XSD - Tipos de dados

Considere os seguintes elementos XML

```
<docente>
   <nome>Anabela Simões</nome>
   <contacto>abs@isec.pt</contacto>
   <departamento>DEIS</departamento>
   <departamento>DEE</departamento>
</docente>
```

```
<aluno>
   <nome>Joana Lopes</nome>
   <contacto>a22222@alunos.isec.pt</contacto>
   <departamento>DEIS</departamento>
   <idade>23</idade>
</aluno>
```

a) Escreva um tipo de dados XSD chamado **pessoaTipo** adequado para os dois elementos

```
<xsd:complexType name="pessoaTipo">
   <xsd:sequence>
         <xsd: element name="nome" type="xsd:string" />
         <xsd: element name="contacto" type="xsd:string" />
         <xsd: element name="departamento" type="xsd:string" maxOccurs ="2" />
         <xsd: element name="idade" type="xsd:integer" minOccurs ="0"/>
   </xsd:sequence>
</xsd:complexType>
```

b) Usando os tipos de dados definidos em a), escreva o XSD para validar os elementos **<docente>** e **<aluno>.**

```
         <xsd: element name="docente" type="pessoaTipo" />
         <xsd: element name="aluno" type="pessoaTipo" />
```

XSD Analise o ficheiro XML

```
<vendas responsavel=”Sofia Melo”>
    <online update=”12-12-2020”>
            <artigo id="x01">
                    <desc>artigo 1</desc>
                    <valor>12</valor>
                    <stock>30</stock>
            </artigo>
            <artigo id="x02">
                    <desc>artigo 2</desc>
                    <valor>45</valor>
                    <stock alerta = “sim”>5</stock>
            </artigo>
            ...
    </online>
    <loja local="Coimbra">
            <artigo id="x04">
                    <desc>artigo 4</desc>
                    <valor>15</valor>
            </artigo>
            <artigo id="x05">
                    <desc>artigo 5</desc>
                    <valor>100</valor>
            </artigo>
    </loja>
    ...
</vendas>
```

1. Faça o XSD para os seguintes elementos, use o comando **ref** sempre que necessário assumindo que elementos/atributos já possam estar definidos

Elemento **desc**

```
<xsd: element name=”desc” type=”xsd:string” />
```

Elemento **stock**

```
<xsd:element name =”stock”>
        <xsd:complexType>
                <xsd:simpleContent>
                        <xsd:extension base=”xsd:integer”>
                                <xsd:attribute ref=”alerta” use=”optional”/>
                        </xsd:extension>
                </xsd:simpleContent>
        </xsd:complexType>
</xsd:element>
```

Atributo **local**

```
<xsd:attribute name=”local” type=”xsd:string”/>
```

Elemento **online**

```
<xsd:element name=”online”>
    <xsd:complexType>
            <xsd:sequence maxOccurs=”unbounded”>
            <xsd:element ref=”artigo”/>
      </xsd:sequence>
      <xsd:attribute ref=”update” use=”required” />
   </xsd:complexType>
</xsd:element>
```

Elemento **vendas**

```
<xsd:element name=”vendas”>
          <xsd:complexType>
                  <xsd:sequence >
                  <xsd:element ref=”online”/>
                  <xsd:element ref=”loja”/>
            </xsd:sequence>
            <xsd:attribute ref=”responsavel” use=”required” />
          </xsd:complexType>
</xsd:element>
```

XSLT

Analise os dois ficheiros XML

Ficheiro A

```
<livros>
  <livro isbn="123">
    <titulo>O Aleph</titulo>
    <autor>Paulo Coelho</autor>
    <preco>20</preco>
  </livro>
  <livro isbn="345">
    <titulo>Os Profetas</titulo>
    <autor>Alice Vieira</autor>
    <preco>15</preco>
  </livro>
  ...
</livros>
```

Ficheiro B

```
<catalogo>
   <livro titulo="O Aleph">
     <isbn>123</isbn>
     <escritor>Paulo Coelho</escritor>
     <preco>20</preco>
   </livro>
   <livro titulo="Os Profetas">
     <isbn>345</isbn>
     <escritor>Alice Vieira</escritor>
     <preco>15</preco>
   </livro>
...
</catalogo>
```

1a) Escreva o XSLT que permita fazer a transformação do **ficheiro A** no **ficheiro B**.

```
<xsl:stylesheet ....>
      <xsl:template match="livros">
            <catalogo>
                <xsl:apply-templates select="livro"/>
            </catalogo>
      </xsl:template>

      <xsl:template match="livro">
            <livro titulo="{titulo}">
                  <isbn><xsl:value-of select="@isbn"/></isbn>
                  <escritor><xsl:value-of select="autor"/></escritor>
                  <xsl:copy-of select="preco"/>
            </livro>
      </xsl:template>
</xsd:stylesheet>
```

1b) Qual a alteração que teria de fazer ao template de a) para obter o resultado da transformação com os livros **ordenados por ordem alfabética de autor**? Escreva apenas o código XSLT que teria de ser diferente.

```
      <xsl:template match="livros">
            <catalogo>
                <xsl:apply-templates select="livro">
                  <xsl:sort select="autor"/>
                </apply-templates>
            </catalogo>
      </xsl:template>
```

2a) Escreva o XQUERY que permita fazer a transformação do **ficheiro A** no **ficheiro B**.

```
<catalogo>
{

}
</catalogo
```

2b) Qual a alteração que teria de fazer em a) para obter o resultado da transformação com os livros **ordenados por ordem alfabética de autor**?

```
<xsl:template match=”livros”>
          <catalogo>
              <xsl:apply-templates select=”livro”>
                <xsl:sort select=”autor”/>
              </apply-templates>
          </catalogo>
</xsl:template>
```

XQuery

a) Suponha que possui o seguinte ficheiro XML e pretende obter a transformação mostrada à direita:

```
[exemplo.xml]
<registos>
  <pessoa id="p01" nome="Ana"/>
  <pessoa id="p02" nome="Laura"/>
  <pessoa id="p03" nome="Pedro"/>
  <pessoa id="p04" nome="Carlos"/>
  <consultas dia="2014-03-01">
    <marcação hora="10:00:00" pessoa="p01"/>
    <marcação hora="11:00:00" pessoa="p02"/>
  </consultas>
  <consultas dia="2014-04-01">
    <marcação hora="13:00:00" pessoa="p04"/>
    <marcação hora="14:00:00" pessoa="p03"/>
    <marcação hora="15:00:00" pessoa="p01"/>
    <marcação hora="16:00:00" pessoa="p02"/>
  </consultas>
</registos>
```

```
[Output Desejado]

<listagem>
   <consulta dia="2014-03-01" hora="10:00:00" nome="Ana"/>
   <consulta dia="2014-03-01" hora="11:00:00" nome="Laura"/>
   <consulta dia="2014-04-01" hora="13:00:00" nome="Carlos"/>
   <consulta dia="2014-04-01" hora="14:00:00" nome="Pedro"/>
   <consulta dia="2014-04-01" hora="15:00:00" nome="Ana"/>
   <consulta dia="2014-04-01" hora="16:00:00" nome="Laura"/>
</listagem>
```

Escreva uma query em XQuery que produza o resultado pretendido.

RESOLUÇÃO

```
<listagem>
      {
      for $x in doc("exemplo.xml")//marcação
      let $p := doc("exemplo.xml")//pessoa[@id=$x/@pessoa]/@nome
      return <consulta dia="{$x/../@dia}" hora="{$x/@hora}" nome="{$p}"/>
      }
</listagem>
```

XQuery

Analise os dois ficheiros XML e responda às questões abaixo:

banco1.xml

```
<?xml version="1.0"?>
<banco>
  <conta conta-num="A0000001" cliente="C0000001">
    <filial>Coimbra, Solum</filial>
    <saldo>17000</saldo>
  </conta>
  <conta conta-num="A0000002" cliente="C0000002">
    <filial>Coimbra, Solum</filial>
    <saldo>20000</saldo>
  </conta>
  <conta conta-num="A0000003" cliente="C0000003">
    <filial>Porto, Boavista</filial>
    <saldo>1000</saldo>
  </conta>
</banco>
```

banco2.xml

```
<?xml version="1.0"?>
<banco>
  <conta cliente="C0000001" saldo=”300”>
    <nome>Laura Matos</nome>
  </conta>
  <conta cliente="C0000002" saldo=”0”>
    <nome>Joana Melo</nome>
  </conta>
  <conta cliente="C0000003" saldo=”24000”>
    <nome>Pedro Lopes</nome>
  </conta>

</banco>
```

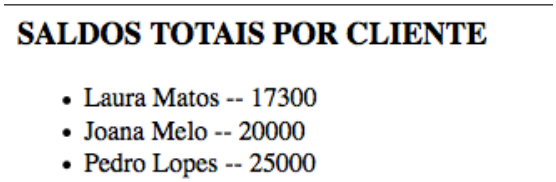
1. Escreva uma expressão FLWOR que leia o ficheiro **banco1.xml** e crie um novo ficheiro XML contendo apenas as contas com saldo acima dos 5000. O novo ficheiro deve estar ordenado por ordem crescente do saldo e deve ter a estrutura apresentada

```
<saldos>
  <conta>
       <num>A0000001</num>
       <saldo>17000</saldo>
  </conta>
  <conta>
       <num>A0000002</num>
       <saldo>20000</saldo>
  </conta>
</saldos>
```

RESOLUÇÃO:

```
<saldos>
{
      for $x in doc(“banco1.xml”)//conta
      where $x/saldo > 5000
      order by number($x/saldo)
      return
            <conta>
                  <num>{$x/@conta-num}</num>
                  <saldo>{$x/saldo/text()}</saldo>
            </conta>
}
</saldos>
```

2. Escreva uma query que mostre o saldo total para cada cliente, ordenado por saldo, tal como se mostra na figura seguinte (HTML):



SALDOS TOTAIS POR CLIENTE

- Laura Matos -- 17300
- Joana Melo -- 20000
- Pedro Lopes -- 25000

O calculo do saldo total deve ser feito usando uma função de nome **CalculaSaldo** que recebe o **id** de um cliente e devolve a soma dos saldos nas várias contas desse cliente.

RESOLUÇÃO:

```
declare namespace xsd = “....”;
declare function local:CalculaSaldo($id as xsd:string) as xsd:float
{
let $s1 := doc(“banco1.xml”)//conta[@cliente = $id]/saldo
let $s2 := doc(“banco2.xml”)//conta[@cliente = $id]/@saldo
return $s1 + $s2
};

<html><body>
<h1>Saldos totais por cliente</h1>
<ul>
{
for $x in doc(“banco2.xml”)//conta
let $s := local:CalculaSaldo($x/@cliente)
order by $s
return <li> {$x/nome} – {$s}</li>
}
</ul>
</body></html>
```

XQUERY e XSLT

Faça a transformação do ficheiro exames.xml no seguinte output XML.
Use XSLT
Os alunos estão ordenados por ordem alfabética.

```
<exames>
    <aluno>
        <nome>Ana Melo</nome>
        <idade>15</idade>
        <nivel>2</nivel>
        <avaliacoes>
            <exame data="2013-02-12" nota="25"/>
            <exame data="2014-06-22" nota="45"/>
            <exame data="2015-04-09" nota="55"/>
        </avaliacoes>
    </aluno>
    <aluno>
        <nome>Pedro Melo</nome>
        <idade>15</idade>
        <nivel>4</nivel>
        <avaliacoes>
            <exame data="2013-02-12" nota="80"/>
            <exame data="2014-06-22" nota="60"/>
            <exame data="2015-04-09" nota="70"/>
        </avaliacoes>
    </aluno>
...
</examess>
```

Integração de Dados – 2019/

```
<listagem>
    <aluno nivel="1">
        <nome>Ana Costa</nome>
        <media>35</media>
        <status>Reprovado</status>
    </aluno>
    <aluno nivel="2">
        <nome>Ana Melo</nome>
        <media>41.66</media>
        <status>Reprovado</status>
    </aluno>
    <aluno nivel="4">
        <nome>Carla Melo</nome>
        <media>65</media>
        <status>Aprovado</status>
    </aluno>
...
</listagem>
```

Resolução XSLT

```
<xsl:stylesheet ....>
      <xsl:template match=”exames”>
            <listagem>
                  <xsl:apply-templates select=”aluno”>
                        <xsl:sort select=”nome”/>
                  </xsl:apply-templates>
            </listagem>
      </xsl:template>

      <xsl:template match=”aluno”>
            <aluno nível=”{nivel}”>
                  <nome><xsl:value-of select=”nome”/></nome>
                  <media><xsl:value-of select=”avg(avaliacoes/exame/@nota)”/></media>
                  <xsl:if test=” avg(avaliacoes/exame/@nota) >= 50”>
                        <status>Aprovado</status>
                  </xsl:if>
                  <xsl:if test=”avg(avaliacoes/exame/@nota) < 50”>
                        <status>Reprovado</status>
                  </xsl:if>

            </aluno>

      </xsl:template>

</xsl:stylesheet>
```

Resolução XQUERY

```
<listagem>
{
for $a doc("exames.xml")//aluno
order by $a/nome
return if (avg($a/avaliações/exame/@nota) >= 50) then
            <aluno nível="{$a/nivel}">
            <nome>{$a/nome/text()}</nome>
            <media>{ avg($a/avaliações/exame/@nota) }</media>
            <status>Aprovado</status>
            </aluno>

      else(
            <aluno nível="{$a/nivel}">
            <nome>{$a/nome/text()}</nome>
            <media>{ avg($a/avaliações/exame/@nota) }</media>
            <status>Reprovado</status>
            </aluno>
         )

}
</listagem>
```

XSLT

Faça a transformação do ficheiro exames.xml no seguinte output HTML. Em XSLT e em XQuery
Os alunos estão ordenados por ordem alfabética. Para cada aluno deve ser calculada a média das notas. Se a média for superior ou igual a 50% deve aparecer a string "Aluno Aprovado", caso contrário aparece a string "Aluno reprovado", seguido da média e da lista de notas ordenadas por ordem crescente.

```
<exames>
    <aluno>
        <nome>Ana Melo</nome>
        <idade>15</idade>
        <nivel>2</nivel>
        <avaliacoes>
            <exame data="2013-02-12" nota="25"/>
            <exame data="2014-06-22" nota="45"/>
            <exame data="2015-04-09" nota="55"/>
        </avaliacoes>
    </aluno>
    <aluno>
        <nome>Pedro Melo</nome>
        <idade>15</idade>
        <nivel>4</nivel>
        <avaliacoes>
            <exame data="2013-02-12" nota="80"/>
            <exame data="2014-06-22" nota="60"/>
            <exame data="2015-04-09" nota="70"/>
        </avaliacoes>
    </aluno>
...
</examess>
```

Resumo dos Alunos

- Aluno Ana Costa:
  - Aluno Reprovado - Média: 35
  - Nota: 35
- Aluno Ana Melo:
  - Aluno Reprovado - Média: 42
  - Nota: 25
  - Nota: 45
  - Nota: 55
- Aluno Carla Melo:
  - Aluno Aprovado - Média: 72
  - Nota: 45
  - Nota: 85
- Aluno Joana Martins:
  - Aluno Reprovado - Média: 35

Integração de Dados – 2019/20

RESOLUÇÃO

```
<html><body>
<h1>Resumo dos alunos</h1>
<ul>
{
for $a doc("exames.xml")//aluno
let $media := avg($a/avaliações/exame/@nota)
order by $a/nome
return if($media >= 50) then
        ( <li>Aluno {$a/nome/text()}: </li>,
        <ul>
              <li> Aluno Aprovado – Média: {round($media)} </li>
              {
              for $n in $a//exame
              order by $n/@nota
              return <li>Nota: {data($n/@nota)}</li>
              }
       </ul>)
       else
       ( <li>Aluno {$a/nome/text()}: </li>,
        <ul>
              <li> Aluno Reprovado – Média: {round($media)} </li>
              {
              for $n in $a//exame
              order by $n/@nota
              return <li>Nota: {data($n/@nota)}</li>
              }
       </ul>)

}
</ul>
</body></html>
```

XSLT e XQUERY

Criar uma Tabela HTML com alunos do nível 4, mostrar a nota mais alta obtida e a data em que foi registada. Ordenar alfabeticamente pelo nome

XSLT e XQUERY

```
<exames>
      <aluno>
            <nome>Ana Melo</nome>
            <idade>15</idade>
            <nivel>2</nivel>
            <avaliacoes>
                  <exame data="2013-02-12" nota="25"/>
                  <exame data="2014-06-22" nota="45"/>
                  <exame data="2015-04-09" nota="55"/>
            </avaliacoes>
      </aluno>
      <aluno>
            <nome>Pedro Melo</nome>
            <idade>15</idade>
            <nivel>4</nivel>
            <avaliacoes>
                  <exame data="2013-02-12" nota="80"/>
                  <exame data="2014-06-22" nota="60"/>
                  <exame data="2015-04-09" nota="70"/>
            </avaliacoes>
      </aluno>
      <aluno>
            <nome>Ana Costa</nome>
            <idade>12</idade>
            <nivel>1</nivel>
            <avaliacoes>
                  <exame data="2014-08-19" nota="35"/>
            </avaliacoes>
      </aluno>
...
```

Exames de alunos de nivel 4

Nome	Idade	Nº de avaliações	Nota max	Data
Carla Melo	18	2	85	2013-02-12
Pedro Melo	15	3	80	2013-02-12
Rui Melo	16	4	91	2014-06-30

RESOLUÇÃO XSLT

```
<xsl:stylesheet ...>
      <xsl:template match=”exames”>
            <html><body>
            <h1>Exames de alunos de nível 4 </h1>
            <table border=”1”>
            <tr><th>Nome</th><th>Idade</th><th>N. avalia</th><th>Nota max</th><th>Data</th></tr>
            <xsl:for-each select=”aluno”>
                  <xsl:sort select=”nome”/>
             <xsl:if test=’nivel = 4’>
                <tr>
                    <td><xsl:value-of select=”nome”/></td>
                    <td><xsl:value-of select=”idade”/></td>
                    <td><xsl:value-of select=”count(avaliacoes/exame)”/></td>
                    <td><xsl:value-of select=”max(avaliacoes/exame/@nota)”/></td>
                    <td><xsl:value-of select=”avaliacoes/exame[@nota=max(../exame/@nota)]/@data)”/></td>
                </tr>
             </xsl:if>
            </xsl:for-each>
          </table> </body> </html>
      </xsl:template>
</xsl:stylesheet>
```

RESOLUÇÃO XQUERY

```
<html><body>
<h1>Exames de alunos de nível 4 </h1>
<table border=”1”>
<tr><th>Nome</th><th>Idade</th><th>N. avalia</th><th>Nota max</th><th>Data</th></tr>
{
for $a in doc(“exames.xml”)//aluno
let $c := count($a/avaliações/exame), $m := max($a/avaliações/exame/@nota)
where $a/nível = 4
order by $a/nome
return <tr>
            <td>{$a/nome/text()}</td>
            <td>{$a/idade/text()}</td>
            <td>{$c}</td>
            <td>{$m}</td>
            <td>{data($a/avaliações/exame[@nota = $m]/@data)}</td>

      </tr>
}
</table> </body> </html>
```