JEGYZŐKÖNYV

Webes adatkezelő környezetek

Féléves feladat

Egy Porsche márkakereskedés adatkezelő rendszere

Készítette: Orosz Zalán Zétény

Neptunkód: XPUYJX

Dátum: 2025. november

Miskolc, 2025

Tartalomjegyzék

1. Bevezetés

A modern vállalati rendszerekben az adatok rendszerezése és kezelése kulcsfontosságú, különösen a prémium autókereskedések esetén, ahol egyszerre kell nyilvántartani az autókat, ügyfeleket, értékesítőket és szervizfolyamatokat. A Porsche márkakereskedések adatkezelési igényei sokrétűek: az értékesítési adatok, ügyféladatok és szervizinformációk integrált, könnyen kezelhető formában történő tárolása nélkülözhetetlen a hatékony működéshez.

2. Feladat leírása

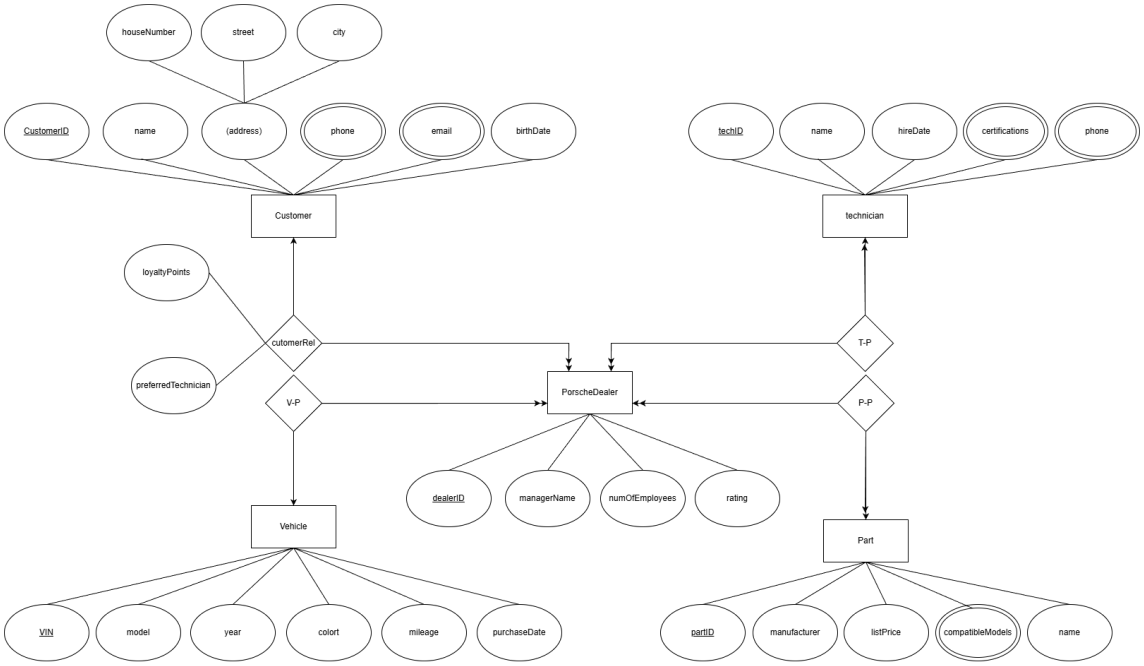
Feladatnak egy Porsche márkakereskedés adatkezelő rendszerét választottam. A féléves feladat során célom az volt, hogy egy ilyen kereskedés adatstruktúráját modellezzem XML és XSD alapú rendszeren keresztül, valamint Java DOM API segítségével bemutassam az adatok feldolgozását, lekérdezését és módosítását.

Ez a munka tehát bemutatja, hogyan lehet egy komplex, valós adatszerkezetet webbarát formában reprezentálni, és hogyan teszi lehetővé a Java-alapú programozás az adatok hatékony kezelését és feldolgozását.

3. Alapvető adatszerkezeti modellek és fileok

3.1. Az adatbázis ER modell tervezése

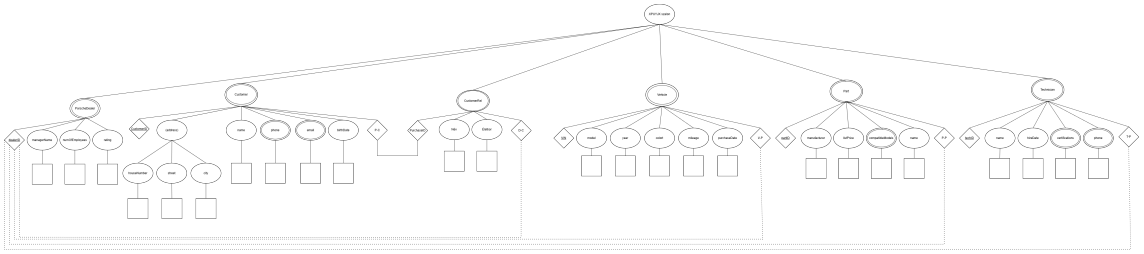
A képen látható ER alapján: a PorscheDealer a központi entitás, minden más egyed idegen kulccsal kapcsolódik hozzá. Vehicle, Customer, Technician és Part egyaránt 1:N kapcsolatban állnak a dealer-rel; Customer és Vehicle között N:M kapcsolat van CustomerRel (rendelés) asszociatív entitással, amely extra adatokat (loyaltyPoints, preferredTechnician) tárol. Attribútumok: kulcsok (VIN, CustomerID, techID, partID), összetett (address $\rightarrow$ city, street, houseNumber), többértékűek (phone, certifications, compatibleModels) és normál mezők (name, model, year, stb.).



1. ábra. A Porsche dealer ER modellje

3.2. Az adatbázis konvertálása XDM modellre

Átalakítottam az ER-modellt XDM-formára: bevezettem egy mesterséges gyökérelemet "XPUYJXszalon", amelynek gyerekei a "PorscheDealer", "Customer", "CustomerRel", "Vehicle", "Part", "Technician". Az XDM-ben minden tulajdonság külön gyerekelem, a kulcsok attribútumokként jelennek meg — a modell hierarchikus felépítésű.



2. ábra. A Porsche dealer XDM modellje

3.3. Az XDM modell alapján XML dokumentum készítése

Az XML struktúrája az XDM szerkezete alapján készült. A gyökérelem "<XPUYJY-Szalon>", ez tartalmazza gyerekelemként az összes többi entitást (pl.: <porscheDealer>,

<Costumer>). Ezeknek saját egyedi azonosító attribútumaik vannak (DealerID, Cid). A kódban található többértékű tulajdonságok több értékkel rendelkeznek.

Listing 1. A Porsche dealer adatszerkezet részlete

```
<PorscheDealer dealerID="D_1">
        <managerName>Gábor Szabó</managerName>
        <numOfEmployees>25</numOfEmployees>
        <rating>4.8</rating>
    </PorscheDealer>

    <Customer Cid="C_1" P-C="PU_1">
        <name>John Doe</name>
        <phone>+36-20-123-4567</phone>
        <email>john.doe@example.com</email>
        <birthDate>1985-04-12</birthDate>
        <address>
            <city>Budapest</city>
            <street>Fő utca</street>
            <houseNumber>10</houseNumber>
        </address>
    </Customer>
```

3.4. Az XML dokumentum alapján XMLSchema készítése

Az XSD az XML dokumentum alapján készült. A komplex típusok elemként és attribútumként jelennek meg. Minden fő elem saját típust kapott, és az összetett elemek külön típusokat kaptak.

Listing 2. Az XSD modell részlete

```
<xs:complexType name="PorscheDealerType">
        <xs:sequence>
            <xs:element name="managerName" type="xs:string"/>
            <xs:element name="numOfEmployees" type="xs:int"/>
            <xs:element name="rating" type="xs:decimal"/>
        </xs:sequence>
        <xs:attribute name="dealerID" type="xs:ID" use="required
           "/>
    </xs:complexType>
```

4. Java adatszerkezet megvalósítása DOM API segítségével

4.1. Adatolvasás

A program célja, hogy XML dokumentumból Java objektum-szerűen kinyerje és megjelenítse az adatokat. A feldolgozás az org.w3c.dom könyvtár DOM API-ján keresztül történik.

Listing 3. Javaban dokumentum létrehozás

```
File xmlFile = new File("XPUYJX_XML.xml");
DocumentBuilderFactory factory = DocumentBuilderFactory.
   newInstance();
DocumentBuilder dBuilder = factory.newDocumentBuilder();
Document doc = dBuilder.parse(xmlFile);
doc.getDocumentElement().normalize();
```

Listing 4. Javaban dealer objektum létrehozás

```
 NodeList dealers = doc.getElementsByTagName("PorscheDealer");
        System.out.println("\nPorscheDealerek:");
        for (int i = 0; i < dealers.getLength(); i++) {
            Element pd = (Element) dealers.item(i);
            System.out.println("Dealer ID: " + pd.getAttribute("
               dealerID"));
            System.out.println("  Manager: " + textOf(pd, "
               managerName"));
            System.out.println("  Employees: " + textOf(pd, "
               numOfEmployees"));
            System.out.println("  Rating: " + textOf(pd, "rating
               "));
        }
}
```

A programrészlet feladata, hogy feldolgozza a dealer elemeit és azok attribútumait, majd kiírja azokat.

4.2. Adat-lekérdezés

A Read-del ellentétben a Query nem csupán adatkiírásra szolgál, hanem emberileg feldolgozható és olvasható formában jeleníti meg az adatokat.

Listing 5. Javaban rendezett adatkiírás

```
NodeList vehicles = doc.getElementsByTagName("Vehicle");
        System.out.println("\nJárművek:");
        for (int i = 0; i < vehicles.getLength(); i++) {
            Element v = (Element) vehicles.item(i);
            System.out.println("  [VIN=" + v.getAttribute("VIN")
                + ", V-P=" + v.getAttribute("V-P") + "] " +
                    textOf(v, "model") + " (" + textOf(v, "year"
                       ) + ") - " + textOf(v, "color") +
                    ", mileage: " + textOf(v, "mileage") + ",
                       purchased: " + textOf(v, "purchaseDate"))
                       ;
        }
```

Ez a programrészlet kinyeri az attribútumokat és az elemeket, majd tömör, emberek számára olvasható formátumban írja ki azokat.

4.3. Adatmódosítás

Listing 6. Javaban adat változtatás

```
Element custC1 = findElementByAttribute(doc.
   getElementsByTagName("Customer"), "Cid", "C_1");
            if (custC1 != null) {
                Element phone = firstElement(custC1, "phone");
                if (phone != null) {
                    phone.setTextContent("+36-20-999-9999");
                    System.out.println("Customer C_1 phone módos
                       ítva: " + phone.getTextContent());
                }
```

Ez a programrészlet megkeresi a Customer elemek között a C1 azonosítójú vevőt (Customer) az Cid attribútum alapján. Ha a keresett elem megtalálható, akkor belül megkeresi a phone al-elemet, és ha az létezik, módosítja annak tartalmát a megadott telefonszámra (+36-20-999-9999). Végül kiírja a konzolra, hogy a C1 azonosítójú vevő telefonszáma sikeresen módosítva lett, és megjeleníti az új értéket.

Listing 7. Javaban file-ba mentés

```
TransformerFactory tf = TransformerFactory.newInstance();
            Transformer transformer = tf.newTransformer();
            transformer.setOutputProperty(OutputKeys.ENCODING, "
               UTF-8");
            transformer.setOutputProperty(OutputKeys.INDENT, "
               yes");
            transformer.setOutputProperty("{http://xml.apache.
               org/xslt}indent-amount", "2");

            DOMSource source = new DOMSource(doc);
            transformer.transform(source, new StreamResult(
               System.out));

            File outFile = new File("XPUYJX_XML_modified.xml");
            transformer.transform(source, new StreamResult(
               outFile));
            System.out.println("\nMentve: " + outFile.
               getAbsolutePath());
```

Ez a kódrészlet létrehoz egy Transformer objektumot, amely a DOM XML dokumentumot UTF-8 kódolással, behúzásokkal formázva írja ki a konzolra, majd ugyanazt az XML-t elmenti egy fájlba (XPUYJXXMLmodified.xml). A program így biztosítja, hogy a módosított XML olvasható és rendezett formában kerüljön tárolásra.