JEGYZŐKÖNYV

Web technológiák 1 gyakorlat

(GE-BProfI)

Féléves feladat

Járműnyilvántartó Rendszer

(Adatkezelés XML környezetben)

Készítette: **Kelemen Ádám**

Neptun Kód: **DBO8MH**

Dátum: **2025. november**

Miskolc, 2025

Tartalomjegyzék

Bevezetés

A **féléves** feladat keretében egy átfogó, XML-alapú **Járműnyilvántartó Rendszer** tervezését és implementálását készítettem el. A rendszer elsődleges feladata egy adminisztrációs adatbázis, amely képes tárolni és kezelni a gépjármű-tulajdonosok adatait, a birtokukban lévő járműveket, a gépkocsikhoz tartozó okmányokat, valamint a járművek szervizelési eseményeit és a karbantartásait végző szervizállomásokat.

A tervezési fázisban egy olyan adatmodellt alkottam meg, amely lefedi a gépjármű-nyilvántartás legfontosabb elemeit. A relációs adatbázis-kezelés elveit követve, de hierarchikus XML környezetre adaptálva, az alábbi kapcsolat-típusokat valósítottam meg:

- **1:N (Egy-a-többhöz) kapcsolat:** Egy **Tulajdonoshoz** több **Jármű** is tartozhat, azonban az adatbázis logikája szerint egy Jármű egy adott időpillanatban kizárólag egy Tulajdonoshoz rendelhető.
- **1:1 (Egy-az-egyhez) kapcsolat:** A balóságnak megfelelve szigorúan 1:1 kapcsolatot definiáltam a **Jármű** és a **Forgalmi engedély** között. Ez biztosítja, hogy minden regisztrált járműhöz pontosan egy érvényes, egyedi azonosítóval ellátott okmány tartozzon, és egy okmány ne lehessen több járműhöz rendelve.
- **M:N (Több-a-többhöz) kapcsolat:** A legösszetettebb kapcsolat a **Járművek** és a **Szervizek** között áll fenn. A valóságban egy autó élete során több különböző szervizben is megfordulhat, és fordítva: egy szervizállomás számtalan autót javít. Ezt a relációt közvetlenül nem lehet hierarchikus fában ábrázolni, ezért egy kapcsoló entitás, a **Szervizelés** (esemény) bevezetésével oldottam meg. Ez a kapcsoló elem nemcsak összeköti a két felet, hanem saját attribútumokkal (Dátum, Leírás) is rendelkezik, így lehetővé teszi a teljes szerviztörténet és a karbantartások naplózását.

Technológiai Megvalósítás A rendszer implementációja során a modern adatkezelési szabványokra támaszkodtam:

1. **Adattárolás (XML):** Az adatok tárolása strukturált, XML (Extensible Markup Language) formátumban történik.
2. **Validáció (XSD):** Az adatbázis integritását és a beviteli szabályok betartását egy szigorú XML Schema Definition (XSD) biztosítja. Itt definiáltam az adattípusokat, a kötelező formátumokat (pl. rendszám regex), valamint a kulcsokat (Key) és a referenciális integritást biztosító idegen kulcsokat (KeyRef).
3. **Feldolgozás (Java DOM):** Az adatok programozott kezelését Java nyelven, a DOM (Document Object Model) technológia segítségével valósítottam meg. Ez lehetővé teszi a teljes XML fa memóriába töltését, a csomópontok közötti navigációt, valamint az adatok olvasását, szűrését és dinamikus módosítását.

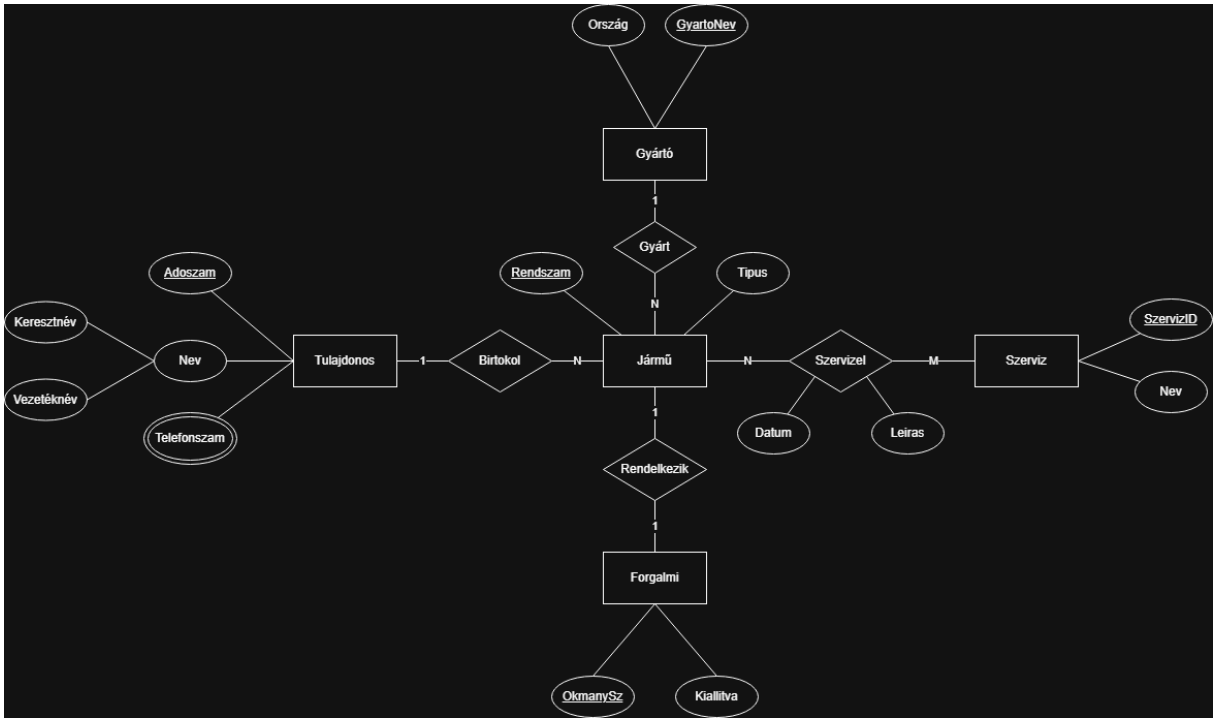
1. Adatmodellezés és XML

1.1 Az ER modell

Az Egyed-Kapcsolat (ER) modellben 5 entitást definiáltam:

1. **Tulajdonos: A jármű birtokosa. Tulajdonságai: Adószám (PK), Név (összetett: Vezetéknév, Keresztnév), Cím, Telefonszám (többértékű).**
2. **Jármű: A rendszer központi eleme. Tulajdonságai: Rendszám (PK), Típus.**
3. **Forgalmi: A jármű okmánya. Tulajdonságai: OkmánySzám (PK), Kiállítva, Érvényesség.**
4. **Szerviz: A javítást végző műhely. Tulajdonságai: SzervizID (PK), Név, Cím.**
5. **Gyártó: A járműveket előállító vállalat. Tulajdonságai: GyártóNév (PK), Ország.**

1:1 (Egy-az-egyhez): Jármű – Forgalmi (Rendelkezik). Egy autónak pontosan egy forgalmija lehet.
1:N (Egy-a-többhöz):
Tulajdonos – Jármű (Birtokol): Egy tulajdonosnak több autója lehet.
Gyártó – Jármű (Gyárt): Egy gyártó több autót készít.
M:N (Több-a-többhöz): Jármű – Szerviz (Szervizel). Egy autó több szervizben is járhat, és egy szerviz több autót is javít. Ezt a kapcsolatot a Szervizel rombusz jelöli, amely saját tulajdonságokkal (Dátum, Leírás) rendelkezik, rögzítve az esemény részleteit.



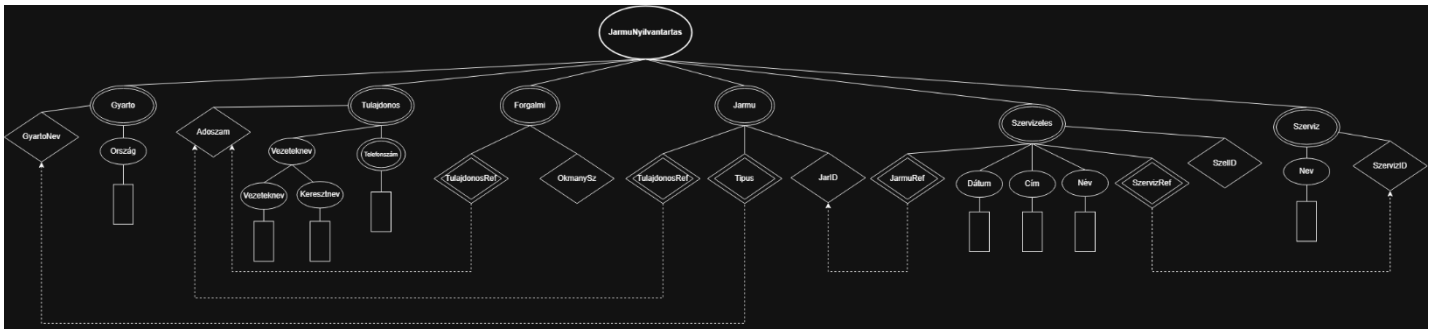
1. ábra: ER Modell

1.2 Az XDM Modell

Az ER modellt egy hierarchikus XDM (XML Data Model) szerkezetre képeztem le, szem előtt tartva a redundancia minimalizálását és a logikus adatstruktúrát. A modell gyökéreleme a JarmuNyilvantartas. A fastruktúra kialakításakor a következő elveket követtem:

1. **Gyártó ág (Törzsadat):** A gyártók adatait (Név, Ország) külön ágon, a gyökér alatt tárolom, mivel ezek független törzsadatok.
2. **Tulajdonos ág (Hierarchikus):** A szülő-gyermek kapcsolatokat egymásba ágyazással valósítottam meg. A Tulajdonos elem tartalmazza a Jármű elemeket, azok pedig a Forgalmi elemet. Ez tükrözi az 1:N és 1:1 kapcsolatokat. A Jármű elem GyartoRef attribútummal hivatkozik vissza a gyártóra.
3. **Szerviz ág (Törzsadat):** A szervizek adatait szintén külön ágon, a gyökér alatt tárolom.
4. **Szervizelés ág (Kapcsoló):** Az M:N kapcsolat feloldására szolgáló Szervizelés entitás a gyökér alatt, külön listában helyezkedik el. A kapcsolatot attribútum-alapú hivatkozásokkal (JarmuRef, SzervizRef) valósítottam meg, amelyek a hierarchiában lévő Jármű és Szerviz kulcsaira mutatnak.

Az ábrán a hivatkozásokat jelölő szaggatott vonalak nem keresztezik egymást, a struktúra átlátható és követi a "fentről-lefelé" hierarchiát.



2. ábra: XDM Modell

1.3 Az XDM modell alapján XML dokumentum készítése

Az XML dokumentumot (*DBO8MH_XML.xml*) az XDM modell alapján hoztam létre. A fájl tartalmazza a konkrét adatokat. Minden entitás-típusból (Tulajdonos, Szerviz, Szervizelés) legalább 3 példányt rögzítettem a demonstráció érdekében. Az azonosítókat (PK) attribútumként, a leíró adatokat elemként tároltam. A kódban kommentekkel jelöltem az egyes logikai blokkokat.

```
<Tulajdonos adoszam="111111-1-11">
   <Nev>
      <Vezeteknev>Kovács</Vezeteknev>
      <Keresztnev>János</Keresztnev>
   </Nev>
   <Telefonszam>+36301234567</Telefonszam>
   <Telefonszam>+36709998877</Telefonszam>

   <Jarmu rendszam="ABC-123">
      <Tipus>Astra</Tipus>
      <GyartoRef>Opel</GyartoRef> </Jarmu>
</Tulajdonos>
```

3. ábra: XML fájl (részlet)

1.4 Az XML dokumentum alapján XMLSchema készítése

Tervezés/megvalósítás rövid leírása: Az XSD séma (DBO8MH_XSD.xsd) definiálja az adatbázis érvényességi szabályait és típusait.

1. **Saját típusok:** Létrehoztam reguláris kifejezésekkel korlátozott egyszerű típusokat (pl. rendszamTipus) és komplex típusokat az összetett elemekhez (pl. nevTipus, gyartoTipus).
2. **Kulcsok (Key):** xs:key elemekkel biztosítottam a Gyártó, Tulajdonos, Jármű és Szerviz egyediségét.
3. **1:1 Kapcsolat:** Az xs:unique megszorítással garantáltam, hogy egy forgalmi engedélyszám (okmanySz) csak egyszer szerepelhessen a rendszerben.
4. **Hivatkozások (KeyRef):** xs:keyref elemekkel biztosítottam a referenciális integritást a Gyártók és a Szervizelések irányába.

```
<xs:complexType name="nevTipus">
   <xs:sequence>
      <xs:element name="Vezeteknev" type="xs:string"/>
      <xs:element name="Keresztnev" type="xs:string"/>
   </xs:sequence>
</xs:complexType>

<xs:unique name="UniqueOkmany">
   <xs:selector xpath="Tulajdonos/Jarmu/Forgalmi"/>
   <xs:field xpath="@okmanySz"/>
</xs:unique>
```

4. ábra: XSD fájl (részlet)

2. Java DOM feldolgozás

Project név: DBO8MHDOMParse

Csomag: hu.domparse.dbo8mh

Osztályok: DBO8MHDOMRead, DBO8MHDOMQuery, DBO8MHDOMModify

2.1. Adatolvasás

A program (`DBO8MHDOMRead.java`) **DocumentBuilderFactory** segítségével tölti be a *DBO8MH_XML.xml* fájlt. A fastruktúra normalizálása után a **getElementsByTagName** metódussal érem el a csomópontokat. A hierarchiát (Tulajdonos → Jármű → Forgalmi) egymásba ágyazott ciklusokkal járom be, attribútumokat és elemtartalmakat olvasva. Az eredményt blokk formátumban írom ki a konzolra, valamint mentem a read_log.txt fájlba.

```
// Ciklus a Tulajdonosok és beágyazott Járműveik bejárására
NodeList ownerList = doc.getElementsByTagName("Tulajdonos");
for (int i = 0; i < ownerList.getLength(); i++) {
    Element elem = (Element) ownerList.item(i);
    String uid = elem.getAttribute("adoszam");
    // ...
    NodeList carList = elem.getElementsByTagName("Jarmu");
    for (int j = 0; j < carList.getLength(); j++) {
       // Beágyazott elemek olvasása
    }
}
```

5. ábra: DBO8MHDOMRead.java (részlet)

2.2. Adat-lekérdezés

A program (`DBO8MHDOMQuery.java`) négy specifikus lekérdezést valósít meg XPath használata nélkül, a DOM fa bejárásával és feltételes vizsgálatokkal (if, equals, contains).

1. **Lekérdezés:** Az összes "Opel" típusú autó listázása rendszámmal együtt.
2. **Lekérdezés:** "Nagy Éva" nevű tulajdonos összes járművének listázása.
3. **Lekérdezés:** Miskolci telephelyű szervizek keresése a Cím mező vizsgálatával.
4. **Lekérdezés:** Egy adott rendszámhoz (ABC-123) tartozó szervizelési naplóbejegyzések kigyűjtése a kapcsoló entitásokból.

```
// 3. Lekérdezés: Miskolci szervizek keresése
NodeList serviceList = doc.getElementsByTagName("Szerviz");
for (int i = 0; i < serviceList.getLength(); i++) {
    Element srv = (Element) serviceList.item(i);
    String address = srv.getElementsByTagName("Cim").item(0).getTextContent();

    if (address.contains("Miskolc")) {
        System.out.println("Találat: " +
srv.getElementsByTagName("Nev").item(0).getTextContent());
    }
}
```

6. ábra: DBO8MHDOMQuery.java (részlet)

2.3. Adatmódosítás

Tervezés/megvalósítás rövid leírása: A program (`DBO8MHDOMModify.java`) a memóriában lévő DOM fán végez CRUD műveleteket.

1. **Szöveg módosítása:** Egy szerviz nevét (setTextContent) írja át.
2. **Elem módosítása:** Egy jármű típusát változtatja meg a fastruktúrában.
3. **Új elem hozzáadása:** Új Szervizeles csomópontot hoz létre createElement metódussal, feltölti gyermekelemekkel, majd az appendChild segítségével a gyökérhez fűzi.
4. **Törlés:** Egy szervizelést töröl az azonosítója (szelID) alapján a removeChild metódussal. A végeredményt a Transformer osztály segítségével írom ki a konzolra ellenőrzésképp.

```
// 4. Módosítás: Elem törlése (SZL002)
NodeList logs = doc.getElementsByTagName("Szervizeles");
for(int i=0; i<logs.getLength(); i++) {
    Element log = (Element) logs.item(i);
    if(log.getAttribute("szelID").equals("SZL002")) {
        log.getParentNode().removeChild(log); // Törlés a szülőből
        System.out.println("\t SZL002 azonosítójú elem törölve.");
        break;
    }
}
```

7. ábra: DBO8MHDOMModify.java (részlet)