JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Készítette: **Veres Marcell**

Neptunkód: **JEDU1N**

A feladat leírása:

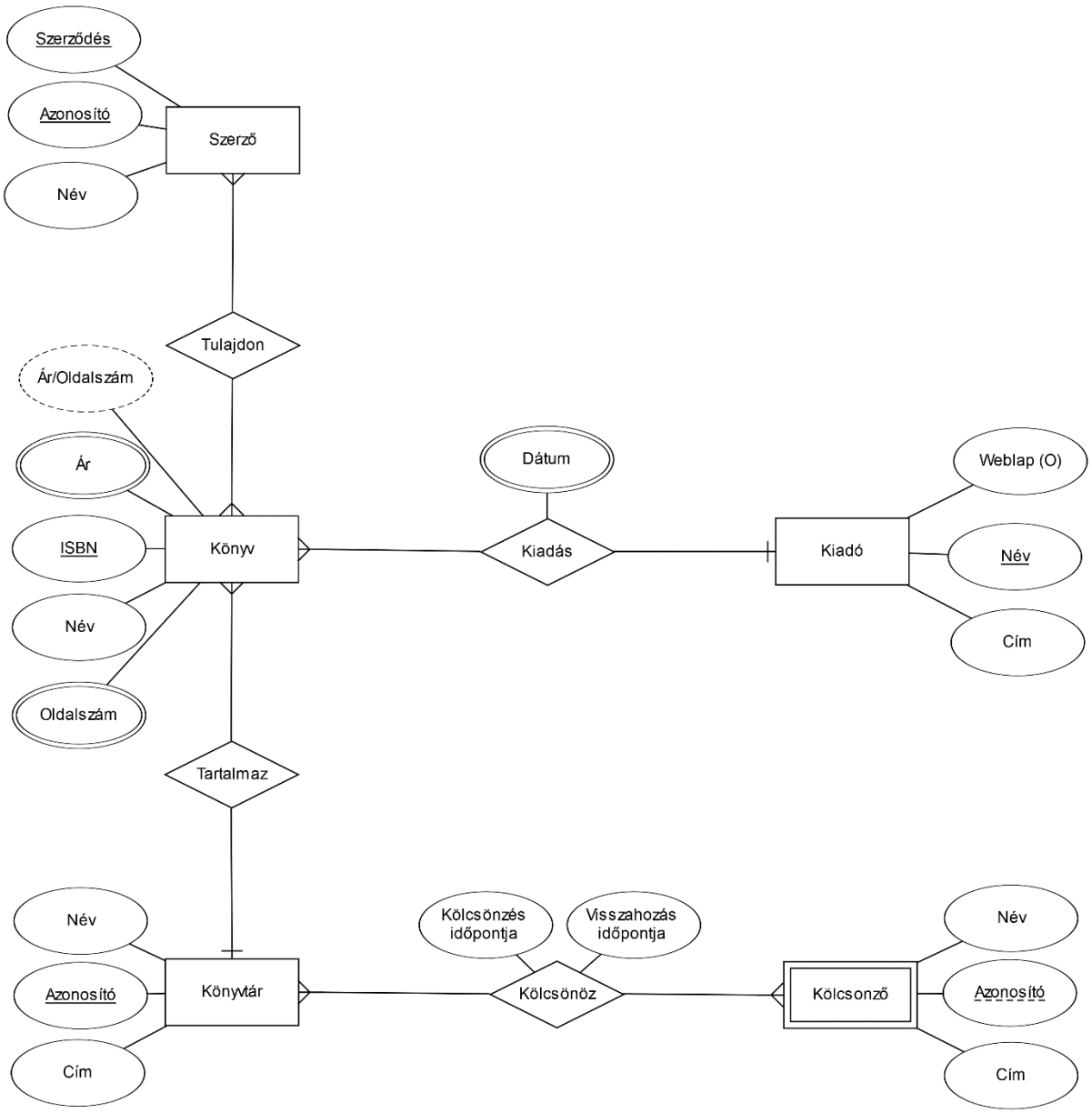
A feladatomban egy 5 egyedből álló adatbázis modellt szeretnék bemutatni, amely a könyvek, kiadók, könyvtárak, stb. kapcsolatáról szól.

Egy könyvtár könyveinek nyilvántartása magában is egy nagy feladat, de tudni azt minden egyes könyv esetén, hogy mikor kölcsönözték ki, kinek a könyvét, az adott könyv milyen tulajdonságokkal bír, stb. már egy nehezebb, valamint komplexebb feladatnak bizonyul.

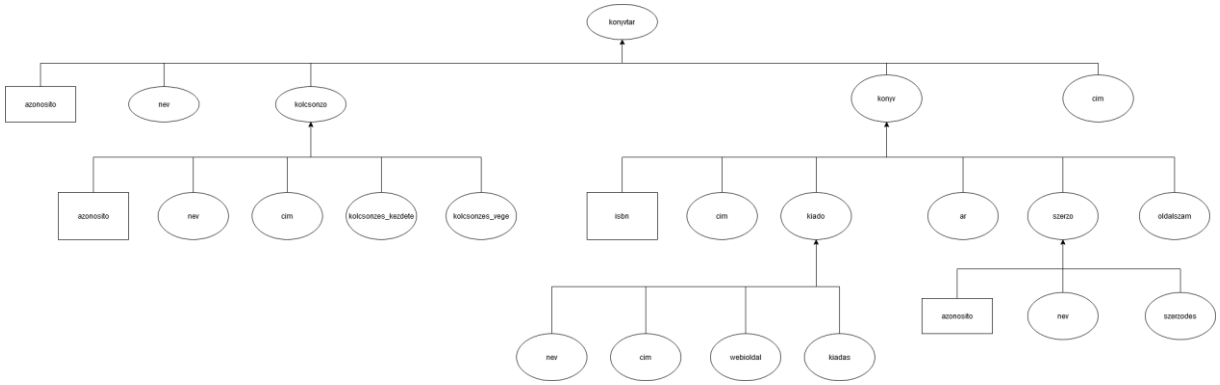
Ezen problémák, nehézségek elkerülése végett készítettem el az alábbiakban részletezésre kerülő adatbázist.

1 feladat

1a) Az adatbázis ER modell:



1b) Az adatbázis konvertálása XDM modellre:



1c) Az XDM modell alapján XML dokumentum készítése:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<konyvtar xmlns="https://www.w3schools.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" azonosito="1" xsi:schemaLocation="https://www.w3schools.com VM_XMLSchema.xsd">
    <nev>Rakoczi Megyei Konyvtar</nev>
    <cim>3500 Miskolc, Gorgey Artur 765</cim>

    <kolcsonzo azonosito="1">
        <nev>Kiss Pista</nev>
        <cim>2345 Kukutyin, Arany utca 56</cim>
        <kolcsonzes_kezdete>2019-07-07</kolcsonzes_kezdete>
        <kolcsonzes_vege>2020-03-03</kolcsonzes_vege>
    </kolcsonzo>

    <konyv isbn="0123456789">
        <cim>Egri Csillagok</cim>
        <ar>2300</ar>
        <oldalszam>300</oldalszam>

        <kiado>
            <nev>LegjobbKiado</nev>
            <cim>4560 Balaton, Fo utca 23</cim>
            <weboldal>www.legjobbkiado.hu</weboldal>
            <kiadas>1976-01-06</kiadas>
        </kiado>

        <szerzo azonosito="1">
            <nev>Gardonyi Geza</nev>
            <szerzodes>LegjobbKiado</szerzodes>
        </szerzo>
    </konyv>

</konyvtar>
```

1d) Az XML dokumentum alapján XMLSchema készítése:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="https://www.w3schools.com"
xmlns="https://www.w3schools.com" elementFormDefault="qualified">
    <xs:element name="konyvtar">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="nev" minOccurs="0" maxOccurs="1" type="xs:string" />
                <xs:element name="cim" minOccurs="0" maxOccurs="1" type="xs:string" />

                <xs:element name="kolcsonzo">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="nev" minOccurs="0" maxOccurs="1" type="xs:string" />
                            <xs:element name="cim" minOccurs="0" maxOccurs="1" type="xs:string" />
                            <xs:element name="kolcsonzes_kezdete" minOccurs="0" maxOccurs="1"
                               type="xs:date" />
                            <xs:element name="kolcsonzes_vege" minOccurs="0" maxOccurs="1"
                               type="xs:date" />
                        </xs:sequence>
                        <xs:attribute name="azonosito" type="xs:int" />
                    </xs:complexType>
                </xs:element>

                <xs:element name="konyv">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="cim" minOccurs="0" maxOccurs="1" type="xs:string" />
                            <xs:element name="ar" minOccurs="0" maxOccurs="1" type="xs:int" />
                            <xs:element name="oldalszam" minOccurs="0" maxOccurs="1" type="xs:int" />

                            <xs:element name="kiado">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="nev" minOccurs="0" maxOccurs="1"
                                           type="xs:string" />
                                        <xs:element name="cim" minOccurs="0" maxOccurs="1"
                                           type="xs:string" />
                                        <xs:element name="weboldal" minOccurs="0" maxOccurs="1"
                                           type="xs:string" />
                                        <xs:element name="kiadas" minOccurs="0" maxOccurs="1"
                                           type="xs:date" />
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>

                            <xs:element name="szerzo">
                                <xs:complexType>
```

```
                                <xs:sequence>
                                    <xs:element name="nev" minOccurs="0" maxOccurs="1"
                                        type="xs:string" />
                                    <xs:element name="szerzodes" minOccurs="0" maxOccurs="1"
                                        type="xs:string" />
                                </xs:sequence>
                                <xs:attribute name="azonosito" type="xs:int" />
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                    <xs:attribute name="isbn" type="isbn" />
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="azonosito" type="xs:int" />
    </xs:complexType>
</xs:element>


<xs:simpleType name="isbn">
    <xs:restriction base="xs:int">
        <xs:pattern value="\d{10,10}" />
    </xs:restriction>
</xs:simpleType>

</xs:schema>
```

2 feladat

A feladat egy DOM program készítése az XML dokumentum adatainak adminisztrálása alapján:

2a) adatolvasás:

```
package com.meiit.xml.domparse;

import java.io.File;
import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.xml.sax.SAXException;

public class DOMRead
{
    private static String separator = "";
    private static String line = "--------------------------------------";
    private static int level = 0;// depth in tree

    /**
     * Gets an XML file and route element.
     * Calls printDocument method with parameter: route element.
     */
    public static void main(String[] args) throws ParserConfigurationException, SAXException,
    IOException
    {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newDefaultInstance();

            DocumentBuilder builder = factory.newDocumentBuilder();
            File xmlFile = new File("C:\\Users\\Marci\\OneDrive\\Egyetem\\V. félév\\XML\\Beadandó\\
            VM_XML.xml");
            Document document = builder.parse(xmlFile);

            Element rootNode = document.getDocumentElement();

            printDocument(rootNode);
    }
```

```java
    /**
     * Prints child elements of given node.
     */
    public static void printDocument(Node rootNode)
    {

        System.out.println(separator + rootNode.getNodeName());
        System.out.println(separator + line);

        separator += "  ";
        level++;

        Node childNode = rootNode.getFirstChild();

        while (childNode != null)
        {

            if (childNode.getNodeType() == Node.ELEMENT_NODE)
            {

                boolean nodeIsComplex = childNode.getTextContent().contains("\n");

                if (nodeIsComplex == false)
                {
                    System.out.print(separator + childNode.getNodeName());
                    System.out.println(": " + childNode.getTextContent());
                }

                else
                {
                    System.out.println("");
                    printDocument(childNode);
                    level--;
                    separator = separator.substring(0, separator.length() - 2);
                }
            }

            childNode = childNode.getNextSibling();
        }
    }
}
```

2b) adatmódosítás:

```
package com.meiit.xml.domparse;

import java.io.File;

public class DOMModify
{
    /**
     * Gets an XML file and route element.
     * Calls findParentNode method with the following parameters:
     * route element, name of node we are looking for.
     *
     * Calls modifyNodeValue method with the following parameters:
     * parent node, name of child element, new value of child element.
     *
     * Calls modifyDocument method with the following parameters:
     * document, result file.
     */
    public static void main(String[] args) throws ParserConfigurationException, SAXException,
    IOException, TransformerException
    {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newDefaultInstance();

        DocumentBuilder builder = factory.newDocumentBuilder();
        File xmlFile = new File("C:\\Users\\Marci\\OneDrive\\Egyetem\\V. félév\\XML\\Beadandó\\
        VM_XML.xml");
        Document document = builder.parse(xmlFile);
        Element rootNode = document.getDocumentElement();

        String parentNode = "kolcsonzo";
        String node = "nev";
        String newValue = "Kiss Pista";

        if (rootNode.getNodeName() == parentNode)
        {
            modifyNodeValue(rootNode, node, newValue);
        }
        else
        {
            Node parent = findParentNode(rootNode, parentNode);

            if (parent != null)
            {
                modifyNodeValue(parent, node, newValue);
                modifyDocument(document, xmlFile);
            }

        }
    }
```

```java
/**
 * Returns the parent node we are looking for.
 */
public static Node findParentNode(Node rootNode, String parentNode)
{
    Node childNode = rootNode.getFirstChild();

    while (childNode != null)
    {

        if (childNode.getNodeType() == Node.ELEMENT_NODE)
        {

            boolean nodeIsComplex = childNode.getTextContent().contains("\n");

            if (nodeIsComplex == true)
            {

                if (childNode.getNodeName() == parentNode)
                {
                    return childNode;
                }
                else
                {
                    findParentNode(childNode, parentNode);
                }
            }

        }

        childNode = childNode.getNextSibling();
    }
    return null;
}
```

```java
    /**
     * Modifies the specified value of node.
     */
    public static void modifyNodeValue(Node parentNode, String node, String newValue)
    {
        Node childNode = parentNode.getFirstChild();

        while (childNode != null)
        {

            if (childNode.getNodeType() == Node.ELEMENT_NODE)
            {

                boolean nodeIsComplex = childNode.getTextContent().contains("\n");

                if (nodeIsComplex == false)
                {

                    if (childNode.getNodeName() == node)
                    {
                        childNode.setTextContent(newValue);
                    }
                }

            }

            childNode = childNode.getNextSibling();
        }
    }

    /**
     * Modifies the given XML document.
     */
    public static void modifyDocument(Document document, File xmlFile) throws TransformerException
    {
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        DOMSource source = new DOMSource(document);
        StreamResult result = new StreamResult(xmlFile);

        transformer.transform(source, new StreamResult(System.out));
        transformer.transform(source, result);
    }
}
```