From XML Schema to Object-Relational Database – an XML Schema-Driven Mapping Algorithm

Irena Mlynkova, Jaroslav Pokorny

{mlynkova,pokorny}@ksi.ms.mff.cuni.cz

Charles University

Faculty of Mathematics and Physics

Department of Software Engineering

Prague, Czech Republic

Introduction

- **XML = a key format for representing and exchanging information**
  - **Growing usage of XML technologies**
  - **Growing demand for effective management of XML documents + querying XML data**

⇒ **Possibility: Storing and managing XML data using (O)RDBS**

⇒ **Advantage: To provide XML with missing database mechanisms (i.e. indexes, multi-user access, etc.)**

Goals of the Presentation

Proposal of an algorithm for mapping

- **XML Schema structures to object-relational database schema**

Content

1. Characterization and classification
2. Mapping rules
3. DOM graph
4. Mapping algorithm
5. Storage algorithm
6. Conclusion

Existing Methods

- Significant amount
- Main difference: Designed for data-centric vs. document-centric XML documents
- Basic classification:
  - Generic – do not use any XML schema
  - Schema-driven – based on the XML schema
    - Fixed vs. flexible
    - Source schema (DTD vs. XML Schema)
    - Target schema (relational vs. object-relational)
  - User-defined – based on user-defined mapping

Classification of the Proposed Algorithm

- Fixed schema-driven mapping algorithm
- Focuses on data-centric XML documents
  - Document-centric extensions
    - Preserving the sibling order
    - Preserving the mixed-content elements
- Source schema: XML Schema
- Target schema: object-relational

Contributions of the Proposed Algorithm

- Focus on complex XML Schema structures
  - Object-oriented features, semantic constraints
  - Not very common
- Exploitation of object-relational features of SQL:1999
  - UDTs, typed tables, references, nesting, etc.
- Modelling XML Schema structures by the auxiliary DOM graph

Content

1. Characterization and classification
2. **Mapping rules**
3. DOM graph
4. Mapping algorithm
5. Storage algorithm
6. Conclusion

Mapping Rules (1)

XML Schema item		OR mapping	
Built-in simple type		SQL simple type (eventually + IC)	
User-defined simple type		SQL simple + IC	
Complex type	Attributes	UDT	Attributes: simple type
	Content: • Simple • Complex (element-subelement relationship)		Content: • Attribute: simple type • Attribute: typed column / reference / array of references
Derived complex type		Inheritance of UDTs	
Element		Own typed table / typed column of parent typed table	

Mapping Rules (2)

- Established with respect to preserve the structure and semantic constraints of the source XML schema
- Document-centric extensions:
  - Preserving the order of sibling elements
    - Naturally – using arrays and their indexes
    - Unnaturally – using additional attribute(s)
  - Preserving the mixed content of elements
    - Like “other” elements + the text parts into a separate multi-valued property

Content

1. Characterization and classification
2. Mapping rules
3. **DOM graph**
4. Mapping algorithm
5. Storage algorithm
6. Conclusion

DOM Graph

- Idea: “XML schema expressed in XML Schema language is at the same time an XML document and thus can be processed using the DOM interface as well.”
- A modification of the DOM tree
  - Nodes: Nodes of the original DOM tree
  - Edges:

    - Directed edges of the original DOM tree
    - Additional edges expressing the "direction" of the usage of globally defined items or data types

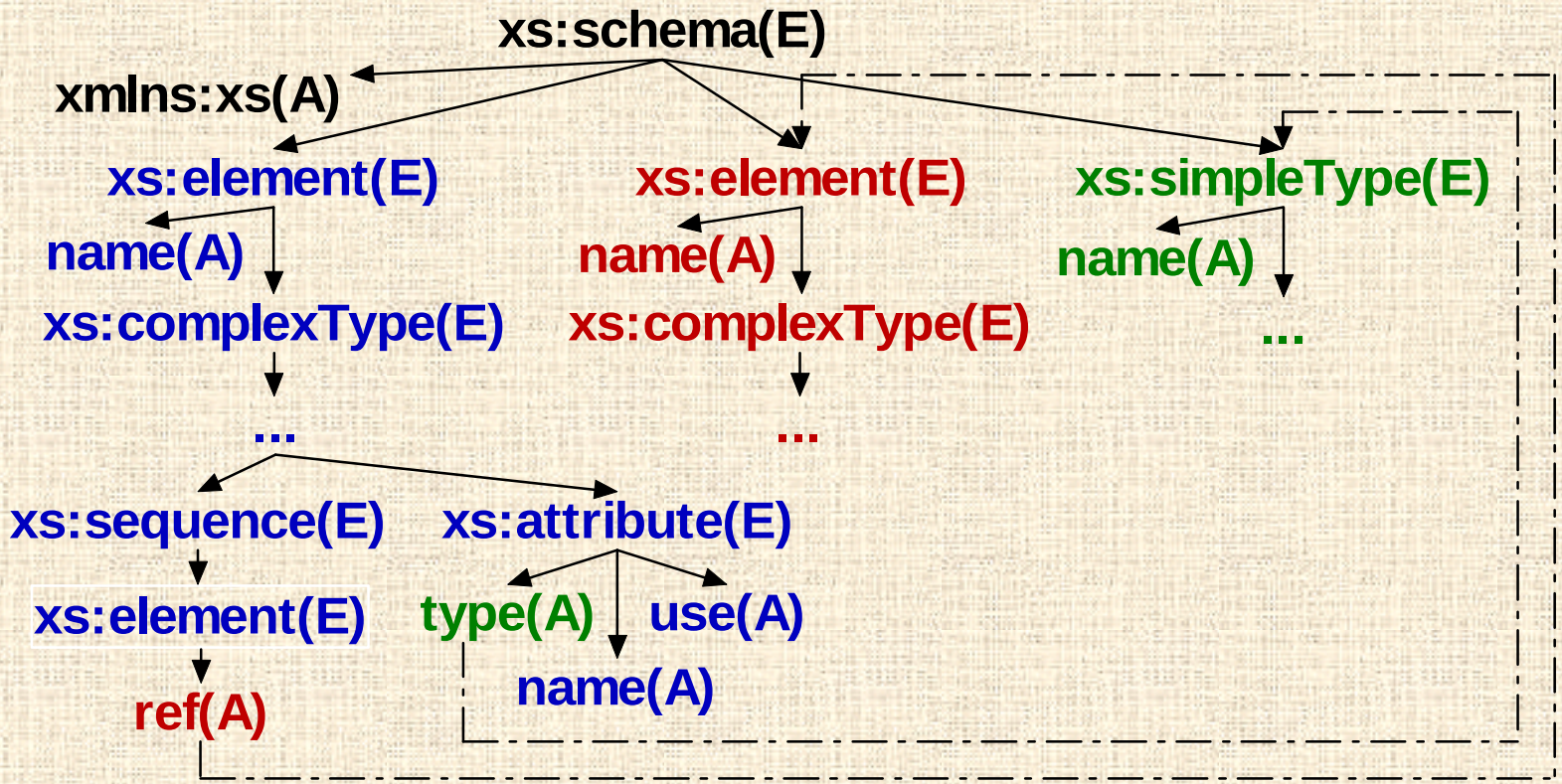
XML Schema Example

```
<xs:schema xmlns:xs="...">
  <xs:element name="Staff">
    <xs:complexType>
      ...
      <xs:sequence>
        <xs:element ref="Name"/>
      </xs:sequence>
      <xs:attribute name="OnHoliday"
                    type="YesNo"
                    use="required"/>
      ...
    </xs:complexType>
  </xs:element>

  <xs:element name="Name">
    <xs:complexType>
      ...
    </xs:complexType>
  </xs:element>

  <xs:simpleType name="YesNo">
    ...
  </xs:simpleType>
</xs:schema>
```

DOM Graph Example



Cycles in DOM Graph

- **Mutual usage of globally defined elements and/or complex types**
- **SQL:1999 allows no cyclic definitions among UDTs**
- **Existing (O)RDMS (e.g. Oracle9i) support *incomplete types***
  - **Forward declaration of a UDT declared without its content**
  - **Can be used instead of the corresponding complete type (except for inheritance ancestor)**

Content

1. Characterization and classification
2. Mapping rules
3. DOM graph
4. **Mapping algorithm**
5. Storage algorithm
6. Conclusion

Mapping Algorithm (1)

- Follows the established mapping rules
- Based on traversing the DOM graph
  - Edges determine the order in which the SQL items (i.e. UDTs, typed tables, references, etc.) should be created
- Two phases:
  1. DOM tree → DOM graph
  2. DOM graph → OR schema
- Resulting OR schema: Set of typed tables "interconnected" using references

Mapping Algorithm (2)

Phase 1. DOM tree → DOM graph

1. **Create a DOM tree of the given XML Schema file.**
2. **For each undirected edge create the corresponding directed one.**
3. **For each base, type, itemType or ref attribute create the corresponding (additional) directed edge.**

Mapping Algorithm (3)

Phase 2. DOM graph → OR schema

1. For each globally defined complex type / element create its incomplete UDT.
2. Starting in the root node process all nodes in the graph recursively:
   1. Process all child nodes of a current node (i.e. create corresponding SQL item / items).
   2. Process the current node.
3. For each incomplete UDT, create corresponding complete one.
4. Create corresponding typed tables.

Content

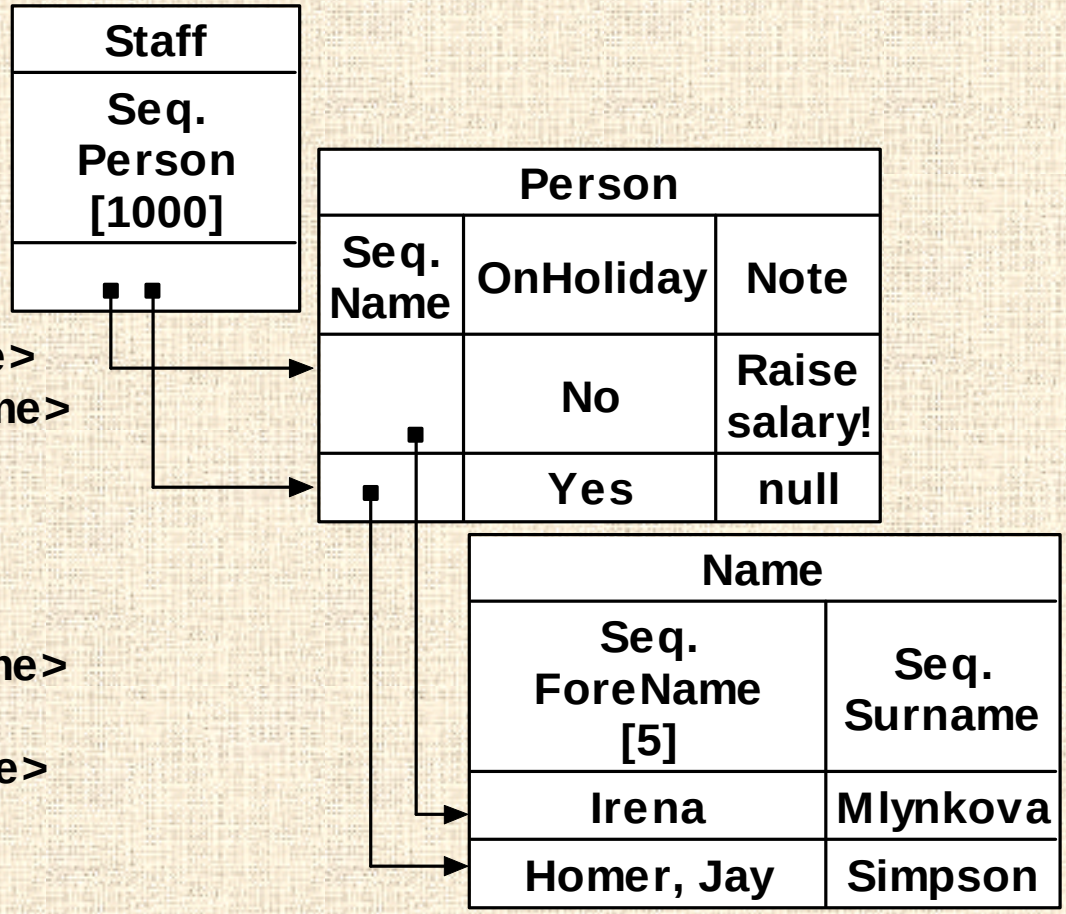
1. Characterization and classification
2. Mapping rules
3. DOM graph
4. Mapping algorithm
5. **Storage algorithm**
6. Conclusion

Storage Algorithm

- Storing the data from XML documents valid against the source XML schema into relations of the target OR schema
- Based on traversing the DOM tree of the XML document and creating SQL constructors of the data
- Driven by:
  - Structure of XML document
  - Auxiliary information about the current schema
    - Types of parent-child mapping, names of tables, etc.

Example

```
<Staff>
  <Person OnHoliday="No"
          Note="Raise salary!">
    <Name>
      <ForeName>Irena</ForeName>
      <Surname>Mlynkova</Surname>
    </Name>
  </Person>
  <Person OnHoliday="Yes">
    <Name>
      <ForeName>Homer</ForeName>
      <ForeName>Jay</ForeName>
      <Surname>Simpson</Surname>
    </Name>
  </Person>
</Staff>
```



Content

1. Characterization and classification
2. Mapping rules
3. DOM graph
4. Mapping algorithm
5. Storage algorithm
6. **Conclusion**

Conclusion

- Contributions
  - Focus on object-oriented features of XML Schema
  - Exploitation of object-relational features of SQL:1999
  - Definition of DOM graph
- Future work:
  - Optimalizations of the created schema
    - For a certain application – flexible methods
    - Generally – definition of a “good” XML schema (such as e.g. normal forms for relations) and ways how to establish it