JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Méhészet

Készítette: **Boján Miron Noel**

Neptunkód: **F4XQRO**

Dátum: **2024.11.20**

Tartalomjegyzék

1. Bevezetés
1.1 Az adatbázis ER modell tervezése
1.2 Az adatbázis konvertálása XDM modellre
1.3 Az XDM modell alapján XML dokumentum készítése
1.4 Az XML dokumentum alapján XMLSchema készítése
2. A második feladat
2.1 Adatolvasás
2.2 Adatírás
2.3 Adatkérdezés
2.4 Adatmódosítás

1. Bevezetés

A feladat leírása

A féléves feladatom célja egy méhészet működésének és felépítésének bemutatása adatmodell segítségével. A projekt során egy olyan rendszert dolgozok ki, amely a méhek életének, a kaptár működésének, valamint a méhészeti folyamatok adatainak rendszerezett és áttekinthető leírását biztosítja XML alapú formátumban.

A feladat során figyelembe veszem a méhészet fontos elemeit, például a különböző méhtípusokat (királynő, dolgozó, here), a kaptár felépítését és a méztermeléshez kapcsolódó adatokat.

A féléves projekt célja, hogy bemutassa az XML használatának gyakorlati alkalmazását egy valós rendszer modellezésében és az adatstruktúrák fejlesztésében.

1.1 Az adatbázis ER modell tervezése

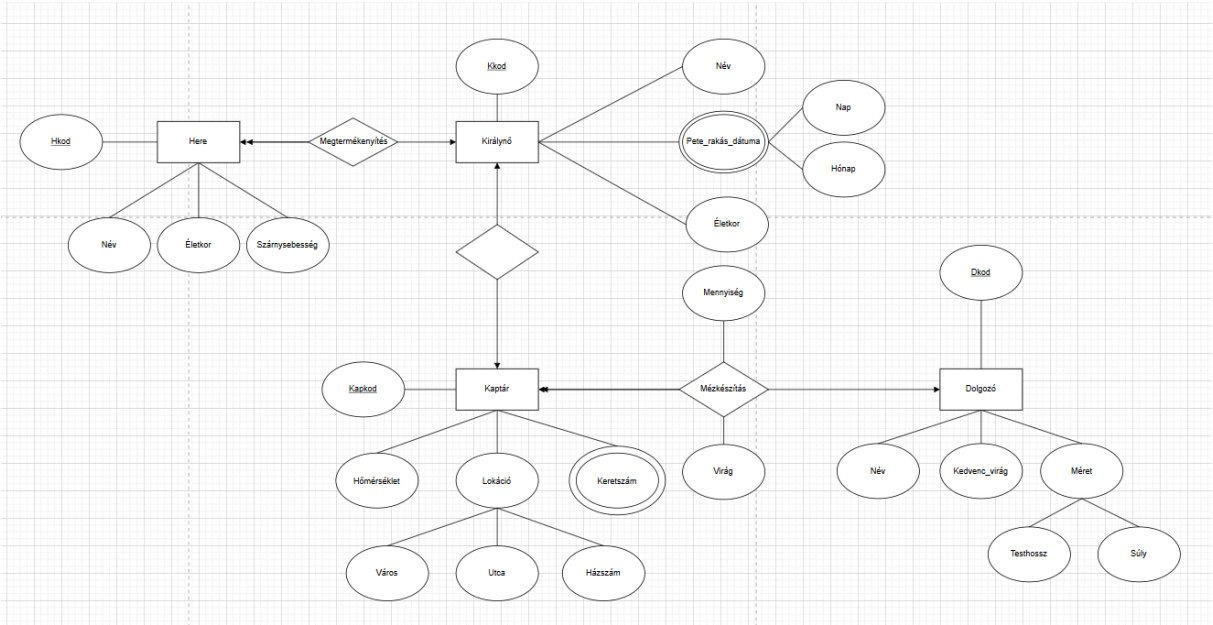
először egy ER modellt készítettem, amely a rendszer alapvető elemeit és azok közötti kapcsolatokat írja le.

A megtervezett ER modell alapján az adatbázis implementációja során a következő lépéseket hajtottam végre:

Adatbázis struktúra létrehozása: Az ER modell entitásait definiáltam, amelyek az attribútumokat is tartalmazzák.

Kapcsolatok kezelése: A táblák között kapcsolatokat hoztam létre az entitások közötti összefüggések fenntartásához.

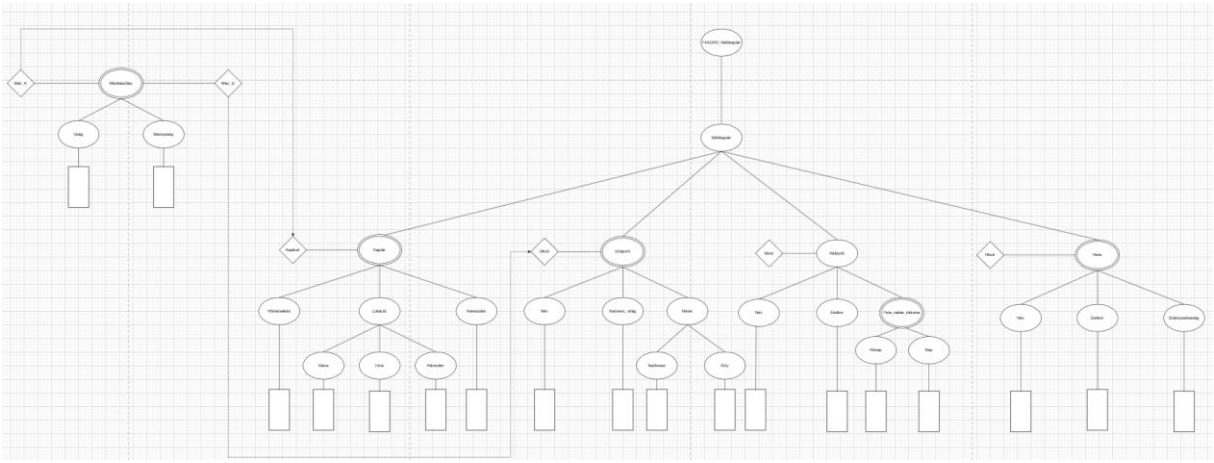
Adatfeltöltés és tesztelés: Példaadatok segítségével ellenőriztem az adatbázis helyes működését és a tervezett kapcsolatok működését.



(1.kép:ER modell)

1.2 Az adatbázis konvertálása XDM modellre

a relációs adatbázis struktúráját egy hierarchikus XDM modellbe konvertáltam. Az adatbázis tábláinak és relációinak hierarchikus XML struktúrába történő leképezését terveztem meg.



(2.kép: XDM modell)

1.3 Az XDM modell alapján XML dokumentum készítése

A modell egyszerű és jól strukturált. Az XML dokumentumban a kaptárak, méhek (királynők, dolgozók, herék), valamint a mézkészítés folyamata szerepel. A kaptárak tartalmazzák a hőmérsékletet és a helyszín adatokat, a méhek egyedi jellemzőket, például méretekkel, életkorral és kedvenc virágokkal rendelkeznek.

A <meheszet> gyökérelem tartalmazza az összes alárendelt adatot, mint például a kaptárakat, királynőket, dolgozókat, heréket és a mézkészítési adatokat.

Az entitások azonosítására kulcsokat használtam, például kapkod (kaptárak), Kkod (királynők), Dkod (dolgozók) és Hkod (herék).

Az összetettebb adatok (pl. lokáció, méret, peté rakás dátuma) egymásba ágyazott elemek segítségével kerültek rögzítésre.

Az azonosítók (pl. kapkod) használata megkönnyíti az entitások közötti kapcsolatok megértését.

```
 1. <?xml version="1.0" encoding="UTF-8"?>
 2.
 3. <meheszet>
 4. <!--Kaptárak-->
 5.     <kaptar kapkod = "kap1">
 6.         <homerseklet>32</homerseklet>
 7.         <lokacio>
 8.             <varos>Győr</varos>
 9.             <utca>Szondi</utca>
10.             <hazszam>5</hazszam>
```

```
        </lokacio>
        <keretszam>16</keretszam>
    </kaptar>

    <kaptar kapkod = "kap2">
        <homerseklet>32</homerseklet>
        <lokacio>
            <varos>Sopron</varos>
            <utca>Kossuth</utca>
            <hazszam>15</hazszam>
        </lokacio>
        <keretszam>8</keretszam>
    </kaptar>

    <kaptar kapkod = "kap3">
        <homerseklet>36</homerseklet>
        <lokacio>
            <varos>Szeged</varos>
            <utca>Petőfi</utca>
            <hazszam>46</hazszam>
        </lokacio>
        <keretszam>32</keretszam>
    </kaptar>

    <!-- Királynők-->
    <kiralyno Kkod = "ki1">
        <nev>ZümZüm</nev>
        <eletkor>1</eletkor>
        <peterakasdatum>
            <honap>Május</honap>
            <nap>1</nap>
        </peterakasdatum>
    </kiralyno>

    <kiralyno Kkod = "ki2">
        <nev>Aranyszárny</nev>
        <eletkor>3</eletkor>
        <peterakasdatum>
            <honap>Június</honap>
            <nap>3</nap>
        </peterakasdatum>
    </kiralyno>

    <kiralyno Kkod = "ki3">
        <nev>Bíborka</nev>
        <eletkor>3</eletkor>
        <peterakasdatum>
            <honap>Április</honap>
            <nap>23</nap>
        </peterakasdatum>
    </kiralyno>

    <!-- Dolgozók-->
    <dolgozo Dkod = "do1">
        <nev>Virágszirom Vilmos</nev>
        <kedvenccirag>Napraforgó</kedvenccirag>
        <meret>
            <testhossz>4</testhossz>
            <suly>6</suly>
```

```
        </meret>
    </dolgozo>

    <dolgozo Dkod = "do2">
        <nev>Sejtépítő Sándor</nev>
        <kedvenccirag>Akác</kedvenccirag>
        <meret>
            <testhossz>5</testhossz>
            <suly>8</suly>
        </meret>
    </dolgozo>

    <dolgozo Dkod = "do3">
        <nev>Szorgos Szilárd</nev>
        <kedvenccirag>Repce</kedvenccirag>
        <meret>
            <testhossz>7</testhossz>
            <suly>4</suly>
        </meret>
    </dolgozo>

    <!-- Herék-->
    <here Hkod="he1">
        <nev>Pollen Péter</nev>
        <eletkor>2</eletkor>
        <szarnysebesseg>11000/s</szarnysebesseg>
    </here>

    <here Hkod="he2">
        <nev>Rovar Róbert</nev>
        <eletkor>1</eletkor>
        <szarnysebesseg>11400/s</szarnysebesseg>
    </here>

    <here Hkod="he2">
        <nev>Gyűjtögető György</nev>
        <eletkor>3</eletkor>
        <szarnysebesseg>10000/s</szarnysebesseg>
    </here>

    <!-- Mézkészítés-->
```

```
    <mezkeszites kapkod = "kap1" Dkod = "do1">
        <virag>Akác</virag>
        <Mennyiség>1</Mennyiség>
    </mezkeszites>

    <mezkeszites kapkod = "kap2" Dkod = "do2">
        <virag>Napraforgó</virag>
        <Mennyiség>5</Mennyiség>
    </mezkeszites>

    <mezkeszites kapkod = "kap3" Dkod = "do3">
        <virag>Levendula</virag>
        <Mennyiség>3</Mennyiség>
    </mezkeszites>

```

```
</meheszet>

```

1.4 Az XML dokumentum alapján XMLSchema készítése

Az XML Schema a méhészet adatainak struktúráját és érvényességi szabályait határozza meg.

A <meheszet> gyökérelem tartalmazza a méhészet összes adatát.
Az elemeket maxOccurs="unbounded" kitétellel jelöltem, hogy többször előfordulhassanak.

A PozitivEgész típus a pozitív egész számokat korlátozza.
A Honap típus előre definiált hónapneveket enged meg.

Lokacio, Meret, és PetekeltetesiDatum összetett típusokat hoztunk létre az adatok logikus csoportosítására.

Ez az XSD biztosítja, hogy a méhészeti XML dokumentum szabályos és következetes legyen.

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <!-- Egyszerű típusok -->
    <xs:simpleType name="pozitivEgesz">
        <xs:restriction base="xs:positiveInteger"/>
    </xs:simpleType>

    <xs:simpleType name="honapTipus">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Január"/>
            <xs:enumeration value="Február"/>
            <xs:enumeration value="Március"/>
            <xs:enumeration value="Április"/>
            <xs:enumeration value="Május"/>
            <xs:enumeration value="Június"/>
            <xs:enumeration value="Július"/>
            <xs:enumeration value="Augusztus"/>
            <xs:enumeration value="Szeptember"/>
            <xs:enumeration value="Október"/>
            <xs:enumeration value="November"/>
```

```
            <xs:enumeration value="December"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- Összetett típusok -->
    <xs:complexType name="lokacioTipus">
        <xs:sequence>
            <xs:element name="varos" type="xs:string"/>
            <xs:element name="utca" type="xs:string"/>
            <xs:element name="hazszam" type="pozitivEgesz"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="meretTipus">
        <xs:sequence>
            <xs:element name="testhossz" type="pozitivEgesz"/>
            <xs:element name="suly">
                <xs:simpleType>
                    <xs:restriction base="xs:positiveInteger">
                        <xs:minInclusive value="1"/>
                        <xs:maxInclusive value="12"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="peterakasDatumTipus">
        <xs:sequence>
            <xs:element name="honap" type="honapTipus"/>
            <xs:element name="nap" type="pozitivEgesz"/>
        </xs:sequence>
    </xs:complexType>


    <xs:element name="meheszet">
        <xs:complexType>
            <xs:sequence>
                <!-- Kaptárak -->
                <xs:element name="kaptar" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="homerseklet" type="pozitivEgesz"/>
                            <xs:element name="lokacio" type="lokacioTipus"/>
                            <xs:element name="keretszam" type="pozitivEgesz"/>
                        </xs:sequence>
                        <xs:attribute name="kapkod" type="xs:ID" use="required"/>
                    </xs:complexType>
                </xs:element>

                <!-- Királynők -->
                <xs:element name="kiralyno" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="nev" type="xs:string"/>
                            <xs:element name="eletkor" type="pozitivEgesz"/>
                            <xs:element name="peterakasdatum" type="peterakasDatumTipus"/>
                        </xs:sequence>
                        <xs:attribute name="Kkod" type="xs:ID" use="required"/>
                    </xs:complexType>
                </xs:element>

                <!-- Dolgozók -->
                <xs:element name="dolgozo" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="nev" type="xs:string"/>
                            <xs:element name="kedvenccirag" type="xs:string"/>
                            <xs:element name="meret" type="meretTipus"/>
                        </xs:sequence>
```

```
                        <xs:attribute name="Dkod" type="xs:ID" use="required"/>
                    </xs:complexType>
                </xs:element>

                <!-- Herék -->
                <xs:element name="here" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="nev" type="xs:string"/>
                            <xs:element name="eletkor" type="pozitivEgesz"/>
                            <xs:element name="szarnysebesseg" type="xs:string"/>
                        </xs:sequence>
                        <xs:attribute name="Hkod" type="xs:ID" use="required"/>
                    </xs:complexType>
                </xs:element>

                <!-- Mézkészítés -->
                <xs:element name="mezkeszites" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="virag" type="xs:string"/>
                            <xs:element name="mennyiseg" type="pozitivEgesz"/>
                        </xs:sequence>
                            <xs:attribute name="kapkod" type="xs:string" use="required" />
                            <xs:attribute name="Dkod" type="xs:string" use="required" />
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>

        <!-- Elsődleges Kulcsok -->
        <xs:key name="kaptar_kulcs">
            <xs:selector xpath="kaptar"/>
            <xs:field xpath="@kapkod"/>
        </xs:key>
        <xs:key name="kiralyno_kulcs">
            <xs:selector xpath="kiralyno"/>
            <xs:field xpath="@Kkod"/>
        </xs:key>
        <xs:key name="dolgozo_kulcs">
            <xs:selector xpath="dolgozo"/>
            <xs:field xpath="@Dkod"/>
        </xs:key>
        <xs:key name="here_kulcs">
            <xs:selector xpath="here"/>
            <xs:field xpath="@Hkod"/>
        </xs:key>

        <!-- Idegen Kulcsok -->
        <xs:keyref name="mezkeszites_kaptar_fk" refer="kaptar_kulcs">
            <xs:selector xpath="mezkeszites"/>
            <xs:field xpath="@kapkod"/>
        </xs:keyref>
        <xs:keyref name="mezkeszites_dolgozo_fk" refer="dolgozo_kulcs">
            <xs:selector xpath="mezkeszites"/>
            <xs:field xpath="@Dkod"/>
        </xs:keyref>
    </xs:element>

</xs:schema>

```

2. A második feladat

2.1 adatolvasás

A program betölti az XML fájlt (XMLNeptunkod.xml) egy DOM objektumként, amely lehetővé teszi az XML dokumentum elemeinek strukturált kezelését.

Az adatok feldolgozása blokkokban történik, amely során minden fontos elem (pl. hallgató neve, születési éve, szakja stb.) kiírásra kerül a konzolra.

A program jelenleg lépésről lépésre olvassa be az XML fájlban található "here" elemek adatait.

```
1. for (int i = 0; i < nodeList.getLength(); i++) {
2.     Node node = nodeList.item(i);
3.     if (node.getNodeType() == Node.ELEMENT_NODE) {
4.         Element elem = (Element) node;
5.         String neve = elem.getElementsByTagName("nev").item(0).getTextContent();
6.         String kor = elem.getElementsByTagName("eletkor").item(0).getTextContent();
7.         String szarnyseb =
elem.getElementsByTagName("szarnysebesseg").item(0).getTextContent();
8.
```

Az elem.getElementsByTagName segítségével kiolvashatók az "nev", "eletkor" és "szarnysebesseg" mezők értékei.

https://github.com/mironprog/XML-SemTask/blob/main/2.feladat/DOMParseF4XQRO/src/DOMReadF4XQRO.java

2.2 Adatírás

XML fájl betöltése és elemzése: Az XMLMeheszet.xml fájlt a DocumentBuilder segítségével betöltjük és a DOM objektummá alakítjuk.

A printNode rekurzív metódus segítségével bejárjuk a dokumentumot, és a gyökérelemtől kezdve kiírjuk a teljes fa struktúrát, beleértve az attribútumokat és szöveges tartalmakat.

Az XMLMeheszet1.xml fájl a DOM objektum tartalmából kerül generálásra a Transformer osztály segítségével, amely a dokumentumot megfelelően formázva menti el.

Ez a program jól mutatja be a DOM API használatát az XML dokumentumok feldolgozására. A Transformer osztály segítségével könnyen szerkeszthető az XML fájl.

```
1. TransformerFactory transformerFactory = TransformerFactory.newInstance();
2.             Transformer transformer = transformerFactory.newTransformer();
3.             transformer.setOutputProperty(OutputKeys.INDENT, "yes");
4.
```

A DOM API Transformer osztályát arra használjuk, hogy egy XML dokumentum DOM objektumát szöveges formátumú XML dokumentummá alakítsuk.

https://github.com/mironprog/XML-SemTask/blob/main/2.feladat/DOMParseF4XQRO/src/DOMWriteF4XQRO.java

2.3 Adatkérdezés

A program DOM API-t használ az XML dokumentum elemzéséhez, és négy különböző lekérdezést hajt végre, amelyek során az elemek adatait gyűjti ki.

Az eredményeket strukturáltan jeleníti meg a konzolon, lehetővé téve az XML dokumentumban tárolt információk olvashatóságát.

A DOM API-val az XML fájlt beolvassuk és a memóriában fa struktúrává alakítjuk.

A DocumentBuilderFactory és DocumentBuilder osztályok segítségével kezeljük az XML-t.

1. lekérdezés: A kaptar elemek "kapkod" attribútumainak listázása.

2. lekérdezés: Az összes lokacio városának kiírása.

3. lekérdezés: A kiralyno elemek neve és életkora.

4. lekérdezés: A mézkészítés típusai (virag) és mennyiségeik.

```
1. System.out.println("\n1. lekérdezés: Kaptárak és azonosítóik:");
2.             NodeList kaptarList = doc.getElementsByTagName("kaptar");
3.             for (int i = 0; i < kaptarList.getLength(); i++) {
4.                 Node kaptarNode = kaptarList.item(i);
5.                 if (kaptarNode.getNodeType() == Node.ELEMENT_NODE) {
6.                     Element kaptarElement = (Element) kaptarNode;
```

```
 7.                     System.out.println("Kaptár azonosító: " +
kaptarElement.getAttribute("kapkod"));
 8.                 }
 9.             }
10.
```

Ez a rész az XML dokumentumból az összes "kaptar" elem tagjait gyűjti össze és kiírja az azokhoz tartozó ID értékeket, amelyek az egyes kaptárak azonosítóit tartalmazzák.

https://github.com/mironprog/XML-SemTask/blob/main/2.feladat/DOMParseF4XQRO/src/DOMQueryF4XQRO.java

2.4 Adatmódosítás

A program betölt egy XML dokumentumot és különféle módosításokat hajt végre. A program egy új "dolgozo" elem tulajdonságát is frissíti, például kedvenc virágának nevét megváltoztatja, majd az eredményül kapott dokumentumot egy új fájlba menti a Transformer API segítségével.

A setTextContent metódussal módosítjuk az elem tartalmát.

A createElement és appendChild metódusokkal új elemeket hozunk létre és illesztünk a dokumentumba.

```
1. NodeList kaptarList = doc.getElementsByTagName("kaptar");
2.             if (kaptarList.getLength() > 0) {
3.                 Element firstKaptar = (Element) kaptarList.item(0);
4.                 firstKaptar.getElementsByTagName("homerseklet").item(0).setText
Content("35");
5.                 System.out.println("1. Módosítás: Az első kaptár hőmérséklete
35-re módosítva.");
6.             }
7.
```

A kód először lekéri az összes "kaptar" elemet az XML dokumentumból, majd ellenőrzi, hogy van-e ilyen elem. Ha található, a legelső "kaptar" elem "homerseklet" értékét 35-re módosítja.

https://github.com/mironprog/XML-SemTask/blob/main/2.feladat/DOMParseF4XQRO/src/DOMModifyF4XQRO.java