Extending E-R for Modelling XML Keys

Martin Necasky
Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic
martin.necasky@mff.cuni.cz

Jaroslav Pokorny
Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic
jaroslav.pokorny@mff.cuni.cz

Abstract

With the growing popularity of XML there is a need not only to describe the structure of XML data but also its semantics. For the conceptual modelling of XML we can use existing conceptual models. However, special features of XML require extensions of these models. In this paper, we study conceptual modelling of XML keys. We extend the notion of E-R keys to be suitable for modelling the semantics of XML keys and we show how to express them on the XML logical level.

1. Introduction

Recently, XML [11] has become an important format for data representation and exchange. Usually, we describe the structure of the XML data by schema languages such as XML Schema [10]. Moreover, for the design and further maintenance of the XML data it is also important to describe its semantics. However, the schema languages are not suitable for such a description. Therefore, there is an emerging area of the conceptual modelling for XML. It is the same idea as in the case of relational data where we use E-R for example.

However, E-R is not suitable for the conceptual modelling of XML data that has special features such as irregular structure, ordering, mixing structured and unstructured data, or hierarchical structure. Therefore, it is necessary to provide new approaches suitable for the modelling of these special features. We provide a survey of this area in [6] where we identify two groups of approaches.

The approaches in the first group are based on extending E-R with new modelling constructs (for example [3], [7], or [8]). However, modelling the required hierarchical structure is problematic with these approaches. It is either left on the system that derives the hierarchical structure automatically or it is left on the designer who denotes the required hierarchical structure directly in the conceptual schema by special hierarchical relationship types.

However, there is a common situation where we need to represent concepts such as books and their authors in two or more different hierarchical structures. The first structure can be a list of books and for each book a list of authors and the second structure can be a list of authors and for each author a list of his or her books. This can not be neither determined automatically by the system without any further extensions of the model nor specified by explicit hierarchical relationship types in the conceptual schema because the modelled semantics of the data would be hidden among a huge number of hierarchical relationship types.

The approaches in the second group emerge from hierarchical structure (for example [2]). The conceptual schemes are trees where the nodes are entity types and edges are relationship types. In the previous example with books we can model each required hierarchy with the separate hierarchical conceptual schema. However, while in E-R or its extensions we can model books by one entity type, in this approach books are represented by two different nodes, one in each schema. Consequently, the modelled semantics is hidden in the hierarchical structures.

The weak points of these approaches result from the modelling of the semantics and the hierarchical structure of the data on the same level. Therefore, it is necessary to further extend recent conceptual models to be suitable for modelling XML data. In [5] we proposed a new conceptual model for XML data called *XSEM*. We tried to precede the problems mentioned above by dividing the modelling process to two levels. The semantics of the data is modelled on the first level with an extension of the E-R model while the hierarchical organization in XML documents is specified on the second level. Therefore, XSEM preserves the advantages of E-R, mainly its simplicity and clearness, and adds the ability to model different hierarchical structures.

In this paper, we further extend XSEM with new constructs for conceptual modelling of *XML keys*. Compared to relational keys, XML keys can be relative, i.e. not valid on the whole XML document but only on its parts. Moreover, we can have a key valid over several XML elements of different types. For example, we can have a common key *label*

1-4244-1476-8/07/$25.00 ©2007 IEEE.

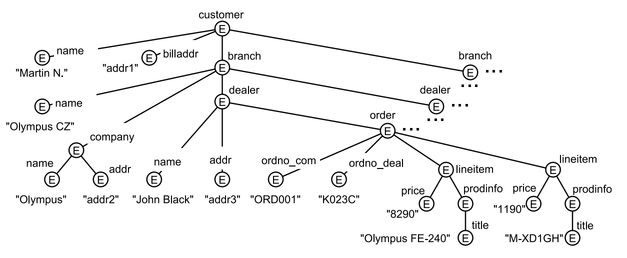

Figure 1. XML Tree

identifying the descendants representing chapters and sections in the context of book. Therefore, if we need to model XML keys on the conceptual level the classical approach using E-R keys is not sufficient.

Hierarchical structure of XML makes the description of the full semantics of keys hard or even impossible. Suppose Fig. 1 showing an XML document as a tree. Suppose a relative XML key specifying that in the context of each element targeted by a path $/customer/branch$ the child elements $dealer$ are identified by their $name$ child element. There is a formalism for the specification of such XML keys proposed in [1]. In this formalism the key is specified as a triple $(/customer/branch, dealer, \{name\})$ composed of an absolute path targeting the context of the key, a relative path targeting the target elements of the key, and a set of relative paths targeting the elements composing the key, respectively.

Such a key does not describe the semantics sufficiently. Each element $dealer$ represents a dealer executing orders (represented by the child elements $order$) ordered by a customer (represented by the grandparent element $customer$) from a branch (represented by the parent element $branch$). The key only specifies that for a $branch$ element there are no two child elements $dealer$ with the same value of $name$. However, what is the semantics of $name$? Does it identify dealers as real-world objects absolutely or only in the context of branches? This question can not be answered by the XML key itself.

There are more complex situations requiring the knowledge of the semantics of XML keys. In our example we have orders ordered by customers from branches. Each branch is a part of a company. Suppose a company and all branches of the company. We have orders each having a value of an attribute $ordno\_com$. This attribute identifies an order in the context of all orders ordered from the branches, i.e. if there are two different orders from the same company they have different value of $ordno\_com$. However, two different orders from two different companies can have the same value of the attribute. Such a key can not be expressed by an XML key in our example XML tree at Fig. 1 because the company is not represented as an ancestor element of the element representing the order. We can only specify an XML key $(/customer/branch, dealer/order, \{ordno\_com\})$. It specifies that no two $order$ elements that are descendants of the same $branch$ element have the same value of their child element $ordno\_com$. However, the full semantics is not captured.

The knowledge of the full semantics of XML keys is crucial for the correct processing of XML data such as querying and transformation. Suppose that we want to transform the structure of our XML document displayed at Fig. 1. We want to swap the elements $customer$ with their child elements $branch$, i.e. for each branch represented as a root element $branch$ there will be a list of child elements $customer$ representing the customers who ordered some orders from the branch. Without the knowledge of the semantics of the key $ordno\_com$ we would have to change the context of the XML key to $/branch/customer$ because we do not know the real semantics of the key. With this knowledge, we set the context correctly to $/branch$. With an additional transformation that swaps $branch$ elements with their $company$ child element, we should set the context of the key to $/company$ according to the semantics. Without the knowledge of the semantics we would set the context of the key to $/company/branch$.

In spite of the importance of the knowledge of the semantics of XML keys, a research into this problem has not been sufficient in recent approaches in the area of conceptual modelling. These approaches just adopt simple E-R keys and do not discuss neither the special features of XML keys nor the suitability of the adopted keys for the conceptual modelling of XML keys. Consequently, an extension of E-R keys suitable for XML keys is still missing.

The contributions of this paper are the following:

- We address the new problem of the conceptual modelling of XML keys.
- We extend basic E-R keys for the conceptual modelling of XML keys.
- We show the representation of extended keys in XML schemes and demonstrate that the full representation is not always possible.

This paper is organised as follows. In Sect. 2 we introduce XSEM briefly. In Sect. 3 we propose the extension of E-R keys. In Sect. 4 we show how to express the extended keys on the XML schema level using XML Schema constructs.

2. XSEM Model

XSEM builds on HERM [9] which is an extension of the E-R model. XSEM is composed of two parts, *XSEM-*

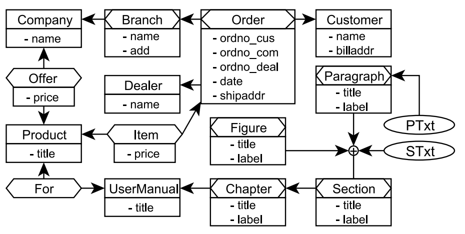

Figure 2. XSEM-ER Schema

ER and *XSEM-H*. We start modelling with XSEM-ER. Fig. 2 shows an XSEM-ER schema modelling part of a simple business domain. In the schema we use the classical modelling constructs:

- *Entity types* defined by a name and a list of attributes. There are strong entity types such as $Company$ modelling companies and weak entity types such as $Branch$ modelling branches of companies. A weak entity type has assigned a set of one or more entity types called *determinants*. For example, $Branch$ has one determinant $Company$.

- *Relationship types* defined by a name and a list of attributes and connecting two or more entity types called *participants* such as $Item$ modelling that products are items in orders.

The only extension is that the list of attributes of an entity or relationship type is ordered. Moreover, we use the following extending constructs:

- *Data node types* defined by a name and assigned to entity types. They are displayed as ellipses. We use them for modelling unstructured data. For example a data node type $PTxt$ assigned to $Paragraph$ models unstructured content of paragraphs.

- *Cluster types* that are used for modelling irregular structure and mixing structured and unstructured data. They are displayed as circles with inner '+'. In our example, we model figures and paragraphs in sections by $Figure$ and $Paragraph$, respectively, and unstructured content of sections by $SText$. The cluster type specifies that the figures and paragraphs are mixed with the unstructured text in each section.

An XSEM-ER schema describes the semantics of the data. It does not describe any hierarchical structure. This is left to the designer who derives hierarchical schema from the XSEM-ER schema for each of the required hierarchical structures using XSEM-H. A schema in XSEM-H is a hierarchical view on the XSEM-ER schema. It does not add

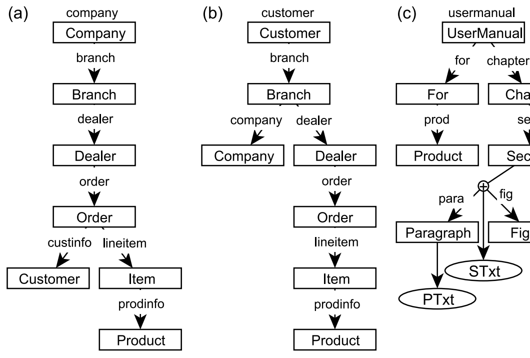

Figure 3. XSEM-H Views

any semantics. There can be several XSEM-H hierarchical views on the same XSEM-ER schema. It allows to specify more different hierarchical structures of the same data.

Fig. 3 shows three XSEM-H views on the XSEM-ER schema from Fig. 2. The views *(a)* and *(b)* specify two different hierarchical structures for orders. The view *(c)* specifies the hierarchical structure for user manuals.

There is a formal background behind XSEM. It serves as the binding between the non-hierarchical XSEM-ER level and the hierarchical XSEM-H level. Because of the lack of the space, we describe it only briefly. An XSEM-H view is a tree with labeled oriented edges. Each node in the tree represents a component of the XSEM-ER schema. Edges specify how instances of the components are organised in hierarchical XML data. Each edge connects only two nodes. However, there can be n-ary relationship and weak entity types as shown at Fig. 2. We need to decompose them to binary relationships that can be represented in XSEM-H views. For this we use so called *hierarchical projections*.

Suppose the weak entity type $Order$ modelling orders ordered by customers from branches of companies and executed by dealers. For example, we can specify a projection of $Order$ to $Customer$ and $Branch$. By this, we get the pairs of customers and branches such that for each pair the customer ordered some orders from the branch. Moreover, we need to specify which of the entity types is superior to the other. In our example, we require $Customer$ to be superior. It is formally specified as a *hierarchical projection* $Order[Customer \rightarrow Branch]$ where the arrow goes from the superior entity type. The projection specifies how the parts of orders composed of customers and branches are organised in hierarchical XML data. For each customer we have the list of branches that are in a projection pair with the customer. This hierarchical projection is represented in the example XSEM-H view (b) by the edge going from $Customer$ to $Branch$. Further, we need a projection of $Order$ to $Branch$ and $Dealer$ as it is shown in (b). However, it can not be a simple projection to pairs of branches and dealers. We must comprehend this projection in the context of $Customer$. It must specify that for each branch in the context of the superior customer (given by the previous projection) we want the list of dealers who executed some orders ordered by the customer from the branch. It is formally specified as $Order^{Customer}[Branch \rightarrow Dealer]$ where $Customer$ is called *context* of the hierarchical projection. It is represented in (b) by the edge going from $Branch$ to $Dealer$. To complete the decomposition of $Order$ we specify the last projection $Order^{Customer,Branch}[Dealer \rightarrow Order]$ that is represented in (b) by the edge going from $Dealer$ to $Order$.

For hierarchical projections we specify *cardinality constraints*. For example, we can specify that the cardinality of $Branch$ in $Order[Customer \rightarrow Branch]$ is $(0, *)$ which specifies that for a given branch there is 0 or more customers who ordered some orders from the branch. Further, we can specify that the cardinality of $Branch$ in $Order^{Customer}[Branch \rightarrow Dealer]$ is $(0, 5)$ which specifies that for a branch in the context of a customer there is from 0 to 5 dealers executing orders ordered by the customer from the branch.

3. Extended E-R Keys

In this section we extend the simple E-R keys for modelling XML keys. We work with XSEM-ER, because we model XML data. We start with examples. For $Section$ from Fig. 2 we can specify an E-R key $label$. Because $Section$ is weak, the key is relative to its determinant, i.e. $Chapter$. It means that $label$ identifies a section only in the context of its chapter. However, we need the key to be relative to $UserManual$ and not only to $Chapter$. Moreover, we need $label$ to be a common key of $Chapter$ and $Section$. This can not be described by E-R keys.

For the modelling of these features on the conceptual level we need a similar mechanism to XPath [12] that is used for the specification of keys in XML schemes. Therefore, we propose *paths* in XSEM-ER schemes that are used for targeting entity types and their determinants (recursively). A path is composed of one or more steps separated by '.'. The first step must be the name of an entity type. Each following step is the name of an entity type or it is a special symbol _ denoting an arbitrary entity type. Moreover, each step except the first one can be specified as a Kleene closure denoted by $*$. For example, we can specify paths $Order.Branch.Company$ or $Paragraph.\_*.UserManual$.

A path P *targets* a set of instances of entity types. If P is composed only of one step which is the name of an entity type E it targets the whole set of instances of E. Otherwise P is a path $P'.S$ where S is the last step of P. We have the set of instances targeted by P'. P targets a set of instances reachable by S from the instances targeted by P'. Assume that S is not a Kleene closure. We take an instance e targeted by P'. Let it be an instance of an entity type E. If there is a determinant D of E having the name specified with S then the target set of P contains an instance d of D that is a value of the determinant D of e. If S is a Kleene closure, we repeat the step recursively. For example, assume the path $Order.Branch.Company$. The first step targets all $Order$ instances. The second step targets each $Branch$ instance that is a determinant value of an $Order$ instance targeted by the previous step. The last step targets each $Company$ instance that is a determinant value of a $Branch$ instance from the previous step. In other words, it identifies each company for which there exists an order from any of its branches. The path $Paragraph.\_*.UserManual$ targets all user manuals that contain a paragraph.

Paths in XSEM-ER schemes are the basic formalism for our extension of E-R keys called *relative weak keys*. A *relative weak key* K is an expression

$$(context(K), target(K), attr(K))_{key}$$

where $target(K)$ is a non-empty set of entity types $E_1, \ldots, E_n$, called *target set* of K, such that $\bigcap_{i=1}^{n} attr(E_i) \neq \emptyset$ (where $attr(E)$ denotes a list of attributes of an entity type E), $attr(K)$ is a non-empty subset of $\bigcap_{i=1}^{n} attr(E_i)$, and $context(K) = \{P_1, \ldots, P_k\}$, $k \geq 0$, is a set of zero or more expressions called *context* of K such that for each $E \in target(K)$ and for each $P \in context(K)$ the expression $E.P$ is a path.

K specifies the following condition:

$$\begin{gathered}(\forall E, E' \in target(K)) \\ (\forall e_1 \in (E.P_1)^C \cap (E'.P_1)^C, \ldots, \\ \forall e_k \in (E.P_k)^C \cap (E'.P_k)^C) \\ (\forall e \in E^C, e' \in E'^C : (\forall 1 \leq i \leq k)(e.P_i = e'.P_i = e_i)) \\ [(\forall A \in attr(K))\,(e(A) = e'(A)) \rightarrow (e = e')]\end{gathered}$$

where E^C denotes a set of instances of an entity type E, P^C denotes a set of instances targeted by a path P, and $e.P$, where e is an instance of an entity type E and $E.P$ is a path, denotes a set of instances targeted by $E.P$ not starting in the whole E^C but only in e. The key K specifies that the attributes from K are a common key of the entity types from $target(K)$ but relative to the specified context.

For example, a relative weak key $(\{\}, \{Product\}, \{title\})_{key}$ has an empty context and specifies that $title$ is an absolute key of $Product$, i.e. two different instances of $Product$ have not the same value of $title$. A relative weak key $(\{\_*.UserManual\},$

$\{Section, Chapter\}$, $\{label\})_{key}$ specifies that in a user manual there are no two sections or chapters with the same value of $label$. More formally, if we take two different instances e and e' from $Section^C \cup Chapter^C$ such that both $e.\_ * .UserManual$ and $e'.\_ * .UserManual$ target the same instance of $UserManual$ then e and e' have different values of $label$. For $Order$ we can specify a key $(\{Customer\}, \{Order\}, \{ordno\_cus\})_{key}$ specifying that each order is identified by its $ordno\_cus$ in the context of the customer who ordered the order. We can also specify a key $(\{Branch.Company\}, \{Order\}, \{ordno\_com\})_{key}$ specifying that each order is identified by its $ordno\_com$ in the context of the company from which the order was ordered. Finally, we can specify a key $(\{Branch.Company, Dealer\}, \{Order\}, \{ordno\_deal\})_{key}$ specifying that each order is identified by its $ordno\_deal$ in the context of the company from which the order was ordered and the dealer executing the order.

4. Expressing Keys on Logical Level

From the XSEM-H views we derive XML schemes. The derivation is straightforward. We suppose the XML Schema language in this paper. Briefly, each node in the XSEM-H view is represented as a complex type definition and each edge is represented as an element declaration with the label of the edge as the name. However, the representation of conceptual relative weak keys in the XML schema is not so straightforward. The resulting XML key depends not only on the relative weak key itself but also on the hierarchical structure described by an XSEM-H view. Therefore, for each XSEM-H view there can be a different XML key derived from the same relative weak key. We also show that there can be an XSEM-H view for which we can not derive an XML key that fully represents the relative weak key.

In Sect. 1 we adopted the formalism for XML keys proposed in [1]. An XML key expressed by this formalism can be easily represented by an XML Schema key. Briefly, an XML key has a form $(p_c, p_s, \{p_{f,1}, \ldots, p_{f,k}\})$ where p_c is an absolute path and $p_s, p_{f,1}, \ldots, p_{f,k}$ are relative paths. The path p_c specifies the elements that are the context of the corresponding XML Schema key. The path p_s is the selector path and $p_{f,1}, \ldots, p_{f,k}$ are the field paths of the XML Schema key. Therefore, the XML key corresponds to the following XML Schema key:

```
<xsd:key>
  <xsd:selector xpath="p_t"/ >
  <xsd:field xpath="p_f,1"/ >
  ...
  <xsd:field xpath="p_f,k"/ >
< /xsd:key>
```

declared in the declarations of the elements specified by p_c.

In this section we show how to derive a logical XML key from a conceptual XSEM-ER relative weak key. Because of the lack of the space we demonstrate the derivation on a set of examples without complex technical details. These examples however explain the ideas sufficiently. The examples also show that it is not always possible to express the full semantics of conceptual relative weak keys. We show the derived XML keys in the formalism adopted from [1]. These XML keys can be directly represented as an XML Schema key as shown above.

First, suppose the relative weak key $(\{\}, \{Product\}, \{title\})_{key}$, that has an empty context, and the example XSEM-H view (a) at Fig. 3. Even though $title$ identifies instances of $Product$ absolutely we can not specify an absolute XML key $(/company, .//prodinfo, \{title\})$ because a product can be ordered in several different orders. Therefore, it can be represented by more elements in an XML document. For these elements we can not use the absolute key.

To specify the correct XML key we have to find the highest node U in the XSEM-H view such that an instance of $Product$ is not represented by more elements in the context of U. For this we use cardinality constraints proposed in Sect. 2. The node representing $Product$ in the XSEM-H view is a part of a hierarchical representation of $Item$. The cardinality of $Product$ in $Item^{Order}[Item \rightarrow Product]$ is $(0, 1)$, i.e. in an order a product is not ordered at all or only once. Therefore, an instance of $Product$ is not represented by more elements in the context of the element representing the order. Further, the cardinality of $Product$ in $Item[Order \rightarrow Product]$ is $(0.*)$, i.e. a product can be ordered in zero or more orders. There can be more orders represented in the XML document. An instance of $Product$ is represented by an element for each order where it was ordered. Therefore, the node representing $Order$ is the highest node satisfying the condition. Therefore, the relative weak key specifies in the hierarchical structure given by the example XML view (a) an XML key $(//order, item/prodinfo, \{title\})$. In a similar way we can find an XML key for $(\{\}, \{Dealer\}, \{name\})_{key}$ which is $(/company/branch, dealer, \{name\})$ for the structure specified by the example XSEM-H view (a).

For a relative weak key that has not an empty context we can find the corresponding XML key in a similar way. However, the context further restricts the context of the XML key. Suppose the relative weak key $(\{Branch.Company\}, \{Order\}, \{ordno\_com\})$. With the previous procedure we find out that for the hierarchical structure specified by the view (a) each $Order$ instance is represented by exactly one element in the XML data. Therefore, the context for the corresponding XML key is not restricted by the hierarchical structure. However, it is restricted by the context of the relative weak key and therefore we get an XML key $(/company, branch/dealer/order, \{ordno\_com\})$. For the hierarchical structure specified by the view (b) the node representing $Company$ is not an ancestor of the node representing $Order$ and therefore it can not be the context node for the XML key. We can specify only an XML key $(/customer/branch, dealer/order, \{ordno\_com\})$. However, this XML key does not describe the full semantics of the original key. This example shows that it is not always possible to fully express the semantics of a relative weak key with an XML key.

Assume further a situation where the context of a relative weak key of an entity type is not represented in the hierarchical structure as an ancestor but as a descendant of the node representing the entity type. In such a case the context of the relative weak key does not restrict the context of the corresponding XML key as in the previous cases. Assume for example the relative weak key $(\{Customer\}, \{Order\}, \{ordno\_cus\})_{key}$ and the example XSEM-H view (a). The weak key is relative to the entity type $Customer$. However, $Customer$ is represented as a descendant of the node representing $Order$.

In such a case we must represent the context of the relative weak key by complementing the attributes of the relative weak key with the attributes of relative weak keys of the entity types composing the context of the relative weak key. In our example, we get an XML key composed of the attributes of the relative weak key complemented with the attributes of the relative weak key of $Customer$ (i.e. the attribute $name$). Therefore, the relative weak key specifies an XML key $(/company, .//order, \{ordno\_cus, custinfo/name\})$. If there is not any relative weak key of $Customer$ it would not be possible to fully represent the relative weak key with an XML key.

At the end, we discuss the relative weak key $(\{\_ * .UserManual\}, \{Section, Chapter\}, \{label\})_{key}$. The difference from the previous situations is that it is the common key for $Section$ and $Chapter$. We can represent this relative weak key as in the previous cases for each of the entity types $Section$ and $Chapter$ and to merge the resulting XML keys. For the structure specified by the example XSEM-H view (c) we get an XML key $(/usermanual, .//chapter|.//section, \{label\})$.

5. Conclusions

In this paper we addressed the important problem of conceptual modelling of XML keys. Capturing the full semantics of XML keys on the conceptual level is important for correct XML data processing including querying and transformation. We introduced a conceptual model for XML data called XSEM and proposed in this model a formal extension of the basic E-R keys motivated by special features of XML keys. We showed that an extended conceptual key can be expressed by different XML keys depending on the required structure of the XML data. We also showed that it is not always possible to fully represent the conceptual key in an XML schema.

6. Acknowledgment

This paper was supported by the National programme of research (Information society project 1ET100300419) and by Grant Agency of Charles University (GAUK), grant number 204-10/257190.

References

[1] P. Buneman, S. Davidson, W. Fan, C. Hara, and W. Tan: Keys for XML. Computer Networks, 39(5), 473-487, 2002.

[2] G. Dobbie, W. Xiaoying, T.W. Ling, M.L. Lee: ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data. TR21/00, Dpt. of Computer Science, National University of Singapore, 2000.

[3] D.W. Embley, S.W. Liddle, R. Al-Kamha: Enterprise Modeling with Conceptual XML. In Proc. ER 2004, Shanghai, China, 150-165.

[4] W. Fan and J. Simeon. Integrity constraints for XML, JCSS, 66(1), 2003, 254291.

[5] M. Necasky: XSEM - A Conceptual Model for XML. In Proc. of APCCM2007, Ballarat, Australia. CRPIT, 67, 2007, 37-48. *http://crpit.com/confpapers/CRPITV67Necasky.pdf*

[6] M. Necasky: Conceptual Modeling for XML: A Survey. TR 2006-3, Dpt. of Software Engineering, Faculty of Mathematics and Physics, Charles University, Prague, 2006, 54 p. *http://www.necasky.net/papers/tr2006.pdf*

[7] G. Psaila: ERX: A Conceptual Model for XML Documents. In Proc. of the 2000 ACM Symposium on Applied Computing, Como, Italy, 2000, 898-903.

[8] A. Sengupta, S. Mohan, R. Doshi: XER - Extensible Entity Relationship Modeling. In Proc. of XML 2003, Philadelphia, USA, 2003, 140-154.

[9] B. Thalheim.: Entity-Relationship Modeling: Foundations of Database Technology. Springer Verlag, 2000, Berlin, Germany. ISBN: 3-540-65470-4

[10] W3C. XML Schema: Primer Second Edition. Recommendation, October 2004.*http://www.w3.org/TR/xmlschema-0*

[11] W3C. Extensible markup language (XML) 1.0 (third edition). Recommendation, February 2004. *http://www.w3.org/TR/2004/REC-xml-20040204*

[12] W3C. XML Path Language (XPath) Version 1.0. Recommendation, November 1999. *http://www.w3.org/TR/xpath*