JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Állatmenhely

Készítette: **Szemán Balázs**
Neptunkód: **BB89VX**
Dátum: **2024.12.09.**

Tartalomjegyzék

1. fejezet

Bevezetés

Feladatom az állatmenhelyek működésének bemutatása volt, különös tekintettel az érintett szereplők – városok, menhelyek, dolgozók, állatok, betegségek és örökbefogadók – közötti kapcsolatokra és azok részleteire. A téma célja egy átlátható rendszer megalkotása volt, amely megmutatja, hogyan működik egy menhely napi szinten és milyen adatokat szükséges nyilvántartani.

Az állatmenhelyek egy-egy városhoz tartoznak, és minden városban legfeljebb egy menhely található. A menhelyek adatai között megtalálható a nevük, befogadóképességük, pontos címük és egyéb alapvető információk. Ezek a menhelyek fogadják be az állatokat, amelyek mindegyike kizárólag egy adott menhelyhez tartozik. Az állatok esetében fontos adatok például az azonosítójuk, nevük, fajtájuk és életkoruk, továbbá rögzítettem a betegségeiket is, beleértve azok nevét, gyógyítási módját, tüneteiket és a gyógyulási időt.

A dolgozók az állatmenhelyek kulcsszereplői, akik gondoskodnak az állatok ellátásáról. Egy dolgozó csak egy menhelyen dolgozhat, és nyilvántartott adataik között szerepel a nevük, beosztásuk, nemük, születési dátumuk, elérhetőségük és lakcímük. Az örökbefogadók szintén fontos szereplők, hiszen rajtuk keresztül találnak otthonra az állatok. Az örökbefogadók adatai között megtalálhatók a nevük, születési dátumuk, elérhetőségük és lakcímük, ami segít követni, hogy ki fogadott örökbe egy-egy állatot.

2. fejezet

Első feladat

2.1. Az adatbázis ER modell tervezése

Az ER modell leírja az állatmenhely működésében részt vevő szereplők közötti kapcsolatokat, bemutatva, hogyan viszonyulnak egymáshoz a különböző elemek. Ezeknek a szereplőknek különféle adataik vannak, amelyek fel vannak tüntetve, valamint megjelennek a köztük lévő kapcsolati tulajdonságok is.

A Város az a földrajzi egység, amelyhez az állatmenhelyek tartoznak. Minden város tartalmazhat egy menhelyet, amely a helyi állatok védelméért és gondozásáért felel.

A Menhely a központi egység, amely befogadja és ellátja a segítségre szoruló állatokat. A menhelyek pontos adatai, például a nevük, címeik, férőhelyük és elérhetőségük is nyilvántartásra kerülnek.

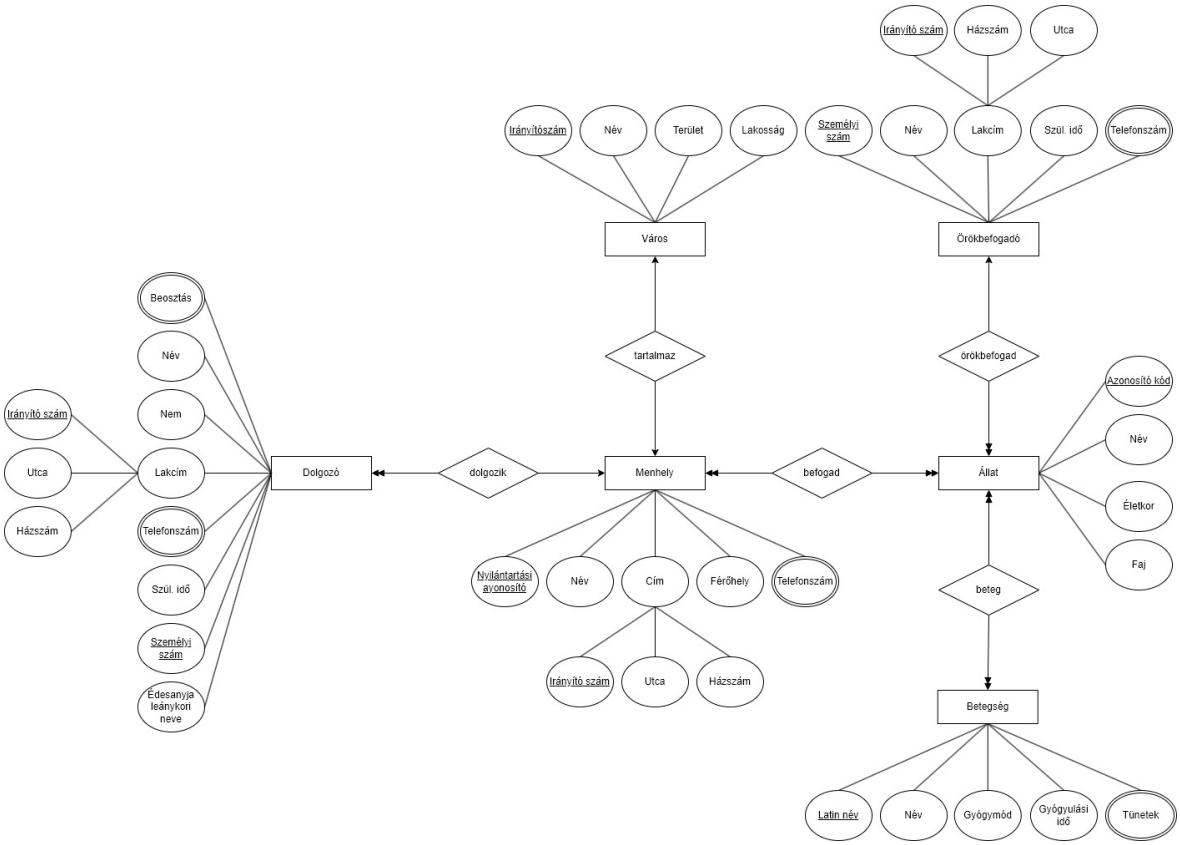
A Dolgozók azok az emberek, akik a menhelyekben tevékenykednek. Feladataik közé tartozik az állatok ellátása, az adminisztráció kezelése és a menhely működésének biztosítása. A dolgozók személyes adatai, mint például a nevük, beosztásuk, lakcímük és egyéb azonosító adataik is rögzítve vannak.

Az Állatok a menhely lakói, akik gondozásra vagy örökbefogadásra várnak. Minden állathoz tartoznak olyan adatok, mint az azonosító kód, név, életkor és faj. Az állatok betegségeinek nyilvántartása is fontos része a

rendszernek.

A Betegség az állatok egészségi állapotát írja le, ahol olyan információk kerülnek rögzítésre, mint a betegség neve, latin neve, a szükséges gyógykezelés, valamint a gyógyulási idő.

Az Örökbefogadók azok az emberek, akik a menhelyről állatokat fogadnak örökbe. Ők rendelkeznek saját adatokkal, mint például a nevük, lakcímük és elérhetőségük.

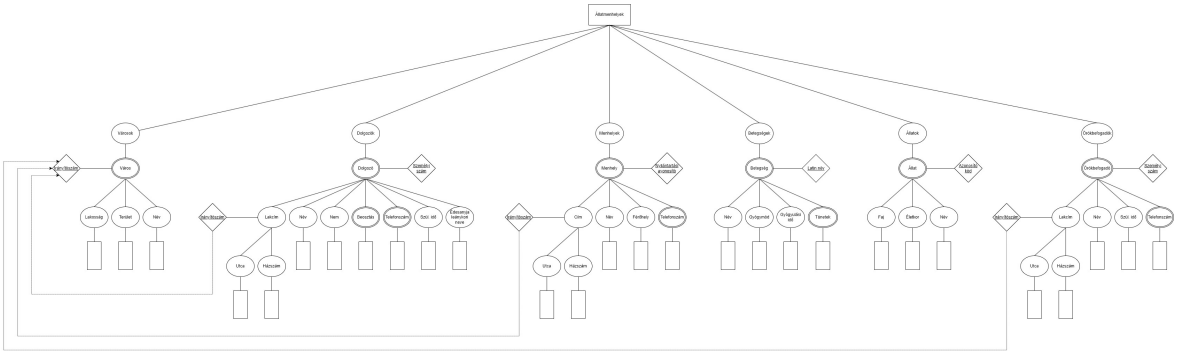


2.1. ábra. Állatmenhely ER modellje

2.2. Az adatbázis konvertálása XDM modellre

Az XDM modellnél a legnagyobb hangsúlyt a fa struktúra adja, erre kell lényegében átkonvertálnunk az eredeti relációs modellt. Az átalakítás során a különböző elemek az állatmenhely működéséhez kapcsolódó entitásokat képviselik, mint például a Városok, Dolgozók, Menhelyek, Állatok, Betegségek és Örökbefogadók. Ezek az elemek különböző tulajdonságokkal lettek ellátva, amelyek a gyerekelemeik formájában jelennek meg. XDM kialakítása során külön figyelmet kellett fordítani az attribútumok és a közöttük fennálló kapcsolatok helyes kialakítására.

Ahogy az ER modellnél itt is érdemes figyelni a szépen kidolgozott küllemre és az átláthatóságra.



2.2. ábra. Állatmenhely XDM modellje

2.3. Az XDM modell alapján XML dokumentum készítése

Ha az XDM modellt megfelelően elkészítettük, az XML dokumentum létrehozása viszonylag egyszerűvé válik, mivel az XDM modell már tartalmaz egy 1:1 séma-szerkezetet az XML kialakításához. A fa struktúrának megfelelően az XML dokumentumban kialakítjuk az elemeket, hozzájuk adjuk a gyerekelemeket és az attribútumokat.

A különböző entitások megkülönböztetésére és egyértelmű azonosítására egyedi azonosítókat (például személyi_szám, azonosito_kod vagy nyilv-

antartasi_azonosito) használunk. Az adatok valós példák alapján kerültek bele az XML dokumentumba, ezért előfordulhat, hogy egyes attribútumok, például telefonszámok vagy címek formátuma különböző lehet.

A városok szekcióban az egyes városokat az irányítószám attribútummal különböztetjük meg, és minden város gyerekelemeként jelenítjük meg a nevüket, területüket és lakosságukat. A dolgozók részben a dolgozókat egyedi személyi_szám attribútummal azonosítjuk, és rögzítjük a beosztásukat, nevüket, nemüket, valamint a lakcímüket, amely külön elemekre, például utca és házszám, van bontva.

Az állatok elemnél az állatok azonosito_kod attribútum alapján kerülnek megkülönböztetésre. A gyerekelemek tartalmazzák az állatok nevét, életkorát és faját. A menhelyek esetében a nyilvantartasi_azonosito szolgál az azonosításra, és a gyerekelemek a menhely nevét, férőhelyeit, telefonszámát és címét tartalmazzák.

A betegségek szekcióban a betegségek latin nevét latin_nev attribútumként tüntetjük fel, a gyerekelemek pedig leírják a betegség magyar nevét, a gyógymódot, a gyógyulási időt és a tüneteket. Az örökbefogadók esetében szintén személyi_szám attribútummal azonosítjuk a szereplőket, és részletezzük a nevüket, születési idejüket, telefonszámukat és lakcímüket.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<allatmenhelyek xmlns:xsi="http://www.w3.org/2001/XMLSchema-
   instance" xsi:noNamespaceSchemaLocation="XMLSchemaBB89VX.
   xsd">
  <varosok>
    <varos iranyitoszam="4000-4063">
      <nev>Debrecen</nev>
      <terulet>461</terulet>
      <lakossag>201704</lakossag>
    </varos>
    <varos iranyitoszam="3300,3304">
```

```
    <nev>Eger</nev>
    <terulet>92</terulet>
    <lakossag>49499</lakossag>
  </varos>
  <varos iranyitoszam="3500-3531,
  3532,3533,3534,3535">
    <nev>Miskolc</nev>
    <terulet>236</terulet>
    <lakossag>143502</lakossag>
  </varos>
</varosok>
<dolgozok>
  <dolgozo szemelyi_szam="123456EE">
    <beosztas>Gondozo</beosztas>
    <nev>Kiss Bela</nev>
    <nem>Ferfi</nem>
    <edesanyja_leanykori_neve>Dobo Beata</
       edesanyja_leanykori_neve>
    <lakhely iranyitoszam="3515">
      <utca>Kossuth utca</utca>
      <hazszam>11</hazszam>
    </lakhely>
    <telefonszam>06301234567</telefonszam>
    <telefonszam>06701234567</telefonszam>
  </dolgozo>
  <dolgozo szemelyi_szam="234567RR">
    <beosztas>Allatorvos</beosztas>
    <nev>Nagy Bela</nev>
    <nem>Ferfi</nem>
    <edesanyja_leanykori_neve>Kiss Margit</
```

```
      edesanyja_leanykori_neve>
    <lakhely iranyitoszam="3525">
      <utca>Petofi utca</utca>
      <hazszam>12</hazszam>
    </lakhely>
  </dolgozo>
  <dolgozo szemelyi_szam="345678TT">
    <beosztas>Menedzser</beosztas>
    <nev>Lakatos Gizella</nev>
    <nem>No</nem>
    <edesanyja_leanykori_neve>Toth Eva</
      edesanyja_leanykori_neve>
    <lakhely iranyitoszam="3515">
      <utca>Hunyadi utca</utca>
      <hazszam>1</hazszam>
    </lakhely>
    <telefonszam>06301234569</telefonszam>
  </dolgozo>
</dolgozok>
<allatok>
  <allat azonosito_kod="00001">
    <nev>Foltos</nev>
    <eletkor>1</eletkor>
    <faj>Kutya</faj>
  </allat>
  <allat azonosito_kod="00002">
    <nev>Cirmi</nev>
    <eletkor>3</eletkor>
    <faj>Macska</faj>
  </allat>
```

```
  <allat azonosito_kod="00003">
    <nev>Tappancs</nev>
    <eletkor>4 ev</eletkor>
    <faj>Nyul</faj>
  </allat>
</allatok>
<menhelyek>
  <menhely nyilvantartasi_azonosito="718293002">
    <nev>Miskolci Allatsegito Alapitvany</nev>
    <ferohely>300</ferohely>
    <telefonszam>06701234501</telefonszam>
    <cim iranyitoszam="3529">
      <utca>Sajoszigeti ut</utca>
      <hazszam>23</hazszam>
    </cim>
  </menhely>
  <menhely nyilvantartasi_azonosito="553242112">
    <nev>Ebrendeszeti Telep</nev>
    <ferohely>400</ferohely>
    <telefonszam>06701234502</telefonszam>
    <telefonszam>06301234502</telefonszam>
    <cim iranyitoszam="4002">
      <utca>Bank</utca>
      <hazszam>72</hazszam>
    </cim>
  </menhely>
  <menhely nyilvantartasi_azonosito="620903992">
    <nev>Allatokat Vedjuk Egyutt Alapitvany</nev>
    <ferohely>200</ferohely>
    <telefonszam>06701234503</telefonszam>
```

```
    <cim iranyitoszam="3300">
      <utca>Kulterulet</utca>
      <hazszam>1</hazszam>
    </cim>
  </menhely>
</menhelyek>
<betegsegek>
  <betegseg latin_nev="influenza_felis">
    <nev>Macska Influenza</nev>
    <gyogymod>Antibiotikumos kezeles</gyogymod>
    <gyogyulasi_ido>7</gyogyulasi_ido>
    <tuntetek>orrfolyas, tusszoges</tuntetek>
  </betegseg>
  <betegseg latin_nev="canis_parvovirus">
    <nev>Parvovirus</nev>
    <gyogymod>Intravenas folyadekpotlas</gyogymod>
    <gyogyulasi_ido>6</gyogyulasi_ido>
    <tuntetek>hasmenes, hanyas</tuntetek>
  </betegseg>
  <betegseg latin_nev="feline_infectious_peritonitis">
    <nev>Fertozo hashartyagyulladas(FIP)</nev>
    <gyogymod>Antibiotikumos kezeles</gyogymod>
    <gyogyulasi_ido>16</gyogyulasi_ido>
    <tuntetek>laz, levertseg, fogyas, etvagytalansag</
       tuntetek>
  </betegseg>
  <betegseg latin_nev="myxomatosis">
    <nev>Myxomatosis</nev>
    <gyogymod>Vedooltas</gyogymod>
    <gyogyulasi_ido>14</gyogyulasi_ido>
```

```
    <tuntetek>bor duzzanatok, etvagytalansag</tuntetek>
  </betegseg>
</betegsegek>
<orokbefogadok>
  <orokbefogado szemelyi_szam="987654AB">
    <nev>Horvath Janos</nev>
    <szul_ido>1992.11.02.</szul_ido>
    <telefonszam>06201234567</telefonszam>
    <telefonszam>06301234567</telefonszam>
    <lakhely iranyitoszam="3515">
      <utca>Beke utca</utca>
      <hazszam>5</hazszam>
    </lakhely>
  </orokbefogado>
  <orokbefogado szemelyi_szam="876543CD">
    <nev>Kovacs Anna</nev>
    <szul_ido>1999.09.09.</szul_ido>
        <telefonszam>06201234568</telefonszam>
        <lakhely iranyitoszam="4001">
                <utca>Tavasz utca</utca>
                <hazszam>8</hazszam>
        </lakhely>
      </orokbefogado>
      <orokbefogado szemelyi_szam="876543EF">
              <nev>Horvath Eszter</nev>
              <szul_ido>1990.12.01.</szul_ido>
              <telefonszam>06201234569</telefonszam>
              <lakhely iranyitoszam="3300">
                      <utca>Nyar utca</utca>
                      <hazszam>13</hazszam>
```

```
            </lakhely>
        </orokbefogado>
    </orokbefogadok>
</allatmenhelyek>
```

2.4. Az XML dokumentum alapján XMLSchema készítése

Az XSD fájl az XML dokumentum létrehozását segíti, megadva, hogy mit és milyen formában írhatunk bele, hasonlóan egy interfészhez. Először egyszerű típusokat hoztam létre, amelyek az alapvető elemeket képviselik, majd ezek referálásával összetett típusokat és saját típusokat definiáltam. Ezeket végül szintén referenciaként használom fel a végső struktúra kialakításakor, így a fájl felépítése egyszerű marad. Fontos volt továbbá, hogy különféle korlátozásokat adjak meg az elemekhez és attribútumokhoz, például a típusukat (int, string vagy egyedi típus), hogy kötelezőek-e, vagy hogy az adott elemből hány példány szerepelhet az XML-ben.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <!-- Gyoker -->
    <xs:element name="allatmenhelyek">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="varosok" type="VarosokTipus"
                    />
                <xs:element name="dolgozok" type="
                    DolgozokTipus"/>
                <xs:element name="allatok" type="AllatokTipus"
                    />
```

```
            <xs:element name="menhelyek" type="
               MenhelyekTipus"/>
            <xs:element name="betegsegek" type="
               BetegsegekTipus"/>
            <xs:element name="orokbefogadok" type="
               OrokbefogadokTipus"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Kulcsok -->
    <xs:key name="varosKulcs">
        <xs:selector xpath="varosok/varos"/>
        <xs:field xpath="@iranyitoszam"/>
    </xs:key>

    <xs:key name="menhelyKulcs">
        <xs:selector xpath="menhelyek/menhely"/>
        <xs:field xpath="@nyilvantartasi_azonosito"/>
    </xs:key>

    <xs:key name="dolgozoKulcs">
        <xs:selector xpath="dolgozok/dolgozo"/>
        <xs:field xpath="@szemelyi_szam"/>
    </xs:key>

    <xs:key name="orokbefogadoKulcs">
        <xs:selector xpath="orokbefogadok/orokbefogado"/>
        <xs:field xpath="@szemelyi_szam"/>
    </xs:key>
</xs:element>
```

```
<!-- Telefonszam tipus -->
<xs:simpleType name="TelefonszamTipus">
    <xs:restriction base="xs:string">
        <xs:pattern value="06[237]0[0-9]{7}"/>
    </xs:restriction>
</xs:simpleType>

<!-- Iranyitoszam tipus -->
<xs:simpleType name="IranyitoszamTipus">
    <xs:restriction base="xs:string">
        <xs:pattern value="[0-9]{4}"/>
    </xs:restriction>
</xs:simpleType>

<!-- Nem tipus -->
<xs:simpleType name="NemTipus">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Ferfi"/>
        <xs:enumeration value="No"/>
    </xs:restriction>
</xs:simpleType>

<!-- Faj tipus -->
<xs:simpleType name="FajTipus">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Kutya"/>
        <xs:enumeration value="Macska"/>
        <xs:enumeration value="Nyul"/>
    </xs:restriction>
```

```
</xs:simpleType>

<!-- Cimek tipusa -->
<xs:complexType name="CimTipus">
    <xs:sequence>
        <xs:element name="utca" type="xs:string"/>
        <xs:element name="hazszam" type="xs:
           positiveInteger"/>
    </xs:sequence>
    <xs:attribute name="iranyitoszam" type="
       IranyitoszamTipus" use="required"/>
</xs:complexType>

<!-- Varosok tipusa -->
<xs:complexType name="VarosokTipus">
    <xs:sequence>
        <xs:element name="varos" maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="nev" type="xs:string"
                       />
                    <xs:element name="terulet" type="xs:
                       float"/>
                    <xs:element name="lakossag" type="xs:
                       positiveInteger"/>
                </xs:sequence>
                <xs:attribute name="iranyitoszam" type="xs:
                   string" use="required"/>
            </xs:complexType>
        </xs:element>
```

```
    </xs:sequence>
</xs:complexType>

<!-- Dolgozok tipusa -->
<xs:complexType name="DolgozokTipus">
    <xs:sequence>
        <xs:element name="dolgozo" maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="beosztas" type="xs:
                       string"/>
                    <xs:element name="nev" type="xs:string"
                       />
                    <xs:element name="nem" type="NemTipus"/
                       >
                    <xs:element name="
                       edesanyja_leanykori_neve" type="xs:
                       string"/>
                    <xs:element name="lakhely" type="
                       CimTipus"/>
                    <xs:element name="telefonszam" type="
                       TelefonszamTipus" minOccurs="0"
                       maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="szemelyi_szam" type="xs
                   :string" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
```

```
<!-- Allatok tipusa -->
<xs:complexType name="AllatokTipus">
    <xs:sequence>
        <xs:element name="allat" maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="nev" type="xs:string"
                       />
                    <xs:element name="eletkor" type="xs:
                       string"/>
                    <xs:element name="faj" type="FajTipus"/
                       >
                </xs:sequence>
                <xs:attribute name="azonosito_kod" type="xs
                   :string" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<!-- Menhelyek tipusa -->
<xs:complexType name="MenhelyekTipus">
    <xs:sequence>
        <xs:element name="menhely" maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="nev" type="xs:string"
                       />
```

```
                    <xs:element name="ferohely" type="xs:
                        positiveInteger"/>
                    <xs:element name="telefonszam" type="
                        TelefonszamTipus" minOccurs="0"
                        maxOccurs="unbounded"/>
                    <xs:element name="cim" type="CimTipus"/
                        >
                </xs:sequence>
                <xs:attribute name="
                    nyilvantartasi_azonosito" type="xs:
                    string" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<!-- Betegsegek tipusa -->
<xs:complexType name="BetegsegekTipus">
    <xs:sequence>
        <xs:element name="betegseg" maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="nev" type="xs:string"
                        />
                    <xs:element name="gyogymod" type="xs:
                        string"/>
                    <xs:element name="gyogyulasi_ido" type=
                        "xs:positiveInteger"/>
                    <xs:element name="tuntetek" type="xs:
                        string"/>
```

```
                </xs:sequence>
                <xs:attribute name="latin_nev" type="xs:
                    string" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<!-- Orokbefogadok tipusa -->
<xs:complexType name="OrokbefogadokTipus">
    <xs:sequence>
        <xs:element name="orokbefogado" maxOccurs="
            unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="nev" type="xs:string"
                        />
                    <xs:element name="szul_ido" type="xs:
                        string"/>
                    <xs:element name="telefonszam" type="
                        TelefonszamTipus" minOccurs="0"
                        maxOccurs="unbounded"/>
                    <xs:element name="lakhely" type="
                        CimTipus"/>
                </xs:sequence>
                <xs:attribute name="szemelyi_szam" type="xs
                    :string" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
```

```
    </xs:complexType>

    <!-- Idegen kulcs kapcsolatok -->
    <xs:keyref name="dolgozoLakhelyRef" refer="varosKulcs">
        <xs:selector xpath="dolgozok/dolgozo/lakhely"/>
        <xs:field xpath="@iranyitoszam"/>
    </xs:keyref>

    <xs:keyref name="menhelyVarosRef" refer="varosKulcs">
        <xs:selector xpath="menhelyek/menhely/cim"/>
        <xs:field xpath="@iranyitoszam"/>
    </xs:keyref>

    <xs:keyref name="orokbefogadoLakhelyRef" refer="
       varosKulcs">
        <xs:selector xpath="orokbefogadok/orokbefogado/lakhely
           "/>
        <xs:field xpath="@iranyitoszam"/>
    </xs:keyref>
</xs:schema>
```

3. fejezet

Második feladat

3.1. Adatolvasás

A DOMReadBB89VX osztály beolvassa az XMLBB89VX dokumentumot és blokk formába kiírja majd menti fájlba.

A betöltés a javax.xml.parsers csomag DocumentBuilderFactory osztályának segítségével történik, ami által a betöltött XML-t Document-ként tudjuk kezelni. Majd a kapott dokumentumban a org.w3c.dom csomag NodeList-jei és Element-jei segítségével valósítom meg a tartalom kiírását. A blokk struktúrát ezen elemeken való végig iterálas közben alakítom ki "for" ciklus segítségével, NodeList-enként.

A menhleyek blokkos formában való kíirásának kódja:

```
System.out.println("\nMENHELYEK ADATAI:");
System.out.println("================");
NodeList menhelyek = doc.getElementsByTagName("menhely");
for (int i = 0; i < menhelyek.getLength(); i++) {
    Element menhely = (Element) menhelyek.item(i);
    Element cim = (Element) menhely.getElementsByTagName("cim
       ").item(0);

    System.out.println("\nAzonosito: " + menhely.getAttribute
       ("nyilvantartasi_azonosito"));
```

```
        System.out.println("Nev: " + menhely.getElementsByTagName
            ("nev").item(0).getTextContent());
        System.out.println("Ferohely: " + menhely.
            getElementsByTagName("ferohely").item(0).
            getTextContent());
        System.out.println("Telefonszam: " + menhely.
            getElementsByTagName("telefonszam").item(0).
            getTextContent());
        System.out.println("Cim: " + cim.getElementsByTagName("
            utca").item(0).getTextContent() + " " + cim.
            getElementsByTagName("hazszam").item(0).getTextContent
            () + " (" + cim.getAttribute("iranyitoszam") + ")");
        System.out.println("--------------------");
    }
```

3.2. Adatírás

A DOMWriteBB89VX osztály a beolvasott XML fájl adatait írja ki fa struktúrában, majd menti egy XMLBB89VX1.xml fájlba.

A beolvasást ugyan úgy végeztem mint az előző osztályban, viszont a fa struktrájú kiírást, más megközelítéssel teljesítettem. A beolvasott Document-et egy stack-be töltöttem és egy "while" ciklussal irattam ki a benne található elemeket, a verem kimerüléséig.

Az adatok fa struktúrában való kiírásáért felelős kódrészlet:

```
while (!stack.empty()) {
    Object[] current = stack.pop();
    Node node = (Node)current[0];
    String indent = (String)current[1];

    if (node.getNodeType() == Node.ELEMENT_NODE) {
```

```
System.out.print(indent + node.getNodeName());

if (node.hasAttributes()) {
    NamedNodeMap attributes = node.getAttributes();
    System.out.print(" [");
    for (int i = 0; i < attributes.getLength(); i++) {
        Node attr = attributes.item(i);
        System.out.print(attr.getNodeName() + "=\"" +
           attr.getNodeValue() + "\"");
        if (i < attributes.getLength() - 1) {
            System.out.print(", ");
        }
    }
    System.out.print("]");
}

if (node.getChildNodes().getLength() == 1 &&
    node.getFirstChild().getNodeType() == Node.
       TEXT_NODE) {
    System.out.println(": " + node.getTextContent().
       trim());
} else {
    System.out.println();
}

NodeList children = node.getChildNodes();
for (int i = children.getLength() - 1; i >= 0; i--) {
    Node child = children.item(i);
    if (child.getNodeType() == Node.ELEMENT_NODE) {
        stack.push(new Object[]{child, indent + " "});
```

```
            }
        }
    }
}
```

3.3. Adatlekérdezést

A DOMQueryBB89VX osztály a beolvasott XML-ben keresi meg:

1. Miskolci dolgozókat
2. 300 férőhelynél nagyobb menhelyeket
3. 3 évnél idősebb állatokat
4. 10 napnál tovább gyógyuló betegségeket

A beolvasást ugyan úgy végeztem mint az előző osztályokban, majd a lekérdezést az első feladatban lévő blokkos kiíráshoz használt módszerhez hasonlóan végeztem, de itt csak a kívánt adatokat kerestem és írtam ki.

Kódreszlett egy keresett városban (jelen esetben Miskolci) lakó dolgozók lekérdezését tartalmazza:

```
String keresettvaros = "Miskolc";
System.out.println("1. " + keresettvaros + "-i dolgozok:");
System.out.println("---------------------------");
NodeList varosok = doc.getElementsByTagName("varos");

String keresettirszam = "";

for (int i = 0; i < varosok.getLength(); i++) {
    Element varos = (Element) varosok.item(i);
    Node nevNode = varos.getElementsByTagName("nev").item(0);
```

```
    if (nevNode != null && Objects.equals(nevNode.
       getTextContent(), keresettvaros)) {
        keresettirszam = varos.getAttribute("iranyitoszam");
        break;
    }
}

if (!keresettirszam.isEmpty()) {
    NodeList dolgozok = doc.getElementsByTagName("dolgozo");

    for (int i = 0; i < dolgozok.getLength(); i++) {
        Element dolgozo = (Element) dolgozok.item(i);
        Node lakhelyNode = dolgozo.getElementsByTagName("
           lakhely").item(0);

        if (lakhelyNode instanceof Element) {
            Element lakhely = (Element) lakhelyNode;
            String iranyitoszam = lakhely.getAttribute("
               iranyitoszam");

            if (Objects.equals(iranyitoszam, keresettirszam))
               {
                Node nevNode = dolgozo.getElementsByTagName("
                   nev").item(0);
                Node beosztasNode = dolgozo.
                   getElementsByTagName("beosztas").item(0);

                if (nevNode != null && beosztasNode != null) {
                    String nev = nevNode.getTextContent();
                    String beosztas = beosztasNode.
```

```
                    getTextContent();
                System.out.println("Nev: " + nev + ",
                    Beosztas: " + beosztas);
            }
        }
    }
}
} else {
    System.out.println("Nem talalhato a keresett Varos a xml-
        ben.");
}
```

3.4. Adatmódosítás

A DOMModifyBB89VX osztály az XML dokumentumomat módosítja:

1. Fizetés attribútum létrehozása
2. Állatok életkorának növelése 1 évvel
3. Menhelyek férőhelyének növelése 10%-al
4. Betegségek súlyosságának megállapítása és attribútum létrehozása

A beolvasás módszerén itt sem változtattam. A módosításokhoz először kiválasztottam a kívánt NodeList-et, majd ezen belül megkerestem a változtatás célpontjait, ez lehetett egy Element is vagy akár több Node és elvégeztem rajtuk a kívánt változtatást, például érték változtatása vagy új attribútum hozzáadása.

Ezek után a szóközöket külön eltávolítottam, mivel kiiratáskor nem tudtam máshogy megoldani a szépen struktúráltságot az új elemek beszúrását követően. Vagy struktúrálatlanul került bele az új adat, szimplán egy sorba, vagy ha használtam transforemer indent-et, akkor fölösleges üres sorok

jöttek létre. A szóközök eltávolítása után transformer segítségével szépen formált XML-t kaptam, majd a dokumentumot XMLBB89VX_modify ként mentettem egy XML fájlba.

A kódrészletben a dolgozókhoz egy új elemet, a fizetést létrehozom és feltöltöm értékekkel:

```
System.out.println("1. Modositas: Dolgozoi fizetesek
   hozzaadasa");
System.out.println("-------------------------------------");
NodeList dolgozok = doc.getElementsByTagName("dolgozo");
for (int i = 0; i < dolgozok.getLength(); i++) {
    Element dolgozo = (Element) dolgozok.item(i);
    Element fizetes = doc.createElement("fizetes");

    // Beosztas alapjan allitjuk be a fizetest
    String beosztas = dolgozo.getElementsByTagName("beosztas"
       ).item(0).getTextContent();
    if (beosztas.equals("Allatorvos")) {
        fizetes.setTextContent("450000");
    } else if (beosztas.equals("Gondozo")) {
        fizetes.setTextContent("300000");
    } else {
        fizetes.setTextContent("400000");
    }

    dolgozo.appendChild(fizetes);
    System.out.println(dolgozo.getElementsByTagName("nev").
       item(0).getTextContent() +
                    " fizetese: " + fizetes.getTextContent() +
                        " Ft");
}
```