XML

What is XML:

- data exchange format
- unstructured/semistructured data
- extensible markup language (a subset of SGML)

XML documents:

- contain user-defined markup (tags) and text
- markup: elements, attributes
- self-describing
- can have a schema or be schemaless
- (most) can be viewed as ordered, node-labelled trees

Other features: entities, namespaces, processing instructions, comments...

Well-formed document:

- contains properly nested elements with a single root element
- can contain empty elements, mixed text and elements

Valid document:

- verifies a Document Type Definition (DTD) or an XML Schema
- DTD: context-free grammar
- XML Schema: type definitions, constraints

DTDs

Elements:

`<!ELEMENT` elementName `(contentModel)>`

Content models:

- element-only content:

  ```
  <!ELEMENT book(title,isbn,note,author*,publisher?,section+)>
  ```

- text-only content: `<!ELEMENT title (#PCDATA)>`

- mixed content:

  ```
  <!ELEMENT section (#PCDATA | title | section)*>
  ```

- empty content: `<!ELEMENT isbn EMPTY>`

- unrestricted content: `<!ELEMENT note ANY>`

Attributes:

`<!ATTLIST` elementName attributeDefinitions`>`

Attribute definitions:

attributeName attributeType defaultDeclaration

Attribute types:

- text: `CDATA`
- identifier (values unique in the document): `ID`
- identifier reference: `IDREF`, `IDREFS`
- enumerated value set: `(value | ...)`
- ...

Default declarations:

- required: `#REQUIRED`
- optional: `#IMPLIED`
- default, can be overridden: “value”
- default, fixed in the DTD: `#FIXED` “value”

XPath

Notation for XML document navigation and node selection.

Used in XQuery, XSLT, XPointer,...

Data model:

- tree-based
- nodes: root, element, attribute, text,...
- document order: left-to-right prefix traversal

XPath expressions

A *path expression*:

- describes a set of paths in a document
- returns a sequence of nodes in document order
- evaluated in a *context*
- absolute (starting at root) or relative
- consists of steps separated by `/`
- wildcards
- union (`|`), intersection, difference

Context

Consists of:

- node
- size
- position
- ...

Path steps

Axis step

axis::nodeTest stepQualifiers

consists of:

- an *axis*:
  - *forward*: `child`, `descendant`, `following-sibling`, `following`, `self`, `descendant-or-self`
  - *backward*: `parent`, `ancestor`, `preceding-sibling`, `preceding`, `ancestor-or-self`
  - `attribute`
- a *node test*: name test (name or wildcard), kind test
- step *qualifiers*: predicate expressions (in square brackets)

Abbreviated syntax

1. `child` is the default axis, can be omitted
2. the `attribute` axis can be abbreviated to `@`
3. `//` is short for `/descendant-or-self::node()/`
4. `.` is short for `self::node()`
5. `..` is short for `parent::node()`
6. a positive integer `K` is short for `[position()=K]`

XQuery

Query language for XML databases.

Features:

- functional
- compositional: made of base expressions that can be nested arbitrarily
- recursion
- declarative: influenced by SQL (and OQL)

XQuery expressions

- Constants: numbers, strings,...
- Variables
- XPath expressions
- Element/attribute constructors
- Operators and functions: arithmetic,...
- FLWOR expressions
- Quantifiers
- Aggregation
- User-defined functions

FLWOR expressions

`for` *variableRangeSpecifications*
`let` *variableDefinitions*
`where` *condition*
`order by` *orderExpression*
`return` resultExpression

Notes:

- aggregate functions: applied to expression results.
- quantifiers: `some`, `every`.
- the condition can refer to document order.

Functions

User-defined functions:

`declare function` Name(Arguments)
`as` Type
{Expression}

Storing XML documents in relational databases

Storing nodes and edges of the document tree:

- a binary `edge` relation
- implementing XPath requires recursion (SQL3)

Encoding the tree structure using *ranges*:

- range of child $\subset$ range of parent
- queries w/o recursive functions can be translated to SQL2