Formal Concept Analysis support for Conceptual Abstraction in Database Reengineering

Carmen Hernández, Félix Prieto, Miguel A. Laguna, Yania Crespo

Computer Science Dept., Universidad de Valladolid, Spain
Phone:+(34)983423670; Fax: +(34)983423671
{chernan, felix, mlaguna, yania}@infor.uva.es

Abstract. This paper presents a proposal that aims to cover some tasks required in the *Database Reengineering* process, mainly in the *Conceptual Abstraction* phase. The principal tasks are: the transformation of enriched logical schema to object-oriented schema, support for process iterations and support for data migration from the legacy database to the new database. The proposal consists of attaining the listed tasks by applying *Formal Concept Analysis* (*FCA*) techniques. It could be said that the advantage of using this proposal is that it is possible to build a tool, based on *FCA*, automatic, graphic and interactive. This tool will be able to support the process iterations, as it will be possible to mark the impact zones of later modifications in the logical schema and to verify how they affect the conceptual scheme obtained. This paper detailed presents the use of *FCA* in the *Conceptual Abstraction* phase. We obtain a class diagram that fits the object model defined for UML and which reflects the underlying model in the original logical schema. In the class diagram, we will have the significant classes of the problem, organized in the specialization/generalization hierarchy and related by means of associations (as a general framework for the rest of the UML relations, e.g. aggregations). The establishment of the associations can give rise to the appearance of new classes as association classes that characterize the defined relation.

Keywords: Database Reengineering, Conceptual Abstraction, Conceptual Schema Migration, Formal Concept Analysis

1 Introduction

The majority of the tasks that must be carried out during software maintenance and evolution are arised because it is necessary to support the new requirements that appear in the organizations, or because the existing ones need to be modified. If the system maintains and processes a great amount of persistent data, a question that has to be tackled is to determine what information should be stored and maintained and how this information can be used in different contexts, according to the variation in the requirements. Some examples of requirements that have produced or will produce massive changes in software, from the point of view of the data, are the Year-2000

problem, the Euro-conversion problem in Europe or the migration of information systems to the Web and electronic commerce. Similarly, the growing use of *Data Warehouses* and *Data Mining* techniques make the incorporation of information from legacy data systems to the *data warehouse* necessary. Then, a consistent transformation of the legacy data structure to the common business data model is needed.

The first requirement in to achieve these processes is to have as much as possible of the documentation of the data structure (conceptual, logical and physical) present in the *Legacy Database* (LDB) and the *Legacy Software System* (LSS). Unfortunately, in most existing LSS, the corresponding documentation is either missing, obsolete or inconsistent. The task of *Database Reverse Engineering* is to recover such information. It is then necessary to redesign the conceptual schema, to transform the LDB into a *New Database* (NDB) and the LSS into a *New Software System* (NSS) and to deal with the data migration from the LDB to the NDB. This complete process is called *Database Reengineering*. Figure 1 summarizes the main phases of *Database Reengineering.*

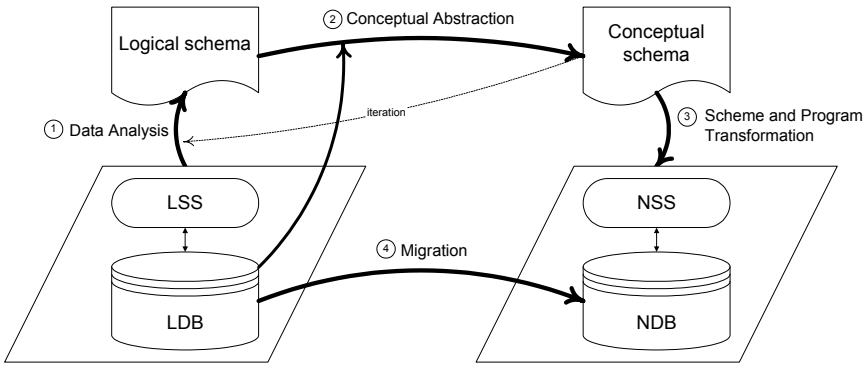

Figure 1. Main phases of database reengineering

In this paper, a proposal is presented that allows some of the tasks carried out in the *Conceptual Abstraction* phase of the *Database Reengineering* process to be automated. This proposal consists of carrying out the transformation of the enriched logical schema to an object-oriented schema, providing support to the iterations inherent to this process and performing the data migration from the LDB to the NDB that has to be built. All these by applying a mathematical technique of information organization, *Formal Concept Analysis* (*FCA*). The rest of the paper is organized as follows: Section 2 delimits the working context in the *Database Reengineering* framework, presents the problem and introduces a proposal to deal with it. Section 3 introduces the *FCA* concepts. The proposal is explained in Section 4 and an example of what can be achieved with this strategy is given. Related work is summarized in Section 5 and, finally, Section 6 gives the conclusions and identifies future lines of work.

2 Context

The *Database Reverse Engineering* process is made up of two main activities, namely *Data Analysis* and *Conceptual Abstraction* [Müller et al., 2000].
The *Data Analysis* activity aims to recover an updated logical data schema that is structurally complete with well-identified and documented semantics. In the case of a relational database, the structural completion activity consists of detecting all the candidate and foreign keys which are not declared explicitly in the database catalogue, in addition to the inclusion dependencies [Elmasri & Navathe, 1994]. On the other hand, the semantic enrichment aims to classify the schema elements according to higher level abstractions such as inheritance and aggregation [Jahnke & Wadsack, 1999].
In general, *Data Analysis* is an exploratory activity with a strong human component that requires a great deal of experience and skill. When dealing with this activity it would be wrong to assume that the data structure being used fits in with good design properties. There are examples that show that LDB designs have many deficiencies and contain numerous errors. This means that the logical schema obtained has many gaps and a lack of precision that will surely appear as the process advances, but which will not be detected in a first phase [Blaha & Premerlani, 1995]. This enriched logical schema with all the semantic information, that could be recovered in this *Data Analysis* activity, is the input to the second activity.
The *Conceptual Abstraction* aims to transform the logical schema derived from the *Data analysis* into an equivalent conceptual schema. The habitual design representations are the Entity-Relationship Model or the Object-Oriented Model. Both provide a sufficient level of abstraction to deal with the subsequent *Reengineering* activities.
The *Database Reengineering* process is not adjusted to a waterfall-model, in which the *Conceptual Abstraction* activity does not begin until the *Data Analysis* has finished [Jahnke & Wadsack, 1999]. Normally, *reengineers* start with the *Forward Engineering* activities when the logical schema is still incomplete or inconsistent. Further knowledge of the LDB structure will surely be accumulated during the development of these activities. That will allow the steps of the *Reengineering* process to be retraced, so that the initial logical schema can gradually be completed.
It is thus necessary to define a process for obtaining and transforming these schemas that can be automated, interactive and iterative, so that the most convenient conceptual schema can finally be obtained and also, a method to support the automatic LDB data migration to the NDB.

2.1 Problem

This paper is centered in the *Conceptual Abstraction* activity, when a first semantically enriched logical schema has already been obtained, the following task is to produce an equivalent conceptual schema. The paper will focus on this important *Database Reengineering* activity, also called *Conceptual Schema Migration*.
Conceptual Schema Migration aims to produce an abstract design for an LDB schema. High-level modeling concepts such as objects, aggregation, and inheritance

are used in this activity. The resulting conceptual schema provides a level of abstraction that facilitates an understanding of an LDB's static structure. Furthermore, it is a prerequisite to achieving a very important maintenance goal, i.e.: integration with emerging technologies such as object-orientation, Internet, and Client-Server architectures.

Several approaches have been presented to carry out this transformation, [Ramanathan & Hodges, 1997], [Behm et al., 1997], [Blaha & Premerlani, 1996], [Jahnke, 1999], [Hainaut et al., 1993], some of which are commented on in Section 5 (Related Work). The most common is to start from a logical relational schema and try to obtain a schema that fits the standard ODMG for object-oriented databases [Cattell et al., 2000]. However, it is also necessary to consider that the *Database Reengineering* process has an explorative and iterative character. Whenever information about the logical schema is revised, the consistency with the conceptual schema that has already been created may be lost. There are different options when the consistency maintenance process is planned:

- To begin the redesign process from the beginning, losing the conceptual schema modifications that had already been made.
- To manually determine the impact of the modifications on the conceptual schema that had already been obtained.

Neither of the two options is satisfactory: the first requires the repetition of operations already done; the second can lead to errors.

A third option, [Jahnke, 1999], proposes an incremental mechanism for reestablishing the consistency between both models. The tool's environment is capable of determining those redesign operations that must be undone and those where it is not necessary.

Our proposal support the *Database Reengineering* process using *FCA*, a mathematical information organization technique. This technique will play multiple roles in the *Database Reengineering* process.

1) In the *Data Analysis* phase: assistance to find out candidate and foreign keys.
2) In the *Conceptual Abstraction* phase: assistance to obtain the new conceptual schema from two points of view:
   a) obtaining a class diagram which reflects the original logical schema,
   b) supporting the process iterations, keeping track of the relation between the original relational schema and the new object-oriented schema. This is going to be done in such a way that if the relational schema changes, the object-oriented schema also changes, as far as it will be possible. If changes can't be applied, some transformations done in previous step must be undone. This is on the line of [Jahnke, 1999].
3) In the *Schema Transformation* and *Migration* phases: assistance indicating the transformations that must be achieved.

In this paper, we are going to present with details how 2a) is accomplished.

The following section introduces the *FCA* concepts needed for understanding this paper.

3 Introducing Formal Concept Analysis

Formal Concept Analysis (*FCA*) was introduced by Wille [Wille, 1982] and is completely developed in [Ganter & Wille, 1999]. It is a mathematical technique that allows the underlying abstractions in a data table, formally a context, to be shown by means of the construction of the concept lattice, also known as the Galois lattice, associated to it. The *FCA* has been used in works related to symbolic data analysis and knowledge representation [Godin et al., 1995]. It has also been used in themes related to Software Engineering, both for the study of inheritance [Godin & Mili, 1993], and the study of the way in which the class characteristics in a hierarchy are used [Snelting & Tip, 1996]. Both approaches have been combined to assist the construction processes of Domain Frameworks [Prieto et al., 2002]. Outside the object-oriented paradigm, these techniques have also been applied to module identification within monolithic code [Siff and Reps, 1999]. It has also been used in the area of databases [Stumme et al., 1998], [Schmitt & Conrad, 1998]. This will be dealt with in more detail in Section 5 (Related work). We shall begin by introducing the basic notions defined by Wille.

Definition Let us call a tupla of sets, (G,M,I), that verify $I \subseteq (G \times M)$, *formal context*.

G is usually called a set of *objects* and M a set of *attributes*. The binary relation I gives the *incidence* of the set of attributes on the set of objects. It is then possible to define the following applications, in whose definitions the notions, respectively, of the set of attributes that certain objects possess, and the set of objects that certain attributes possess, can be seen:

$$\begin{array}{lccl} \varphi: & \wp(M) & \rightarrow & \wp(G) \\ & A & \mapsto & A^{\uparrow} = \{m \in A \mid (g,m) \in I, \forall g \in G\} \\ \psi: & \wp(G) & \rightarrow & \wp(M) \\ & B & \mapsto & B^{\downarrow} = \{g \in B \mid (g,m) \in I, \forall m \in M\} \end{array}$$

For the elements $g \in G, m \in M$, we shall use the notation $g^{\uparrow}$ and $m^{\downarrow}$ instead of $\{g\}^{\uparrow}$ and $\{m\}^{\downarrow}$ when there will be no resulting confusion.

These two definitions allow the following definition to be made, that reflects the informal notion of concept as a set of objects and attributes that are mutually determined.

Definition Let us call a pair $(A,B) \in \wp(G) \times \wp(M)$ that verifies $A^{\uparrow} = B$ and $B^{\downarrow} = A$, a *formal concept*. Normally, the first set in the pair will be called the concept *extent* and the second, the concept *intent*. The set of formal concepts associated to a context (G,M,I) will be denoted as $\mathbf{G}(G,M,I)$.

On $\mathbf{G}(G,M,I)$ a partial order relation can be defined through the following formula where $(A,B),(A',B') \in \mathbf{G}(G,M,I)$:

$$(A,B) \leq (A',B') \Leftrightarrow A \subseteq A' (\Leftrightarrow B \supseteq B')$$

From this definition, the following result can now be proved:

Theorem (fundamental for concept lattices): The set $\mathbf{G}(G,M,I)$ with the defined partial order relation forms a complete lattice in which the lowest and highest are given by the following formulas where T denotes a set of indices, not necessarily finite, and $\forall t \in T, (A_t,B_t) \in \mathbf{G}(G,M,I)$:

$$\bigwedge_{t\in T}(A_t,B_t) = \left(\bigcap_{t\in T} A_t, \left(\bigcup_{t\in T} B_t \right)^{\downarrow\uparrow} \right)$$

$$\bigvee_{t\in T}(A_t,B_t) = \left(\left(\bigcup_{t\in T} A_t \right)^{\uparrow\downarrow}, \bigcap_{t\in T} B_t \right)$$

The existence of the lowest and highest for any set of concepts allows the following functions to be defined:

$$\begin{aligned} \gamma: \; G &\rightarrow \mathbf{G}(G,M,I) \\ g &\mapsto \bigwedge_{\{(A,B)\in(G,M,I)\mid g\in A\}} (A,B) \\ \mu: \; M &\rightarrow \mathbf{G}(G,M,I) \\ m &\mapsto \bigvee_{\{(A,B)\in(G,M,I)\mid m\in B\}} (A,B) \end{aligned}$$

It is thus easy to show that these functions admit a much simpler notation, as follows:

$$\forall g \in G, \gamma(g) = (g^{\uparrow\downarrow}, g^{\uparrow})$$
$$\forall m \in M, \mu(m) = (m^{\downarrow}, m^{\downarrow\uparrow})$$

This provides a practical way of determining the largest concept in whose extent a certain object appears, or which other objects share all the attributes of a given object.
The Galois lattices are normally represented by their Hasse diagram, as in this paper. Each node in such a diagram represents a formal concept and each arc indicates an order relation between two concepts, where the larger is placed above the smaller, with the restriction that no intermediate concept exists.
Labeling the nodes of the Hasse diagram with the complete description of the associated concept is difficult to read, so objects are normally labeled with the lowest

concept in whose extent they appear, $\gamma(g)$, while the attributes do so with $\mu(m)$, the highest concept in whose intent they appear.
It should be pointed out that the original lattice and the context itself can be reconstructed from this representation. The sets of objects and attributes are directly obtained from the sets of labels, while the incidence matrix is obtained from the expression $(g,m) \in I \Leftrightarrow \gamma(g) \leq \mu(m)$.

	author	booktitle	chapter	editor	institution	journal	note	pages	publisher	shool	title	year
ARTICLE	✓					✓					✓	✓
BOOK	✓			✓					✓		✓	✓
BOOKLET											✓	
INBOOK	✓		✓	✓				✓	✓		✓	✓
INCOLLECTION	✓	✓							✓		✓	✓
INPROCEEDINGS	✓	✓									✓	✓
MANUAL											✓	
MASTERTHESIS	✓									✓	✓	✓
MISC												
PHDTHESIS	✓									✓	✓	✓
PROCEEDINGS											✓	✓
TECHREPORT	✓				✓						✓	✓
UNPUBLISHED	✓						✓				✓	

Figure 2. Incidence table for the formal context extracted from the bibliographic reference types used in BibTeX

To illustrate the potential of these techniques, we shall consider the formal context associated to the reference types used in BibTeX, as shown in Figure 2. In this case, the bibliographic reference types will be represented as objects and their compulsory characteristics as attributes. In this case, the incidence is produced when the characteristic is compulsory for a certain reference type.
The Hasse diagram associated to the lattice shown in Figure 3 can be obtained algorithmically from the data of this table. This diagram shows all the information of the original table. However, it gives a clearer view of some aspects of the structure of the reference types used in BibTeX. The following can then be stressed:

- The appearance of a node labeled with one reference type underneath one labeled with another, means that all the compulsory characteristics of the first are present in the second.
- Some nodes are labeled with several reference types, thus showing that, with the information from the table, the reference types are equivalent.
- Some nodes are not labeled with any reference type, but with some characteristic. Such nodes provide new abstractions that can be useful for understanding the structure of the bibliographic references in BibTeX, providing a single point of definition to these characteristics.

- There are even nodes without any labeling at all. Such nodes represent an abstraction that brings together several reference types used as the starting point for the incremental construction of other types.

The practical interest of the *FCA* is guaranteed by the existence of several algorithms and tools, which allow the Galois lattice to be obtained from a formal context. On the other hand, as already indicated, this way of representing the information from the formal context shows the existing structure in the original data set. This structure can be analyzed manually, or by using tools that implement algorithms based on the mathematical properties of the complete lattices.

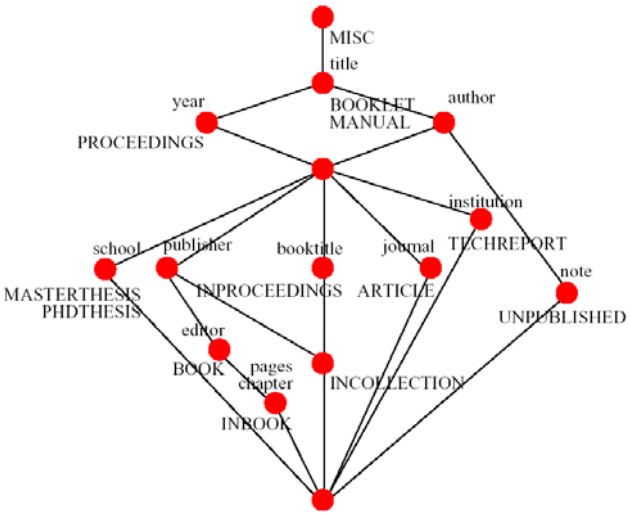

Figure 3 Lattice associated to the bibliographic reference types used in BibTeX

The application of this technique to the analysis of real information in a concrete domain requires the establishment of the way in which the incidence matrix will be constructed, determining what will be interpreted as objects and attributes respectively, and the way in which the lattice should be interpreted later. Different incidence matrices provide different views of the structure present in the original data.

4 Formal Concept Analysis support to Conceptual Abstraction

As we previously mention, this paper present a part of our proposal, the part which deals with the *FCA* support to the *Conceptual Abstraction* phase. This phase is in charge of the relational schema transformation into the object-oriented schema. The transformation start with an enriched logical schema (from the *Data Analysis* phase) and obtain a class diagram.

This transformation requires some information concerning the semantics of the original relational schema. A diagram of classes semantically equivalent to the initial logical schema is obtained by applying *FCA* techniques.

The process we propose for obtaining the class diagram starts from the following information:

- A list of the relations and their attributes,
- The verified inclusion dependencies, and
- The candidate and foreign keys of the relations.

The aim is to obtain a class diagram that fits the object model defined for UML and which reflects the underlying model in the original logical schema. In the class diagram, we have:

- The significant classes of the problem that have been identified,
- The specialization/generalization hierarchy between identified classes, and
- The association relations (as a general framework for the rest of the UML relations, e.g. aggregations) between the identified classes. The establishment of the associations can give rise, if the original schema so requires, to the appearance of new classes as association classes that characterize the defined relation.

The process is divided into two phases. Class identification and there organization in a specialization/generalization hierarchy is obtained outright in the first phase of the process. On the other hand, obtaining the associations and their characteristics by means of association classes is the result of the second phase of the process.

Each phase uses a different *FCA* application (to cover different objectives). As mentioned in the previous section, *FCA* application requires the problem we wish to solve to be modeled in terms of objects, attributes and a binary relation between them (involving the way in which the incidence matrix will be built), so that the interpretation of the Galois lattice obtained from the application of the algorithms answers the aims of the problem to be solved.

There are two aims in the first phase: the identification of significant classes and their classification in a specialization/generalization hierarchy. The application of *FCA* to achieve this aim is not new. In fact, most work on *FCA* application is in this area. [Dao et al., 2002] states that the hierarchy that can be obtained from the correct application of *FCA* in this way, is the hierarchy which complies with the criteria of the best factorization of the class characteristics. For many authors, the hierarchy obtained (Galois sub-hierarchy) has a maximum quality, is the ideal hierarchy.

From the initial information, the formal context (G, M, I) is built:

1. Sets *G* and *M*:
    a. Objects (*G*): the relations of the initial conceptual schema.
    b. Attributes (*M*): the attributes of each relation that is NOT a primary or candidate key.
2. Binary relation *I*:
    a. If a relation of the initial logical schema has an attribute *m*, then there is an incidence between the object *g* that represents the relation and the attribute *m*, $(g,m) \in I$.
    b. The following rule of implication is established: If there is an inclusion dependency between two relations, then the object *g* that represents the

relation that depends on the other, has an incidence with all the attributes m_i of the dependant relation ($(g,m_i) \in I$, $\forall\ m_i$).

The formal concepts (objects and attributes that are mutually determined) are obtained by applying the *FCA* algorithms to this formal context. Each formal concept obtained will be interpreted as an identified class. The Hasse diagram that presents the Galois lattice, built from the formal context, indicates a relation between the formal concepts detected (the classes). This relation is the specialization/generalization relation between the classes.

Thus, from the formal concepts obtained and the Hasse diagram, a first version of the desired class diagram can be obtained immediately and automatically.

It should be noted that the result of the transformation is not trivial. A class for each relation in the initial conceptual schema is not obtained. Some classes coincide with relations, but they may have their attributes intrinsic or attributes from another class. New classes that do *not* coincide with relations of the schema are obtained, and some relations have no direct correspondence in the detected classes.

The second phase aims to discover the association relations and their characterization by means of association classes.

As far as we know, up to the present this way of applying the *FCA*, as proposed by us for this second phase, has not been used.

In this case, the formal context (G,M,I) is built using the initial information:

1. Sets *G* and *M*:
    a. Objects (*G*): the relations of the initial conceptual schema
    b. Attributes (*M*): the attributes of each relation that ARE primary, candidate or foreign keys.
2. Binary relation *I*:
    a. If a relation of the initial logical schema has an attribute *m*, $m \in M$, then the object *g*, that represents the relation, is related to the attribute *m*, $(g,m) \in I$.

Applying the *FCA* results in a Galois lattice where the formal concepts represent classes of the problem and candidate classes for association class. On the other hand, the Hasse diagram indicates a relation between the classes. The associations between the classes are obtained from this relation. The way in which the concepts are related in the diagram follow three patterns:

1. A concept is below another in the lattice and linked to it: If both represent classes, there must be an association between both classes.
   In the initial problem, this means that the relations gave rise to classes. On the other hand, the Hasse diagram shows that the attributes of the higher concept are included in the lower concept. As only key attributes are included, this means that the relation which gave rise to the lower concept contains the keys of the higher one. If the keys are understood as references to objects, this indicates that the objects of the class represented by the lower concept can act as a reference for the objects of the class represented by the higher concept. Thus, there is an association between both classes.
2. A concept is below another two and linked to both: If the higher concepts represent classes, there must be an association between both classes. If the lower

concept is associated to attributes, that represent non-key attributes, then an association class characterizes the association between the classes.
In the initial problem, this means that the relations reflected in the higher concepts gave rise to classes. In the lower concept, the attributes that represent the keys of both relations of the initial schema are included. If, in the relational model, there is a relation that includes the keys of two relations, then it represents an interrelation between both. If, in addition, the said relation has other attributes, these will characterize the interrelation. Thus, the lower node indicates the existence of an association between the classes corresponding to the higher nodes which, in the presence of attributes, will be modeled as an association characterized by an association class.

3. A concept is below *more than two* concepts and is linked to them: If all the higher concepts represent classes, there must be several binary associations between the classes that represent the higher nodes and the class that represents the lower node. It should be noted that in some cases, the class representing the lower node may not be previously discovered, but is identified as a class due to this process.
   This pattern is a generalization of the previous pattern. The existence of non-binary relations in the relational model is considered. In the UML object model, associations can only be binary. Thus, in the new model, the non-binary relations must be converted to binary associations. This requires the relation to be broken down into binary associations with another class and the appearance of this class, if it did not already exist.

The class diagram produced in the first phase of the process is completed using the results obtained. This step is also immediate and automatic due to the fact that the mathematical properties of the complete lattices allow algorithms to be defined for analyzing the results of applying the *FCA*.
The following section presents an example of the application of the defined method.

4.2 A practical example

In this section, the conceptual schema migration strategy based on *FCA* is explained by means of an example. We have a relational schema of an LDB. Example 1 shows the different relations that make up this database, along with its attributes; the primary key of each table is underlined. For the sake of simplicity, the existence of candidate keys has been not considered. These relations will be used to illustrate the later examples that are presented.

Example 1 (Example of a relational schema)
Author (AuthID, Name, FirstName, Origin) contains the authors of the books in the library
Book (BookID, Title, Publisher, Abstract) contains the books of which copies are stored in the library
Written (BookID, AuthID) defines which authors wrote which books
Copy (BookID, SerNumber, DateAcq, State) contains instances of books which are physically stored in the library
Member (PID, Name, FirstName, PhoneNumber) contains people who can borrow books from the library

Addresses (PID, AddressID, Street, City) contains addresses of the library users
Project (PCode, Name) contains projects for which books can be lent
ExtProject (PCode, Duration) specialization of projects with external partner
Lending (BookID, SerNumber, LendingDate, PCode, PID) contains copies which are actually borrowed by a certain library user for a certain project
EndLending (BookID, SerNumber, LendingDate, EndDate, PCode, PID) contains the history of when copies have been lent out

Table 1 (below) shows the set of foreign keys and inclusion dependencies of this example. The origin of the information included in it is to be found in the database catalogue and/or known by an expert who knows the details of LDB, in a process assisted by *FCA*, as we have proposed in Section 2.1. As indicated above, this information will be the input for the *FCA* algorithms that will allow the classes, the inheritance hierarchy of the classes and the associations between them to be obtained.

	Type	Table	Foreign Key	Table	Primary Key
1	Foreign Key	Written	BookID	Book	BookID
2	Foreign Key	Written	AuthID	Author	AuthID
3	Foreign Key	Copy	BookID	Book	BookID
4	Foreign Key	Lending	BookID, SerNumber	Copy	BookID, SerNumber
5	Foreign Key	Lending	PCode	Project	PCode
6	Foreign Key	Lending	PID	Member	PID
7	Foreign Key	EndLending	BookID, SerNumber	Copy	BookID, SerNumber
8	Foreign Key	EndLending	PCode	Project	PCode
9	Foreign Key	EndLending	PID	Member	PID
10	Foreign Key	Addressess	PID	Member	PID
11	Inclusion Dep.	ExtProject	PCode	Project	PCode

Table 1: Foreign keys and inclusion dependencies of the example.

The first phase the *FCA* algorithm is applied with the previously presented criteria and the formal concept which is used as input is shown in Table 2.

	Name	FirstName	Origin	Title	Publisher	Abstract	DateAcq	State	PhoneNumber	Street	City	Duration	LendingDate	PCode	PID	EndDate
Author	✓	✓	✓													
Book				✓	✓	✓										
Copy							✓	✓								
Member	✓	✓							✓							
Addresses										✓	✓					
Project	✓															
ExtProject	✓											✓				
Lending													✓	✓	✓	
EndLending														✓	✓	✓

Table 2 Incidence table for the first formal context

Applying the algorithm results in the lattice shown in Figure 4. It shows the inheritance relations between the classes that have been found. For instance, a new abstract class *Atr(2/1)* is obtained from which *Member* and *Author* inherit. With this result, the reengineer can interact and accept the new class by naming it *Person*.

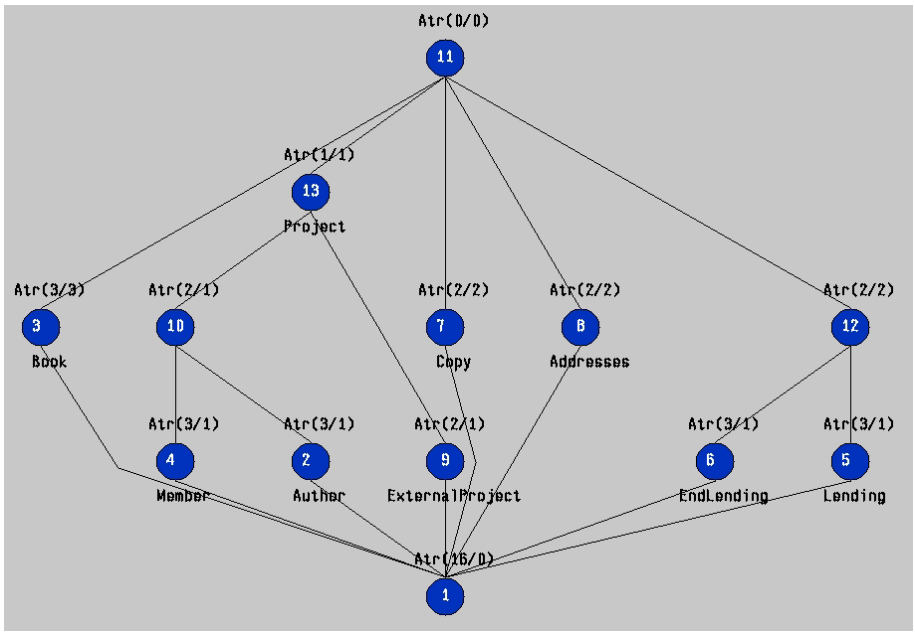

Figure 4: Lattice obtained with the first application of *FCA* techniques (inheritance analysis).

Although it may be a conceptual error to represent both *Atr(2/1)* and *ExternalProject* as inheriting from *Project*, it is shown as so here because the attribute *Name* is shared by the relations *Member*, *Author* and *Project*. The fact that the algorithm has picked up this incidence leads us to consider the possibility of working in the area of semantic enrichment as well, detecting some possible inconsistencies that can appear in the data and which the reengineer may not have controlled.

The first version of the class diagram is established using this lattice.

The algorithm has been applied in the second phase with a formal context, according to the criteria defined above, as shown in Table 3. The result of applying the *FCA* is another new lattice which is shown in Figure 5 and from which the associations between classes are derived.

The node labeled *Member* and that labeled *Addresses* fit the first pattern presented. In this case, there is an association between the classes *Member* and *Addresses*.

The nodes labeled *Book*, *Author* and *Written* fit the second pattern presented. In this case, there is an association between the classes *Book* and *Author* (higher nodes). No association class characterize this association because the lower node (*Written*) doesn't have attributes but the key attributes.

	AuthID	BookID	SerNumber	PID	AddressID	PCode	LendingDate
Author	✓						
Book		✓					
Written	✓	✓					
Copy		✓	✓				
Member				✓			
Addresses				✓	✓		
Project						✓	
ExtProject						✓	
Lending		✓	✓	✓		✓	✓
EndLending		✓	✓	✓		✓	✓

Table 3 Incidence table for the second formal context

The nodes labeled *Member*, *Project* and *Copy*, and the lower node *Lending* fit the third pattern presented. In this case, there are associations between the classes *Member* and *Lending; Project* and *Lending; Copy* and *Lending* in the class diagram. Note that the node labeled *Project* is also labeled *ExternalProject*. If the classes in the same node are ancestors and descendents, the superclass is always choosen here and the association relation is inherited by descendents.

The first version of the class diagram is completed with the result of this second phase.

As already mentioned, the class diagram, expressed in UML, can be obtained automatically. Figure 6 shows this diagram as it is after the first iteration.

This *FCA* support is the basis of our proposal. With it, we hope to be able to gather

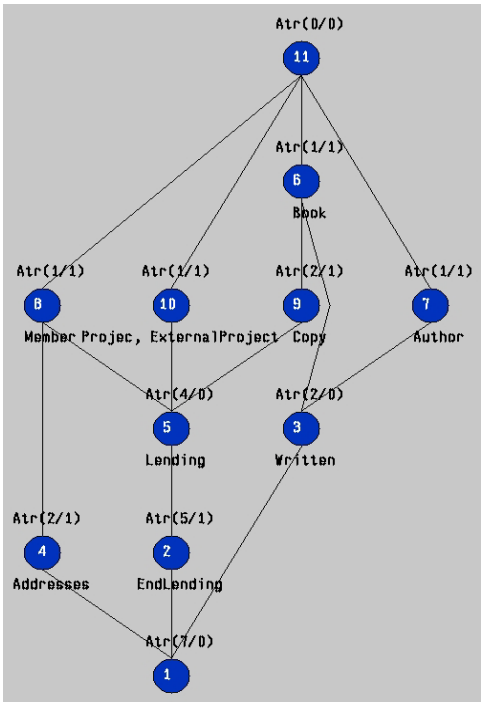

Figure 5 Lattice obtained by applying the *FCA* techniques a second time (associations analysis)

the maximum amount of semantics concerning the data and thus construct a new approach in the task of *Conceptual Abstraction*. This is completed with the rest of the points we present in Section 2.1 in order to obtain *FCA* support to the whole *Database Reengineering* process.

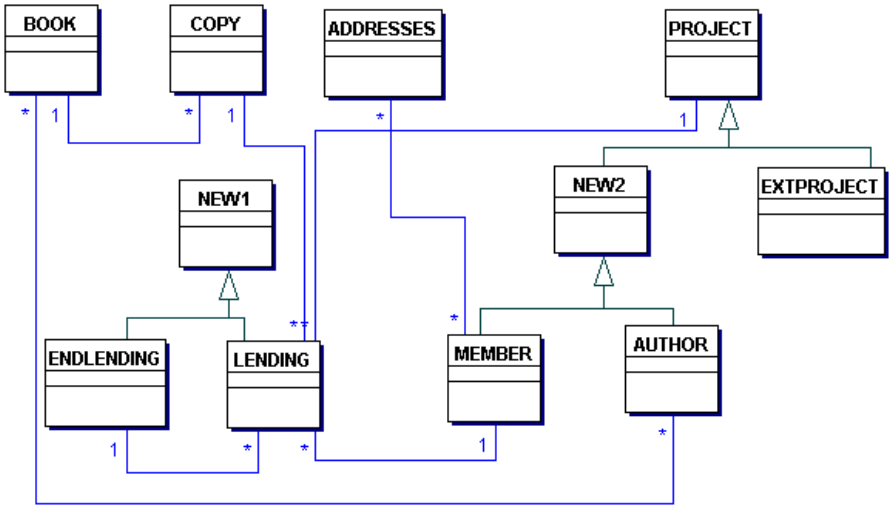

Figure 6 The class diagram obtained from the lattice information

5 Related work

Much of the work in the *Database Reengineering* area concentrates on the *Conceptual Abstraction* phase, although there are references to the *Data Analysis* and *Migration* phases and some also take into account the whole *Reengineering* process.
Most research starts with a relational schema as the initial input of any task. It is supposed that the semantic information has been obtained through an exhaustive analysis of this relational schema and, as a general rule, it is assumed that the relational schema should be in the Third Normal Form. One exception to this assumption is to be found in [Premerlani & Blaha, 1994]. This proposal starts from relational schemas that may not be well-designed and a class diagram is obtained according to the OMT object model [Rumbaugh et al., 1991]. The main inconvenience is that the method is not easily automated as it is based on the perceptions of the users.
In [Behm et al., 1997] a mechanism for migrating relational databases to object-oriented databases, which also takes into account the *Data Migration* phase, is presented. This migration process is divided into two parts: migration (transformation) of schemas and data migration, which is itself divided into four

subparts: rules for class creation, rules for attribute creation, instance creation process and attribute assignment process. A formal notation for describing relational and object-oriented schemas and the relationships between them is introduced. Then the operations (transformation rules) with which an object-oriented schema can be created step by step, are investigated. A transformation rule transforms a part of the relational schema, called a pattern, into a part of an object-oriented schema. For example, a foreign key dependency can be transformed into a "part-of" relation or a relation can be transformed into a class.

Similarly, in [Alhajj & Polat, 2001] a new approach for carrying out the transformation of a relational database to an object-oriented database is presented. Once the new schema has been obtained, the method provides an algorithm that manages the migration of data from the tuplas of the relational database to the objects of the new database.

In [Jahnke, 1999], an approach that aims to cover all the phases of the *Reengineering* process is presented. *Data Analysis* and *Conceptual Abstraction* are stressed. For this phase, graph grammar is applied to transform the structure of the analyzed logical data into an object-oriented conceptual data schema. This structure can be interactively improved and redesigned according to needs. The possibility of carrying out redesigning operations is formally defined using graph transformations. Based on this formalism, a management component has been developed for the consistency that incrementally propagates the modification of the logical structure to a conceptual structure (redesigned), in the case where iterations are carried out.

Finally, in [Hainaut et al., 1993] a model of common generic data is used which subsumes the conceptual, logical and physical constructors that are used to design a database. A catalogue of schema transformations is defined and used to gradually replace implementation constructors with others of a higher level, while at the same time losing the initial schema. The results of this work can be found in a CASE tool that is still being developed [Roland et al., 2000].

The work we presented here also assumes that the relational schema is in the Third Normal Form. We are in the line of [Jahnke, 1999], but we think the *FCA* technique guaranteed the quality of the obtained object-oriented schema.

As it was already mentioned (Section 3) the use of *FCA* in the Database area is not new. [Schmitt and Conrad, 1999] present an approach to transform a class hierarchy into what the authors call a "normalized" form. Given a class hierachy, they applied *FCA* algorithms and obtain a new hierarchy which fits the maximal factorization criterion.

Another use of *FCA* in Databases is to support data exploration, allowing the detection of conceptual aspects. This application is used in the TOSCANA software system [Stumme et al., 1998]. TOSCANA is a tool for knowledge discovering in databases. The tool works with the information the database stores and facilitates *Data Mining* tasks.

5 Conclusions and future work

An automation proposal for some of the tasks that are carried out in the *Conceptual Abstraction* phase of the *Database Reengineering* process have been presented. This proposal is based on *Formal Concept Analysis* techniques and allows an enriched logical schema to be transformed into an object-oriented schema. The continuity of this work requires a deeper study of the possibilities that the *FCA* based techniques provide for the different phases of the *Database Reengineering* process. In the paper we have mentioned points where we explore the *FCA* usefulness. This proposal should also be verified by means of its application to real LDBs, by means of adequate metrics to establish a comparative study between the results obtained from our proposal and from other different techniques. This research line requires the construction of an interactive *Database Reengineering* tool capable of applying the *FCA* based techniques and of supporting interaction with the developer. On the other hand, the iterative nature of the *Database Reengineering* process means that the analysis of the impact of the changes in the logical schema on the conceptual schema obtained is especially important. In this sense, it is to be expected that the properties of the Galois lattices allow these modifications in the conceptual schema to be conveniently isolated, thus notably facilitating the complete process.

References

[Aiken, 1998] P. Aiken "Reverse Engineering of Data". *IBM Systems Journal* 37(2), 1998.

[Alhajj and Polat, 2001] R. Alhajj and F. Polat. "Reengineering Relational Databases to Object-Oriented: Constructing the Class Hierarchy and Migrating the Data". In *Proceedings of the Eighth Working Conference on Reverse Engineering*. IEEE Computer Society Press, 2001.

[Behm et al., 1997] A. Behm, A. Geppert, K. R. Dittrich: "On the Migration of Relational Schemas and Data to Object-Oriented Database Systems". In *5th International Conference on Re-Technologies for Information Systems*, Klagenfurt, Austria, December, 1997.

[Blaha and Premerlani, 1995] M. Blaha and W. Premerlani. "Observed idiosyncrasies of relational database designs". In *Second Working Conference on Reverse Engineering,* Toronto, Ontario, Canada. IEEE Computer Society Press, 1995.

[Blaha and Premerlani, 1996] M. Blaha and W. Premerlani. "A catalog of object model transformations". In *Proc. of 3rd Working Conference on Reverse Engineering,* Monterey, California. USA. IEEE Computer Society Press, 1996.

[Cattell et al., 2000] R. G. G. Cattell, D. Barry, M. Berler, J. Eastman, D. Jordan, C. Russell, O. Schadow, T. Staniendam, and F. Velez. "*The Object Data Standard: ODMG 3.*0". Morgan Kaufmann Publishers, San Francisco (CA), USA, 2000.

[Dao et al., 2002] M. Dao, M. Huchard, T. Libourel, and C. Roume. *"Evaluating and Optimizing Factorization in Inheritance Hierarchies"*. Proceedings of The Inheritance Workshop. ECOOP'2002 Workshop, June, 2002.

[Elmasri and Navathe, 1994] R. Elmasri and S. B. Navathe. "*Fundamentals of Database Systems*". Benjamin/Cummings, Redwood City, 2nd edition, 1994.
[Ganter and Wille, 1999] B. Ganter and R. Wille. "*Formal concept analysis: mathematical foundations*". Springer, 1999.
[Godin et al.,1995] R. Godin, G. Mineau, R. Missaoui and H. Mili. "Méthodes de clasification conceptuelle basées sur les treillis de galois et applications". *Revue d'Intelligence Artificielle*, 9(2):105-137, 1995.
[Godin and Mili,1993] R.Godin and H.Mili. "Building and maintaining analysis-level class hierarchies using Galois lattices". *Proceedings of OOPSLA* 1993.
[Hainaut et al., 1993] J-L. Hainaut, V. Englebert, J. Henrard, J-M. Hick, and D. Roland. "Requirements for information system reverse engineering support". Technical Report RP-95-13, University of Namur, Belgium, 1993.
[Hainaut, 2000] – Hainaut, J-L., "The Nature of Data Reverse Engineering", *Data Reverse Engineering Worksho*p, EuroRef, Seventh Reengineering Forum, Reengineering Week 2000, Zurich, Switzerland, March 2000.
[Jahnke and Wadsack, 1999] J.-H. Jahnke and J. P. Wadsack. "*Varlet: Human-Centered Tool Support for Database Reengineering"*. Workshop on Software-Reengineering, Bad Honnef, Germany. Technical Report (Fachbericht INFORMATIK 7/99), University Koblenz-Landau. J. Ebert, B. Kullbach, F. Lehner (Editors). May 1999.
[Jahnke, 1999] J. H. Jahnke. "Management of Uncertainty and Inconsistency in Database Reengineering Processes". PhD thesis, University of Paderborn, Dept. of Mathematics and Computer Science, 33095 Paderborn, Germany, September 1999.
[Jahnke et al., 2002] J.-H. Jahnke, W. Schäfer, J. P. Wadsack and A. Zündorf. "*Supporting Iterations in Exploratory Database Reengineering Processes"*. Science of Computer Programming (Special Issue). Elsevier Science, 2002. (to appear).
[Müller et al., 2000] Hausi A. Müller, Jens H. Jahnke, Dennis B. Smith, Margaret-Anne D. Storey, Scott R. Tilley, Kenny Wong: "Reverse Engineering: A Roadmap," *The Future of Software Engineering Track at the 22nd International Conference on Software Engineering (ICSE 2000)*, Limerick, Ireland, June 4-10, 2000.
[Premerlani and Blaha, 1994] W. Premerlani and R. Blaha. "An Approach for Reverse Engineering of Relational Databases". *Communications of the ACM, 37 (5), pp. 42-49, 1994*.
[Prieto et al., 2002] F. Prieto,Y. Crespo, J. M. Marqués and M. A. Laguna. "Formal Concept Analysis support to Domain Frameworks Construction". *V Iberoamerican Workshop on Requirements Engineering and Software Environments Development*. La Habana, Cuba, Abril 2002. *(In Spanish)*
[Ramanathan and Hodges, 1997] S. Ramanathan and J. Hodges. "Extraction of object-oriented structures from existing relational databases". *ACM SIGMOD Recor*d, 26(1), 1997.
[Roland et al., 2000] D.Roland, J.-L.Hainaut, J.-M.Hick, J.Henrard, V.Englebert, "Database Engineering Processes with DB-MAIN". In *Proc. of the 8th Conference on Information Systems (ECIS2000)*. Vienna, Austria, 2002.
[Rumbaugh et al., 1991] J.Rumbaugh, M.Blaha, W.Premerlani, F.Eddy and W.Lorensen. *"Object-Oriented Modeling and Design"*, Prentice Hall, Inc., 1991.

[Schmitt and Conrad, 1999] I. Schmitt and S. Conrad. “Restructuring Object-Oriented Database Schemata by Concept Analysis”. In “*Fundamentals of Information Systems*”, Kluwer Academic Publishers, pp. 177—185 (Chapter 12), 1999.
[Siff and Reps, 1999] M. Siff and T. Reps. “Identifying modules via concept analysis”. *IEEE Transactions on Software Engineering*, 25, December, 1999.
[Snelting and Tip, 1998] G. Snelting and F. Tip. “Reengineering class hierarchies using concept analysis”. *ACM SIGSOFT Software Engineering Notes, 23(6)*, 1998.
[Stevens and Pooley, 1998] P. Stevens and R. Pooley. “Systems reengineering patterns”. In *Proc. of ACM Foundations of Software Engineering, Lake Buena Vista, Florida, US*A, pages 17-23. ACM Press, 1998.
[Stumme et al., 1998] G. Stumme, R. Wille and U. Wille, *“Conceptual Knowledge Discovery in Databases Using Formal Concept Analysis Methods”*, LNCS 1510, pp. 450-458, 1998.
[Wille, 1982] R. Wille. “Restructuring lattice theory: An approach based on hierarchies of concepts”. In *Ordered Sets*, pages 225-470. Reidel, Dordrecht-Boston, 1982.