Module 3

XML Processing

(XPath, XQuery, XUpdate)

Part 2: XML Data Models

Data Models for XML - Introduction

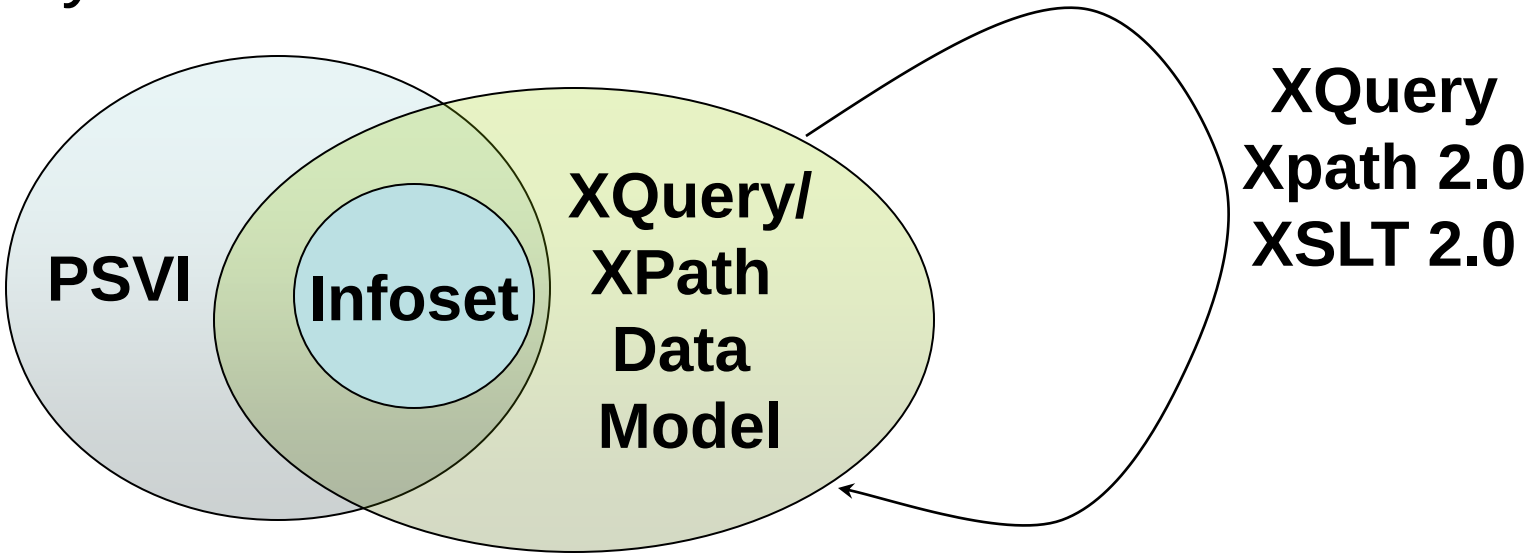
- We looked at syntactic aspects of XML so far
- Need for a abstract description for processing
  - Hide differences of representations (&auml vs. ä, …)
  - Simplify interactions with data
  - Formally define the data items and provide the basis for operations on them
- Same role for APIs/query+transformation languages as the relational data model for SQL
- Purely logical --- no standard storage or access model (on purpose)

Components of Data Processing Stack

	Relational	XML	
Query Language	SQL-DML	XQuery, XSLT 2.0, XPath 2.0	None
Data Model	Relations	XQuery Data Model	InfoSet
Metadata	SQL-DDL	DTD, XML Schema	
Data Syntax	None	XML	

XML Data Models - Overview

- Multiple data models for XML exist
  - **InfoSet / PSVI (P**ost-**S**chema-**V**alidation **I**nfoset**)**
  - API oriented (DOM, SAX, JDOM, …)
  - **XPath 2.0/XQuery/XSLT (XDM)**
  - …
- XQuery/XSLT 2.0/XPath 2.0 closed wrt XDM



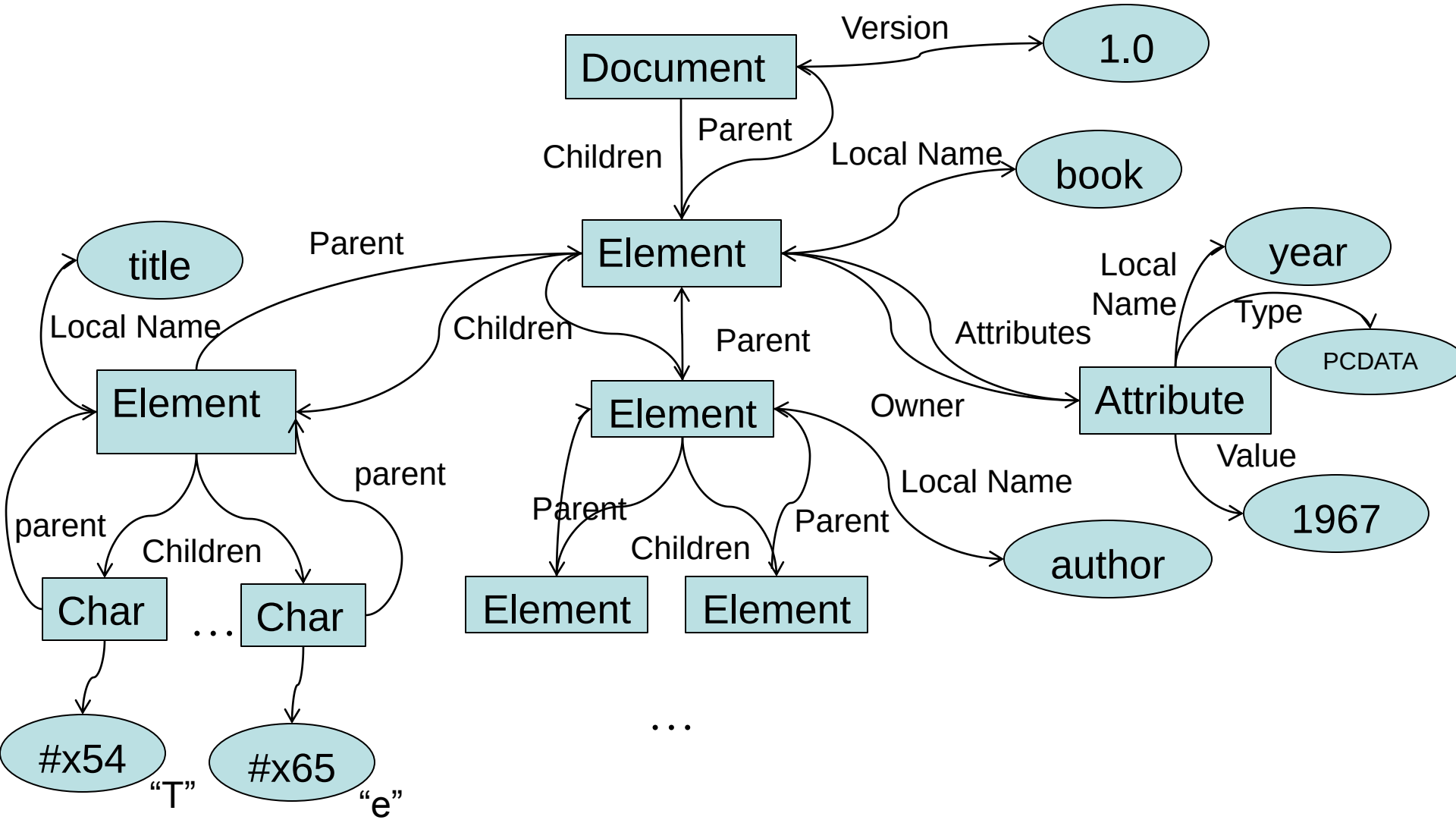
XML Infoset

- The first and most popular XML data model
- Nodes of XML tree: Information Items
- There are 11 Information Items
  - Document, Element, Attribute, ... (+DTD-related, e.g. Entity)
- Associate Properties to each Information Item
- There are more than 10 Properties
  - Children, base URI, version, namespace, ...
  - Properties define InfoItem uniquely
- Different InfoItems have different Properties
- Schema Validation impacts Values of Properties
- InfoSet can be generated from and serialized into "non-XML" representations

Example of InfoSet

```
<book year="1967">
    <title>The politics of experience
    </title>
     <author>
            <firstname>Ronald</firstname>
            <lastname>Laing</lastname>
     </author>
</book>
```

(Incomplete) Infoset Visualization



InfoSet: Document InfoItem

- Children: List of InfoItems
  - InfoItems of PIs, Comments, Root element
  - Example: InfoItem of [book]
- Document Element: Element InfoItem
  - Example: InfoItem of [book]
- Notation, Unparsed entitities: empty in example
- Base URI: URI of the Document
  - Warning: "streaming data" (e.g. SOAP Message)
- Standalone: empty (can also be "yes" or "no")
- Version: in Example, 1.0
- Encoding scheme: default is UTF-8

Element InfoItem (e.g. [book])

- Namespace Name: if element type is NS
- Local name: in Example: "book"
- Prefix: in Example: empty
- Children: in Example: [title] and [author]
- Attributes: in Example: [year]
- Namespace Attributes: local NS Definitions
- In-scope Namespaces: applicable NS Defs, ...
- Base-URI: ...
- Parent: Example: Document InfoItem

Element InfoItem [title]

- Title has as "children" a Char Info Item (Text)
- (everything else analogous to [book])
- The Char Info Item has Properties
  - Value of the Text (“T”) in ISO 10646 Code (Unicode)
  - Whitespace handling (yes or no)
  - Parent is [title] in this example

Attribute InfoItem (e.g. [price])

- Namespace Name, local name, prefix: ...
- Normalized value:
  - Value after *Normalization (cleaning up whitespace)*
  - Example: “1967”
- Attribute Type:
  - Only DTD types allowed
  - Example: PCDATA
- References:
  - If IDREFs, the corresponding InfoItems
  - Example: empty
- Owner Element: [book] in example

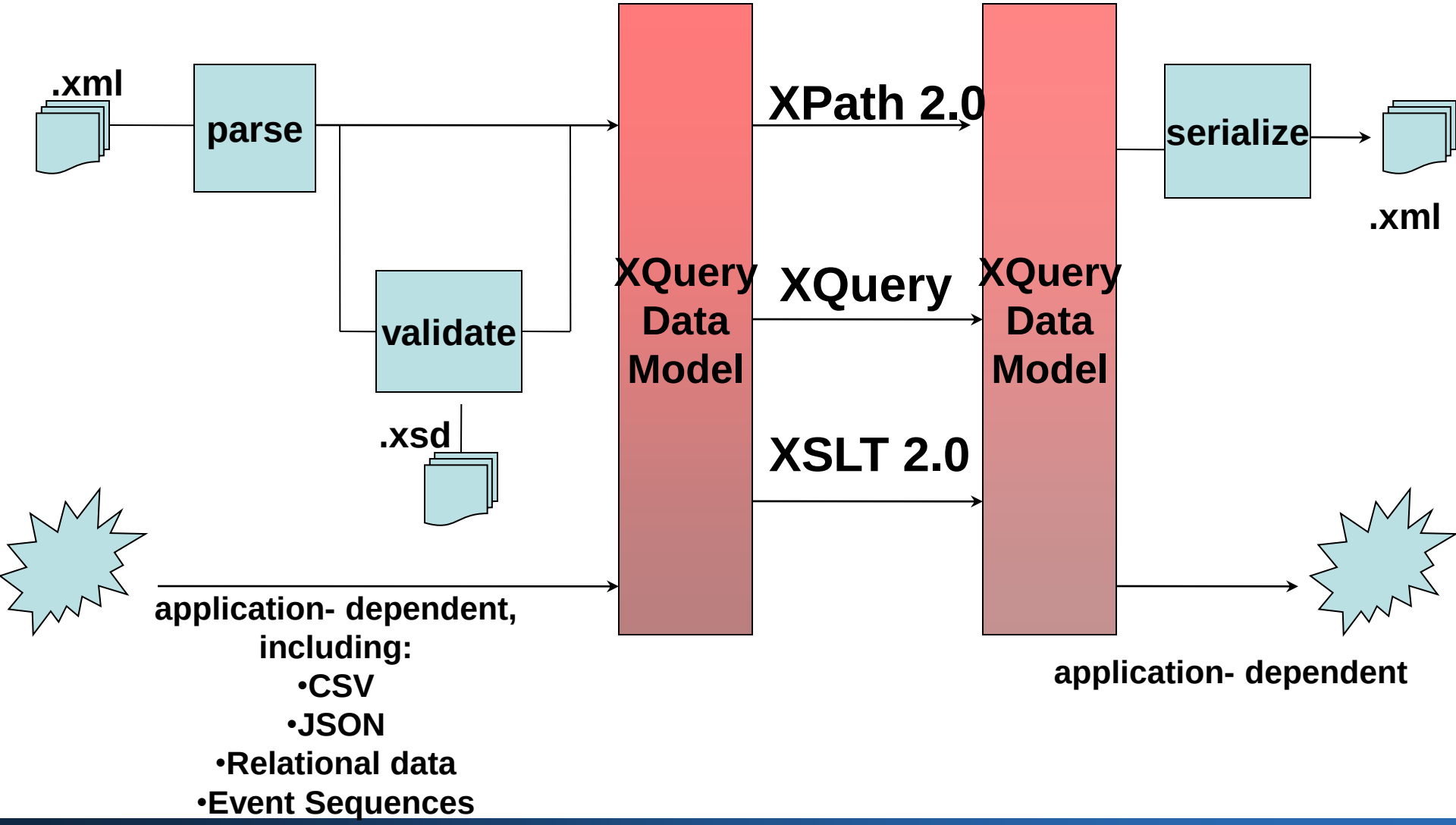
Reasons for XQuery Data Model (XDM)

Unify different Data Models

- InfoSet
- XPath 1.0 Data Model

- Support results that are not XML
  - Forests of nodes
  - Arithmetic expressions
- Represent intermediate results in same data model
- (See XQuery from the Experts, chapter 2)

XDM Life Cycle



XQuery Data Model

- Instance of the data model:
  a *sequence* composed of zero or more *items*
- The *empty sequence* often considered as the "null value"
- Item: *node* or *atomic value*
- Nodes: all 7 XML node types
- Atomic values
  - Instances of all XML Schema atomic types: `string, boolean, ID, IDREF, decimal, QName, URI, ...`
  - Untyped atomic values
- Typed (i.e. schema validated) and untyped (i.e. non schema validated) nodes and values

Sequences

- Can be heterogeneous (nodes *and* atomic values)

  (<a/>, 3)
- Can contain duplicates (by value and by identity)

  (1,1,1)
- Are not necessarily ordered in document order
- Nested sequences are automatically flattened

  (1, 2, (3, 4)) = (1, 2, 3, 4)
- Single items and singleton sequences are the same

  1 = (1)

Atomic values

- The values of the 19 *atomic types* available in XML Schema
  - E.g. xs:integer, xs:boolean, xs:date
- All the *user defined derived atomic types*
  - E.g myNS:ShoeSize
- xs:untypedAtomic
- Atomic values carry their type together with the value
  - (8, myNS:ShoeSize) is not the same as (8, xs:integer)

XML nodes

- 7 types of nodes:
  - document | element | attribute | text | namespaces | PI | comment
- Every node has a unique *node identifier*
  - Scope of node identifier uniqueness is implementation dependent
- Nodes have children and an optional parent
  - conceptual "tree"
- Nodes are ordered based on the topological order in the tree ("document order")

Node accessors

Properties of XDM nodes can be accessed using functions:

- node-kind : xs:string (“document”, “element”, …)
- node-name : xs:QName ? (“book”)
- parent : node() ?
- string-value : xs:string (“001”)
- typed-value : xs:anyAtomicType * (1)
- type-name : xs:QName ? (“xs:integer”)
- children : node()*
- attributes : attribute() *
- namespaces : node() *
- …

Untyped (=non-validated) XML in XDM

```
<book year="1967" >
     <title>The politics of experience</title>
     <author>R.D. Laing</author>
</book>
```

- 3 element nodes, 1 attribute node, 2 text nodes
  - name(book element) = {-}:book
- In the absence of schema validation
  - type(book element) = xs:untyped
  - type(author element) = xs:untyped
  - type(year attribute) = xs:untypedAtomic
  - typed-value(author element) = ("R.D. Laing" , xs:untypedAtomic)
  - typed-value(year attribute) = ("1967", xs:untypedAtomic)

XML schema for Type Information

```
<type name="book-type">
  <sequence>
   <attribute name="year" type="xs:integer"/>
   <element name="title" type="xs:string"/>
   <sequence minoccurs="0">
     <element name="author"type="xs:string/>
   </sequence>
  </sequence>
</type>
<element name="book" type="book-type">
```

Schema validated XML data in XDM

```
<book year="1967" >
    <title>The politics of experience</title>
    <author>R.D. Laing</author>
</book>
```

- After schema validation
  - type(book element) = {uri}:book-type
  - type(author element) = xs:string
  - type(year attribute) = xs:integer
  - typed-value(author element) = ("R.D. Laing" , xs:string)
  - typed-value(year attribute) = (1967 , xs:integer)
- Schema validation impacts the data model representation and therefore the XQuery semantics!!

Lexical and binary aspect

- Every node holds (logically) redundant information:

  <a xsi:type="xs:integer">001

  - dm:string-value () "001" as xs:string
  - dm:typed-value ()
    - "001" as an xdt:untyped *before* validation
    - 1 as an xs:integer *after* validation
- Implementations can store：
  - The string value
    - Retrieve the typed value dynamically based on the type, every time is needed
  - The typed value
    - Retrieve an acceptable lexical value for that type every time this is required
  - Both
- In case of unvalidated data the two are the same