

JEGYZŐKÖNYV

Adatkezelés XML

környezetben

Féléves feladat

Tartalomjegyzék

A feladat leírása.....	3
1. Feladat.....	4
1.1: ER Modell	4
1.2: XDM Modell	5
1.3: Az XML kód.....	6
1.4: Az XSD dokumentum.....	10
2.Feladat.....	14
2.1: Az Java beolvasó és kiíró program.	14
Kimenet:	16
2.2: Az Java módosító program.....	21
Kimenet:	23
2.3: Az Java lekérdező program.	24
Kimenet:	27

A feladat leírása

Az én témám egy város tömegközlekedési vállalatának a modellezése. Az adatbázisának ER-modellezése, konvertálása XDM modellre, majd az XDM modell alapján XML dokumentum és XMLSchema készítés, majd DOM programok készítése az XML dokumentum adatainak olvasására és módosítására.

Az ER-modell az alábbi jellemzőkkel épül fel:

A **Jármű** egyedhez tartoznak a *Rendszámtábla*, a *Férőhely*, a *Gyártó* és a *Típus* tulajdonságok. A *Rendszámtábla* kulcsként funkcionál, mert egyértelműen azonosítja az adott **Járművet**.

A **Járművezető** egyednek a *Mikor vezeti* lesz a kulcsa. A **Járművezetőt** a **Jármű** egyeddel összekötő kapcsolat a VEZETI. A **Járművezetőnek** az *Életkor* attribútuma származtatott, ugyanis az kiszámítható a *Születési dátum* tulajdonságából. A *Jogosítványok* egy többértékű tulajdonság, hiszen egy **Járművezetőnek** több, különböző jogosítványa is lehet. A *Név* egy összetett attribútum, amelyet a *Vezetéknév* és a *Utónév* altulajdonságok alkotnak.

A **Jármű** és a **Járművezető** egyedek közötti kapcsolat kötelező, mert valakinek vezetnie kell az adott járművet és a járművezetőnek is kell vezetnie egy járművet (ekkor kerül adatbázisba). 1:1 típusú, hiszen egy járművet egy ember vezethet és egy ember is egy járművet képes vezetni.

A **Menetrend** egyed tulajdonságai a *Menetrend azonosító*, az *Indulás időpontja* és az *Érkezés időpontja*. A *Menetrend azonosító* szolgál kulcsként.

A **Menetrend** és a **Jármű** egyedek közötti KÖZLEKEDIK kapcsolat 1:N típusú, több **Jármű** is közlekedhet ugyanazon menetrend szerint. Kötelező kapcsolat.

Az **Ellenőr** egyed tulajdonsága az *Ellenőr azonosító*, amely a kulcs szerepét is betölti. Az **Ellenőr** egyed továbbá jelentkezik egy *Név* összetett tulajdonságból, amely a *Vezetéknév* és *Utónév* tagokból áll össze.

Az **Ellenőr** és a **Jármű** egyedek között a TARTÓZKODIK kapcsolat áll fenn, amely a **Jármű** felé kötelező, az **Ellenőr** felé opcionális (az ellenőrnek járművön kell lennie, de nem kell minden járművön ellenőrnek lennie) és 1:1 típusú (egy járművön egy ellenőr lehet).

Az **Útvonal** egyedhez tartozó tulajdonságok a *Vonalszám*, a *Távolság*, a *Kilométerár* és a *Viteldíj*. A *Vonalszám* kulcs, amely egyértelműen azonosítja az adott **Útvonalat**. A *Viteldíj* egy származtatott tulajdonság, a *Távolság* és a *Kilométerár*ból számolható.

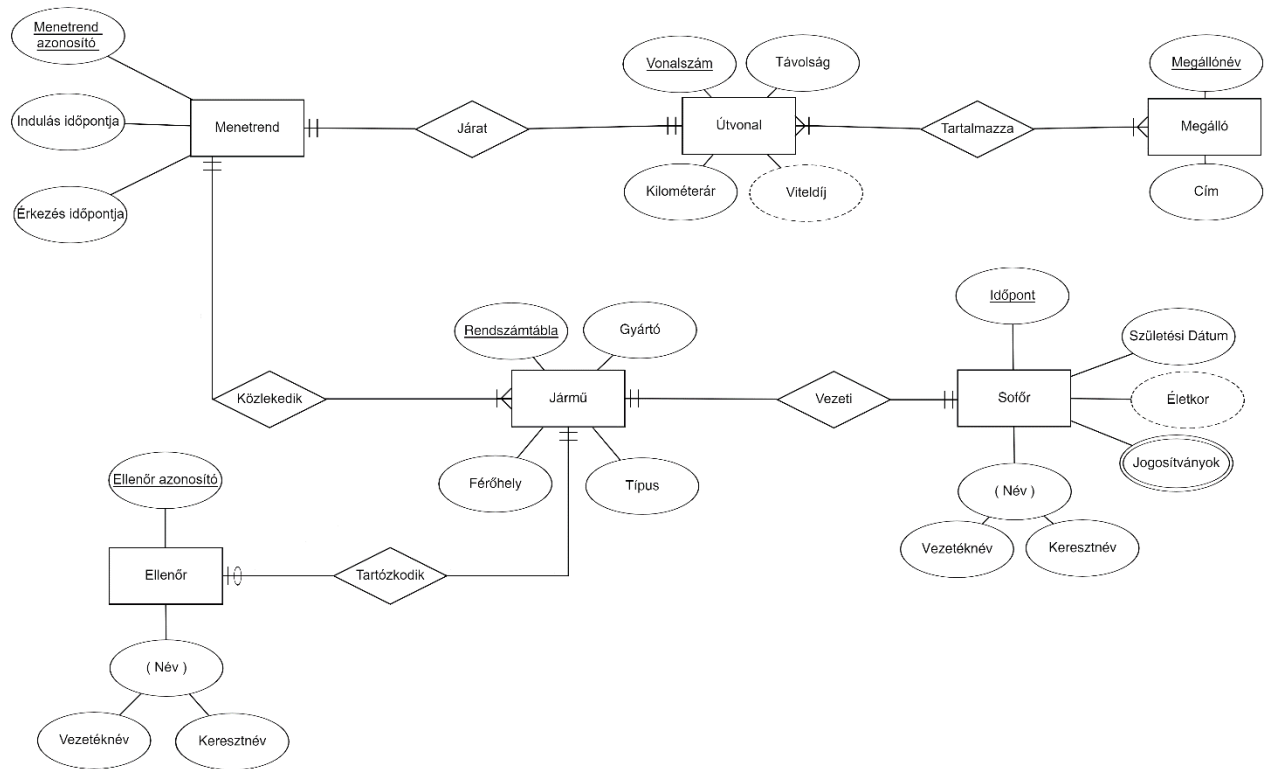
Az **Útvonal** és a **Menetrend** egyedek között lévő JÁRAT kapcsolat kötelező (minden útvonalon van menetrend) és 1:1 típusú (egy útvonalon egyféle menetrend).

A **Megálló** is egy egyed, amelyben a *Megállónév* tulajdonsága kulcsként funkcionál. A **Megálló** egyednek tulajdonsága továbbá a *Cím*.

Az **Útvonal** és a **Megálló** egyedek között a TARTALMAZZA kapcsolat van, amely N:N típusú, ugyanis egy útvonalhoz több megálló is tartozhat és egy megállóhoz is tartozhat többféle útvonal. Kötelező kapcsolat, hiszen megállók nélkül nincs útvonal és útvonal nélkül nincs értelme egy megállónak sem.

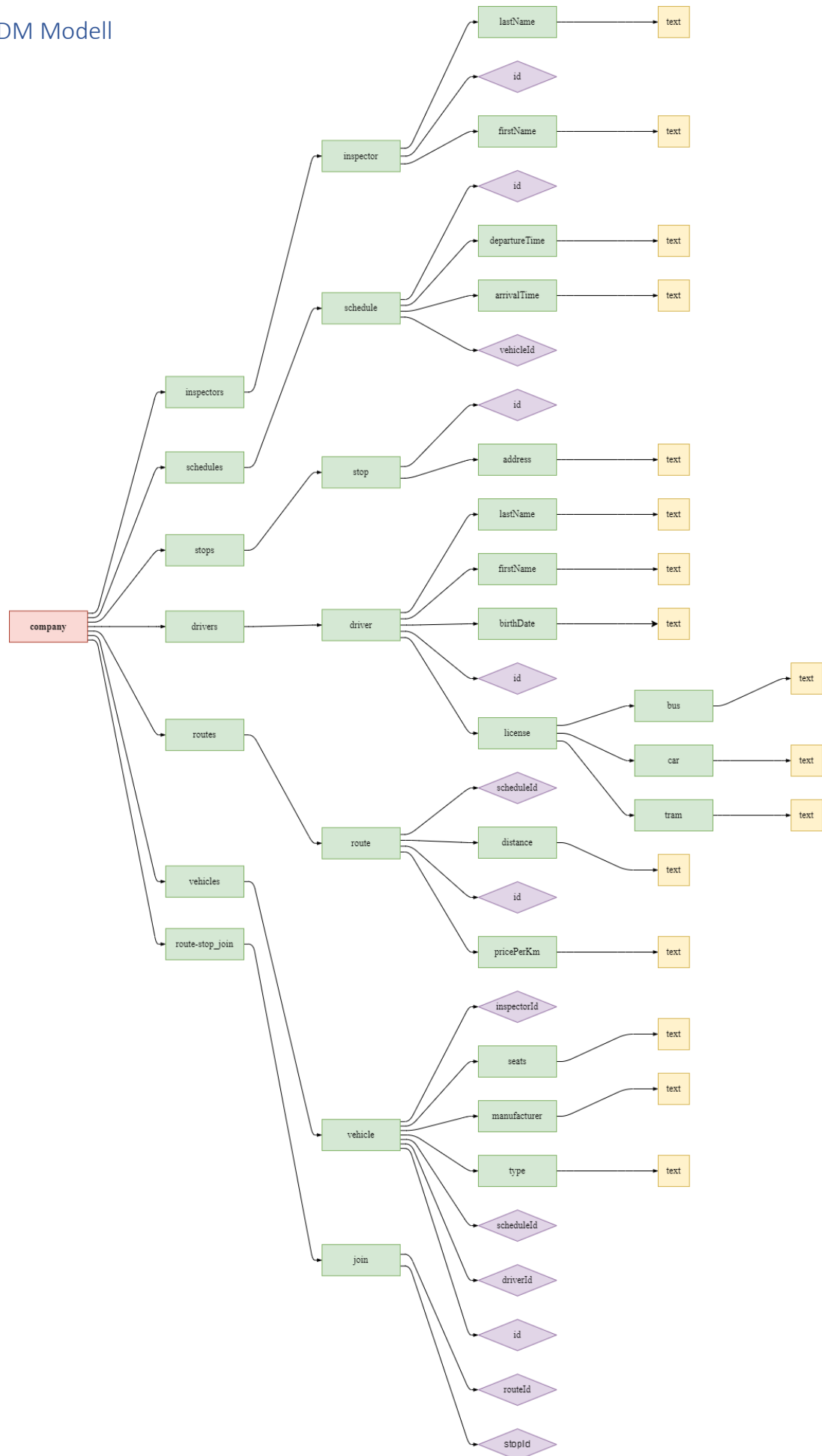
1. Feladat

1.1: ER Modell



Az ER-modelt az ERDplus programmal készítettem el (<https://erdplus.com/>)

1.2: XDM Modell



Az XDM modellt a draw.io programmal szerkesztettem (<https://app.diagrams.net/>).

Az egyedekből elemek keletkeztek, az elsődleges és idegen kulcsokból attribútumokká alakultak. A többértékű tulajdonságból egy külön elemet készítünk ahol a megadható értékeket gyerekelemekként rögzítjük. Az 1:1 és 1:N típusú kapcsolatokat csak idegen kulcsként rögzítjük, az N:M kapcsolatokat pedig kapcsolótáblával kezeljük. Az összetett tulajdonságokat elhagyjuk, csak azokat a tulajdonságokat rögzítjük elemként amikből az összetett tulajdonság előáll. A származtatott tulajdonságokat is szintén el- hagyjuk hiszen azt ki tudjuk mi is számolni a programunkban.

1.3: Az XML kód

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <company xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="XMLSchemaNDRQ1Y.xsd">
3   <inspectors>
4     <inspector id="2">
5       <lastName>Jean</lastName>
6       <firstName>Todt</firstName>
7     </inspector>
8     <inspector id="4">
9       <lastName>Laboda</lastName>
10      <firstName>Dániel</firstName>
11    </inspector>
12    <inspector id="6">
13      <lastName>Elek</lastName>
14      <firstName>Zoltán</firstName>
15    </inspector>
16  </inspectors>
17  <schedules>
18    <schedule id="1" vehicleId="ASD-123">
19      <arrivalTime>05:50:00</arrivalTime>
20      <departureTime>06:15:00</departureTime>
21    </schedule>
22    <schedule id="3" vehicleId="QWE-968">
23      <arrivalTime>08:30:00</arrivalTime>
24      <departureTime>08:35:00</departureTime>
25    </schedule>
26    <schedule id="5" vehicleId="ADC-666">
27      <arrivalTime>10:17:00</arrivalTime>
28      <departureTime>10:50:00</departureTime>
29    </schedule>
30    <schedule id="7" vehicleId="BOT-404">
31      <arrivalTime>12:40:00</arrivalTime>
32      <departureTime>13:00:00</departureTime>
33    </schedule>
34    <schedule id="8" vehicleId="TOP-282">
35      <arrivalTime>20:22:00</arrivalTime>
36      <departureTime>20:50:00</departureTime>
37    </schedule>
38  </schedules>
```

```

39 <stops>
40   <stop id="Bacsó Béla Utca">
41     <address>Bacsó Béla út 40.</address>
42   </stop>
43   <stop id="Soltész Nagy Kálmán">
44     <address>Soltész Nagy Kálmán út 8.</address>
45   </stop>
46   <stop id="Vízügyi igazgatóság">
47     <address>Vörösmarty Mihály utca. 77</address>
48   </stop>
49   <stop id="Miskolctapolca-Barlangfürdő">
50     <address>Pazár István Sétány 1.</address>
51   </stop>
52   <stop id="Római part">
53     <address>Római Part 30</address>
54   </stop>
55 </stops>
56 <drivers>
57   <driver id="1111-03-22">
58     <birthDate>1969-01-03</birthDate>
59     <lastName>Mischael</lastName>
60     <firstName>Schumacher</firstName>
61     <license>
62       <car>true</car>
63       <bus>true</bus>
64       <tram>true</tram>
65     </license>
66   </driver>
67   <driver id="1111-03-23">
68     <birthDate>1960-03-21</birthDate>
69     <lastName>Ayrton</lastName>
70     <firstName>Senna</firstName>
71     <license>
72       <car>true</car>
73       <bus>true</bus>
74       <tram>false</tram>
75     </license>
76   </driver>
77   <driver id="1111-03-24">
78     <birthDate>1955-02-24</birthDate>
79     <lastName>Alain</lastName>
80     <firstName>Prost</firstName>
81     <license>
82       <car>false</car>
83       <bus>false</bus>
84       <tram>true</tram>
85     </license>
86   </driver>

```

```

87     <driver id="1111-03-25">
88         <birthDate>1987-07-03</birthDate>
89         <lastName>Sebastian</lastName>
90         <firstName>Vettel</firstName>
91         <license>
92             <car>false</car>
93             <bus>true</bus>
94             <tram>true</tram>
95         </license>
96     </driver>
97     <driver id="1111-03-26">
98         <birthDate>1981-07-29</birthDate>
99         <lastName>Fernando</lastName>
100        <firstName>Alonso</firstName>
101        <license>
102            <car>true</car>
103            <bus>true</bus>
104            <tram>false</tram>
105        </license>
106    </driver>
107 </drivers>
108 <routes>
109     <route id="1A" scheduleId="1">
110         <distance>12</distance>
111         <pricePerKm>30.2</pricePerKm>
112     </route>
113     <route id="20B" scheduleId="3">
114         <distance>14</distance>
115         <pricePerKm>30.75</pricePerKm>
116     </route>
117     <route id="43" scheduleId="5">
118         <distance>20</distance>
119         <pricePerKm>40.25</pricePerKm>
120     </route>
121     <route id="1" scheduleId="7">
122         <distance>17</distance>
123         <pricePerKm>25.0</pricePerKm>
124     </route>
125     <route id="12G" scheduleId="8">
126         <distance>21</distance>
127         <pricePerKm>35.0</pricePerKm>
128     </route>
129 </routes>
130 <vehicles>
131     <vehicle id="ASD-123" driverId="1111-03-22" scheduleId="1" inspectorId="2">
132         <seats>48</seats>
133         <manufacturer>MAN</manufacturer>
134         <type>bus</type>
135     </vehicle>

```



```

136     <vehicle id="QWE-968" driverId="1111-03-23" scheduleId="3" inspectorId="4">
137         <seats>34</seats>
138         <manufacturer>Ikarus</manufacturer>
139         <type>bus</type>
140     </vehicle>
141     <vehicle id="ADC-666" driverId="1111-03-24" scheduleId="5" inspectorId="6">
142         <seats>70</seats>
143         <manufacturer>Renault</manufacturer>
144         <type>bus</type>
145     </vehicle>
146     <vehicle id="BOT-404" driverId="1111-03-25" scheduleId="7">
147         <seats>230</seats>
148         <manufacturer>Skoda</manufacturer>
149         <type>tram</type>
150     </vehicle>
151     <vehicle id="TOP-282" driverId="1111-03-26" scheduleId="8">
152         <seats>200</seats>
153         <manufacturer>Skoda</manufacturer>
154         <type>tram</type>
155     </vehicle>
156 </vehicles>
157 <route-stop_join>
158     <join routeId="1A" stopId="Bacsó Béla Utca" />
159     <join routeId="1A" stopId="Római part" />
160     <join routeId="20B" stopId="Soltész Nagy Kálmán" />
161     <join routeId="20B" stopId="Vízügyi igazgatóság" />
162     <join routeId="43" stopId="Soltész Nagy Kálmán" />
163     <join routeId="43" stopId="Miskolctapolca-Barlangfürdő" />
164     <join routeId="43" stopId="Római part" />
165     <join routeId="1" stopId="Vízügyi igazgatóság" />
166     <join routeId="1" stopId="Miskolctapolca-Barlangfürdő" />
167     <join routeId="12G" stopId="Bacsó Béla Utca" />
168     <join routeId="12G" stopId="Vízügyi igazgatóság" />
169     <join routeId="12G" stopId="Római part" />
170 </route-stop_join>
171 </company>

```

1.4: Az XSD dokumentum

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3      <!-- gyökér csomópont és az abból származó komplex típusú elemek -->
4      <xs:element name="company">
5          <xs:complexType>
6              <xs:sequence>
7                  <xs:element name="inspectors">
8                      <xs:complexType>
9                          <xs:sequence>
10                             <xs:element name="inspector" type="inspector" minOccurs="0" maxOccurs="unbounded" />
11                         </xs:sequence>
12                     </xs:complexType>
13                 </xs:element>
14                 <xs:element name="schedules">
15                     <xs:complexType>
16                         <xs:sequence>
17                             <xs:element name="schedule" type="schedule" minOccurs="0" maxOccurs="unbounded" />
18                         </xs:sequence>
19                     </xs:complexType>
20                 </xs:element>
21                 <xs:element name="stops">
22                     <xs:complexType>
23                         <xs:sequence>
24                             <xs:element name="stop" type="stop" minOccurs="0" maxOccurs="unbounded" />
25                         </xs:sequence>
26                     </xs:complexType>
27                 </xs:element>
28                 <xs:element name="drivers">
29                     <xs:complexType>
30                         <xs:sequence>
31                             <xs:element name="driver" type="driver" minOccurs="0" maxOccurs="unbounded" />
32                         </xs:sequence>
33                     </xs:complexType>
34                 </xs:element>
35                 <xs:element name="routes">
36                     <xs:complexType>
37                         <xs:sequence>
38                             <xs:element name="route" type="route" minOccurs="0" maxOccurs="unbounded" />
39                         </xs:sequence>
40                     </xs:complexType>
41                 </xs:element>
42                 <xs:element name="vehicles">
43                     <xs:complexType>
44                         <xs:sequence>
45                             <xs:element name="vehicle" type="vehicle" minOccurs="0" maxOccurs="unbounded" />
46                         </xs:sequence>
47                     </xs:complexType>
48                 </xs:element>
49                 <xs:element name="route-stop_join">
```

```

50         <xs:complexType>
51             <xs:sequence>
52                 <xs:element name="join" type="route-stop" minOccurs="0" maxOccurs="unbounded" />
53             </xs:sequence>
54         </xs:complexType>
55     </xs:element>
56 </xs:sequence>
57 </xs:complexType>
58
59 <!-- key - keyref azonossági megszorítások -->
60 <xs:unique name="inspectorId">
61     <xs:selector xpath="//vehicle" />
62     <xs:field xpath="@inspectorId" />
63 </xs:unique>
64 <xs:keyref name="inspectorIdref" refer="inspectorId">
65     <xs:selector xpath="//inspector" />
66     <xs:field xpath="@id" />
67 </xs:keyref>
68
69 <xs:key name="routeId">
70     <xs:selector xpath="//route" />
71     <xs:field xpath="@id" />
72 </xs:key>
73 <xs:keyref name="routeIdref" refer="routeId">
74     <xs:selector xpath="//join" />
75     <xs:field xpath="@routeId" />
76 </xs:keyref>
77
78 <xs:key name="scheduleId">
79     <xs:selector xpath="//schedule" />
80     <xs:field xpath="@id" />
81 </xs:key>
82 <xs:keyref name="scheduleIdref" refer="scheduleId">
83     <xs:selector xpath="//route | //vehicle" />
84     <xs:field xpath="@scheduleId" />
85 </xs:keyref>
86
87 <xs:key name="stopId">
88     <xs:selector xpath="//stop" />
89     <xs:field xpath="@id" />
90 </xs:key>
91 <xs:keyref name="stopIdref" refer="stopId">
92     <xs:selector xpath="//join" />
93     <xs:field xpath="@stopId" />
94 </xs:keyref>
95
96 <xs:key name="driverId">
97     <xs:selector xpath="//driver" />

```

```

98         <xs:field xpath="@id" />
99     </xs:key>
100     <xs:keyref name="driverIdref" refer="driverId">
101         <xs:selector xpath="." />
102         <xs:field xpath="@driverId" />
103     </xs:keyref>
104 </xs:element>
105
106 <!-- egyedileg definiált komplex típusok -->
107 <xs:complexType name="inspector">
108     <xs:sequence>
109         <xs:element name="lastName" type="xs:string" />
110         <xs:element name="firstName" type="xs:string" />
111     </xs:sequence>
112     <xs:attribute name="id" type="xs:int" use="required" />
113 </xs:complexType>
114
115 <xs:complexType name="schedule">
116     <xs:sequence>
117         <xs:element name="arrivalTime" type="xs:time" />
118         <xs:element name="departureTime" type="xs:time" />
119     </xs:sequence>
120     <xs:attribute name="id" type="xs:int" use="required" />
121     <xs:attribute name="vehicleId" type="licensePlateNumber" use="required" />
122 </xs:complexType>
123
124 <xs:complexType name="stop">
125     <xs:sequence>
126         <xs:element name="address" type="xs:string" />
127     </xs:sequence>
128     <xs:attribute name="id" type="xs:string" use="required" />
129 </xs:complexType>
130
131 <xs:complexType name="driver">
132     <xs:sequence>
133         <xs:element name="birthDate" type="xs:date" />
134         <xs:element name="lastName" type="xs:string" />
135         <xs:element name="firstName" type="xs:string" />
136         <xs:element name="license" type="license" />
137     </xs:sequence>
138     <xs:attribute name="id" type="xs:date" use="required" />
139 </xs:complexType>
140
141 <xs:complexType name="license">
142     <xs:sequence>
143         <xs:element name="car" type="xs:boolean" />
144         <xs:element name="bus" type="xs:boolean" />
145         <xs:element name="tram" type="xs:boolean" />
146     </xs:sequence>

```

```

147 </xs:complexType>
148
149 <xs:complexType name="route">
150   <xs:sequence>
151     <xs:element name="distance" type="xs:int" />
152     <xs:element name="pricePerKm" type="xs:decimal" />
153   </xs:sequence>
154   <xs:attribute name="id" type="xs:string" use="required" />
155   <xs:attribute name="scheduleId" type="xs:int" use="required" />
156 </xs:complexType>
157
158 <xs:complexType name="vehicle">
159   <xs:sequence>
160     <xs:element name="seats" type="xs:int" />
161     <xs:element name="manufacturer" type="xs:string" />
162     <xs:element name="type">
163       <xs:simpleType>
164         <xs:restriction base="xs:string">
165           <xs:enumeration value="bus" />
166           <xs:enumeration value="tram" />
167         </xs:restriction>
168       </xs:simpleType>
169     </xs:element>
170   </xs:sequence>
171   <xs:attribute name="id" type="licensePlateNumber" use="required" />
172   <xs:attribute name="driverId" type="xs:date" use="required" />
173   <xs:attribute name="scheduleId" type="xs:int" use="required" />
174   <xs:attribute name="inspectorId" type="xs:int" use="optional" />
175 </xs:complexType>
176
177 <xs:complexType name="route-stop">
178   <xs:attribute name="routeId" type="xs:string" use="required" />
179   <xs:attribute name="stopId" type="xs:string" use="required" />
180 </xs:complexType>
181
182 <!-- egyedileg definiált egyszerű típusok -->
183 <xs:simpleType name="licensePlateNumber">
184   <xs:restriction base="xs:string">
185     <xs:pattern value="[A-Z]{3}-[0-9]{3}" />
186   </xs:restriction>
187 </xs:simpleType>
188 </xs:schema>
189

```

2.Feladat

2.1: Az Java beolvasó és kiíró program.

```
1 package XMLFeladatNDRQ1Y;
2
3 import java.io.*;
4
5 public class DOMReadNDRQ1Y {
6     public static void main(String[] args) throws ParserConfigurationException, SAXException, IOException {
7         // XML fájl beolvasása
8         File xml = new File("XMLNDRQ1Y.xml");
9
10        // XML fájl DOM document való formába való alakítása
11        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
12        DocumentBuilder builder = factory.newDocumentBuilder();
13        Document document = builder.parse(xml);
14
15        // DOM document átalakítása DOM DocumentTraversal formába
16        DocumentTraversal traversal = (DocumentTraversal) document;
17
18        // DOM TreeWalker inicializálása
19        // a gyökérelemről kezdve bejárhatjuk az összes elemet és szöveget tartalmazó
20        // csomópontot
21        TreeWalker walker = traversal.createTreeWalker(document.getDocumentElement(),
22            NodeFilter.SHOW_ELEMENT | NodeFilter.SHOW_TEXT, null, true);
23
24        // a DOM bejárása és kiírása rekurzívan
25        DomTraverser.traverseLevel(walker, "");
26    }
27
28    private static class DomTraverser {
29        public static void traverseLevel(TreeWalker walker, String indent) {
30            // kimentjük az aktuális csomópontot
31            Node node = walker.getCurrentNode();
32
33            // kiíratiuk a megfelelő metódussal
34            if (node.getNodeType() == Node.ELEMENT_NODE) {
35                printElementNode(node, indent);
36            } else {
37                printTextNode(node, indent);
38            }
39
40            // rekurzívan meghívjuk a bejárást a DOM fa egyvel mélyebben lévő csomópontjára,
41            // majd azok testvér csomópontjaira
42            for (Node n = walker.firstChild(); n != null; n = walker.nextSibling()) {
43                traverseLevel(walker, indent + "    ");
44            }
45        }
46    }
47 }
```

```

47         traverseLevel(walker, indent + "    ");
48     }
49
50     walker.setCurrentNode(node);
51 }
52
53 private static void printElementNode(Node node, String indent) {
54     System.out.print(indent + node.getNodeName());
55
56     printElementAttributes(node.getAttributes());
57 }
58
59 private static void printElementAttributes(NamedNodeMap attributes) {
60     int length = attributes.getLength();
61
62     if (length > 0) {
63         System.out.print(" (");
64
65         for (int i = 0; i < length; i++) {
66             Node attribute = attributes.item(i);
67
68             System.out.printf("%s=%s", attribute.getNodeName(), attribute.getNodeValue(),
69                 i != length - 1 ? ", " : "");
70         }
71
72         System.out.println(")");
73     } else {
74         System.out.println();
75     }
76 }
77
78 private static void printTextNode(Node node, String indent) {
79     String content_trimmed = node.getTextContent().trim();
80
81     if (content_trimmed.length() > 0) {
82         System.out.print(indent);
83         System.out.printf("%s\n", content_trimmed);
84     }
85 }
86 }
87 }

```

Kimenet:

```
company (xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance, xsi:noNamespaceSchemaLocation=XMLSchemaNDRQ1Y.xsd)
  inspectors
    inspector (id=2)
      lastName
        Jean
      firstName
        Todt
    inspector (id=4)
      lastName
        Laboda
      firstName
        Dániel
    inspector (id=6)
      lastName
        Elek
      firstName
        Zoltán
  schedules
    schedule (id=1, vehicleId=ASD-123)
      arrivalTime
        05:50:00
      departureTime
        06:15:00
    schedule (id=3, vehicleId=QWE-968)
      arrivalTime
        08:30:00
      departureTime
        08:35:00
    schedule (id=5, vehicleId=ADC-666)
      arrivalTime
        10:17:00
      departureTime
        10:50:00
    schedule (id=7, vehicleId=BOT-404)
      arrivalTime
        12:40:00
      departureTime
        13:00:00
    schedule (id=8, vehicleId=TOP-282)
      arrivalTime
        20:22:00
      departureTime
```



```

    departureTime
      20:50:00
  stops
    stop (id=Bacsó Béla Utca)
      address
        Bacsó Béla út 40.
    stop (id=Soltész Nagy Kálmán)
      address
        Soltész Nagy Kálmán út 8.
    stop (id=Vízügyi igazgatóság)
      address
        Vörösmarty Mihály utca. 77
    stop (id=Miskolctapolca-Barlangfürdő)
      address
        Pazár István Sétány 1.
    stop (id=Római part)
      address
        Római Part 30
  drivers
    driver (id=1111-03-22)
      birthDate
        1969-01-03
      lastName
        Michael
      firstName
        Schumacher
      license
        car
        true
        bus
        true
        tram
        true
    driver (id=1111-03-23)
      birthDate
        1960-03-21
      lastName
        Ayrton
      firstName
        Senna
      license
        car
        true

```

```

        true
        bus
        true
        tram
        false
driver (id=1111-03-24)
  birthDate
    1955-02-24
  lastName
    Alain
  firstName
    Prost
  license
    car
    false
    bus
    false
    tram
    true
driver (id=1111-03-25)
  birthDate
    1987-07-03
  lastName
    Sebastian
  firstName
    Vettel
  license
    car
    false
    bus
    true
    tram
    true
driver (id=1111-03-26)
  birthDate
    1981-07-29
  lastName
    Fernando
  firstName
    Alonso
  license
    car
    true

```

```

        true
        bus
        true
        tram
        false
routes
  route (id=1A, scheduleId=1)
    distance
    12
    pricePerKm
    30.2
  route (id=20B, scheduleId=3)
    distance
    14
    pricePerKm
    30.75
  route (id=43, scheduleId=5)
    distance
    20
    pricePerKm
    40.25
  route (id=1, scheduleId=7)
    distance
    17
    pricePerKm
    25.0
  route (id=12G, scheduleId=8)
    distance
    21
    pricePerKm
    35.0
vehicles
  vehicle (driverId=1111-03-22, id=ASD-123, inspectorId=2, scheduleId=1)
    seats
    48
    manufacturer
    MAN
    type
    bus
  vehicle (driverId=1111-03-23, id=QWE-968, inspectorId=4, scheduleId=3)
    seats
    34
    type
    tram

```

```

    seats
      34
    manufacturer
      Ikarus
    type
      bus
  vehicle (driverId=1111-03-24, id=ADC-666, inspectorId=6, scheduleId=5)
    seats
      70
    manufacturer
      Renault
    type
      bus
  vehicle (driverId=1111-03-25, id=B0T-404, scheduleId=7)
    seats
      230
    manufacturer
      Skoda
    type
      tram
  vehicle (driverId=1111-03-26, id=TOP-282, scheduleId=8)
    seats
      200
    manufacturer
      Skoda
    type
      tram
route-stop_join
  join (routeId=1A, stopId=Bacsó Béla Utca)
  join (routeId=1A, stopId=Római part)
  join (routeId=20B, stopId=Soltész Nagy Kálmán)
  join (routeId=20B, stopId=Vízügyi igazgatóság)
  join (routeId=43, stopId=Soltész Nagy Kálmán)
  join (routeId=43, stopId=Miskolctapolca-Barlangfürdő)
  join (routeId=43, stopId=Római part)
  join (routeId=1, stopId=Vízügyi igazgatóság)
  join (routeId=1, stopId=Miskolctapolca-Barlangfürdő)
  join (routeId=12G, stopId=Bacsó Béla Utca)
  join (routeId=12G, stopId=Vízügyi igazgatóság)
  join (routeId=12G, stopId=Római part)

```

2.2: Az Java módosító program.

```
1 package XMLFeladatNDRQ1Y;
2
3 import java.io.*;
4
15 public class DOMModifyNDRQ1Y {
16     public static void main(String[] args) throws ParserConfigurationException, SAXException, IOException,
17         XPathExpressionException, DOMException, ParseException {
18         // XML fájl beolvasása
19         File xml = new File("XMLNDRQ1Y.xml");
20
21         // XML fájl DOM document való formába való alakítása
22         DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
23         DocumentBuilder builder = factory.newDocumentBuilder();
24         Document document = builder.parse(xml);
25
26         // a DOM document lévő adatok módosítása
27         DomModifier.modifyDom(document);
28
29         // DOM document átalakítása DOM DocumentTraversal formába
30         DocumentTraversal traversal = (DocumentTraversal) document;
31
32         // DOM TreeWalker inicializálása
33         // a gyökérelmentől kezdve bejárhatjuk az összes elemet és szöveget tartalmazó
34         // csomópontot
35         TreeWalker walker = traversal.createTreeWalker(document.getDocumentElement(),
36             NodeFilter.SHOW_ELEMENT | NodeFilter.SHOW_TEXT, null, true);
37
38         // a DOM bejárása rekurzívan
39         DomTraverser.traverseLevel(walker, "");
40     }
41
42     private static class DomModifier {
43         public static void modifyDom(Document document) throws XPathExpressionException, DOMException, ParseException {
44             XPathFactory factory = XPathFactory.newInstance();
45             XPath xpath = factory.newXPath();
46
47             // 1.) Michael Schumacher megszervezte a villamoshoz a jogosítványt,
48             // és ezt kellene hozzáadnunk az adatbázisunkhoz
49             // XPath segítségével lekérdezzük a megfelelő elemet/csomópontot a DOM documentból
50             Node driver = (Node) xpath.evaluate("//driver[./lastName='Michael' and ./firstName='Schumacher']/license/bus",
51                 document, XPathConstants.NODE);
52
53             driver.setTextContent("true");
54
55             // 2.) Egy városi rendelet szerint a 15 km-nél rövidebb útvonalaknak 20%-kal nő
56             // a kilométerára
57             NodeList routes = (NodeList) xpath.evaluate("//route[./distance<15]/pricePerKm", document,
58                 XPathConstants.NODESET);
59
60             for (int i = 0; i < routes.getLength(); i++) {
61                 Node route = routes.item(i);
62
63                 double pricePerKm = Double.parseDouble(route.getTextContent());
64                 route.setTextContent(Double.toString(pricePerKm * 1.20));
65             }
66
67             // 3.) A 08:35-kor beérkező járatnak 15 perccel nőtt a menetideje, ezért később fog beérkezni
68             Node schedule = (Node) xpath.evaluate("//schedule[./departureTime='08:35:00']/departureTime", document,
69                 XPathConstants.NODE);
70
71             DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm:ss");
72             LocalTime departureTime = LocalTime.parse(schedule.getTextContent(), formatter);
73             departureTime = departureTime.plusMinutes(15);
74             schedule.setTextContent(formatter.format(departureTime));
75
76             // 4.) A 43-es járat át lett nevezve, mivel ez egy ID ezért az ehhez kapcsolódó
77             // elemekben is módosítanunk kell a kapcsolatot
78             NodeList nodes = (NodeList) xpath.evaluate("//route[@id='43']/@id | /*[@routeId='43']/@routeId", document,
79                 XPathConstants.NODESET);
80             String newId = "666";
81
82             for (int i = 0; i < nodes.getLength(); i++) {
83                 Node node = nodes.item(i);
84
85                 node.setTextContent(newId);
86             }
87         }
88     }
89
90     private static class DomTraverser {
91         public static void traverseLevel(TreeWalker walker, String indent) {
92             // kimentjük az aktuális csomópontot
93             Node node = walker.getCurrentNode();
94
95             // kiirattjuk a megfelelő metódussal
96             if (node.getNodeType() == Node.ELEMENT_NODE) {
```

```

96         if (node.getNodeType() == Node.ELEMENT_NODE) {
97             printElementNode(node, indent);
98         } else {
99             printTextNode(node, indent);
100         }
101
102         // rekurzívan meghívjuk a bejárást a DOM fa eggyel mélyebben lévő csomópontjára,
103         // majd azok testvér csomópontjaira
104         for (Node n = walker.firstChild(); n != null; n = walker.nextSibling()) {
105             traverseLevel(walker, indent + "    ");
106         }
107
108         walker.setCurrentNode(node);
109     }
110
111     private static void printElementNode(Node node, String indent) {
112         System.out.print(indent + node.getNodeName());
113
114         printElementAttributes(node.getAttributes());
115     }
116
117     private static void printElementAttributes(NamedNodeMap attributes) {
118         int length = attributes.getLength();
119
120         if (length > 0) {
121             System.out.print(" (");
122
123             for (int i = 0; i < length; i++) {
124                 Node attribute = attributes.item(i);
125
126                 System.out.printf("%s=%s", attribute.getNodeName(), attribute.getNodeValue(),
127                     i != length - 1 ? ", " : "");
128             }
129
130             System.out.println(")");
131         } else {
132             System.out.println();
133         }
134     }
135
136     private static void printTextNode(Node node, String indent) {
137         String content_trimmed = node.getTextContent().trim();
138

```

```

136     private static void printTextNode(Node node, String indent) {
137         String content_trimmed = node.getTextContent().trim();
138
139         if (content_trimmed.length() > 0) {
140             System.out.print(indent);
141             System.out.printf("%s\n", content_trimmed);
142         }
143     }
144 }
145 }

```

Kimenet:

```
2010an
schedules
  schedule (id=1, vehicleId=ASD-123)
    arrivalTime
      05:50:00
    departureTime
      06:15:00
  schedule (id=3, vehicleId=QWE-968)
    arrivalTime
      08:30:00
    departureTime
      08:50:00
  schedule (id=5, vehicleId=ADC-666)
    arrivalTime
      10:17:00
    departureTime
      10:50:00
  schedule (id=7, vehicleId=BOT-404)
    arrivalTime
      12:40:00
    departureTime
      13:00:00
  schedule (id=8, vehicleId=TOP-282)
    arrivalTime
      20:22:00
    departureTime
      20:50:00
```

```
route (id=1A, scheduleId=1)
  distance
    12
  pricePerKm
    36.239999999999995
route (id=20B, scheduleId=3)
  distance
    14
  pricePerKm
    36.9
route (id=666, scheduleId=5)
  distance
    20
  pricePerKm
    40.25
route (id=1, scheduleId=7)
  distance
    17
  pricePerKm
    25.0
route (id=12G, scheduleId=8)
  distance
    21
  pricePerKm
    35.0
```

```
driver (id=1111-03-22)
  birthDate
    1969-01-03
  lastName
    Michael
  firstName
    Schumacher
  license
    car
      true
    bus
      true
    tram
      true
```

2.3: Az Java lekérdező program.

```
1 package XMLFeladatNDRQ1Y;
2
3 import java.io.File;
4
5 public class DOMQueryNDRQ1Y {
6     public static void main(String[] args) throws ParserConfigurationException, SAXException, IOException,
7         XPathExpressionException, DOMException, ParseException {
8         // XML fájl beolvasása
9         File xml = new File("XMLNDRQ1Y.xml");
10
11         // XML fájl DOM document való formába való alakítása
12         DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
13         DocumentBuilder builder = factory.newDocumentBuilder();
14         Document document = builder.parse(xml);
15
16         // DOM document átalakítása DOM DocumentTraversal formába
17         DocumentTraversal traversal = (DocumentTraversal) document;
18
19         NodeList vehicles = document.getElementsByTagName("vehicle");
20
21         // Lekérdezzük azokat a járműveket amelyeknek a gyártója Skoda.
22         for (int i = 0; i < vehicles.getLength(); i++) {
23             Node temp = vehicles.item(i);
24             NodeList tempchildren = temp.getChildNodes();
25             for (int j = 0; j < tempchildren.getLength(); j++) {
26                 Node temp2 = tempchildren.item(j);
27                 if (temp2.getNodeName().equals("manufacturer") && temp2.getTextContent().equals("Skoda")) {
28                     TreeWalker walker = traversal.createTreeWalker(temp,
29                         NodeFilter.SHOW_ELEMENT | NodeFilter.SHOW_TEXT, null, true);
30                     DomTraverser.traverseLevel(walker, "");
31                 }
32             }
33         }
34
35         // Lekérdezzük azokat a sofőröket akiknek van jogosítványuk villamosra.
36         NodeList drivers = document.getElementsByTagName("driver");
37
38         for (int i = 0; i < drivers.getLength(); i++) {
39             Node temp = drivers.item(i);
40             NodeList tempchildren = temp.getChildNodes();
41             for (int j = 0; j < tempchildren.getLength(); j++) {
```



```

61     Node temp2 = tempchildren.item(j);
62     if (temp2.getNodeName().equals("license")) {
63         NodeList temp2children = temp2.getChildNodes();
64         for (int k = 0; k < temp2children.getLength(); k++) {
65             Node temp3 = temp2children.item(k);
66
67             if (temp3.getNodeName().equals("tram") && temp3.getTextContent().equals("true")) {
68
69                 TreeWalker walker = traversal.createTreeWalker(temp,
70                     NodeFilter.SHOW_ELEMENT | NodeFilter.SHOW_TEXT, null, true);
71                 DomTraverser.traverseLevel(walker, "");
72             }
73         }
74     }
75 }
76
77 }
78 }
79
80
81
82
83 }
84 private static class DomTraverser {
85     public static void traverseLevel(TreeWalker walker, String indent) {
86         // kimentjük az aktuális csomópontot
87         Node node = walker.getCurrentNode();
88
89         // kiíratiuk a megfelelő metódussal
90         if (node.getNodeType() == Node.ELEMENT_NODE) {
91             printElementNode(node, indent);
92         } else {
93             printTextNode(node, indent);
94         }
95
96         // rekurzívan meghívjuk a bejárást a DOM fa eggyel mélyebben lévő csomópontjára,
97         // majd azok testvér csomópontjaira
98         for (Node n = walker.firstChild(); n != null; n = walker.nextSibling()) {
99             traverseLevel(walker, indent + "    ");
100         }
101
102         walker.setCurrentNode(node);
103     }

```

```

104
105● private static void printElementNode(Node node, String indent) {
106     System.out.print(indent + node.getNodeName());
107
108     printElementAttributes(node.getAttributes());
109 }
110
111● private static void printElementAttributes(NamedNodeMap attributes) {
112     int length = attributes.getLength();
113
114     if (length > 0) {
115         System.out.print(" (");
116
117         for (int i = 0; i < length; i++) {
118             Node attribute = attributes.item(i);
119
120             System.out.printf("%s=%s%s", attribute.getNodeName(), attribute.getNodeValue(),
121                 i != length - 1 ? ", " : "");
122         }
123
124         System.out.println(")");
125     } else {
126         System.out.println();
127     }
128 }
129
130● private static void printTextNode(Node node, String indent) {
131     String content_trimmed = node.getTextContent().trim();
132
133     if (content_trimmed.length() > 0) {
134         System.out.print(indent);
135         System.out.printf("%s\n", content_trimmed);
136     }
137 }
138 }
139 }

```

Kimenet:

```
vehicle (driverId=1111-03-25, id=B0T-404, scheduleId=7)
  seats
    230
  manufacturer
    Skoda
  type
    tram
vehicle (driverId=1111-03-26, id=TOP-282, scheduleId=8)
  seats
    200
  manufacturer
    Skoda
  type
    tram
driver (id=1111-03-22)
  birthDate
    1969-01-03
  lastName
    Michael
  firstName
    Schumacher
  license
    car
      true
    bus
      true
    tram
      true
```

```
driver (id=1111-03-24)
  birthDate
    1955-02-24
  lastName
    Alain
  firstName
    Prost
  license
    car
      false
    bus
      false
    tram
      true
driver (id=1111-03-25)
  birthDate
    1987-07-03
  lastName
    Sebastian
  firstName
    Vettel
  license
    car
      false
    bus
      true
    tram
      true
```