

# JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Készítette: Tóth Gergő

Neptunkód: FOQH34

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

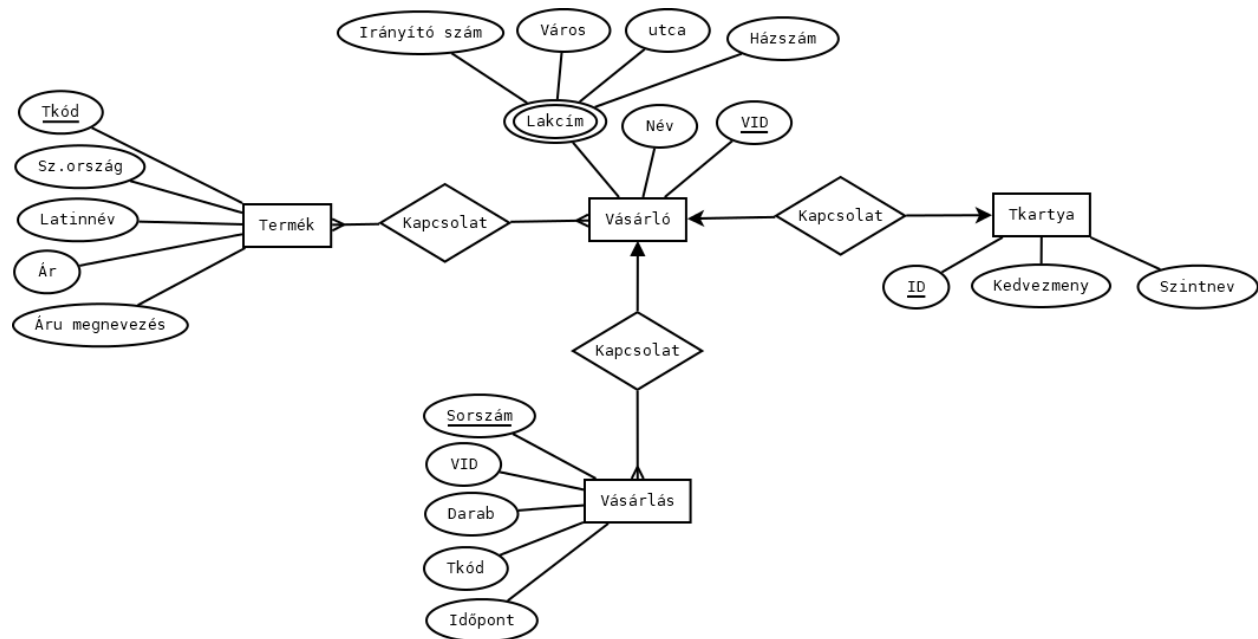
-törzsvásárlói kártya melynek tulajdonságai az azonosítója, a törzsvásárlói szint megnevezése és a hozzátartozó kedvezmény mértéke.

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

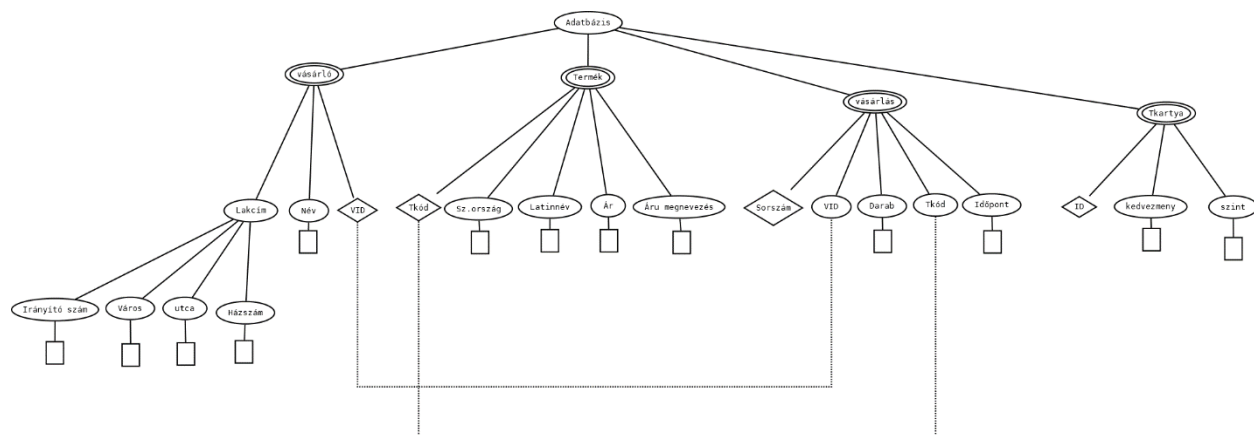
-vásárlók amik a nevéből, lakcíméből, törzsvásárlói kártyájukból, illetve egyedi vásárló azonosítójából áll.

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>12</vid>
40  <darab>10</darab>
41  <tKod>01</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>
50

```

## 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="vasarLoTípus" minOccurs="0" maxOccurs="unbounded"/>
7          <xs:element name="termek" type="termekTípus" maxOccurs="unbounded"/>
8          <xs:element name="vasarLas" type="vasarLasTípus" maxOccurs="unbounded"/>
9          <xs:element name="tkartya" type="tkartyaTípus" 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
18      <xs:key name="tkartyaKulcs">
19        <xs:selector xpath="tkartya"/>
20        <xs:field xpath="@id" />
21      </xs:key>
22
23      <xs:key name="termekKulcs">
24        <xs:selector xpath="termek"/>
25        <xs:field xpath="@id" />
26      </xs:key>
27
28      <xs:key name="vasarLasKulcs">
29        <xs:selector xpath="vasarLas"/>
30        <xs:field xpath="@id" />
31      </xs:key>
32    </xs:element>
33    <xs:complexType name="vasarLoTípus">
34      <xs:sequence>
35        <xs:element name="lakcim" type="lakcimTípus" />
36        <xs:element name="nev" type="nev"/>
37      </xs:sequence>
38      <xs:attribute name="id" type="xs:ID" use="required" />
39    </xs:complexType>
40
41    <xs:complexType name="lakcimTípus">
42      <xs:sequence>
43        <xs:element name="iranyitoSzam" type="iranyitoSzam"/>
44        <xs:element name="varos" type="varos"/>
45        <xs:element name="utca" type="utca"/>
46        <xs:element name="hazszam" type="hazszam"/>
47      </xs:sequence>
48    </xs:complexType>
49
50    <xs:simpleType name="iranyitoSzam">
51      <xs:restriction base="xs:positiveInteger"/>
52    </xs:simpleType>
53
54    <xs:simpleType name="varos">
55      <xs:restriction base="xs:string"/>
56    </xs:simpleType>
57
58    <xs:simpleType name="utca">
59      <xs:restriction base="xs:string"/>
60    </xs:simpleType>
61
62    <xs:simpleType name="hazszam">
63      <xs:restriction base="xs:positiveInteger"/>
64    </xs:simpleType>
65
66    <xs:simpleType name="nev">
67      <xs:restriction base="xs:string"/>
68    </xs:simpleType>
69
70    <xs:complexType name="termekTípus">
71      <xs:sequence>
72        <xs:element name="szarmazasiOrszag" type="xs:string"/>
73        <xs:element name="latinNev" type="xs:string"/>
74        <xs:element name="ar" type="xs:positiveInteger"/>
75
76        <xs:element name="ar" type="xs:positiveInteger"/>
77        <xs:element name="arulegnevezes" type="xs:string"/>
78      </xs:sequence>
79      <xs:attribute name="id" type="xs:ID" use="required" />
80    </xs:complexType>
81
82    <xs:complexType name="tkartyaTípus">
83      <xs:sequence>
84        <xs:element name="szint" type="xs:string"/>
85        <xs:element name="kedvezmeny" type="xs:positiveInteger"/>
86      </xs:sequence>
87      <xs:attribute name="id" type="xs:ID" use="required" />
88    </xs:complexType>
89
90    <xs:complexType name="vasarLasTípus">
91      <xs:sequence>
92        <xs:element name="vid" type="xs:positiveInteger"/>
93        <xs:element name="darab" type="xs:string"/>
94        <xs:element name="tKod" type="xs:positiveInteger"/>
95        <xs:element name="idopont" type="xs:date"/>
96      </xs:sequence>
97      <xs:attribute name="id" type="xs:ID" use="required" />
98    </xs:complexType>
99  </xs:schema>

```

## 2. feladat:

```
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 { public static void main(String argv[]) throws SAXException, IOException, ParserConfigurationException {
17     File xmlFile = new File("XMLfoqh34.xml");
18
19     //DOMFA felépítése
20     DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
21     DocumentBuilder dBuilder = factory.newDocumentBuilder();
22     Document doc = dBuilder.parse(xmlFile);
23     doc.getDocumentElement().normalize();
24
25     //Adattagokon végig megyünk
26     System.out.println("Root element: "+doc.getDocumentElement().getNodeName());
27     String[] tagNames= {"vasarlo", "termek", "vasarlas", "tkartya"};
28     for(String tagName :tagNames) {
29         NodeList nList = doc.getElementsByTagName(tagName);
30
31         for(int i=0; i< nList.getLength();i++) {
32             Node nNode = nList.item(i);
33             System.out.println("\nCurrent Element: " + nNode.getNodeName());
34
35             if (nNode.getNodeType() == Node.ELEMENT_NODE) {
36                 Element elem =(Element) nNode;
37
38                 Element elem =(Element) nNode;
39
40                 //Azonosító kiírása
41                 String id = elem.getAttribute("id");
42                 System.out.println(" ID: " +id);
43                 //Tulajdonságok kiírása
44
45                 String nodeContent="";
46                 NodeList childNodes =elem.getChildNodes();
47
48                 for(int j =0; j< childNodes.getLength();j++) {
49                     if(childNodes.item(j).getTextContent().trim() != "") {
50                         nodeContent =normalizeText(childNodes.item(j).getTextContent().trim());
51                         System.out.println(" "+childNodes.item(j).getNodeName()+": "+nodeContent);
52                     }
53                 }
54                 System.out.println();
55             }
56         }
57     }
58 }
59
60 private static String normalizeText(String text) {
61     text=text.replaceAll("\n", " ");
62     text=text.replaceAll("\s+", " ");
63     return text;
64 }
65
66 }
```

## 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.dompars.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.getAttributes().getNamedItem("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
41                             if(childNode.getNodeName().equals("nev")) {
42                                 childNode.setTextContent("Kertesz Laszlo");
43                             }
44                         }
45                     }
46                 }
47             }
48             //T02 ID-vel rendelkező termék nevének módosítása Pink Rose-ra
49             nodes = doc.getElementsByTagName("termek");
50             for(int i =0; i<nodes.getLength(); i++) {
51                 Node node = nodes.item(i);
52                 if(node.getNodeType() == Node.ELEMENT_NODE) {
53                     if(node.getAttributes().getNamedItem("id").getTextContent().equals("T02")) {
54                         NodeList childNodes = node.getChildNodes();
55                         for(int j=0; j < childNodes.getLength(); j++) {
56                             Node childNode = childNodes.item(j);
57                             if(childNode.getNodeName().equals("aruMegnevezes")) {
58                                 childNode.setTextContent("Pink Rose");
59                             }
60                         }
61                     }
62                 }
63             }
64             //T02 ID-vel rendelkező termék származási országának módosítása Kubára
65             nodes = doc.getElementsByTagName("termek");
66             for(int i =0; i<nodes.getLength(); i++) {
67                 Node node = nodes.item(i);
68                 if(node.getNodeType() == Node.ELEMENT_NODE) {
69                     if(node.getAttributes().getNamedItem("id").getTextContent().equals("T02")) {
70                         NodeList childNodes = node.getChildNodes();
71                         for(int j=0; j < childNodes.getLength(); j++) {
72                             Node childNode = childNodes.item(j);
73                             if(childNode.getNodeName().equals("szarmazasiOrszag")) {
74                                 childNode.setTextContent("Kuba");
75                             }
76                         }
77                     }
78                 }
79             }
80         }
81     }
82 }
```

```

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
90 private static void writeXml(Document doc, File output) throws TransformerException,UnsupportedEncodingException{
91     TransformerFactory transformerFactory= TransformerFactory.newInstance();
92     Transformer transf =transformerFactory.newTransformer();
93     transf.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
94     transf.setOutputProperty(OutputKeys.INDENT, "yes");
95     transf.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "2");
96     DOMSource source = new DOMSource(doc);
97
98     StreamResult console = new StreamResult(System.out);
99     StreamResult file = new StreamResult(output);
100
101     transf.transform(source, console);
102     transf.transform(source, file);
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             ex.printStackTrace();
38         }
39     }
40
41     private static String normalizeText(String text) {
42         text=text.replaceAll("\n", " ");
43         text=text.replaceAll("\s+", " ");
44         return text;
45     }
46
47     private static void Lekerdezes1(Document doc) {
48         NodeList termekek = doc.getElementsByTagName("termek");
49         for(int i = 0; i < termekek.getLength(); i++) {
50             Element termek =(Element)termekek.item(i);
51             NodeList childNodes = termek.getChildNodes();
52             for(int j =0;j<childNodes.getLength();j++) {
53                 Node childNode = childNodes.item(j);
54                 if(childNode.getNodeName().equals("ar")) {
55                     if(Integer.parseInt(childNode.getTextContent())>1000) {
56                         printElement(termek);
57                     }
58                 }
59             }
60         }
61     }
62
63     private static void Lekerdezes2(Document doc) {
64         NodeList vevok = doc.getElementsByTagName("vasarlo");
65         for (int i = 0; i < vevok.getLength(); i++) {
66             Element vevok = (Element)vevok.item(i);
67             if (vevok.getAttributeNode("id").getValue().equals("V13"))
68                 printElement(vevok);
69         }
70     }
71 }

```

```
73     }
74
75     private static void printElement(Element elem) {
76         String id = elem.getAttribute("id");
77         System.out.println("    ID: " + id);
78
79         String nodeContents="";
80         NodeList childNodes = elem.getChildNodes();
81         for(int j =0;j<childNodes.getLength() ; j++) {
82             if(childNodes.item(j).getTextContent().trim()!="") {
83                 nodeContent = normalizeText(childNodes.item(j).getTextContent().trim());
84                 System.out.println(childNodes.item(j).getNodeName()+" : "+nodeContent);
85             }
86         }
87     }
88 }
89
90 }
91
```