STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 1, 2014

AN FCA DRIVEN ANALYSIS OF MAPPING CONCEPTUAL DESIGNS TO XML SCHEMAS

VIORICA VARGA AND CHRISTIAN SĂCĂREA

Abstract. XML is a popular data representation and exchange format over the web. Many articles propose different conceptual models for XML design, but there is no standard format for conceptual design of XML data. Franceschet et al. (2013) demonstrate that nested schemas are globally more efficient than flat ones. In this paper an FCA based representation of XML conceptual modeling is proposed. Entity-Relationship data model is used initially. A Relational FCA approach is given for the relations between entity sets. A mapping from Entity-Relationship model to a schema for XML in form of concept lattices are presented for relations of type one-to-one, one-to-many and many-to-many. The obtained concept lattices reflect these three different relationship types, also the possibility of nested XML scheme design can be seen on the lattices.

1. Introduction and Previous Work

As XML becomes a popular data representation and exchange format over the web, XML schema [7] design has become an important research area. In database environment, the Extended Entity-Relational (EER) model is used as a conceptual schema to represent the structure of the data and the relationships in a relational database, but there is no standard format to represent XML data structure and hierarchies. This article is a contribution to the development of design methodologies for XML databases, advocating the use of Formal Concept Analysis (FCA).

There are many other approaches toward this goal. For instance, Elmasri et al. ([2]) propose a visual XML Schema Designer including EER modeling and UML class diagrams. Fong et al. ([3]) are introducing an XML Tree Model as a conceptual scheme of an XML model. A very recent contribution, by Franceschet et al. [4] propose a mapping from conceptual design to logical

Received by the editors: April 2, 2014.

2010 *Mathematics Subject Classification.* 68P15, 03G10.

1998 *CR Categories and Descriptors.* H.2.1 [**D**atabase Management]: Logical design – Scheme and subschema.

Key words and phrases. XML data design, Formal Concept Analysis.

schemas for XML data. They adopt the Entity-Relationship (ER) model extended with specialization as the conceptual model for native XML databases. Starting with the observation that, in general, the same Entity-Relationship (ER) conceptual schema can be mapped in different XML schemas, they present two alternative ways of mapping ER conceptual schemas into XML: a flat relational-style design methodology and a nesting approach. The former never nests an entity into another entity, but the latter nests entities as much as possible. Also, an experimental evaluation and comparison of these two mapping mechanisms over large datasets is presented. This evaluation shows that both validation and query processing are globally more efficient with nested schemas than with flat ones. This motivates the theoretical problem of finding the best possible nesting for the design.

In this paper a Formal Concept Analysis [5] based representation of XML conceptual modeling is proposed. Entity-Relationship data model is used initially. A Relational FCA approach is given for the relations between entity sets. A mapping from Entity-Relationship model to a schema for XML in form of concept lattices are presented for relations of type one-to-one, one-to-many and many-to-many. The obtained concept lattices [8] reflect these three different relationship types, also the possibility of nested XML scheme design can be seen on the lattices. At this point, Formal Concept Analysis proves to be a valuable tool for the design of such schemas. It gives an expressive graphical representation of the relationships between entities. None of the other approaches for XML scheme design use the two cardinality constraints for binary relationships as Franceschet et al. (2013) [4]. It is a difficult task to understand these two cardinality constraints without a graphical representation. These two cardinality constraints are crucial for nesting. Our approach is based on [4], which is a very recent result, so our solution is basically different from the older proposals. It takes into account two cardinality constraints for binary relationships, it gives an expressive representation of them.

2. Conceptual Hierarchy Representation of Database Conceptual Design

In this section, we propose an FCA grounded approach to conceptual database and XML data design. This problem is definitely not a new one. Even if there are many approaches concept lattices prove to be, once again, the best platform to communicate, understand and discriminate between different conceptual design related notions, due to their expressiveness. This approach is based on the relational context families introduced by Huchard et al. (2007) in [6], as well as on the known notion of a many-valued context.

Definition 1. A relational context family is a pair $(\mathbb{K}, R)$, where

- $\mathbb{K}$ is a set of formal contexts $\mathbb{K}_i = (G_i, M_i, I_i)$, $i \in I$,
- R is a set of binary relations $R_{kl} \subseteq G_k \times G_l$, $k, l \in I$. The contexts $\mathbb{K}(R_{kl}) := (G_k, G_l, R_{kl})$ are called relational contexts.

One of the most common method for conceptual database design is the Entity-Relationship (E-R) model (see [1]), which is a high level conceptual data model and describes in an abstract way the database. An E-R schema consists of entity sets, attributes, and relationships between entity sets. Since attributes are representing properties of real world objects, they are many-valued, hence we can represent every entity set of the E-R Model as a many-valued context.

Let be $E_1, E_2, \ldots, E_l$ be entity sets from the E-R diagram. Every entity set E_i, $i = 1, \ldots, l$ is represented as a many-valued context. If we denote by $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$ the attributes of E_i, the objects are the entities of the entity set E_i and will be denoted by $e_1, e_2, \ldots$. The corresponding values are the values of an entity for a given attribute. We denote the key of the entity set E_i with K_{E_i}.

Entity sets are connected by relations, which represent semantic connections between the entities. The simplest form of relationship is the binary relation. Relationships involving more than two entity sets are called n-ary relations and can equivalently expressed by multiple binary relations. Usually, these relationships are characterized by two roles expressing the function that the two related entities are playing in the relation. These roles can be annotated with maximum cardinality constraints. These constrains are denoting the maximum number of objects, i.e., entities of the codomain of the relation to which any entity of the source set can be related.

Binary relationships have two cardinality constraints ([4]) of the form (x, y), where x is a natural number, that specifies the minimum cardinality or participation constraint. The second cardinality constraint y is the maximum cardinality constraint, which is a positive natural number or N which represents an arbitrarily large natural number.

Let us consider two entity sets A and B and a binary relation R between A and B with left cardinality constraint (x_1, y_1) and right cardinality contraint (x_2, y_2). We use the $A \overset{(x_1,y_1)}{\longleftrightarrow} R \overset{(x_2,y_2)}{\longleftrightarrow} B$ notation for this.

Based on their maximum cardinality constraints, we have

(1) one-to-one relationships, if both roles have maximum cardinality 1;
(2) one-to-many, if one role has maximum cardinality 1 and the other has maximum cardinality N;
(3) many-to-many, if both roles have maximum cardinality N.

We are going to use FCA in order to represent these roles in a concept lattice which will serve as basis of further understanding and communication, as well as for the database conceptual design.

The entity sets A and B can be represented as a many-valued context and every entity from this set being represented as an object in the context. The relationship between entities from A and B will be represented using their entity keys in a formal context.

Definition 2. Let R_{ij} be a relation between the entity sets E_i and E_j, $i, j \in I$. The relational context of R_{ij} is defined as the context having as object sets different key values of K_{E_i} and objects different key values of K_{E_j}, the incidence relation being R_{ij}.

We illustrate the one-to-one relationships in Table 1, the one-to-many in Table 2 and the many-to-many in Table 3. The conceptul lattice representation reflects the conceptual database design, which is independent of database model.

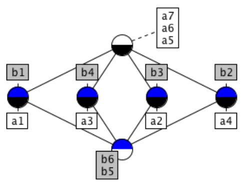
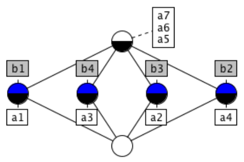
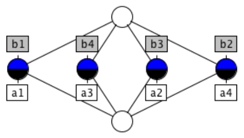
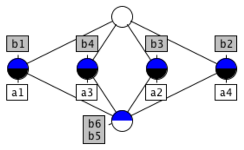
We study also the possibility of mapping the ER Model to the structure of XML data in XSN (introduced in [4]) form. In [4], the authors give two alternative definitions of this mapping, one modeling entities as global XML elements and expressing relationships between them in terms of keys and key references (flat design), the other one modeling relationships by including elements for some entities into the elements for the other entities (nest design). They compare the flat design with the nested one and prove that the nested approach leads to improvements in both query and validation performance.

In the following, we give the nested structure of XML data where it is possible. If the flat structure is the same as the nest structure, no information is provided, since the flat structure has already been studied in the authors previous work.

We denote by KA the key of the entity set A and KB the key of B. In the many-to-many case, nesting of the element for one entity into the element for the other one is never possible. The 0 value for the minimum cardinality constraint specifies that there are elements which are not related to the elements of the other entity set. This determines that the nested design is arguable. We will analyze the different cases below. The representation of an XML binary relationship using concept lattices is more expressive than the classical $A \overset{(x_1,y_1)}{\longleftrightarrow} R \overset{(x_2,y_2)}{\longleftrightarrow} B$ notation. We focus on nested XML data design, where the value of x_1 and x_2 is decisive.

2.1. **One-to-One Relationships.** Let be A and B two entity sets and a R one-to-one relationship between them. For this type of relationship the minimum and maximum cardinality constraint values are 0 or 1. If we analyze

Table 1. Conceptual Hierarchy Representation of One-to-One Relationship

Nr.	**Relationship**	**Conceptual Hierarchy**	**Nested XSN**
1.	$A \overset{(0,1)}{\longleftrightarrow} R \overset{(0,1)}{\longleftrightarrow} B$		
2.	$A \overset{(0,1)}{\longleftrightarrow} R \overset{(1,1)}{\longleftrightarrow} B$		A (KA, R?) R (B) B(KB)
3.	$A \overset{(1,1)}{\longleftrightarrow} R \overset{(1,1)}{\longleftrightarrow} B$		A (KA, R) R (B) B(KB) OR B (KB, R) R (A) A(KA)
4.	$A \overset{(1,1)}{\longleftrightarrow} R \overset{(0,1)}{\longleftrightarrow} B$		B (KB, R?) R (A) A(KA)

the lattices of Table 1 we can observe, that for every concept (excepting the top an bottom of the lattice) there exists one element from A and one element from B. In these lattices, the representation of one-to-one relationship is very expressive. The concept lattices vary only in bottom and top elements, but these are decisive in the nested XML scheme. As we can see on Table 1 the nested XML schemes differ in these four cases. In the following, we analyse the different labeling of the top and bottom elements and hence different types of relations.

Case 1: $A \overset{(0,1)}{\longleftrightarrow} R \overset{(0,1)}{\longleftrightarrow} B$. There are elements of A which are not related to any element of B, and elements of B which are not related to any element of A. They are exactly the intent of the greatest element and the extent of the smallest element of the concept lattice, respectively. Consider for example, the entity set $A = \{a_1, a_2, \ldots, a_7\}$ and $B = \{b_1, b_2, \ldots, b_6\}$as a working example to generate the concept lattice. The elements of A which are not related to elements of B are displayed at the top of the lattice ($\{a_5, a_6, a_7\}$), while the elements of B which are not related to elements of A ($\{b_5, b_6)\}$), appear at the bottom of the lattice. The nested design of XML data is not possible in this case. The inclusion of B in A would lead to the loss of B elements which are not related to elements of A, and vice-versa.

Example: `Men` $\overset{(0,1)}{\longleftrightarrow}$ `spouse` $\overset{(0,1)}{\longleftrightarrow}$ `Women`. One man can be married in christianity with one woman, but there can be unmarried men or women.

Case 2: $A \overset{(0,1)}{\longleftrightarrow} R \overset{(1,1)}{\longleftrightarrow} B$. Consider $A = \{a_1, a_2, \ldots, a_7\}$ and $B = \{b_1, b_2, b_3, b_4\}$. In this case, there are elements of A which are not related to elements of B, and they are displayed at the top of the lattice ($\{a_5, a_6, a_7\}$) as the intent of the greatest concept, but every element of B is related with one element of A. The nested design is possible, including B in A, but not the inverse.

Example 1. `AcademicStaff` $\overset{(0,1)}{\longleftrightarrow}$ `advisor` $\overset{(1,1)}{\longleftrightarrow}$ `StudentsGroups`. Every group of students has one instructor as advisor, but not every staff member is advisor. So StudentsGroup can be nested in AcademicStaff, but not the inverse case. The XSN scheme is the following:

```
AcademicStaff (AName, Gender, BirthDate, advisor?)
  advisor (StudentsGroups)
    StudentsGroups (GroupCode, GroupName?, StudentsNR)
```

Case 3: $A \overset{(1,1)}{\longleftrightarrow} R \overset{(1,1)}{\longleftrightarrow} B$. In this case, every element of A is related to one element of B, and also every element of B is related with one element of A. Consider $A = \{a_1, a_2, a_3, a_4\}$ and $B = \{b_1, b_2, b_3, b_4\}$. The intent of the top element and extent of the bottom element of the concept lattice are empty. In this case both nesting is correct, we can include A in B or B in A.

Example 2. `Faculties` $\overset{(1,1)}{\longleftrightarrow}$ `managed` $\overset{(1,1)}{\longleftrightarrow}$ `Deans`. Every faculty has a dean and every dean manages only one faculty. There are no faculties without dean and no dean who does not manages a faculty. The XSN scheme is the following:

```
  Faculties (FName, DeanName, Address, Homepage, managed)
     managed (Deans)
       Deans (AName, Gender, BirthDate)
```

The inverse is correct too.

```
  Deans (AName, Gender, BirthDate, managed)
    managed (Faculties)
      Faculties (FName, DeanName, Address, Homepage)
```

Case 4: $A \overset{(1,1)}{\longleftrightarrow} R \overset{(0,1)}{\longleftrightarrow} B$. This case allows the existance of elements of B not related to elements of A, but every element of A is related with one element of B.For $A = \{a_1, a_2, a_3, a_4\}$ and $B = \{b_1, b_2, \ldots, b_6\}$, the are elements of B which are not related to elements of A ($\{b_5, b_6)\}$) are displayed at the bottom of the lattice. This is the inverse of case 2, A can be included in B.

Example 3. `Departments` $\overset{(1,1)}{\longleftrightarrow}$ `managed` $\overset{(0,1)}{\longleftrightarrow}$ `AcademicStaff`. Every department has only one manager, but not every staff member is a manager. The XSN scheme is the following:

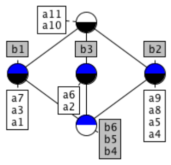
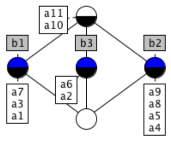
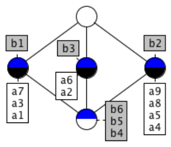
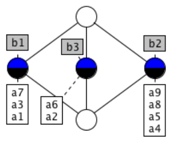
```
  AcademicStaff (AName, Gender, BirthDate, managed?)
    managed (Departments)
       Departments (DName, DAddress, DHomepage, worksIn+)
```

2.2. **One-to-Many Relationships.** The R relationship between A and B is a one-to-many relationship. For this type of relationship the minimum cardinality constraint values are 0 or 1 and the maximum cardinality constraint values are 0 or N (an arbitrarily large natural number). If we analyze the lattices of Table 2 we can observe, that for every concept (excepting the top an bottom) exists one element from B and more than one element from A. This illustrates the relationship between elements of A and elements of B. The many part of the one-to-many relationship is A, so on the lattice we can see this assignment between the two entity sets. The XML nested structure includes A (the many part) in B (the one part) if it is possible.

In **case 5**, we have $A \overset{(0,1)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$. Consider $A = \{a_1, a_2, \ldots, a_{11}\}$ and $B = \{b_1, b_2, \ldots, b_6\}$ as a working example to generate the concept lattice. There are elements of A which are not related to elements of B, being displayed at the top of the lattice ($\{a_{10}, a_{11}\}$), as well as elements of B which are not related to elements of A ($\{b_4, b_5, b_6)\}$), appearing at the bottom of the lattice. Having elements in A not related to a parent, hierarchical design is not possible, thus flat and nest mappings coincide.

Example 4. `Students` $\overset{(0,1)}{\longleftrightarrow}$ `Offered` $\overset{(0,N)}{\longleftrightarrow}$ `Groups`. Let us imagine some groups for students, from which every student can choose 0 or 1 group and

Table 2. One-to-Many Relationship Lattice Representation

Nr.	Relationship	Conceptual Hierarchy	Nested XSN
5.	$A \overset{(0,1)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$		
6.	$A \overset{(0,1)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$		
7.	$A \overset{(1,1)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$		B (KB, R*) R (A) A(KA)
8.	$A \overset{(1,1)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$		B (KB, R+) R (A) A(KA)

in every group can participate 0 or N students. So there can be students which don't choose a group and groups which are empty. The hierarchical design should include students in the selected group, but students can not be included in offered groups, because there can be some students which do not participate in a group.

In **case 6**, we have $A \overset{(0,1)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$. Take $A = \{a_1, a_2, \ldots, a_{11}\}$ and $B = \{b_1, b_2, b_3\}$. For these cardinality constrains, there are elements of A which are not related to elements of B, as we can see looking at the top of the lattice ($\{a_{10}, a_{11}\}$), but every element of B is related with elements of A. For the same reason as in case 5, nested design of XML structure is not possible.

Example 5. `Publications` $\overset{(0,1)}{\longleftrightarrow}$ `Has` $\overset{(1,N)}{\longleftrightarrow}$ `PublicationTypes`. We will give some type to publications, like journal paper, conference paper, etc., but there can be some publications which can not be included in one of the PublicationTypes. Every publication type has more publications included, one publication is included in utmost one type. So nested design is not possible. If we try to include Publications in PublicationTypes, then publications which has no type will be lost.

In **case 7**, we have $A \overset{(1,1)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$. Consider as a working example $A = \{a_1, a_2, \ldots, a_9\}$ and $B = \{b_1, b_2, \ldots, b_6\}$. There are elements of B, which are not related to elements of A ($\{b_4, b_5, b_6)\}$), appearing in the bottom of the lattice, but every element of A is related with one element of B. Hierarchical design of XML data is possible, including A in B. There are elements of B without child elements and this is expressed by the notation $R*$.

Example 6. `Specializations` $\overset{(1,1)}{\longleftrightarrow}$ `SpecOfferedBy` $\overset{(0,N)}{\longleftrightarrow}$ `Faculties`. Every specialization is included only in one faculty, one faculty has more specializations included, but there can be faculties which are not divided in specializations. So nested design is possible, specializations are included in corresponding faculties. The XSN scheme is the following:

```
Faculties (FName, DeanName, Address, Homepage, SpecOfferedBy*)
   SpecOfferedBy (Specialization)
      Specialization (SpecName, AcceptedStudentsNR)
```

`Faculties` without specializations will have no `SpecOfferedBy` elements.

In **case 8**, we have $A \overset{(1,1)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$. For these constrains, every element of A is related to one element of B, and also every element of B is related with elements of A. The intent of the top element and the intent of the bottom element are empty. The concept lattice has been generated for $A = \{a_1, a_2, \ldots, a_9\}$ and $B = \{b_1, b_2, b_3\}$. Nested structure of XML data is possible, there are no parent elements without child elements and we denote this by $R+$.

Example 7. `AcademicStaff` $\overset{(1,1)}{\longleftrightarrow}$ `worksIn` $\overset{(1,N)}{\longleftrightarrow}$ `Departments`. Every staff member works in a department, but in one department there are more staff members. There are no departments without members and there are no members which are not in some department. The XSN scheme is the following:

```
Departments (DName, DAddress, DHomepage, worksIn+)
  worksIn (AcademicStaff)
    AcademicStaff (AName, Gender, BirthDate)
```

2.3. **Many-to-Many Relationships.** In this paragraph we study the many-to-many relationships. In contrast with concept lattices for one-to-one and one-to-many relationships, if we analyze the lattices displayed in Table 3 we can observe that the structure of the concept lattice is more complex. For one-to-one and one-to-many relationships they are just nominal scales with additional information at the top and bottom element of the lattice. If the relationship is of type one-to-one concepts in every level are labeled only with attribute (the key of one element from entity set B) and one object (the key of one element from entity set A). If the relationship is of type one-to-many, then the the concepts are labeled by one attribute (the key of one element from entity set B, the parent) and more objects (the keys of elements from entity set A, the child). Concepts of conceptual lattices which represent many-to-many relationships are labeled by more attributes (the keys of elements from entity set B) and more objects (the keys of elements from entity set A) and these concepts are on more hierarchical levels. The top and bottom of the concept lattice are labeled if there are dangling (they are not related by relation R to the other entity set) elements in the entity sets A and B. In cases 9, 10, and 11 there are dangle elements, which are displayed at the bottom and the top of the lattice. Nested design is not possible for many-to-many relationships, so we will not analyze in detail this kind of relationship.

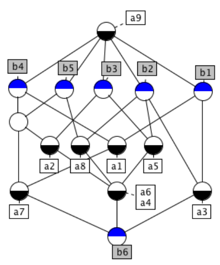
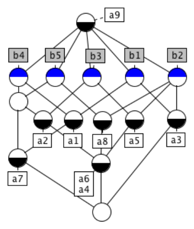
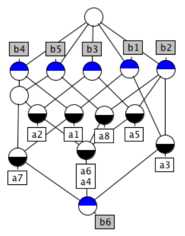
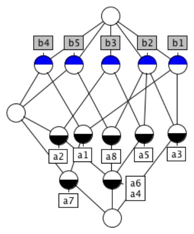
Example 8. `Students` $\overset{(0,N)}{\longleftrightarrow}$ `choose` $\overset{(0,N)}{\longleftrightarrow}$ `Courses`. One course can be chosen by 0 or more students and there can be students which choose 0 or more courses. This is an example for case 9. This example can be considered for case 10 as well, but with different constraints: every course was chosen by 1 or more students. The example can be similarly adapted for case 11 and 12.

3. Conclusion and Further Work

This article is a contribution to XML data design. It is a new proposal since it is based on Franceschet et al. (2013) [4]. Concept lattices give an expressive graphical representation of the relationships between entities. Our approach uses two cardinality constraints for binary relationships as Franceschet et al. [4]. The representation of XML data relationships using concept lattices proves to be not only expressive, but also very useful, especially for the 0 minimum cardinality constraints.

As future work we propose to implement a graphical tool, which give the possibility to construct Entity-Relationship Model with minimum and maximum constraints on binary relationship. The software will build the conceptual lattice from E-R model and the nested or flat XML scheme will be proposed.

Table 3. Many-to-Many Relationship Lattice Representation

Nr.	**Relationship**	**Conceptual Hierarchy**
9.	$A \overset{(0,N)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$	 a9 b4 b5 b3 b2 b1 a2 a8 a1 a5 a7 a6 a4 a3 b6
10.	$A \overset{(0,N)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$	 a9 b4 b5 b3 b1 b2 a2 a1 a8 a5 a3 a7 a6 a4
11.	$A \overset{(1,N)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$	 b4 b5 b3 b1 b2 a2 a1 a8 a5 a7 a6 a4 a3 b6
12.	$A \overset{(1,N)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$	 b4 b5 b3 b2 b1 a2 a1 a8 a5 a3 a7 a6 a4

References

[1] R. Elmasri, S.B. Navathe: *Fundamentals of Database Systems*, Addison Wesley, (2010).
[2] R. Elmasri et al.: *Conceptual modeling for customized XML schemas*, Data and Knowledge Engineering 54, pp. 57-76 (2005).
[3] J. Fong, S. K. Cheung, H. Shiu: *The XML Tree Model toward an XML conceptual schema reversed from XML Schema Definition*, Data and Knowledge Engineering 64, pp. 624-661, (2008)
[4] M. Franceschet, D. Gubiani, A. Montanari, C. Piazza: *A Graph-Theoretic Approach to Map Conceptual Designs to XML Schemas*, ACM Transactions on Database Systems, Vol. 38, pp. 6-44 (2013)
[5] B. Ganter, R. Wille.: *Formal Concept Analysis.* Mathematical Foundations. Springer, Berlin-Heidelberg-New York(1999)
[6] M. Huchard, M.R. Hacene, C. Roume, P. Valtchev.: *Relational concept discovery in structured datasets*, Ann. Math. Artif. Intell. 49(1-4) (2007) pp. 39-76
[7] W3C. *XML Schema*, http://www.w3.org/TR/xmlschema-0/ (2004)
[8] S.A. Yevtushenko: *System of data analysis "Concept Explorer"*. (In Russian). Proceedings of the 7th national conference on Artificial Intelligence KII, Russia, pp. 127-134 (2000).

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Kogălniceanu 1, 400084 Cluj-Napoca, Romania
E-mail address: ivarga@cs.ubbcluj.ro
E-mail address: csacarea@math.ubbcluj.ro