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

<table>
<thead>
<tr>
<th>Terms</th>
<th>Comp. Sc.</th>
<th>Courses</th>
<th>Seminars</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CS I</td>
<td>Algebra I and Analysis I</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CS II and Hardware Basics</td>
<td>Algebra II</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CS III</td>
<td>Graph Theory and App. Analysis</td>
<td>Programming Seminar or System Seminar</td>
</tr>
<tr>
<td>4</td>
<td>CS IV and Advanced Algorithms</td>
<td>Stochastics or Numerical Mathematics</td>
<td>Hardware Seminar or Logics Seminar</td>
</tr>
</tbody>
</table>
This is a typical XML representation of the timetable:

```xml
<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>
```
Using *Grouping Constructs* could yield the following XML representation:

```xml
<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 $T = (\text{Nodes, Edges})$ is a rooted DAG, where for every node $n \in \text{Nodes}$ there is a unique path from the root $\text{root}$ to $n$.

Definition 4.1 (elementary data tree)

An elementary data tree $\text{DT}$, with set of nodes $\text{Nodes}$, set of edges $\text{Edges}$ and root $\text{root}$, is a tree represented by the tuple $(\text{Nodes, name, children, root})$, where:

- $\text{name} : \text{Nodes} \to \text{Labels}$ is a function mapping each node to its label

- $\text{children} : \text{Nodes} \to \text{Lists(Nodes)}$ is a function such that if $(n, m) \in \text{Edges}$ then $m \in \text{children}(n)$
Definition 4.2 (data tree with grouping facets)
Given a set $G$ of grouping facets, a data tree with grouping facets is defined as a tuple $(\text{Nodes}, \text{name}, \text{children}, \text{root}, \text{grouping})$, where:

- $(\text{Nodes}, \text{name}, \text{children}, \text{root})$ is an elementary data tree
- $\text{grouping} : \text{Nodes} \rightarrow \text{Power}(G)$ 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 $G \in grouping_{DT}(N)$ and children $T_1, \ldots, T_n$ is interpreted as its correspondent forest of data trees $I(N_G)$ with root node $N$ and without $G$ as defined in the following table.

$I$ 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 $I(DT)$. 
Example:

- A
- B
- C
- D
- E

XOR

OR

Data Model
<table>
<thead>
<tr>
<th>enriched subtree $N_G$</th>
<th>interpreted as</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I(N())$</td>
<td>${N()}$</td>
</tr>
<tr>
<td>$I(N(T_1,\ldots,T_n))$</td>
<td>${N(T'_1,\ldots,T'_n) \mid T'_i \in I(T_i), 1 \leq i \leq n}$</td>
</tr>
<tr>
<td>$I(N{})$</td>
<td>$I(N())$</td>
</tr>
<tr>
<td>$I(N{T_1,\ldots,T_n})$</td>
<td>$\bigcup{I(N(T_{\pi(1)},\ldots,T_{\pi(n)})) \mid \pi \text{ permutation of } {1,\ldots,n}}$</td>
</tr>
<tr>
<td>$I(N_{\epsilon}())$</td>
<td>$I(N{})$</td>
</tr>
<tr>
<td>$I(N_{\epsilon}(T_1,\ldots,T_n))$</td>
<td>$I(N{T_1,\ldots,T_n})$</td>
</tr>
<tr>
<td>$I(N_{\text{AND}}())$</td>
<td>$I(N{})$</td>
</tr>
<tr>
<td>$I(N_{\text{AND}}(T_1,\ldots,T_n))$</td>
<td>$I(N{T_1,\ldots,T_n})$</td>
</tr>
<tr>
<td>$I(N_{\text{OR}}())$</td>
<td>$I(N{})$</td>
</tr>
<tr>
<td>$I(N_{\text{OR}}(T_1,\ldots,T_n))$</td>
<td>$\bigcup{I(N{P_1,\ldots,P_k}) \mid {P_1,\ldots,P_k} \subseteq {T_1,\ldots,T_n}, 1 \leq k \leq n}$</td>
</tr>
<tr>
<td>$I(N_{\text{ord.}}())$</td>
<td>$I(N{})$</td>
</tr>
<tr>
<td>$I(N_{\text{ord.}}(T_1,\ldots,T_n))$</td>
<td>$I(N(T_1,\ldots,T_n))$</td>
</tr>
<tr>
<td>$I(N_{\text{unord.}}())$</td>
<td>$I(N{})$</td>
</tr>
<tr>
<td>$I(N_{\text{unord.}}(T_1,\ldots,T_n))$</td>
<td>$I(N{T_1,\ldots,T_n})$</td>
</tr>
<tr>
<td>enriched subtree $N_G$</td>
<td>interpreted as</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>$\mathbf{I}(N_{\text{repeat}}())$</td>
<td>$\mathbf{I}(N{})$</td>
</tr>
<tr>
<td>$\mathbf{I}(N_{\text{repeat}}(T_1, \ldots, T_n))$</td>
<td>$\mathbf{I}(N{})$ $\cup {\mathbf{I}(N{T'_1 \circ \ldots \circ T'_n})\mid T'_i = (T_i, \ldots, T_i),</td>
</tr>
<tr>
<td>$\mathbf{I}(N_{i \text{ to } j}())$</td>
<td>$\mathbf{I}(N{})$ $\cup {\mathbf{I}(N{P_1, \ldots, P_k})\mid {P_1, \ldots, P_k} \subseteq {T_1, \ldots, T_n}, i \leq k \leq j}$</td>
</tr>
<tr>
<td>$\mathbf{I}(N_{i \text{ to } j}(T_1, \ldots, T_n))$</td>
<td>$1 \leq i \leq j \leq n$</td>
</tr>
<tr>
<td>$\mathbf{I}(N_{\text{AND}}())$</td>
<td>$\mathbf{I}(N{})$ $\cup {\mathbf{I}(N{P_1, \ldots, P_k})\mid {P_1, \ldots, P_k} \subseteq {T_1, \ldots, T_n}, i \leq k \leq j}$</td>
</tr>
<tr>
<td>$\mathbf{I}(N_{\text{AND}}(T_1, \ldots, T_n))$</td>
<td>$\mathbf{I}(N{})$ $\cup {\mathbf{I}(N{P_1, \ldots, P_k})\mid {P_1, \ldots, P_k} \subseteq {T_1, \ldots, T_n}, i \leq k \leq j}$</td>
</tr>
<tr>
<td>$\mathbf{I}(N_{\text{exclude}}()) \neg (M)$</td>
<td>$\mathbf{I}(N{})$ $\cup {\mathbf{I}(N{P_1, \ldots, P_k})\mid {P_1, \ldots, P_k} \subseteq {T_1, \ldots, T_n}, i \leq k \leq j}$</td>
</tr>
<tr>
<td>$\mathbf{I}(N_{\text{exclude}}(T_1, \ldots, T_n))$</td>
<td>$\mathbf{I}(N{})$ $\cup {\mathbf{I}(N{P_1, \ldots, P_k})\mid {P_1, \ldots, P_k} \subseteq {T_1, \ldots, T_n}, i \leq k \leq j}$</td>
</tr>
<tr>
<td>$\mathbf{I}(N_{\text{XOR}}()) \neg (M)$</td>
<td>$\mathbf{I}(N{})$ $\cup {\mathbf{I}(N{P_1, \ldots, P_k})\mid {P_1, \ldots, P_k} \subseteq {T_1, \ldots, T_n}, i \leq k \leq j}$</td>
</tr>
<tr>
<td>$\mathbf{I}(N_{\text{XOR}}(T_1, \ldots, T_n)$</td>
<td>$\mathbf{I}(N{})$ $\cup {\mathbf{I}(N{P_1, \ldots, P_k})\mid {P_1, \ldots, P_k} \subseteq {T_1, \ldots, T_n}, i \leq k \leq j}$</td>
</tr>
<tr>
<td>$\neg (M)$)</td>
<td>$1 \leq i, j \leq n, j \neq i$</td>
</tr>
</tbody>
</table>
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 $\mathcal{R} \subseteq \text{Nodes}_{DT_1} \times \text{Nodes}_{DT_2}$ is an **elementary simulation** on $DT_1$ and $DT_2$ if it satisfies

- if $n_1 \mathcal{R} n_2$, then $\text{name}(n_1) = \text{name}(n_2)$

- $\forall n_1, n'_1 \in \text{Nodes}_{DT_1} \forall n_2 \in \text{Nodes}_{DT_2}$

  $(n_1 \mathcal{R} n_2 \land n'_1 \in \text{children}(n_1) \Rightarrow \exists n'_2 \in \text{Nodes}_{DT_2} (n'_1 \mathcal{R} n'_2 \land n'_2 \in \text{children}(n_2)))$

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

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 $R \subseteq \text{Nodes}_{DT_1} \times \text{Nodes}_{DT_2}$ is a grouping simulation on $DT_1$ and $DT_2$ if it satisfies

$$\exists I_1 \in \mathcal{I}_G(DT_1) \ \exists I_2 \in \mathcal{I}_G(DT_2) \ (I_1 \text{sim}_R I_2 \Rightarrow DT_1 \text{sim}_R DT_2)$$

If $R$ is a grouping simulation on $DT_1$ and $DT_2$ with grouping, then we shall write $DT_1 \text{sim}^g_R DT_2$ instead of $DT_1 \text{sim}_R DT_2$. 
Example:

```
A
B

C
D

DT_1
XOR

A
B
C
D

DT_2
OR
XOR

A
B
C
D
E
F

Interpretations DT_1

Interpretations DT_2

1
A
B
C
D

2
A
B
C
D

3
A
B
C
D

1
A
B
C
D

10
A
B
C
D

11
A
B
C
D
```
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 $DT_1 \mathsf{sim}_{R_1} DT_2$ and $DT_1 \mathsf{sim}_{R_2} DT_2$ then $DT_1 \mathsf{sim}_{R_1 \cup R_2} DT_2$.

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

<table>
<thead>
<tr>
<th>database</th>
<th>pattern</th>
<th>combined result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε AND ε</td>
<td>ε AND ε</td>
<td>ε AND ε</td>
</tr>
<tr>
<td>OR εε</td>
<td>ε OR ε</td>
<td>OR ε</td>
</tr>
<tr>
<td>XOR εε</td>
<td>ε XOR</td>
<td>XOR ε</td>
</tr>
<tr>
<td>ε AND OR</td>
<td>AND AND</td>
<td>AND AND</td>
</tr>
<tr>
<td>XOR</td>
<td>AND XOR</td>
<td>XOR</td>
</tr>
<tr>
<td>ε OR</td>
<td>OR OR</td>
<td>OR</td>
</tr>
<tr>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
</tr>
<tr>
<td>ε XOR</td>
<td>XOR XOR</td>
<td>XOR XOR</td>
</tr>
<tr>
<td>AND</td>
<td>XOR XOR</td>
<td>XOR XOR</td>
</tr>
</tbody>
</table>

1. AND and XOR will not generate a match if the no. of elements is larger than 1.
<table>
<thead>
<tr>
<th>database</th>
<th>Grouping Facet in the pattern</th>
<th>combined result</th>
</tr>
</thead>
<tbody>
<tr>
<td>unordered ordered</td>
<td>$\epsilon$</td>
<td>unordered ordered $^2$</td>
</tr>
<tr>
<td>$\epsilon$ unordered ordered</td>
<td>unordered</td>
<td>unordered unordered $^2$</td>
</tr>
<tr>
<td>$\epsilon$ unordered unordered ordered</td>
<td>ordered</td>
<td>ordered ordered ordered $^2$</td>
</tr>
<tr>
<td>$i$ to $k$ $l$ to $m$</td>
<td>$i$ to $k$ $l$ to $m$</td>
<td>$i$ to $k$ $l$ to $m$</td>
</tr>
</tbody>
</table>

$^1$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 $m < i$
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


