Jegyzőkönyv

Webes adatkezelő környezetek
Féléves feladat

Könyvesbolt
nyilvántartási rendszer

Készítette: **Ferencsik Róbert**
Neptunkód: **BQLOTW**
Dátum: **2025. November**

Miskolc, 2025

Tartalomjegyzék

Bevezetés

Ez a dokumentáció egy könyvesbolt számára készült, XML-alapú adatkezelő rendszer tervezését és megvalósítását mutatja be. A projekt keretében kidolgoztuk a rendszer alapját képező ER- (Entity-Relationship) és XDM- (XML Data Model) adatmodelleket, amelyek logikailag leírják az adatok szerkezetét és kapcsolatait.

A modellek alapján egy konkrét XML-dokumentumot (BQLOTW_XML.xml) és egy azt leíró XML Schema-t (XSD) hoztunk létre. Az XSD séma felel a bevitt adatok helyességéért és a teljes adatbázis konzisztenciájáért.

A dokumentáció második része a Java nyelven, DOM-szabvány segítségével írt feldolgozó funkciókat ismerteti. Ezek a funkciók teszik lehetővé az XML-adatok beolvasását, a bennük való keresést és a meglévő adatok módosítását. A főbb műveletek a következők:

Adatok beolvasása: A teljes XML-tartalom megjelenítése strukturált formában.

Adatok lekérdezése: Négy különböző, releváns keresési művelet végrehajtása.

Adatok módosítása: Négy különböző, adatmegőrző módosítás elvégzése és az eredmény kiírása.

A dokumentum elkészítése során követtük a formai előírásokat, beleértve a címszintek használatát, a sorkizárt igazítást, az ábrák és kódrészletek megfelelő jelölését és hivatkozását.

1. Adatmodellezés és XML-alapú adatábrázolás

A feladat első részében a könyvesbolt adatbázisának logikai és fizikai struktúráját terveztük meg. A folyamat az absztrakt ER-modelltől haladt a konkrét, adatokkal feltöltött XML-dokumentumig és a hozzá tartozó sémaleírásig.

1.1 Az adatbázis ER-modelljének tervezése

A tervezés első lépése egy konceptuális ER-modell (Egyed-Kapcsolat modell) létrehozása volt, amely a könyvesbolt rendszerének legfontosabb entitásait és a köztük lévő logikai kapcsolatokat írja le.

A modellben a következő fő egyedek szerepelnek: Vásárló, Rendelés, Könyv, Szerző és Kiadó. A kapcsolatok a valós működést tükrözik: egy Vásárló leadhat több Rendelést (1:N), egy Rendelés több Könyvet is tartalmazhat, és egy Könyvet több Rendelésben is megvehetnek (M:N). A Könyv és Szerző között szintén M:N kapcsolat van. A modell tartalmaz egyszerű (pl. ár), összetett (pl. cím) és többértékű (pl. műfaj) attribútumokat, valamint elsődleges kulcsokat minden egyedhez.

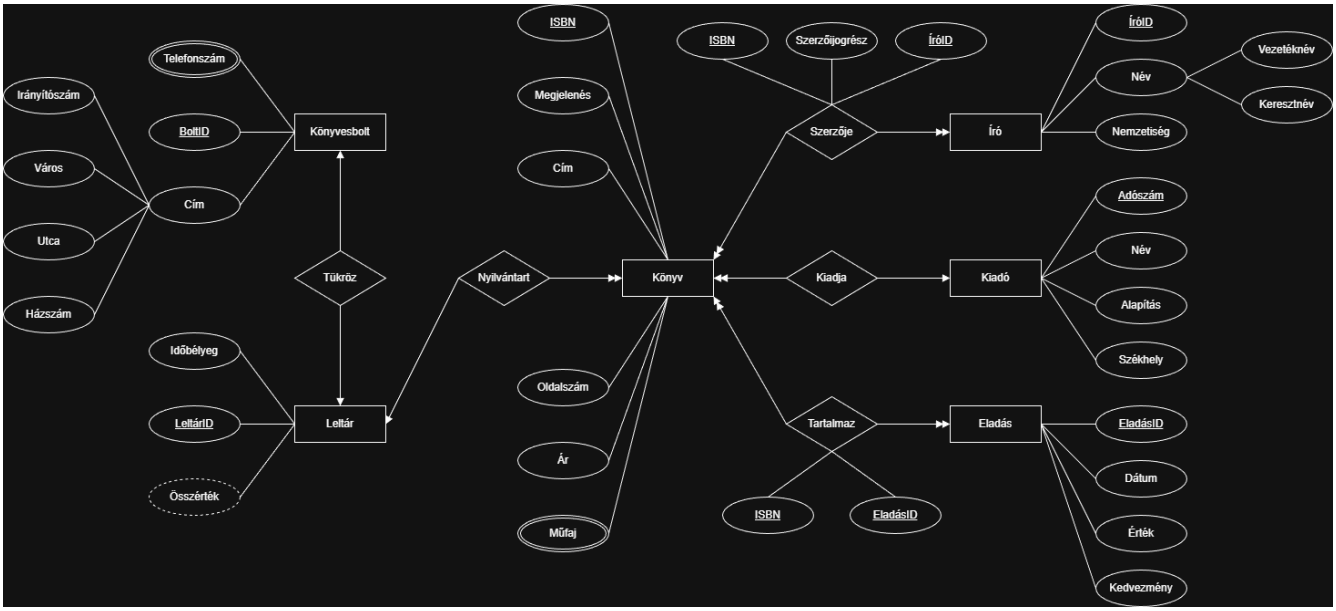

Figure 1: A szabványis jelölésrendszerrel elkészített ER-diagram

1.2 Az adatbázis konvertálása XDM-modellre

Az ER-modell alapján egy hierarchikus XDM (XML Data Model) modellt hoztunk létre, amely már az XML-dokumentum faszerkezetét vázolja fel. A konverzió során a relációs struktúrát egy gyökér-elem köré szerveztük.

Gyökérelemnek a <könyvesboltok> mesterslges gyökér elemet választottuk. Alatta kapnak helyet a főbb entitásokat tartalmazó gyűjtőelemek: <könyvesbolt>, <leltár>, <könyv>, <író>, <kiadó> és eladás. Az egyedek közötti kapcsolatokat az XML-ben azonosítók (ID) segítségével, referenciákkal képeztük le, elkerülve az adatok redundáns ismétlését. Ez a megközelítés egy tisztább és könnyebben kezelhető hierarchiát eredményez. Illetve mesterséges elemek is bevezetésre kerültek a több-több kapcsolatok megvalósítására.

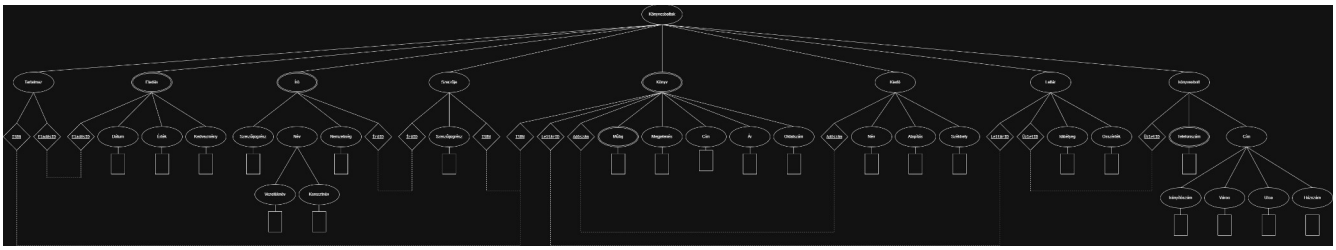

Figure 2: Az XDM diagram

1.3 XML-dokumentum készítése az XDM-modell alapján

Az XDM-modell alapján elkészítettük a konkrét, adatokat tartalmazó BQLOTW_XML.xml dokumentumot. A fájl valósághű példaadatokkal lett feltöltve, hogy a rendszer működése tesztelhető legyen. A specifikációnak megfelelően minden többször előforduló elemből megfelelő mennyiségű példány szerepel a dokumentumban.

A kapcsolatok leképezése ID-referenciákkal történik. Az alábbi kódrészlet egy rendelést mutat be, ahol a vasarloID a vevőre, a konyvID pedig a megvásárolt könyvre hivatkozik.

```
<rendeles rendelesID="R001" vasarloID="V001">
  <datum>2025-10-15</datum>
  <rendeltKonyv>
    <tetel sorszam="1" konyvID="K001"/>
    <tetel sorszam="2" konyvID="K003"/>
  </rendeltKonyv>
</rendeles>
```

1.4 XML Schema készítése az XML-dokumentumhoz

Az XML-dokumentum szerkezeti helyességének és adatintegritásának biztosítására elkészítettük a XMLSchemaBQLOTW.XSD nevű XML Schema leírást. A séma formális szabályokat definiál a dokumentum felépítésére vonatkozóan.

A megvalósítás során több speciális XSD-eszközt is alkalmaztunk:

Saját típusok: Az egyedeknek saját komplex típusokat (complexType) hoztunk létre, ami növeli a séma modularitását és újrafelhasználhatóságát.

Integritási szabályok: A key és keyref megszorításokkal biztosítottuk az elsődleges- és idegenkulcs kapcsolatokat. Például a konyvID kulcsként van definiálva a könyvek listájában, és a rendelések keyref segítségével hivatkoznak rá.

Az alábbi kódrészlet a konyvID kulcs és a rá mutató hivatkozás definícióját szemlélteti:

```
<xs:key name="könyvKulcs">
        <xs:selector xpath="könyv"/>
        <xs:field xpath="@isbn"/>
</xs:key>

 <xs:keyref refer="kiadóKulcs" name="kiadókönyvIKulcs">
         <xs:selector xpath="könyv"/>
         <xs:field xpath="@adószám"/>
 </xs:keyref>
```

2. XML-dokumentum feldolgozása DOM-mal Java nyelven

A projekt második részében a BQLOTW_XML.xml dokumentum Java nyelven történő feldolgozása valósult meg a DOM (Document Object Model) API segítségével. A DOM az XML-dokumentumot egy fa-struktúrájú objektumként tölti be a memóriába, amelyen keresztül az adatok bejárhatók, lekérdezhetők és módosíthatók.

A feldolgozó logika a BQLOTWDOMParse projektben, a bqlotw.domparse.hu csomagban található, és három fő osztályra bontható: BQLOTWDomRead, BQLOTWDomQuery és BQLOTWDomModify.

2.1 Adatok beolvasása és strukturált kiírása

Az adatok beolvasását és egy ember által olvasható, formázott kimenet generálását a BQLOTWDomRead.java osztály végzi.

A program nem egyszerűen csak kiírja az XML-fa tartalmát, hanem először felépít segéd-adatstruktúrákat, hogy az összetartozó adatokat hatékonyan tudja összekapcsolni. Ezt követően egy formázott kimenetet generál a könyvesbolt, a készlet és az eladások adatairól, amelyet egyszerre ír ki a konzolra és ment el a BQLOTWDomRead_output.txt fájlba.

Az alábbi kódrészlet a könyvek és a hozzájuk tartozó, már előfeldolgozott szerzői nevek kiírását mutatja be:

```
printlnBoth(pw, "\nInventory (Books and Authors)");
NodeList konyvek = doc.getElementsByTagName("könyv");
for (int i = 0; i < konyvek.getLength(); i++) {
    Element e = (Element) konyvek.item(i);
    String isbn = e.getAttribute("isbn");

    String authorId = authorIdsByIsbn.get(isbn);
    String authorName = authorNames.getOrDefault(authorId, "Unknown Author");

    printlnBoth(pw, "\n  Book: " + getText(e, "cím") + " (ISBN: " + isbn + ")");
    printlnBoth(pw, "    Author: " + authorName);
    printMultiTag(pw, e, "műfaj", "    Genre: ");
    printlnBoth(pw, "    Price: " + getText(e, "ár"));
    printlnBoth(pw, "    Stock: " + getText(e, "készlet"));
}
```

2.2 Adatok lekérdezése XPath segítségével

A BQLOTWDomQuery.java osztály specifikus adatok kinyerését valósítja meg az XML-dokumentumból. A program a javax.xml.xpath csomagot használja, amely lehetővé teszi, hogy komplex lekérdezéseket is tömören, XPath kifejezésekkel fogalmazzunk meg.

A program a következő négy lekérdezést hajtja végre:

A 'Magvető Kiadó' által kiadott összes könyv címének lekérdezése.

A 'Fantasy' műfajú könyvek szerzőinek listázása.

A 10-nél kevesebb példánnyal rendelkező könyvek adatainak megjelenítése.

Az 5000-nél nagyobb értékű eladások azonosítóinak és értékének kiírása.

Az alábbi kódrészlet a harmadik lekérdezést (Készlet < 10) mutatja be, ahol egy XPath kifejezés segítségével választjuk ki a megfelelő könyv csomópontokat:

```
startBlock("3. Books with stock < 10");
// Az XPath kifejezés, amely a feltételnek megfelelő könyveket választja ki
String query3 = "//könyv[készlet < 10]";
NodeList lowStockBooks = (NodeList) xpath.compile(query3).evaluate(doc,
XPathConstants.NODESET);
// Az eredmény bejárása és kiírása
for (int i = 0; i < lowStockBooks.getLength(); i++) {
    Element bookElement = (Element) lowStockBooks.item(i);
    System.out.println("- " + getText(bookElement, "cím") + " (Stock: " +
getText(bookElement, "készlet") + ")");
}
endBlock();
```

2.3 Adatok módosítása és mentése

A BQLOTWDomModify.java osztály az XML-dokumentum tartalmának programozott megváltoztatását és az eredmény elmentését demonstrálja. A módosítások a memóriában lévő DOM-fán történnek, majd a Transformer osztály segítségével a teljes módosított dokumentum egy új fájlba, a BQLOTW_XML_modified.xml-be íródik.

A program a következő módosításokat hajtja végre:

Egy teljesen új könyv elem hozzáadása a dokumentumhoz.

Az '1984' című könyv árának frissítése ID (ISBN) alapján.

Egy új író (Frank Herbert) hozzáadása a DOM-fához.Az alábbi részlet az új könyv elem dinamikus létrehozását és a dokumentumhoz való hozzáadását szemlélteti:

```
System.out.println("1. Új könyv hozzáadása...");
Element root = doc.getDocumentElement();
Element newKonyv = doc.createElement("könyv");
newKonyv.setAttribute("isbn", "978-1-23-456789-0");
newKonyv.setAttribute("leltarID", "1");
newKonyv.setAttribute("adoszam", "12345678");

Element cim = doc.createElement("cím");
cim.appendChild(doc.createTextNode("Az XML-feldolgozás művészete"));
newKonyv.appendChild(cim);

Element mufaj = doc.createElement("műfaj");
mufaj.appendChild(doc.createTextNode("Műszaki"));
newKonyv.appendChild(mufaj);

Element ar = doc.createElement("ár");
ar.appendChild(doc.createTextNode("7990.00"));
newKonyv.appendChild(ar);

Element keszlet = doc.createElement("készlet");
keszlet.appendChild(doc.createTextNode("25"));
newKonyv.appendChild(keszlet);

root.appendChild(newKonyv);
System.out.println("   Új könyv hozzáadva.");
```