Adatkezelés XML-ben

BSc
Féléves feladat
2023

Készítette:
Szendrei Gábor
Programtervezõ Informatikus
V9ZK10

Tartalomjegyzék

1a) A feladat témája

A beadanóm témája egy olyan adatbázis, amely autos cégeket és hozzújuk kapcsolódó egyedeket tartja nyilván.

- **Autóscég**

  ceg_kod – Elsődleges kulcs

  cegnev – Cég nevét tartalmazza

  helyrajziSzam – Cég elhelyezkedését tartalmazza

  dolgozokSzama – Tárolja a dolgozók számát.

- **Vásárlók**

  jog_kod – Elsődleges kulcs

  email_cim – Vásárló email címe

  cim – Összetett tulajdonság (Orszag, isz, varos, uHsz)

  Telefonszam – Többértékű tulajdonság

- **Autós adatok**

  forgalmi_kod – Elsődleges kulcs

  ar – Mennyibe kerül az eladásra váró auto

  kmOra – Mennyi kilómérert tettek már meg az autóval

  statusz – Szöveges típús amibe leírást lehet adni, pl. felújított

- **Autós típus**

  tipus_kod – Elsődleges kulcs

  gyartasiEv – Mikor gyártották az adott autót

  marka – Az autó márkáját tartalmazza

  nev – Az auto teljes nevét tartalmazza

- **Számlázás**

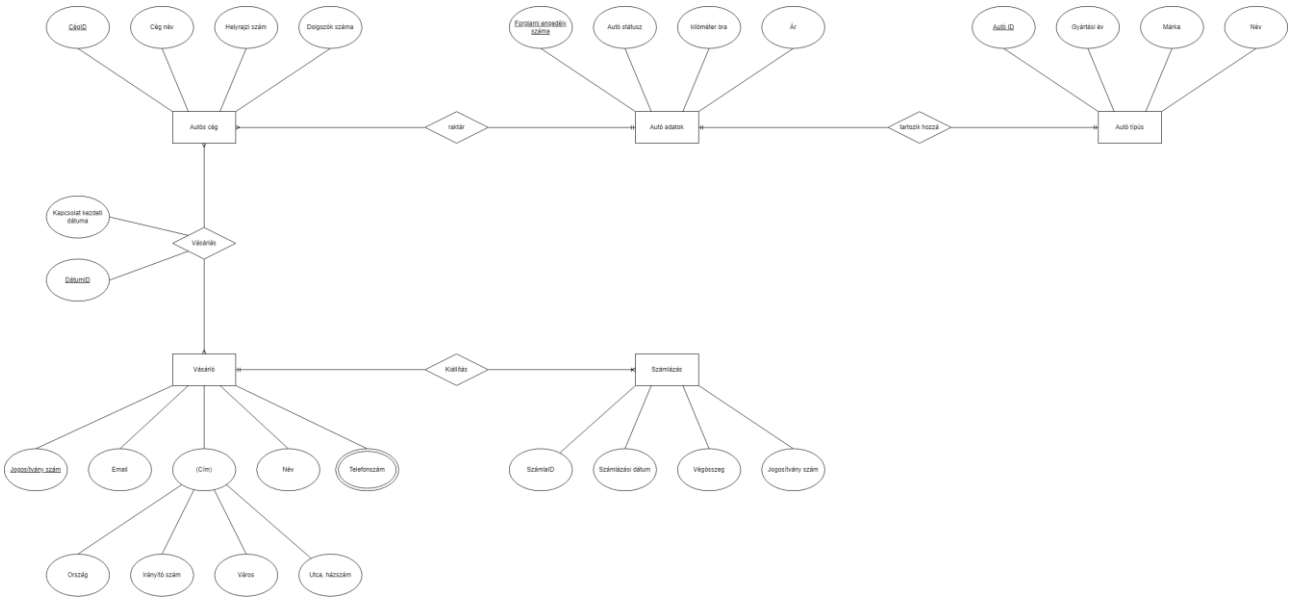
  szamlakod– Elsődleges kulcs

  szamlaDatum – Mikor állították ki a számlát

  végösszeg – adókkal mindennel együtt mennyit fizettek

  jogsiszam – A vásárló azonosítására szolgál

A feladat ER modellje:



Az Egyedek közötti kapcsolatok

Autós cég – Autó adatok: 1:n Egy autó több cégnél is megtud fordulni.

Autó adatok – Autó típús 1:1 Egy auto adataihoz egy típús tartozhat csak.

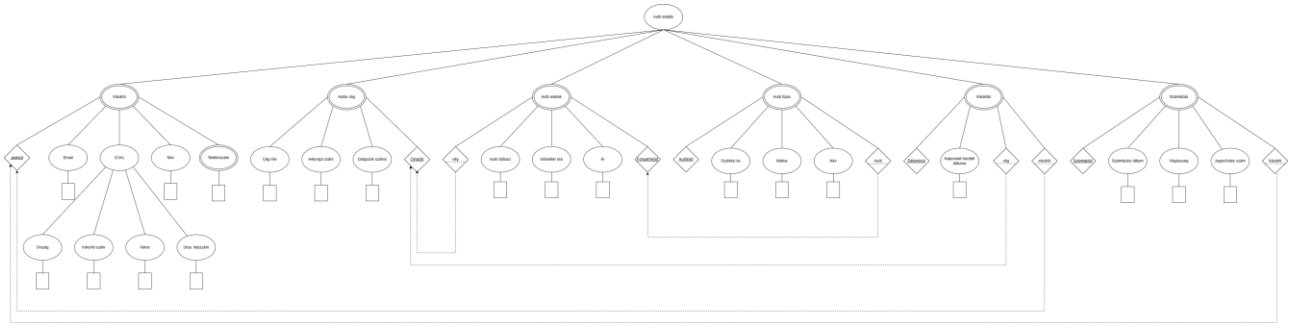
Autós cég – Vásárló n:m Több vásárló több céghez tartozhat.

Vásárló – Számlázás Egy vásárlóhoz több számla is tartozhat.

1b) Az ER-modell konvertálása XDM modellre

XDM modellnél háromféle jelölést alkalmazhatunk. Ezek az ellipszis, a rombusz, illetve a téglalap. Az ellipszis jelöli az elemeket minden egyedből elem lesz, ezen felül a tulajdonságokból is. A rombusz jelöli az attribútumokat, amelyek a kulcs tulajdonságokból keletkeznek. A téglalap jelöli a szöveget, amely majd az XML dokumentumban fog megjelenni. Azoknak az elemeknek, amelyek többször is előfordulhatnak, a jelölése dupla ellipszissel történik. Az idegenkulcsok és a kulcsok közötti kapcsolatot szaggatott vonalas nyíllal jelöljük.

A feladat XDM modellje:



1c) XML Dokumentum készítése

Az XDM modell alapján az XML dokumentumot úgy készítettem el, hogy először is a gyökér elementtel kezdtem, ami az AutosCegERV9ZK10 volt. A gyermek elemeiből 3-3 példányt hoztam létre, ezeknek az elemeknek az attribútumai közé tartoznak a kulcsok, illetve idegenkulcsok is, mindezek után ezeknek az elemeknek létrehoztam a többi gyermek elementet is. XML dokumentum forráskódja

```xml
<?xml version="1.0" encoding="UTF-8"?>
<AutosCegERV9ZK10 xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:noNamespaceSchemaLocation="ERV9ZK10.xsd">
    <Vasarlo jogkod="01">
        <email_cim>jane.smith456@example.com</email_cim>
        <cim>
            <orszag>Magyarorszag</orszag>
            <isz>1234</isz>
            <varos>Budapest</varos>
            <uHsz>Kossuth utca 10</uHsz>
        </cim>
        <nev>Kovacs Peter</nev>
        <Telefonszam>0630-987-6543</Telefonszam>
        <Telefonszam>0630-987-6453</Telefonszam>
    </Vasarlo>

    <Vasarlo jogkod="02">
        <email_cim>john.doe456@example.co.uk</email_cim>
        <cim>
            <orszag>Magyarorszag</orszag>
            <isz>5678</isz>
            <varos>Szeged</varos>
            <uHsz>Rakoczi ut 5</uHsz>
        </cim>
        <nev>Nagy Anna</nev>
        <Telefonszam>0620-765-4321</Telefonszam>
        <Telefonszam>0620-765-0000</Telefonszam>
    </Vasarlo>
```

```xml
<Vasarlo jogkod="03">
    <email_cim>mark.johnson789@example.co.uk</email_cim>
    <cim>
        <orszag>Magyarorszag</orszag>
        <isz>9876</isz>
        <varos>Debrecen</varos>
        <uHsz>Petofi ter 3</uHsz>
    </cim>
    <nev>Kiss Eva</nev>
    <Telefonszam>0612-345-6789</Telefonszam>
    <Telefonszam>0630-987-6543</Telefonszam>
</Vasarlo>

<AutosCeg ceg_kod="11">
    <cegnev>AutaPlusz Kft.</cegnev>
    <helyrajziSzam>BUD-123456</helyrajziSzam>
    <dolgozokSzama>15</dolgozokSzama>
</AutosCeg>

<AutosCeg ceg_kod="12">
    <cegnev>Autovilag Bt.</cegnev>
    <helyrajziSzam>DEB-654321</helyrajziSzam>
    <dolgozokSzama>10</dolgozokSzama>
</AutosCeg>

<AutosCeg ceg_kod="13">
    <cegnev>Kocsis Kft.</cegnev>
    <helyrajziSzam>SZEG-987654</helyrajziSzam>
    <dolgozokSzama>20</dolgozokSzama>
</AutosCeg>
```

```xml
<AutosAdatok forgalmi_kod="21" ceg_kod="11">
    <ar>45000</ar>
    <kmOra>78900</kmOra>
    <statusz>uj</statusz>
</AutosAdatok>

<AutosAdatok forgalmi_kod="22" ceg_kod="11">
    <ar>55000</ar>
    <kmOra>65400</kmOra>
    <statusz>hasznalt</statusz>
</AutosAdatok>

<AutosAdatok forgalmi_kod="23" ceg_kod="13">
    <ar>60000</ar>
    <kmOra>89000</kmOra>
    <statusz>hasznalt</statusz>
</AutosAdatok>

<AutosTipus tipus_kod="31" forgalmi_kod="21">
    <gyartasiEv>2022</gyartasiEv>
    <marka>Toyota</marka>
    <nev>Corolla Hybrid</nev>
</AutosTipus>

<AutosTipus tipus_kod="32" forgalmi_kod="22">
    <gyartasiEv>2021</gyartasiEv>
    <marka>Volkswagen</marka>
    <nev>Golf GTI</nev>
</AutosTipus>
```

```
    <AutosTipus tipus_kod="33" forgalmi_kod="23">
        <gyartasiEv>2020</gyartasiEv>
        <marka>Ford</marka>
        <nev>Focus Titanium</nev>
    </AutosTipus>

    <Vasarlas datumkod="41" ceg_kod="11" vasarlo_kod="01">
        <kezdDatum>2023-01-15</kezdDatum>
    </Vasarlas>

    <Vasarlas datumkod="42" ceg_kod="12" vasarlo_kod="02">
        <kezdDatum>2023-02-28</kezdDatum>
    </Vasarlas>

    <Vasarlas datumkod="43" ceg_kod="13" vasarlo_kod="02">
        <kezdDatum>2023-03-10</kezdDatum>
    </Vasarlas>

    <Szamlazas szamlakod="51" vasarlo_kod="01">
        <szamlaDatum>2023-01-20</szamlaDatum>
        <vegosszeg>55000</vegosszeg>
        <jogsiszam>54321</jogsiszam>
    </Szamlazas>

    <Szamlazas szamlakod="52" vasarlo_kod="02">
        <szamlaDatum>2023-03-01</szamlaDatum>
        <vegosszeg>60000</vegosszeg>
        <jogsiszam>65432</jogsiszam>
    </Szamlazas>
```

```
    <Szamlazas szamlakod="53" vasarlo_kod="03">
        <szamlaDatum>2023-04-12</szamlaDatum>
        <vegosszeg>70000</vegosszeg>
        <jogsiszam>76543</jogsiszam>
    </Szamlazas>

</AutosCegERV9ZK10>
```

XML dokumentum alapján XMLSchema készítése

Az ERV9ZK10.xsd séma file leírja mindazon megkötéseket, amelyeknek az XML dokumentumnak meg kell felelnie. Itt definiálunk minden típust, amit az XML file-ban használni szeretnénk, valamint az adatbázis kapcsolatait xs:unique és xs:keyref bejegyzésekkel hozom létre. . Az XML Schémám meghatározza az adatokat, mint például a cég nevét, vásárlók címét, telefonszámokat, amelyeket egy telefonszamTipus típus korlátozza, hogy csak bizonyos formátumú (4 szám – 3 szám – 4 szám) fogadja el. Továbbá létezik egy emailCimTipus is, amely szabályozza az emailcím formátumát, amit elfogad. Komplex típusokat is definiáltam, például a vasarloTipus, amiben egy komplex típus definiálja a cím formátumát, autosCegTipus, autosAdatokTipus, autosTipusTipus, vasarlasTipus, szamlazasTipus. Az adatbázis integritásának megőrzése érdekében elsődleges (PK) illetve idegen kulcsok (FK) meghatározására került sor. Az XML séma így biztosítja, hogy az adatok szerkezete és kapcsolatai érvényesek és következetesek legyenek.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <!-- Define simple types -->
    <xs:element name="nev" type="xs:string" />
    <xs:element name="cegnev" type="xs:string" />
    <xs:element name="helyrajziSzam" type="xs:string" />
    <xs:element name="dolgozokSzama" type="xs:positiveInteger" />
    <xs:element name="ar" type="xs:positiveInteger" />
    <xs:element name="kmOra" type="xs:positiveInteger" />
    <xs:element name="statusz" type="xs:string" />
    <xs:element name="gyartasiEv" type="xs:positiveInteger" />
    <xs:element name="marka" type="xs:string" />
    <xs:element name="kezdDatum" type="xs:date" />
    <xs:element name="szamlaDatum" type="xs:date" />
    <xs:element name="vegosszeg" type="xs:positiveInteger" />
    <xs:element name="jogsiszam" type="xs:positiveInteger" />

    <xs:simpleType name="emailCimTipus">
        <xs:restriction base="xs:string">
            <xs:pattern
                value="([0-9a-zA-Z]([-.\w]*[0-9a-zA-Z])*@([0-9a-zA-Z][-\w]*[0-9a-zA-Z]\.)+[a-zA-Z]{2,9})" />
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="telefonszamTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="\d{4}-\d{3}-\d{4}" />
        </xs:restriction>
    </xs:simpleType>
```

```
<!-- Define complex types -->
<xs:complexType name="vasarloTipus">
    <xs:sequence>
        <xs:element name="email_cim" type="emailCimTipus" />
        <xs:element name="cim">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="orszag" type="xs:string" />
                    <xs:element name="isz" type="xs:positiveInteger" />
                    <xs:element name="varos" type="xs:string" />
                    <xs:element name="uHsz" type="xs:string" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element ref="nev" />
        <xs:element name="Telefonszam" type="telefonszamTipus" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="jogkod" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="autosCegTipus">
    <xs:sequence>
        <xs:element ref="cegnev" />
        <xs:element ref="helyrajziSzam" />
        <xs:element ref="dolgozokSzama" />
    </xs:sequence>
    <xs:attribute name="ceg_kod" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="autosAdatokTipus">
    <xs:sequence>
        <xs:element ref="ar" />
        <xs:element ref="kmOra" />
        <xs:element ref="statusz" />
    </xs:sequence>
    <xs:attribute name="forgalmi_kod" type="xs:integer" use="required" />
    <xs:attribute name="ceg_kod" type="xs:integer" use="required" />
</xs:complexType>
```

```
<xs:complexType name="autosTipusTipus">
    <xs:sequence>
        <xs:element ref="gyartasiEv" />
        <xs:element ref="marka" />
        <xs:element ref="nev" />
    </xs:sequence>
    <xs:attribute name="tipus_kod" type="xs:integer" use="required" />
    <xs:attribute name="forgalmi_kod" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="vasarlasTipus">
    <xs:sequence>
        <xs:element ref="kezdDatum" />
    </xs:sequence>
    <xs:attribute name="datumkod" type="xs:integer" use="required" />
    <xs:attribute name="ceg_kod" type="xs:integer" use="required" />
    <xs:attribute name="vasarlo_kod" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="szamlazasTipus">
    <xs:sequence>
        <xs:element ref="szamlaDatum"></xs:element>
        <xs:element ref="vegosszeg"></xs:element>
        <xs:element ref="jogsiszam"></xs:element>
    </xs:sequence>
    <xs:attribute name="szamlakod" type="xs:integer" use="required" />
    <xs:attribute name="vasarlo_kod" type="xs:integer" use="required" />
</xs:complexType>
```

```xml
<!-- Root element with keys and keyrefs -->
<xs:element name="AutosCegERV9ZK10">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Vasarlo" type="vasarloTipus" minOccurs="0" maxOccurs="100" />
            <xs:element name="AutosCeg" type="autosCegTipus" minOccurs="0" maxOccurs="100" />
            <xs:element name="AutosAdatok" type="autosAdatokTipus" minOccurs="0"
                maxOccurs="unbounded" />
            <xs:element name="AutosTipus" type="autosTipusTipus" minOccurs="0"
                maxOccurs="unbounded" />
            <xs:element name="Vasarlas" type="vasarlasTipus" minOccurs="0"
                maxOccurs="unbounded" />
            <xs:element name="Szamlazas" type="szamlazasTipus" minOccurs="0"
                maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>

    <!-- Primary keys -->
    <xs:key name="vasarlo_kulcs">
        <xs:selector xpath="Vasarlo" />
        <xs:field xpath="@jogkod" />
    </xs:key>
    <xs:key name="AutosCeg_kulcs">
        <xs:selector xpath="AutosCeg" />
        <xs:field xpath="@ceg_kod" />
    </xs:key>
    <xs:key name="AutosAdatok_kulcs">
        <xs:selector xpath="AutosAdatok" />
        <xs:field xpath="@forgalmi_kod" />
    </xs:key>
    <xs:key name="AutosTipus_kulcs">
        <xs:selector xpath="AutosTipus" />
        <xs:field xpath="@tipus_kod" />
    </xs:key>
    <xs:key name="Vasarlas_kulcs">
        <xs:selector xpath="Vasarlas" />
        <xs:field xpath="@datumkod" />
    </xs:key>
```

```
        <xs:key name="Vasarlas_kulcs">
            <xs:selector xpath="Vasarlas" />
            <xs:field xpath="@datumkod" />
        </xs:key>
        <xs:key name="Szamlazas_kulcs">
            <xs:selector xpath="Szamlazas" />
            <xs:field xpath="@szamlakod" />
        </xs:key>

        <!-- Foreign keys -->
        <xs:keyref name="AutosCeg_ceg_kulcs" refer="AutosCeg_kulcs">
            <xs:selector xpath="AutosAdatok" />
            <xs:field xpath="@ceg_kod" />
        </xs:keyref>
        <xs:keyref name="Auto_tipus_AutosAdatok_kulcs" refer="AutosAdatok_kulcs">
            <xs:selector xpath="AutosTipus" />
            <xs:field xpath="@forgalmi_kod" />
        </xs:keyref>
        <xs:keyref name="Vasarlas_ceg_kulcs" refer="AutosCeg_kulcs">
            <xs:selector xpath="Vasarlas" />
            <xs:field xpath="@ceg_kod" />
        </xs:keyref>
        <xs:keyref name="Vasarlas_vasarlo_kulcs" refer="vasarlo_kulcs">
            <xs:selector xpath="Vasarlas" />
            <xs:field xpath="@vasarlo_kod" />
        </xs:keyref>
        <xs:keyref name="Szamlazas_vasarlo_kulcs" refer="vasarlo_kulcs">
            <xs:selector xpath="Szamlazas" />
            <xs:field xpath="@vasarlo_kod" />
        </xs:keyref>
        <!-- Unique constraint for 1:1 relationship -->
        <xs:unique name="AutosAdatok_Auto_tipusok_egyegy">
            <xs:selector xpath="AutosTipus" />
            <xs:field xpath="@forgalmi_kod" />
        </xs:unique>
    </xs:element>
</xs:schema>
```

Adatolvasás

A kód egy Java alapú XML feldolgozó program, amely a DOM (Document Object Model) parserét használja. A DOM parser a teljes XML dokumentumot memóriába tölti, ami gyors hozzáférést biztosít az elemekhez, de nagyobb dokumentumok esetén jelentős memóriaigényt jelenthet. A program beolvassa az XML fájlt, normalizálja azt, és különböző függvények segítségével feldolgozza az XML elemeket, melyek az 'AutosCeg', 'AutosAdatok', AutosTipus,'Vasarlo', Szamlazas. Minden elemcsoport feldolgozása külön függvényben történik, ami javítja a kód olvashatóságát és karbantarthatóságát. Hibakezelés is implementálva van a fájlbeolvasás és parse-lás során.

```java
package domparse;

import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.*;
import java.util.StringJoiner;

public class DomReadV9ZK10 {
    Run | Debug
    public static void main(String[] args) {
        try {
            File inputFile = new File(pathname:"XMLTaskV9ZK10\\ERV9ZK10.xml");

            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            File outputFile = new File(pathname:"XMLTaskV9ZK10\\ERV9ZK10_1.xml");
            PrintWriter writer = new PrintWriter(new FileWriter(outputFile, append:true));

            // kiirja az XML prológust
            System.out.print(s:"<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
            writer.print(s:"<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");

            // Gyökér elem kiiratása
            Element rootElement = doc.getDocumentElement();
            String rootName = rootElement.getTagName();
            StringJoiner rootAttributes = new StringJoiner(delimiter:" ");
            NamedNodeMap rootAttributeMap = rootElement.getAttributes();

            for (int i = 0; i < rootAttributeMap.getLength(); i++) {
                Node attribute = rootAttributeMap.item(i);
                rootAttributes.add(attribute.getNodeName() + "=\"" + attribute.getNodeValue() + "\"");
            }

            System.out.print("<" + rootName + " " + rootAttributes.toString() + ">\n");
            writer.print("<" + rootName + " " + rootAttributes.toString() + ">\n");
```

```java
            // kiirja az az elemek nevét
            printNodeList(doc.getElementsByTagName(tagname:"Vasarlo"), writer);
            printNodeList(doc.getElementsByTagName(tagname:"AutosCeg"), writer);
            printNodeList(doc.getElementsByTagName(tagname:"AutosAdatok"), writer);
            printNodeList(doc.getElementsByTagName(tagname:"AutosTipus"), writer);
            printNodeList(doc.getElementsByTagName(tagname:"Vasarlas"), writer);
            printNodeList(doc.getElementsByTagName(tagname:"Szamlazas"), writer);

            // Lezárja az elemet
            System.out.println("</" + rootName + ">");
            writer.append("</" + rootName + ">");

            writer.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void printNodeList(NodeList nodeList, PrintWriter writer) {
        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            printNode(node, indent:1, writer);
            System.out.println(x:"");
            writer.println(x:"");
        }
    }
```

```
    private static void printNode(Node node, int indent, PrintWriter writer) {
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;
            String nodeName = element.getTagName();
            StringJoiner attributes = new StringJoiner(delimiter:" ");
            NamedNodeMap attributeMap = element.getAttributes();

            for (int i = 0; i < attributeMap.getLength(); i++) {
                Node attribute = attributeMap.item(i);
                attributes.add(attribute.getNodeName() + "=\"" + attribute.getNodeValue() + "\"");
            }

            System.out.print(getIndentString(indent));
            System.out.print("<" + nodeName + " " + attributes.toString() + ">");
            writer.print(getIndentString(indent));
            writer.print("<" + nodeName + " " + attributes.toString() + ">");

            NodeList children = element.getChildNodes();

            if (children.getLength() == 1 && children.item(index:0).getNodeType() == Node.TEXT_NODE) {
                System.out.print(children.item(index:0).getNodeValue());
                writer.print(children.item(index:0).getNodeValue());
            } else {
                System.out.println();
                writer.println();
                for (int i = 0; i < children.getLength(); i++) {
                    printNode(children.item(i), indent + 1, writer);
                }
                System.out.print(getIndentString(indent));
                writer.print(getIndentString(indent));
            }

            System.out.println("</" + nodeName + ">");
            writer.println("</" + nodeName + ">");
        }
    }

    private static String getIndentString(int indent) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < indent; i++) {
            sb.append(str:"  ");
        }
        return sb.toString();
    }
}
```

Adatírás

A kód fő funkciója, hogy beolvassa és kiírja az XML tartalmakat a konzolra és fájlba is. A {main metódusban az XMLV9ZK10 fájlt olvassa be a megadott útvonalról, használva a DocumentBuilderFactory és DocumentBuilder osztályokat az XML struktúra elemzéséhez. Ezután normalizálja a dokumentumot a normalize metódussal, ami segít a DOM fa struktúrájának rendezésében.

```java
package domparse;

import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.*;

public class DomWriteV9ZK10 {

    Run | Debug
    public static void main(String[] args) {
        try {
            File inputFile = new File(pathname:"XMLTaskV9ZK10\\ERV9ZK10.xml");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            printNode(doc.getDocumentElement(), depth:0);
            writeDocumentToFile(doc, filename:"XMLTaskV9ZK10\\ERV9ZK10_1.xml");

            System.out.print(s:"Output written to the file successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void printNode(Node node, int depth) {
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            String tagName = node.getNodeName();
            NamedNodeMap attributes = node.getAttributes();

            printIndentation(depth);
            System.out.print("<" + tagName);
```

```
            // Kiirja az attribútomot ha létezik
            if (attributes.getLength() > 0) {
                for (int i = 0; i < attributes.getLength(); i++) {
                    Node attribute = attributes.item(i);
                    System.out.print(" " + attribute.getNodeName() + "=\"" + attribute.getTextContent() + "\"");
                }
            }

            NodeList children = node.getChildNodes();
            // If there are child elements, print them
            if (children.getLength() > 1) {
                System.out.println(x:">");
                for (int i = 0; i < children.getLength(); i++) {
                    Node child = children.item(i);
                    if (child.getNodeType() == Node.ELEMENT_NODE) {
                        printNode(child, depth + 1);
                    }
                }
                printIndentation(depth);
                System.out.println("</" + tagName + ">");
                System.out.println(x:"");
            } else {
                // Ha nincs gyerek elem kiirja a tartalmat
                String textContent = node.getTextContent().trim();
                if (!textContent.isEmpty()) {
                    System.out.println(">" + textContent + "</" + tagName + ">");
                } else {
                    System.out.println(x:"/>");
                }
            }
        }
    }
```

```
    private static void printIndentation(int depth) {
        for (int i = 0; i < depth; i++) {
            System.out.print(s:"  ");
        }
    }

    private static void writeDocumentToFile(Document doc, String filename) throws TransformerException {
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.INDENT, value:"yes");
        transformer.setOutputProperty(name:"{http://xml.apache.org/xslt}indent-amount", value:"2"); // Behúzás szabályozása

        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(new File(filename));
        transformer.transform(source, result);
    }
}
```

Adatmódosítás

Ez a Java program, a DOMModifyV9ZK10, egy XML fájlt olvas be és módosítja azt a DOM(Document Object Model) API segítségével. Az XML fájl, amely egy állatkert adatait tartalmazza, egy előre meghatározott útvonalon található, és a DocumentBuilder osztály segítségévelparse-oljuk. A program két módosítást hajt végre, az első ilyen, hogy a vásárló első elemének város attribútumát Budapest értékre állítja. A második módosítás az autós adatok első elemének árát módosítja 600000-re. A módosítások után a program egy Transformer segítségével visszaalakítja és kiírja a módosított DOM-ot XML formátumban. Az XML kiírás során a Transformer beállításai biztosítják a formázott, olvasható kimenetet, az OutputKeys.INDENT beállítás segítségével.

```java
package domparse;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

public class DomModifyV9ZK10 {
    Run | Debug
    public static void main(String[] args) {
        try {

            File xmlFile = new File(pathname:"XMLTaskV9ZK10\\ERV9ZK10_1.xml");

            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

            Document doc = dBuilder.parse(xmlFile);

            // első vásárló városának módosítása
            NodeList vasarloList = doc.getElementsByTagName(tagname:"Vasarlo");
            Element vasarlo = (Element) vasarloList.item(index:0);
            vasarlo.getElementsByTagName(name:"varos").item(index:0).setTextContent(textContent:"Budapest");

            // az első autó adatok árának módosítása
            NodeList autosAdatokList = doc.getElementsByTagName(tagname:"Autos_adatok");
            Element autosAdatok = (Element) autosAdatokList.item(index:0);
            autosAdatok.getElementsByTagName(name:"ar").item(index:0).setTextContent(textContent:"600000");
```

```
            // Kiiratja a konzolra a módosított XML-t
            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            DOMSource source = new DOMSource(doc);
            StreamResult consoleResult = new StreamResult(System.out);
            transformer.transform(source, consoleResult);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Adatlekérdezés

A program a Java Parser-t használja XML fájlunk feldolgozására, ami lehetővé teszi a XML elemek olvasását és manipulálását egy objektumorientált módon. A kód, az XML fájlt, a File objektumon keresztül tölti be, biztosítva ezzel a fájl elérését és kezelését. A DocumentBuilderFactory és DocumentBuilder osztályok használata a fájl DOM reprezentációjának létrehozásához szükséges, ami egy strukturált, fa-szerű modellt biztosít az XML adatok számára. A program minden egyes lekérdezést egy for ciklus segítségével hajt végre, ahol a getElementsByTagName metódus segítségével specifikus XML elemeket keres. Az elemek feldolgozása során a Node és Element interfészeket használja, amelyek lehetővé teszik az egyes elemek attribútumainak és tartalmának elérését. A lekérdezések eredményét egy StringBuilder objektumba gyűjti, amely hatékonyan kezeli a nagy mennyiségű stringek összefűzését. A kód, az összegyűjtött adatokat XML-szerű formátumban állítja elő.

```java
package 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.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DomQueryV9ZK10 {
    Run | Debug
    public static void main(String argv[]) throws SAXException, IOException, ParserConfigurationException {
        File xmlFile = new File(pathname:"XMLTaskV9ZK10\\ERV9ZK10.xml");

        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = factory.newDocumentBuilder();

        org.w3c.dom.Document doc = dBuilder.parse(xmlFile);
        doc.getDocumentElement().normalize();

        // Querying Vasarlo elements
        NodeList vasarloList = doc.getElementsByTagName(tagname:"Vasarlo");
        System.out.println(x:"\n----------Vásárlók----------");

        for (int i = 0; i < vasarloList.getLength(); i++) {
            Node node = vasarloList.item(i);

            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;

                String email = element.getElementsByTagName(name:"email_cim").item(index:0).getTextContent();
                String name = element.getElementsByTagName(name:"nev").item(index:0).getTextContent();
                NodeList phoneNumbers = element.getElementsByTagName(name:"Telefonszam");
```

```java
                // Extracting address information
                Element cimElement = (Element) element.getElementsByTagName(name:"cim").item(index:0);
                String orszag = cimElement.getElementsByTagName(name:"orszag").item(index:0).getTextContent();
                String isz = cimElement.getElementsByTagName(name:"isz").item(index:0).getTextContent();
                String varos = cimElement.getElementsByTagName(name:"varos").item(index:0).getTextContent();
                String uHsz = cimElement.getElementsByTagName(name:"uHsz").item(index:0).getTextContent();

                System.out.println("Email: " + email);
                System.out.println("Név: " + name);
                System.out.println("Cím: " + orszag + ", " + isz + " " + varos + ", " + uHsz);
                System.out.println(x:"Telefon:");

                for (int j = 0; j < phoneNumbers.getLength(); j++) {
                    String phoneNumber = phoneNumbers.item(j).getTextContent();
                    System.out.println("- " + phoneNumber);
                }

                System.out.println();
            }
        }

        // Querying AutosCeg elements
        NodeList autosCegList = doc.getElementsByTagName(tagname:"AutosCeg");
        System.out.println(x:"\n----------Autós cégek----------");

        for (int i = 0; i < autosCegList.getLength(); i++) {
            Node node = autosCegList.item(i);

            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;

                String companyCode = element.getAttribute(name:"ceg_kod");
                String companyName = element.getElementsByTagName(name:"cegnev").item(index:0).getTextContent();
                String employeeCount = element.getElementsByTagName(name:"dolgozokSzama").item(index:0).getTextContent();
```

```
            System.out.println("Cég kód: " + companyCode);
            System.out.println("Név: " + companyName);
            System.out.println("Dolgozók száma: " + employeeCount);
            System.out.println();
        }
    }

    // Querying AutosAdatok elements
    NodeList autosAdatokList = doc.getElementsByTagName(tagname:"AutosAdatok");
    System.out.println(x:"\n----------Autós adatok----------");

    for (int i = 0; i < autosAdatokList.getLength(); i++) {
        Node node = autosAdatokList.item(i);

        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;

            String forgalmiKod = element.getAttribute(name:"forgalmiKod");
            String adatokCompanyCode = element.getAttribute(name:"ceg_kod");
            String price = element.getElementsByTagName(name:"ar").item(index:0).getTextContent();
            String km = element.getElementsByTagName(name:"kmOra").item(index:0).getTextContent();
            String status = element.getElementsByTagName(name:"statusz").item(index:0).getTextContent();

            System.out.println("Forgalmi Kód: " + forgalmiKod);
            System.out.println("Cég kód: " + adatokCompanyCode);
            System.out.println("Price: " + price);
            System.out.println("KM: " + km);
            System.out.println("Status: " + status);
            System.out.println();
        }
    }

    // Querying AutosTipus elements
    NodeList autosTipusList = doc.getElementsByTagName(tagname:"AutosTipus");
    System.out.println(x:"\n----------Autós típusok----------");
```

```
    for (int i = 0; i < autosTipusList.getLength(); i++) {
        Node node = autosTipusList.item(i);

        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;

            String tipusKod = element.getAttribute(name:"tipus_kod");
            String tipusForgalmiKod = element.getAttribute(name:"forgalmiKod");
            String gyev = element.getElementsByTagName(name:"gyartasiEv").item(index:0).getTextContent();
            String marka = element.getElementsByTagName(name:"marka").item(index:0).getTextContent();
            String nev = element.getElementsByTagName(name:"nev").item(index:0).getTextContent();

            System.out.println("Típus Kód: " + tipusKod);
            System.out.println("Forgalmi Kód: " + tipusForgalmiKod);
            System.out.println("Gyártási Év: " + gyev);
            System.out.println("Márka: " + marka);
            System.out.println("Név: " + nev);
            System.out.println();
        }
    }

    // Querying Vasarlas elements
    NodeList vasarlasList = doc.getElementsByTagName(tagname:"Vasarlas");
    System.out.println(x:"\n----------Vásárlások----------");

    for (int i = 0; i < vasarlasList.getLength(); i++) {
        Node node = vasarlasList.item(i);

        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;

            String vasarlasDatumkod = element.getAttribute(name:"datumkod");
            String vasarlasCompanyCode = element.getAttribute(name:"ceg_kod");
            String vasarlasVasarloKod = element.getAttribute(name:"vasarlo_kod");
            String kezdDatum = element.getElementsByTagName(name:"kezdDatum").item(index:0).getTextContent();
```

```
            System.out.println("Dátumkód: " + vasarlasDatumkod);
            System.out.println("Cég kód: " + vasarlasCompanyCode);
            System.out.println("Vásárló Kód: " + vasarlasVasarloKod);
            System.out.println("Kezdő Dátum: " + kezdDatum);
            System.out.println();
        }
    }

    // Querying Szamlazas elements where vegosszeg is greater than 60.000
    NodeList szamlazasList = doc.getElementsByTagName(tagname:"Szamlazas");
    System.out.println(x:"\n----------Számlázások ami nagyobb mint 60.000----------");

    for (int i = 0; i < szamlazasList.getLength(); i++) {
        Node node = szamlazasList.item(i);

        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;

            String szamlaKod = element.getAttribute(name:"szamlakod");
            String szamlazasVasarloCode = element.getAttribute(name:"vasarlo_kod");
            String szamlaDatum = element.getElementsByTagName(name:"szamlaDatum").item(index:0).getTextContent();
            String vegOsszeg = element.getElementsByTagName(name:"vegosszeg").item(index:0).getTextContent();
            String jogsiszam = element.getElementsByTagName(name:"jogsiszam").item(index:0).getTextContent();

            // Ellenőrizd, hogy a végösszeg nagyobb vagy egyenlő mint 60000
            if (Integer.parseInt(vegOsszeg) >= 60000) {
                System.out.println("Számlakód: " + szamlaKod);
                System.out.println("Vásárló Kód: " + szamlazasVasarloCode);
                System.out.println("Számla Dátum: " + szamlaDatum);
                System.out.println("Végösszeg: " + vegOsszeg);
                System.out.println("Jogsiszám: " + jogsiszam);
                System.out.println();
            }
        }
    }
```

```
    // Querying Autós_adatok and Autós_tipus for 1:1 relationship
    NodeList autosAdatokList2 = doc.getElementsByTagName(tagname:"AutosAdatok");
    System.out.println(x:"\n----------Autós adatok és az azokhoz tartozó típusok----------");

    for (int i = 0; i < autosAdatokList2.getLength(); i++) {
        Node node = autosAdatokList2.item(i);

        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element autosAdatokElement = (Element) node;
            String forgalmiKod = autosAdatokElement.getAttribute(name:"forgalmi_kod");
            String adatokCompanyCode = autosAdatokElement.getAttribute(name:"ceg_kod");
            String ar = autosAdatokElement.getElementsByTagName(name:"ar").item(index:0).getTextContent();
            String AutosAdatok = autosAdatokElement.getElementsByTagName(name:"kmOra").item(index:0).getTextContent();
            String status = autosAdatokElement.getElementsByTagName(name:"statusz").item(index:0).getTextContent();

            // Querying AutosTipus for the same forgalmiKod
            NodeList autosTipusList2 = doc.getElementsByTagName(tagname:"AutosTipus");
            for (int j = 0; j < autosTipusList2.getLength(); j++) {
                Node autosTipusNode = autosTipusList2.item(j);

                if (autosTipusNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element autosTipusElement = (Element) autosTipusNode;
                    String tipusForgalmiKod = autosTipusElement.getAttribute(name:"forgalmi_kod");

                    if (forgalmiKod.equals(tipusForgalmiKod)) {
                        String gyev = autosTipusElement.getElementsByTagName(name:"gyartasiEv").item(index:0).getTextContent();
                        String marka = autosTipusElement.getElementsByTagName(name:"marka").item(index:0).getTextContent();
                        String nev = autosTipusElement.getElementsByTagName(name:"nev").item(index:0).getTextContent();

                        System.out.println("Forgalmi Kód: " + forgalmiKod);
                        System.out.println("Cég kód: " + adatokCompanyCode);
                        System.out.println("Ár: " + ar);
                        System.out.println("KM óra: " + AutosAdatok);
                        System.out.println("Státusz: " + status);
                        System.out.println("Gyártási Év: " + gyev);
                        System.out.println("Márka: " + marka);
                        System.out.println("Név: " + nev);
                        System.out.println();
                    }
                }
            }
```

```java
// Querying AutosCeg and AutosAdatok for 1:n relationship
NodeList autosCegList1 = doc.getElementsByTagName(tagname:"AutosCeg");
System.out.println(x:"\n----------Autós cégek és az azokhoz tartozó adatok----------");

for (int i = 0; i < autosCegList1.getLength(); i++) {
    Node node = autosCegList1.item(i);

    if (node.getNodeType() == Node.ELEMENT_NODE) {
        Element autosCegElement = (Element) node;
        String companyCode = autosCegElement.getAttribute(name:"ceg_kod");
        String companyName = autosCegElement.getElementsByTagName(name:"cegnev").item(index:0).getTextContent();

        // Querying AutosAdatok for the same ceg_kod
        NodeList autosAdatokList1 = doc.getElementsByTagName(tagname:"AutosAdatok");
        for (int j = 0; j < autosAdatokList1.getLength(); j++) {
            Node autosAdatokNode = autosAdatokList1.item(j);

            if (autosAdatokNode.getNodeType() == Node.ELEMENT_NODE) {
                Element autosAdatokElement = (Element) autosAdatokNode;
                String adatokCompanyCode = autosAdatokElement.getAttribute(name:"ceg_kod");

                if (companyCode.equals(adatokCompanyCode)) {
                    String ar = autosAdatokElement.getElementsByTagName(name:"ar").item(index:0).getTextContent();
                    String AutosAdatok = autosAdatokElement.getElementsByTagName(name:"kmOra").item(index:0)
                            .getTextContent();
                    String status = autosAdatokElement.getElementsByTagName(name:"statusz").item(index:0).getTextContent();

                    System.out.println("Cég kód: " + companyCode);
                    System.out.println("Cég név: " + companyName);
                    System.out.println("Ár: " + ar);
                    System.out.println("KM óra: " + AutosAdatok);
                    System.out.println("Státusz: " + status);
                    System.out.println();
                }
            }
        }
    }
}
```

```java
// Querying Vasarlo and Vasarlas for n:m relationship
NodeList vasarloList1 = doc.getElementsByTagName(tagname:"Vasarlo");
System.out.println(x:"\n----------Vásárlók és az azokhoz tartozó vásárlások----------");

for (int i = 0; i < vasarloList1.getLength(); i++) {
    Node node = vasarloList1.item(i);

    if (node.getNodeType() == Node.ELEMENT_NODE) {
        Element vasarloElement = (Element) node;
        String vasarloKod = vasarloElement.getAttribute(name:"jogkod");
        String vasarloEmail = vasarloElement.getElementsByTagName(name:"email_cim").item(index:0).getTextContent();
        String vasarloName = vasarloElement.getElementsByTagName(name:"nev").item(index:0).getTextContent();

        // Querying Vasarlas for the same vasarlo_kod
        NodeList vasarlasList1 = doc.getElementsByTagName(tagname:"Vasarlas");
        for (int j = 0; j < vasarlasList1.getLength(); j++) {
            Node vasarlasNode = vasarlasList1.item(j);

            if (vasarlasNode.getNodeType() == Node.ELEMENT_NODE) {
                Element vasarlasElement = (Element) vasarlasNode;
                String vasarlasVasarloKod = vasarlasElement.getAttribute(name:"vasarlo_kod");

                if (vasarloKod.equals(vasarlasVasarloKod)) {
                    String vasarlasDatumkod = vasarlasElement.getAttribute(name:"datumkod");
                    String vasarlasCompanyCode = vasarlasElement.getAttribute(name:"ceg_kod");
                    String kezdDatum = vasarlasElement.getElementsByTagName(name:"kezdDatum").item(index:0)
                            .getTextContent();

                    System.out.println("Vásárló Kód: " + vasarloKod);
                    System.out.println("Név: " + vasarloName);
                    System.out.println("Email: " + vasarloEmail);
                    System.out.println("Vásárlás Dátumkód: " + vasarlasDatumkod);
                    System.out.println("Cég kód: " + vasarlasCompanyCode);
                    System.out.println("Első vásárlás: " + kezdDatum);
                    System.out.println();
                }
            }
        }
    }
}
```

Miskolc,
2023