EXERCISE

Define the referential integrity constraints to the following fragment of MONDIAL:

```
<xs:element name="country">
  <xs:attribute name="car_code" type="mondial:carcode"/>
  <xs:attribute name="capital"  type="string" />
  <xs:keyref> ... </xs:keyref>
</xs:element>
<xs:key> ... </xs:key>

<xs:element name="city">
  <xs:attribute name="country" type="mondial:carcode"/>
  <xs:element   name="name"       type="string"/>
  <xs:keyref> ... </xs:keyref>
</xs:element>
<xs:key> ... </xs:key>

<country car_code="D" capital="Berlin"/>
<city in_country="D">
  <name>Berlin</name>
</city>
```


Chapter 11
Implementation and Algorithms

- XML as a data structure:
  - *abstract datatype* with API: DOM
  - (mainly main-memory) implementations
  - low-level API with variable-based access
- Databases?
  - high-level API: XPath, XQuery
  - mapping to relational model (Oracle, IBM DB2) or ObjectTypes (Oracle, DB2)
  - “Native” storage: Software AG-Tamino
  - database functionality: multiuser, transactions, recovery
- Stream Processing:
  - XML data transfer as sequence of events
  - SAX (Simple Application Interface for XML)

11.1 DOM

- DOM (Document Object Model) defines a platform- and language-independent object-oriented *interface* (i.e., an *abstract datatype*) for generating, processing and manipulating XML data.
- language base of the DOM specification: OMG-IDL
- implementations in Java, C++, etc; usually main-memory-based;
  specialized Java interface definition: JDOM
- DOM *specifies* a *logical* data model (see "XML as data model"), where XML data is represented as trees:
  - document: represents the complete document,
    - Query-Methods, e.g. NodeList getElementsByTagName(string)
  - class "Node": getNodeType(), getChildren(), getFirstChild(), getNextSibling(), getParentNode(), ...
  - class "Element": getTagName(), getattributes(), ...
  - class "attribute": getName(), getValue(), ...
  - corresponding methods for generating and changing nodes.

DOM: Functionality Levels

- DOM is a specification of an interface/abstract datatype, *not a* data model and *not a* programming language!

Definition of DOM "Levels" with increasing functionality:

- DOM Level 1: Basics
- DOM Level 2: + namespaces
- DOM Level 3: + handling of DTDs, XML Schema, validation ...

... everything you need to program with XML in programming languages like Java.

DOM: Principles

- only one document in a DOM
- step-by-step-access to the data:
  based on variable assignments in the surrounding imperative/object-oriented programming language and on iterators (cf. proceeding in the network data model)
- no declarative access
- based on DOM, XPath and XQuery can be implemented (cf. Apache Xerces (XML/DOM)/Xalan (XSLT)/Xindice (DB))
- Main-memory-based: only for small application programs
  (most of the "lightweight"-tools used in the course are internally based on DOM)
- often inefficient (no indexes, query optimization)

⇒ databases must follow other ideas: what *physical data model* is best suited for XML?

11.2 SAX

SAX (The Simple API for XML) is another, *event-based interface/model*:

Serialized approach to XML:

- represents/processes an XML document as a sequence of events (depth-first traversal), e.g.
  - startDocument(), endDocument()
  - startElement(Name, attributesList) – attributes not split
  - endElement(Name)
  - characters(string,int,int)
- for SAX-based programming, an event handler is written that reacts on the incoming events and does something (application-dependent)
- SAX was the first API for XML – every XML application at that time was programmed as a state machine.