Jegyzőkönyv

Adatkezelés XML környezetben

Féléves feladat

MVK adatbázis

Készítette: **Sikora Dávid Ádám**

Neptunkód: **IRE699**

Dátum: **2023.12.04**

Tartalomjegyzék

Bevezetés

Az adatkezelés XML környezetben című tantárgyra készült féléves feladatom az MVK Zrt. adatbázisának egy lehetséges megoldását mutatja be. A Miskolc Városi Közlekedési Zrt. felelős Miskolc tömegközlekedésének a lebonyolításáért és menedzseléséért.

A cégnek nyilván kell tartania mind a fennálló humán erőforrást, mind az eszközparkot és az ezek szervezetten történő működtetéséhez szükséges adatokat. A cég mindig igyekszik a város lakóinak eleget tenni a menetrendeket és járatokat tekintve, így rengeteg adatváltozás következik be, szinte havi rendszerességgel. Ezeket mind könnyen módosíthatónak, szerkeszthetőnek kell lennie, más esetben a gyakori változtatások nehézkesek lennének.

Az is szükséges továbbá, hogy az eszközpark elemeinek mozgása, használata lekövethető legyen akár utólag is. Erre a karbantartási munkálatok miatt van főleg szükség, azonban esetleges KRESZ szabálysértések, vagy rongálások, balesetek esetén is hasznos lehet.

Az említett szempontok alapján indultam el a megfelelő adatbázis rendszer kialakításának irányába.

1. Feladat

1a) Az Adatbázis ER modell tervezése

A bevezetésben említett szempontok alapján került megtervezésre a következő ER modell. 5 egyed került megállapításra sofőr, busz, útvonal, megálló és helyszín névvel. Miden egyed egy elsődleges kulccsal rendelkezik, mely egy szám alapú azonosításra szolgál az egyedek kapcsolatakor.

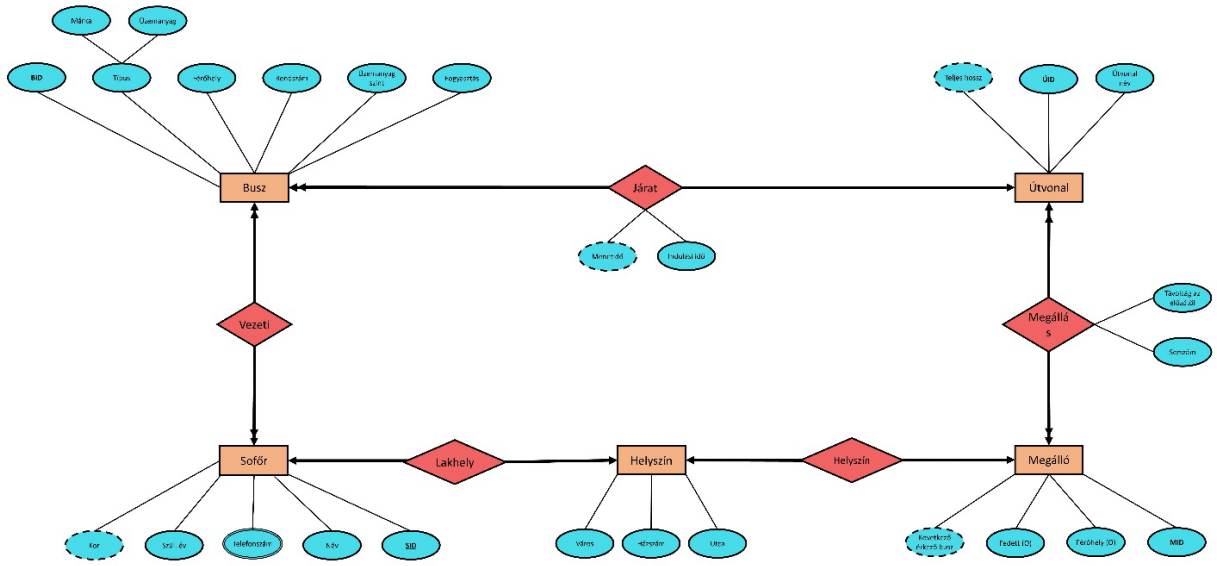
A Sofőr az egyes sofőrök alapadatait tartalmazza, a nevét, a születési évét és a telefonszámait. Ez egy 1:1 kapcsolattal kötődik a helyszín egyeddel, mely az adott sofőr lakhelyét jelenti. A sofőr egyed továbbá egy N:M kapcsolattal kapcsolódik busz egyedekhez. Ez azt jelenti, hogy mely buszt mely sofőr vezet.

A busz egyed az eszközpark egyes buszainak tulajdonságait tartalmazza. Ezek a férőhelyek száma, a rendszám, az üzemanyagszint, a fogyasztás és a busz típusa, mely magában foglalja az üzemanyag típusát és a márkát is. A busz egyedek egy N:1 módon kapcsolódnak az útvonalakhoz. Ennek a kapcsolatnak járat a neve, mivel ez határozza meg hogy mely busz, mely járaton ment, így a kapcsolatnak van egy saját tulajdonsága is, az indulási idő, mely azt jelzi, hogy a busszal mikor indultak el a járaton.

Az útvonal egyed az elsődleges kulcson kívül csak az útvonal nevét tartalmazza, mint tulajdonság, mivel egy útvonal a nevén kívül gyakorlatban a megállók sorrendiségét jelenti. Egy N:M kapcsolattal kapcsolódik ez a megálló egyedekhez. A kapcsolat neve megállás, azaz, hogy az adott útvonal mely megállókat tartalmazza, így a kapcsolat 2 tulajdonságot is tartalmaz, a sorszámát az útvonalon és azt, hogy mennyi a távolság az előtte lévő megállótól.

A megállók legfontosabb jellemzője a helyzete, ezt egy 1:1 kapcsolaton keresztül kötöttem össze a helyszín egyeddel, azonban ezen felül tartalmaz 2 opcionális tagot, a megállóban lévő férőhelyeket és azt, hogy fedett-e vagy sem.

Az ER modellben az átláthatóság okán az egyedeket bézs, a kapcsolatokat piros, a tulajdonságokat pedig kék háttérrel jelöltem.

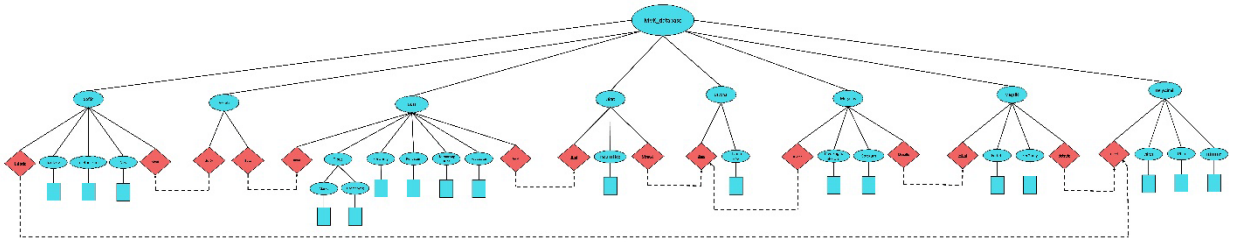


1. ábra ER modell

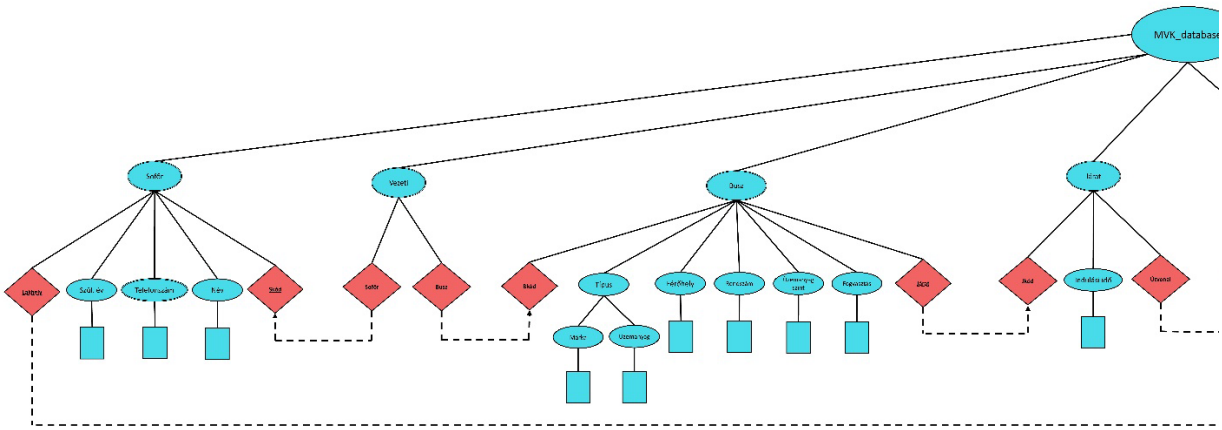
1b) Az adatbázis konvertálása XDM modellre

Az XDM modellre történő konvertálás egészen egyszerű volt ez esetben. A legfontosabb elvégzendő feladat az elsődleges és az idegen kulcsok egyeztetése, attribútummá alakítása volt. Az egyese egyedekből többszörösen megjelenő elemek lettek, ezen egyedek tulajdonságai pedig az említett elemek gyerekelemei lettek. Az egy összetett tulajdonság esetén szintén még két gyerekelem keletkezett.

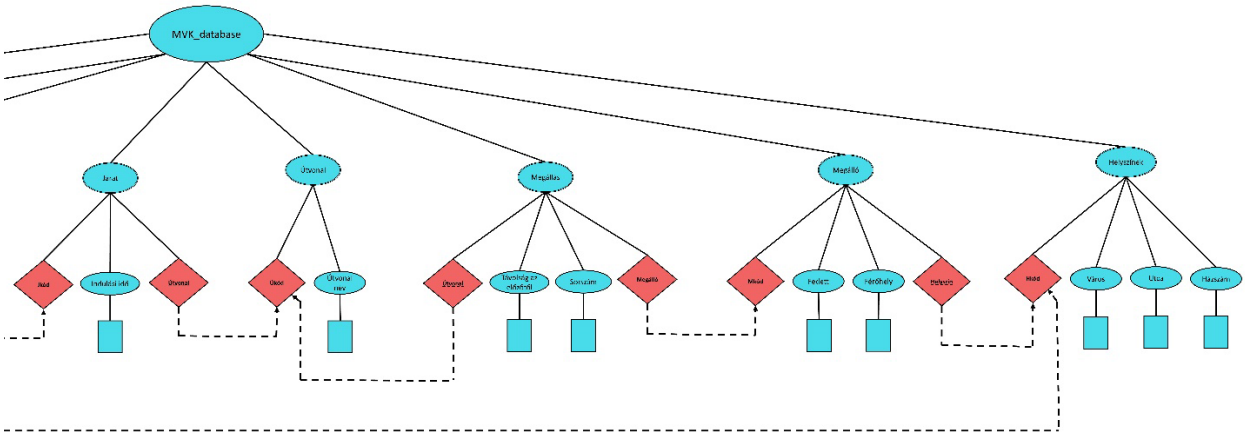
Az XDM modellt az átláthatóság és érthetőség miatt, az ER modellhez hasonlóan színeztem, itt az elemek kékkel lettek színezve, az attribútumok pedig piros színnel.



2. ábra Az XDM modell



3. ábra Az XDM modell első (bal oldali) fele



4. ábra Az XDM modell második (jobb oldali) fele

1c) Az XDM modell alapján XML dokumentum készítése

Az XML dokumentum elkészítése ezek után nagyon egyszerű volt. A megtervezett XDM modell alapján kellett csak implementálni az egyes elemeket. A feladatkiírás szerint minden többször előforduló elemből legalább 3 példányt kellett készíteni. Nem tartottam valóságszerűnek, hogy ugyanazon útvonalon menjenek a járatok, emiatt a helyszínekből, megállásokból, megállókból jóval több lett, így egy valósághű példát láthatunk a dokumentumban.

Az elemeket és az attribútumokat egy az egyben lehet implementálni az XML dokumentumban, ezt tettem meg az 1. programkód esetén.

```
?xml version="1.0" encoding="UTF-8" standalone="no"?><MVK_database
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="XMLSchemaIRE699.xsd">


    <!--Sofőrök-->

    <Sofor lakhely="1" skod="1">
        <szul_ev>1980</szul_ev>
        <tel_szam>+36301234566</tel_szam>
        <tel_szam>+36701234566</tel_szam>
        <nev>Nagy József</nev>
    </Sofor>

    <Sofor lakhely="2" skod="2">
        <szul_ev>1976</szul_ev>
        <tel_szam>+36301235466</tel_szam>
        <tel_szam>+36401232366</tel_szam>
        <nev>Kiss János</nev>
    </Sofor>

    <Sofor lakhely="3" skod="3">
        <szul_ev>1999</szul_ev>
        <tel_szam>+363012764566</tel_szam>
        <tel_szam>+36301267566</tel_szam>
        <nev>Adorján István</nev>
    </Sofor>
```

```
<!--Mely sofőr mely buszt vezeti-->

 <Vezeti busz="2" sofor="1"/>
 <Vezeti busz="3" sofor="2"/>
 <Vezeti busz="1" sofor="3"/>

 <!--Buszok-->

 <Busz bkod="1" jarat="3">
     <tipus>
         <marka>BYD</marka>
         <uzemanyag>Elektromos</uzemanyag>
     </tipus>
     <ferohely>145</ferohely>
     <rendszam>AABC-123</rendszam>
     <uzemanyag_szint>80</uzemanyag_szint>
     <fogyasztas>30</fogyasztas>
 </Busz>

 <Busz bkod="2" jarat="2">
     <tipus>
         <marka>BYD</marka>
         <uzemanyag>Elektromos</uzemanyag>
     </tipus>
     <ferohely>145</ferohely>
     <rendszam>AABC-133</rendszam>
     <uzemanyag_szint>74</uzemanyag_szint>
     <fogyasztas>30</fogyasztas>
 </Busz>

 <Busz bkod="3" jarat="1">
     <tipus>
         <marka>MAN</marka>
         <uzemanyag>Dízel</uzemanyag>
     </tipus>
     <ferohely>110</ferohely>
     <rendszam>AABC-143</rendszam>
     <uzemanyag_szint>20</uzemanyag_szint>
     <fogyasztas>12</fogyasztas>
 </Busz>
```

```
<!--Járatok-->

 <Jarat jkod="1" utvonal="3">
     <indulasi_ido>10:00</indulasi_ido>
 </Jarat>

 <Jarat jkod="2" utvonal="2">
     <indulasi_ido>15:00</indulasi_ido>
 </Jarat>

 <Jarat jkod="3" utvonal="1">
     <indulasi_ido>18:20</indulasi_ido>
 </Jarat>

 <!--Útvonalak-->

 <Utvonal utkod="1">
     <utvonal_nev>1A</utvonal_nev>
 </Utvonal>

 <Utvonal utkod="2">
     <utvonal_nev>2</utvonal_nev>
 </Utvonal>

 <Utvonal utkod="3">
     <utvonal_nev>3A</utvonal_nev>
 </Utvonal>

 <!--Az egyes útvonalak megállói-->
 <!--3A-->

 <Megallas megallo="3" utvonal="3">
     <tav_elo>10</tav_elo>
     <sorszam>1</sorszam>
 </Megallas>

 <Megallas megallo="1" utvonal="3">
     <tav_elo>5</tav_elo>
     <sorszam>2</sorszam>
 </Megallas>

 <Megallas megallo="4" utvonal="3">
     <tav_elo>1</tav_elo>
     <sorszam>3</sorszam>
 </Megallas>
```

```
        <!--2-->

    <Megallas megallo="2" utvonal="2">
        <tav_elo>8</tav_elo>
        <sorszam>1</sorszam>
    </Megallas>

    <Megallas megallo="6" utvonal="2">
        <tav_elo>25</tav_elo>
        <sorszam>2</sorszam>
    </Megallas>

    <Megallas megallo="3" utvonal="2">
        <tav_elo>30</tav_elo>
        <sorszam>3</sorszam>
    </Megallas>

        <!--1A-->

    <Megallas megallo="5" utvonal="1">
        <tav_elo>15</tav_elo>
        <sorszam>1</sorszam>
    </Megallas>

    <Megallas megallo="3" utvonal="1">
        <tav_elo>19</tav_elo>
        <sorszam>2</sorszam>
    </Megallas>

    <Megallas megallo="2" utvonal="1">
        <tav_elo>8</tav_elo>
        <sorszam>3</sorszam>
    </Megallas>

    <!--Megállók-->

    <Megallo helyszin="4" mkod="1">
        <fedett>I</fedett>
        <m_ferohely>10</m_ferohely>
    </Megallo>

    <Megallo helyszin="9" mkod="2">
        <fedett>N</fedett>
        <m_ferohely>50</m_ferohely>
    </Megallo>
```

```
    <Megallo helyszin="6" mkod="3">
        <fedett>N</fedett>
        <m_ferohely>70</m_ferohely>
    </Megallo>

    <Megallo helyszin="5" mkod="4">
        <fedett>I</fedett>
        <m_ferohely>12</m_ferohely>
    </Megallo>

    <Megallo helyszin="7" mkod="5">
        <fedett>I</fedett>
        <m_ferohely>55</m_ferohely>
    </Megallo>

    <Megallo helyszin="8" mkod="6">
        <fedett>N</fedett>
        <m_ferohely>30</m_ferohely>
    </Megallo>

    <!--Helyszínek-->

    <Helyszin hkod="1">
        <varos>Miskolc</varos>
        <utca>Ady Endre</utca>
        <hazszam>10</hazszam>
    </Helyszin>

    <Helyszin hkod="2">
        <varos>Mályi</varos>
        <utca>Fő</utca>
        <hazszam>17</hazszam>
    </Helyszin>

    <Helyszin hkod="3">
        <varos>Miskolc</varos>
        <utca>Széchenyi János</utca>
        <hazszam>34</hazszam>
    </Helyszin>

    <Helyszin hkod="4">
        <varos>Miskolc</varos>
        <utca>Balaton</utca>
        <hazszam>170</hazszam>
    </Helyszin>
```

```
   <Helyszin hkod="5">
        <varos>Miskolc</varos>
        <utca>Balaton</utca>
        <hazszam>1</hazszam>
    </Helyszin>

    <Helyszin hkod="6">
        <varos>Miskolc</varos>
        <utca>Tiszai-PU</utca>
        <hazszam>1</hazszam>
    </Helyszin>

    <Helyszin hkod="7">
        <varos>Miskolc</varos>
        <utca>Egyetem</utca>
        <hazszam>1</hazszam>
    </Helyszin>

    <Helyszin hkod="8">
        <varos>Miskolc</varos>
        <utca>Reptéri</utca>
        <hazszam>11</hazszam>
    </Helyszin>

    <Helyszin hkod="9">
        <varos>Miskolc</varos>
        <utca>Széchenyi János</utca>
        <hazszam>1</hazszam>
    </Helyszin>


</MVK_database>
```

1. programkód Az XML dokumentum

1d) Az XML dokumentum alapján XML Schema készítése

Az XML Schema készítése során igyekeztem minél pontosabban meghatározni minden szabályt, hogy véletlenül se kerülhessen helytelen adat a dokumentumba.

Például az üzemanyag a buszok típusa esetén csak dízel, elektromos vagy hibrid lehet. A rendszám támogatja mind a régi, 3+3 karakteres és az új 4+3 karakteres rendszámokat is. Az üzemanyag szintet százalékban kell megadják, így az csakis a [0, 100] intervallumba eső szám lehet. Az indulási idő csakis óra:perc formátumú lehet, minden esetben 2-2 számmal reprezentálva az értékeket, például 10:01. Végül a fedettséget jelző elem csakis I vagy N értéket vehet fel. Többek között ezek megvalósítása is látható a 2. programkódban.

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

    <!--Egyszerű tipusok-->

    <xs:element name="szul_ev" type="xs:integer" />
    <xs:element name="nev" type="xs:string" />
    <xs:element name="tel_szam" type="xs:string" />
    <xs:element name="marka" type="xs:string" />
    <xs:element name="ferohely" type="xs:integer" />
    <xs:element name="fogyasztas" type="xs:integer"/>
    <xs:element name="utvonal_nev" type="xs:string" />
    <xs:element name="tav_elo" type="xs:integer" />
    <xs:element name="sorszam" type="xs:integer" />
    <xs:element name="m_ferohely" type="xs:integer" />
    <xs:element name="varos" type="xs:string" />
    <xs:element name="utca" type="xs:string" />
    <xs:element name="hazszam" type="xs:integer" />

    <xs:element name="uzemanyag" type="uzemanyagTipus"/>
    <xs:element name="rendszam" type="rendszamTipus"/>
    <xs:element name="uzemanyag_szint" type="uzemanyag_szintTipus"/>
    <xs:element name="indulasi_ido" type="indulasi_idoTipus"/>
    <xs:element name="fedett" type="fedettTipus"/>
    <xs:element name="tipus" type="tipusTipus"/>
```

```
<!--Saját tipusok-->

<xs:simpleType name="uzemanyagTipus">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Elektromos"/>
        <xs:enumeration value="Dízel"/>
        <xs:enumeration value="Hibrid"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="rendszamTipus">
    <xs:restriction base="xs:string">
        <xs:pattern value="[A-Z][A-Z][A-Z][A-Z]-[0-9][0-9][0-9]"/>
        <xs:pattern value="[A-Z][A-Z][A-Z]-[0-9][0-9][0-9]"></xs:pattern>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="uzemanyag_szintTipus">
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="100"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="indulasi_idoTipus">
    <xs:restriction base="xs:string">
        <xs:pattern value="[0-9][0-9]:[0-9][0-9]"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="fedettTipus">
    <xs:restriction base="xs:string">
        <xs:enumeration value="N"/>
        <xs:enumeration value="I"/>
    </xs:restriction>
</xs:simpleType>

<!--Komplex tipusok-->

<xs:complexType name="tipusTipus">
    <xs:sequence>
        <xs:element ref="marka"/>
        <xs:element ref="uzemanyag"/>
    </xs:sequence>
</xs:complexType>
```

```
    <xs:complexType name="soforTipus">
        <xs:sequence>
            <xs:element ref="szul_ev" maxOccurs="1"/>
            <xs:element ref="tel_szam" minOccurs="1" maxOccurs="unbounded"/>
            <xs:element ref="nev" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="skod" type="xs:integer" use="required" />
        <xs:attribute name="lakhely" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="vezetiTipus">
        <xs:attribute name="sofor" type="xs:integer" use="required" />
        <xs:attribute name="busz" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="buszTipus">
        <xs:sequence>
            <xs:element ref="tipus" maxOccurs="1"/>
            <xs:element ref="ferohely" maxOccurs="1"/>
            <xs:element ref="rendszam" maxOccurs="1"/>
            <xs:element ref="uzemanyag_szint" maxOccurs="1"/>
            <xs:element ref="fogyasztas" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="bkod" type="xs:integer" use="required" />
        <xs:attribute name="jarat" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="jaratTipus">
        <xs:sequence>
            <xs:element ref="indulasi_ido"/>
        </xs:sequence>
        <xs:attribute name="jkod" type="xs:integer" use="required" />
        <xs:attribute name="utvonal" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="utvonalTipus">
        <xs:sequence>
            <xs:element ref="utvonal_nev" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="utkod" type="xs:integer" use="required" />
    </xs:complexType>
    <xs:complexType name="megallasTipus">
        <xs:sequence>
            <xs:element ref="tav_elo" maxOccurs="1"/>
            <xs:element ref="sorszam" maxOccurs="1"/>
        </xs:sequence>
```

```
        <xs:attribute name="utvonal" type="xs:integer" use="required" />
        <xs:attribute name="megallo" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="megalloTipus">
        <xs:sequence>
            <xs:element ref="fedett" maxOccurs="1"/>
            <xs:element ref="m_ferohely" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="mkod" type="xs:integer" use="required" />
        <xs:attribute name="helyszin" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="helyszinTipus">
        <xs:sequence>
            <xs:element ref="varos" maxOccurs="1"/>
            <xs:element ref="utca" maxOccurs="1"/>
            <xs:element ref="hazszam" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="hkod" type="xs:integer" use="required" />
    </xs:complexType>

    <!--Gyökérelem elemei-->

    <xs:element name="MVK_database">
        <xs:complexType>
            <xs:sequence>
                    <xs:element name="Sofor" type="soforTipus" minOccurs="1"
maxOccurs="unbounded"/>
                  <xs:element name="Vezeti" type="vezetiTipus" minOccurs="1"
maxOccurs="unbounded"/>
                    <xs:element name="Busz" type="buszTipus" minOccurs="1"
maxOccurs="unbounded"/>
                    <xs:element name="Jarat" type="jaratTipus" minOccurs="1"
maxOccurs="unbounded"/>
                <xs:element name="Utvonal" type="utvonalTipus" minOccurs="1"
maxOccurs="unbounded"/>
               <xs:element name="Megallas" type="megallasTipus" minOccurs="1"
maxOccurs="unbounded"/>
                <xs:element name="Megallo" type="megalloTipus" minOccurs="1"
maxOccurs="unbounded"/>
               <xs:element name="Helyszin" type="helyszinTipus" minOccurs="1"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
```

```
    <!--Elsődleges kulcsok-->

        <xs:key name="Sofor_kod">
            <xs:selector xpath="Sofor"/>
            <xs:field xpath="@skod"/>
        </xs:key>
        <xs:key name="Busz_kod">
            <xs:selector xpath="Busz"/>
            <xs:field xpath="@bkod"/>
        </xs:key>
        <xs:key name="Jarat_kod">
            <xs:selector xpath="Jarat"/>
            <xs:field xpath="@jkod"/>
        </xs:key>
        <xs:key name="Utvonal_kod">
            <xs:selector xpath="Utvonal"/>
            <xs:field xpath="@utkod"/>
        </xs:key>
        <xs:key name="Megallo_kod">
            <xs:selector xpath="Megallo"/>
            <xs:field xpath="@mkod"/>
        </xs:key>
        <xs:key name="Helyszin_kod">
            <xs:selector xpath="Helyszin"/>
            <xs:field xpath="@hkod"/>
        </xs:key>

        <!--Idegen kulcsok-->

        <xs:keyref name="Lakhely_kulcs" refer="Helyszin_kod">
            <xs:selector xpath="Sofor"/>
            <xs:field xpath="@lakhely"/>
        </xs:keyref>
        <xs:keyref name="Vezeto_kulcs" refer="Sofor_kod">
            <xs:selector xpath="Vezeti"/>
            <xs:field xpath="@sofor"/>
        </xs:keyref>
        <xs:keyref name="Busz_kulcs" refer="Busz_kod">
            <xs:selector xpath="Vezeti"/>
            <xs:field xpath="@busz"/>
        </xs:keyref>
        <xs:keyref name="Jarat_kulcs" refer="Jarat_kod">
            <xs:selector xpath="Busz"/>
            <xs:field xpath="@jarat"/>
        </xs:keyref>
```

```
        <xs:keyref name="Utvonal_jarat_kulcs" refer="Utvonal_kod">
            <xs:selector xpath="Jarat"/>
            <xs:field xpath="@utvonal"/>
         </xs:keyref>
         <xs:keyref name="Utvonal_megallas_kulcs" refer="Utvonal_kod">
            <xs:selector xpath="Megallas"/>
            <xs:field xpath="@utvonal"/>
         </xs:keyref>
         <xs:keyref name="Megallo_kulcs" refer="Megallo_kod">
            <xs:selector xpath="Megallas"/>
            <xs:field xpath="@megallo"/>
         </xs:keyref>
         <xs:keyref name="Helyszin_kulcs" refer="Helyszin_kod">
            <xs:selector xpath="Megallo"/>
            <xs:field xpath="@helyszin"/>
         </xs:keyref>

         <!-- Az 1:1 kapcsolatok megvalósitása-->

         <xs:unique name="Megallo_Helyszin_egyegy">
            <xs:selector xpath="Megallo"/>
            <xs:field xpath="@helyszin"/>
         </xs:unique>
         <xs:unique name="Sofor_Helyszin_egyegy">
            <xs:selector xpath="Sofor"/>
            <xs:field xpath="@lakhely"/>
         </xs:unique>

    </xs:element>
</xs:schema>
```

2. programkód Az XML Schema dokumentum

2. Feladat

2a) Adatolvasás

A DOMReadIRE699 osztály képes beolvasni az XMLIRE699.xml dokumentum tartalmát és azt a konzolra strukturált formában kiírni. A DOMReadIRE699.java file-on belül létre hoztam egy másik osztályt is, melyet több másik osztály is használ a terminálra való kiíratáshoz. Ez a DomRead osztály. ez az osztály képes egy Document típusú file kiírására, mely formailag megfelel a feladatban megadott XML-nek. A file tartalma látható a 3. programkódban.

```
package hu.domparse.ire699;

import org.w3c.dom.*;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.File;
import java.util.ArrayList;

public class DOMReadIRE699 {

    public static void main(String[] args) {

        try {
            File xml = new File("XMLIRE699.xml");
            DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = dbf.newDocumentBuilder();
            Document doc = builder.parse(xml);
            doc.getDocumentElement().normalize();

            DomRead reader = new DomRead();
            reader.printDocument(doc);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
class DomRead {

    private int indent = 0;

    public void printDocument(Document doc){

        //Getting the root element from the document
        Element root = doc.getDocumentElement();

        //Get the nodeLists from the root element
        NodeList soforok = root.getElementsByTagName("Sofor");
        NodeList vezeti = root.getElementsByTagName("Vezeti");
        NodeList buszok = root.getElementsByTagName("Busz");
        NodeList jaratok = root.getElementsByTagName("Jarat");
        NodeList utvonalak = root.getElementsByTagName("Utvonal");
        NodeList megallasok = root.getElementsByTagName("Megallas");
        NodeList megallok = root.getElementsByTagName("Megallo");
        NodeList helyszinek = root.getElementsByTagName("Helyszin");

        //Print root node Start
        System.out.println("\n<" + root.getNodeName() + ">" );
        indent++;

        //Print Sofor nodes
        System.out.println();
        for (int i = 0; i < soforok.getLength(); i++){

            Element element = (Element)soforok.item(i);

            Element szul_ev =
(Element)element.getElementsByTagName("szul_ev").item(0);
            ArrayList<Element> tel_szamok = new ArrayList<>();
            for (int x = 0; x <
element.getElementsByTagName("tel_szam").getLength() ;x++ ) {
                tel_szamok.add( (Element)
element.getElementsByTagName("tel_szam").item(x) );
            }
            Element nev =
(Element)element.getElementsByTagName("nev").item(0);

            printElementStartWithAttributes(element);
            indent++;
            printWithIndent("<szul_ev>" + szul_ev.getTextContent() +
"</szul_ev>");
```

```
            for (Element tel_szam:tel_szamok) {
                printWithIndent("<tel_szam>" +
tel_szam.getTextContent() + "</tel_szam>");
            }
            printWithIndent("<nev>" + nev.getTextContent() +
"</nev>");

            indent--;
            printElementEndWithAttributes(element);


        }

        //Print vezeti nodes
        System.out.println();
        for (int i = 0; i < vezeti.getLength(); i++){

            printElementOneLinerWithAttributes((Element)vezeti.item(i)
);
        }

        //Print Busz nodes
        System.out.println();
        for (int i = 0; i < buszok.getLength(); i++){

            Element element = (Element)buszok.item(i);

            Element tipus =
(Element)element.getElementsByTagName("tipus").item(0);
                Element marka =
(Element)tipus.getElementsByTagName("marka").item(0);
                Element uzemanyag =
(Element)tipus.getElementsByTagName("uzemanyag").item(0);
            Element ferohely =
(Element)element.getElementsByTagName("ferohely").item(0);
            Element rendszam =
(Element)element.getElementsByTagName("rendszam").item(0);
            Element uzemanyag_szint =
(Element)element.getElementsByTagName("uzemanyag_szint").item(0);
            Element fogyasztas =
(Element)element.getElementsByTagName("fogyasztas").item(0);

            printElementStartWithAttributes(element);
            indent++;
```

```
            printWithIndent("<tipus>");
            indent++;
            printWithIndent("<marka>" + marka.getTextContent() +
"</marka>");
            printWithIndent("<uzemanyag>" + uzemanyag.getTextContent()
+ "</uzemanyag>");
            indent--;
            printWithIndent("</tipus>");

            printWithIndent("<ferohely>" + ferohely.getTextContent() +
"</ferohely>");
            printWithIndent("<rendszam>" + rendszam.getTextContent() +
"</rendszam>");
            printWithIndent("<uzemanyag_szint>" +
uzemanyag_szint.getTextContent() + "</uzemanyag_szint>");
            printWithIndent("<fogyasztas>" +
fogyasztas.getTextContent() + "</fogyasztas>");

            indent--;
            printElementEndWithAttributes(element);
        }

        //Print Jarat nodes
        System.out.println();
        for (int i = 0; i < jaratok.getLength(); i++){

            Element element = (Element) jaratok.item(i);

            Element indulasi_ido =
(Element)element.getElementsByTagName("indulasi_ido").item(0);

            printElementStartWithAttributes(element);
            indent++;

            printWithIndent("<indulasi_ido>" +
indulasi_ido.getTextContent() + "</indulasi_ido>");

            indent--;
            printElementEndWithAttributes(element);
        }
```

```
        //Print Utvonal nodes
        System.out.println();
        for (int i = 0; i < utvonalak.getLength(); i++){

            Element element = (Element) utvonalak.item(i);

            Element utvonal_nev =
(Element)element.getElementsByTagName("utvonal_nev").item(0);

            printElementStartWithAttributes(element);
            indent++;

            printWithIndent("<utvonal_nev>" +
utvonal_nev.getTextContent() + "</utvonal_nev>");

            indent--;
            printElementEndWithAttributes(element);
        }

        //Print Megallas nodes
        System.out.println();
        for (int i = 0; i < megallasok.getLength(); i++){

            Element element = (Element) megallasok.item(i);

            Element tav_elo =
(Element)element.getElementsByTagName("tav_elo").item(0);
            Element sorszam =
(Element)element.getElementsByTagName("sorszam").item(0);

            printElementStartWithAttributes(element);
            indent++;

            printWithIndent("<tav_elo>" + tav_elo.getTextContent() +
"</tav_elo>");
            printWithIndent("<sorszam>" + sorszam.getTextContent() +
"</sorszam>");

            indent--;
            printElementEndWithAttributes(element);
        }
```

```
        //Print Megallo nodes
         System.out.println();
         for (int i = 0; i < megallok.getLength(); i++){

             Element element = (Element) megallok.item(i);

             Element fedett =
(Element)element.getElementsByTagName("fedett").item(0);
             Element m_ferohely =
(Element)element.getElementsByTagName("m_ferohely").item(0);

             printElementStartWithAttributes(element);
             indent++;

             printWithIndent("<fedett>" + fedett.getTextContent() +
"</fedett>");
             printWithIndent("<m_ferohely>" +
m_ferohely.getTextContent() + "</m_ferohely>");

             indent--;
             printElementEndWithAttributes(element);
         }

         //Print Helzsyinek nodes
         System.out.println();
         for (int i = 0; i < helyszinek.getLength(); i++){

             Element element = (Element) helyszinek.item(i);

             Element varos =
(Element)element.getElementsByTagName("varos").item(0);
             Element utca =
(Element)element.getElementsByTagName("utca").item(0);
             Element hazszam =
(Element)element.getElementsByTagName("hazszam").item(0);

             printElementStartWithAttributes(element);
             indent++;

             printWithIndent("<varos>" + varos.getTextContent() +
"</varos>");
             printWithIndent("<utca>" + utca.getTextContent() +
"</utca>");
             printWithIndent("<hazszam>" + hazszam.getTextContent() +
"</hazszam>");
```

```
            indent--;
            printElementEndWithAttributes(element);
        }

        //Print root node End
        System.out.println("\n<" + root.getNodeName() + ">" );

    }

    private void printElementStartWithAttributes(Element element){
        printWithIndent("<" + element.getNodeName() +
formatAttributes(element.getAttributes()));
    }

    private void printElementEndWithAttributes(Element element){
        printWithIndent("<" + element.getNodeName() + "/>");
    }

    private void printElementOneLinerWithAttributes(Element element){
        printWithIndent("<" + element.getNodeName() +
formatAttributes(element.getAttributes()) + "/>");
    }

    //Formatting the Attributes
    private String formatAttributes(NamedNodeMap attributes){
        int attrLength = attributes.getLength();
        if (attrLength == 0){
            return ">";
        }
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < attrLength; i++){
            sb.append(" ");
            sb.append(attributes.item(i).getNodeName()).append("=").ap
pend(attributes.item(i).getNodeValue());
        }
        sb.append(">");
        return sb.toString();
    }

    //Print text with indentation
    private void printWithIndent(String in){
        indent();
        System.out.println(in);
    }
```

```
    //Printing the indentation
    private void indent(){
        for (int i = 0; i < indent; i++){
            System.out.print("    ");
        }
    }
}
```

3. programkód A DOMReadIRE699.java file tartalma

2b) Adatmódosítás

A DOMModifyIRE699 osztály kiírja a terminálra a korábbi DomRead osztály használatával a dokumentum tartalmát, majd módosítja az adatokat a DomModifyDoc metódust használva, ezután pedig a korábbi módon szintén kiírja a már módosított dokumentumot.

```
package hu.domparse.ire699;

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

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.File;


public class DOMModifyIRE699 {

    public static void main(String[] args) {
        try {
            File xml = new File("XMLIRE699.xml");
            DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = dbf.newDocumentBuilder();
            Document doc = builder.parse(xml);
            doc.getDocumentElement().normalize();

            DomRead reader = new DomRead();

            System.out.println("\n-----------------------------------
---------------Original document:-----------------------------------
--------------");
            reader.printDocument(doc);

            System.out.println("\n-----------------------------------
-----------------Modifications:---------------------------------------
--------------");
            DomModifyDoc(doc);
```

```
            System.out.println("--------------------------------------
-------------Modified document:-----------------------------------------
------------");
            reader.printDocument(doc);

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


    }

    public static void DomModifyDoc(Document doc) {

        Element root = doc.getDocumentElement();

        //Get the nodeLists from the root element
        NodeList soforok = root.getElementsByTagName("Sofor");
        NodeList buszok = root.getElementsByTagName("Busz");
        NodeList jaratok = root.getElementsByTagName("Jarat");
        NodeList megallok = root.getElementsByTagName("Megallo");
        NodeList helyszinek = root.getElementsByTagName("Helyszin");

        for (int i = 0; i < soforok.getLength(); i++){

            Element element = (Element)soforok.item(i);

            Element nev =
(Element)element.getElementsByTagName("nev").item(0);

            if(nev.getTextContent().equals("Nagy József")){
                nev.setTextContent("Nagy Sándor");
                break;
            }
        }
        System.out.println("Modified: Nagy József --> Nagy Sándor");

        for (int i = 0; i < buszok.getLength(); i++){

            Element element = (Element)buszok.item(i);

            if(element.getAttribute("bkod").equals("1")){
```

```
                Element uzemanyag_szint =
(Element)element.getElementsByTagName("uzemanyag_szint").item(0);
                uzemanyag_szint.setTextContent(String.valueOf(Integer.
parseInt(uzemanyag_szint.getTextContent())*0.9).split("\\.")[0]);
                break;
            }
        }
        System.out.println("Modified: Busz (id=1) reduced fuel by
10%");

        for (int i = 0; i < jaratok.getLength(); i++){

            Element element = (Element)jaratok.item(i);

            if(element.getAttribute("jkod").equals("1")){

                Element indulasi_ido =
(Element)element.getElementsByTagName("indulasi_ido").item(0);
                indulasi_ido.setTextContent("10:25");
                break;
            }
        }
        System.out.println("Modified: Járat (id=1) delayed to 10:25");

        for (int i = 0; i < megallok.getLength(); i++){

            Element element = (Element)megallok.item(i);

            if(element.getAttribute("mkod").equals("6")){

                Element fedett =
(Element)element.getElementsByTagName("fedett").item(0);
                fedett.setTextContent("I");
                break;
            }
        }
        System.out.println("Modified: In megallo (id=6) built roof -->
fedett=I ");

        System.out.println("Modified: Járat (id=1) delayed to 10:25");

        for (int i = 0; i < helyszinek.getLength(); i++){

            Element element = (Element)helyszinek.item(i);
```

```
            if(element.getAttribute("hkod").equals("3")){

                Element hazszam =
(Element)element.getElementsByTagName("hazszam").item(0);
                hazszam.setTextContent("35");
                break;
            }
        }
        System.out.println("Modified: Helyszin (id=3)
misadministrated: address number 34 --> 35 ");
    }
}
```

4. programkód A DOMModifyIRE699.java file tartalma

2a) Adatlekérdezés

Az adatlekérdezés megvalósítása látható a DOMQueryIRE699.java file-ban. A dokumentum beolvasása után a printQuery() metódus használatával írhatjuk ki az előre megadott lekérdezéseket a dokumentumból. Ezt a metódust a DomQuery osztály részeként került implementálásra az újrafelhasználhatóság miatt. A printQuery egyszerre valósítja meg a lekérdezéseket és a kiírást is a terminálra. Ezek a funkciók implementálása az 5. programkódban látható.

```
package hu.domparse.ire699;

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

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.File;
import java.util.ArrayList;

public class DOMQueryIRE699 {

    public static void main(String[] args) {

        try {
            File xml = new File("XMLIRE699.xml");
            DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = dbf.newDocumentBuilder();
            Document doc = builder.parse(xml);
            doc.getDocumentElement().normalize();

            DomQuery query = new DomQuery();
            query.printQuery(doc);
        } catch (Exception e) {
            e.printStackTrace();
        }


    }
}
```

```
class DomQuery {

    private int indent = 0;

    public void printQuery(Document doc) {

        //Getting the root element from the document
        Element root = doc.getDocumentElement();

        //Get the nodeLists from the root element
        NodeList soforok = root.getElementsByTagName("Sofor");
        NodeList buszok = root.getElementsByTagName("Busz");
        NodeList jaratok = root.getElementsByTagName("Jarat");
        NodeList utvonalak = root.getElementsByTagName("Utvonal");
        NodeList megallasok = root.getElementsByTagName("Megallas");
        NodeList megallok = root.getElementsByTagName("Megallo");
        NodeList helyszinek = root.getElementsByTagName("Helyszin");


        //Print all driver data
        System.out.println("\nPrint all driver data:");
        System.out.println();
        for (int i = 0; i < soforok.getLength(); i++) {

            Element element = (Element) soforok.item(i);

            Element szul_ev = (Element)
element.getElementsByTagName("szul_ev").item(0);
            ArrayList<Element> tel_szamok = new ArrayList<>();
            for (int x = 0; x <
element.getElementsByTagName("tel_szam").getLength(); x++) {
                tel_szamok.add((Element)
element.getElementsByTagName("tel_szam").item(x));
            }
            Element nev = (Element)
element.getElementsByTagName("nev").item(0);

            printElementStartWithAttributes(element);
            indent++;
```

```
            printWithIndent("<szul_ev>" + szul_ev.getTextContent() +
"</szul_ev>");
            for (Element tel_szam : tel_szamok) {
                printWithIndent("<tel_szam>" +
tel_szam.getTextContent() + "</tel_szam>");
            }
            printWithIndent("<nev>" + nev.getTextContent() +
"</nev>");

            indent--;
            printElementEndWithAttributes(element);


        }

        //Print BYD buses
        System.out.println("\nPrint all BYD buses:");
        System.out.println();
        for (int i = 0; i < buszok.getLength(); i++) {

            Element element = (Element) buszok.item(i);

            Element tipus = (Element)
element.getElementsByTagName("tipus").item(0);
            Element marka = (Element)
tipus.getElementsByTagName("marka").item(0);
            Element uzemanyag = (Element)
tipus.getElementsByTagName("uzemanyag").item(0);
            Element ferohely = (Element)
element.getElementsByTagName("ferohely").item(0);
            Element rendszam = (Element)
element.getElementsByTagName("rendszam").item(0);
            Element uzemanyag_szint = (Element)
element.getElementsByTagName("uzemanyag_szint").item(0);
            Element fogyasztas = (Element)
element.getElementsByTagName("fogyasztas").item(0);

            if (marka.getTextContent().equals("BYD")) {

                printElementStartWithAttributes(element);
                indent++;
```

```
                printWithIndent("<tipus>");
                indent++;
                printWithIndent("<marka>" + marka.getTextContent() +
"</marka>");
                printWithIndent("<uzemanyag>" +
uzemanyag.getTextContent() + "</uzemanyag>");
                indent--;
                printWithIndent("</tipus>");

                printWithIndent("<ferohely>" +
ferohely.getTextContent() + "</ferohely>");
                printWithIndent("<rendszam>" +
rendszam.getTextContent() + "</rendszam>");
                printWithIndent("<uzemanyag_szint>" +
uzemanyag_szint.getTextContent() + "</uzemanyag_szint>");
                printWithIndent("<fogyasztas>" +
fogyasztas.getTextContent() + "</fogyasztas>");

                indent--;
                printElementEndWithAttributes(element);
            }
        }

        //Print lines that starts after 12:00
        System.out.println("\nPrint all lines that starts after
12:00");
        System.out.println();
        for (int i = 0; i < jaratok.getLength(); i++) {

            Element element = (Element) jaratok.item(i);

            Element indulasi_ido = (Element)
element.getElementsByTagName("indulasi_ido").item(0);

            if
(Integer.parseInt(indulasi_ido.getTextContent().split(":")[0]) >= 12)
{

                printElementStartWithAttributes(element);
                indent++;

                printWithIndent("<indulasi_ido>" +
indulasi_ido.getTextContent() + "</indulasi_ido>");
```

```
                indent--;
                printElementEndWithAttributes(element);
            }
        }

        //Print Stops that have roof
        System.out.println("\nPrint stops that have roof");
        System.out.println();
        for (int i = 0; i < megallok.getLength(); i++) {

            Element element = (Element) megallok.item(i);

            Element fedett = (Element)
element.getElementsByTagName("fedett").item(0);

            if (fedett.getTextContent().equals("I")) {

                Element m_ferohely = (Element)
element.getElementsByTagName("m_ferohely").item(0);

                printElementStartWithAttributes(element);
                indent++;

                printWithIndent("<fedett>" + fedett.getTextContent() +
"</fedett>");
                printWithIndent("<m_ferohely>" +
m_ferohely.getTextContent() + "</m_ferohely>");

                indent--;
                printElementEndWithAttributes(element);
            }
        }

        //Print places that are in Miskolc
        System.out.println("\nPrint all places in Miskolc:");
        System.out.println();
        for (int i = 0; i < helyszinek.getLength(); i++) {

            Element element = (Element) helyszinek.item(i);

            Element varos = (Element)
element.getElementsByTagName("varos").item(0);
```

```
            if (varos.getTextContent().equals("Miskolc")) {

                Element utca = (Element)
element.getElementsByTagName("utca").item(0);
                Element hazszam = (Element)
element.getElementsByTagName("hazszam").item(0);

                printElementStartWithAttributes(element);
                indent++;

                printWithIndent("<varos>" + varos.getTextContent() +
"</varos>");
                printWithIndent("<utca>" + utca.getTextContent() +
"</utca>");
                printWithIndent("<hazszam>" + hazszam.getTextContent()
+ "</hazszam>");

                indent--;
                printElementEndWithAttributes(element);
            }
        }

        //Complex: print 1A line's second stop
        System.out.println("\nComplex Query: Print 1A line's second
stop");
        System.out.println();
        String utvonalID = "";
        String megalloID = "";
        String helyID = "";

        //get utvonalID where name is 1A
        for (int i = 0; i < utvonalak.getLength(); i++) {

            Element element = (Element) utvonalak.item(i);
            Element utvonal_nev = (Element)
element.getElementsByTagName("utvonal_nev").item(0);

            if (utvonal_nev.getTextContent().equals("1A")){
                utvonalID = element.getAttribute("utkod");
                break;
            }
        }
```

```
        //get megalloID where utvonalID matches and is second on the
line
        for (int i = 0; i < megallasok.getLength(); i++) {

            Element element = (Element) megallasok.item(i);

            if (element.getAttribute("utvonal").equals(utvonalID)) {

                Element sorszam = (Element)
element.getElementsByTagName("sorszam").item(0);

                if (sorszam.getTextContent().equals("2")){
                    megalloID = element.getAttribute("megallo");
                    break;
                }
            }
        }

        //get helyID from megallo that's ID matches with megalloID
        for (int i = 0; i < megallok.getLength(); i++) {

            Element element = (Element) megallok.item(i);

            if (element.getAttribute("mkod").equals(megalloID)) {

                helyID = element.getAttribute("helyszin");
                break;
            }
        }

        //print place data where ID is helyID
        for (int i = 0; i < helyszinek.getLength(); i++) {

            Element element = (Element) helyszinek.item(i);

            if (element.getAttribute("hkod").equals(helyID)){

                Element varos = (Element)
element.getElementsByTagName("varos").item(0);

                Element utca = (Element)
element.getElementsByTagName("utca").item(0);
                Element hazszam = (Element)
element.getElementsByTagName("hazszam").item(0);
```

```
                indent--;
                printElementEndWithAttributes(element);
            }
        }
    }

    private void printElementStartWithAttributes(Element element) {
        printWithIndent("<" + element.getNodeName() +
formatAttributes(element.getAttributes()));
    }

    private void printElementEndWithAttributes(Element element) {
        printWithIndent("<" + element.getNodeName() + "/>");
    }

    //Formatting the Attributes
    private String formatAttributes(NamedNodeMap attributes) {
        int attrLength = attributes.getLength();
        if (attrLength == 0) {
            return ">";
        }
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < attrLength; i++) {
            sb.append(" ");
            sb.append(attributes.item(i).getNodeName()).append("=").ap
pend(attributes.item(i).getNodeValue());
        }
        sb.append(">");
        return sb.toString();
    }

    //Print text with indentation
    private void printWithIndent(String in) {
        indent();
        System.out.println(in);
    }

    //Printing the indentation
    private void indent() {
        for (int i = 0; i < indent; i++) {
            System.out.print("    ");
        }}}
```

5. programkód A DOMQeryIRE699.java file tartalma

2a) Adatírás

A DOMWriteIRE699 osztályban először a terminálra kerül kiírásra a dokumentum tartalma a DomRead osztály használatával, majd a file-ban implementált DomWrite osztály writeToFile() metódusa képes a strukturált dokumentumot transformer használatával kiírni a bemenetként megadott file-ba, mely sikerességéről a terminálon visszajelzést is közöl. Ez látható a 6. programkódban.

```
package hu.domparse.ire699;

import org.w3c.dom.*;

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 java.io.File;
import java.io.FileOutputStream;

public class DOMWriteIRE699 {

    public static void main(String[] args) {
        try {
            File xml = new File("XMLIRE699.xml");
            DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = dbf.newDocumentBuilder();
            Document doc = builder.parse(xml);
            doc.getDocumentElement().normalize();

            DomRead reader = new DomRead();
            reader.printDocument(doc);

            DomWrite writer = new DomWrite();
            writer.writeToFile(doc, "XMLIRE6991.xml");


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

```
class DomWrite {

    public void writeToFile(Document doc, String file) {

        try {
            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transformer =
transformerFactory.newTransformer();
            DOMSource source = new DOMSource(doc);
            FileOutputStream output = new FileOutputStream(file);
            StreamResult result = new StreamResult(output);

            transformer.transform(source, result);

            System.out.println("\nFile written!");

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

}
```

6. programkód A DOMWriteIRE699.java file tartalma