Towards Grouping Constructs for Semistructured Data

François Bry, Dan Olteanu, Sebastian Schaffert

http://www.pms.informatik.uni-muenchen.de

7. September 2001

Abstract

Markup languages for semistructured data like XML are of growing importance as means for data exchange and storage. In this paper we propose an enhancement for the semistructured data model that allows to express more semantics. A data model is proposed and the implications on pattern matching are investigated.

1. Introduction

Meta-level information in semistructured databases is expressed

- through the naming of elements *and/or*
- implemented in the application that processes the data

Grouping Constructs as an enhancement to the semistructured data model

- allow to add *generic* metainformation explicitly
- are applicable to *data documents*, *schema/query documents* and *answers to a query*

2. Motivation

Terms	Courses		
	Comp. Sc.	Mathematics	Seminars
1	CS I	Algebra I and Analysis I	
2	CS II and Hardware Basics	Algebra II	
3	CS III	Graph Theory and App. Analysis	Programming Seminar or System Seminar or Hardware Seminar or Logics Seminar
4	CS IV and Advanced Algorithms	Stochastics or Numerical Mathematics	

This is a typical XML representation of the timetable:

Example

```
<course_of_studies>
  ...
  <term>
    <number>4</number>
    <computer_sciences>
      <course>CS IV</course>
      <course>Advanced Algorithms</course>
    </computer_sciences>
    <mathematics>
      <course>Stochastic</course>
      <course>Numerical Mathematics</course>
    </mathematics>
    <seminars>
      <course>Programming Seminar</course>
      <course>System Seminar</course>
      ...
    </seminars>
  </term>
</course_of_studies>
```

Introduction
Motivation
Grouping Facets
Data Model
Matching
Answer Semantics
Summary

Using *Grouping Constructs* could yield the following XML representation:

Example

```
<course_of_studies>
  ...
  <term>
    <number>4</number>
    <computer_sciences>
      <AND>
        <course>CS IV</course>
        <course>Advanced Algorithms</course>
      </AND>
    </computer_sciences>
    <mathematics>
      <OR>
        <course>Stochastic</course>
        <course>Numerical Mathematics</course>
      </OR>
    </mathematics>
    <seminars>
      <OR>
        <course>Programming Seminar</course>
        <course>System Seminar</course>
        ...
      </OR>
    </seminars>
  </term>
</course_of_studies>
```

3. Grouping Facets

The Grouping Constructs consist of any of the following **Grouping Facets**:

- *connector* [data,schema]: properties “and”, “or”, “xor”
- *order* [data,schema]: properties “ordered”, “unordered”
- *repetition* [schema]: properties “allowed” and “not allowed”
- *selection* [data,schema]: property “n to m”
- *exclusion* [schema]: for excluding certain items

4. Data Model

4.1. Data Trees (DTs)

A tree T = $(\mathrm{Nodes}, \mathrm{Edges})$ is a rooted DAG, where for every node $n \in \mathrm{Nodes}$ there is a unique path from the root **root** to n.

Definition 4.1 (elementary data tree)
*An **elementary data tree** DT, with set of nodes* Nodes*, set of edges* Edges *and root **root**, is a tree represented by the tuple* (Nodes*, name, children, **root**), where:*

- $name : \mathrm{Nodes} \to \mathrm{Labels}$ *is a function mapping each node to its label*
- $children : \mathrm{Nodes} \to \mathbf{Lists}(\mathrm{Nodes})$ *is a function such that if* $(n, m) \in \mathrm{Edges}$ *then* $m \in children(n)$

Definition 4.2 (data tree with grouping facets)
Given a set $\mathbf{G}$ *of grouping facets, a* data tree with grouping facets *is defined as a tuple* (Nodes,*name*,*children*,***root***,*grouping*)*, where:*

- *(*Nodes*, name, children,* ***root****) is an elementary data tree*
- $grouping : \mathrm{Nodes} \rightarrow \mathbf{Power}(\mathbf{G})$ *is a function mapping each node to a set of corresponding grouping facets.*

Notation:

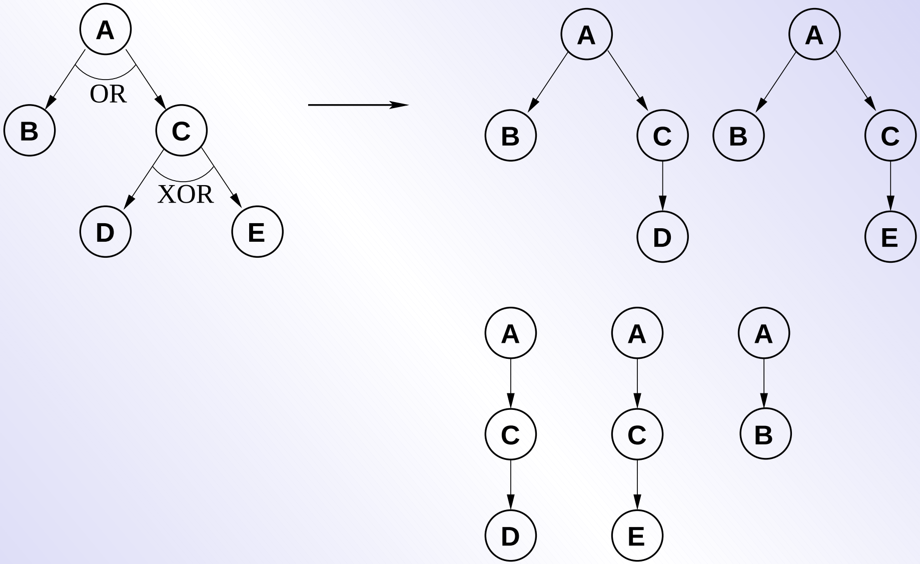
- $A(B_1, \ldots, B_n)$ denotes a tree with root A and the children B_i *in the given order*
- $A\{B_1, \ldots, B_n\}$ denotes a tree with root A and the children B_i *in any order*

4.2. Semantics of Data Trees with Grouping

Definition 4.3 (Interpretation of grouping facets)
Let $DT = (Nodes_{DT}, name, children, root, grouping_{DT})$ *be a data tree with grouping facets. A given node* $N \in Nodes_{DT}$ *with a grouping facet* $\mathcal{G} \in grouping_{DT}(N)$ *and children* $T_1, \ldots, T_n$ *is interpreted as its correspondent forest of data trees* $\mathbf{I}(N_{\mathcal{G}})$ *with root node* N *and without* $\mathcal{G}$ *as defined in the following table.*

$\mathbf{I}$ *applied recursively to all nodes from the data tree* DT *beginning with the root node generates a forest of elementary data trees. This forest is called the* interpretation *of* DT*, written* $\mathbf{I}(DT)$*.*

Example:



enriched subtree $N_{\mathcal{G}}$	interpreted as
$\mathbf{I}(N())$	$\{$ N() $\}$
$\mathbf{I}(N(T_1,\ldots,T_n))$	$\{N(T'_1,\ldots,T'_n) \mid T'_i \in \mathbf{I}(T_i), 1 \leq i \leq n\}$
$\mathbf{I}(N\{\})$	$\mathbf{I}(N())$
$\mathbf{I}(N\{T_1,\ldots,T_n\})$	$\bigcup\{\mathbf{I}(N(T_{\pi(1)},\ldots,T_{\pi(n)})) \mid$ $\pi\ permutation\ of\ \{1,\ldots,n\}\}$
$\mathbf{I}(N_\epsilon())$	$\mathbf{I}(N\{\})$
$\mathbf{I}(N_\epsilon(T_1,\ldots,T_n))$	$\mathbf{I}(N\{T_1,\ldots,T_n\})$
$\mathbf{I}(N_{AND}())$	$\mathbf{I}(N\{\})$
$\mathbf{I}(N_{AND}(T_1,\ldots,T_n))$	$\mathbf{I}(N\{T_1,\ldots,T_n\})$
$\mathbf{I}(N_{OR}())$	$\mathbf{I}(N\{\})$
$\mathbf{I}(N_{OR}(T_1,\ldots,T_n))$	$\bigcup\{\mathbf{I}(N\{P_1,\ldots,P_k\}) \mid \{P_1,\ldots,P_k\} \subseteq \{T_1,\ldots,T_n\}, 1 \leq k \leq n\}$
$\mathbf{I}(N_{ord.}())$	$\mathbf{I}(N())$
$\mathbf{I}(N_{ord.}(T_1,\ldots,T_n))$	$\mathbf{I}(N(T_1,\ldots,T_n))$
$\mathbf{I}(N_{unord.}())$	$\mathbf{I}(N\{\})$
$\mathbf{I}(N_{unord.}(T_1,\ldots,T_n))$	$\mathbf{I}(N\{T_1,\ldots,T_n\})$

enriched subtree $N_{\mathcal{G}}$	interpreted as
$\mathbf{I}(N_{repeat}())$ $\mathbf{I}(N_{repeat}(T_1,\ldots,T_n))$	$\mathbf{I}(N\{\})$ $\bigcup\{\mathbf{I}(N\{T'_1 \circ \ldots \circ T'_n\}) \mid$ $T'_i = (T_i,\ldots,T_i),\ \lvert T'_i\rvert = k_i, 1 \leq i \leq n, k_i \geq 0\}$
$\mathbf{I}(N_{i\ to\ j}())$ $\mathbf{I}(N_{i\ to\ j}(T_1,\ldots,T_n))$ $1 \leq i \leq j \leq n$	$\mathbf{I}(N\{\})$ $\bigcup\{\mathbf{I}(N\{P_1,\ldots,P_k\}) \mid$ $\{P_1,\ldots,P_k\} \subseteq$ $\{T_1,\ldots,T_n\}, i \leq k \leq j\}$
$\mathbf{I}(N_{AND}())$ $\mathbf{I}(N_{AND}(T_1,\ldots,T_n))$	$\mathbf{I}(N_{AND}()\neg(\emptyset))$ $\mathbf{I}(N_{AND}(T_1,\ldots,T_n)\neg(\emptyset))$
$\mathbf{I}(N_{exclude}()\neg(M))$ $\mathbf{I}(N_{exclude}(T_1,\ldots T_n))$	$\mathbf{I}(N\{\}\neg(M))$ $\{\mathbf{I}(N\{\}\neg(M\bigcup(T_1,\ldots,T_n)))\}$
$\mathbf{I}(N_{XOR}()\neg(M))$ $\mathbf{I}(N_{XOR}(T_1,\ldots,T_n)$ $\neg(M))$	$\mathbf{I}(N\{\}\neg(M))$ $\bigcup\{\mathbf{I}(N\{T_i\}\neg(M\bigcup T_j) \mid$ $1 \leq i,j \leq n, j \neq i\}$

5. Matching

Matching with Grouping Constructs is necessary

- for answering queries *and*
- for checking the validity of a database against a schema

Matching for Data Trees is based on a technique called simulation.

5.1. Simulation for data trees

Definition 5.1 (elementary simulation)
Given two elementary data trees DT_1 and DT_2, a binary relation $\mathcal{R} \subseteq \mathbf{Nodes}_{DT_1} \times \mathbf{Nodes}_{DT_2}$ is an ***elementary simulation*** *on DT_1 and DT_2 if it satisfies*

- *if $n_1 \ \mathcal{R} \ n_2$, then name(n_1) = name(n_2)*
- $\forall n_1, n_1' \in \mathbf{Nodes}_{DT_1} \ \forall n_2 \in \mathbf{Nodes}_{DT_2}$
  $(n_1 \mathcal{R} n_2 \wedge n_1' \in children(n_1) \Rightarrow$
  $\exists n_2' \in \mathbf{Nodes}_{DT_2} \ (n_1' \ \mathcal{R} \ n_2' \wedge \ n_2' \in children(n_2)))$

If $\mathcal{R}$ is a simulation on two elementary data trees DT_1 and DT_2, then we shall write $DT_1 \mathbf{sim}_{\mathcal{R}} DT_2$.

If the roots r_1 and r_2 of DT_1 and DT_2 are in the simulation ($r_1 \mathcal{R} r_2$), then the simulation is called rooted.

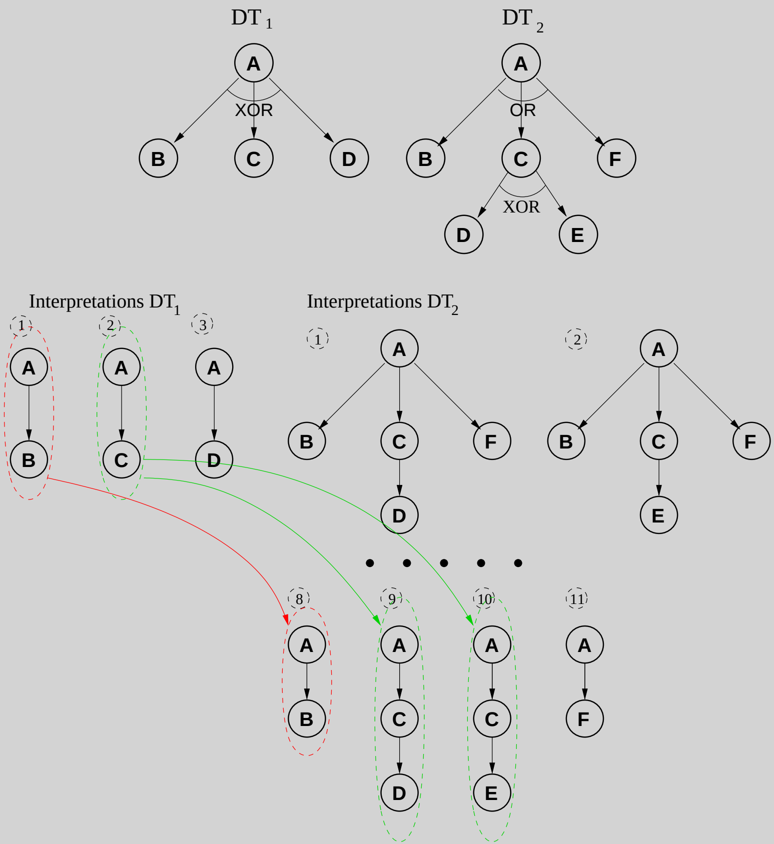
5.2. Naïve Matching with Grouping

Definition 5.2 (grouping simulation)
Given two enriched data trees DT_1 *and* DT_2 *with grouping facets, an elementary relation* $\mathcal{R} \subseteq Nodes_{DT_1} \times Nodes_{DT_2}$ *is a* ***grouping simulation*** *on* DT_1 *and* DT_2 *if it satisfies*

$$\exists\, I_1 \in \mathcal{I}_{\mathcal{G}}(DT_1)\ \exists\, I_2 \in \mathcal{I}_{\mathcal{G}}(DT_2)\ (I_1 \mathbf{sim}_{\mathcal{R}} I_2 \Rightarrow DT_1 \mathbf{sim}_{\mathcal{R}} DT_2)$$

If $\mathcal{R}$ *is a grouping simulation on* DT_1 *and* DT_2 *with grouping, then we shall write* $DT_1 \mathbf{sim}^{g}_{\mathcal{R}} DT_2$ *instead of* $DT_1 \mathbf{sim}_{\mathcal{R}} DT_2$.

Example:


DT 1
A
XOR
B
C
D
DT 2
A
OR
B
C
F
XOR
D
E
Interpretations DT1
1
2
3
Interpretations DT2
1
2
8
9
10
11

6. Answer Semantics

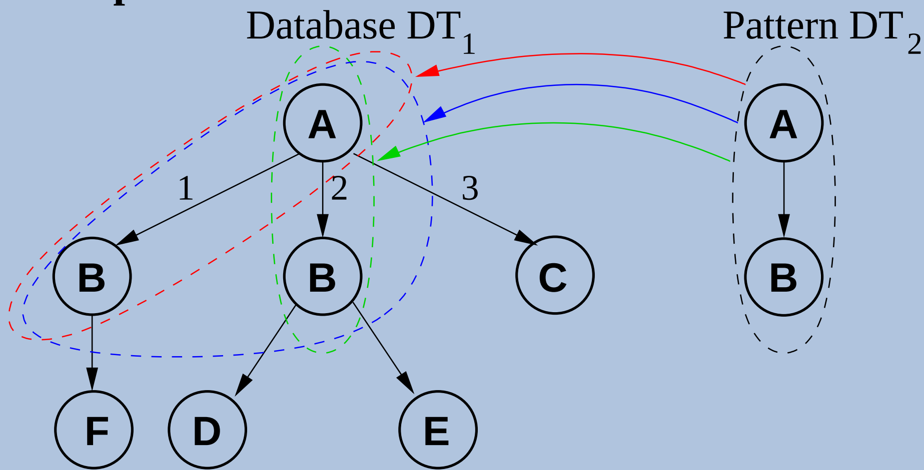
6.1. Simulation as Result

A straightforward method is to use the simulation relation to construct the answer.

However, this approach has some deficiencies:

- the nodes that are in the simulation are already in the pattern and thus known; usually one is interested in the *context* in which they are in the database
- in the general case, there is more than one simulation between a pattern and a database

Example:



There are three simulations between the two trees.

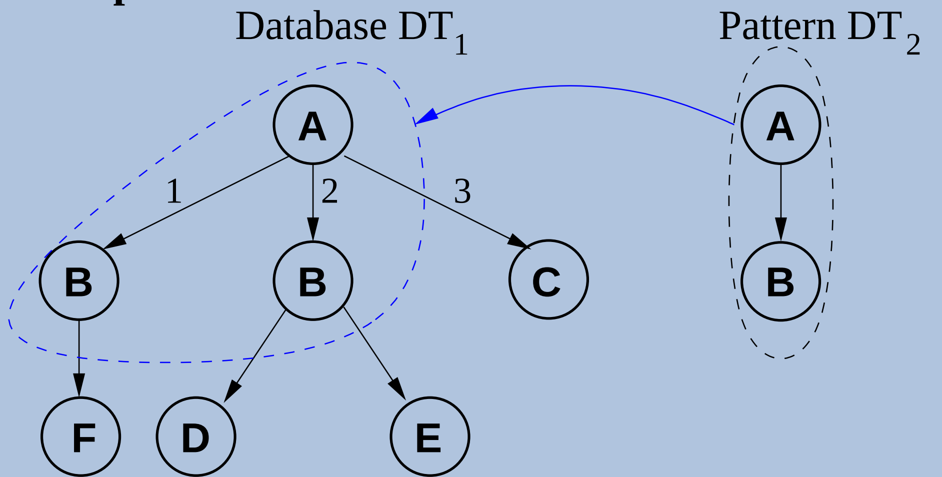
6.2. Maximal Simulation

For elementary data trees, this problem can be addressed by a technique called **maximal simulation**:

Proposition 6.1 (see Abiteboul, page 136)
If $DT_1 \mathbf{sim}_{\mathcal{R}_1} DT_2$ *and* $DT_1 \mathbf{sim}_{\mathcal{R}_2} DT_2$ *then* $DT_1 \mathbf{sim}_{\mathcal{R}_1 \cup \mathcal{R}_2} DT_2$.

Computing the maximal simulation is not difficult and will result in the largest matching fragment of the database.

Example:



The maximal simulation between the trees.

6.3. Grouping Inheritance: Non-Naïve Matching with Grouping

Grouping Inheritance treats the grouping facets on a more abstract level by simply comparing between the grouping facets and then “inheriting” the facets to the maximal simulation:

1. Generate the result from the maximal simulation between the two trees without taking into consideration the grouping properties
2. For each node in the resulting tree, inherit the grouping facet according to the relationships in the following table

Grouping Facet in the		
database	pattern	combined result
ϵ	ϵ	ϵ
AND	ϵ	AND
OR	ϵ	OR
XOR	ϵ	XOR
ϵ	AND	AND
AND	AND	AND
OR	AND	AND
XOR	AND	-[1]
ϵ	OR	OR
AND	OR	OR
OR	OR	OR
XOR	OR	XOR
ϵ	XOR	XOR
AND	XOR	-[1]
OR	XOR	XOR
XOR	XOR	XOR

[1] AND and XOR will not generate a match if the no. of elements is larger than 1

Grouping Facet in the		
database	pattern	combined result
unordered	ϵ	unordered
ordered	ϵ	ordered [2]
ϵ	unordered	unordered
unordered	unordered	unordered
ordered	unordered	ordered [2]
ϵ	ordered	ordered
unordered	ordered	ordered
ordered	ordered	ordered [2]
i to k	l to m	- [3]
i to k	l to m	- [4]
i to k	l to m	$max(i, l)$ to $min(k, m)$

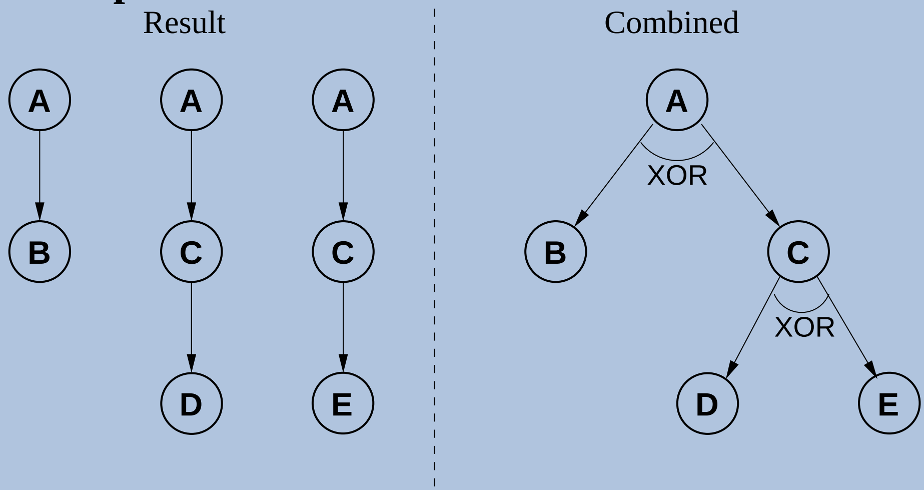
[1] AND and XOR will not generate a match if the no. of elements is larger than 1

[2] if children in pattern appear in the same order as in the database, - otherwise

[3] if result contains less than $max(i, l)$ children

[4] if $l < k$ or $m < i$

Example:



The result for the example used previously.

7. Summary

- In this work we presented an extension to semistructured data that adds generic *grouping constructs* to the data.
- Grouping constructs are applicable in the *data*, in a *schema*, in a *query* and in an *answer*.
- We presented a data model for our extension and introduced a method to match two data trees with grouping facets.

Literatur

[1] Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web. From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, San Francisco, CA, 2000.

[2] P. Buneman, S. Davidson, and D. Suciu. Programming constructs for unstructured data. In *DBLP*, 1995.

[3] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Modeling and querying semi-structured data. *Network and Information Systems*, 2(2):253–273, 1999.

[4] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogenous information sources. In *Information Processing Society of Japan*, 1994.

[5] Wenfei Fan, Gabriel M. Kuper, and Jérôme Siméon. *A Unified Constraint Model for XML*. Temple University, Bell Laboratories.

[6] Wenfei Fan and Jérôme Siméon. *Integrity Constraints for XML*. Temple University, Bell Laboratories.

[7] R. Durbin J. Thierry-Mieg. Syntactic definitions for the ACeDB data base manager. Technical report, MRC-LMB xx.92, MRC Laboratory for Molecular Biology, Cambridge, 1992.

[8] Daniela Florescu Jonathan Robie, Don Chamberlin. *QUILT: an XML query language*. http://www.almaden.ibm.com /cs/people/chamberlin/quilt_euro.html, March 2000.

[9] Dieter Jungnickel. *Graphen, Netzwerke und Algorithmen*. BI Wissenschaftsverlag Mannheim, 1994.

[10] Pekka Kilpeläinen. *Tree matching problems with application to structured text databases*. PhD thesis, Department of Computer Science, University of Helsinki, 1992.

[11] Peer Kröger. Modeling of biological data. Master's thesis, Institute for Computer Sciences, University of Munich, http://www.pms.informatik.uni-muenchen.de/lehre/projekt-diplom-arbeit/biological-data.html, 2001, to appear.

[12] Holger Meuss, Klaus Schulz, and François Bry. Towards aggregated answers for semistructured data. In *International Conference on Database Theory*, 2001.

[13] Roger King Richard Hull. Semantic database modeling: Survey, applications, and research issues. *ACM Computing Surveys*, 19(3):201–260, September 1987.

[14] Jonathan Robie. *XQL: XML Query Language*. http://metalab.unc.edu/xql/xql-proposal.xml, August 1999.

[15] Bernhard Thalheim. *Entity-Relantionship Modeling. Foundations of Database Technology*. Springer, 2000.

[16] W3C, http://www.w3.org/TR/1998/NOTE-XML-data-0105/. *XML-Data*, Jan. 1998.

[17] W3C, http://www.w3.org/TR/NOTE-ddml. *Document Definition Markup Language (DDML) Specification, Version 1.0*, Jan. 1999.

[18] W3C, http://www.w3.org/TR/xpath. *XML Path Language (XPath)*, 1999.

[19] W3C, http://www.w3.org/Style/XSL/. *Extensible Stylesheet Language (XSL)*, 2000.

[20] W3C, http://www.w3.org/TR/xptr. *XML Pointer Language (XPointer)*, 2000.

[21] W3C, http://www.w3.org/XML/Schema. *XML Schema*, March 2001.

[22] W3C, http://www.w3.org/TR/xquery/. *XQuery: A Query Language for XML*, Feb 2001.

[23] Philip Wadler. *A formal semantics of patterns in XSLT*. Bell Labs, Lucent Technologies, March 2000.

[24] *XML Query working group*. http://www.w3.org/XML/Query.