Jegyzőkönyv

Webes adatkezelő környezetek

Féléves feladat

Logisztikai raktár kezelő rendszer

Készítette: Nyakó Sándor

Neptunkód: PHRO7R

Dátum: 2025. december

Miskolc, 2025

Tartalomjegyzék

Bevezetés 3
A feladat leírása 3
1. Feladat 4
1.1 Az adatbázis ER modell tervezése 4
1.3 Az adatbázis konvertálása XDM modellre 7
1.3 Az XDM modell alapján XML dokumentum készítése 8
1.4 Az XML dokumentum alapján XML Schema (XSD) készítése 9
2. feladat 16
2.1 Adatolvasás (DOM) 16
2.2 Adat-lekérdezés 18
2.3 Adatmódosítás (DOM) 21

Bevezetés

A jegyzőkönyv célja egy XML-alapú adatstruktúrára épülő logisztikai raktárkezelő rendszer megtervezése és bemutatása. Ennek során először az adatmodellezési lépések kerülnek kidolgozásra: a fogalmi szintet lefedő ER modell, majd ennek hierarchikus, fastruktúrájú megfelelője, az XDM. Ezek után ismertetésre kerül a modell alapján felépített XML dokumentum, valamint az ahhoz tartozó XML séma, amely a szerkezetet, adattípusokat és érvényességi feltételeket rögzíti.

A beszámoló második része a dokumentum programozott feldolgozására fókuszál, a DOM szabvány alkalmazásával. A cél a PHRO7R_XML.xml állomány beolvasása, áttekintése, az adatok lekérdezése és a dokumentum tartalmának módosítása Java nyelv segítségével. A feldolgozás során:

- megtörténik a teljes XML dokumentum blokkszerű kiírása,
- legalább négy hasznos információ kinyerése lekérdezésekkel,
- valamint legalább négy, az adatintegritást megőrző módosítás végrehajtása, a változtatások és az érintett elemek blokkszerű megjelenítésével.

A feladat leírása

A beadandó témája egy olyan logisztikai raktárkezelő rendszer volt, amely a raktárak mindennapi működésének támogatására és a készletek pontos nyilvántartására szolgál. A rendszer célja, hogy strukturált és átlátható módon kezelje mindazon információkat, amelyek egy modern logisztikai vállalkozás működéséhez szükségesek, ideértve a beszállítókat, a fuvarozó cégeket, a dolgozókat, a raktárakat, a különböző termékeket és a készletmozgásokat.

A rendszer struktúrája lehetővé teszi a teljes logisztikai folyamat átlátható, következetes és pontos dokumentálását — a beszállítók és szállítók kezelésétől kezdve a raktárak működésén át egészen a termékek fizikai és mennyiségi nyilvántartásáig.

1. Feladat

1.1 Az adatbázis ER modell tervezése

A raktárkezelő rendszer fogalmi adatmodelljének elkészítéséhez ER-diagramot hoztam létre, amelyben meghatároztam a rendszerhez tartozó entitásokat, azok tulajdonságait és a közöttük lévő kapcsolatokat. A cél egy olyan átlátható adatstruktúra kialakítása volt, amely lefedi a raktárak készletezését, a beszállítók és szállítócégek kezelését, valamint a raktárak személyi felelőseit.

A tervezés kezdetén számításba vettem egy megrendelés-kezelési modul beépítését is, amely a raktárakhoz kapcsolódó bejövő és kimenő megrendeléseket kezelte volna. A modell átgondolása során azonban kiderült, hogy ez a megoldás jelentősen bonyolítaná a struktúrát, miközben a beadandó lényegi fókusza a raktárak, a termékek és a logisztikai folyamatok modellezése.

Emiatt a modul helyett egy külön SzállítóCég entitást vezettem be, amely egyszerűen és hatékonyan reprezentálja a kiszállítási feladatokat. Ezáltal a rendszer felépítése letisztultabb, könnyebben áttekinthető lett, és jobban követi a valós logisztikai működést.

Entitások és tulajdonságaik

1. **Beszállító**
    - *BeszallitoId*: egyedi azonosító
    - *Cegnev*: a beszállító neve
    - *Adoszam*: adószám
    - *Elerhetoseg*: többértékű (pl. telefon, e-mail)
2. **SzállítóCég**
    - *SzallitoCegId*: egyedi azonosító
    - *Nev*: a cég neve
    - *Adoszam*: adószám
    - *Elerhetoseg*: többértékű

3. **Dolgozó**
    - *DolgozoId*: egyedi azonosító
    - *Nev*: dolgozó neve
    - *Beosztas*: munkakör
    - *Email*: e-mail cím
    - *Telefonszam*: elérhetőség
4. **Raktár**
    - *RaktarId*: egyedi azonosító
    - *Nev*: raktár neve
    - *Cim*: összetett (irányítószám, város, utca, házszám)
    - *Tipus*: raktár típusa (pl. Főraktár)
5. **Termék**
    - *TermekId*: egyedi azonosító
    - *Nev*: terméknév
    - *Meret*: összetett (hossz, szélesség, magasság)
    - *Egysegar*: egységár
6. **Raktár–Termék kapcsolat**
    - *RaktarId*: hivatkozás a raktárra
    - *TermekId*: hivatkozás a termékre
    - *KeszletDb*: készlet mennyisége az adott raktárban

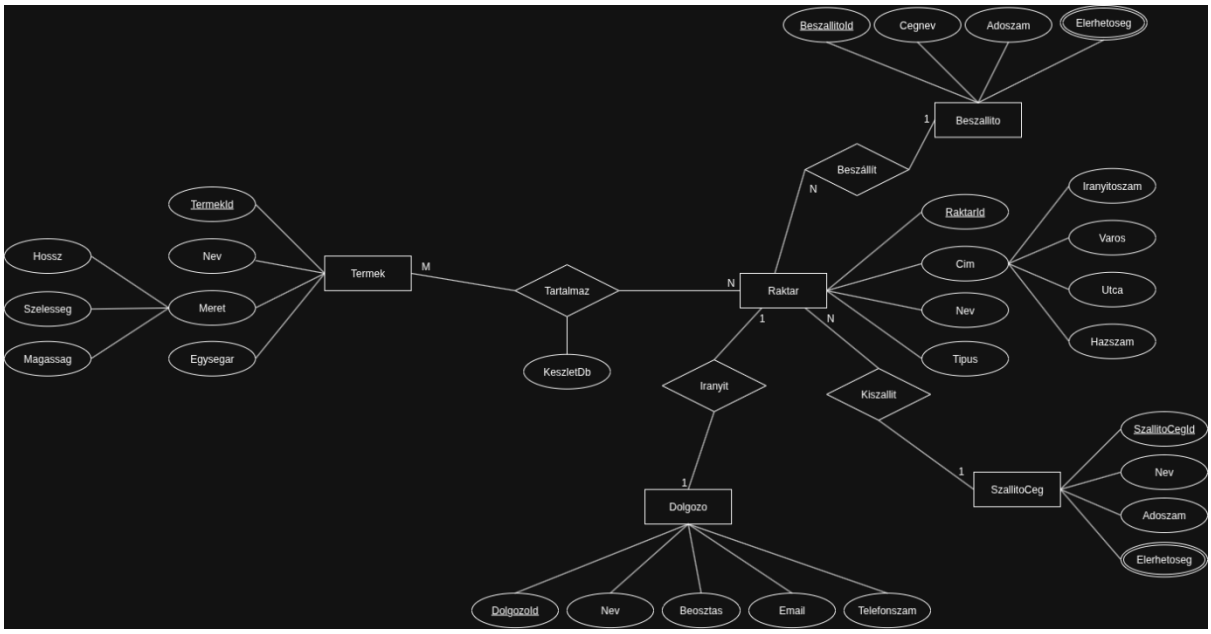
Kapcsolatok

- **Beszállító – Raktár (1:N):** egy beszállító több raktárt is elláthat.
- **SzállítóCég – Raktár (1:N):** egy szállító partner több raktár kiszállítását végezheti.

- **Dolgozó – Raktár (1:1):** minden raktárhoz egy kijelölt felelős dolgozó tartozik.
- **Raktár – Termék (N:M):** egy raktár többféle terméket tárolhat, és egy termék több raktárban is elérhető; a kapcsolat attribútuma a *KeszletDb*.

Összegzés

Az elkészült ER modell hat fő entitást és többféle kapcsolatot tartalmaz (1:1, 1:N, N:M), továbbá normál, kulcs-, összetett és többértékű attribútumokat is. A modell jól tükrözi a logisztikai raktárak működését és biztos alapot ad a további XDM és XML struktúrák kialakításához.



1. ábra: ER diagram

1.3 Az adatbázis konvertálása XDM modellre

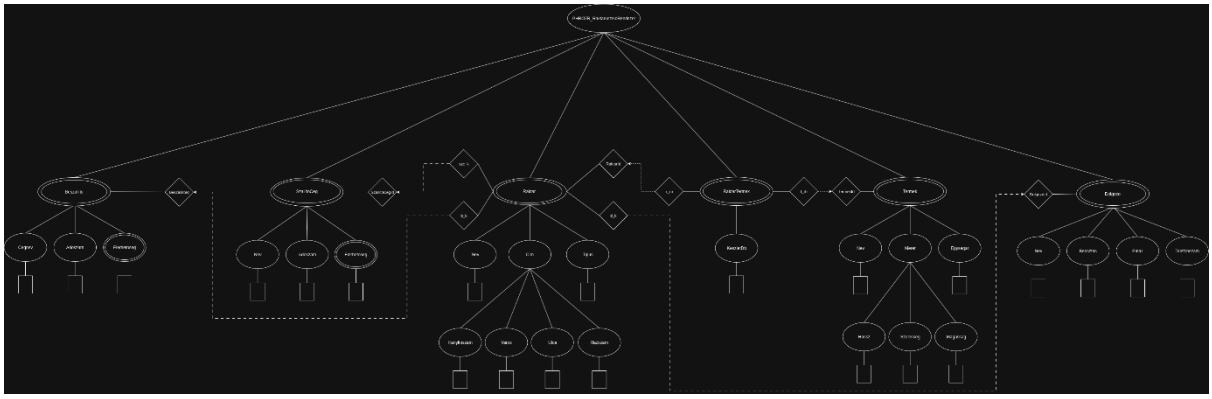
Az XDM modell kialakításakor a korábban megtervezett ER modellt és az egyedek közötti kapcsolatokat vettem alapul. Míg az ER modellben a kapcsolatok és kardinalitások dominálnak, az XDM már az XML dokumentum hierarchikus felépítését jeleníti meg. Itt a hangsúly a szintek és a szülő–gyerek viszonyok ábrázolásán van, amelyek az XML fa struktúráját tükrözik.

A fa jellegű felépítés miatt ebben a modellben minden entitás külön elemként jelenik meg, és a korábbi attribútumok nagy része is önálló alárendelt elemként szerepel. Attribútumként gyakorlatilag csak az elsődleges és idegen kulcsok maradnak meg. A teljes struktúra egyértelmű gyökéreleme a PHRO7R_RaktarKezeloRendszer, amely alá az összes 1. szintű entitás – Beszallito, SzallitoCeg, Raktar, Termek, Dolgozo és RaktarTermek – közvetlenül kapcsolódik.

Az XDM modellben az entitásokat ellipszisekkel jelöltem, az elsődleges és idegen kulcsokat rombusz szimbólumokkal. A többértékű elemek – például az Elérhetőség – dupla keretes megjelenítést kaptak, ezzel jelezve, hogy ugyanazon szülőelem alatt több előfordulásuk is lehet. Azon elemek esetén, amelyeknek már nincs további alárendelt gyerekelemük, téglalap jelzi, hogy az XML dokumentumban ezen a ponton már csak szöveges érték szerepel.

A modellben a hierarchiaviszonyokat egyszerű vonalak jelölik; ezek itt nem a relációs kapcsolatokat, hanem a szülő–gyerek struktúrát mutatják. Az idegen kulcsokból induló hivatkozásokat szaggatott vonallal ábrázoltam, amelyek a megfelelő entitások elsődleges kulcsaira mutatnak. Ez különösen fontos a Raktár–Termék (N:M) kapcsolat esetén, ahol a RaktarTermek kapcsolóentitás tartalmazza a két hivatkozott entitás idegen kulcsait, valamint a készlet mennyiségét (KeszletDb).

Az elkészült XDM modell így egy jól áttekinthető, XML dokumentummá közvetlenül alakítható hierarchiát ad, amely követi az ER modell felépítését, de már az XML adatstruktúrák logikájához igazodik.



2. ábra: XDM modell

1.3 Az XDM modell alapján XML dokumentum készítése

Az XDM hierarchiát követve elkészítettem a PHRO7R_RaktarkezeloRendszer gyökérelemű XML dokumentumot. A fő entitások önálló elemek (Beszallito, SzallitoCeg, Dolgozo, Raktar, Termek, RaktarTermek), az összetett tulajdonságok (pl. Cim, Meret) beágyazott al-elemek, a többértékű elemek (Elerhetoseg) ismétlődhetnek. Az azonosítók és hivatkozások XDM-nek megfelelően attribútumként jelennek meg (pl. Termekid, Raktarid, illetve r_rb, t_rb, R_k, d_rb).

Lényeges kódrészletek és magyarázat

- Gyökérelem + XSD hivatkozás

```
<PHRO7R_RaktarkezeloRendszer
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="PHRO7R_XMLSchema.xsd">
```

  A dokumentum egyértelmű gyökérrel indul, és az XSD séma fájlra hivatkozik a validációhoz.

- Többértékű elem (Elerhetoseg) – legalább két előfordulás/elem

```
<!-- 1. Beszállító példány -->
<Beszallito>
      <Cegnev>TechParts Kft.</Cegnev>
      <Adoszam>12345678-2-42</Adoszam>
      <Elerhetoseg>+36-1-234-5678</Elerhetoseg>
      <Elerhetoseg>techparts@email.hu</Elerhetoseg>
</Beszallito>
```

  A több értékű elérhetőség többször is megadható ugyanazon szülő alatt.

- Összetett attribútum elemként (Cím)

```
<Raktar Raktarid="R001" R_k="SC001" d_rb="D001">
      <Nev>Központi Raktár</Nev>
      <Cim>
            <Iranyitoszam>1117</Iranyitoszam>
            <Varos>Budapest</Varos>
            <Utca>Irinyi József utca</Utca>
            <Hazszam>42</Hazszam>
      </Cim>
      <Tipus>Főraktár</Tipus>
</Raktar>
```

- N:M kapcsolóelem (RaktarTermek) kulcsreferenciákkal

```
<RaktarTermek r_rb="R001" t_rb="T001">
   <KeszletDb>150</KeszletDb>
</RaktarTermek>
```

  A r_rb és t_rb attribútumok a hivatkozott raktár és termék azonosítói; a kapcsolat saját adata a készlet.

- A dokumentum végig kommentált, a blokkok és példányok jól elkülönülnek.

Teljes PHRO7R_XML.xml fájl

1.4 Az XML dokumentum alapján XML Schema (XSD) készítése

Az XDM modellből kiindulva elkészítettem az XML Schema-t, amely formálisan rögzíti a dokumentum szerkezetét, az adattípusokat és a hivatkozási (kulcs–kulcshivatkozás) szabályokat. A megvalósítás menete:

1. Saját egyszerű típusok definiálása (azonosító minták, formátumok).
2. Komplex típusok felépítése (elemsorrend, ismétlődés).
3. Gyökérelem megadása, a fő gyűjtőelemek felsorolásával.

4. Kulcsok (xs:key) és kulcshivatkozások (xs:keyref) rögzítése az adatintegritásért.

Teljes PHRO7R_XMLSchema.xsd fájl

Lényeges elemek:

- Saját egyszerű típusok – formátumok és minták (pl. adószám, azonosítók, e-mail, telefon):

```
<xs:simpleType name="AdoszamType">
  <xs:annotation>
     <xs:documentation>Magyar adószám formátum: ########-
###</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
     <xs:pattern value="[0-9]{8}-[0-9]{1}-[0-9]{2}"/>
  </xs:restriction>
</xs:simpleType>
```

Így garantáljuk a magyar adószám és azonosítók helyes formátumát

- Komplex típusok – összetett mezők és ismétlődések (XDM logikát követve):

```
<!-- Cím összetett típus -->
    <xs:complexType name="CimType">
        <xs:annotation>
            <xs:documentation>Cím típus: irányítószám, város,
utca, házszám</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <!-- Irányítószám elem -->
            <xs:element name="Iranyitoszam"
type="IranyitoszamType">
                <xs:annotation>
                    <xs:documentation>4 számjegyű magyar
irányítószám</xs:documentation>
                </xs:annotation>
            </xs:element>
            <!-- Város elem -->
            <xs:element name="Varos" type="NevType">
                <xs:annotation>
                    <xs:documentation>Település
neve</xs:documentation>
                </xs:annotation>
```

```
            </xs:element>
            <!-- Utca elem -->
            <xs:element name="Utca" type="xs:string">
                <xs:annotation>
                    <xs:documentation>Utca neve (közterület
típussal)</xs:documentation>
                </xs:annotation>
            </xs:element>
            <!-- Házszám elem -->
            <xs:element name="Hazszam" type="xs:string">
                <xs:annotation>
                    <xs:documentation>Házszám (szám vagy
szám/betű kombináció)</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
         </xs:complexType>

<!-- Beszállító összetett típus -->
    <xs:complexType name="BeszallitoType">
        <xs:annotation>
            <xs:documentation>Beszállító entitás: cég adatai
és elérhetőségek</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <!-- Cégnév elem -->
            <xs:element name="Cegnev" type="NevType">
                <xs:annotation>
                    <xs:documentation>Beszállító cég hivatalos
neve</xs:documentation>
                </xs:annotation>
            </xs:element>
            <!-- Adószám elem -->
            <xs:element name="Adoszam" type="AdoszamType">
                <xs:annotation>
                    <xs:documentation>Beszállító
adószáma</xs:documentation>
                </xs:annotation>
            </xs:element>
            <!-- Elérhetőség elemek (legalább 1, lehet több) -
->
            <xs:element name="Elerhetoseg" type="xs:string"
maxOccurs="unbounded">
```

```
                <xs:annotation>
                    <xs:documentation>Elérhetőség: telefon
vagy email (többszörös)</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
         </xs:complexType>
```

Az összetett attribútumok elemekre bontva, a többértékű mezők (pl. Elerhetoseg) ismétlődhetnek.

- Gyökérelem és fő gyűjtőelemek:

```
<xs:element name="PHRO7R_RaktarkezeloRendszer">
        <xs:annotation>
            <xs:documentation>
                PHRO7R Raktárkezelő Rendszer gyökéreleme.
                Tartalmazza az összes entitást: beszállítók,
szállító cégek,
                dolgozók, raktárak, termékek és raktár-termék
kapcsolatok.
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <!-- Beszállító elemek (legalább 1) -->
                <xs:element name="Beszallito"
type="BeszallitoType" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Beszállító entitás
példányok</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <!-- Szállító cég elemek (legalább 1) -->
                <xs:element name="SzallitoCeg"
type="SzallitoCegType" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Szállító cég entitás
példányok</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <!-- Dolgozó elemek (legalább 1) -->
```

```
                <xs:element name="Dolgozo" type="DolgozoType"
maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Dolgozó entitás
példányok</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <!-- Raktár elemek (legalább 1) -->
                <xs:element name="Raktar" type="RaktarType"
maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Raktár entitás
példányok</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <!-- Termék elemek (legalább 1) -->
                <xs:element name="Termek" type="TermekType"
maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Termék entitás
példányok</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <!-- Raktár-Termék kapcsolat elemek (legalább
1) -->
                <xs:element name="RaktarTermek"
type="RaktarTermekType" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Raktár-Termék
kapcsolat példányok (készlet nyilvántartás)</xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
             </xs:complexType>
```

Az XDM-ben tervezett hierarchia visszaköszön: minden fő entitás a gyökér alatt szerepel, tetszőleges számban.

- Kulcsok és hivatkozások – PK/FK megfeleltetés XSD-ben

```
<!-- KULCS ÉS HIVATKOZÁS MEGKÖTÉSEK -->

        <!-- Szállító cég elsődleges kulcs -->
        <xs:key name="SzallitoCegKey">
            <xs:annotation>
                <xs:documentation>Szállító cég egyediség
biztosítása</xs:documentation>
            </xs:annotation>
            <xs:selector xpath="SzallitoCeg"/>
            <xs:field xpath="@sz_k"/>
        </xs:key>

        <!-- Dolgozó elsődleges kulcs -->
        <xs:key name="DolgozoKey">
            <xs:annotation>
                <xs:documentation>Dolgozó egyediség
biztosítása</xs:documentation>
            </xs:annotation>
            <xs:selector xpath="Dolgozo"/>
            <xs:field xpath="@Dolgozoid"/>
        </xs:key>

        <!-- Raktár elsődleges kulcs -->
        <xs:key name="RaktarKey">
            <xs:annotation>
                <xs:documentation>Raktár egyediség
biztosítása</xs:documentation>
            </xs:annotation>
            <xs:selector xpath="Raktar"/>
            <xs:field xpath="@Raktarid"/>
        </xs:key>

        <!-- Termék elsődleges kulcs -->
        <xs:key name="TermekKey">
            <xs:annotation>
                <xs:documentation>Termék egyediség
biztosítása</xs:documentation>
            </xs:annotation>
            <xs:selector xpath="Termek"/>
            <xs:field xpath="@Termekid"/>
```

```
        </xs:key>

        <!-- Raktár -> Szállító cég idegen kulcs hivatkozás --
>
        <xs:keyref name="RaktarSzallitoCegRef"
refer="SzallitoCegKey">
            <xs:annotation>
                <xs:documentation>Raktár hivatkozása szállító
cégre</xs:documentation>
            </xs:annotation>
            <xs:selector xpath="Raktar"/>
            <xs:field xpath="@R_k"/>
        </xs:keyref>

        <!-- Raktár -> Dolgozó idegen kulcs hivatkozás -->
        <xs:keyref name="RaktarDolgozoRef" refer="DolgozoKey">
            <xs:annotation>
                <xs:documentation>Raktár hivatkozása felelős
dolgozóra</xs:documentation>
            </xs:annotation>
            <xs:selector xpath="Raktar"/>
            <xs:field xpath="@d_rb"/>
        </xs:keyref>

        <!-- RaktarTermek -> Raktár idegen kulcs hivatkozás --
>
        <xs:keyref name="RaktarTermekRaktarRef"
refer="RaktarKey">
            <xs:annotation>
                <xs:documentation>Raktár-Termék kapcsolat
hivatkozása raktárra</xs:documentation>
            </xs:annotation>
            <xs:selector xpath="RaktarTermek"/>
            <xs:field xpath="@r_rb"/>
        </xs:keyref>

        <!-- RaktarTermek -> Termék idegen kulcs hivatkozás --
>
        <xs:keyref name="RaktarTermekTermekRef"
refer="TermekKey">
            <xs:annotation>
```

```
                <xs:documentation>Raktár-Termék kapcsolat
hivatkozása termékre</xs:documentation>
            </xs:annotation>
            <xs:selector xpath="RaktarTermek"/>
            <xs:field xpath="@t_rb"/>
             </xs:keyref>
```

Ezzel biztosítjuk, hogy a RaktarTermek csak létező Raktárra és Termékre hivatkozhat.

2. feladat

Project name: PHRO7RDOMParse

Package: phro7r.domparse.hu

Class names: (PHRO7RDomRead, PHRO7RDomModify, PHRO7RDomQuery)

A második feladatnál az volt a cél, hogy a korábban elkészített XML fájlt programból is kezelhetővé tegyem a DOM feldolgozási modell segítségével. Ehhez egy önálló Java projektet készítettem, amelyben külön osztályok felelnek az adatok betöltéséért, kiolvasásáért, illetve módosításáért. A folyamat során az XML dokumentumot először parse-oltam, majd a DOM fa elemein keresztül hajtottam végre a szükséges műveleteket. A feladat része volt többféle lekérdezés megvalósítása, valamint olyan módosítások elvégzése, amelyek megfelelnek az adatstruktúra szabályainak. Az eredményeket minden esetben a konzolra írtam ki. A kód kialakításakor törekedtem a logikus felépítésre és a részletes kommentelésre.

2.1 Adatolvasás (DOM)

A feladat célja az XML állomány DOM-mal történő teljes beolvasása és „blokkszerű” kiírása a konzolra, valamint a kimenet fájlba mentése. A megoldás egy külön Java osztályban készült (PHRO7RDomRead.java), amely:

- betölti és normalizálja az XML-t,
- bejárja az egyes elemeket (Beszallito, SzallitoCeg, Dolgozo, Raktar, Termek, RaktarTermek),
- kinyeri a fontos mezőket (összetett elemeket is, pl. Cim, Meret),
- formázott, blokkos megjelenítéssel konzolra ír, és a kimenetet állományba menti.

Lényeges kódrészletek és magyarázat

1. XML beolvasása és normalizálása

```
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
document = builder.parse(new File("PHRO7R_XML.xml"));
document.getDocumentElement().normalize();
```

Létrehozza a DOM fát a PHRO7R_XML.xml alapján, majd a normalize() egységesíti a szövegcsomópontokat.

2. Egyszerű segédfüggvény csomópontértékhez

```
private static String getErtek(Element szulo, String gyermekNev) {
        NodeList lista = szulo.getElementsByTagName(gyermekNev);
        return lista.getLength() > 0 ?
lista.item(0).getTextContent() : "";
}
```

Biztonságos lekérés: ha nincs gyermek, üres stringet ad vissza (nem dob kivételt).

3. Összetett elemek kezelése (Cím / Méret)

```
Element cim = (Element) elem.getElementsByTagName("Cim").item(0);
kiir("   Cím: " + getErtek(cim, "Iranyitoszam") + " " +
     + getErtek(cim, "Varos") + ", " + getErtek(cim, "Utca") + " "
     + getErtek(cim, "Hazszam"));
```

A Cim alatti tagolt mezők (irányítószám, város, utca, házszám) külön-külön olvashatók ki.

4. Blokkszerű kiírás és fájlmentés

```
// Kiírás konzolra és pufferbe
private static void kiir(String szoveg) {
   System.out.println(szoveg);
   kimenet.append(szoveg).append("\n");
}
// Mentés fájlba
private static void mentesFajlba(String fajlnev) {
   try (PrintWriter writer = new PrintWriter(new
FileWriter(fajlnev))) {
            writer.print(kimenet.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
   }
```

A kiírás egyszerre megy a konzolra és egy pufferbe, amelyet a program a végén fájlba ír.

Teljes PHRO7RDomRead.java fájl

2.2 Adat-lekérdezés

Ebben az alfejezetben a PHRO7R_XML.xml állományon hajtok végre DOM alapú lekérdezéseket XPath használata nélkül. A megoldás a PHRO7RDomQuery.java osztályban készült, amely:

- betölti és normalizálja az XML dokumentumot,
- getElementsByTagName() és attribútum-ellenőrzés segítségével iterál az elemeken,
- legalább 4 különböző lekérdezést futtat és az eredményt blokkszerűen kiírja a konzolra.

Lényeges kódrészletek

1. XML betöltése és normalizálása

```
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
document = builder.parse(new File("PHRO7R_XML.xml"));
document.getDocumentElement().normalize();
```

DOM fa létrehozása, majd a szövegcsomópontok egységesítése.

2. Összes raktár neve

```
NodeList raktarak = document.getElementsByTagName("Raktar");
for (int i = 0; i < raktarak.getLength(); i++) {
      Element r = (Element) raktarak.item(i);
      System.out.println("   - " +
      r.getElementsByTagName("Nev").item(0).getTextContent());
}
```

Egyszerű lista-lekérdezés elemnév alapján.

3. Termék keresése ID alapján (pl. T001)

```
NodeList termekek = document.getElementsByTagName("Termek");
for (int i = 0; i < termekek.getLength(); i++) {
      Element t = (Element) termekek.item(i);
      if (t.getAttribute("Termekid").equals("T001")) {
          System.out.println("   Név: " +
t.getElementsByTagName("Nev").item(0).getTextContent());
          System.out.println("   Ár: " +
t.getElementsByTagName("Egysegar").item(0).getTextContent() + "
Ft");
}
```

4. Szűrés összetett elemre (Budapesti raktárak)

```
for (int i = 0; i < raktarak.getLength(); i++) {
    Element r = (Element) raktarak.item(i);
    Element cim = (Element) r.getElementsByTagName("Cim").item(0);
    if
(cim.getElementsByTagName("Varos").item(0).getTextContent().equals("
Budapest")) {
    System.out.println("   - " +
r.getElementsByTagName("Nev").item(0).getTextContent());
    }
}
```

5. Drága termékek (ár > 3000 Ft)

```
for (int i = 0; i < termekek.getLength(); i++) {
     Element t = (Element) termekek.item(i);
     int ar =
Integer.parseInt(t.getElementsByTagName("Egysegar").item(0).getTextC
ontent());
     if (ar > 3000) {
           System.out.println("   - " +
t.getElementsByTagName("Nev").item(0).getTextContent() + " (" + ar +
" Ft)");
      }
}
```

Numerikus feltételek kezelése a DOM-ban.

6. Készlet összesítés raktáranként

```
NodeList kapcsolatok =
document.getElementsByTagName("RaktarTermek");
     for (int i = 0; i < raktarak.getLength(); i++) {
          Element r = (Element) raktarak.item(i);
          String rid = r.getAttribute("Raktarid");
          int ossz = 0;
          for (int j = 0; j < kapcsolatok.getLength(); j++) {
              Element k = (Element) kapcsolatok.item(j);
              if (k.getAttribute("r_rb").equals(rid)) {
                        ossz +=
Integer.parseInt(k.getElementsByTagName("KeszletDb").item(0).getText
Content());
              }
          }
          System.out.println("   " +
r.getElementsByTagName("Nev").item(0).getTextContent() + ": " + ossz
+ " db");
    }
```

Kézi join + csoportos összegzés

7. Szállítócégek listázása azonosítóval

```
NodeList szallitok = document.getElementsByTagName("SzallitoCeg");
      for (int i = 0; i < szallitok.getLength(); i++) {
            Element s = (Element) szallitok.item(i);
                  System.out.println("   - " +
s.getElementsByTagName("Nev").item(0).getTextContent() +
            [" + s.getAttribute("sz_k") + "]");
}
```

2.3 Adatmódosítás (DOM)

Ebben a részben a PHRO7R_XML.xml dokumentum DOM-alapú módosítását valósítottam meg a PHRO7RDomModify.java osztályban. A program több, adatintegritást megőrző műveletet hajt végre: új elemek beszúrása, meglévő elemek/attribútumok módosítása, kapcsolatok törlése és a változások új XML fájlba mentése.

1. Új termék felvétele

```
ujTermek("T003", "Raklapmozgató", "125000", "1200", "800", "200");
```

2. Elem felépítése és beszúrása a gyökér alá

```
Element ujTermek = document.createElement("Termek");
ujTermek.setAttribute("Termekid", id);

ujTermek.appendChild(createElem("Nev", nev));

Element meret = document.createElement("Meret");
meret.appendChild(createElem("Hossz", h));
meret.appendChild(createElem("Szelesseg", sz));
meret.appendChild(createElem("Magassag", m));
ujTermek.appendChild(meret);

ujTermek.appendChild(createElem("Egysegar", ar));

NodeList termekek = document.getElementsByTagName("Termek");
Node utolso = termekek.item(termekek.getLength() - 1);
gyoker.insertBefore(ujTermek, utolso.getNextSibling());
```

Új Termek elem létrehozása XDM szerinti szerkezettel (Meret összetett elem), majd beillesztés a gyökér alá.

3. Termék ármódosítás

```
termekArModositas("T001", "5500");
```

```
private static void termekArModositas(String id, String ujAr) {
        NodeList termekek = document.getElementsByTagName("Termek");
        for (int i = 0; i < termekek.getLength(); i++) {
            Element t = (Element) termekek.item(i);
            if (t.getAttribute("Termekid").equals(id)) {
                Element ar = (Element)
t.getElementsByTagName("Egysegar").item(0);
                String regi = ar.getTextContent();
                ar.setTextContent(ujAr);
                System.out.println("   ✓ " + regi + " -> " + ujAr +
" Ft");
                return;
            }
        }
    }
```

Mezőmódosítás biztonságos megtalálással, régi→új érték naplózása.

4. Raktár–Termék kapcsolat törlése

```
kapcsolatTorles("R002", "T002");
```

```
private static void kapcsolatTorles(String raktarId, String
termekId) {
        NodeList kapcsolatok =
document.getElementsByTagName("RaktarTermek");
        for (int i = 0; i < kapcsolatok.getLength(); i++) {
            Element k = (Element) kapcsolatok.item(i);
            if (k.getAttribute("r_rb").equals(raktarId) &&
k.getAttribute("t_rb").equals(termekId)) {
                k.getParentNode().removeChild(k);
                System.out.println("   ✓ Törölve");
                return;
            }
        }
    }
```

Az N:M kapcsolóelem („kapcsoló entitás”) fizikai törlése a gyökérből.

5. Új dolgozó beszúrása

```
ujDolgozo("D003", "Molnár Gábor", "Műszakvezető",
"molnar@raktar.hu", "+36-70-123-4567");
```

Dolgozo entitás létrehozása, kötelező gyerekekkel és Dolgozoid attribútumma

6. Mentés xml fájlba

```
private static void mentes(String fajlnev) throws
TransformerException {
        TransformerFactory tf = TransformerFactory.newInstance();
        Transformer transformer = tf.newTransformer();
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
        transformer.transform(new DOMSource(document), new
StreamResult(new File(fajlnev)));
        System.out.println("Mentve: " + fajlnev);
    }
```

Teljes PHRO7RDomModify.java fájl