JEGYZŐKÖNYV

Adatkezelés XML-ben

Féléves feladat

Új munkagépek rendszerezése

Jobbágy Dániel

G0P9OJ

mérnökinformatikus hallgató

2024.12.08.

Tartalomjegyzék

Témakör leírása

Az általam választott témakör a munkagépek és azok tulajdonságainak modellezése. A téma célja egy olyan adatbázismodell kidolgozása, amely részletesen leírja az újonnan legyártott munkagépekkel kapcsolatos alapvető információkat, a hozzájuk tartozó tulajdonosokkal, gyártási adatokkal, motorokkal és karosszériákkal együtt. A modellben 5 különböző egyed szerepel: Munkagép, Tulajdonos, Motor, Karosszéria és Gyártás. Ezek mind reguláris elemek, vagyis az adatbázisban nincs gyengén típusos komponens. A különféle egyedek közötti kapcsolatok elemzése során megkülönböztethetünk identifikáló (azonosítást biztosító) és nem identifikáló kapcsolatokat, de jelen modellben az összes kapcsolat nem identifikáló. Ez azt jelenti, hogy az egyes egyedek egymáshoz való kapcsolódása nem jár az egyik entitás teljes azonosításának függőségével.

Egyedek, attribútumok, kapcsolatok

Az egyedek közötti kapcsolatok típusai a következők:

- Motor-Munkagép kapcsolat: HAS_A, egy a többhöz kapcsolat, ahol egy motor több járműhöz is tartozhat.
- Gyártó-Munkagép kapcsolat: HAS_A, egy a többhöz kapcsolat, ahol egy gyártó több gépet is gyárthat.
- Karosszéria-Munkagép kapcsolat: HAS_A, egy az egyhez kapcsolat, amely biztosítja, hogy minden munkagép egyedi karosszériával rendelkezzen.
- Tulajdonos-Munkagép kapcsolat: HAS_A, egy a többhöz kapcsolat, amely lehetővé teszi, hogy egy tulajdonos több munkagépet is birtokolhasson.

Egyedek és attribútumaik:

<u>Munkagép</u>

- Rendszám (elsődleges kulcs): Egyedi azonosító, amely két betű, két betű és három szám kombinációjából áll.
- Alváz száma (idegen kulcs): Hivatkozás a Karosszéria egyedhez.
- Tulaj jogsi (idegen kulcs): Kapcsolat a Tulajdonos jogosítványszámával.
- Motor száma (idegen kulcs): Kapcsolat a Motor egyedhez.
- Gyártás sorszám (idegen kulcs): Hivatkozás a Gyártás egyedre.
- Kor: A gyártási évből számított érték.
- Biztosító azonosító: Opcionális attribútum, minimális előfordulás: 0.

<u>Tulajdonos</u>

- Jogosítvány száma (elsődleges kulcs): Egyedi azonosító, két betűből és hat számból áll.
- Név (összetett): Vezetéknév és keresztnév külön mezőkben.

Motor

- Motorszám (elsődleges kulcs): Egyedi azonosító, amely 14 karakter hosszú és betűk vagy számok kombinációjából áll.
- Üzemanyag: Meghatározza a motor típusát; értékei: benzin, dízel, hibrid, elektromos.
- Lóerő: A motor teljesítménye lóerőben kifejezve.
- Nyomaték: A motor forgatónyomatéka.

Karosszéria

- Alvázszám (elsődleges kulcs): Egyedi azonosító, amely 17 karakter hosszú és betűket vagy számokat tartalmaz.
- Felület: Az autó karosszériájának állapota; lehet matt, részleges vagy teljes fényezés.
- Szín: Többértékű attribútum, amely a munkagép színeit írja le.
- Matricák: Boolean (van vagy nincs).

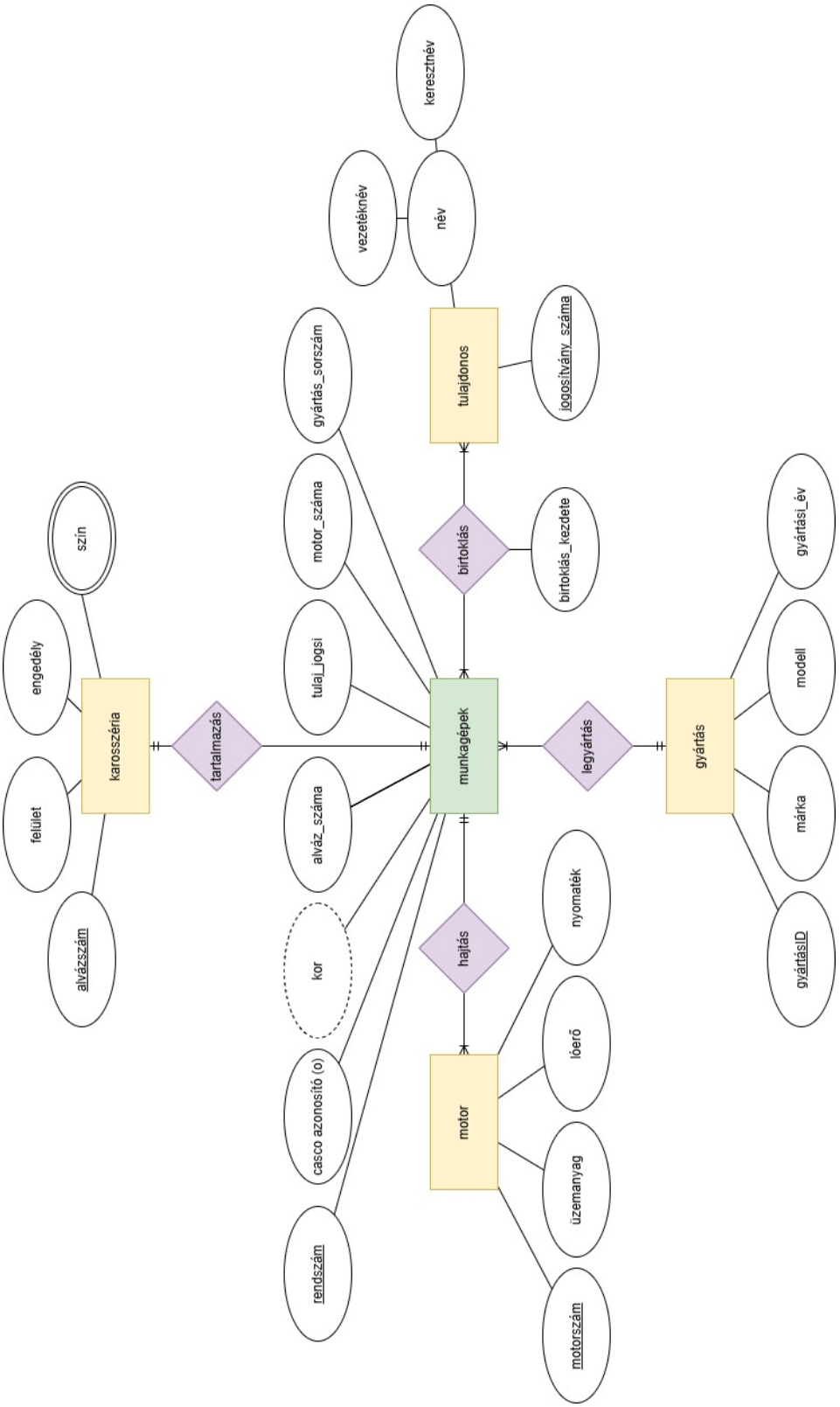
Gyártás

- GyártásID (elsődleges kulcs): Egyedi azonosító, két számból és egy betűből áll.
- Márka: A gyártó neve.
- Modell: A gyártott munkagép típusa.
- Gyártási év: Valós évszámot tartalmaz, amely a munkagép elkészítésének évét mutatja.

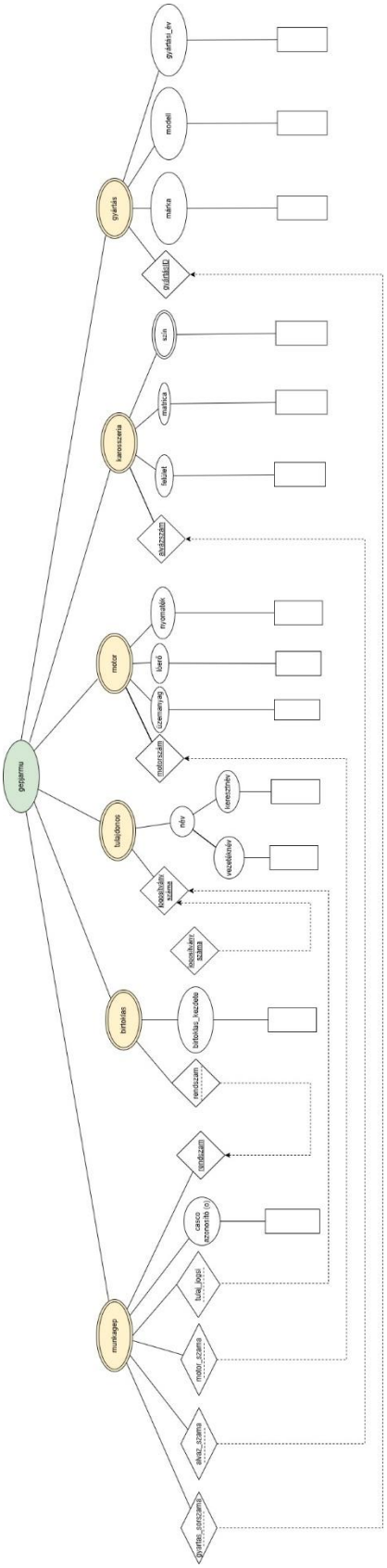
Birtoklás

- Birtoklás kezdete: Valós évszám, amely azt az évet jelzi, amikor a tulajdonos megszerezte a munkagépet.

Az ER modell megalkotása



Az ER modell átkonvertálása XDM modellre



Az XML dokumentum elkészítése

Az XDM modell alapján készítettem el az XML dokumentumot, amelynek első lépése a gyökérelem definiálása volt. Fontosnak tartottam, hogy a struktúra világos és szemléletes legyen, ezért gondosan ügyeltem arra, hogy minden szülőelemhez legalább három gyermekelem tartozzon. Ez a megoldás nemcsak a modell teljességét és részletezettségét mutatja, hanem az adatok sokféleségét is jól szemlélteti.

Az opcionális elemek esetében változatos előfordulásokat alkalmaztam: bizonyos helyeken szerepeltettem őket, míg más szülőelemeknél kihagytam, hogy ezzel is tükrözzem az opcionális jellegből adódó rugalmasságot. A többértékű elemeknél pedig arra törekedtem, hogy különféle kombinációk jelenjenek meg a szülőelemeken belül. Így egyesesetekben csak egy érték szerepel, máshol kettő, vagy akár három is előfordul, hogy a modell jól bemutassa a többértékű attribútumok kezelésének lehetőségeit.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<gepjarmu xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" xs:noNamespaceSchemaLocation="XMLSchemaG0P9OJ.xsd">
    <munkagep rendszam="AB-CD-123">
        <tulajID>HU345678</tulajID>
        <motorID>100FDSAF008F1G</motorID>
        <jarmuID>0FSFDSFFF70000019</jarmuID>
        <gyartasID>15B</gyartasID>
        <biztosit>FSDAG34</biztosit>
    </munkagep>

    <munkagep rendszam="AA-BB-720">
        <tulajID>HU346544</tulajID>
        <motorID>100FDFGDS58F1G</motorID>
        <jarmuID>0FSFDSCCC70000019</jarmuID>
        <gyartasID>10B</gyartasID>
        <biztosit>HSDAG34</biztosit>
    </munkagep>

    <munkagep rendszam="KD-QS-510">
        <tulajID>RO378544</tulajID>
        <motorID>540FDFGDS58F1G</motorID>
        <jarmuID>3HSFDSCCC70000019</jarmuID>
        <gyartasID>98C</gyartasID>
    </munkagep>

    <tulajdonos jogositvany_szama="HU345678">
        <nev>
            <vnev>Kovacs</vnev>
            <knev>Arpad</knev>
        </nev>
    </tulajdonos>

```

```xml
    <tulajdonos jogositvany_szama="HU346544">
        <nev>
            <vnev>Szabo</vnev>
            <knev>Peter</knev>
        </nev>
    </tulajdonos>

    <tulajdonos jogositvany_szama="RO378544">
        <nev>
            <vnev>Elekes</vnev>
            <knev>Tibor</knev>
        </nev>
    </tulajdonos>

    <motor motorszam="100FDSAF008F1G">
        <meghajtas>dizel</meghajtas>
        <loero>68</loero>
        <nyomatek>160 Nm</nyomatek>
    </motor>

    <motor motorszam="100FDFGDS58F1G">
        <meghajtas>benzin</meghajtas>
        <loero>75</loero>
        <nyomatek>107 Nm</nyomatek>
    </motor>

    <motor motorszam="540FDFGDS58F1G">
        <meghajtas>elektromos</meghajtas>
        <loero>150</loero>
        <nyomatek>310 Nm</nyomatek>
    </motor>
```

```xml
    <karosszeria alvazszam="0FSFDSFFF70000019">
        <felulet>matt</felulet>
        <engedely>nincs</engedely>
        <szin>feher</szin>
        <szin>szurke</szin>
    </karosszeria>

    <karosszeria alvazszam="0FSFDSCCC70000019">
        <felulet>teljes</felulet>
        <engedely>van</engedely>
        <szin>feher</szin>
        <szin/>
    </karosszeria>

    <karosszeria alvazszam="3HSFDSCCC70000019">
        <felulet>reszleges</felulet>
        <engedely>nincs</engedely>
        <szin>zold</szin>
        <szin>fekete</szin>
        <szin>szurke</szin>
    </karosszeria>

    <gyartas gyartasID="15B">
        <marka>Hyundai</marka>
        <modell>R25Z-9AK</modell>
        <evszam>2004</evszam>
    </gyartas>

    <gyartas gyartasID="10B">
        <marka>JCB</marka>
        <modell>1CX</modell>
        <evszam>2011</evszam>
    </gyartas>
```

```
98
99      <gyartas gyartasID="98C">
100         <marka>JCB</marka>
101         <modell>4CX</modell>
102         <evszam>2021</evszam>
103     </gyartas>
104
105     <birtoklas jogositvany_szama="HU345678" rendszam="AB-CD-123">
106         <birtoklas_kezdete>2023-01-01</birtoklas_kezdete>
107     </birtoklas>
108
109     <birtoklas jogositvany_szama="HU346544" rendszam="AA-BB-720">
110         <birtoklas_kezdete>2022-11-04</birtoklas_kezdete>
111     </birtoklas>
112
113     <birtoklas jogositvany_szama="RO378544" rendszam="KD-QS-510">
114         <birtoklas_kezdete>2021-05-18</birtoklas_kezdete>
115     </birtoklas>
116 </gepjarmu>
```

Az XMLSchema elkészítése

Az XML dokumentum validálásához egy saját sémát készítettem, amelynél igyekeztem kreatív és alapos lenni. A validációs megkötések kialakítása során saját típusokat definiáltam, valamint olyan alapértelmezett típusokat is felhasználtam, amelyeket az XSD nyelv már eleve biztosít (például az xs:gYear típust). A séma szerkezetét átgondoltan és logikusan alakítottam ki, az alábbi lépések szerint: először egyszerű elemeket hoztam létre, amelyekre később hivatkoztam, majd meghatároztam a saját típusokat, hogy a témakörhöz tartozó egyedi megkötéseket érvényesítsem. Ezt követően megterveztem a séma általános felépítését, meghatároztam az elsődleges kulcsokat, és végül az idegen kulcsokat is hozzáadtam, hogy az elemek közötti kapcsolatok pontosan definiálva legyenek.

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


    <xs:element name="tulajID" type="tulajIDTipus"/>
    <xs:element name="motorID" type="motorIDTipus"/>
    <xs:element name="jarmuID" type="jarmuID"/>
    <xs:element name="gyartasID" type="gyartoIDTipus"/>
    <xs:element name="biztosit" type="xs:string"/>

    <xs:element name="vnev" type="xs:string"/>
    <xs:element name="knev" type="xs:string"/>

    <xs:element name="meghajtas" type="meghajtasTipus"/>
    <xs:element name="loero" type="xs:string"/>
    <xs:element name="nyomatek" type="xs:string"/>

    <xs:element name="felulet" type="feluletTipus"/>
    <xs:element name="engedely" type="engedelyTipus"/>
    <xs:element name="szin" type="xs:string"/>

    <xs:element name="marka" type="xs:string"/>
    <xs:element name="modell" type="xs:string"/>
    <xs:element name="evszam" type="xs:gYear"/>

    <xs:element name="idopont" type="xs:date"/>


    <xs:simpleType name="rendszamTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="[A-Z][A-Z]-[A-Z][A-Z]-[0-9][0-9][0-9]"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="tulajIDTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="[A-Z][A-Z][0-9][0-9][0-9][0-9][0-9][0-9]"/>
        </xs:restriction>
```

```
38      </xs:simpleType>
39
40      <xs:simpleType name="motorIDTipus">
41          <xs:restriction base="xs:string">
42              <xs:pattern value="[A-Z0-9]{14}"/>
43          </xs:restriction>
44      </xs:simpleType>
45
46      <xs:simpleType name="meghajtasTipus">
47          <xs:restriction base="xs:string">
48              <xs:pattern value="benzin|dizel|hibrid|elektromos"/>
49          </xs:restriction>
50      </xs:simpleType>
51
52      <xs:simpleType name="jarmuID">
53          <xs:restriction base="xs:string">
54              <xs:pattern value="[A-Z0-9]{17}"/>
55          </xs:restriction>
56      </xs:simpleType>
57
58      <xs:simpleType name="feluletTipus">
59          <xs:restriction base="xs:string">
60              <xs:pattern value="matt|reszleges|teljes"/>
61          </xs:restriction>
62      </xs:simpleType>
63
64      <xs:simpleType name="engedelyTipus">
65          <xs:restriction base="xs:string">
66              <xs:pattern value="van|nincs"/>
67          </xs:restriction>
68      </xs:simpleType>
69
70      <xs:simpleType name="gyartoIDTipus">
```

```
        <xs:restriction base="xs:string">
            <xs:pattern value="[0-9][0-9][A-Z]"/>
        </xs:restriction>
    </xs:simpleType>


    <xs:complexType name="gepjarmuType">
        <xs:sequence>
            <xs:element name="munkagep" minOccurs="1" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="tulajID" />
                        <xs:element ref="motorID"/>
                        <xs:element ref="jarmuID"/>
                        <xs:element ref="gyartasID" />
                        <xs:element ref="biztosit" minOccurs="0"/>
                    </xs:sequence>
                    <xs:attribute name="rendszam" type="rendszamTipus" use="required"/>
                </xs:complexType>
            </xs:element>

            <xs:element name="tulajdonos" minOccurs="1" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="nev">
                         <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="vnev" />
                                <xs:element ref="knev" />
                            </xs:sequence>
                        </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                    <xs:attribute name="jogositvany_szama" type="tulajIDTipus" use="required"/>
```

```
105                </xs:complexType>
106            </xs:element>
107
108            <xs:element name="motor" minOccurs="1" maxOccurs="unbounded">
109                <xs:complexType>
110                    <xs:sequence>
111                        <xs:element ref="meghajtas" />
112                        <xs:element ref="loero" />
113                        <xs:element ref="nyomatek" />
114                    </xs:sequence>
115                    <xs:attribute name="motorszam" type="motorIDTipus" use="required"/>
116                </xs:complexType>
117            </xs:element>
118
119            <xs:element name="karosszeria" minOccurs="1" maxOccurs="unbounded">
120                <xs:complexType>
121                    <xs:sequence>
122                        <xs:element ref="felulet" />
123                        <xs:element ref="engedely" />
124                        <xs:element ref="szin" maxOccurs="unbounded"/>
125                    </xs:sequence>
126                    <xs:attribute name="alvazszam" type="jarmuID" use="required"/>
127                </xs:complexType>
128            </xs:element>
129
130            <xs:element name="gyartas" minOccurs="1" maxOccurs="unbounded">
131                <xs:complexType>
132                    <xs:sequence>
133                        <xs:element ref="marka" />
```

```
133                        <xs:element ref="marka" />
134                        <xs:element ref="modell" />
135                        <xs:element ref="evszam" />
136                    </xs:sequence>
137                    <xs:attribute name="gyartasID" type="gyartoIDTipus" use="required"/>
138                </xs:complexType>
139            </xs:element>
140
141            <xs:element name="birtoklas" minOccurs="1" maxOccurs="unbounded">
142                <xs:complexType>
143                    <xs:sequence>
144                        <xs:element ref="birtoklas_kezdete"/>
145                    </xs:sequence>
146                        <xs:attribute name="rendszam" type="rendszamTipus" use="required"/>
147                        <xs:attribute name="jogositvany_szama" type="tulajIDTipus" use="required"/>
148                </xs:complexType>
149            </xs:element>
150        </xs:sequence>
151    </xs:complexType>
152
153    <xs:element name="gepjarmu" type="gepjarmuType"/>
154
155
156
157    <xs:key name="munkagep_kulcs">
158        <xs:selector xpath="munkagep" />
159        <xs:field xpath="@rendszam" />
160    </xs:key>
161
162    <xs:key name="tulajdonos_kulcs">
163        <xs:selector xpath="tulajdonos" />
164        <xs:field xpath="@jogositvany_szama" />
```

```
164         <xs:field xpath="@jogositvany_szama" />
165     </xs:key>
166
167     <xs:key name="motor_kulcs">
168         <xs:selector xpath="motor" />
169         <xs:field xpath="@motorszam" />
170     </xs:key>
171
172     <xs:key name="karosszeria_kulcs">
173         <xs:selector xpath="karosszeria" />
174         <xs:field xpath="@alvazszam" />
175     </xs:key>
176
177     <xs:key name="gyartas_kulcs">
178         <xs:selector xpath="gyartas" />
179         <xs:field xpath="@gyartasID" />
180     </xs:key>
181
182
183
184     <xs:keyref refer="tulajdonos_kulcs" name="tulaj_idegen_kulcs">
185         <xs:selector xpath="munkagep" />
186         <xs:field xpath="tulajID" />
187     </xs:keyref>
188
189     <xs:keyref refer="motor_kulcs" name="motor_idegen_kulcs">
190         <xs:selector xpath="munkagep" />
191         <xs:field xpath="motorID" />
192     </xs:keyref>
193
194     <xs:keyref refer="karosszeria_kulcs" name="karosszeria_idegen_kulcs">
195         <xs:selector xpath="munkagep" />
196         <xs:field xpath="jarmuID" />
197     </xs:keyref>
```

```
198
199     <xs:keyref refer="gyartas_kulcs" name="gyartas_idegen_kulcs">
200         <xs:selector xpath="munkagep" />
201         <xs:field xpath="gyartasID" />
202     </xs:keyref>
203
204     <xs:keyref refer="munkagep_kulcs" name="birtoklas_idegen_kulcs"/>
205         <xs:selector xpath="birtoklas" />
206         <xs:field xpath="@rendszam" />
207         <xs:field xpath="@jogositvany_szama" />
208     </xs:keyref>
209
210     <xs:unique name="uniqueTartalmazas">
211         <xs:selector xpath="karosszeria"/>
212         <xs:field xpath="@alvazszam"/>
213     </xs:unique>
214
215 </xs:schema>
```

Adatok beolvasása

A teljes XML dokumentum feldolgozásához egy osztályszintű metódust hoztam létre, amelyet a program futása során kétszer hívok meg. Az első hívás eredményét a konzolra írjuk ki, míg a másodikét egy szöveges dokumentumba mentjük el. A feldolgozást végző Feldolgozas() metódus belsejében először inicializáljuk a beolvasandó fájlt, majd létrehozzuk a parser objektumot, és kiírjuk a gyökérelemet.

A további lépések során az XML dokumentum struktúráját követve iterálunk az adatokon, és minden eltárolt információt megjelenítünk. Az opcionális elemek esetében, ha egy adott szülőelemben nem találhatók meg, akkor "nincs" értéket írunk ki. A többértékű elemeknél pedig egy for ciklust használunk, amely segítségével az összes előforduló értéket külön-külön sorszámmal látjuk el, majd kiírjuk. Ez a módszer biztosítja, hogy a dokumentum adatai teljes mértékben feldolgozottak és jól áttekinthetők legyenek mindkét kimeneti formátumban.

```java
package DomReadG0P9OJ;

import java.io.File;
import java.util.ArrayList;
import java.util.List;
import javax.xml.parsers.*;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.*;
import org.xml.sax.SAXException;

public class DomReadG0P9OJ {
    Run | Debug
    public static void main(String[] args) {
        try {
            // kimeneti fajl inicializalasa
            File newXMLFile = new File(pathname:"XML_read_output.txt");
            StreamResult newXmlStream = new StreamResult(newXMLFile);

            // dom parser factory inicializalasa
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

            // dom parser letrehozasa
            DocumentBuilder builder = factory.newDocumentBuilder();

            // xml fajl beolvasasa a dom-ba
            Document doc = builder.parse(new File(pathname:"XMLG0P9OJ.xml"));

            // ures szovegek torlese a dom-bol
            cutEmptyStrings(doc.getDocumentElement());

            // xml fajlba iras
            writeDoc(doc, newXmlStream);

```

```
            // konzolra kiiras
            System.out.println(makeToXMLFormat(doc));

        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    // ures szovegek torlese a dom-bol
    private static void cutEmptyStrings(Node root) {
        NodeList nodeList = root.getChildNodes();
        List<Node> deleteEmptyLists = new ArrayList<>();
        for (int i = 0; i < nodeList.getLength(); i++) {
            if (nodeList.item(i).getNodeType() == Node.TEXT_NODE
                    && nodeList.item(i).getTextContent().isEmpty()) {
                deleteEmptyLists.add(nodeList.item(i));
            } else {
                cutEmptyStrings(nodeList.item(i));
            }
        }
        for (Node node : deleteEmptyLists) {
            root.removeChild(node);
        }
    }


```

```
    // dom irasa xml fajlba
    public static void writeDoc(Document document, StreamResult output) {
        try {
            // transformer factory inicializalasa
            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            // transformer letrehozasa
            Transformer transformer = transformerFactory.newTransformer();

            // kimeneti formazas beallitasa
            transformer.setOutputProperty(OutputKeys.ENCODING, value:"UTF-8");
            transformer.setOutputProperty(OutputKeys.INDENT, value:"yes");
            transformer.setOutputProperty(name:"{http://xml.apache.org/xslt}indent-amount", value:"2");

            // dom forras inicializalasa
            DOMSource source = new DOMSource(document);
            // dom irasa kimeneti stream-re
            transformer.transform(source, output);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    // xml formazasa
    public static String makeToXMLFormat(Document document) {
        return "<?xml version=\"" + document.getXmlVersion() + "\" encoding=\"" + document.getXmlEncoding() + "\" ?>"
                + elementsToXMLFormat(document.getDocumentElement(), indent:0);
    }


```

```java
// xml elemek formazasa
public static String elementsToXMLFormat(Node node, int indent) {
    if (node.getNodeType() != Node.ELEMENT_NODE) {
        return "";
    }
    StringBuilder output = new StringBuilder();
    output.append(getIndent(indent)).append("<").append(((Element) node).getTagName());
    if (node.hasAttributes()) {
        for (int i = 0; i < node.getAttributes().getLength(); i++) {
            Node attribute = node.getAttributes().item(i);
            output.append(" ").append(attribute.getNodeName()).append("=\"").append(attribute.getNodeValue())
                    .append("\"");
        }
    }
    NodeList children = node.getChildNodes();
    if (children.getLength() == 1 && children.item(0).getNodeType() == Node.TEXT_NODE) {
        output.append(">").append(children.item(0).getTextContent().trim()).append("</")
                .append(((Element) node).getTagName()).append(">\n");
    } else {
        output.append(">\n");
        for (int i = 0; i < children.getLength(); i++) {
            output.append(elementsToXMLFormat(children.item(i), indent + 1));
        }
        output.append(getIndent(indent)).append("</").append(((Element) node).getTagName()).append(">\n");
    }
    return output.toString();
}
```

```java
    // ures helyek szamanak lekerdezese
    private static String getIndent(int indent) {
        StringBuilder indentation = new StringBuilder();
        for (int i = 0; i < indent; i++) {
            indentation.append("    ");
        }
        return indentation.toString();
    }
}
```

Adatmódosítások

Az adatmódosítás során a cél az, hogy az eredeti XML dokumentum tartalmát olvassuk be, de a módosítások eredményét egy különálló fájlban tároljuk el, amelynek neve XMLG0P9OJModify.xml. Ez a megközelítés biztosítja, hogy az eredeti fájl változatlan maradjon, és az eredeti adatok ne vesszenek el. A módosítások során az alábbi hat változtatást végezzük el az adatokon:

1. A második munkagép elem rendszámát módosítjuk: A meglévő rendszám helyére egy új értéket állítunk be.
2. Az első tulajdonos jogosítvány számát átírjuk: Az aktuális jogosítvány szám helyett egy új értéket adunk meg.
3. Az összes karosszérián beállítjuk, hogy van engedély: Az eddigi állapottól függetlenül minden karosszéria elemnél az engedélyek értékét "van"-ra módosítjuk.
4. A benzines motorokat dízelre állítjuk: Az üzemanyagtípus mezőjét minden olyan motor esetében, ahol "benzin" szerepel, "dízel"-re cseréljük.
5. Az evszam elem nevét evjaratra változtatjuk: A séma kompatibilitásával összhangban az elemnevet módosítjuk, hogy az új név tükrözze az adat jelentését.

6. Az összes birtoklás kezdetét egy évvel korábbra állítjuk: Minden birtoklás kezdete értéket módosítunk úgy, hogy az eredeti dátumnál egy évvel korábbit tároljunk el.

```java
package DomModifyG0P9OJ;

import java.io.File;
import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DomModifyG0P9OJ {

    Run | Debug
    public static void main(String argv[]) throws ParserConfigurationException, SAXException, IOException, TransformerException {
        File inputFile = new File(pathname:"XMLG0P9OJ.xml");

        DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
        Document doc = documentBuilder.parse(inputFile);
        doc.getDocumentElement().normalize();


        // masodik munkagep attributumanak modositasa
        Node jarmu = doc.getElementsByTagName(tagname:"munkagep").item(index:1);
        NamedNodeMap attr = jarmu.getAttributes();
        Node nodeAttr = attr.getNamedItem(name:"rendszam");
        nodeAttr.setTextContent(textContent:"AA-BB-123");
```

```
37        System.out.println(x:"munkagep attributum modositva");
38
39
40        // elso tulajdonos attributumanak modositasa
41        Node tulaj = doc.getElementsByTagName(tagname:"tulajdonos").item(index:0);
42        attr = tulaj.getAttributes();
43        nodeAttr = attr.getNamedItem(name:"jogositvany_szama");
44        nodeAttr.setTextContent(textContent:"BB112233");
45        System.out.println(x:"tulajdonos attributum modositva");
46
47
48        // osszes egedely beallitasa van-ra
49        NodeList nodes = doc.getElementsByTagName(tagname:"karosszeria");
50
51        for (int i = 0; i < nodes.getLength(); i++) {
52            Node node = nodes.item(i);
53
54            if (node.getNodeType() == Node.ELEMENT_NODE) {
55                NodeList childNodes = node.getChildNodes();
56
57                for (int j = 0; j < childNodes.getLength(); j++) {
58                    Node childNode = childNodes.item(j);
59                    if (childNode.getNodeName().equals(anObject:"egedely")) {
60                        childNode.setTextContent(textContent:"van");
61                    }
62                }
63            }
64        }
65        System.out.println(x:"egedely modositasok kesz");
66
67
68        // osszes benzines motor atallitasa dizelre
69        nodes = doc.getElementsByTagName(tagname:"motor");
70
```

```java
        for (int i = 0; i < nodes.getLength(); i++) {
            Node node = nodes.item(i);

            if (node.getNodeType() == Node.ELEMENT_NODE) {
                NodeList childNodes = node.getChildNodes();

                for (int j = 0; j < childNodes.getLength(); j++) {
                    Node childNode = childNodes.item(j);
                    if (childNode.getNodeName().equals(anObject:"meghajtas") && childNode.getTextContent().equals(anObject:"benzin")) {
                        childNode.setTextContent(textContent:"dizel");
                    }
                }
            }
        }
        System.out.println(x:"benzin motorok atallitva dizelre");


        // evszam element atnevezese evjaratra
        nodes = doc.getElementsByTagName(tagname:"evszam");
        for (int i = 0; i < nodes.getLength(); i++) {
            doc.renameNode(nodes.item(i), namespaceURI:null, qualifiedName:"evjarat");
        }
        System.out.println(x:"evszam atnevezeve evjaratra");


        // birtoklasok kezdeti datumnak a modositasa mindenhol egy evvel hamarabbra
        nodes = doc.getElementsByTagName(tagname:"birtoklas_kezdete");

        for (int i = 0; i < nodes.getLength(); i++) {
            Node node = nodes.item(i);

            if (node.getNodeType() == Node.ELEMENT_NODE) {
                node.setTextContent(decrementYear(node.getTextContent()));
            }
```

```java
            }
            System.out.println(x:"birtoklas_kezdete modositasa kesz");


            // modositott xml dokumentum elmentese
            writeXml(doc, new File(pathname:"XMLG0P9OJModify.xml"));
        }


        private static void writeXml(Document doc, File output) throws TransformerException {
            Transformer transformer = TransformerFactory.newInstance().newTransformer();
            DOMSource source = new DOMSource(doc);

            StreamResult file = new StreamResult(output);

            transformer.transform(source, file);
        }


        private static String decrementYear(String date) {
            int year = Integer.parseInt(date.substring(beginIndex:0, endIndex:4)) - 1;
            String monthDay = date.substring(beginIndex:4);
            return year + monthDay;
        }
    }
```

Az adatok lekérdezése

Az adatlekérdezés során strukturált és rendezett módon nyerjük ki az információkat az adatbázisból, ügyelve arra, hogy az eredmények áttekinthetők és letisztultak legyenek. A forráskódban minden lekérdezés részletesen dokumentált, és a könnyebb kezelhetőség érdekében ki is van kommentelve. Az alkalmazott lekérdezések a következő információkat biztosítják:

1. Az adatbázisban szereplő összes tulajdonos és minden hozzájuk tartozó adat: Teljes áttekintést ad az adatbázis tulajdonosokról, beleértve a kapcsolódó attribútumaikat is.
2. Az adatbázisban a 10B azonosítóval rendelkező gyártási adatok: Kiemelten csak az adott azonosítóhoz tartozó gyártási információk kerülnek lekérdezésre.
3. Az adatbázis utolsó motorrekordja: Az adatbázisban tárolt motorok közül a legutolsó rekord adatait adja vissza.
4. Az összes olyan karosszéria, amelynek elsődleges színe fehér: Kizárólag azokat a karosszériákat listázza, ahol az elsődleges színként a fehér szerepel.
5. Az összes olyan gyártási információ, amely 2008 után keletkezett: Csak a megadott évnél későbbi gyártási rekordokat tartalmazza.
6. Az adatbázisban tárolt összes birtoklás tulajdonságai: Minden birtoklási rekordot lekérdez, a kapcsolódó attribútumokkal együtt.

```java
package DomQueryG0P9OJ;

import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;

import org.xml.sax.SAXException;

public class DomQueryG0P9OJ {

    Run | Debug
    public static void main(String[] args) {
        try {
            // xml fajl beolvasasa es dom objektum letrehozasa
            DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
            Document document = documentBuilder.parse(uri:"XMLG0P9OJ.xml");

            // dom objektum normalizalasa
            document.getDocumentElement().normalize();

            // xpath inicializalasa
            XPath xPath = XPathFactory.newInstance().newXPath();
```

```
        // lekerdezesek
        String expression = "";


        // kulonbozo xpath lekerdezesek es kifejezesek

        //expression = "/gepjarmu/tulajdonos";
        //expression = "/gepjarmu/gyartas[@gyartasID='10B']";
        //expression = "/gepjarmu/motor[last()]";
        //expression = "/gepjarmu/karosszeria[szin='fekete']";
        //expression = "/gepjarmu/gyartas[evszam>2013]";
        expression = "/gepjarmu/birtoklas";
        NodeList nodeList = (NodeList) xPath.compile(expression).evaluate(document, XPathConstants.NODESET);

        for (int i = 0; i < nodeList.getLength(); i++) {
          Node node = nodeList.item(i);
          System.out.println("\nAktualis elem: " + node.getNodeName());

          // gepjarmu elem kiirasa
          if (node.getNodeType() == Node.ELEMENT_NODE && node.getNodeName().equals(anObject:"munkagep")) {
            Element element = (Element) node;

            System.out.println("Rendszam: " + element.getAttribute(name:"rendszam"));
            System.out.println("Tulaj jogsi: " + element.getElementsByTagName(name:"tulajID").item(index:0).getTextContent());
            System.out.println("Alvaz szama: " + element.getElementsByTagName(name:"jarmuID").item(index:0).getTextContent());
            System.out.println("Motor szama: " + element.getElementsByTagName(name:"motorID").item(index:0).getTextContent());
            System.out.println("Gyartasi szam: " + element.getElementsByTagName(name:"gyartasID").item(index:0).getTextContent());
          }


          // tulajdonos elem kiirasa
          if (node.getNodeType() == Node.ELEMENT_NODE && node.getNodeName().equals(anObject:"tulajdonos")) {
            Element element = (Element) node;

            System.out.println("Jogositvany szama: " + element.getAttribute(name:"jogositvany_szama"));
            System.out.println("Keresztnev: " + element.getElementsByTagName(name:"knev").item(index:0).getTextContent());
            System.out.println("Vezeteknev: " + element.getElementsByTagName(name:"vnev").item(index:0).getTextContent());
          }


          // motor elem kiirasa
          if (node.getNodeType() == Node.ELEMENT_NODE && node.getNodeName().equals(anObject:"motor")) {
            Element element = (Element) node;

            System.out.println("Motor szama: " + element.getAttribute(name:"motorszam"));
            System.out.println("Uzemanyag: " + element.getElementsByTagName(name:"meghajtas").item(index:0).getTextContent());
            System.out.println("Loero: " + element.getElementsByTagName(name:"loero").item(index:0).getTextContent());
            System.out.println("Nyomatek: " + element.getElementsByTagName(name:"nyomatek").item(index:0).getTextContent());
          }


          // karosszeria elem kiirasa
          if (node.getNodeType() == Node.ELEMENT_NODE && node.getNodeName().equals(anObject:"karosszeria")) {
            Element element = (Element) node;

            System.out.println("Alvazszam: " + element.getAttribute(name:"alvazszam"));
            System.out.println("Felulet: " + element.getElementsByTagName(name:"felulet").item(index:0).getTextContent());
            System.out.println("Engedely: " + element.getElementsByTagName(name:"engedely").item(index:0).getTextContent());
            System.out.println("Elsodleges szin: " + element.getElementsByTagName(name:"szin").item(index:0).getTextContent());
          }
```

```
            // gyartas elem kiirasa
            if (node.getNodeType() == Node.ELEMENT_NODE && node.getNodeName().equals(anObject:"gyartas")) {
              Element element = (Element) node;

              System.out.println("ID: " + element.getAttribute(name:"gyartasID"));
              System.out.println("Marka: " + element.getElementsByTagName(name:"marka").item(index:0).getTextContent());
              System.out.println("Modell: " + element.getElementsByTagName(name:"modell").item(index:0).getTextContent());
              System.out.println("Gyartasi ev: " + element.getElementsByTagName(name:"evszam").item(index:0).getTextContent());
            }


            // birtoklasok kiirasa
            if (node.getNodeType() == Node.ELEMENT_NODE && node.getNodeName().equals(anObject:"birtoklas")){
              Element element = (Element) node;

              System.out.println("Rendszam: " + element.getAttribute(name:"rendszam"));
              System.out.println("Jogositvany szama: " + element.getAttribute(name:"jogositvany_szama"));
              System.out.println("Birtoklas kezdete: " + element.getElementsByTagName(name:"birtoklas_kezdete").item(index:0).getTextContent());
            }
          }

        } catch (ParserConfigurationException | SAXException | IOException | XPathExpressionException e) {
          e.printStackTrace();
        }
    }
}
```

Az adatok íratása

Az adatírási feladat megoldásához egy olyan DOM API-alapú programot készítettem, amely képes az XMLG0P9OJ.xml fájlt beolvasni, majd annak fa struktúráját megjeleníteni a konzolon, illetve az eredményt egy új fájlba, az XMLG0P9OJ1.xml-be elmenteni. A feladat során arra törekedtem, hogy az XML dokumentum szerkezete pontosan megmaradjon, miközben a másolat elkészül.

A munka során két programot is létrehoztam. Az első verzió elkészítése után úgy döntöttem, hogy azt nem törlöm ki a beadandóból, hiszen hasznos lehet az összehasonlításhoz és a fejlesztési folyamat bemutatásához. Az új programot DomWriteG0P9OJ néven hoztam létre, és ebben már teljesen kidolgoztam az XMLG0P9OJ.xml fájl fa struktúrájának feldolgozását. Az eredményt nemcsak a konzolra íratom ki, hanem egy új fájlba, az XMLG0P9OJ1.xml-be is elmentem. Ezzel a megközelítéssel sikerült egy pontos, jól strukturált adatírási folyamatot kialakítanom, amely megőrizte az eredeti fájl tartalmát, miközben az új fájlban megjelenítette azt.

```java
package DomWriteG0P9OJ;

import java.io.File;
import java.io.StringWriter;
import java.text.SimpleDateFormat;
import java.util.Date;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class DomWriteG0P9OJ {

    Run | Debug
    public static void main(String[] args) throws Exception {
        // letrehozunk egy uj dokumentumot
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document doc = builder.newDocument();

        // gyoker elem letrehozasa
        Element gepjarmuElement = doc.createElement(tagName:"gepjarmu");
        gepjarmuElement.setAttribute(name:"xmlns:xs", value:"http://www.w3.org/2001/XMLSchema-instance");
        gepjarmuElement.setAttribute(name:"xs:noNamespaceSchemaLocation", value:"XMLSchemaG0P9OJ.xsd");
        doc.appendChild(gepjarmuElement);

        // munkagepek
        addMunkagep(doc, gepjarmuElement, rendszam:"AB-CD-123", tulajJogsi:"HU345678",
        motorSzam:"100FDSAF008F1G", alvazSzam:"0FSFDSFFF70000019", gyartasSorszam:"15B",
                cascoAzon:"FSDAG34");
        addMunkagep(doc, gepjarmuElement, rendszam:"AA-BB-720", tulajJogsi:"HU346544",
        motorSzam:"100FDFGDS58F1G", alvazSzam:"0FSFDSCCC70000019", gyartasSorszam:"10B",
                cascoAzon:"HSDAG34");
        addMunkagep(doc, gepjarmuElement, rendszam:"KD-QS-510", tulajJogsi:"RO378544",
        motorSzam:"540FDFGDS58F1G", alvazSzam:"3HSFDSCCC70000019", gyartasSorszam:"98C",
                cascoAzon:null);

        // tulajdonosok
        addTulajdonos(doc, gepjarmuElement, jogositvanySzama:"HU345678", vnev:"Kovacs", knev:"Arpad");
        addTulajdonos(doc, gepjarmuElement, jogositvanySzama:"HU346544", vnev:"Szabo", knev:"Peter");
        addTulajdonos(doc, gepjarmuElement, jogositvanySzama:"RO378544", vnev:"Elekes", knev:"Tibor");

        // motorok
        addMotor(doc, gepjarmuElement, motorSzam:"100FDSAF008F1G", meghajtas:"dizel", loero:"68", nyomatek:"160 Nm");
        addMotor(doc, gepjarmuElement, motorSzam:"100FDFGDS58F1G", meghajtas:"benzin", loero:"75", nyomatek:"107 Nm");
        addMotor(doc, gepjarmuElement, motorSzam:"540FDFGDS58F1G", meghajtas:"elektromos", loero:"150", nyomatek:"310 Nm");

        // karosszeriak
        addKarosszeria(doc, gepjarmuElement, alvazSzam:"0FSFDSFFF70000019", felulet:"matt", engedely:"nincs", ...szinek:"feher", "szurke");
        addKarosszeria(doc, gepjarmuElement, alvazSzam:"0FSFDSCCC70000019", felulet:"teljes", engedely:"van", ...szinek:"feher", null);
        addKarosszeria(doc, gepjarmuElement, alvazSzam:"3HSFDSCCC70000019", felulet:"reszleges", engedely:"nincs",
        ...szinek:"zold", "fekete", "szurke");

        // gyartasi infok
        addGyartas(doc, gepjarmuElement, gyartasID:"15B", marka:"Hyundai", modell:"R25Z-9AK", gyartasiEv:"2020");
        addGyartas(doc, gepjarmuElement, gyartasID:"10B", marka:"JCB", modell:"1CX", gyartasiEv:"2008");
        addGyartas(doc, gepjarmuElement, gyartasID:"98C", marka:"JCB", modell:"4CX", gyartasiEv:"2019");
```

```
        // birtoklasi kapcsolatok
        addBirtoklas(doc, gepjarmuElement, rendszam:"AB-CD-123", jogositvanySzama:"HU345678", idoKezdete:"2023-01-01");
        addBirtoklas(doc, gepjarmuElement, rendszam:"AA-BB-720", jogositvanySzama:"HU346544", idoKezdete:"2022-11-04");
        addBirtoklas(doc, gepjarmuElement, rendszam:"KD-QS-510", jogositvanySzama:"RO378544", idoKezdete:"2021-05-18");

        // fajl mentese
        String fileName = "XMLG0P9OJ2.xml";
        saveXML(doc, fileName);

        // XML kiíratása a konzolra
        printXML(doc);
    }


    // munkagep elem hozzadasa az xml-hez
    private static void addMunkagep(Document doc, Element parent, String rendszam, String tulajJogsi,
            String motorSzam, String alvazSzam, String gyartasSorszam, String cascoAzon) {
        Element munkagepElement = doc.createElement(tagName:"munkagep");
        munkagepElement.setAttribute(name:"rendszam", rendszam);

        addChildElement(doc, munkagepElement, name:"tulajID", tulajJogsi);
        addChildElement(doc, munkagepElement, name:"motorID", motorSzam);
        addChildElement(doc, munkagepElement, name:"jarmuID", alvazSzam);
        addChildElement(doc, munkagepElement, name:"gyartasID", gyartasSorszam);

        if (cascoAzon != null) {
            addChildElement(doc, munkagepElement, name:"biztosit", cascoAzon);
        }

        parent.appendChild(munkagepElement);
    }
```

```
    // tulajdonos elem hozzadasa az xml-hez
    private static void addTulajdonos(Document doc, Element parent, String jogositvanySzama, String vnev,
            String knev) {
        Element tulajdonosElement = doc.createElement(tagName:"tulajdonos");
        tulajdonosElement.setAttribute(name:"jogositvany_szama", jogositvanySzama);

        Element nevElement = doc.createElement(tagName:"nev");
        addChildElement(doc, nevElement, name:"vnev", vnev);
        addChildElement(doc, nevElement, name:"knev", knev);

        tulajdonosElement.appendChild(nevElement);
        parent.appendChild(tulajdonosElement);
    }


    // motor elem hozzadasa az xml-hez
    private static void addMotor(Document doc, Element parent, String motorSzam, String meghajtas, String loero,
            String nyomatek) {
        Element motorElement = doc.createElement(tagName:"motor");
        motorElement.setAttribute(name:"motorszam", motorSzam);

        addChildElement(doc, motorElement, name:"meghajtas", meghajtas);
        addChildElement(doc, motorElement, name:"loero", loero);
        addChildElement(doc, motorElement, name:"nyomatek", nyomatek);

        parent.appendChild(motorElement);
    }
```

```java
    // karosszeria elem hozzadasa az xml-hez
    private static void addKarosszeria(Document doc, Element parent, String alvazSzam, String felulet, String engedely,
            String... szinek) {
        Element karosszeriaElement = doc.createElement(tagName:"karosszeria");
        karosszeriaElement.setAttribute(name:"alvazszam", alvazSzam);

        addChildElement(doc, karosszeriaElement, name:"felulet", felulet);
        addChildElement(doc, karosszeriaElement, name:"engedely", engedely);

        for (String szin : szinek) {
            addChildElement(doc, karosszeriaElement, name:"szin", szin);
        }

        parent.appendChild(karosszeriaElement);
    }


    // gyartas elem hozzadasa az xml-hez
    private static void addGyartas(Document doc, Element parent, String gyartasID, String marka, String modell,
            String gyartasiEv) {
        Element gyartasElement = doc.createElement(tagName:"gyartas");
        gyartasElement.setAttribute(name:"gyartasID", gyartasID);

        addChildElement(doc, gyartasElement, name:"marka", marka);
        addChildElement(doc, gyartasElement, name:"modell", modell);
        addChildElement(doc, gyartasElement, name:"evszam", gyartasiEv);

        parent.appendChild(gyartasElement);
    }

```

```java
    // birtoklas kapcsolat hozzadasa az xml-hez
    private static void addBirtoklas(Document doc, Element parent, String rendszam, String jogositvanySzama,
            String idoKezdete) {
        Element birtoklasElement = doc.createElement(tagName:"birtoklas");
        birtoklasElement.setAttribute(name:"rendszam", rendszam);
        birtoklasElement.setAttribute(name:"jogositvany_szama", jogositvanySzama);

        addChildElement(doc, birtoklasElement, name:"birtoklas_kezdete", idoKezdete);

        parent.appendChild(birtoklasElement);
    }


    // gyerek elem hozzadasa az xml-hez
    private static void addChildElement(Document doc, Element parent, String name, String value) {
        Element childElement = doc.createElement(name);
        childElement.appendChild(doc.createTextNode(value));
        parent.appendChild(childElement);
    }


    // xml dokumentum mentese fajlba
    private static void saveXML(Document doc, String fileName) throws Exception {
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.ENCODING, value:"UTF-8");
        transformer.setOutputProperty(OutputKeys.INDENT, value:"yes");

        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(new File(fileName));
        transformer.transform(source, result);
    }

```

```
192     // xml kiiratasa konzolra
193     private static void printXML(Document doc) throws Exception {
194         TransformerFactory transformerFactory = TransformerFactory.newInstance();
195         Transformer transformer = transformerFactory.newTransformer();
196         transformer.setOutputProperty(OutputKeys.ENCODING, value:"UTF-8");
197         transformer.setOutputProperty(OutputKeys.INDENT, value:"yes");
198 
199 
200     // stringwriter letrehozasa
201         StringWriter writer = new StringWriter();
202         StreamResult result = new StreamResult(writer);
203 
204 
205     // dom forras
206         DOMSource source = new DOMSource(doc);
207 
208 
209     // strinwriter-be transzformalas
210         transformer.transform(source, result);
211 
212 
213     // konzolra iratas
214         System.out.println(x:"XML dokumentum:");
215         System.out.println(writer.toString());
216     }
217 }
```