

JEGYZŐKÖNYV

Adatkezelés XML környezetben

Virágbolt vevőkör nyilvántartó rendszer

Készítette: Tóth Gergő

Neptunkód: FOQH34

1) Feladat leírása:

Az adatbázis rendszerem alapja egy virágbolt láncolat nyilvántartási rendszere melyben egyed szerepet tölt be a

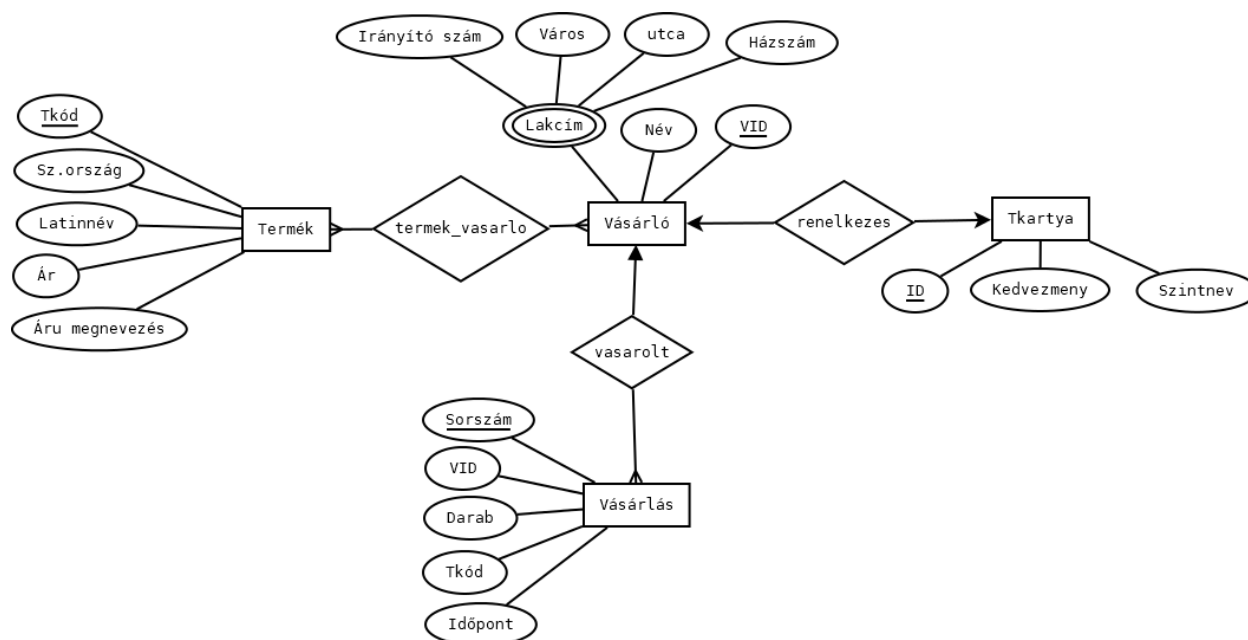
-törzsvásárlói kártya melynek tulajdonságai az azonosítója, a törzsvásárlói szint megnevezése és a hozzátartozó kedvezmény mértéke.

-bolt árukészlete tulajdonságai az áru megnevezése, az áru latin neve, ára, termék kódja valamint a származási ország neve.

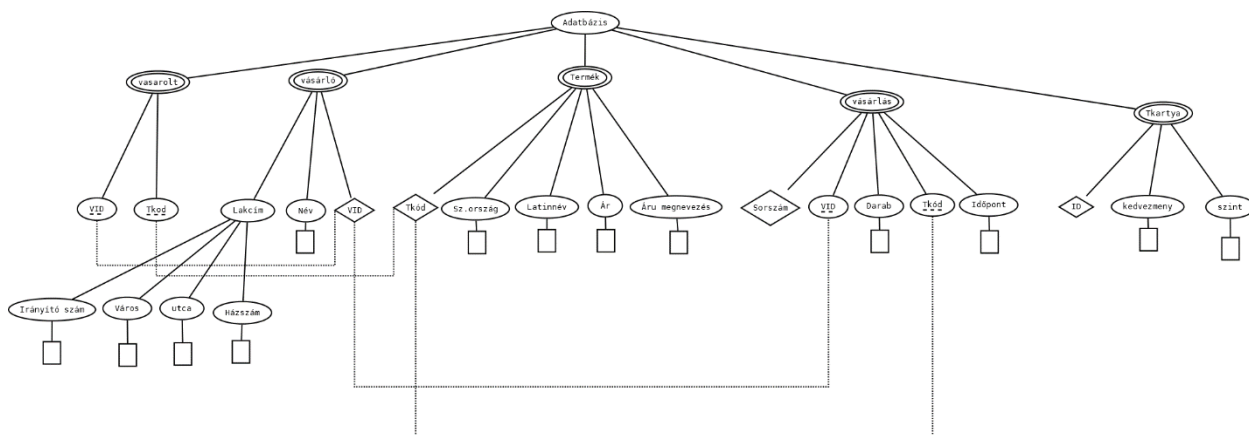
-vásárlók amik a nevéből, lakcíméből, törzsvásárlói kártyájukból, illetve egyedi vásárló azonosítójából áll.

illetve a vásárlás adatai ami egy egyéni sorszámmal rendelkezik valamint egy vásárló azonosítóval, egy dátummal, a termék kódjával és a vásárolt termék darabszámával.

1a) Az adatbázis ER modell:



1b) Az adatbázis konvertálása XDM modellre:



1c) Az XDM modell alapján XML dokumentum készítése:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <adatbázis xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="XMLSchemaFOQH34.xsd">
3
4   <vasarlo id="V12">
5     <lakcim>
6       <iranyitoSzam>3535</iranyitoSzam>
7       <varos>Miskolc</varos>
8       <utca>Hegyalja utca</utca>
9       <hazszam>15</hazszam>
10    </lakcim>
11    <nev>Tóth Gergő</nev>
12  </vasarlo>
13
14  <vasarlo id="V13">
15    <lakcim>
16      <iranyitoSzam>3516</iranyitoSzam>
17      <varos>Miskolc</varos>
18      <utca>Csermak Antal utca</utca>
19      <hazszam>41</hazszam>
20    </lakcim>
21    <nev>Toth Levente</nev>
22  </vasarlo>
23
24  <termek id="T01">
25    <szarmazasiOrszag>Hollandia</szarmazasiOrszag>
26    <latinNev>Rosa</latinNev>
27    <ar>850</ar>
28    <aruMegnevezes>Vörös rózsák</aruMegnevezes>
29  </termek>
30
31  <termek id="T02">
32    <szarmazasiOrszag>Magyar</szarmazasiOrszag>
33    <latinNev>-</latinNev>
34    <ar>1200</ar>
35    <aruMegnevezes>illat gyertya</aruMegnevezes>
36  </termek>
37
```

```

37
38Ⓜ <vasarlas id="S01">
39   <vid>V12</vid>
40   <darab>10</darab>
41   <tkod>T01</tkod>
42   <idopont>2022-12-05</idopont>
43 </vasarlas>
44
45Ⓜ <tkartya id="TK01">
46   <szint>platina</szint>
47   <kedvezmeny>10</kedvezmeny>
48 </tkartya>
49 </adatbazis>

```

1d) Az XML dokumentum alapján XMLSchema készítése (saját típusok):

```

1 <?xml version="1.0" encoding="UTF-8"?>
2Ⓜ <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://www.example.org/XMLSchemaFOQH34" elementFormDefault="qualified">
3Ⓜ   <xs:element name="adatbazis">
4Ⓜ     <xs:complexType>
5Ⓜ       <xs:sequence>
6         <xs:element name="vasarlo" type="vasarloTipus" minOccurs="0" maxOccurs="unbounded"/>
7         <xs:element name="termek" type="termekTipus" maxOccurs="unbounded"/>
8         <xs:element name="vasarlas" type="vasarlasTipus" maxOccurs="unbounded"/>
9         <xs:element name="tkartya" type="tkartyaTipus" maxOccurs="unbounded"/>
10      </xs:sequence>
11    </xs:complexType>
12
13Ⓜ   <xs:key name="vasarloKulcs">
14     <xs:selector xpath="vasarlo"/>
15     <xs:field xpath="@id" />
16   </xs:key>
17Ⓜ   <xs:key name="tkartyaKulcs">
18     <xs:selector xpath="tkartya"/>
19     <xs:field xpath="@id" />
20   </xs:key>
21Ⓜ   <xs:key name="termekKulcs">
22     <xs:selector xpath="termek"/>
23     <xs:field xpath="@id" />
24   </xs:key>
25Ⓜ   <xs:key name="vasarlasKulcs">
26     <xs:selector xpath="vasarlas"/>
27     <xs:field xpath="@id" />
28   </xs:key>
29
30Ⓜ   <xs:keyref name="vasarlas-termek" refer="termekKulcs">
31     <xs:selector xpath="vasarlas/tkod"/>
32     <xs:field xpath="."/>
33   </xs:keyref>
34Ⓜ   <xs:keyref name="vasarlas-vasarlo" refer="vasarloKulcs">
35     <xs:selector xpath="vasarlas/vid"/>
36     <xs:field xpath="."/>
37   </xs:keyref>
38Ⓜ   <xs:keyref name="vasarlo-tkartya" refer="tkartyaKulcs">
39     <xs:selector xpath="vasarlo/tkartyaid"/>
40     <xs:field xpath="."/>
41   </xs:keyref>
42 </xs:element>
43Ⓜ   <xs:complexType name="vasarloTipus">
44     <xs:sequence>
45       <xs:element name="lakcim" type="lakcimTipus" />
46       <xs:element name="nev" type="nev"/>
47     </xs:sequence>
48     <xs:attribute name="id" type="xs:ID" use="required" />
49   </xs:complexType>
50
51Ⓜ   <xs:complexType name="lakcimTipus">
52     <xs:sequence>
53       <xs:element name="iranyitoSzam" type="iranyitoSzam"/>
54       <xs:element name="varos" type="varos"/>
55       <xs:element name="utca" type="utca"/>
56       <xs:element name="hazszam" type="hazszam"/>
57     </xs:sequence>
58   </xs:complexType>
59
60Ⓜ   <xs:simpleType name="iranyitoSzam">
61     <xs:restriction base="xs:positiveInteger"/>
62   </xs:simpleType>
63
64Ⓜ   <xs:simpleType name="varos">
65     <xs:restriction base="xs:string"/>
66   </xs:simpleType>
67
68Ⓜ   <xs:simpleType name="utca">
69     <xs:restriction base="xs:string"/>
70   </xs:simpleType>
71
72Ⓜ   <xs:simpleType name="hazszam">
73     <xs:restriction base="xs:positiveInteger"/>
74   </xs:simpleType>

```

```

74     </xs:simpleType>
75
76     <xs:simpleType name="nev">
77       <xs:restriction base="xs:string"/>
78     </xs:simpleType>
79
80     <xs:complexType name="termekTipus">
81       <xs:sequence>
82         <xs:element name="szarmazasiOrszag" type="xs:string"/>
83         <xs:element name="latinNev" type="xs:string"/>
84         <xs:element name="ar" type="xs:positiveInteger"/>
85         <xs:element name="aruMegnevezes" type="xs:string"/>
86       </xs:sequence>
87       <xs:attribute name="id" type="xs:ID" use="required" />
88     </xs:complexType>
89     <xs:complexType name="tkartyaTipus">
90       <xs:sequence>
91         <xs:element name="szint" type="xs:string"/>
92         <xs:element name="kedvezmeny" type="xs:positiveInteger"/>
93       </xs:sequence>
94       <xs:attribute name="id" type="xs:ID" use="required" />
95     </xs:complexType>
96     <xs:complexType name="vasarlasTipus">
97       <xs:sequence>
98         <xs:element name="vid" type="xs:string"/>
99         <xs:element name="darab" type="xs:string"/>
100        <xs:element name="tkod" type="xs:string"/>
101        <xs:element name="idopont" type="xs:date"/>
102      </xs:sequence>
103      <xs:attribute name="id" type="xs:ID" use="required" />
104    </xs:complexType>
105  </xs:schema>

```

2. feladat:

2a) adatolvasás

```

1  package hu.domparse.FOQH34;
2
3  import java.io.File;
4  import java.io.IOException;
5
6  import javax.xml.parsers.DocumentBuilder;
7  import javax.xml.parsers.DocumentBuilderFactory;
8  import javax.xml.parsers.ParserConfigurationException;
9
10 import org.w3c.dom.Document;
11 import org.w3c.dom.Element;
12 import org.w3c.dom.Node;
13 import org.w3c.dom.NodeList;
14 import org.xml.sax.SAXException;
15
16 public class DomReadFOQH34 {
17     public static void main(String argv[]) throws SAXException, IOException, ParserConfigurationException {
18         File xmlFile = new File("XMLfoqh34.xml");
19
20         //DOMFA felépítése
21         DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
22         DocumentBuilder dBuilder = factory.newDocumentBuilder();
23         Document doc = dBuilder.parse(xmlFile);
24         doc.getDocumentElement().normalize();
25
26         //Adattagokon végig megyünk
27         System.out.println("Root element: "+doc.getDocumentElement().getNodeName());
28         String[] tagNames = {"vasarlo", "termek", "vasarlas", "tkartya"};
29         for(String tagName : tagNames) {
30             NodeList nList = doc.getElementsByTagName(tagName);
31
32             for(int i=0; i< nList.getLength();i++) {
33                 Node nNode = nList.item(i);
34                 System.out.println("\nCurrent Element: " + nNode.getNodeName());
35
36                 if (nNode.getNodeType() == Node.ELEMENT_NODE) {
37                     Element elem = (Element) nNode;

```

```

37         Element elem =(Element) nNode;
38
39         //Azonosító kiírása
40         String id = elem.getAttribute("id");
41         System.out.println("    ID: " +id);
42         //Tulajdonságok kiírása
43
44         String nodeContent="";
45         NodeList childNodes =elem.getChildNodes();
46
47         for(int j =0; j< childNodes.getLength();j++) {
48             if(childNodes.item(j).getTextContent().trim() != "") {
49                 nodeContent =normalizeText(childNodes.item(j).getTextContent().trim());
50                 System.out.println("        "+childNodes.item(j).getNodeName()+": "+nodeContent);
51             }
52         }
53     }
54 }
55 System.out.println();
56 }
57
58
59
60 }
61 }
62 }
63 private static String normalizeText(String text) {
64     text=text.replaceAll("\\n", " ");
65     text=text.replaceAll("\\s+", " ");
66     return text;
67 }
68 }
69

```

2b) Adatmódosítás

V12 ID-vel rendelkező vásárló nevének módosítása Kertész László-ra

T02 ID-vel rendelkező termék nevének módosítása Pink Rose-ra

T02 ID-vel rendelkező termék származási országának módosítása Kubára

```

1 package hu.domparsing.FOQH34;
2
3 import java.io.File;
4 import java.io.IOException;
5 import java.io.UnsupportedEncodingException;
6
7 import javax.xml.parsers.DocumentBuilder;
8 import javax.xml.parsers.DocumentBuilderFactory;
9 import javax.xml.parsers.ParserConfigurationException;
10 import javax.xml.transform.OutputKeys;
11 import javax.xml.transform.Transformer;
12 import javax.xml.transform.TransformerException;
13 import javax.xml.transform.TransformerFactory;
14 import javax.xml.transform.dom.DOMSource;
15 import javax.xml.transform.stream.StreamResult;
16
17 import org.w3c.dom.Document;
18 import org.w3c.dom.Node;
19 import org.w3c.dom.NodeList;
20 import org.xml.sax.SAXException;
21
22 public class DomModifyFOQH34 {
23     public static void main(String[] args) {
24         try {
25             File xmlFile = new File("XMLfoqh34.xml");
26             DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
27             DocumentBuilder dBuilder = factory.newDocumentBuilder();
28             Document doc = dBuilder.parse(xmlFile);
29             doc.getDocumentElement().normalize();
30
31             //V12 ID-vel rendelkező vásárló nevének módosítása Kertész László-ra
32             NodeList nodes = doc.getElementsByTagName("vasarlo");
33             for(int i =0; i<nodes.getLength();i++) {
34                 Node node = nodes.item(i);
35                 if(node.getNodeType() == Node.ELEMENT_NODE) {
36                     if(node.getAttribute("id").getTextContent().equals("V12")) {
37                         NodeList childNodes = node.getChildNodes();
38                         for(int j=0; j < childNodes.getLength();j++) {
39                             Node childNode = childNodes.item(j);
40                             if(childNode.getNodeName().equals("nev")) {
41                                 childNode.setTextContent("Kertesz Laszlo");
42                             }
43                         }
44                     }
45                 }
46             }
47             //T02 ID-vel rendelkező termek nevének módosítása Pink Rose-ra
48             nodes = doc.getElementsByTagName("termek");
49             for(int i =0; i<nodes.getLength();i++) {
50                 Node node = nodes.item(i);
51                 if(node.getNodeType() == Node.ELEMENT_NODE) {
52                     if(node.getAttribute("id").getTextContent().equals("T02")) {
53                         NodeList childNodes = node.getChildNodes();
54                         for(int j=0; j < childNodes.getLength();j++) {
55                             Node childNode = childNodes.item(j);
56                             if(childNode.getNodeName().equals("aruMegnevezes")) {
57                                 childNode.setTextContent("Pink Rose");
58                             }
59                         }
60                     }
61                 }
62             }
63             //T02 ID-vel rendelkező termek származási országának módosítása Kubára
64             nodes = doc.getElementsByTagName("termek");
65             for(int i =0; i<nodes.getLength();i++) {
66                 Node node = nodes.item(i);
67                 if(node.getNodeType() == Node.ELEMENT_NODE) {
68                     if(node.getAttribute("id").getTextContent().equals("T02")) {
69                         NodeList childNodes = node.getChildNodes();
70                         for(int j=0; j < childNodes.getLength();j++) {
71                             Node childNode = childNodes.item(j);
72                             if(childNode.getNodeName().equals("szarmazasiOrszag")) {
73                                 childNode.setTextContent("Kuba");
74                             }
75                         }
76                     }
77                 }
78             }
79
80             File file =new File("XMLFOQH34.xml");
81             writeXml(doc,file);
82
83         }
84         catch(ParserConfigurationException | IOException | SAXException | TransformerException ex){
85             System.out.println("Some error happened:\n"+ex.getMessage());
86             ex.printStackTrace();
87         }
88     }
89     private static void writeXml(Document doc, File output) throws TransformerException,UnsupportedEncodingException{
90         TransformerFactory transformerFactory= TransformerFactory.newInstance();
91         Transformer transf =transformerFactory.newTransformer();
92         transf.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
93         transf.setOutputProperty(OutputKeys.INDENT, "yes");
94         transf.setOutputProperty("http://xml.apache.org/xslt indent-amount", "2");
95         DOMSource source = new DOMSource(doc);
96
97         StreamResult console = new StreamResult(System.out);
98         StreamResult file = new StreamResult(output);
99
100         transf.transform(source, console);
101         transf.transform(source, file);
102     }
103 }
104

```

3c) Lekérdezés:

```

1 package hu.domparse.FOQH34;
2
3 import java.io.File;
4 import java.io.IOException;
5
6 import javax.xml.parsers.DocumentBuilder;
7 import javax.xml.parsers.DocumentBuilderFactory;
8 import javax.xml.parsers.ParserConfigurationException;
9
10 import org.w3c.dom.Document;
11 import org.w3c.dom.Element;
12 import org.w3c.dom.Node;
13 import org.w3c.dom.NodeList;
14 import org.xml.sax.SAXException;
15
16 public class DomQueryFOQH34 {
17
18     public static void main(String[] args) {
19         try {
20             File xmlFile = new File("XMLfoqh34.xml");
21             DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
22             DocumentBuilder dBuilder = factory.newDocumentBuilder();
23             Document doc = dBuilder.parse(xmlFile);
24             doc.getDocumentElement().normalize();
25
26             String feladat="Ezer forintnál drágább termékek:";
27             System.out.println(feladat + "\n\n");
28             Lekerdezes1(doc);
29
30             String feladat2="V13 kódú vásárló adatai: ";
31             System.out.println(feladat2 + "\n\n");
32             Lekerdezes2(doc);
33
34         }
35         catch(ParserConfigurationException | IOException | SAXException ex){
36             System.out.println("Some error happened:\n"+ex.getMessage());
37
38             System.out.println("Some error happened:\n"+ex.getMessage());
39             ex.printStackTrace();
40         }
41     }
42
43     private static String normalizeText(String text) {
44         text=text.replaceAll("\n", " ");
45         text=text.replaceAll("\s+", " ");
46         return text;
47     }
48
49     private static void Lekerdezes1(Document doc) {
50         NodeList termek = doc.getElementsByTagName("termek");
51         for(int i = 0; i < termek.getLength(); i++) {
52             Element termek = (Element)termek.item(i);
53             NodeList childNodes = termek.getChildNodes();
54             for(int j =0;j<childNodes.getLength();j++) {
55                 Node childNode = childNodes.item(j);
56                 if(childNode.getNodeName().equals("ar")) {
57                     if(Integer.parseInt(childNode.getTextContent())>1000) {
58                         printElement(termek);
59                     }
60                 }
61             }
62         }
63     }
64
65     private static void Lekerdezes2(Document doc) {
66         NodeList vevok = doc.getElementsByTagName("vasarlo");
67         for (int i = 0; i < vevok.getLength(); i++) {
68             Element vevo = (Element)vevok.item(i);
69             if (vevo.getAttributeNode("id").getValue().equals("V13"))
70                 printElement(vevo);
71         }
72     }
73
74     private static void printElement(Element elem) {
75         String id = elem.getAttribute("id");
76         System.out.println("    ID: " + id);
77
78         String nodeContent="";
79         NodeList childNodes = elem.getChildNodes();
80         for(int j =0;j<childNodes.getLength(); j++) {
81             if(childNodes.item(j).getTextContent().trim()!="") {
82                 nodeContent = normalizeText(childNodes.item(j).getTextContent().trim());
83                 System.out.println(childNodes.item(j).getNodeName()+" "+nodeContent);
84             }
85         }
86     }
87
88 }
89
90 }
91

```