Conceptual Knowledge Discovery in Databases

Summary of Ph. D. T H E S I S

Katalin Tünde Jánosi-Rancz

The PhD School of Mathematics and Computer Scinece

Academic advisor:

Prof. Dr. Zoltán Kása

Department of Mathematics
Faculty of Mathematics and Computer Science
Babeş-Bolyai University
Cluj-Napoca, Romania
2014

Abstract

In this thesis, we propose to extract high-level conceptual knowledge from databases. Our main contribution is in the mining of different types of data dependencies (in relational and XML data) – we also focus on dependency management, inconsistency prevention and semantic data extraction, showing how our approaches contribute to conceptual knowledge discovery. We propose to study how Formal Concept Analysis (FCA) can offer a natural approach for discovering formal concepts in the data which are described in the form of formal context, discovering the data dependencies, and visualizing them by a single conceptual structure called the concept lattice.

First, we show how FCA can be used to characterize Functional Dependencies (FDs). The way in which we contribute to conceptual knowledge discovery has been shown in our approach to building inverted index files, in order to optimize the construction of the formal context of functional dependencies.

In the second part of this thesis, we introduced our approach to dependency management and inconsistency prevention. We have described our strategy, for finding Conditional Functional Dependencies (CFD) and Association Rules (AR), and instead of using them to clean dirty data we use them to prevent its appearance in the database. Our method helps the users to prevent inconsistencies, fix bugs and optimize their queries and applications by providing a lattice of dependencies, using usefulness as the relation.

We then present our tool which supports data extraction from various types of data sources. The tool we have created simplifies the extraction of data, provides the possibility of creating several extraction strategies and enables semantic searches for systems that use the tool. Results of experiments using our tool have proved the feasibility of our approach by enhancing extraction in terms of precision and speed.

List of publications

The present dissertation is primarily based on the following articles:

1. **K.T. Janosi-Rancz**, A. Lajos, Semantic Data Extraction, Accepted for The 8th International Conference INTER-ENG 2014, Interdisciplinarity in Engineering, 9 - 10 October 2014, "Petru Maior" University of Tîrgu Mureş, Romania

2. **K.T. Janosi-Rancz**, Finding, Managing and Inforcing CFDs and Ars via a semi-automatic learning strategy, 10th Joint Conference on Mathematics and Computer Science, May 21-25, Cluj Napoca, Romania, 2014

3. **K.T. Janosi-Rancz**, V. Varga, XML Schema Refinement Through Formal Concept Analysis, Studia Universitatis Babes-Bolyai, Informatica, Volume LVII, Number 3, 2012, 49-64

4. V. Varga, **K. T. Janosi Rancz**, C. Sacarea, K. Csioban, XML Design: an FCA Point of View, Proceedings of 2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Theta 17th edition, 2010, 165-170

5. **K. T. Janosi-Rancz**, V. Varga and T. Nagy, Detecting XML Functional Dependencies through Formal Concept Analysis, 14th East European Conference on Advances in Databases and Information Systems (ADBIS), Novi Sad, Serbia, Springer LNCS 6295, 2010, 595-598 **(Cited by 2)**

6. D. Ignatov, **K. T. Janosi-Rancz**, S. Kuznetsov, Towards a framework for near-duplicate detection in document collections based on closed sets of attributes. Acta Universitatis Sapientiae, Informatica, 2/2009, 215-233

7. **K. T. Janosi Rancz**, V. Varga, J. Puskas, A Soft Tool for Relational Database Design using Formal Concept Analysis, 7th Joint Conference on Mathematics and Computer Science, Studia Universitatis Babes-Bolyai, Informatica, vol. LIII, No. 2, 2008, 67-78 **(Cited by 1)**

8. **K. T. Janosi Rancz**, V. Varga, A method for mining functional dependencies in relational database design using FCA. Studia Universitatis Babes-Bolyai, Informatica, vol. LIII, No. 1, 2008, 17-28 **(Cited by 6)**

9. V.Varga, **K. T. Janosi Rancz**, A Software Tool to Transform Relational Databases in Order to Mine Functional Dependencies in it Using Formal Concept Analysis, Proceedings of the Sixth Int. Conf. on Concept Lattices and Their Applications (CLA), Olomouc, Czech Republic, 2008, 1-9 **(Cited by 2)**

Keywords: Knowledge Discovery, XML design, Formal Concept Analysis, Semantic Search, Semantic Set, Data Mining, Information Extraction, Dependency Mining, Association Rule, Dependency Management, Concept Lattice, Data Management, Optimization, Data Extraction, Information retrieval

Contents of the Thesis

Contents of Summary

1 Introduction

1.1 Conceptual Knowledge Discovery in Databases

With today's unprecedented increase in the generation and accumulation of online data, our reliance on databases to store that data has increased too. There is an urgent demand for new methods of addressing a long standing challenge: How do we efficiently identify, extract and present useful information contained within those databases? and ultimately; how do we present it in a way that is beneficial to the current and future needs of the end-user? Finding the answers to these challenges was our inspiration for developing the approaches that are presented in this thesis.

Knowledge Discovery in Databases (KDD) [17] is aimed at the development of methods, techniques, and tools that support human analysts in the overall process of deriving knowledge units that can be further used in solving real-world problems. Concepts are necessary for expressing human knowledge [40]. Recently, lattice theory has brought mathematical thinking for knowledge representation and discovery of knowledge. Formal Concept Analysis (FCA) [39; 14; 4; 5; 31; 22] offers a natural approach for discovering formal concepts in the data which are described in the form of formal context, discovering the data dependencies, and visualizing them by a single conceptual structure called the concept lattice. Graphical representation of concept lattices has proved highly useful in the discovery and understanding of conceptual relationships within a given set of data. The concept lattice has also proved itself useful for other applications relating to information and knowledge processing, such as visualization, data analysis, data mining and knowledge management. FCA provides

support for so-called Conceptual Knowledge Discovery in Databases (CKDD), which aims to support a human-centered process of discovering knowledge from data by visualizing and analyzing the conceptual structure of the data [19].

Our goal was to extract high-level conceptual knowledge from databases. Our main contribution is in the mining of different types of data dependencies (in relational and XML data) using FCA – we also focus on dependency management, inconsistency prevention and semantic data extraction. All our research is done in the field of CKDD.

1.2 Contributions and Structure of the Thesis

In this section we outline our main contributions and how they will be presented within the structure of this thesis.

Our main contribution concerns the mining of different types of data dependencies (in relational and XML data) using Formal Concept Analysis – we also focus on dependency management, inconsistency prevention and data extraction, showing how our approaches contribute to conceptual knowledge discovery.

The thesis is divided into 7 chapters and follows our research study chronologically.

In Chapter 1 we introduce our main contributions and how they are structured in this thesis.

In Chapter 2 we briefly summarise the main definitions that are used throughout this thesis.

In Chapter 3 we show how FCA can be used to characterize Functional Dependencies (FDs). A primary problem when using FCA to compute FDs is that datasets are generally many-valued, not binary. This means that the set of data must somehow be transformed to obtain a binary context, whose implications are equivalent to those dependencies. Unfortunately, this leads to a new data representation that is quadratic in the number of objects w.r.t. the original data, which does not allow this method to be applied on large datasets. This provided our motivation to create a method of building inverted index files in order to optimize the construction of the formal context of functional dependencies. Our method allows an equivalent characterization, but with better computational properties. Also in this chapter, we present our software (FCAFuncDepMine) which can be used in relational database design and for detecting functional dependencies in existing tables. This software can be considered as a conceptual knowledge discovery support environment, one that promotes a human-centered approach to dependency discovery processes and the representation of the resulting knowledge.

In Chapter 4 we propose a framework in which the XML document is parsed and the formal context, corresponding to the flat representation of the XML data, is constructed. It is then analysed in terms of knowledge discovery. The concept lattice obtained in this way is a useful graphical representation of the analyzed XML document's elements and their hierarchy. Also in this chapter, our software (FCAMineXFD) is introduced. This software

finds the keys and functional dependencies in XML data which are attribute implications in the constructed formal context. The scheme of the XML document is transformed in GTT-XNF using the detected functional dependencies.

In Chapter 5 we introduce our approach to dependency management and inconsistency prevention. We describe our strategy, which finds Conditional Functional Dependencies (CFDs) [15] and Association Rules (ARs)[1], and instead of using them to clean dirty data we use them to prevent its appearance in the database. We achieve this by differentiating Strict from Apparent dependencies. If a dependency is Strict, then we consider all possible exceptions to the dependency invalid, therefore upon validation, constraints are generated which prevent inserts and updates that do not comply with the validated dependency. The number of patterns might be very large, therefore we have introduced the More Useful (MU) poset and in each validation instance the supremums, that is, the most useful dependencies are taken into account. The poset of MU is extremely helpful, because it prevents flooding the database with valid dependencies that are the consequences of more useful dependencies. The approach of preventing the occurrence of dirty data in the database helps to maintain the consistency of the stored data. Along with complete management of dependencies our implemented application, DependencyManager, also uses Formal Concept Analysis methods to analyze the Strict dependencies and draw useful conclusions; thereby helping the users prevent inconsistencies, fix bugs and optimize their queries and applications by providing a lattice of dependencies, using usefulness as the relation.

In Chapter 6 we present our tool which supports data extraction from various types of data sources. Data extractors are generally difficult to maintain as they are required to cope with an infinite number of possible data structures and endless possible ways of describing the same thing. To simplify the difficult tasks of integrating new data sources and handling structural changes within existing data sources, we have implemented a new approach called *RK*. In *RK* the semantic tree generated by the semantic rules (used as input), describes a pattern that represents the structure of the data source. This tool is feasible as it is, but its maximum potential is attained, when collaboration is created between the owner of the data source and the extractor, the semantic rules can then be updated whenever the structure of the data source is changed. This would make the tool immune to structural changes. Enabling extraction of data semantically, would enable semantic searches based on the extracted data. Results of experiments using the *RK* tool have proved the feasibility of our approach by enhancing extraction in terms of precision and speed.

1.3 Previously Published Contributions

The results presented in this thesis have already been published. Our method of building inverted index files, to optimize the construction of the formal context of functional dependencies, has appeared in [36; 22]. Our software, FCAFuncDepMine, for mining functional dependencies that hold in a set of data using FCA, can be found in [26].

Our framework, which offers an interactive visualization for dependency exploration in XML data, can be found in [25; 21; 37]. Dependency management and inconsistency prevention are also studied by us in [24]. Our tool, for supporting data extraction from various types of data sources, has appeared in [23].

2

Theoretical Foundations

In the thesis in chapter 2 we briefly summarise the main definitions that are used throughout this thesis. First we will introduce Formal Concept Analysis as it is relevant to this thesis, then we recall basic concepts about data dependencies which are relevant to our work.

3

Functional Dependency (FD) Characterization with Formal Context

In this chapter we present our first contribution, which concerns on developing optimized approaches for FD mining using FCA. The results presented in this chapter have already been published in [36; 22; 26].

A primary problem when computing FDs with FCA is that datasets are generally many-valued, and not binary. This means that the set of data must be somehow transformed to obtain a binary context, whose implications are equivalent to those dependencies. Hereth in [18] offers a solution for this binarization. Unfortunately, in the method described in [18], the number of objects of the resulting context is quadratic w.r.t. the original, so this method cannot be applied to large datasets. This was our motivation for creating a method which builds inverted index files in order to optimize the construction of the formal context of functional dependencies. We aimed to discover conceptual knowledge but the primary goal was to minimize the input data before applying FCA. We optimized Hereths method by significantly reducing the size of the context file.

In the following we put forward our method for constructing the context of functional dependencies of a database table.

In the context creation procedure we exclude tuple pairs which are symmetric and reflexive to avoid redundancy. At the top of the concept lattice will be tuple pairs, in which there are no common values for the corresponding attributes, so we can omit generating these pairs of tuples. Pairs of form (t, t), where t is a tuple of the table, have all attributes in common, these objects will arrive in the bottom of the lattice, so they can be omitted too, because

they do not change the implications in the context. In order to optimize the construction of the formal context, we build inverted index files for the values of every attribute. With inverted indexes, the number of rows in the data file of formal context resulting from our method is half of the same value resulting from Hereth's method in [18]. In consequence, we can reduce the time taken to build the concept lattice for functional dependencies and we can eliminate useless dependencies.

3.1 Experimental Results

To evaluate our context creation method using inverted indexes we have conducted experiments on different datasets.
Consider the following poorly designed table:

```
StudMarks [StudID, StudName, GroupID, Email, SpecID,
 SpecName, Language, DiscID, DName, CreditNr, Mark]
```

This table contained 19500 rows and 11 columns; our method reduced the number of objects from 380,250,000 to 110,073,342 in the context file.

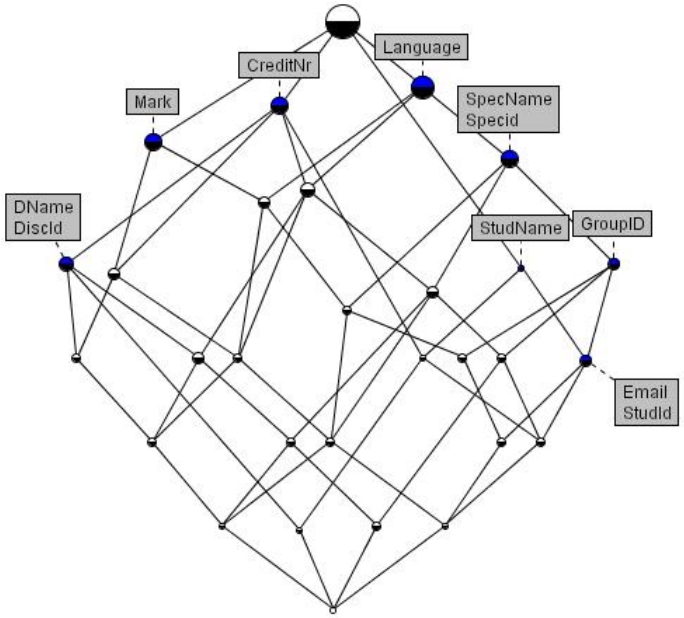

Fig. 3.1: Concept lattice for $FD\left(StudMarks, \overrightarrow{\mathbb{K}}(Uni)\right)$

We applied FCA over the reduced context; in Figure 3.1 is the resulted concept lattice for the $StudMarks$ table. The implications in this lattice, which are functional dependencies in the table, can be seen as follows: the concept with label $GroupID$ is a subconcept of concept with labels $SpecID$ and $SpecName$. This means, in every tuple pair where the $GroupID$ field has the same value, the specialization ID and name is the same. So we have the following implications, which are functional dependencies:

$$GroupID \rightarrow SpecID,\ SpecName$$

In the same manner the following functional dependencies can be read from the concept lattice:

$$DiscID \rightarrow CreditNr$$
$$DName \rightarrow CreditNr$$
$$SpecID \rightarrow Language$$
$$SpecName \rightarrow Language$$
$$StudID \rightarrow StudName$$
$$Email \rightarrow StudName$$

We also conducted experiments used of the well-known datasets from the UCI machine learning repository http://archive.ics.uci.edu/ml/, namely, the Wisconsin breast cancer (WBC) and Mushroom datasets. We can conclude that using inverted indexes the number of rows in the data file of formal context resulted by our method is half of the same value resulting from Hereth's method in [18]. Table 3.1 summarizes the performance of our method. Thus, we can reduce the time needed to build the concept lattice for functional dependencies and we can eliminate useless dependencies. We would like to emphasize that with context size optimization the quality of the resulted dependency set did not deteriorate; our method allows an equivalent characterization, but coming with better computational properties.

Dataset	Arity	No. Rows	No. of objects with our method	No. of objects with Hereth's method [18]
Mushroom	22	8,124	32,995,626	65,999,376
WBC	11	699	210,934	481,636
StudentMarks	11	19,500	110,073,342	380,250,000

Table 3.1: Experimental results: optimized context size

We have implemented the method presented in this section and completed it with a software tool which analyzes an existing relational database table. Our software, named FCAFuncDepMine, constructs an optimized formal context of functional dependencies. The software can be used in relational database design and for detecting functional dependencies in existing tables respectively. FCAFuncDepMine currently supports three types of input data file formats: XML, comma separated values (csv) and relational database tables. FCAFuncDepMine outputs the formal context in the widely used .cex format.

Optimisations are taking place in FCAFuncDepMine makes it worthy for converting large datasets into formal contexts. Our software can be considered as conceptual knowledge discovery support environment that promote human-centered dependency discovery processes and representations of their findings.

4

XML Functional Dependencies with FCA

As a second contribution we propose a framework which parses the XML document and constructs the Formal Context corresponding to the flat representation of the XML data. The results presented in this chapter have already been published. [25; 21; 37]. Our method is supported by a framework named FCAMineXFD, which finds the keys and functional dependencies in XML data, which are attribute implications in the constructed Formal Context. The obtained Concept Lattice is a useful graphical representation of the analyzed XML document's elements and their hierarchy. The scheme of the XML document is transformed in GTT-XNF [42] using the detected functional dependencies.

To achieve this, XML data must be converted into FCA formal contexts. Our software can analyze the whole XML document or a *tuple class* C_p given by the path p. XML data can be converted into a fully unnested relation, a single relational table, and existing FD discovery algorithms can be applied directly. Given an XML document, which contains the schema of the data at the beginning, we create generalized tree tuples from it.

As a first step, we need to define the objects and attributes of interest and create models of XML in terms of context.

- Choice of FCA Attributes: *PathEnd/ElementName*

  – for non-leaf level nodes the name of the attribute is constructed as: <ElementName>+"@key" and its value will be the associated key value. More elements, which have the same path, will have the same attribute name, but the values will be different.

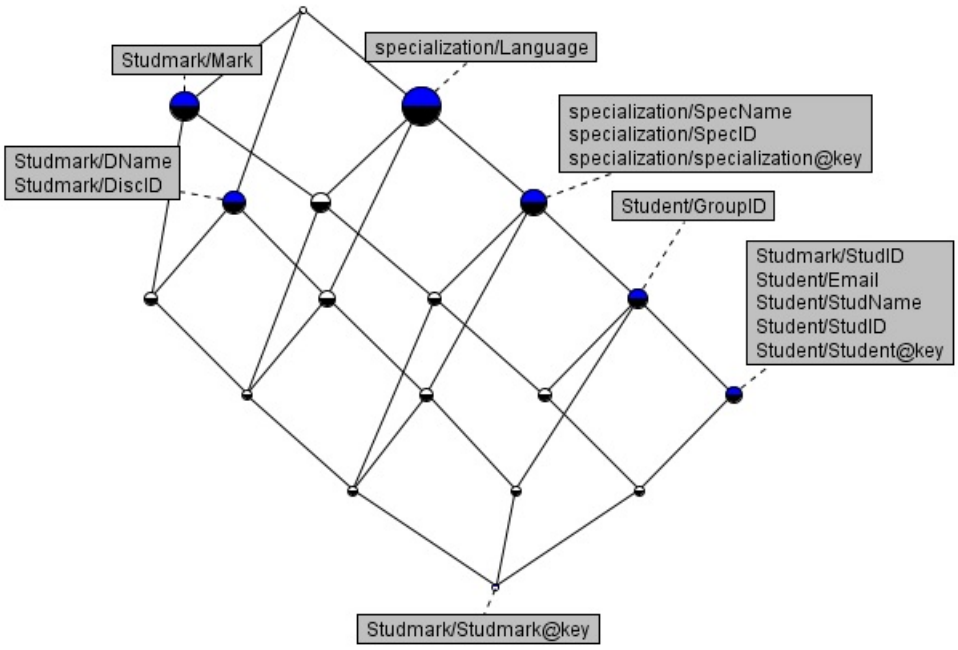

Fig. 4.1: Concept Lattice for tuple class $C_{specialization}$

  - the leaves (not the values of the leaves, but the element names of the leaves) of the tree tuple.

- Choice of Objects: the objects are considered to be the tree tuple pairs, actually the tuple pairs of the flat table. The key values associated to non-leaf elements and leaf element values are used in these tuple pairs.

The analyzed XML document may have a large number of tree tuples. We use the same optimization techniques as in previous chapter, we filter the tuple pairs and we leave out those pairs in which there are no common attributes, which does not alter the conceptual hierarchy.

Once the objects and attributes of the context are defined, we generate the concepts and create the concept lattice in the same way as we did in case of relation databases. A concept lattice for a university XML database (tuple class $C_{specialization}$) can be seen in Figure 4.1, we can interpret each concept and generate the list of all functional dependencies..

Example 4.1. *In concept node with label specialization/specialization@key, specialization/SpecID, specialization/SpecName the associated objects are tree tuple pairs, where the values for specialization/SpecID are the same. So we have the next XML FDs:*

$$\langle C_{specialization}, ./SpecID, ./SpecName \rangle$$

$$\langle C_{specialization}, ./SpecID, ./specialization@key \rangle$$

$$\langle C_{specialization}, ./specialization@key, ./SpecID \rangle$$

$$\langle C_{specialization}, ./specialization@key, ./SpecName\rangle$$

$$\langle C_{specialization}, ./SpecName, ./specialization@key\rangle$$

$$\langle C_{specialization}, ./SpecName, ./SpecID\rangle$$

From the concept lattice we can read the relationship between concepts. There are 1:n relationships, from *Specialization* to *Group*, from *Group* to *Students*, from *Students* to *Studmark*. *Disciplines* are in n:m relationship with *Students*, linked by *Studmark* node in this case.

In our approach, beside XFDs, we find the XML keys of a given XML document. The implications found by FCAMineXFD contain some FDs with RHS as ./@key values. These can be used to detect the *keys* in XML. This is a big step forward in FCA based knowledge discovery, because till now one could read only the dependencies from the resulting concept lattice, but using our method we can find the keys too.
There are two FDs of Example 4.1 with RHS as $./specialization@key$ in tuple class $C_{specialization}$, so the detected XML keys are:
$\langle C_{specialization}, ./SpecID\rangle$, $\langle C_{specialization}, ./SpecName\rangle$.

Given the set of keys and the set of dependencies discovered by our tool, we convert the XML schema into a correct one.

As a second contribution we proposed a framework for parsing the XML document and constructing the formal context, (corresponding to the flat representation of the XML data) and its analysis in terms of knowledge discovery. As we have noted, the concept lattice produced in this way is a valuable graphical representation of the analyzed XML document's elements and their hierarchy. Our software, FCAMineXFD, finds keys and functional dependencies within the XML data which are attribute implications in the constructed formal context. The scheme of the XML document is transformed in GTT-XNF using the detected functional dependencies.

5

Dependency Management and Inconsistency Prevention

In this chapter we describe our third contribution, which finds Conditional Functional Dependencies (CFDs) [15] and Association Rules (ARs)[1], and instead of using them to clean dirty data we use them to prevent their appearance in the database. We achieve this by differentiating Strict from Apparent dependencies. If we know about a dependency that it will be valid in the future, we can rely on them by creating constraints which guarantee that the CFD-rule will not be breached by insertions or modifications. Along with complete management of dependencies our implemented application called DependencyManager also uses Formal Concept Analysis methods to analyze the Strict dependencies and draw useful conclusions, helping the users of the application to prevent inconsistencies, fix bugs and optimize their queries and applications by providing a lattice of dependencies, using usefulness as the relation. Dependency management and inconsistency prevention are also studied by us in [24].

Our application learns FDs, CFDs and ARs from an arbitrary database. We can validate the learned dependencies by accepting or rejecting them, thus modifying its data from candidate to Strict(SD) or Apparent Dependencies(AD). We have created support for the simplification of the validation by using the relation of MU as the most useful candidates should be taken into account from all the other dependencies, therefore we drastically reduce the number of dependencies to be taken into account and we only store the most useful SDs because all the other dependencies are implied by them. The result of our research is a system which is able to prevent all the possible inconsistencies resulting from inserts or

updates which do not comply with the schema of the accepted SDs using constraints. We use FCA methods to analyze the strict CFDs/ARs and draw useful conclusions. In other words, we exploit FCA here to derive ontology containing concepts.

Our strategy is based on automatic learning, semi-automatic validation and automatic forgetting. We have defined an algorithm which is used by the application to learn new CFDs and ARs. To automatically accept all learned dependencies can be risky especially if the date is critical. Validation is not always easy for users, because there can be many dependencies learned, but we used the More Useful (MU) relation, which reduces significantly the difficulty of validation if used properly. As we mentioned earlier, the user sees a list of most useful CFDs/ARs, and can decide whether they are SDs or ADs. If the user accepts a dependency D_1, then all D_2 will instantly become invalid which meets the criteria of $D_1 \,\mathrm{MU}\, D_2$. As a result any D_2 less useful than D_1 is redundant with D_1. If the user rejects a CFD/AR D_1, then it will become invalid and all CFDs/ARs D_2 which meet the criteria of

$$(D_1 \,\mathrm{MU}\, D_2) \wedge (\nexists D_3 \text{ such that } D_1 \,\mathrm{MU}\, D_3 \,\mathrm{MU}\, D_2)$$

will be shown to the user, because they will become most useful ARs/CFDs. We estimated the benefit of using the MU relation. If a table has n columns and a dependency has a column for the condition, s columns in the determinant column set and d columns in the dependent column set, then there are $2^{(n-s-d-1)} + 2^d - 1$ possible dependencies which are less useful. If we *accept* the given dependency, we automatically $refuse$ all the less useful dependencies, which optimizes by helping the system/user with the automatic reduction of the dependencies to be taken into account by a maximum of $2^{(n-s-d-1)} + 2^d - 1$ dependencies. This method helps a lot in the validation and the usage of the dependencies. This is an exponential optimization.

5.1 Experimental Results

We evaluated the efficiency and effectiveness of our algorithm on four datasets. In datasets named Numbers1 and Numbers2 the rows were generated numbers, using formulas to guarantee the occurrence of CFDs and ARs. We also conducted experiments used real datasets from the UCI machine learning repository http://archive.ics.uci.edu/ml/, namely, the Wisconsin breast cancer (WBC). The Exceptions dataset was also generated by us using real data.

Table 5.1. describes the parameters of the used datasets, the MinOccurrenceFrequency and MinOccurrenceRate used in our experiments and the number of accepted/rejected dependencies: SDs and ADs for each dataset.

Based on our analysis, we managed to significantly reduce the number of dependencies to be considered, we found that only 10% -30% of dependencies were useful.

Dataset	Arity	No. Rows	MOF	MOR	SD	AD
Numbers1	6	20000	200	0.15	95	13
Numbers2	5	200000	650	0.2	118	11
WBC	11	699	37	0.06	242	21
Exceptions	5	11192	1000	0.1	43	8

Table 5.1: Experimental results (MOF = MinOccurrenceFrequency, MOR = MinOccurrenceRate)

If an SD pattern is accepted (validated), then a constraint prevents inserts and updates inconsistent with the accepted pattern. We quantified the amount of data protected by our system against inconsistency violation. In our dataset Numbers1 8620, Numbers2 102034, BCW 384, Exceptions 7083 presents the records complied to at list an SD and an arbitrary insert or update in the future has a probability of 43.1 %, 51.01 %, 54.93 % and 63.28% to be protected by our system against inconsistencies violating our SDs. If an error occurs because of mishandling data where an SD is applicable, then the problem can be identified in the error logs and potentially, bugs can be found and fixed.

In DependencyManager we have implemented an FCA module, which studies the properties of SDs discovered with our application and draw interesting conclusions, further knowledge about them. Our approach was agnostic towards the nature of the content of the database; we have implemented an FCA applicable to accepted dependencies of any database using our system. First, we transformed existing knowledge into an FCA context based on SDs. In this context the objects are the SDs and the attributes are possible properties for the objects. The attributes are Weakness, Determinant Column Cluster's Size, Column Cluster's Size, Frequency of Occurrence and Almost Symmetrical. We have defined the Fuzzy sets of None, Very Low, Low, Medium, High, Very High and Total for this attribute. Second, we created the concepts, (not only by grouping objects which have the same set of attributes, but also comparing their value too), and then we transformed them into a concept lattice (without binarization), which was the basis for further analysis. In other words, we exploit FCA here to derive ontology containing concepts. FCA is relevant in this study, by providing conceptual information as the result of SD-analysis with the attributes described previously. To our knowledge no previous work has utilised FCA implementation where the input objects were SDs and the input attributes were properties of these dependencies.

Along with complete management of dependencies our strategy prevents the appearance of inconsistencies instead of cleaning already existent ones. We can conclude that our application is scalable, easy to use and powerful in preventing inconsistencies. Our algorithm represents a new approach to CFD-handling and a novelty for FCA analysis, by analyzing abstract database patterns.

6

Data Extraction using Semantic Trees

A major finding of this research is a fast and precise tool which supports semantic data extraction from various types of data sources. To simplify the difficult tasks of integrating new data sources and handling structural changes within existing data sources, we have implemented a new approach called *RK*. *RK* is a tool which serves as a guide for extraction purposes; the semantic tree generated by the semantic rules (used as input), describes a pattern that represents the structure of the data source.

The aim of our research is to simplify the labour-intensive process of data extractor creation. We intend to motivate owners of data sources to assist data extractor teams in the task of extracting data from their data sources. To achieve this goal, we have created *RK*, a tool which utilises an interpreted semantic language to define the semantic rules that map the concepts, including their relative structural path, parent-child relationship, plurality, set of synonyms/keywords, priority and other attributes describing the concept. We use the term *concept* here to denote all rules having the same keyword. For example, if the information to be extracted relates to the real-estate sector, then possible concepts might include *size* or *price*. From these rules *RK* generates objects serving as nodes when building the semantic tree. The tree describes the structural-semantic pattern used as a guide for the extraction functionalities. These methods, along with navigational functionalities are defined for general purposes, however they can be customized, and in the case of navigational functions, overwritten if needed. Our system supports data extraction from hierarchical-structured data sources, such as web-pages, among others. The main benefits of our proposal include the potential to support conceptual searches and the possibility of structural changes without the need for changing data extractor applications that mine from data sources whose

structure has changed.

Our tool supports portability, customizability and conceptual, semantic extraction. The current implementation can be integrated into web-pages and serves the purpose of giving instructions for crawlers. The extractor, after checking to see if *RK* is integrated to the data source, can extract the whole data set by calling a single function.

To support the definition of semantic rules we have developed the *RK* semantic language. Each rule represents a concept and is aware of its parent or ancestor concept, its relative location to the structural position of its parent or descendant concept and the relative structural location of the value of the concept. Polymorphism is supported, as the same concept type can take several possible shapes in the structure. This means that several rules can describe the same concept to handle the different cases of its occurrence. In such cases they are identified by indexes. A concept might have several potential parent or ancestor. Synonyms are supported because of cultural divergence. There is no standard ontology; concept names differ from culture to culture and many concepts have different names with-in the same culture. The tool respects cultural diversity by providing the possibility to define synonyms, that is, different keywords for the same concept.

The rules are stored in .rk files. Customizing the extraction is possible by their combination. We call such a combination a search type. Ideally these rules are created and maintained by the owners of the data source, but they can be defined by anybody who has access to the *.rk* files or they can be generated automatically by a module. Extraction works based on structural-semantic rules. If advertisements are not considered to be relevant content, then no semantic rules are defined to support them, so the extractors will not even consider these parts.

The rules are converted to objects. The parent-child, ancestor-descendant relation of the rules implies a hierarchy. By creating an abstract root node, which has all the concepts with unspecified predecessors as children a tree is defined, where all the nodes correspond to rules and the vertices are the predecessor-successor relation. The semantic tree describes the structural-semantic work-plan for the extractor. Structural modifications have always been difficult to handle with extractors. If one uses our tool, then in the case of structural modifications, only the semantic rules have to be maintained. The extractors using the data source will not feel the effects of this change (which they otherwise would do). Instead they can extract data as if there was no structural modification.

As we have seen, the semantic tree describes the data structure. Because of the hierarchical structure of the semantic tree, the extractor can repeatedly narrow down the search-space, because a concept structurally holds its sub-concepts, thus when searching for the sub-concepts of a concept, the structural representation of the concept can serve as root of the search. The extraction task is solved by simplifying the search-space repeatedly. If a concept is not plural, then after the first instance is found inside its parent or ancestor, no other instances will be searched for. If it is plural, then a full search is done, searching for all the occurrences. With our tool extraction is not more complicated than a function call.

The tool can be integrated into a data-source by copying its source-code and creating and publishing the .rk files. In some cases the navigational functions must be overwritten.

6.1 Experimental Results

We conducted experiments on different real life websites. The extractor, after checking to see if *RK* is integrated to the data source, extracted the whole data set by calling a single function. An additional function was used to move to the next page.

	No. of rules	No. of Extracted Objects	Total Download Time (millisec)	No. of Objects/ Page	Total Extraction Time (millisec)	Extraction Time/ Page (millisec)	Extraction Time/ Object (millisec)
Test RealEstate grid page	17	11376	513748	10	38372	33.72	3.37
Test RealEstate details page	21	11376	7566787	10	265288	233.12	23.32
www.boatsandoutboards.co.uk	6	7721	2586972	20	55070	148.84	7.13
http://prep-adm.cloudapp.net/	13	160012	6840503	20	521756	65.13	3.25

Table 6.1: Experimental Results

Table 6.1. summarizes the results of the experiment. All experiments were conducted on a machine equipped with a Pentium Dual-Core CPU T4300 working at 2.1 GHz , with 4 GB of RAM.

One of the tested website was *www.boatsandoutboards.co.uk*. from which we extracted *Image*, *Link*, *Title*, *Properties*, and *Description* for boats. In this case the data source contained 7721 boats or accessories and the average extraction time/object was 7.13 milliseconds. During the extraction process, our tool does not perform unnecessary operations, therefore the speed of the extraction is optimal. Instead of evaluating hundreds or even thousands of regular expressions (like in [43]) within the whole web-page it evaluates only a few dozen of structural-semantic rules, builds a semantic-tree and then uses that to extract all (and only) the required data.

Another website we have tested our extraction tool on was: $http://prep-adm.cloudapp.net$, which contains the results of final exams for all Romanian pupils leaving elementary school. From this webpage we successfully extracted the *County*, *School*, the *Name* and grades for 160012 pupils. For this experiment we defined 13 structural-semantic rules.

The average extraction time per page was 65.13 milliseconds, which corresponds to 3.25 milliseconds per student. This website offers the possibility to make searches on it, but if one wants to analyse the all educational system in Romania then they need all the data from this webpage, which is unreachable. Our method offers a solution to this by extracting semantically all the desired data, allowing further data analysis to be applied.

We do not report precision and recall computed over the quality of extracted data, since we did not register any loss in our experiments. The correctness of extraction was 100% as long as the structural-semantic rules were defined correctly. The time of the extraction was

equal to the cumulated time needed for building up the semantic tree plus extracting all the desired data using jQuery, less than 10 milliseconds on average, (on a home computer having the parameters described previously). Of course, we could use a dedicated super server and multithreading to achieve even better results.

We can conclude that the tool we have created simplifies the extraction of data, provides the possibility of creating several extraction strategies and enables semantic searches for systems that use the tool. The extraction is optimal because with correct semantic rules the algorithm repeatedly narrows down the search-space to increase search-performance. Instead of text analysis, we have provided the alternative of hierarchical searches which consistently work on the sub-tree of a particular concept to find the values of the sub-concepts based on the semantic tree built from the rule-set. With correct rules the extracted data should be correct, and if performed in an environment of collaboration, precision can be maintained in the long term. Results of experiments using the *RK* tool have proved the feasibility of our approach by enhancing extraction in terms of precision and speed.

7

Future Work

In this thesis we have shown how FCA can be used to characterize Functional Dependencies and mine conceptual knowledge from databases. In the future we aim to characterize functional dependencies using the formalism of pattern structures [5], an extension of classical FCA to handle complex data. We also propose to develop a software which will build the *tricontext* of an XML tree and will give the functional dependencies from XML tree. On the other hand, we would like to mine, in terms of FCA, other different types of dependencies that may hold in a given set of data (not only in relational and XML, but in NoSQL databases too).

In Chapter 5 we introduced DependencyManager, our approach to dependency management and inconsistency prevention. In the future we intend to make this strategy even more useful, we intend to find and analyze cross-table SD-candidates. We also want to generalize the set of conditions by using more columns in the boolean functions instead of only one and using more operator types, (our current implementation considers equality as the only conditional operator). Automatization of the validation process would be useful; this feature will probably be based on user-defined rules which will help the learner module to automatically determine whether an SD-candidate is an AD or an SD. The system stores the errors originating from unsuccessful inserts and updates which were prevented by the constraints created for SDs; this will generate knowledge which can be used to determine the cause of failure of inserts and updates. If the cause of failure is an incorrectly accepted pattern, the pattern should be dismissed; otherwise the incorrect user action or incorrect functionality can be detected.

Known SDs can be used to increase application performances. As a result, backup storage

space/performance can be optimized by creating templates instead of storing all instances of redundant data matching the valid patterns. Valid patterns can be cached, enabling client-machines to deduce values and thus reduce the number of requests to databases. Therefore, the size of the responses of the databases can be decreased, optimizing the communication between client machines, application servers and database servers.

In Chapter 6 we introduced our *RK* tool, which supports data extraction from various types of data-sources. Currently in *RK* the set of rules are easy to define, but the rules must be typed by hand, because no automatic rule-definition module has been created yet. In the future we plan to create browser add-ons and desktop applications which support graphical rule definitions, generating the RK rules based on the actions of the user. The Semantic Tree Builder reads the rules from the set of rules and creates a semantic tree, creating an abstract root node and other nodes, forming a tree. The abstract root node exists to guarantee that the data structure will be a tree, while the other ones represent rules. The relations between the nodes form the vertices of the tree. Currently there is no guarantee that the graph will not contain cycles, which means that in the case of a developer accidentally creating a cycle, the system does not prevent it. In the future we plan to create a validator for the semantic tree, which would consider the tree to be valid if it is really a tree and does not contain cycles. The Extractor module of RK mines the textual representation of the concepts located in the semantic tree. We plan to implement an upper layer, which will use the Extractor and will support the extraction of other types of meta-data, such as sound, video, image or other types of files based on the textual data mined by the Extractor. The Extractor does not support the parsing of textual data; the upper layer will support textual parsing too. The ideas behind the Semantic Tree Builder, Extractor and Navigator modules as a whole are technology-agnostic; they support the extraction from any data source with hierarchical data-structure, (our current implementation only supports the extraction from web-pages). We plan to support other types of data sources as well. Synonym mapping supports a multicultural extractor system, which helps the data miners to extract data from multiple data sources that differ in their language and word usage.

Bibliography

[1] R. Agrawal, R. Srikant. Fast algorithms for mining association rules in large databases, In VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994, 487–499

[2] S. Andrews. In-Close, a Fast Algorithm for Computing Formal Concepts, In Rudolph, Dau, Kuznetsov (Eds.), Supplementary Proc. of ICCS'09, CEUR WS 483, 2009

[3] M. Arenas, L. Libkin, W. Fan. On the Complexity of Verifying Consistency of XML Specifications, SIAM J. Comput. 38(3), 2008, 841–880

[4] J. Baixeries. A Formal Concept Analysis framework to model functional dependencies. Mathematical Methods for Learning, 2004

[5] J. Baixeries, M. Kaytoue, A. Napoli. Characterizing Functional Dependencies in Formal Concept Analysis with Pattern Structures, Annals of Mathematics and Artificial Intelligence, 2014

[6] P. Bohannon, W. Fan, F. Geerts, X. Jia, A. Kementsietsidis. Conditional functional dependencies for data cleaning, In ICDE, IEEE, 2007, 746–755

[7] P. V. Borza, O. Sabou, C. Sacarea. OpenFCA, an open source formal concept analysis toolbox, IEEE International Conference on Automation Quality and Testing Robotics AQTR, Vol 3, 2010, 1–5

[8] A. Buzmakov, S. O. Kuznetsov, A. Napoli. Scalable Estimates of Concept Stability, ICFCA, 2014, 157–172

[9] C. H. Chang, M. Kayed, M. R. Girgis, K. F. Shaalan, A survey of web information extraction systems, IEEE Transactions on Knowledge and Data Engineering, 18(10), 2006, 1411–1427

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

[11] G. Cong, W. Fan, F. Geerts, X. Jia, S. Ma. Improving data quality: Consistency and accuracy, VLDB, ACM, 2007, 315–326

[12] V. Crescenzi, G. Mecca, P. Merialdo. Automatic web information extraction in the roadrunner system, In Revised Papers from the HUMACS, DASWIS, ECOMO, and DAMA on ER 2001 Workshops, 2002, 264–277

[13] V. Crescenzi, D. Qiu, P. Merialdo. A Framework for Learning Web Wrappers from the Crowd, Proceedings of the 22nd WWW Conference, 2013, 261–272

[14] J. Demetrovics, L. Libkin, I. Muchnik. Functional Dependencies in Relational Databases: a Lattice Point of View, Discrete Applied Mathematics, Volume 40, Issue 2, 1992, 155–185

[15] W. Fan, F. Geerts, X. Jia, A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies, ACM Trans. Database Syst., 33(2), 2008

[16] W. Fan, F. Geerts, J. Li, M. Xiong. Discovering conditional functional dependencies, IEEE Trans. Knowl. Data Eng., 23(5):2011, 683–698

[17] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth. From Data Mining to Knowledge Discovery in Databases, Advances in knowledge discovery and data mining. American Association for Artificial Intelligence, 1996, 1 – 34

[18] J. Hereth. Relational Scaling and Databases, Proceedings of the 10th International Conference on Conceptual Structures: Integration and Interfaces LNCS 2393, Springer Verlag, 2002, 62–76

[19] J. Hereth, G. Stumme, R. Wille, U. Wille. Conceptual Knowledge Discovery - a Human-Centered Approach, Journal of Applied Artificial Intelligence, 17:2003, 281–301

[20] D. Ignatov, K. T. Janosi-Rancz, S. Kuznetsov. Towards a framework for near-duplicate detection in document collections based on closed sets of attributes, Acta Universitatis Sapientiae, Informatica, 2, 2009, 215–233

[21] K. T. Janosi-Rancz, V. Varga, T. Nagy Detecting XML Functional Dependencies through Formal Concept Analysis, 14th ADBIS Conference, LNCS 6295, Springer, 2010, 595–598

[22] K. T. Janosi-Rancz, V. Varga. A method for mining functional dependencies in relational database design using FCA, Studia Universitatis Babes-Bolyai, Informatica, vol. LIII, No. 1, 2008, 17–28

[23] K.T. Janosi-Rancz, A. Lajos. Semantic Data Extraction, Accepted for The 8th International Conference INTER-ENG 2014, Interdisciplinarity in Engineering, 9 - 10 October 2014, "Petru Maior" University of Tîrgu Mureş, Romania

[24] K.T. Janosi-Rancz. Finding, Managing and Inforcing CFDs and Ars via a semi-automatic learning strategy, 10th Joint Conference on Mathematics and Computer Science, May 21-25, Cluj Napoca, Romania, Studia Universitatis Babes-Bolyai, Informatica, 2014

[25] K.T. Janosi-Rancz, V. Varga. XML Schema Refinement Through Formal Concept Analysis, Studia Universitatis Babes-Bolyai, Informatica, Volume LVII, Number 3, 2012, 49–64

[26] K. T. Janosi-Rancz, V. Varga, J. Puskas, A Soft Tool for Relational Database Design using Formal Concept Analysis, 7th Joint Conference on Mathematics and Computer Science Cluj, 3-6 July 2008, Studia Universitatis Babes-Bolyai, Informatica, vol. LIII, No. 2:2008, 67–78

[27] M. Kaytoue. Mining numerical data with formal concept analysis and pattern structures (PhD Thesis), Université Henri Poincaré - Nancy, 2011

[28] S. Kolahi, L. Libkin. XML design for relational storage, In Proceedings of the 16th International World Wide Web Conference, 2007, 1083–1092

[29] S.O. Kuznetsov, S. A. Obiedkov. Comparing performance of algorithms for generating concept lattices, Journal of Experimental and Theoretical Artificial Intelligence, Vol. 14, Nos 2-3, 2002, 189–216

[30] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, J. S. Teixeira. A brief survey of web data extraction tools, SIGMOD Rec. 31, 2/2002, 84–93

[31] S. Lopes, J. M. Petit, L. Lakhal. Functional and approximate dependency mining: database and fca points of view, J. Exp. Theor. Artif. Intell., 14(2-3):, 2002, 93–114

[32] R. Medina, L. Nourine. Conditional functional dependencies: An fca point of view, In L. Kwuida and B. Sertkaya, editors, ICFCA, volume 5986 of LNCS, Springer, 2010, 161–176

[33] J. Poelmans, D. I. Ignatov, S. O. Kuznetsov, G. Dedene: Formal concept analysis in knowledge processing: A survey on applications, Expert Syst. Appl. 40(16):2013, 6538–6560

[34] U. Priss. Formal Concept Analysis in Information Science, In Cronin, B. (ed.) Annual Review of Information Science and Technology, ASIST, vol. 40, 2008

[35] I. N. Md. Shaharanee, F. Hadzic, T. S. Dillon. Interestingness measures for association rules based on statistical validity, Knowledge-Based Systems, Volume 24, Issue 3, 2011, 386–392

[36] V. Varga, K. T. Janosi Rancz. A Software Tool to Transform Relational Databases in Order to Mine Functional Dependencies in it Using Formal Concept Analysis, Proceedings of the Sixth International Conference on Concept Lattices and Their Applications, Olomouc, Czech Republic, 2008, 1–9

[37] V. Varga, K. T. Janosi Rancz, C. Sacarea, K. Csioban, XML Design: an FCA Point of View, Proceedings of 2010 IEEE International Conference on Automation, Quality and Testing, Robotics, Theta 17th edition, 2010, 165–170

[38] M. W. Vincent, J. Liu. Functional dependencies for XML, In Proceedings of the 5th Asian-Pacific Web Conference, 2003, 22–34

[39] B. Ganter and R. Wille. Formal Concept Analysis: Mathematical Foundations, Springer, 1999

[40] R. Wille. Why can concept lattices support knowledge discovery in databases? Journal of Experimental and Theoretical Artificial Intelligence, 14(2-3), 2004, 81–92

[41] T. L. Wong, W. Lam. Unsupervised Extraction of Popular Product Attributes from Web Sites, 8th Asia Information Retrieval Societies Conference, 2012, 437–446

[42] C. Yu, H. V. Jagadish. XML schema refinement through redundancy detection and normalization, VLDB J. 17(2):2008, 203–223

[43] X. Zheng, Y. Gu, Y. Li. Data extraction from web pages based on structural-semantic entropy, In Proceedings of the 21st WWW Conference, 2012, 93–102