University of Mila

Department of computer science

3rd Year Computer Science 2025 / 2026

Semi-Structured Data
Chapter 3 — XML Core Lab Serie

Exercise 1: A public library wants to store its book catalog in XML. Each book follows strict ordering rules and some fields are optional.

Structure rules:

- The root element <library> contains a name element, followed by one or more <book> elements.
- Each <book> contains (in order): <title>, <subtitle> (optional), one or more <author> elements, <year>, zero or more <genre> elements, and a <description>.
- All leaf elements contain text.
- <book> has a language attribute (default value 'Arabic') and an edition attribute (optional).

Questions:

1. Write the complete internal DTD. Use the correct cardinality operator for each element.
2. Write a minimal valid XML instance (only required fields) and a full XML instance (all fields present).
3. What does the DEFAULT keyword do? What is the difference between DEFAULT and #IMPLIED for the edition attribute?

Exercise 2: An online shop needs an XML format for customer orders. Payment can be made by card, PayPal or bank transfer — but only one method per order.

Structure rules:

- Root <orders> contains one or more <order> elements.
- Each <order> contains (in order): <customer>, <items>, <payment>, and an optional <note>.
- <customer> contains: <name>, <email>, <phone> (optional).
- <items> contains one or more <item> elements. Each <item> has a <productName>, <quantity>, and <unitPrice>.
- <payment> contains exactly ONE of: <card>, <paypal>, or <bankTransfer>. Each contains only text.
- <order> has a required status attribute that must be one of: pending, confirmed, shipped, delivered, cancelled.
- <item> has a required currency attribute: DZD, EUR, or USD.

Questions:

1. Write the complete DTD (internal format).
2. Write a valid XML instance: one order paid by card (status='confirmed'), one paid by PayPal (status='pending').
3. What error occurs if an <order> includes both <card> and <paypal> inside <payment>?
4. Can an ENUMERATION attribute have a DEFAULT value? Write an example.

Exercise 3: A university stores its staff and departments in XML. Relationships between staff members and departments must be modeled with ID/IDREF links.

Structure rules:

- Root <university> has a required name attribute.
- <university> contains: one or more <department> elements, then one or more <staff> elements.
- Each <department> has: deptID, deptName. Contains a <location> text element.

- Each <staff> has: staffID, deptRef (references a department's deptID), managerRef (references another staffID, optional).
- Each <staff> contains (in order): <fullName>, <role>, <email> (optional).
- A <project> element has a teamRefs attribute of type IDREFS listing the IDs of all staff members on the team.

Questions:

1. Write the complete DTD.
2. Write a valid XML instance: 2 departments, 3 staff members, 1 project involving 2 of them.
3. What is the rule that ID values must satisfy across the whole document?
4. What happens at validation time if an IDREF points to a value that does not exist as an ID?
5. Why can IDREFS not be used as an element type — only as an attribute type — in DTD?

Exercise 4: A warehouse uses XML to track product inventory. The document format has strict rules about attribute defaults and fixed values.

Structure rules:

- Root <inventory> has a fixed version attribute = '2.0' and a fixed format attribute = 'XML'.
- <inventory> contains one or more <product> elements.
- Each <product> has: pid, category (electronics, clothing, food, tools, default 'tools'), status (active, discontinued, required), warehouse (text).
- Each <product> contains: <name>, <stock> (text), <price>, an optional <image> (EMPTY element with a src attribute), and an optional <notes> element declared as ANY.

Questions:

1. Write the complete DTD.
2. What is the effect of #FIXED? What happens if an XML author writes version='1.0' in the instance?
3. Write a valid XML instance with at least 2 products — one minimal, one with all optional elements.
4. When is the ANY content model useful? What is its disadvantage?
5. Demonstrate the EMPTY element <image> with a self-closing tag in your XML instance.

Exercise 5: write the DTD from scratch.

Requirements:

- Root: <conference> with attributes: title, year, edition (FIXED '1'), location (DEFAULT 'Online').
- <conference> contains: one or more <speaker> elements, then one or more <session> elements.
- Each <speaker>: speakerID, affiliation. Children (in order): <name>, <bio> (optional).
- Each <session>: sessionID, type (keynote, workshop, panel, REQUIRED), room. Children: <sessionTitle>, then a CHOICE between <talk> or <roundtable> (one or more), then optional <break>.
- Each <talk>: talkID, speakerRef. Children: <talkTitle>, <abstract>, optional <slides> (EMPTY with url attribute).
- Each <roundtable>: panelRefs (lists speaker IDs). Children: <topic>, zero or more <question>.
- <break> is EMPTY with a duration attribute (DEFAULT '15min').

Questions:

1. Write the complete DTD.
2. Write a valid XML instance: 2 speakers, 1 keynote session (1 talk), 1 panel session (1 roundtable).
3. A session has type='workshop' but contains a <roundtable>. Is this a DTD validation error? Explain.

Exercise 6: Define stand-alone simple types with restrictions for a student registration system. Each type must be defined as a named simpleType so it can be reused.

Types to define:

- studentIDType — a string matching the pattern: S-[0-9]{6} (e.g. S-004523).
- gradeType — a decimal between 0.00 and 20.00 with at most 2 fraction digits.
- yearOfStudyType — an integer restricted to the values 1, 2, or 3 only (use enumeration).
- emailType — a string matching the pattern: .+@.+\..+ .
- birthYearType — an integer between 1980 and 2010.
- genderType — one of: M, F.

Questions:

1. Write all six named simple type definitions inside a <xs:schema> root.
2. Declare a <student> element whose child elements use these types.
3. Write one valid and one invalid XML value for each type and explain the error.

Exercise 7: Model a patient record system using complex types with sequences. Focus on correct ordering of child elements and attribute declarations.

Structure rules:

- Define a named complex type addressType with a sequence: street, city, wilaya (text), postalCode (string, 5 digits pattern).
- Define a named complex type patientType with a sequence: patientID (string), fullName, dateOfBirth (xs:date), gender (use genderType from Ex.1 or re-declare inline), address (of type addressType), bloodGroup (one of A+,A-,B+,B-,AB+,AB-,O+,O-).
- patientType has attributes: active (xs:boolean, default true), source (xs:string, use='optional').
- Declare a root element <patientRecord> of type patientType.

Questions:

1. Write the complete XSD.
2. Write a valid XML instance for a patient.
3. In a sequence, can elements appear in any order? What happens if <city> appears before <street>?
4. What is the difference between use='required', use='optional', and a default attribute value in XSD?

Exercise 8: Travel booking system.

Structure rules:

- Define a personInfoType using xs:all (all 3 must appear, any order): firstName, lastName, passportNumber. Make lastName optional within xs:all
- Define a travelModeType using xs:choice: exactly one of <flight>, <train>, or <bus>. Each contains a text element <routeCode>.
- Define a bookingType using xs:sequence: personInfo (personInfoType), travelMode (travelModeType), departureDate (xs:date), returnDate (xs:date, minOccurs='0'), totalPrice (xs:decimal).
- Add an attribute bookingRef (xs:string, required) and a status attribute (booked, cancelled, pending, default 'pending').

Questions:

1. Write the complete XSD with all three compositor types (sequence, all, choice).
2. Write 3 XML instances: one with flight, one with train (return date omitted), one with bus.
3. Can xs:all contain elements with maxOccurs > 1? What constraint does XSD impose on xs:all?

Exercise 9: A school report card system requires precise control over how many times each element can appear.

Requirements:

- Root <reportCard> contains: exactly one <studentInfo>, exactly one <schoolYear> (string), one or more <subject> elements, zero or one <teacherComment>, and zero to three <award> elements.
- Each <subject> has: <subjectName> (exactly once), <coefficient> (exactly once), one or more <examResult> elements, and zero or more <absenceDate> (xs:date) elements.
- Each <examResult> has a type attribute (midterm|final|quiz, required) and contains a <score> (gradeType: 0–20) and an optional <remark>.
- <studentInfo> contains: <studentID>, <fullName>, <class>, optional <photo> (xs:anyURI).

Questions:

1. Write the complete XSD, using minOccurs/maxOccurs explicitly on every element.
2. How do you express 'between 2 and 5 occurrences' in XSD? Write an example element declaration.
3. Write a valid XML instance for a student with 2 subjects (one has 2 exam results, the other has 3).

Exercise 10: Build a complete XSD from scratch for a digital library.

Specification:

- Named simple types needed: isbnType (pattern 978-[0-9]{10}), yearType (integer 1000–2025), ratingType (decimal 0.0–5.0, 1 fraction digit), languageCode (2-letter string: pattern [a-z]{2}).
- Named complex type: publisherType — xs:all containing: <publisherName>, <country>, optional <website> (xs:anyURI). Add attribute pubID (xs:string, required).
- Named complex type: authorType — xs:sequence: <firstName>, <lastName>, optional <nationality>. Attribute authorID (xs:string, required).
- Root element <elibrary> — xs:sequence: one or more <author>, one or more <publisher>, one or more <book>.
- <book> — xs:sequence: <isbn> (isbnType), <bookTitle>, one to three <authorRef> (xs:string each — references authorID values), <publisherRef> (xs:string), <year> (yearType), <language> (languageCode), optional <rating> (ratingType), zero or more <tag> (xs:string), a CHOICE of <ebook> or <physical> (each containing details).
- <ebook>: xs:sequence: <fileFormat> (ENUMERATION: pdf|epub|mobi), <fileSizeKB> (xs:positiveInteger). Attribute downloadable (xs:boolean, default true).
- <physical>: xs:sequence: <pages> (xs:positiveInteger), <binding> (ENUMERATION: hardcover|paperback|spiral). Attribute inStock (xs:boolean, required).
- <book> attributes: bookID (xs:string, required), accessLevel (ENUMERATION: free|premium|restricted, default free).

Questions:

1. Write the complete XSD. Use named types where specified; use anonymous inline types elsewhere.
2. Write a valid XML instance: 2 authors, 1 publisher, 2 books (one ebook, one physical).
3. Identify in your XSD: one example each of sequence, all, choice, simpleType restriction, attribute with default, attribute with use='required', minOccurs/maxOccurs, and xs:enumeration.
4. A student wrote <xs:all> for the book's children to allow any order. List two reasons why xs:all would NOT work for <book> in this schema.