Jegyzőkönyv

Webes adatkezelő környezetek

Féléves feladat

Gyógyszertár adatkezelő rendszere

Készítette: Mizerák Bence

Neptunkód: MO0VEY

Dátum: 2025. november

Miskolc, 2025

Tartalomjegyzék

1. Bevezetés **2**

2. A rendszer áttekintése **2**

2.1. Domain és fájlstruktúra 2

3. ER modell **3**

4. XDM modell **3**

5. XML és XML Schema **4**

6. Java DOM programok **4**

7. Összegzés **7**

1. Bevezetés

Ez a jegyzőkönyv a *Webes adatkezelő környezetek* tárgy féléves feladatát mutatja be. A projekt célja egy gyógyszertár adatkezelő rendszerének modellje volt XML, XML Schema (XSD) és Java DOM API felhasználásával.

Az XML a strukturált adatok platformfüggetlen cseréjére alkalmas szabványos formátum. A séma (XSD) biztosítja, hogy az XML dokumentum szerkezete, kulcsai és idegen kulcsai jól definiáltak legyenek, míg a Java DOM API lehetővé teszi az adatok programozott beolvasását, módosítását és kiírását.

2. A rendszer áttekintése

2.1. Domain és fájlstruktúra

A modell egy gyógyszertár rendszerének főbb adatait tartalmazza. A központi XML állomány a `MO0VEY_XML.xml`, amelynek gyökéreleme a <GyogyszertarRendszer>. A főbb részegységek:

- `Gyogyszerek/Gyogyszer`: gyógyszerek törzse (név, hatóanyag, receptköteles, kategória, leírás, mellékhatások),
- `Kiszerelesek/Kiszereles`: egyes gyógyszerek kiszerelései (forma, mennyiség, ár, egység),
- `Betegsegek/Betegseg`: betegségek leírása, figyelmeztetések, tünetek,
- `BetegsegGyogyszerKapcsolatok/BetegsegGyogyszer`: mely gyógyszer mely betegségre ajánlott,
- `Rendelesek`: vevői rendelések fej- és tételadatai,
- `Szallitok/Szallito`: beszállítók adatai,
- `Beszerzesek`: beszerzési számlák fej- és tételadatai.

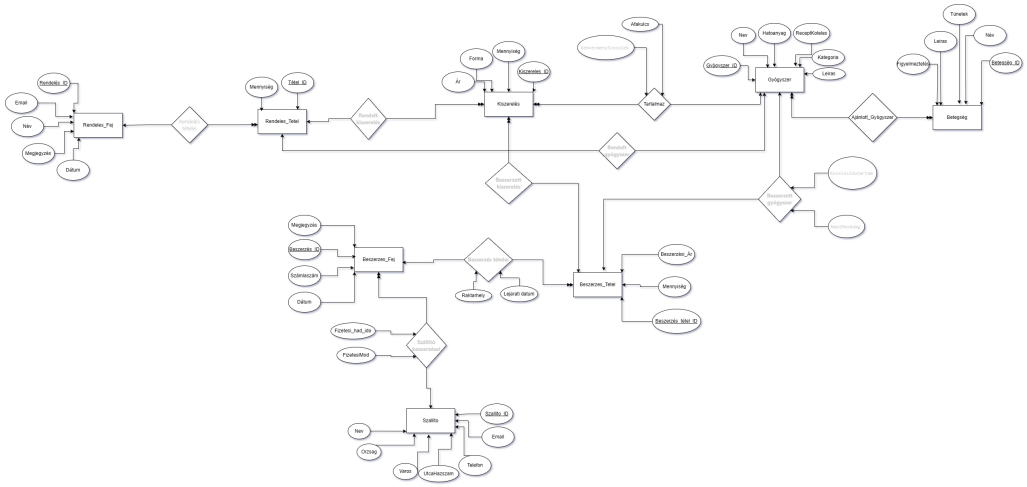
A fenti XML-hez tartozó sémafájl a `MO0VEY_XMLSchema.xsd`, míg a DOM alapú Java programok a `MO0VEYDOMParse` mappában találhatók: `MO0VEYDomRead.java`, `MO0VEYDomWrite.java`, `MO0VEYDomModify.java`, `MO0VEYDomQuery.java`. A módosított állapotot a `MO0VEY_XML_modified.xml` tartalmazza.

3. ER modell

A logikai adatmodellt először egy entitás–kapcsolat (ER) diagramon ábrázoltuk. A fő entitások: *Gyogyszer*, *Kiszereles*, *Betegseg*, *Szallito*, *Rendeles_Fej*, *Rendeles_Tetel*, *Beszerzes_Fej*, *Beszerzes_Tetel*, valamint a *BetegsegGyogyszer* kapcsoló entitás.

A kapcsolatok között megjelennek az 1:N és N:M típusok is. Például egy *Gyogyszer*-hez több *Kiszereles* tartozhat (1:N), míg egy *Betegseg*-hez több *Gyogyszer* is ajánlható, és fordítva (N:M), amit a *BetegsegGyogyszer* entitás közvetít.

Az ER modellt az 1 ábra szemlélteti.



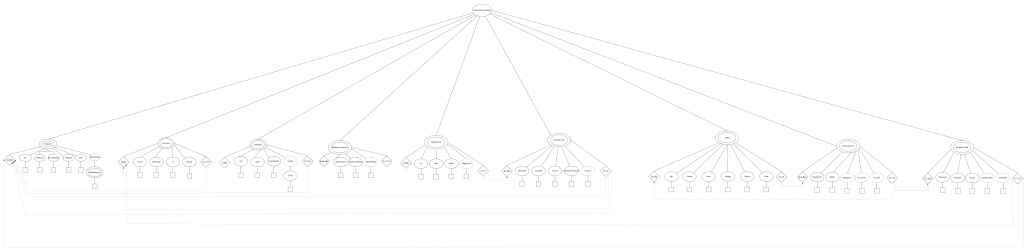
1. ábra. 1. ábra: A gyógyszertár ER modellje

4. XDM modell

Az ER diagramból XDM (XML Data Model) modellt készítettünk. Ebben minden entitás egy XML elemnek felel meg, a kulcs tulajdonságok pedig attribútumként jelennek meg. A gyökér a `GyogyszertarRendszer` elem, ez tartalmazza a fenti részfákat (`Gyogyszerek`, `Kiszerelesek`, `Betegsegek`, stb.).

Külön figyelmet kapott, hogy minden `*_ID` azonosító *attribútumként*, ne pedig gyerekelemként szerepeljen, így a kulcs/idegen kulcs kapcsolatok természetesebben írhatók le az XSD-ben `xs:key` és `xs:keyref` használatával.

Az XDM modell felépítését a 2 ábra mutatja be.



2. ábra. 2. ábra: A gyógyszertár XDM modellje

5. XML és XML Schema

Az XDM modell alapján készült el a `MO0VEY_XML.xml` állomány. Az azonosítók mindenhol attribútumként jelennek meg, például:

- `<Gyogyszer Gyogyszer_ID="G001"> ... </Gyogyszer>`,
- `<Kiszereles Kiszereles_ID="K001" Gyogyszer_ID="G001"> ...`,
- `<Betegseg Betegseg_ID="B001"> ...`,
- `<Rendeles_Fej Rendeles_ID="R001"> ....`

Az XSD-ben ezekhez kulcsokat és idegen kulcsokat definiáltunk. Például `GyogyszerKey` a gyógyszerek elsődleges kulcsa, míg a `Kiszereles_GyogyszerRef` biztosítja, hogy minden `Kiszereles@Gyogyszer_ID` egy létező `Gyogyszer@Gyogyszer_ID`-re hivatkozzon. Hasonló kulcs–kulcsidegen kapcsolat van a rendelések és rendeléssorok, valamint a beszerzések és beszerzéssorok között is.

Külön feladat volt, hogy minden ID-elem attribútummá alakítása után az `xs:key` és `xs:keyref` kifejezésekben a mezőket `@AttribútumNév` formában hivatkoztassuk, illetve a kulcsok láthatósági tartományát úgy állítsuk be, hogy a vonatkozó keyref-ek ne essenek ki a scope-ból.

6. Java DOM programok

A projekt második része a Java DOM API használata volt. Négy önálló program készült:

- **MO0VEYDomRead**: beolvassa az XML-t és fastruktúraként kiírja a konzolra,
- **MO0VEYDomWrite**: hasonló fa-jellegű kiírást végez, majd elmenti az eredményt a `MO0VEY1XML.xml` állományba,

- **MO0VEYDomModify**: módosítja az adatokat (új gyógyszer felvétele, megjegyzés változtatása, mennyiség és összeg újraszámolása, beszállító törlése) és elmenti a `MO0VEY_XML_modified.xml` fájlba,
- **MO0VEYDomQuery**: egy konkrét rendelés adatait gyűjti össze és számolja ki a végösszeget.

Azonosítókhoz a Java kódban már mindenhol attribútumként férünk hozzá, például `getAttribute("Rendeles_ID")` vagy `getAttribute("Kiszereles_ID")`. Az alábbi kódrészlet a `MO0VEYDomQuery` osztályból mutatja be, hogyan történik egy konkrét rendelés (`R001`) lekérdezése és összegzése.

Listing 1. 3. ábra: Részlet a MO0VEYDomQuery osztályból

```
1  private static void printRendelesById(Document doc, String
       rendelesId) {
2      NodeList rendelFejek = doc.getElementsByTagName("
           Rendeles_Fej");
3      for (int i = 0; i < rendelFejek.getLength(); i++) {
4          Element fej = (Element) rendelFejek.item(i);
5          String rId = fej.getAttribute("Rendeles_ID");
6          if (rendelesId.equals(rId)) {
7              String nev = fej.getElementsByTagName("Nev").item(0)
                   .getTextContent();
8              String email = fej.getElementsByTagName("Email").
                   item(0).getTextContent();
9              String datum = fej.getElementsByTagName("Datum").
                   item(0).getTextContent();
10             String megjegyzes = fej.getElementsByTagName("
                   Megjegyzes").item(0)
11                                         .getTextContent();
12
13             System.out.println("Rendeles ID: " + rId);
14             System.out.println("Nev: " + nev);
15             System.out.println("E-mail: " + email);
16             System.out.println("Datum: " + datum);
17             System.out.println("Megjegyzes: " + megjegyzes);
18
19             int osszeg = 0;
20             NodeList tetelek = doc.getElementsByTagName("
                   Rendeles_Tetel");
21             for (int j = 0; j < tetelek.getLength(); j++) {
```

```
22                  Element tetel = (Element) tetelek.item(j);
23                  String tetelRId = tetel.getAttribute("
                        Rendeles_ID");
24                  if (rendelesId.equals(tetelRId)) {
25                      int tetelOsszeg = Integer.parseInt(
26                          tetel.getElementsByTagName("Osszeg").
                                item(0).getTextContent()
27                      );
28                      osszeg += tetelOsszeg;
29                  }
30              }
31              System.out.println("Vegosszeg:␣" + osszeg + "␣Ft");
32              break;
33          }
34      }
35  }
```

A kódrészlet jól szemlélteti, hogy a rendelés fej- és tételadatai attribútumalapú azonosítók segítségével kapcsolódnak össze, ami megfelel az XSD-ben definiált kulcs–kulcsidegen kapcsolatoknak.

7. Összegzés

A projekt során egy teljes gyógyszertári adatkezelő példát valósítottunk meg az ER modelltől kezdve az XDM modellen, XML/XSD pároson át egészen a Java DOM alapú feldolgozásig. A rendszer jól demonstrálja, hogyan lehet az elméleti adatmodelleket gyakorlatban is használható, validálható XML struktúrává, majd programozottan feldolgozható adathalmazzá alakítani.