JEGYZŐKÖNYV

Webes adatkezelő környezetek

Féléves feladat

Kávézólánc nyílvántartó rendszer

Készítette: **Vass Martin**
Neptunkód: **I03URB**
Dátum: **2025. december**

Miskolc, 2025

Tartalomjegyzék

Bevezetés

A feladat célja az XML technológiák alkalmazásának gyakorlati bemutatása egy valós, jól strukturált adatmodellen keresztül. A beadandó témája egy laptop- és telefon szerviz adatainak leírása, modellezése és feldolgozása. A projekt két nagy részre tagolódik: az első részben az adatok modellezése és az XML, illetve XSD állományok elkészítése történik, míg a második részben ezeknek a fájloknak a feldolgozása valósul meg Java nyelven, DOM technológiával.
A feladat célja, hogy a hallgató elsajátítsa az XML dokumentumok szerkezetének megtervezését, validálását, és ezek programozott feldolgozását. A projekt során létrejött fájlok (ER modell, XDM modell, XML dokumentum, XSD séma, DOM feldolgozó Java osztályok) együttesen mutatják be az XML technológia teljes életciklusát, az adattervezéstől egészen a feldolgozásig.

A feladat leírása

A projekt egy kávézólánc működésének adatkezelését modellezi. A lánchoz tartozó üzletek különféle termékeket (kávéféléket, süteményeket, üdítőket) értékesítenek, amelyek előállításához többféle alapanyag szükséges. Ezeket az alapanyagokat különböző beszállítók biztosítják, mennyiségi és készletadatokkal együtt. Minden üzlet saját dolgozókkal rendelkezik, akik különböző beosztásban (barista, pénztáros, műszakvezető) dolgoznak, továbbá az egyes üzletek eltérő kínálattal és árazással rendelkezhetnek. A termék–alapanyag és üzlet–termék kapcsolatok N:M jellegűek, így összetett adatstruktúra alakul ki, amely jól modellezi egy valódi kávézólánc működését.

A feladat célja ennek az adatszerkezetnek a megtervezése, először adatmodell szinten (ER és XDM diagram segítségével), majd az adatok rögzítése XML és XSD állományokban, végül pedig Java programmal történő feldolgozása DOM technológia alkalmazásával. A projekt teljes folyamatában a hallgató gyakorolja az XML dokumentumok szerkezeti megtervezését, validálását és programozott kezelését.

Az első feladat során elkészült:

– **az ER modell**, amely bemutatja az üzletek, dolgozók, beszállítók, alapanyagok és termékek közötti kapcsolatokat,
– **az XDM modell**, amely az XML dokumentum hierarchiáját és elemeit írja le,

– **az XML dokumentum**, amely a kávézólánc konkrét adatait tartalmazza,
– **az XSD séma**, amely validálja az XML szerkezetét és adatainak helyességét.

A második feladat során a Java DOM feldolgozással három program készült:

– **I03URBDomRead.java:** adatbeolvasás, az XML dokumentum teljes feldolgozása és blokkonkénti kiírása,
– **I03URBDomQuery.java:** különféle adatlekérdezések megvalósítása XPath nélkül,
– **I03URBDomModify.java:** az XML dokumentum módosítása, új elemek beszúrása és adatok átírása.

1. Kávézólánc

XML és XSD dokumentumok készítése egy kávézólánc működéséhez szükséges adatok alapján.

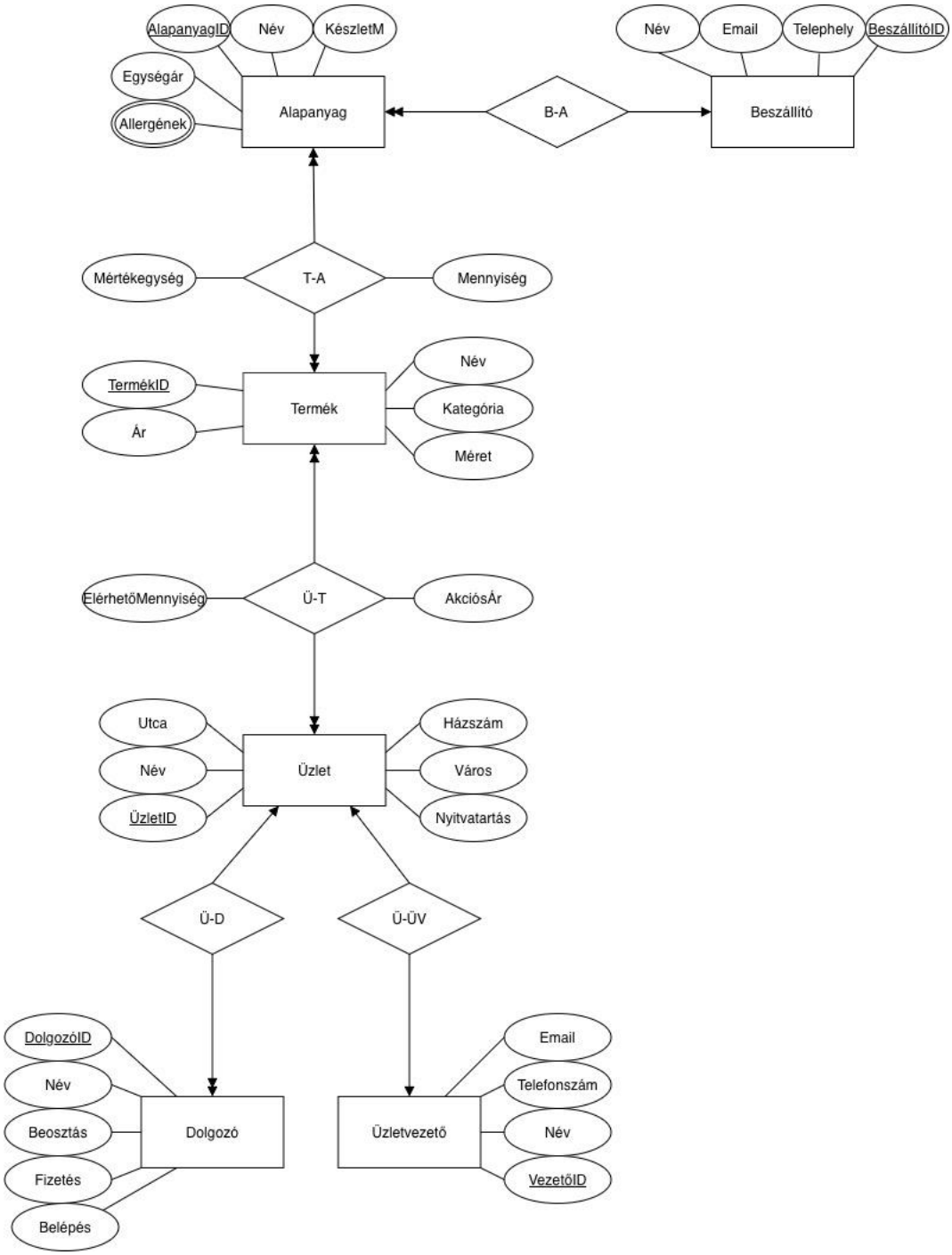
1.1 Az adatbázis ER modell tervezése

A tervezés során egy kávézólánc működését modelleztem, ahol több üzlet tartozik a vállalathoz, minden üzletben saját dolgozók dolgoznak, és az üzletek különféle termékeket értékesítenek. A termékek előállításához különböző alapanyagok szükségesek, amelyeket több beszállító biztosít. Egy termék több alapanyagból épül fel, és egy alapanyag több termékben is felhasználható, így a modell több N:M kapcsolatot is tartalmaz. Mindezek mellett az üzletek saját kínálattal és árazással rendelkezhetnek.

Az ER modell legalább öt entitást tartalmaz: **Üzlet**, **Dolgozó**, **Termék**, **Alapanyag** és **Beszállító**. A kapcsolatok típusa vegyes:

– **Üzlet** – **Dolgozó** között **1:N** kapcsolat,
– **Üzlet** – **Termék** **(Kínálat)** között **N:M** kapcsolat,
– **Termék** – **Alapanyag** között **N:M** kapcsolat,
– **Beszállító** – **Alapanyag** között **1:N** kapcsolat,
– **Üzlet – Üzletvezető** között **1:1** kapcsolat.

A modell tartalmaz **kulcs**, **összetett** és **többértékű tulajdonságokat** is (pl. alapanyag-allergének vagy összetett cím adatok).
A modell draw.io programban készült, és **I03URB_ER.jpg** néven lett elmentve.



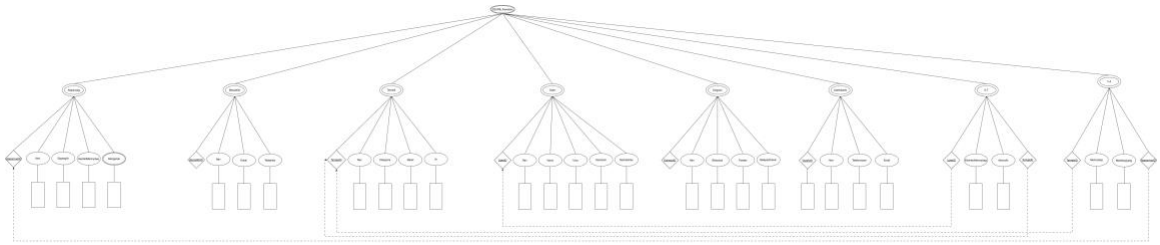
1. ábra: A kávézólánc ER modellje

1.2 Az adatbázis konvertálása XDM modellre

Az XDM modell a korábban elkészített, az **1. ábrán** látható ER modell XML-alapú leképezése, amely hierarchikusan mutatja be az adatokat.
A gyökérelem a **kavezolanc**, ezen belül helyezkednek el az üzletek, minden üzlethez több dolgozó tartozhat, továbbá az üzletek kínálatához különböző termékek kapcsolódnak. A termékeken belül jelennek meg a felhasznált alapanyagok, amelyek több beszállítótól is származhatnak. Az alapanyagok között többértékű tulajdonságok is megjelenhetnek, például az allergének felsorolása.

Az XDM modell az ER modellben megtervezett kapcsolatok hierarchikus formában való megjelenítésére szolgál, az elemek közötti összefüggések és az attribútumok megőrzésével. A modell egyértelműen ábrázolja az elemek közötti kapcsolatokat és az adatszerkezet öröklődési viszonyait, ezzel biztosítva az XML dokumentum szerkezeti alapját.

Az XDM modell **draw.io** programmal készült, és **I03URB_XDM.jpg** néven került elmentésre.



2. ábra: A kávézólánc XDM modellje

1.3 Az XDM modell alapján XML dokumentum készítése

A **2. ábrán** bemutatott XDM modell struktúrája alapján készült el az XML dokumentum, amely a kávézólánc adatait tartalmazza.
A dokumentum gyökéreleme a **Kavezolanc**, amely magában foglalja a beszállítókat, az alapanyagokat, a termékeket, az üzleteket, valamint az üzletvezetők és dolgozók adatait.

Az XML dokumentumban több beszállító (például *CoffeeBeans Kft.* és *SweetBake Bt.*) szerepel, amelyek különböző alapanyagokat biztosítanak a kávézólánc számára. Az alapanyagok egy része allergéneket is tartalmazhat, amelyek többértékű elemként jelennek meg, igazodva az XDM modellben rögzített struktúrához. A termékek többféle alapanyagból épülnek fel, és minden felhasznált alapanyag külön elemként szerepel a megfelelő mennyiség megadásával.

A dokumentumban két üzlet (Central Coffee és Uni Cafe) található, mindkettő saját címmel, nyitvatartással, dolgozókkal és termékkínálattal rendelkezik. A kínálat elemek az üzlet–termék N:M kapcsolatot jelenítik meg, árakkal és elérhető mennyiségekkel együtt. Az üzletvezetők külön szekcióban jelennek meg, és hivatkozással kapcsolódnak a hozzájuk tartozó üzlethez, a modellben meghatározott 1:1 kapcsolatnak megfelelően.

A dokumentum az XDM modellhez igazodva tartalmazza az attribútumokat (pl. **id**, **beszallitoRef**, **alapanyagRef**, **termekRef**, **uzletRef**, **vezetoRef**), amelyek az egyes elemek egyértelmű azonosítását és kapcsolódását biztosítják.
Az XML-ben minden többszörösen előforduló elem (pl. alapanyag, termék, dolgozó) több példányban szerepel, ezzel megjelenítve a modellben meghatározott adatmennyiséget és összefüggéseket.

A fájl neve: **I03URB_XML.xml**.

1.4 Az XML dokumentum alapján XMLSchema készítése

Az XSD séma biztosítja az XML dokumentum formai helyességét és szerkezeti szabályait. A sémában több saját típus is létrejött, például **TelefonszamTipus**, **EmailTipus**, **DatumTipus**, valamint külön típusok készültek a kávézólánc gyakran ismétlődő adatszerkezeteihez.

A komplex típusok (**BeszallitoTipus**, **AlapanyagTipus**, **TermekTipus**, **UzletTipus**, **DolgozoTipus**, **KinalatElemTi**

pusstb.) az egyes XML elemek struktúráját írják le, beleértve az attribútumokat és az egymásba ágyazott elemeket is.
A séma tartalmaz **key**, **keyref** és **ref** elemeket is, amelyek az idegen kulcs kapcsolatok jelölésére szolgálnak (pl. *beszallitoRef*, *alapanyagRef*, *termekRef*, *uzletRef*, *vezetoRef*). A séma validálása sikeres volt, az XML dokumentum minden elemében megfelel a megadott szabályoknak és az XDM modellben meghatározott szerkezeti követelményeknek.
A fájl neve: **I03URB_XMLSchema.xsd**.

2. Kódolás

DOM feldolgozás kávézólánc XML dokumentumhoz

Project name: I03URBDOMParse

Package: i03urb.domparse.hu

Class names: I03URBDomRead, I03URBDomQuery, I03URBDomModify.

2.1 Adatolvasás

Az I03URBDomRead.java fájl beolvassa az XML dokumentumot és a teljes tartalmát blokk formában kiírja a konzolra.
A program a DocumentBuilderFactory és DocumentBuilder osztályokat használja, a feldolgozás DOM alapú.
A kód bejárja az összes ügyfelet, eszközt, megrendelést, szerelést és alkatrészt, és minden adatot kiír hierarchikusan, tagolva.
A beolvasás során a getTextContent segédfüggvény használatával történik a szöveges adatok biztonságos kinyerése.
A futás eredményeként a konzolon a teljes XML tartalma megjelenik strukturált formában.

Az alábbi kódpéldában látható az **alapanyag** elemek kiírása konzolra.

```
private static void printAlapanyagok(Document doc) {
        NodeList nodes = doc.getElementsByTagName("Alapanyag");

        for (int i = 0; i < nodes.getLength(); i++) {
            Element a = (Element) nodes.item(i);

            System.out.println("Alapanyag ID: " + a.getAttribute("id"));
            System.out.println("\tNév:           " + text(a, "Nev"));
            System.out.println("\tEgységár:      " + text(a, "Egysegar"));
            System.out.println("\tSzavatosság:   " + text(a, "SzavatossagNap") + "
nap");

            Element keszlet = (Element)
a.getElementsByTagName("KeszletMennyiseg").item(0);
            System.out.println("\tKészlet:       " + keszlet.getTextContent()
                    + " " + keszlet.getAttribute("mertekegyseg"));

            System.out.print("\tAllergének:   ");
            NodeList allergenek = a.getElementsByTagName("Allergen");
            for (int j = 0; j < allergenek.getLength(); j++) {
                System.out.print(allergenek.item(j).getTextContent());
                if (j < allergenek.getLength() - 1) System.out.print(", ");
            }
            System.out.println();
            System.out.println("----------------------------------------");
        }
    }
```

2.2 Adat-lekérdezés

A **I03URBDomQuery.java** fájl feladata az XML dokumentumban lévő adatok lekérdezése a meghatározott feltételek alapján.A program három lekérdezést hajt végre:

– az összes **üzlet** neve és azonosítója,
– az adott üzlethez tartozó **termékkínálat** listázása (elérhető mennyiség és akciós ár megjelenítésével),
– az összes **alapanyag** és hozzájuk tartozó allergének megjelenítése.

A kód DOM metódusokkal dolgozik, **xPath kifejezéseket nem használ**.
A futás eredményeként a konzolon jelennek meg a kiválasztott adatok rendezett formában.

2.3 Adatmódosítás

A **I03URBDomModify.java** fájl példát mutat a DOM dokumentum módosítására és bővítésére.

A program két műveletet végez:

– **módosítja a T001 azonosítójú termék árát**, például 890 Ft-ról egy új értékre,
– **új dolgozót szúr be az U001 azonosítójú üzlethez**, egyedi azonosítóval és a szükséges adatokkal (név, beosztás, fizetés, belépés dátuma).

Az új elem teljes mértékben illeszkedik az XML struktúrájába, tartalmazza az összes szükséges al-elemet és attribútumot.

A módosított DOM fájl **I03URB_XML_modified.xml** néven kerül mentésre, UTF-8 formátumban, megfelelő behúzásokkal formázva.
A program sikeresen lefutott, és a létrejött XML fájlban megjelenik mind a módosított termékadat, mind pedig az újonnan beszúrt dolgozó.

Az alábbi kód módosítja egy termék árát:

```
private static void modifyTermekAr(Document doc, String termekId, int ujAr) {
        NodeList termekek = doc.getElementsByTagName("Termek");
        for (int i = 0; i < termekek.getLength(); i++) {
            Element t = (Element) termekek.item(i);
            if (!termekId.equals(t.getAttribute("id"))) {
                continue;
            }

            t.getElementsByTagName("Ar").item(0)
                    .setTextContent(Integer.toString(ujAr));

            System.out.println("1) Termék ár módosítva: " + termekId + " → " + ujAr);
            return;
        }
        System.out.println("1) Nem található termék: " + termekId);
}
```

Míg az alábbi kód elmenti a dokumentumot:

```
private static void saveDocument(Document doc, String fileName) throws Exception {
        TransformerFactory tf = TransformerFactory.newInstance();
        Transformer transformer = tf.newTransformer();
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");

        transformer.transform(new DOMSource(doc),
                new StreamResult(new File(fileName)));
}
```

Gyakorlatvezető: Dr. Bednarik László