Jegyzőkönyv

Webes adatkezelő környezetek

Féléves feladat

Autószerviz

Készítette: **Takács Bálint Zétény**

Neptunkód: **GJWXEU**

Dátum: **2025.11.20.**

Miskolc, 2025

Tartalomjegyzék:

Bevezetés

A Webes adatkezelő környezetek tantárgy keretében a féléves beadandó célja, hogy gyakorlati tapasztalatot szerezzek az XML-alapú adatkezelés, az adatmodell-tervezés. A feladat során egy teljes adatkezelő rendszert kellett megtervezni, amely magában foglalja az adatbázis logikai modelljének megalkotását, majd annak XML formátumba történő leképezését. Ezen felül a feladathoz tartozik egy XML séma (XSD) létrehozása, amely biztosítja az adatszerkezet és az adatintegritás formai ellenőrzését.

A projekt során megismerhettem az ER modell tervezésének logikáját, az ER modellről XDM modellre való leképzését, az XDM modell alkalmazását XML-re történő leképezéshez, valamint az XML és XSD dokumentumok összehangolt megalkotását. A feladat végeredménye egy validálható, jól strukturált XML adatbázis, amely megfelel a sémafájlban meghatározott szerkezeti és tartalmi előírásoknak.

A feladat leírása

A beadandó gyakorlati témája egy autószerviz nyilvántartó rendszer megtervezése és implementálása XML technológiák segítségével. Az elkészített rendszer célja, hogy az autószerviz mindennapi működésében előforduló adatok – például ügyfelek, javítások, szerelők és felhasznált alkatrészek – egységes, logikusan strukturált módon kerüljenek nyilvántartásra.

A rendszer logikai alapját az ER modell adja, amely leírja az egyedek és azok kapcsolatát. Az ER modellből a XML Data Model (XDM) segítségével hoztuk létre az XML dokumentum logikai struktúráját, amely pontosan leképezi az eredeti adatbázis szerkezetét. Az XML dokumentum (Gjwxeu_XML.xml) reprezentálja a szerviz összes adatát, beleértve a kapcsolatokhoz tartozó idegen kulcsokat, amelyeket az XSD séma (Gjwxeu_XMLSchema.xsd) ellenőriz és érvényesít.

A rendszerben megjelenő főbb adatelemek:

- **Ügyfelek**, akik rendelkeznek személyes adatokkal (név, cím, telefonszám, életkor) és több autóval is.
- **Autók**, amelyek egy adott ügyfélhez tartoznak, és több javításon is áteshetnek.
- **Javítások**, amelyek egy adott autóhoz kapcsolódnak, meghatározott hibák, időtartam és költség mellett.
- **Szerelők**, akik végzik a javításokat, többféle végzettséggel és feladatkörrel.

- **Alkatrészek**, amelyek felhasználásra kerülnek a különböző javítások során.

A projektben kiemelt figyelmet kapott az adatkapcsolatok megőrzése és az adatkonzisztencia biztosítása. Ennek érdekében a kulcsokat (xs:key) és az idegen kulcsokat (xs:keyref) precízen definiáltuk az XSD sémafájlban. Ez lehetővé teszi, hogy az XML dokumentum automatikusan ellenőrizhető legyen, és minden kapcsolat valós, létező elemre mutasson.

A feladat megoldása során a következő lépések történtek:

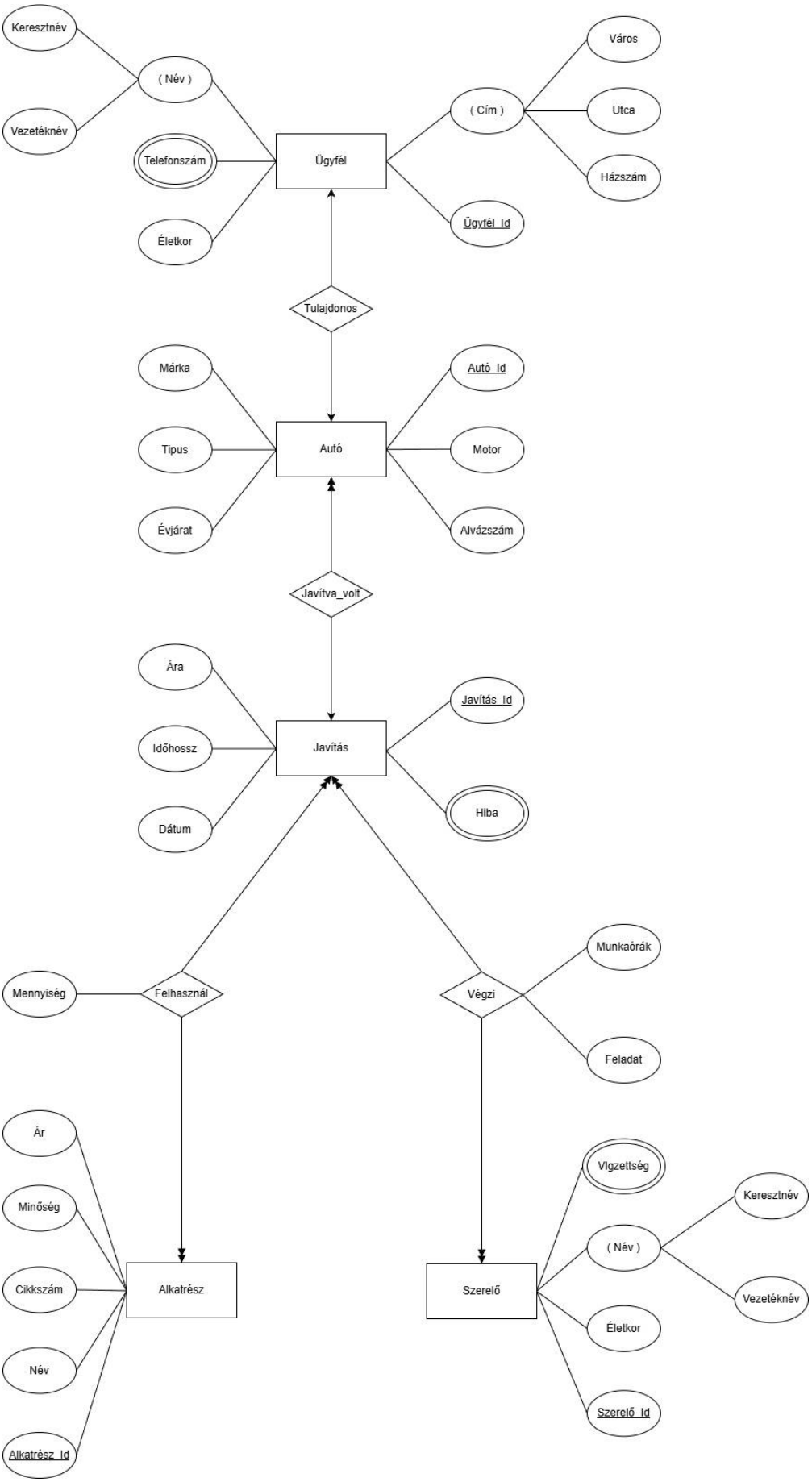
1. Az ER modell megtervezése, az egyedek és kapcsolatok meghatározásával.
2. Az XDM modell elkészítése, amely az ER struktúrát XML-hierarchiába rendezi.
3. Az XML dokumentum létrehozása, a modell alapján kitöltött mintaadatokkal.
4. Az XML séma megírása, amely validálja az XML dokumentum szerkezetét, adattípusait és kapcsolatainak érvényességét.

Ezen folyamat eredményeként egy olyan XML alapú nyilvántartási rendszer született, amely teljes mértékben modellezi az autószerviz adatstruktúráját és működését.

1. Autószerviz

1.1 Az adatbázis ER modell tervezése

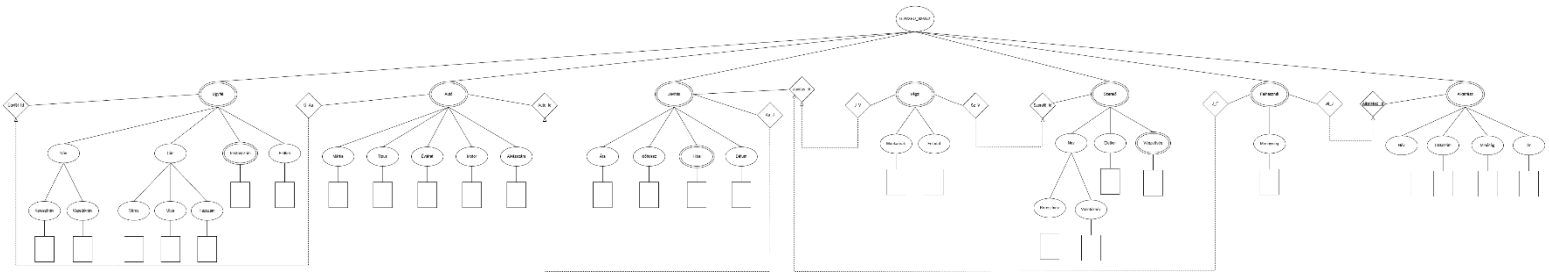
Az ER modell egy autószerviz adatbázisának egyed-kapcsolat modelljét mutatja. A központi elemek az Ügyfél, az Autó, a Javítás, a Szerelő és az Alkatrész entitások. Az Ügyfél több autó tulajdonosa lehet, amit a Tulajdonos kapcsolat jelez. Az Autó entitás a jármű adatait (márka, típus, évjárat, motor, alvázszám) tartalmazza. A Javítás entitás rögzíti a javítási eseményeket, a hozzá tartozó Szerelő és Alkatrész adatokkal együtt. A kapcsolatok (pl. Végzi, Felhasznál) biztosítják, hogy minden javítás összekapcsolódjon a megfelelő szakemberrel és alkatrészekkel. A modell jól áttekinthetően mutatja az autószerviz működéséhez szükséges adatkapcsolatokat.



1. ábra: ER modell

1.2 Az adatbázis konvertálása XDM modellre

A második ábra az előző ER-diagram részletesebb, hierarchikus formája. Ez a logikai adatmodell az entitásokat és attribútumaikat fa-struktúrában ábrázolja, kiemelve az elsődleges kulcsokat és összefüggéseket. Az Ügyfél entitás tartalmazza a személyes és elérhetőségi adatokat, az Autó azonosítóval és műszaki adatokkal rendelkezik, míg a Javítás az ár, hiba és dátum mezőket tartalmazza. A modell jól szemlélteti az adatbázis szerkezetét, és közvetlenül felhasználható relációs adatbázis táblák megtervezéséhez.



2. ábra: XDM modell

1.3 Az XDM modell alapján XML dokumentum készítése

Az XML-dokumentum egy autószerviz adatbázisának adatait tartalmazza strukturált formában. A gyökérelem a <GJWXEU_Szerviz>, amely több főbb elemet foglal magába: ügyfél, autó, javítás, szerelő, alkatrész, valamint az ezeket összekapcsoló végzi és felhasznál elemeket.

Az ügyfél elemek az ügyfelek személyes és elérhetőségi adatait tartalmazzák, beleértve a nevet, címet, telefonszámokat és életkort. Minden ügyfélhez tartozhat egy vagy több autó, amelynek adatai között megtalálható a márka, típus, évjárat, motor és alvázszám.

A javítás elemek az autókhoz kapcsolódó szervizeléseket írják le, rögzítve az árát, időtartamát, a hibákat és a javítás dátumát. A szerelő elemek tartalmazzák a javításokat végző szakemberek adatait, mint a név, életkor és végzettség. Az alkatrész elemek a felhasznált alkatrészeket jellemzik névvel, cikkszámmal, minőséggel és árral.

A végzi kapcsolat köti össze a szerelőket a javításokkal, míg a felhasznál kapcsolat a javítások és alkatrészek közötti összefüggést mutatja. Ezek az elemek további attribútumokat (pl. munkaórák, mennyiség) is tartalmaznak. Összességében a fájl egy jól strukturált XML-leírást ad az autószerviz működéséről, amely alkalmas adatbázis importálásra, XML-séma validálásra és adatkezelési műveletek gyakorlására.

https://hu.wikipedia.org/wiki/XML

```
<!--  Végzik, a javítást és a szerelőt összekötés -->
<végzi j_v="j1" sz_v="sz1">
      <munkaórák>2</munkaórák>
      <feladat>olaj csere</feladat>
</végzi>
<végzi j_v="j2" sz_v="sz1">
      <munkaórák>4</munkaórák>
      <feladat>kuplung csere</feladat>
</végzi>
<végzi j_v="j2" sz_v="sz2">
      <munkaórák>4</munkaórák>
      <feladat>kuplung csere</feladat>
</végzi>
<végzi j_v="j3" sz_v="sz1">
      <munkaórák>2</munkaórák>
      <feladat>olaj csere</feladat>
</végzi>
```

3. ábra: XML-ben egy kapcsolótábla

1.4 Az XML dokumentum alapján XMLSchema készítése

Az XML séma a GJWXEU_Szerviz XML-dokumentum szerkezetét és adattípusait határozza meg. Célja, hogy ellenőrizze az adatok helyességét, felépítését és az elemek közötti kapcsolatok érvényességét. A gyökérelem a <GJWXEU_Szerviz>, amely több összetett típust tartalmaz: ügyfél, autó, javítás, szerelő, alkatrész, valamint az ezeket összekapcsoló végzi és felhasznál elemeket. Az egyes típusokhoz külön komplex és egyszerű típusok vannak definiálva (pl. névTipus, címTipus, életkorTipus, minőségTipus), amelyek leírják az adatok belső szerkezetét és értéktartományát. A séma kulcsokat (xs:key) és idegen kulcsokat (xs:keyref) is tartalmaz, ezzel biztosítva az adatintegritást, például, hogy egy autó mindig létező ügyfélhez tartozzon, vagy hogy minden javítás érvényes autóhoz legyen rendelve. A típusdefiníciók korlátozásokat is tartalmaznak, mint például az életkor (0–150 év között) vagy az évjárat (1900–2100 között). Az enumeration elem a minőség mező lehetséges értékeit (magas, közepes, alacsony) szabályozza.

https://hu.wikipedia.org/wiki/XML-s%C3%A9ma

```
 <!-- Egyszerű típusok -->
<xs:complexType>
    <xs:sequence>
        <xs:element name="ügyfél" type="ügyfélTipus" minOccurs="0" maxOccurs="unbounded"></xs:element>
        <xs:element name="autó" type="autóTipus" minOccurs="0" maxOccurs="unbounded"></xs:element>
        <xs:element name="javítás" type="javításTipus" minOccurs="0" maxOccurs="unbounded"></xs:element>
        <xs:element name="szerelő" type="szerelőTipus" minOccurs="0" maxOccurs="unbounded"></xs:element>
        <xs:element name="alkatrész" type="alkatrészTipus" minOccurs="0" maxOccurs="unbounded"></xs:element>
        <xs:element name="végzi" type="végziTipus" minOccurs="0" maxOccurs="unbounded"></xs:element>
        <xs:element name="felhasznál" type="felhasználTipus" minOccurs="0" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
</xs:complexType>
```

4. ábra: XML Schema egyszerű típusainak a létrehozás

2. Autószerviz DOM

2.1 Adatolvasás

A program célja egy autószerviz adatait tartalmazó XML állomány feldolgozása a DOM (Document Object Model) módszer segítségével. A megoldás Java nyelven készült, a javax.xml.parsers és org.w3c.dom könyvtárak felhasználásával. A program beolvassa az XML fájlt, normalizálja a dokumentumot, majd az egyes elemeket – például ügyfél, autó, javítás, szerelő és alkatrész – feldolgozza. Az adatok tartalma megjelenik a konzolon, valamint egy output.txt nevű fájlban is rögzítésre kerül. A program helyesen kezeli az elemek közötti kapcsolatokat és a többértékű adatokat, ezáltal biztosítva az XML struktúra pontos feldolgozását. A futtatás eredménye egy jól működő, DOM alapú XML olvasó alkalmazás, amely sikeresen megvalósítja a tervezett célokat. https://www.geeksforgeeks.org/java/java-dom-parser-1/

```
//A végzés feldolgozása.
for (int i = 0; i < vegziList.getLength(); i++) {
	Node vegziNode = vegziList.item(i);
	System.out.println("\nAktuális elem: " + vegziNode.getNodeName());
	writer.println("Aktuális elem: " + vegziNode.getNodeName());
	if (vegziNode.getNodeType() == Node.ELEMENT_NODE) {
		Element elem = (Element) vegziNode;
		String j_v = elem.getAttribute("j_v");
		String sz_v = elem.getAttribute("sz_v");
		Node node1 = elem.getElementsByTagName("munkaórák").item(0);
		String munkaorak = node1.getTextContent();
		Node node2 = elem.getElementsByTagName("feladat").item(0);
		String feladat = node2.getTextContent();
		System.out.println("Végzés FK j_v: " + j_v);
		System.out.println("Végzés FK sz_v: " + sz_v);
		System.out.println("Végzés munkaórái: " + munkaorak);
		System.out.println("Végzés feladata: " + feladat);
		writer.println("Végzés FK j_v: " + j_v);
		writer.println("Végzés FK sz_v: " + sz_v);
		writer.println("Végzés munkaórái: " + munkaorak);
		writer.println("Végzés feladata: " + feladat);
	}
}
```

5. ábra: A "végzi" elemeket dolgozza fel és írja ki konzolra, fájlba

2.2 Adat-lekérdezés

A feladat során egy XML állományból kellett különböző adatokat kiolvasnom, illetve lekérdezéseket készítenem XPath használata nélkül. Emiatt a megoldást DOM (Document Object Model) alapú feldolgozásra építettem, amelyben az egész XML dokumentum betöltődik memóriába, majd a szükséges elemeket kiszűrtem.

A lekérdezések:

- Ügyfelek neveinek kilistázása
- 44 évnél idősebb ügyfelek
- Az autók adatainak kilistázása
- Ford autókhoz tartozó javítások hibáinak listázása

A DOM alapú XML feldolgozás során négy különböző lekérdezést készítettem, amelyek mind manuális bejárással, NodeList-ek segítségével valósultak meg. A megoldás nem használ XPath-ot, mégis logikailag ugyanúgy képes összetett szűrésekre és kapcsolatok felismerésére az XML struktúrában.
https://www.w3.org/TR/2001/WD-query-datamodel-20010215/

```
NodeList ugyfelek = document.getElementsByTagName("ügyfél");

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

	Node node = ugyfelek.item(i);

	if (node.getNodeType() == Node.ELEMENT_NODE) {

		Element element = (Element) node;

		String keresztnev = element.getElementsByTagName("keresztnév").item(0).getTextContent();

		String vezeteknev = element.getElementsByTagName("vezetéknév").item(0).getTextContent();

		System.out.println("A(z) " + (i+1) + ". ügyfél vezetélneve: " + vezeteknev + ",
keresztneve: " + keresztnev + ".");

	}

}
```

6. ábra: Az első lekérdezés kódja

2.3 Adatmódosítás

A program célja az XML állomány módosítása a DOM (Document Object Model) technológia alkalmazásával. A megoldás Java nyelven készült, és a javax.xml.parsers, valamint a javax.xml.transform könyvtárak segítségével valósítja meg az adatok beolvasását, feldolgozását és visszaírását. A program betölti az autószerviz adatait tartalmazó XML fájlt, majd bizonyos elemek értékeit megváltoztatja. Az ügyfelek esetében az életkorokat módosítja, például a 45 éves korú ügyfelet 40 évesre, a 30 éveset pedig 35 évesre. Az autók esetében az évjáratokat frissíti, például a 2005-ös modellt 2010-esre, a 2019-es modellt pedig 2020-asra cseréli. A módosításokat követően a program a teljes dokumentum tartalmát kiírja a konzolra, így ellenőrizhető a változtatások eredménye. A program sikeresen demonstrálja a DOM elvű XML-kezelés gyakorlati alkalmazását, különös tekintettel a csomópontok módosítására és az adatok dinamikus frissítésére.

https://www.tutorialspoint.com/java_xml/java_dom_modify_document.htm

```
//Az ügyfeleket ellenőrizük, az életkorukat és módosítjuk.
for (int i = 0; i < eletkorList.getLength(); i++) {
	Node node = eletkorList.item(i);
	if (node.getNodeType() == Node.ELEMENT_NODE) {
		Element element = (Element) node;
		if ("életkor".equals(element.getNodeName())) {
			if ("45".equals(element.getTextContent())) {
				element.setTextContent("40");
			}
			if ("30".equals(element.getTextContent())) {
				element.setTextContent("35");
			}
		}
	}
}
```

7. ábra: Az ügyfelek ellenőrzése majd az élekoruk módosítása