare to obtain a set Σ of rules.
5. The set Σ of ER is applied to the noisy relation r' to correct inconsistencies.

Datasets	#Attributes	#Tuples	Size (Mo)
Hospital Compare Table	12	170 000	33

Our experimentation have been performed on the same configuration than in Chapter 5. Additional parameters are set: the noise rate of the relation is defined as the ratio between the number of noisy tuples and the total number of tuples. The accuracy is in term of percentage of errors corrected, it is defined as the ratio between the total number of corrected tuples and the total number of noisy tuples.

Experimentation are run on Hosp dataset with 10% of noise according to our protocol. The response time, the memory usage and the number of ER effectively applied are measured for the heuristics. The number of ER discovered with respect to the Hosp dataset is about 6000 (Figure 5.7) when the support is minimal. We consider this number of ER as input for the correction. For the measure of accuracy of data repair, we have varied the noise rate from 1 to 10%, which is a realistic interval. All the results are aggregated on Figure 6.3 illustrating heuristics of data repair using ER.

Whatever the heuristic is, all the ER are not applied because the application of one ER may invalidate some others. Heuristics H_1 and H_2 are a little more time consuming (Fig 1). In fact they impact more tuples due to the sorting strategy even if they apply less ER (Fig 3). The memory usage is quite similar for all non iterative strategies (Fig 2) because there is only a single pass on ER and non applied rule are tested even if "useless". When the iterative strategies are applied the memory usage slightly increases (Fig 6), ER not applied at step i are marked and may be applied at step i+1. Iteration increases the number of applied ER (Fig 7). When a rule with a large support is applied, the response time increases (Fig 5) because the repair impact more tuples.

Generally, the quality of the repair decreases when the noise rate increases, but it remains superior to 80% for all strategies (Fig 4 and Fig 8). Heuristics are quite equivalent when inconsistencies are in low rate, i.e. less than 3% for non iterative heuristics and less than 4% for iterative heuristics. The distance between non iterative strategies is 90-80 = 10% of correction. For iterative strategies the distance is 6%. So the distance decreases because iterations maximize repairs. Iterative strategies outperforms non iterative ones in term of correction. They repair more inconsistencies while the memory cost and the response time remain acceptable with respect to non iterative strategies. The heuristic H_2^* gives the best accuracy rate (96%) for the worst case of 10% of noise.

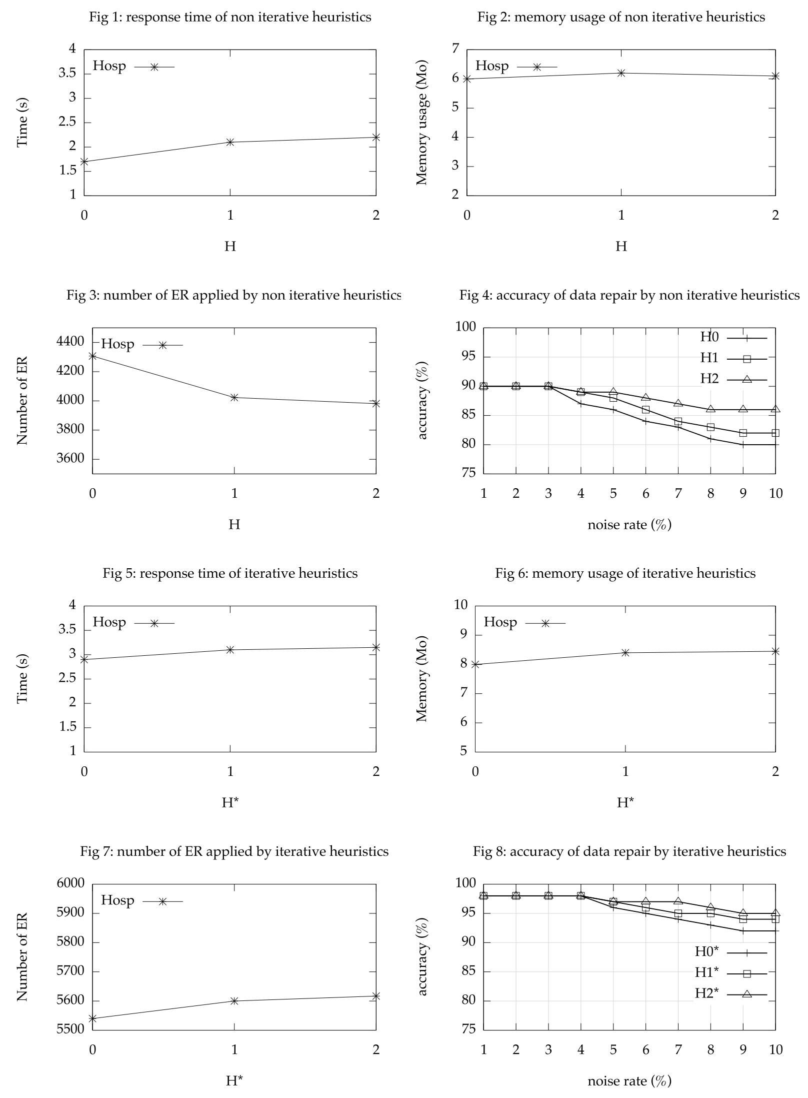


Figure 6.3: Response time, Memory usage, Number of ER and accuracy of Heuristics

6.5 Summary

In this Chapter, we presented main issues when solving dirty data using CFD. The repair techniques based on CFD does not ensure a quality of repair in term of good value to choose when correcting inconsistencies unlike ER. The semantics of ER carries precisely the value taken from master data to consider when repairing. The solution proposed for the mining problem of ER helped us build heuristics in order to apply automatically discovered ER for data cleaning. The most basic of them apply the set of ER naively without any optimization. We improved this baseline approach by iterating over rules to maximize the repair. A benefit is also observed when the rules are sorted with respect to the support.

CHAPTER 7

CONCLUSION

Chapter Outline

7.1 Summary of Contributions **86**
7.2 Discussion and Future works **87**

In this last Chapter we first summarize our contributions in the domain of mining data quality rules and data cleaning. We finally open discussions about these issues and we underline some relevant future works to consider.

7.1 Summary of Contributions

Dirty data is still an important issue to solve. Lots of time and money are spent by companies to have techniques in place that can efficiently clean data. Mining data quality rules is an important step towards solving this issue. The emergence of MDM enhance data quality and can improve rules. This thesis brings contributions for the discovery of CFD and also for the mining of ER.

We have proposed two approaches for the discovery problem of constant CFD. We have named the first one **conditional agree set**, a contribution that extends well known agree sets [Fagin and Vardi, 1983] to automatically extract CFD. The proposal is suitable only for mining all possible constant CFD without any threshold. The second contribution for the discovery problem of constant CFD is an **extension of the** FUN **approach** [Novelli and Cicchetti, 2001b]. The advantage of this approach is to easily deal with the frequency of constant CFD which was not integrated in the approach based on conditional agree sets. The notion of free sets of FD was extended to get conditional free sets. **The** CFUN **algorithm** is based on the concepts of well known APRIORI [Agrawal and Srikant, 1994] algorithm to find conditional free sets. Once the conditional free sets are discovered with the corresponding frequency for each level, the canonical cover of constant CFD is generated through the closure. The two approaches have been tested against different synthetic and real life datasets using CFUN and `Cag`. The results show a linear scalability in the size of the relation, but remain exponential with respect to the arity. CFUN is less time and memory consuming when the support of constant CFD increase. CFUN outperforms `Cag` approach when the correlation rate of attribute values decreases.

ER are a new class of data quality rules boosted by the emergence of master data both in industry [Deloitte and Oracle, 2005, Russom, 2008, Power, 2010] and academia [Fan et al., 2010]. In the same line with the discovery of CFD, we also attack the discovery problem of ER from master data and source data. The problem has not been yet addressed by the data mining community to the best of our knowledge. We have proposed a **new semantics of ER** in order to be able to infer ER from existing source database and a corresponding master relation. Based on this powerful semantics, we have proposed a **mining process of ER**. We first find one-to-one correspondences through a mapping function from source to master attributes using unary IND. The experimental results have showed the scalability of the approach. Then we mined a cover of constant CFD satisfied by the master relation before building ER thanks to the mapping function. As a result, we were able to find all ER manually designed and listed by [Fan et al., 2010].

We have also proposed **heuristics to clean data by applying ER**. The most basic H_0 applies ER randomly, in the same order they are discovered. Two strategies of improvement have been proposed. One sort rules with respect to the left hand sides (H_1) and the other one (H_2) take into account the support in H_1. The repair process reaches more tuples when ER with largest support are applied first because they impact more inconsistencies. Both heuristics show good results when data is very dirty. Additionally, we experimentally verified that the iteration over rules improve the efficiency of the cleaning process.

7.2 Discussion and Future works

We did not experiment the repair process based on CFD and so we did not check its limitations. Nevertheless, according to [Fan et al., 2010] ER are more efficient than CFD. An experimental comparison of these two classes of data quality rules can be relevant as future works. The heuristic repair approach based on applying one Editing Rule at a time must be improved. A better strategy must be found to solve the problem of failure to terminate.
Moreover, since ER is a new class of rules, it is interesting to experiment the power of ER on existing data repair techniques. For example ER may be used in existing framework such as [Yakout et al., 2011] to give the user appropriate control of the repair process, and thus improve repair efficiency.
In a more general formal way, a uniform dependency language for improving data quality have been recently proposed by [Fan and Geerts, 2011]. This uniform framework for dependency powerfully helps the use of ER instead of other constraints rule like CFD. The data repairing strategies may be boosted thanks to Quality Improving Dependencies language of [Fan and Geerts, 2011]. As future works, the heuristics presented in this thesis should be implemented with respect to this language. Furthermore, experimentation with real life master dataset is a must.

ER are a new subject and naturally open new research subject perspectives. In this thesis, we only consider one-to-one mapping. As future works we may extend the mapping function to deal with other cases of mapping such as partial and many-to-many. By the way, the inference of many-to-many correspondences instead of one-to-one is challenging but helps address the problem of mining ER when an entire source database and a master database is consider instead of a single source relation and a single master relation. The main problem is then to propagate the rule into the source data with the “right” attribute.
In addition, the approach based on approximative IND to find the mapping may be enforced with the notion of equivalence classes explored by [Bohannon et al., 2005]. Nevertheless, the part of finding a mapping function between source attributes and master attributes in the resolution of the discovery problem of ER may be avoided. Indeed, one can use record matching techniques to identify links between sources and master data through for example Matching Dependencies rules [Fan et al., 2009].

In this thesis we concentrate attention to the discovery of constant CFD. As

feature works, we plan to address the problem of discovering variable CFD. For instance level-wise strategies can be applied for variable CFD even if as such, this pattern enumeration problem has to be proved as representable as sets [Mannila and Toivonen, 1997]. It is also possible to try to infer variable CFD from constant ones already discovered. Actually, if the pattern of a single constant CFD cover all the active domain of an right hand side attribute, then the set of the active domain can be replaced by the unnamed variable "_" which finally corresponds to the structure of an variable CFD.
In addition, the extension of agree sets we proposed may help to study the existence of Armstrong relation [Beeri et al., 1984b] for CFD, i.e. a relation that uniquely satisfies a given set of CFD. It may be interesting to study the complexity, the properties and the building process of such a relation.
It is worth noticing that mining frequent CFD share some characteristics with the problem of mining frequent projection selection conjunctive queries addressed by [Jen et al., 2008]. It may be interesting to investigate the possible cross-fertilization between these two problems like in [Goethals et al., 2010] using CFD instead of FD. This interaction may also extends the pattern condition of a CFD stated using just equality towards a pattern condition stated by a query with operators like inferior or superior. This correlation may goes further to query optimization. Indeed, the resolution tree of a query through traditional FD may be extended, enriched and more precise when built through CFD and then may improve and fasten access to query resolution.
Discovering Conditional IND is a another perspective towards extending our work of mining CFD from a relation to a database in order to take into account links between relations. Conditional IND may also be considered to define Armstrong database, for example by extending the work of [Fabien De Marchi, 2005].

MDM is a growing subject that receives more and more attention from companies in particular. In this thesis we give an overview of MDM by presenting different vision through definitions and practice. We highlighted MDM key futures and presented the main issues. The most common implementation styles of an MDM solution was also presented. The choice of the style to implement generally depends on business needs. In this thesis, the most relevant added value of MDM is the codependence between MDM and Data Quality. We saw that MDM can improve quality of data, and in another side Data Quality Management can boost an MDM solution. This interaction is the intuition behind the definition of ER to overcome the limits of data dependency rules like CFD when used to clean data. The way of choosing data to master was out of the scope of this thesis. An entire detailed study is necessary to figure out which data to master with respect to the business need and the complexity of the information system in place. In this thesis we found that master data and data cleaning can be complementary and can boost efficiency of traditional cleansing techniques. We think that other relevant subjects can also benefit from MDM.

LIST OF ALGORITHMS

1 Minimal cover for CFD 28
2 The space search of constant CFD 35
3 The conditional agree sets between each tuple of a relation and a pattern 37
4 The conditional agree sets between a relation and a pattern . . 38
5 CFUN 48
6 Condensed representation of relation r 61
7 Closure of an attribute with respect to the condensed representation 62
8 Scalable Closure 65
9 (SR2MR) Mapping from Source Relation to Master Relation . . 66
10 FindBest: find the best corresponding attribute 66
11 Discovering ER 67
12 Heuristic H_0: Baseline 79

List of Figures

2.1 A word cloud of MDM survey definitions 11
2.2 Different applications modeling the same customer differently [Loshin, 2009] 12
2.3 Different MDM architecture implementations [Gartner, 2010] . . . 14
2.4 An overview of MDM Implementation styles [Dreibelbis et al., 2008] 16
2.5 Data Quality Dimensions [Otto and Ebner, 2010] 17

3.1 The relation Person 21
3.2 relation *cust* 22
3.3 Inference system for CFD [Bohannon et al., 2007] 27
3.4 Source relation r and master database $\{s, u\}$ 29

4.1 An Overview of the Steps That Compose the KDD Process [Fayyad et al., 1996] 32
4.2 A toy relation r over $R = \{A, B, C, D\}$ 36
4.3 Illustration of the proposed characterisation 49
4.4 Execution time , number of CFD, and memory usage for the *Wisconcin Breast Cancer* and *Chess* real life datasets 51
4.5 Execution time, number of CFD, and memory usage for various number of tuples 52
4.6 Execution time, number of CFD, and memory usage for various number of attributes 53
4.7 Execution time, number of CFD, and memory usage for various data correlation rates 54

5.1 A master database with two source databases 56
5.2 Source relation r and master relation s from Figure 3.4 58
5.3 Condensed Representation of the toy illustration relation r 60
5.4 Response time and Memory usage with respect to instance size for different attribute size and correlation rate 71
5.5 Response time and Memory usage with respect to the number of attributes for a 10 000 tuples instance with a correlation rate of 80% 72
5.6 Response time, Memory usage, Number of IND for a 100 000 tuples and 100 attributes instance 73
5.7 Execution time, number of ER, and Memory usage for the *Hosp* real life dataset 74

6.1 Relation r_1 over $ABCDE$ and a set of CFD 77

6.2 Relation r_2 after applying $\varrho_1, \varrho_2, \varrho_3, \varrho_4$ and relation r_3 after applying $\varrho_4, \varrho_3, \varrho_2, \varrho_1$ 78
6.3 Response time, Memory usage, Number of ER and accuracy of Heuristics 83

BIBLIOGRAPHY

[Abiteboul et al., 2000] Abiteboul, S., Hull, R., and Vianu, V. (2000). *Fondations of Databases*. Vuibert. 20

[Agrawal et al., 1993] Agrawal, R., Imielinski, T., and Swami, A. N. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press. 5, 33, 47

[Agrawal and Srikant, 1994] Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of the 1994 VLDB International Conference, 1994*. VLDB Press. 6, 47, 65, 86

[Armstrong and Delobel, 1980] Armstrong, W. and Delobel, C. (1980). Decompositions and functional dependencies in relations. *ACM*. 5

[Armstrong, 1974] Armstrong, W. W. (1974). Dependency structures of data base relationships. *DBLP*, pages 580–583. 23, 27

[Batini and Scannapieco, 2006] Batini, C. and Scannapieco, M. (2006). *Data Quality: Concepts, Methodologies and Techniques (Data-Centric Systems and Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. 17, 92

[Bauckmann et al., 2007] Bauckmann, J., Leser, U., Naumann, F., and Tietz, V. (2007). Efficiently detecting inclusion dependencies. In *ICDE*, pages 1448–1450. 68

[Baudinet et al., 1995] Baudinet, M., Chomicki, J., and Wolper, P. (1995). Constraint-generating dependencies. *Journal of Computer and System Sciences*, 59:322–337. 32

[Beeri et al., 1984a] Beeri, C., Dowd, M., Fagin, R., and Statman, R. (1984a). On the structure of armstrong relations for functional dependencies. *J. ACM*, 31(1):30–46. 5, 6

[Beeri et al., 1984b] Beeri, C., Dowd, M., Fagin, R., and Statman, R. (1984b). On the structure of armstrong relations for functional dependencies. *J. ACM*, 31(1):30–46. 32, 37, 88

[Bellahsene et al., 2011] Bellahsene, Z., Bonifati, A., and Rahm, E., editors (2011). *Schema Matching and Mapping*. Springer. 15

[Berge, 1987] Berge, C. (1987). *Hypergraphes*. BORDAS. 41

[Berti-Equille, 2006] Berti-Equille, L. (2006). *Qualité des données*. Technique de l'ingénieur. H3700. 4

[Birkhoff, 1967] Birkhoff, G. (1967). *Lattices Theory*. Coll. Pub. XXV, vol. 25, 3rd edition. 46

[Bohannon et al., 2005] Bohannon, P., Fan, W., and Flaster, M. (2005). A cost-based model and effective heuristic for repairing constraints by value modification. In *In ACM SIGMOD International Conference on Management of Data*, pages 143–154. 6, 80, 87

[Bohannon et al., 2007] Bohannon, P., Fan, W., Geerts, F., Jia, X., and Kementsietsidis, A. (2007). Conditional functional dependencies for data cleaning. In *ICDE*, pages 746–755. 4, 5, 6, 23, 27, 28, 33, 77, 90

[Bonnet, 2009] Bonnet, P. (2009). *Management des données de l'entreprise*. Lavoisier. 10

[Bonnet, 2010] Bonnet, P. (June 2010). *Enterprise Data Governance Master Data Management and Semantic Modeling: MDM*. Wiley-ISTE. 10, 11

[Borgelt and Kruse, 2002] Borgelt, C. and Kruse, R. (2002). Induction of association rules: Apriori implementation. In *Proc. 15th Conf. on Computational Statistics (Compstat 2002, Berlin, Germany)*, pages 395–400, Heidelberg, Germany. Physika Verlag. 47

[Bra and Paredaens, 1983] Bra, P. D. and Paredaens, J. (1983). Conditional dependencies for horizontal decompositions. In *Proceedings of the 10th Colloquium on Automata, Languages and Programming*, pages 67–82, London, UK. Springer-Verlag. 33

[Business-Intelligent-Network, 2007] Business-Intelligent-Network (2007). *Getting Started with Master Data Management*. Business Intelligent Network research report. 11

[Chiang and Miller, 2008] Chiang, F. and Miller, R. J. (2008). Discovering data quality rules. *PVLDB*, 1(1):1166–1177. 33

[Chiang and Miller, 2011] Chiang, F. and Miller, R. J. (2011). A unified model for data and constraint repair. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, pages 446–457, Washington, DC, USA. IEEE Computer Society. 80

[Chomicki and Marcinkowski, 2005] Chomicki, J. and Marcinkowski, J. (2005). Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1/2):90–121. 80

[Cong et al., 2007] Cong, G., Fan, W., Geerts, F., Jia, X., and Ma, S. (2007). Improving data quality: Consistency and accuracy. *VLDB*. 6, 80

[Cosmadakis et al., 1986] Cosmadakis, S., Kanellakis, P., and Spyratos, N. (1986). Partition Semantics for Relations. *Journal of Computer and System Sciences*, 33(2):203–233. 48

[Deloitte and Oracle, 2005] Deloitte and Oracle (2005). *Getting Started with Master Data Management*. Deloitte and Oracle White paper. 4, 10, 11, 29, 74, 86

[Diallo et al., 2010a] Diallo, T., Doré, M., and Mestelan, J.-B. (2010a). Core cache externalization. *Orchestra Networks internal report*. 92

[Diallo et al., 2010b] Diallo, T., Doré, M., and Mestelan, J.-B. (2010b). Query optimization on externalized mode. *Orchestra Networks internal report*. 92

[Diallo and Novelli, 2010] Diallo, T. and Novelli, N. (2010). Découverte des dépendances fonctionnelles conditionnelles fréquentes. In *10ièmes Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances (EGC'10)*, RNTI E-19, pages 315–326. 7

[Diallo et al., 2012] Diallo, T., Novelli, N., and Petit, J.-M. (2012). Discovering (frequent) constant conditional functional dependencies. *International Journal of Data Mining, Modelling and Management (IJDMMM)*, Special issue "Interesting Knowledge Mining":1–20. 7

[Diallo et al., 2012a] Diallo, T., Petit, J.-M., and Servigne, S. (2012a). Discovering Editing Rules for Data Cleaning. In *10th International Workshop on Quality in Databases In conjunction with VLDB (Very Large Databases) Istanbul 2012*. 7

[Diallo et al., 2012b] Diallo, T., Petit, J.-M., and Servigne, S. (2012b). Règles d'Edition: fouille et application au nettoyage de données. In *28ème journées Base de Données Avancées. Clermont Ferrand 2012*. 7

[Dreibelbis et al., 2008] Dreibelbis, A., Hechler, E., Milman, I., Run, P. V., and Wolfson, D. (2008). *Introducing Master Data Management, An SOA Approach to Managing Core Information*. IBM Press. 16, 90

[Eckerson, 2002] Eckerson, W. (2002). *Data Quality and the Bottom Line*. Technical report. iii, 4, 92

[Eschinger, 2008] Eschinger, C. (2008). *Report Highlight for Market Trends: Master Data Management Growing Despite Worldwide Economic Gloom, 20007-2012*. Gartner. 4

[Euzenat and Shvaiko, 2007] Euzenat, J. and Shvaiko, P. (2007). *Ontology matching*. Springer-Verlag, Heidelberg (DE). 68

[Fabien De Marchi, 2005] Fabien De Marchi, J.-M. P. (2005). Semantic sampling of existing databases through informative armstrong databases. *Science Direct*. 88

[Fagin and Vardi, 1983] Fagin, R. and Vardi, M. Y. (1983). Armstrong databases for functional and inclusion dependencies. *Information Processing Letters*, pages 13–19. 53, 86

[Fan and Geerts, 2011] Fan, W. and Geerts, F. (2011). Uniform dependency language for improving data quality. *IEEE Data Eng. Bull.*, 34(3):34–42. 87

[Fan and Geerts, 2012] Fan, W. and Geerts, F. (2012). *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers. 17

[Fan et al., 2008a] Fan, W., Geerts, F., Jia, X., and Kementsietsidis, A. (2008a). Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, 33(2). 26, 27

[Fan et al., 2011] Fan, W., Geerts, F., Li, J., and Xiong, M. (2011). Discovering conditional functional dependencies. *IEEE Trans. Knowl. Data Eng.*, 23(5):683–698. 33, 36, 44, 50

[Fan et al., 2009] Fan, W., Jia, X., Li, J., and Ma, S. (2009). Reasoning about record matching rules. *Proc. VLDB Endow.*, 2(1):407–418. 87

[Fan et al., 2010] Fan, W., Li, J., Ma, S., Tang, N., and Yu, W. (2010). Towards certain fixes with editing rules and master data. In *Proceedings of VLDB'10*. 5, 29, 68, 69, 73, 74, 75, 86, 87

[Fan et al., 2008b] Fan, W., Ma, S., Hu, Y., Liu, J., and Wu, Y. (2008b). Propagating functional dependencies with conditions. *Proc. VLDB Endow.*, 1(1):391–407. 80

[Fayyad et al., 1996] Fayyad, U., Piatetsky-shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54. 32, 90

[Flouvat et al., 2009] Flouvat, F., Marchi, F. D., and Petit, J.-M. (2009). The open source library izi for pattern mining problems. In *Proceedings of the Open Source in Data Mining (OSDM) workshop, in conjunction with PAKDD'09*, pages 14–25, Bangkok, Thailand. 42, 49

[Galhardas et al., 2001] Galhardas, H., Florescu, D., and Shasha, D. (2001). Declarative data cleaning: Language, model, and algorithms. In *VLDB*, pages 371–380. 80

[Ganter and Wille, 1999] Ganter, B. and Wille, R. (1999). *Formal Concept Analysis: Mathematical Foundations*. Springer. 46

[Gartner, 2010] Gartner (September 2010). *Magic Quadrant for Master Data Management of Product Data*. Gartner. 4, 14, 90

[Goethals et al., 2010] Goethals, B., Laurent, D., and Le Page, W. (2010). Discovery and application of functional dependencies in conjunctive query mining. In *Proceedings of the 12th international conference on Data warehousing and knowledge discovery*, DaWaK'10, pages 142–156, Berlin, Heidelberg. Springer-Verlag. 33, 88

[Goethals and Zaki, 2003] Goethals, B. and Zaki, M. J., editors (2003). *FIMI 2003, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org. 92

[Golab et al., 2008] Golab, L., Karloff, H., Korn, F., Srivastava, D., and Yu, B. (2008). On generating near-optimal tableaux for conditional functional dependencies. *Proc. VLDB Endow.*, 1(1):376–390. 33

[Gottlob and Libkin, 1990] Gottlob, G. and Libkin, L. (1990). Investigations on Armstrong Relations, Dependency Inference, and Excluded Functional Dependencies. *Acta Cybernetica*, 9(4):385–402. 46

[Huhtala et al., 1998] Huhtala, Y., Kärkkäinen, J., Porkka, P., and Toivonen, H. (1998). Efficient Discovery of Functional and Approximate Dependencies Using Partitions. In *ICDE'98, Orlando, Florida, USA*, pages 392–401. 48

[Huhtala et al., 1999a] Huhtala, Y., Karkkainen, J., Porkka, P., and Toivonen, H. (1999a). Tane: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(3):100–111. 33, 34

[Huhtala et al., 1999b] Huhtala, Y., Karkkainen, J., Porkka, P., and Toivonen, H. (1999b). TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *The Computer Journal*, 42(2):100–111. 48

[IBM, 2007] IBM (2007). *MDM: looking beyong the single view to find the right view*. IBM White paper. 11, 12

[Jen et al., 2008] Jen, T.-Y., Laurent, D., and Spyratos, N. (2008). Mining all frequent projection-selection queries from a relational table. In *EDBT 2008, 11th International Conference on Extending Database Technology, Nantes, France, March 25-29, 2008, Proceedings*, pages 368–379. 88

[Kang and Naughton, 2003] Kang, J. and Naughton, J. F. (2003). On schema matching with opaque column names and data values. In *In SIGMOD*, pages 205–216. ACM Press. 68

[Kivinen and Mannila, 1995] Kivinen, J. and Mannila, H. (1995). Approximate inference of functional dependencies from relations. *Theor. Comput. Sci.*, 149(1):129–149. 5

[Levene and Loizou, 1999] Levene, M. and Loizou, G. (1999). *A Guided Tour of Relational Databases and Beyond*. Springer. 20, 21, 57, 59

[Lopes et al., 2000] Lopes, S., Petit, J.-M., and Lakhal, L. (2000). Efficient discovery of functional dependencies and armstrong relations. In *EDBT*, volume 1777 of *LNCS*, pages 350–364, Konstanz, Germany. Springer. 33, 48

[Lopes et al., 2002] Lopes, S., Petit, J.-M., and Toumani, F. (2002). Discovering interesting inclusion dependencies: application to logical database tuning. *Inf. Syst.*, 27(1):1–19. 7, 60, 63, 68, 75

[Loser et al., 2004] Loser, C., Legner, D. C., and Gizanis, D. (2004). Master data management for collaborative service processes. *Institute of Information Management research report , University of St.Gallen*. 10

[Loshin, 2009] Loshin, D. (2009). *Master Data Management*. Morgan Kaufmann. 4, 11, 12, 13, 17, 29, 90

[Maher and Srivastava, 1996] Maher, M. J. and Srivastava, D. (1996). Chasing constrained tuple-generating dependencies. In *PODS*, pages 128–138. 32

[Mannila and Räihä, 1994] Mannila, H. and Räihä, K.-J. (1994). Algorithms for Inferring Functional Dependencies from Relations. *DKE*, 12:83–99. 39, 40

[Mannila and Toivonen, 1997] Mannila, H. and Toivonen, H. (1997). Levelwise search and borders of theories in knowledge discovery. *DMKD*, 1(3):241–258. 40, 41, 42, 88

[Medina and Nourine, 2009] Medina, R. and Nourine, L. (2009). A unified hierarchy for functional dependencies, conditional functional dependencies and association rules. In *ICFCA*, Lecture Notes in Computer Science, pages 235–248. Springer. 33

[Morris and Vesset, 2005] Morris, H. and Vesset, D. (2005). *Managing master data for business performance management: the issues and hyperion's solution*. IDC White paper. 10

[Novelli and Cicchetti, 2000] Novelli, N. and Cicchetti, R. (2000). Mining functional and embedded dependencies using free sets. In *Actes de la 16ème conférence Bases de Données Avancées(BDA'00)*, pages 201–220. 45

[Novelli and Cicchetti, 2001a] Novelli, N. and Cicchetti, R. (2001a). Fun: An efficient algorithm for mining functional and embedded dependencies. In *ICDT*, pages 189–203. 6, 32, 33, 43, 45, 46, 47, 48

[Novelli and Cicchetti, 2001b] Novelli, N. and Cicchetti, R. (2001b). Functional and embedded dependency inference: a data mining point of view. *Information Systems (IS)*, 26(7):477–506. 43, 45, 47, 48, 53, 86

[Orchestra-Networks, 2012] Orchestra-Networks (2012). *www.orchestranetworks.com*. 4

[Otto and Ebner, 2010] Otto, B. and Ebner, V. (2010). Measuring master data quality. *Institute of Information Management, University of St. Gallen*. 17, 90

[Pasquier et al., 1999] Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In *ICDT*, pages 398–416. 33, 43

[Power, 2010] Power, D. (2010). *A Real Multidomain MDM or a Wannabe*. Orchestra Networks white paper. 4, 74, 86

[Rahm and Bernstein, 2001] Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10:334–350. 7, 15, 69

[Rahm and Do, 2000] Rahm, E. and Do, H. H. (2000). Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23:2000. 77

[Redman, 1998] Redman, T. (1998). *The Impact of Poor Data Quality on the Typical Enterprise*, volume 41. CACM. 4

[Rockwell, 2012] Rockwell, D. (2012). *5 Tips for Cleaning Your Dirty Data*. Technical report. iii, 4

[Russom, 2008] Russom, P. (2008). *Defining Master Data Management*. the data warehouse institute. 4, 10, 11, 12, 17, 29, 74, 86

[Shilakes and Tylman, 1998] Shilakes, C. C. and Tylman, J. (1998). *Entreprise Information Portal*. Merill Lynch. 4

[Spyratos, 1987] Spyratos, N. (1987). The partition model: A deductive database model. *ACM TODS*, 12(1):1–37. 48

[The-Data-Warehouse-Institute, 2012] The-Data-Warehouse-Institute (2012). *tdwi.org*. 13

[Webster, 2001] Webster, B. F. (2001). *Pattern in IT Litigation: Systems Failure (1976-2000)*. PricewaterhouseCoopers. 4

[Wyss et al., 2001] Wyss, C., Giannella, C., and Robertson, E. (2001). Fastfds: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances extended abstract. *Data Warehousing and Knowledge Discovery*, pages 101–110. 34

[Yakout et al., 2011] Yakout, M., Elmagarmid, A. K., Neville, J., Ouzzani, M., and Ilyas, I. F. (2011). Guided data repair. *Proc. VLDB Endow.*, 4(5):279–289. 6, 80, 87

[Zaki, 2004] Zaki, M. J. (2004). Mining non-redundant association rules. *Data Min. Knowl. Discov.*, 9(3):223–248. 33

[Zhang et al., 2010] Zhang, M., Hadjieleftheriou, M., Ooi, B. C., Procopiuc, C. M., and Srivastava, D. (2010). On multi-column foreign key discovery. *PVLDB*, 3(1):805–814. 68