University of Biskra. Academic year: **2024/2025**
Faculty of Exact Sciences Level: **3rd Year**
Computer science Department. Module : **SSD**

Exam

Date : May 12th, 2025 **Time : 1h30**

Questions (05 Marks):

1. Provide an example of how the following table would be represented in XML format.

name	phone	email
alan	3127786	alan@abc.com
sara	2136877	sara@xyz.edu
fred	7786312	fd@ac.uk

2. Why do we use an XML parser? (Give at least 2 reasons)
3. List the advantages of JSON over XML (Provide at least 2 answers)
4. List the advantages of XML over JSON (Provide at least 2 answers)

Exercise 1 (03 Marks) : Consider the following two DTDs:.

<!—First DTD -->	<!—Second DTD -->
<!ELEMENT root ((a,b*,a)|b)*>	<!ELEMENT root (a|(b,a*,b))*>
<!ELEMENT a (#PCDATA)>	<!ELEMENT a (#PCDATA)>
<!ELEMENT b (#PCDATA)>	<!ELEMENT b (#PCDATA)>

1. Give an XML document that is valid for the first DTD but invalid for the second.
2. Provide an XML document that is valid for the second DTD but invalid for the first DTD.

Exercise 2 (12 Marks):

We consider XML documents that describe the internal organization of a company. Such a document is structured according to the following rules:

- The root tag is organization.
- The root tag has two child elements, departments and employees (in that order).
- The departments tag contains a child element named department for each department described in the document.
- The employees tag contains a child element named employee for each employee described in the document.

Each department is described as follows:

- It has an attribute named *num* serving as its identifier.
- It contains a child element specifying its name (for example; Management, Accounting, Communication, ….).
- It may contain an empty child element named manager with an attribute name, referencing an employee.

Each employee is described as follows:

- He has an id attribute serving as its identifier.
- He has a name element containing their family name.
- He may have a first name element.
- He has an activities element describing the departments they work in. This element contains empty activity elements, each with an attribute referencing a department defined elsewhere.
- He may have an optional details element, which can include age and gender elements (each optional).

1. Provide a DTD describing the XML documents used for company organization.
2. Provide an XML Schema describing the same XML documents.
3. Write XPath queries to retrieve specific data:
   a) Find the last names of employees who are 50 years old or older.
   b) Get the last names of employees who work in the Accounting department (identified by department ID " Account").
   c) List the IDs of employees who work in two or more departments.
   d) Count how many employees are involved in the Management department (identified by department ID " manag ").

We want to flatten the original document, meaning it should only contain the services section, but with employees explicitly listed within each service. Additionally, for this task, we will not retain any information about managers persons.

For illustration, here's the expected output for the first service (management):

```
<service num=" manag ">
  <nom> management </nom>
  <employes>
    <employe>
      <nom>Nixon</nom>
      <prenom>John</prenom>
    </employe>
    <employe>
      <nom>Peter</nom>
      <details>
        <age>35</age>
      </details>
    </employe>
  </employes>
</service>
```

4. Write an XSLT code that generates the required document.
5. Provide an XQuery query to construct the requested document.

University of Biskra.
Faculty of Exact Sciences
Computer science Department.

Academic year: **2024/2025**
Level: **3rd Year**
Module : **SSD**

Corrigé type

Questions (05 Marks):

1. Example of the table in XML format:

```
<contacts>
    <contact>
        <name>alan</name>
        <phone>3127786</phone>
        <email>alan@abc.com</email>
    </contact>
    <contact>
        <name>sara</name>
        <phone>2136877</phone>
        <email>sara@xyz.edu</email>
    </contact>
    <contact>
        <name>fred</name>
        <phone>7786312</phone>
        <email>fd@ac.uk</email>
    </contact>
</contacts>
```

2. Why do we use an XML parser? (Give at least 2 reasons)
   - The XML Parser is designed to read the XML and create a way for programs to use XML.
   - XML parser validates the document and check that the document is well formatted.
3. Advantages of JSON over XML (Provide at least 2 answers)
   - **Lightweight**: JSON has a simpler and more compact syntax, leading to smaller file sizes and faster parsing.
   - JSON is natively supported in JavaScript, making it ideal for web applications and APIs.
   - JSON's structure is more intuitive and easier for humans to understand compared to XML's verbose syntax.
   - Parsing JSON is generally faster than parsing XML due to its simplicity.

4. Advantages of XML over JSON (Provide at least 2 answers)
    1. XML supports attributes, namespaces, and complex hierarchical data better than JSON.
    2. XML can be validated against DTD or XML Schema, ensuring strict data format compliance.
    3. XML allows attributes alongside elements, making it easier to include metadata.
    4. XML has extensive support for transformations (XSLT), queries (XPath), and other advanced features.

Exercise 1 (03 Marks) :

1. XML Valid for First DTD but Invalid for Second DTD

```
<root>
  <a>…</a>
  <b>…</b>
  <a>…</a>
</root>
```

2. XML Valid for Second DTD but Invalid for First DTD

```
<root>
  <b>…</b>
  <a>…</a>
  <b>…</b>
</root>
```

Exercise 2 (12 Marks):

1. Provide a DTD describing the XML documents used for company organization.

```
<!ELEMENT organization (departments, employees)>
<!ELEMENT departments (department+)>
<!ELEMENT employees (employee+)>
<!ELEMENT department (name, manager?)>
<!ATTLIST department num ID #REQUIRED>
<!ELEMENT manager EMPTY>
<!ATTLIST manager name IDREF #REQUIRED>
<!ELEMENT employee (name, firstname?, activities, details?)>
<!ATTLIST employee id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT activities (activity*)>
<!ELEMENT activity EMPTY>
<!ATTLIST activity department IDREF #REQUIRED>
<!ELEMENT details (age?, gender?)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
```

2. XML Schema for Company Organization

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="organization">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="departments">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="department"
maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="name" type="xs:string"/>
                    <xs:element name="manager" minOccurs="0">
                      <xs:complexType>
                        <xs:attribute name="name"
type="xs:IDREF" use="required"/>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name="num" type="xs:ID"
use="required"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
```

```
        </xs:element>
        <xs:element name="employees">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="employee"
maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="name" type="xs:string"/>
                    <xs:element name="firstname"
type="xs:string" minOccurs="0"/>
                    <xs:element name="activities">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="activity"
maxOccurs="unbounded">
                            <xs:complexType>
                             <xs:attribute name="department"
type="xs:IDREF" use="required"/>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="details" minOccurs="0">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="age"
type="xs:integer" minOccurs="0"/>
                          <xs:element name="gender"
type="xs:string" minOccurs="0"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name="id" type="xs:ID"
use="required"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

3. XPath Queries

a) Find last names of employees aged 50 or order

```
//employee[details/age >= 50]/name/text()
```

b) Last names of employees in Accounting (ID="Account")

```
//employee[activities/activity[@department="Account"]]/name/tex
t()
```

c) IDs of employees in 2+ departments

```
//employee[count(activities/activity) >= 2]/@id
```

d) Count employees in Management (ID="manag")

```
count(//employee[activities/activity[@department="manag"]])
```

4. XSLT to Flatten the Document

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="/organization">
    <xsl:apply-templates select="departments/department"/>
  </xsl:template>

  <xsl:template match="department">
    <service num="{@num}">
      <nom><xsl:value-of select="name"/></nom>
      <employes>
```

```
        <xsl:for-each
select="//employee[activities/activity/@department =
current()/@num]">
          <employe>
            <nom><xsl:value-of select="name"/></nom>
            <xsl:if test="firstname">
              <prenom><xsl:value-of
select="firstname"/></prenom>
            </xsl:if>
            <xsl:if test="details">
              <details>
                <xsl:if test="details/age">
                  <age><xsl:value-of
select="details/age"/></age>
                </xsl:if>
              </details>
            </xsl:if>
          </employe>
        </xsl:for-each>
      </employes>
    </service>
  </xsl:template>
</xsl:stylesheet>
```

5. XQuery to Generate Flattened Document

```
for $dept in
doc("company.xml")/organization/departments/department
return
  <service num="{$dept/@num}">
    <nom>{$dept/name/text()}</nom>
    <employes>
      {
        for $emp in
doc("company.xml")/organization/employees/employee
        where some $act in $emp/activities/activity satisfies
$act/@department = $dept/@num
        return
          <employe>
            <nom>{$emp/name/text()}</nom>
            {if ($emp/firstname) then
<prenom>{$emp/firstname/text()}</prenom> else ()}
            {if ($emp/details) then
              <details>
                {if ($emp/details/age) then
<age>{$emp/details/age/text()}</age> else ()}
              </details>
            else ()}
          </employe>
      }
    </employes>
  </service>
```