JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat Virágbolt

Készítette: **Drahos Alinka**

Neptunkód: **EG55OI**

Dátum:2024.11.30.

**Tartalomjegyzék:**

**Bevezetés**

# 1. Virágbolt modell megtervezése

[**1.1**.Az adatbázis ER modell tervezése](#_Virágbolt_modell_megtervezése)

[**1.2**.Az adatbázis konvertálása XDM modellre](#_Az_adatbázis_konvertálása)

[**1.3**. Az XDM modell alapján XML dokumentum készítése](#_Az_XDM_modell)

[**1.4**. AZ XML dokumentum alapján XMLSchema készítése](#_Az_XML_dokumentum)

**2. Lekérdező program írása**

**2.1**.DOMReadEG55OI.java

**2.2** DOMWriteEG55OI.java

**2.3**. DOMQueryEG55OI.java

**2.4**. DOMModifyEG55OI.java

**Bevezetés**

**A feladat leírása:**

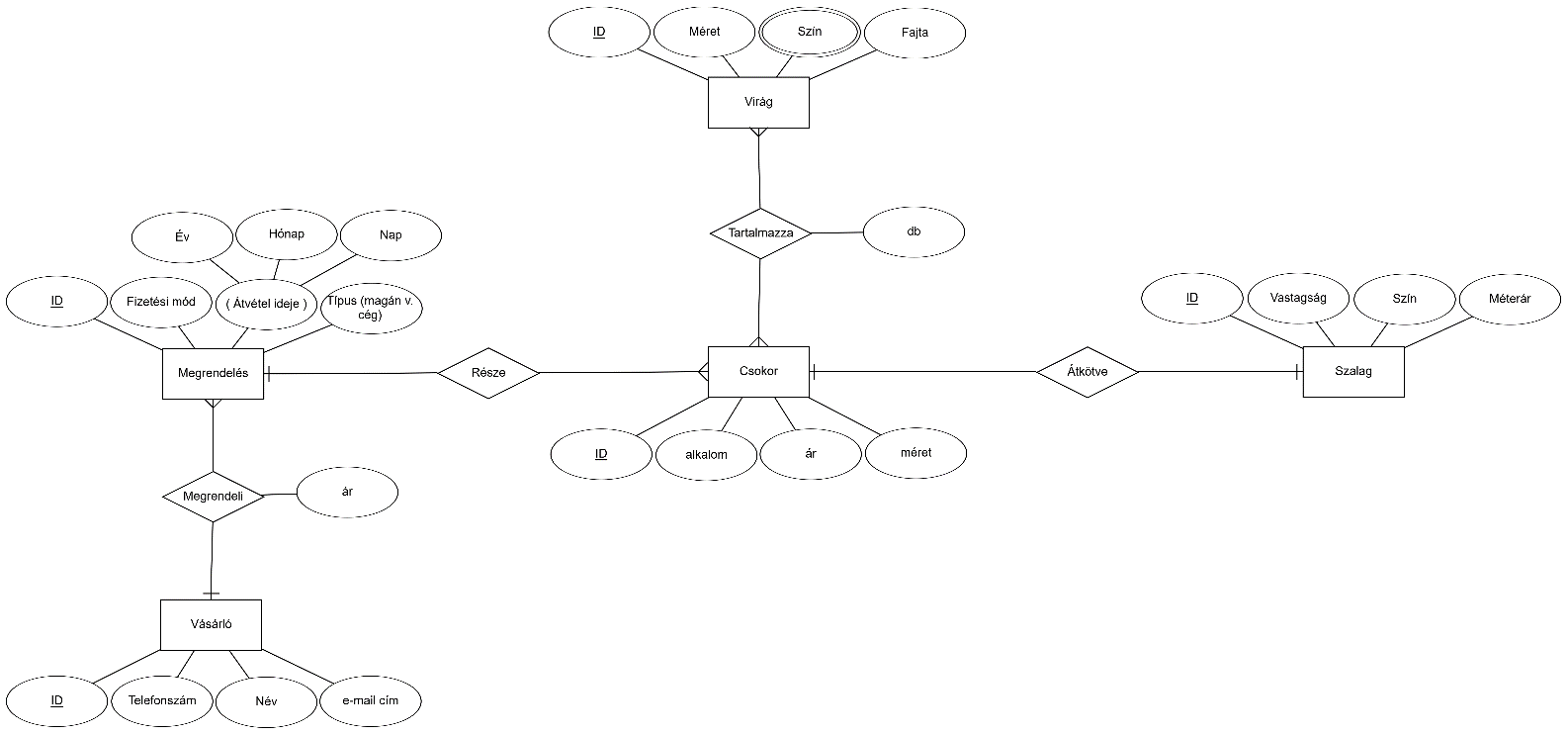
A feladat egy virágbolt megrendeléseit kezelő adatstruktúra létrehozása és kezelése volt, mely leírja egy-egy megrendelés tulajdonságait, mint például a megrendelőt, a hozzátartozó csokrokat, vagy a csokrokhoz tartozó virágokat XML, valamint XSD, és Java nyelven.

A virágboltba vásárlók jönnek be, akik csokrokat rendelnek meg. Egy megrendelésen belül több csokor is szerepelhet (például amikor esküvőre egyben rendelik meg a menyasszonyi és a koszorúslány csokrokat is). Minden csokornak megvan a maga alkalma, (ami irányt mutat a csokor jellege felé) és az ára. Minden csokor több kisebb-nagyobb virágból áll, és egy színes szalaggal van átkötve, melynek van méterára, szélessége és természetesen színe. Egy vásárlónak nyilván tartjuk a nevét, és elérhetőségeit (telefonszámát, e-mailcímét), valamint a megrendeléshez rögzítjük annak árát, és az időpontot, mikorra annak el kell készülnie

# Virágbolt modell megtervezése

* 1. **Az adatbázis ER modell tervezése**

Az ER (Entity-Relationship) modell egyszerűen mutatja be a virágbolt megrendeléseit. Öt egyeddel rendelkezik, melyek rendre a: megrendelés, vásárló, csokor, virág és szalag nevet viselik, melyek mind több tulajdonsággal rendelkeznek.

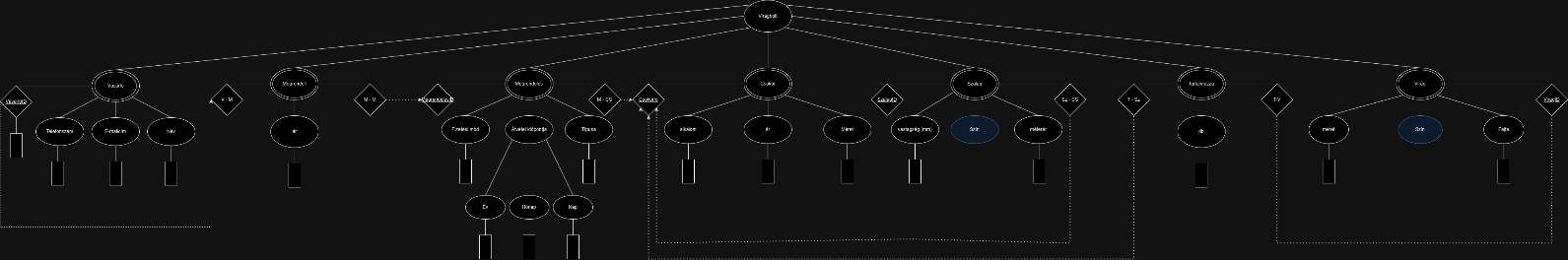
Az ER modell az erdplus oldal által nyújtott eszközökkel készült.

1.ábra: A virágbolt ER modellje

# Az adatbázis konvertálása XDM modellre

Az XDM (Experience Data Model) modellt az ER modell konvertálásával készítettem a draw.io oldal használatával.

Az ER modell entitásai és attribútumai XDM objektumai és mezői, míg kapcsolatai az XDM-ben hierarchikus struktúra illetve referenciamezőként került létrehozásra.

Az objektumokat a Virágbolt objektum fogja össze, ezzel teljessé téve a struktúrát.

2.ábra: Az XDM modell

# Az XDM modell alapján XML dokumentum készítése

Az XDM modellalapján elkészítettem az XML dokumentumot. A dokumentumban három megrendelés szerepel, mindháromhoz különböző vásárló tartozik, és mindhárom megrendelés különböző alkalomra készült (esküvő, diplomaosztó, céges vacsora).

Az XML dokumentum:

<Megrendelések>

<Megrendelés id="1">

<FizetésiMód>Bankkártya</FizetésiMód>

<ÁtvételIdőpontja>

<év>2024</év>

<honap>12</honap>

<nap>6</nap>

</ÁtvételIdőpontja>

<Tipus>Magán</Tipus>

<Megrendelő>

<Név>Kovács Anna</Név>

<Email>anna.kovacs@gmail.com</Email>

<Telefon>+36703332211</Telefon>

</Megrendelő>

<!-- Első megrendelés csokrai -->

<Csokor id="1">

<Alkalom>Esküvő</Alkalom>

<Ár>24000</Ár>

<Méret>Közepes</Méret>

<Virág>

<Fajta>Rózsa</Fajta>

<Szín>Vörös</Szín>

<Méret>Közepes</Méret>

<darab>8</darab>

</Virág>

<Virág>

<Fajta>Lizianthusz</Fajta>

<Szín>Fehér</Szín>

<Méret>Kicsi</Méret>

<darab>10</darab>

</Virág>

<Virág>

<Fajta>Szegfű</Fajta>

<Szín>Vörös</Szín>

<Méret>Közepes</Méret>

<darab>3</darab>

</Virág>

<Virág>

<Fajta>Szegfű</Fajta>

<Szín>Fehér</Szín>

<Méret>Közepes</Méret>

<darab>5</darab>

</Virág>

<Szalag>

<Szín>Arany</Szín>

<Vastagság>2 cm</Vastagság>

<Méterár>500</Méterár>

</Szalag>

</Csokor>

<Csokor id="2">

<Alkalom>Koszorúslány csokor</Alkalom>

<Ár>16000</Ár>

<Méret>Kis</Méret>

<Virág>

<Fajta>Lizianthusz</Fajta>

<Szín>Fehér</Szín>

<Méret>Kicsi</Méret>

<darab>5</darab>

</Virág>

<Virág>

<Fajta>Szegfű</Fajta>

<Szín>Vörös</Szín>

<Méret>Kicsi</Méret>

<darab>2</darab>

</Virág>

<Virág>

<Fajta>Szegfű</Fajta>

<Szín>Fehér</Szín>

<Méret>Kicsi</Méret>

<darab>4</darab>

</Virág>

<Szalag>

<Szín>Piros</Szín>

<Vastagság>2 cm</Vastagság>

<Méterár>450</Méterár>

</Szalag>

</Csokor>

<Csokor id="3">

<Alkalom>Koszorúslány csokor</Alkalom>

<Ár>16000</Ár>

<Méret>Kis</Méret>

<Virág>

<Fajta>Lizianthusz</Fajta>

<Szín>Fehér</Szín>

<Méret>Kicsi</Méret>

<darab>5</darab>

</Virág>

<Virág>

<Fajta>Szegfű</Fajta>

<Szín>Vörös</Szín>

<Méret>Kicsi</Méret>

<darab>4</darab>

</Virág>

<Virág>

<Fajta>Szegfű</Fajta>

<Szín>Fehér</Szín>

<Méret>Kicsi</Méret>

<darab>2</darab>

</Virág>

<Szalag>

<Szín>Piros</Szín>

<Vastagság>2 cm</Vastagság>

<Méterár>450</Méterár>

</Szalag>

</Csokor>

</Megrendelés>

</Megrendelések>

Az XML dokumentumban szerepel még másik két hasonló megrendelés is.

# Az XML dokumentum alapján XMLSchema készítése:

# Az XML dokumentum után az XMLSchema létrehozása következett. A sémában sajáttípusokat és speciális elemeket definiáltam, mint például a megrendelésTípus, FizetésiMódTípus vagy VirágTípus.

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

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

<!-- Gyökérelem -->

<xs:element name="Megrendelések">

<xs:complexType>

<xs:sequence>

<xs:element name="Megrendelés" type="MegrendelésTípus" maxOccurs="unbounded"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- Megrendelés típusa -->

<xs:complexType name="MegrendelésTípus">

<xs:sequence>

<xs:element name="FizetésiMód" type="FizetésiMódTípus"/>

<xs:element name="ÁtvételIdőpontja" type="IdőpontTípus"/>

<xs:element name="Tipus" type="xs:string"/>

<xs:element name="Megrendelő" type="MegrendelőTípus"/>

<xs:element name="Csokor" type="CsokorTípus" maxOccurs="unbounded"/>

</xs:sequence>

<xs:attribute name="id" type="xs:positiveInteger" use="required"/>

</xs:complexType>

<!-- Fizetési mód típusa -->

<xs:simpleType name="FizetésiMódTípus">

<xs:restriction base="xs:string">

<xs:enumeration value="Bankkártya"/>

<xs:enumeration value="Készpénz"/>

<xs:enumeration value="Átutalás"/>

</xs:restriction>

</xs:simpleType>

<!-- Időpont típusa -->

<xs:complexType name="IdőpontTípus">

<xs:sequence>

<xs:element name="év" type="xs:gYear"/>

<xs:element name="honap" type="xs:gMonth"/>

<xs:element name="nap" type="xs:positiveInteger"/>

</xs:sequence>

</xs:complexType>

<!-- Megrendelő típusa -->

<xs:complexType name="MegrendelőTípus">

<xs:sequence>

<xs:element name="Név" type="xs:string"/>

<xs:element name="Email" type="xs:string"/>

<xs:element name="Telefon" type="TelefonTípus"/>

</xs:sequence>

</xs:complexType>

<!-- Telefon típusa -->

<xs:simpleType name="TelefonTípus">

<xs:restriction base="xs:string">

<xs:pattern value="\+36[0-9]{9}"/>

</xs:restriction>

</xs:simpleType>

<!-- Csokor típusa -->

<xs:complexType name="CsokorTípus">

<xs:sequence>

<xs:element name="Alkalom" type="xs:string"/>

<xs:element name="Ár" type="xs:positiveInteger"/>

<xs:element name="Méret" type="MéretTípus"/>

<xs:element name="Virág" type="VirágTípus" maxOccurs="unbounded"/>

<xs:element name="Szalag" type="SzalagTípus"/>

</xs:sequence>

<xs:attribute name="id" type="xs:positiveInteger" use="required"/>

</xs:complexType>

<!-- Virág típusa -->

<xs:complexType name="VirágTípus">

<xs:sequence>

<xs:element name="Fajta" type="xs:string"/>

<xs:element name="Szín" type="xs:string"/>

<xs:element name="Méret" type="MéretTípus"/>

<xs:element name="darab" type="xs:positiveInteger"/>

</xs:sequence>

</xs:complexType>

<!-- Szalag típusa -->

<xs:complexType name="SzalagTípus">

<xs:sequence>

<xs:element name="Szín" type="xs:string"/>

<xs:element name="Vastagság" type="xs:string"/>

<xs:element name="Méterár" type="xs:positiveInteger"/>

</xs:sequence>

</xs:complexType>

<!-- Méret típusa -->

<xs:simpleType name="MéretTípus">

<xs:restriction base="xs:string">

<xs:enumeration value="Kicsi"/>

<xs:enumeration value="Közepes"/>

<xs:enumeration value="Nagy"/>

</xs:restriction>

</xs:simpleType>

</xs:schema>

**2. Lekérdező program írása**

* 1. **DOMReadEG55OI.java**

A DOMReadEG55OI.java fájl az XML dokumentum beolvasását és strukturált kiíratását hivatott megvalósítani. Elemenként olvassa be az XML kódot, majd egyesével íratja is ki azt.

Az alábbi kódrészletben a csokrok feldolgozását végző kód látszódik:

// Csokrok feldolgozása

NodeList bouquets = order.getElementsByTagName("Csokor");

for (int j = 0; j < bouquets.getLength(); j++) {

Node bouquetNode = bouquets.item(j);

if (bouquetNode.getNodeType() == Node.***ELEMENT\_NODE***) {

Element bouquet = (Element) bouquetNode;

output.append(" Csokor ID: ").append(bouquet.getAttribute("id")).append("\n");

output.append(" Alkalom: ").append(*getElementValue*(bouquet, "Alkalom")).append("\n");

output.append(" Ár: ").append(*getElementValue*(bouquet, "Ár")).append("\n");

output.append(" Méret: ").append(*getElementValue*(bouquet, "Méret")).append("\n");

// Virágok feldolgozása

NodeList flowers = bouquet.getElementsByTagName("Virág");

for (int k = 0; k < flowers.getLength(); k++) {

Element flower = (Element) flowers.item(k);

output.append(" Virág: ").append(*getElementValue*(flower, "Fajta")).append("\n");

output.append(" Szín: ").append(*getElementValue*(flower, "Szín")).append("\n");

output.append(" Méret: ").append(*getElementValue*(flower, "Méret")).append("\n");

output.append(" Darab: ").append(*getElementValue*(flower, "darab")).append("\n");

}

// Szalag adatok feldolgozása

Element ribbon = (Element) bouquet.getElementsByTagName("Szalag").item(0);

output.append(" Szalag:\n");

output.append(" Szín: ").append(*getElementValue*(ribbon, "Szín")).append("\n");

output.append(" Vastagság: ").append(*getElementValue*(ribbon, "Vastagság")).append("\n");

output.append(" Méterár: ").append(*getElementValue*(ribbon, "Méterár")).append("\n");

}

}

**2.2. DOMWriteEG55OI.java**

A DOMWriteEG55OI.java fájl az XML dokumentum kiíratását hivatott elvégezni.

A következő kódrészlet a createBoquet függvényt mutatja be, mely a virágcsokrok, és az azok által tartalmazott virágok feldolgozását végzi.

private static void createBouquet(Document document, Element orderElement, int bouquetId, String[][] bouquetDetails) {

Element bouquetElement = document.createElement("Csokor");

bouquetElement.setAttribute("id", String.*valueOf*(bouquetId));

// Basic Bouquet Details

bouquetElement.appendChild(*createTextElement*(document, "Alkalom", bouquetDetails[0][0]));

bouquetElement.appendChild(*createTextElement*(document, "Ár", bouquetDetails[0][1]));

bouquetElement.appendChild(*createTextElement*(document, "Méret", bouquetDetails[0][2]));

// Flowers

for (int i = 1; i < bouquetDetails.length - 1; i++) {

String[] flower = bouquetDetails[i];

Element flowerElement = document.createElement("Virág");

flowerElement.appendChild(*createTextElement*(document, "Fajta", flower[0]));

for (String color : flower[1].split(",")) {

flowerElement.appendChild(*createTextElement*(document, "Szín", color.trim()));

}

flowerElement.appendChild(*createTextElement*(document, "Méret", flower[3]));

flowerElement.appendChild(*createTextElement*(document, "darab", flower[2]));

bouquetElement.appendChild(flowerElement);

}

// Ribbon

String[] ribbon = bouquetDetails[bouquetDetails.length - 1];

Element ribbonElement = document.createElement("Szalag");

ribbonElement.appendChild(*createTextElement*(document, "Szín", ribbon[0]));

ribbonElement.appendChild(*createTextElement*(document, "Vastagság", ribbon[1]));

ribbonElement.appendChild(*createTextElement*(document, "Méterár", ribbon[2]));

bouquetElement.appendChild(ribbonElement);

orderElement.appendChild(bouquetElement);

}

* 1. **DOMQueryEG55OI.java**

A DOMQueryEG55OI.java osztály feladata az XML dokumentumból való lekérdezés végrehajtása volt xPath metódusok használata nélkül.

Az alábbi kódrészletben az egy megrendeléshez tartozó csokrok számát kérdezi le:

// 2. Lekérdezés: Csokrok darabszáma megrendelésenként

System.***out***.println("\n2. Csokrok darabszáma megrendelésenként:");

for (int i = 0; i < megrendelesek.getLength(); i++) {

Node megrendeles = megrendelesek.item(i);

if (megrendeles.getNodeType() == Node.***ELEMENT\_NODE***) {

Element elem = (Element) megrendeles;

NodeList csokrok = elem.getElementsByTagName("Csokor");

System.***out***.println("Megrendelés ID: " + elem.getAttribute("id") + ", Csokrok száma: " + csokrok.getLength());

}

}

* 1. **DOMModifyEG55OI.java**

A DOMModifyEG55OI.java osztály feladata az XML dokumentum módosítása volt.

Az alábbi kódrészletben a második csokor szalagjának vastagságát módosítja:

// 3. Módosítás: Egy meglévő szalag elem attribútumának módosítása

NodeList ribbons = document.getElementsByTagName("Szalag");

if (ribbons.getLength() > 0) {

Element ribbon = (Element) ribbons.item(1); // Második szalag

Node thicknessNode = ribbon.getElementsByTagName("Vastagság").item(0);

thicknessNode.setTextContent("3 cm");

System.***out***.println("\n3. Módosítás: A második szalag vastagsága '3 cm'-re módosítva.");

}