JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Cukrászda

Készítette: Nagy Róbert

Neptunkód: JMDRGG

Dátum: 2022.11.13

Tartalomjegyzék

A feladat leírása:

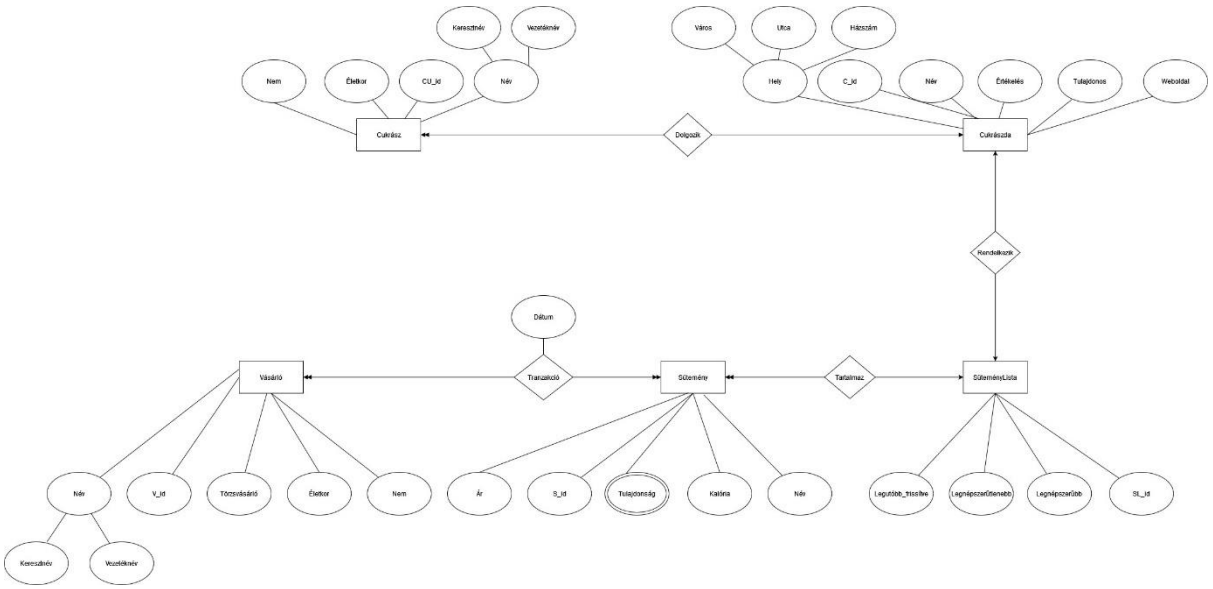
A feladat témája cukrászdák termékeinek és tranzakcióinak nyilvántartása. Az adatbázis nyilvántartja a cukrászdákat, a cukrászdák által kínált süteményeket, a cukrászokat, akik a cukrászdánál dolgoznak, valamint a vásárlókat, akik tranzakciókat hoznak létre.

Az adatbázis 6 darab táblából áll: Cukrász, Cukrászda, SüteményLista, Sütemény, Vásárló. A Cukrász tábla rekordjai CU_id elsődleges kulccsal rendelkeznek, mezői: nem, életkor, összetett név tulajdonság, amelynek két része van, keresztnév és vezetéknév. A Cukrászda tábla rekordjai C_id elsődleges kulccsal rendelkeznek, mezői: név, értékelés, tulajdonos, weboldal és összetett tulajdonság hely, amely város, utca és házszám tulajdonságokból áll. A cukrász és cukrászda között több az egyhez kapcsolat van. A SüteményLista tábla SL_id elsődleges kulccsal rendelkeznek, és három darab mezőből áll, legutóbb_frissítve, legnépszerűbb és legnépszerűtlenebb. A cukrászda és sütemény lista között egy az egyhez kapcsolat van. A sütemény tábla S_id elsődleges kulccsal rendelkezik, három darab tulajdonsága van: ár, kalória és név, valamint rendelkezik egy tulajdonság többérékű tulajdonsággal is. A sütemény és a sütemény lista között több az egyhez kapcsolat van. A vásárló tábla V_id elsődleges kulccsal rendelkezik, három darab tulajdonsága van: törzsvásárló, életkor, nem és egy összetett tulajdonság: név, amely keresztnév és vezetéknév tulajdonságokból áll. A vásárló és sütemény lista között több-több kapcsolat van. A tranzakció tábla a sütemény és a vásárló kapcsolat kapcsolótáblája, amely dátum tulajdonsággal rendelkezik.

1. feladat

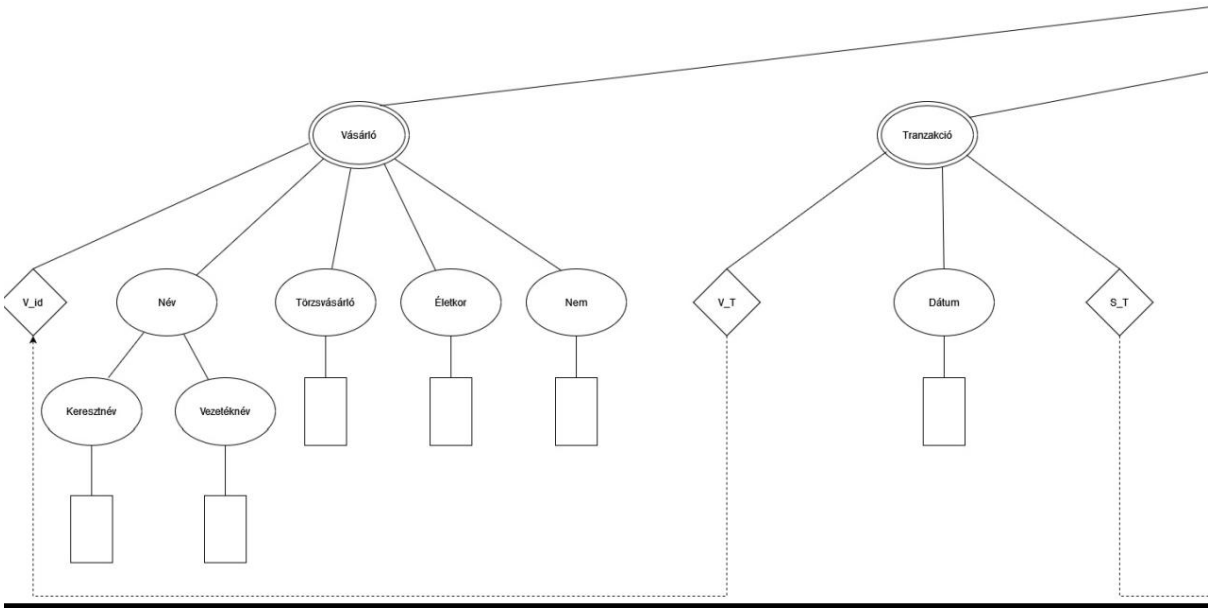
1a) Az adatbázis ER modell

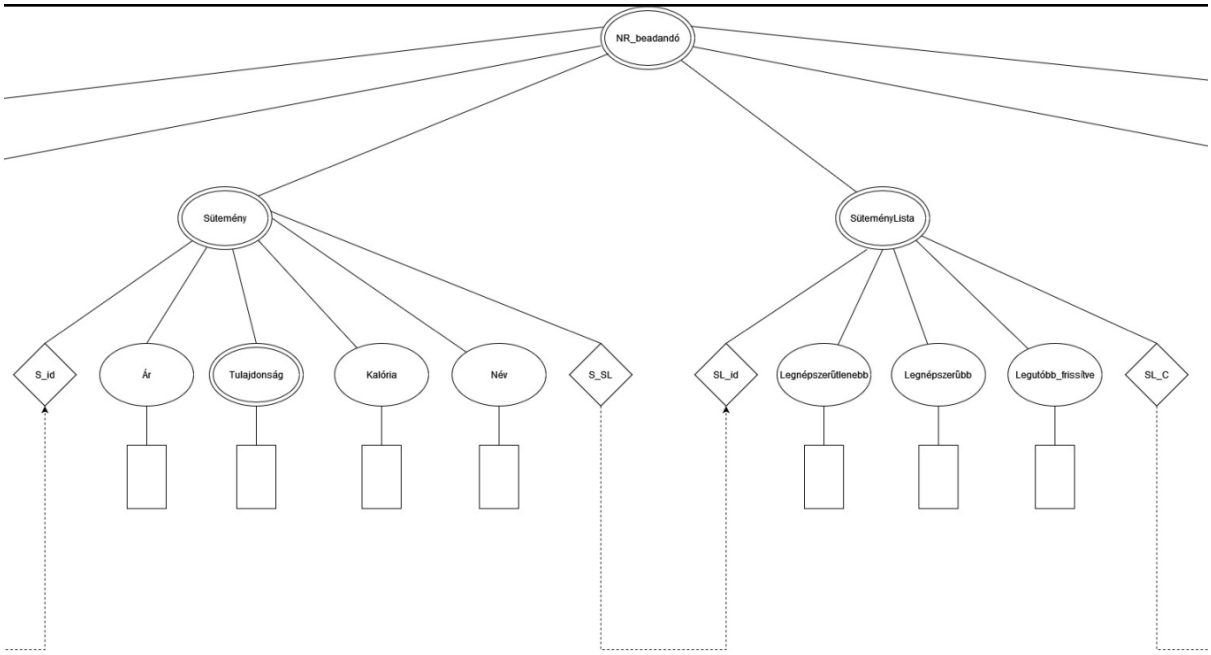
Az adatbázis átkonvertálása során 5 darab elem és 4 darab kapcsolat jött létre.

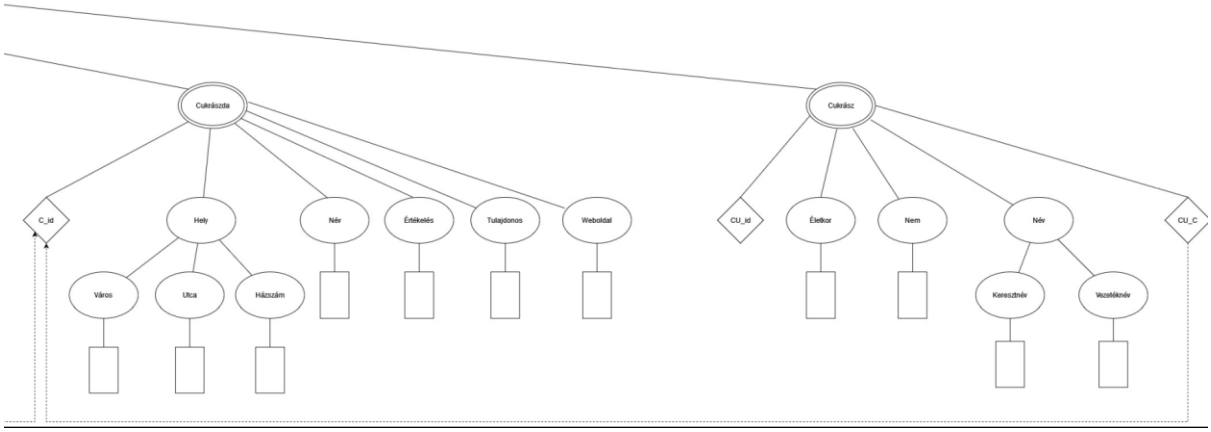


1b) Az adatbázis konvertálása XDM modellre

Az XDM modell 8 darab többértékű, 3 darab összetett és 23 darab sima elemből és 10 darab attribútumból áll. A gyökérelem neve NR_beadandó. A tranzakció elem rendelkezik v_t és s_t idegen kulcsokkal. A sütemény elem rendelkezik egy s_sl idegen kulccsal. A sütemény lista elem rendelkezik sl_c idegen kulccsal. A cukrász elem rendelkezik cu_c idegen kulccsal.







1c) Az XDM modell alapján XML dokumentum készítése:

```xml
<NR_beadando xsi:noNamespaceSchemaLocation="XMLSchemaJmdrgg.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <!-- Cukrászdák -->
  <cukraszda c_id="c1">
    <nev>Daubner Cukrászda</nev>
    <hely>
      <varos>Budapest</varos>
      <utca>Szépvölgyi </utca>
      <hazszam>50</hazszam>
    </hely>
    <ertekeles>4.3</ertekeles>
    <tulajdonos>Pista Laci</tulajdonos>
    <weboldal>daubnercukraszda.hu</weboldal>
  </cukraszda>
  <cukraszda c_id="c2">
    <nev>Frey Cukrászda</nev>
    <hely>
      <varos>Pécs</varos>
      <utca>Hungária</utca>
      <hazszam>53/1</hazszam>
    </hely>
    <ertekeles>4.7</ertekeles>
    <tulajdonos>Rotyis Bálint</tulajdonos>
    <weboldal>freycukraszda.hu</weboldal>
  </cukraszda>
  <cukraszda c_id="c3">
    <nev>Szűőke cukrászda</nev>
    <hely>
      <varos>Debrecen</varos>
      <utca>Úrrétje</utca>
      <hazszam>14</hazszam>
    </hely>
    <ertekeles>4.5</ertekeles>
```

```xml
<!-- Cukrászok -->
<cukrasz cu_id="cu1" cu_c="c1">
  <nev>
    <vezeteknev>Tompa</vezeteknev>
    <keresztnev>Tamás</keresztnev>
  </nev>
  <eletkor>57</eletkor>
  <nem>férfi</nem>
</cukrasz>
<cukrasz cu_id="cu2" cu_c="c1">
  <nev>
    <vezeteknev>Patkós</vezeteknev>
    <keresztnev>Petra</keresztnev>
  </nev>
  <eletkor>21</eletkor>
  <nem>nő</nem>
</cukrasz>
<cukrasz cu_id="cu3" cu_c="c1">
  <nev>
    <vezeteknev>Nagy</vezeteknev>
    <keresztnev>Bence</keresztnev>
  </nev>
  <eletkor>18</eletkor>
  <nem>férfi</nem>
</cukrasz>
<cukrasz cu_id="cu4" cu_c="c2">
  <nev>
    <vezeteknev>Szabó</vezeteknev>
    <keresztnev>Bence</keresztnev>
  </nev>
  <eletkor>21</eletkor>
```

```xml
<!-- Sütemény listák -->
<sutemeny_lista sl_id="sl1" sl_c="c1">
  <legnepszerubb>Mignon</legnepszerubb>
  <legnepszerutlenebb>Almás pite</legnepszerutlenebb>
  <legutobb_frissitve>2022-04-26</legutobb_frissitve>
</sutemeny_lista>
<sutemeny_lista sl_id="sl2" sl_c="c2">
  <legnepszerubb>Képviselőfánk</legnepszerubb>
  <legnepszerutlenebb>Piskótatekercs</legnepszerutlenebb>
  <legutobb_frissitve>2022-06-22</legutobb_frissitve>
</sutemeny_lista>
<sutemeny_lista sl_id="sl3" sl_c="c3">
  <legnepszerubb>Joghurtos epertorta szelet</legnepszerubb>
  <legnepszerutlenebb>Somlói galuska</legnepszerutlenebb>
  <legutobb_frissitve>2022-03-20</legutobb_frissitve>
</sutemeny_lista>
```

```xml
<!-- Sütemények -->
<sutemeny s_id="s1" s_sl="sl1">
  <nev>Almás pite</nev>
  <ar>740</ar>
  <kaloria>296</kaloria>
  <tulajdonsag>klasszikus</tulajdonsag>
  <tulajdonsag>fahéjas</tulajdonsag>
  <tulajdonsag>almás</tulajdonsag>
  <tulajdonsag>pite</tulajdonsag>
</sutemeny>
<sutemeny s_id="s2" s_sl="sl1">
  <nev>Mandarinos túrótorta szelet</nev>
  <ar>990</ar>
  <kaloria>360</kaloria>
  <tulajdonsag>csokoládés piskóta</tulajdonsag>
  <tulajdonsag>gyümölcsös</tulajdonsag>
  <tulajdonsag>klasszikus tortaszelet</tulajdonsag>
  <tulajdonsag>tejszínes</tulajdonsag>
  <tulajdonsag>torta szelet</tulajdonsag>
  <tulajdonsag>túrós</tulajdonsag>
</sutemeny>
<sutemeny s_id="s3" s_sl="sl1">
  <nev>Cukormentes málnatorta szelet</nev>
  <ar>860</ar>
  <kaloria>200</kaloria>
  <tulajdonsag>Cukormentes </tulajdonsag>
  <tulajdonsag>málnás</tulajdonsag>
  <tulajdonsag>Mandulás</tulajdonsag>
  <tulajdonsag>piskóta</tulajdonsag>
  <tulajdonsag>torta szelet</tulajdonsag>
</sutemeny>
<sutemeny s_id="s4" s_sl="sl1">
  <nev>Mignon</nev>
```

```xml
<!-- Vásárlók -->
<vasarlo v_id="v1">
  <nev>
    <vezeteknev>Nagy</vezeteknev>
    <keresztnev>Róbert</keresztnev>
  </nev>
  <torzsvasarlo>igen</torzsvasarlo>
  <eletkor>21</eletkor>
  <nem>férfi</nem>
</vasarlo>
<vasarlo v_id="v2">
  <nev>
    <vezeteknev>Sándor</vezeteknev>
    <keresztnev>Béla</keresztnev>
  </nev>
  <torzsvasarlo>nem</torzsvasarlo>
  <eletkor>26</eletkor>
  <nem>nő</nem>
</vasarlo>
<vasarlo v_id="v3">
  <nev>
    <vezeteknev>Ferenc</vezeteknev>
    <keresztnev>Viktor</keresztnev>
  </nev>
  <torzsvasarlo>nem</torzsvasarlo>
  <eletkor>56</eletkor>
  <nem>férfi</nem>
</vasarlo>
```

```
<!-- Tranzakciók -->
<tranzakcio v_t="v1" s_t="s1">
  <datum>2022-03-20</datum>
</tranzakcio>
<tranzakcio v_t="v1" s_t="s2">
  <datum>2022-03-20</datum>
</tranzakcio>
<tranzakcio v_t="v1" s_t="s4">
  <datum>2022-04-01</datum>
</tranzakcio>
<tranzakcio v_t="v1" s_t="s5">
  <datum>2022-04-01</datum>
</tranzakcio>
<tranzakcio v_t="v1" s_t="s9">
  <datum>2022-04-01</datum>
</tranzakcio>
<tranzakcio v_t="v2" s_t="s2">
  <datum>2022-04-12</datum>
</tranzakcio>
<tranzakcio v_t="v2" s_t="s1">
  <datum>2022-04-12</datum>
</tranzakcio>
<tranzakcio v_t="v2" s_t="s4">
  <datum>2022-05-02</datum>
</tranzakcio>
<tranzakcio v_t="v2" s_t="s1">
  <datum>2022-09-11</datum>
</tranzakcio>
```

1d) Az XML dokumentum alapján XMLSchema készítése

Az xml sémában 7 darab egyszerű típus van:

torzsvasarlo_type két értéke lehet, igen vagy nem.

eletkor_type legalább 1

nem_type két értéke lehet, férfi vagy nő.

ar_type: legalább 1

kaloria_type: 0 és 4000 között.

ertekeles_type: 0 és 10 között.

weboldal_type: megfelelő formátumúnak kell lennie.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- egyszeru tipusok -->
  <xs:simpleType name="torzsvasarlo_type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="igen"></xs:enumeration>
      <xs:enumeration value="nem"></xs:enumeration>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="eletkor_type">
    <xs:restriction base="xs:int">
      <xs:minExclusive value="0" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="nem_type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="nő" />
      <xs:enumeration value="férfi" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="ar_type">
    <xs:restriction base="xs:int">
      <xs:minExclusive value="0" />
    </xs:restriction>
  </xs:simpleType>
```

```
<xs:simpleType name="kaloria_type">
  <xs:restriction base="xs:int">
    <xs:minExclusive value="0" />
    <xs:maxInclusive value="4000" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ertekeles_type">
  <xs:restriction base="xs:float">
    <xs:minInclusive value="1" />
    <xs:maxInclusive value="10" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="weboldal_type">
  <xs:restriction base="xs:string">
    <xs:pattern value="[-a-zA-Z0-9@:%._\+~#=]{1,256}\.[a-zA-Z]{1,6}" />
  </xs:restriction>
</xs:simpleType>
```

```
<!— attributumok —>
<xs:attribute name="v_id" type="xs:NCName" />
<xs:attribute name="v_t" type="xs:NCName" />
<xs:attribute name="s_t" type="xs:NCName" />
<xs:attribute name="s_id" type="xs:NCName" />
<xs:attribute name="s_sl" type="xs:NCName" />
<xs:attribute name="sl_id" type="xs:NCName" />
<xs:attribute name="sl_c" type="xs:NCName" />
<xs:attribute name="c_id" type="xs:NCName" />
<xs:attribute name="cu_id" type="xs:NCName" />
<xs:attribute name="cu_c" type="xs:NCName" />
```

```
<!— egyszeru egyedek —>
<xs:element name="keresztnev" type="xs:string" />
<xs:element name="vezeteknev" type="xs:string" />
<xs:element name="torzsvasarlo" type="torzsvasarlo_type" default="nem" />
<xs:element name="eletkor" type="eletkor_type" />
<xs:element name="nem" type="nem_type" default="nő" />
<xs:element name="datum" type="xs:date" />
<xs:element name="ar" type="ar_type" />
<xs:element name="tulajdonsag" type="xs:string" />
<xs:element name="kaloria" type="kaloria_type" />
<xs:element name="nev" type="xs:string" />
<xs:element name="varos" type="xs:string" />
<xs:element name="utca" type="xs:string" />
<xs:element name="hazszam" type="xs:string" />
<xs:element name="legnepszerubb" type=" xs:string" />
<xs:element name="legnepszerutlenebb" type=" xs:string" />
<xs:element name="legutobb_frissitve" type=" xs:date" />
<xs:element name="ertekeles" type=" ertekeles_type" />
<xs:element name="tulajdonos" type=" xs:string" />
<xs:element name="weboldal" type=" weboldal_type" />
```

```
<!-- komplex tipusok -->
<xs:complexType name="nev_type">
  <xs:sequence>
    <xs:element ref="vezeteknev" />
    <xs:element ref="keresztnev" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="hely_type">
  <xs:sequence>
    <xs:element ref="varos" />
    <xs:element ref="utca" />
    <xs:element ref="hazszam" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="vasarlo_type">
  <xs:sequence>
    <xs:element name="nev" type="nev_type" />
    <xs:element ref="torzsvasarlo" />
    <xs:element ref="eletkor" />
    <xs:element ref="nem" />
  </xs:sequence>
  <xs:attribute ref="v_id" use="required" />
</xs:complexType>
```

```
<xs:complexType name="tranzakcio_type">
  <xs:sequence>
    <xs:element ref="datum" />
  </xs:sequence>
  <xs:attribute ref="v_t" use="required" />
  <xs:attribute ref="s_t" use="required" />
</xs:complexType>

<xs:complexType name="sutemeny_type">
  <xs:sequence>
    <xs:element ref="nev" />
    <xs:element ref="ar" />
    <xs:element ref="kaloria" />
    <xs:element ref="tulajdonsag" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute ref="s_id" use="required" />
  <xs:attribute ref="s_sl" use="required" />
</xs:complexType>

<xs:complexType name="sutemeny_lista_type">
  <xs:sequence>
    <xs:element ref="legnepszerubb" />
    <xs:element ref="legnepszerutlenebb" />
    <xs:element ref="legutobb_frissitve" minOccurs="1" />
  </xs:sequence>
  <xs:attribute ref="sl_id" use="required" />
  <xs:attribute ref="sl_c" use="required" />
</xs:complexType>
```

```xml
<xs:complexType name="cukraszda_type">
  <xs:sequence>
    <xs:element ref="nev" minOccurs="1" />
    <xs:element name="hely" type="hely_type" minOccurs="1" />
    <xs:element ref="ertekeles" />
    <xs:element ref="tulajdonos" />
    <xs:element ref="weboldal" />
  </xs:sequence>
  <xs:attribute ref="c_id" use="required" />
</xs:complexType>

<xs:complexType name="cukrasz_type">
  <xs:sequence>
    <xs:element name="nev" type="nev_type" minOccurs="1" />
    <xs:element ref="eletkor" />
    <xs:element ref="nem" />
  </xs:sequence>
  <xs:attribute ref="cu_id" use="required" />
  <xs:attribute ref="cu_c" use="required" />
</xs:complexType>
```

```xml
<!— root —>
<xs:element name="NR_beadando">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="cukraszda" type="cukraszda_type" minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="cukrasz" type="cukrasz_type" minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="sutemeny_lista" type="sutemeny_lista_type" minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="sutemeny" type="sutemeny_type" minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="vasarlo" type="vasarlo_type" minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="tranzakcio" type="tranzakcio_type" minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
```

```
<!-- kulcsok -->
<xs:key name="cukraszda_key">
  <xs:selector xpath="cukraszda" />
  <xs:field xpath="@c_id" />
</xs:key>

<xs:keyref name="cukrasz_cukraszda_key" refer="cukraszda_key">
  <xs:selector xpath="cukrasz" />
  <xs:field xpath="@cu_c" />
</xs:keyref>

<xs:keyref name="sutemeny_lista_cukraszda_key" refer="cukraszda_key">
  <xs:selector xpath="sutemeny_lista" />
  <xs:field xpath="@sl_c" />
</xs:keyref>

<xs:key name="sutemeny_lista_key">
  <xs:selector xpath="sutemeny_lista" />
  <xs:field xpath="@sl_id" />
</xs:key>

<xs:keyref name="sutemeny_sutemeny_lista_key" refer="sutemeny_lista_key">
  <xs:selector xpath="sutemeny" />
  <xs:field xpath="@s_sl" />
</xs:keyref>

<xs:key name="vasarlo_key">
  <xs:selector xpath="vasarlo" />
  <xs:field xpath="@v_id" />
</xs:key>
```

```
  <xs:keyref name="vasarlo_tranzakcio_key" refer="vasarlo_key">
    <xs:selector xpath="tranzakcio" />
    <xs:field xpath="@v_t"></xs:field>
  </xs:keyref>

  <xs:key name="sutemeny_key">
    <xs:selector xpath="sutemeny" />
    <xs:field xpath="@s_id" />
  </xs:key>

  <xs:keyref name="sutemeny_tranzakcio_key" refer="sutemeny_key">
    <xs:selector xpath="tranzakcio" />
    <xs:field xpath="@s_t"></xs:field>
  </xs:keyref>

 </xs:element>
</xs:schema>
```

2. feladat

2a) adatolvasás

DomReadJmdrgg.java: Egyetlen privát Document típusú változóval rendelkező objektum, amely a cukrászdákról szóló xml fájl kiírására szolgál.

```java
package hu.domparse.jmdrgg;

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

public class DomReadJmdrgg {

  private Document doc;

  /**
   * Beadandó feladat xml dokumentum kiírására szolgáló Objektum
   * @param doc Dokumentum Objektum
   */
  public DomReadJmdrgg(Document doc) {
    this.doc = doc;
  }
```

```java
  /**
   * Cukrászdák kiírása
   * @param nl Cukrászdák NodeList objektum
   */
  public void printCukraszdak(NodeList nl) {
    System.out.println(x: "Cukrászdák: ");
    for (int i = 0; i < nl.getLength(); i++) {
      Element elem = (Element) nl.item(i);

      System.out.println("id: " + elem.getAttribute(name: "c_id"));

      System.out.println(
        "Cukrászda neve: " +
        elem.getElementsByTagName(name: "nev").item(index: 0).getTextContent()
      );

      System.out.println(
        "Címe: " +
        elem.getElementsByTagName(name: "varos").item(index: 0).getTextContent() +
        " " +
        elem.getElementsByTagName(name: "utca").item(index: 0).getTextContent() +
        " " +
        elem.getElementsByTagName(name: "hazszam").item(index: 0).getTextContent()
      );

      System.out.println(
        "Értékelése: " +
        elem.getElementsByTagName(name: "ertekeles").item(index: 0).getTextContent()
      );

      System.out.println(
        "Weboldala: " +
        elem.getElementsByTagName(name: "weboldal").item(index: 0).getTextContent()
      );
    }
```

```java
/**
 * Cukrászok kiírása
 * @param nl Cukrászok NodeList objektum
 */
public void printCukrasz(NodeList nl) {
  System.out.println(x: "Cukrászok: ");

  for (int i = 0; i < nl.getLength(); i++) {
    Element elem = (Element) nl.item(i);

    System.out.println("id: " + elem.getAttribute(name: "cu_id"));

    System.out.println(
      "Cukrász neve: " +
      elem.getElementsByTagName(name: "vezeteknev").item(index: 0).getTextContent() +
      " " +
      elem.getElementsByTagName(name: "keresztnev").item(index: 0).getTextContent()
    );

    System.out.println(
      "Életkora: " +
      elem.getElementsByTagName(name: "eletkor").item(index: 0).getTextContent()
    );

    System.out.println(
      "Neme: " + elem.getElementsByTagName(name: "nem").item(index: 0).getTextContent()
    );
  }
}
```

```java
/**
 * Sütemény listák kiírása
 * @param nl Sütemény lista NodeList objektum
 */
public void printSutemenyLista(NodeList nl) {
  System.out.println(x: "Sütemény listák: ");

  for (int i = 0; i < nl.getLength(); i++) {
    Element elem = (Element) nl.item(i);

    System.out.println("id: " + elem.getAttribute(name: "sl_id"));

    System.out.println(
      "Legnépszerűbb sütemény: " +
      elem.getElementsByTagName(name: "legnepszerubb").item(index: 0).getTextContent()
    );

    System.out.println(
      "Legnépszerűtlenebb sütemény: " +
      elem.getElementsByTagName(name: "legnepszerutlenebb").item(index: 0).getTextContent()
    );

    System.out.println(
      "Legutobb frissitve: " +
      elem.getElementsByTagName(name: "legutobb_frissitve").item(index: 0).getTextContent()
    );
  }
}
```

```java
 * Sütemények kiírása
 * @param nl Sütemény NodeList objektum
 */
public void printSutemeny(NodeList nl) {
  System.out.println(x: "Sütemények: ");

  for (int i = 0; i < nl.getLength(); i++) {
    Element elem = (Element) nl.item(i);

    System.out.println("id: " + elem.getAttribute(name: "s_id"));

    System.out.println(
      "Név: " + elem.getElementsByTagName(name: "nev").item(index: 0).getTextContent()
    );

    System.out.println(
      "Ár: " + elem.getElementsByTagName(name: "ar").item(index: 0).getTextContent()
    );

    System.out.println(
      "Kalória: " +
      elem.getElementsByTagName(name: "kaloria").item(index: 0).getTextContent()
    );

    int len = elem.getElementsByTagName(name: "tulajdonsag").getLength();

    System.out.println(x: "tulajdonságok: ");
    for (int j = 0; j < len; j++) {
      System.out.println(
        elem.getElementsByTagName(name: "tulajdonsag").item(j).getTextContent()
      );
    }
  }
}
```

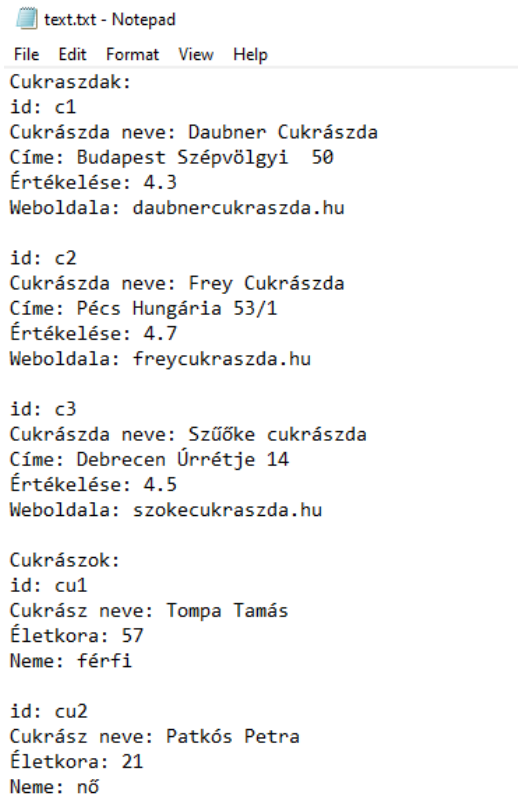
```java
/**
 * Vásárlók kiírása
 * @param nl Vásárló NodeList objektum
 */
public void printVasarlo(NodeList nl) {
  System.out.println(x: "Vásárlók: ");

  for (int i = 0; i < nl.getLength(); i++) {
    Element elem = (Element) nl.item(i);

    System.out.println("id: " + elem.getAttribute(name: "v_id"));

    System.out.println(
      "Név: " +
      elem.getElementsByTagName(name: "vezeteknev").item(index: 0).getTextContent() +
      " " +
      elem.getElementsByTagName(name: "keresztnev").item(index: 0).getTextContent()
    );

    System.out.println(
      "Törzsvásárló: " +
      elem.getElementsByTagName(name: "torzsvasarlo").item(index: 0).getTextContent()
    );

    System.out.println(
      "Életkor: " +
      elem.getElementsByTagName(name: "eletkor").item(index: 0).getTextContent()
    );

    System.out.println(
      "Nem: " + elem.getElementsByTagName(name: "nem").item(index: 0).getTextContent()
    );
  }
}
```

```java
/**
 * Tranzakciók kiírása
 * @param nl Tranzakció NodeList objektum
 */
public void printTranzakcio(NodeList nl) {
  System.out.println(x: "Tranzakciók: ");

  for (int i = 0; i < nl.getLength(); i++) {
    Element elem = (Element) nl.item(i);

    System.out.println(
      "Dátum: " + elem.getElementsByTagName(name: "datum").item(index: 0).getTextContent()
    );
  }
}

/**
 * Teljes Dokumentum kiírása
 */
public void printXml() {
  printCukraszdak(doc.getElementsByTagName(tagname: "cukraszda"));
  printCukrasz(doc.getElementsByTagName(tagname: "cukrasz"));
  printSutemenyLista(doc.getElementsByTagName(tagname: "sutemeny_lista"));
  printSutemeny(doc.getElementsByTagName(tagname: "sutemeny"));
  printVasarlo(doc.getElementsByTagName(tagname: "vasarlo"));
  printTranzakcio(doc.getElementsByTagName(tagname: "tranzakcio"));
}
```

```java
/**
 * Tetszőleges gyökérelem kiírása
 * @param nodeList gyökérelem
 */
public void printNodes(NodeList nodeList) {
  switch (nodeList.item(index: 0).getNodeName()) {
    case "cukraszda":
      printCukraszdak(nodeList);
    case "cukrasz":
      printCukrasz(nodeList);
    case "sutemeny_lista":
      printSutemenyLista(nodeList);
    case "sutemeny":
      printSutemeny(nodeList);
    case "vasarlo":
      printVasarlo(nodeList);
    default:
      return;
  }
}
```

DomReadJmdrgg kimenete:

```
Cukrászdák:
id: c1
Cukrászda neve: Daubner Cukrászda
Címe: Budapest Szépvölgyi  50
Értékelése: 4.3
Weboldala: daubnercukraszda.hu

id: c2
Cukrászda neve: Frey Cukrászda
Címe: Pécs Hungária 53/1
Értékelése: 4.7
Weboldala: freycukraszda.hu

id: c3
Cukrászda neve: Sz??ke cukrászda
Címe: Debrecen ?rrétje 14
Értékelése: 4.5
Weboldala: szokecukraszda.hu

Cukrászok:
id: cu1
Cukrász neve: Tompa Tamás
Életkora: 57
Neme: férfi

id: cu2
Cukrász neve: Patkós Petra
Életkora: 21
Neme: n?

id: cu3
Cukrász neve: Nagy Bence
Életkora: 18
Neme: férfi
```



2b) adatmódosítás

DomModifyJmdrgg.java: Két darab privát változóval rendelkezik, egy Dokumentummal és egy kimeneti filenévvel. Az osztály függvényei lehetővé teszik Cukrászok, Cukrászdák, Sütemények, Sütemény listák, vásárlók és tranzakciók hozzáadására, query vagy id alapján elemek törlésére, query alapján elemek módosítására, amelyben megadhatjuk a tag nevet és az új értéket.

```java
package hu.domparse.jmdrgg;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
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 javax.xml.transform.stream.StreamSource;
import org.w3c.dom.DOMException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class DomModifyJmdrgg {

  private Document doc;
  private String outputFilename;

  /**
   * Cukrászda xml fájl módosítására szolgáló objektum
   * @param doc
   * @param outputFilename kimeneti xml fájl neve
   */
  public DomModifyJmdrgg(Document doc, String outputFilename) {
    this.doc = doc;
    this.outputFilename = outputFilename;
  }
```

```java
/**
 * Várárló hozzáadása
 * @param vezeteknev
 * @param keresztnev
 * @param torzsvasarlo
 * @param eletkor
 * @param nem
 */
public void addVasarlo(
  String vezeteknev,
  String keresztnev,
  boolean torzsvasarlo,
  int eletkor,
  String nem
) {
  Element root = doc.getDocumentElement();

  NodeList vasarlok = doc.getElementsByTagName(tagname: "vasarlo");

  // új vásárló elem építés
  Element newVasarlo = doc.createElement(tagName: "vasarlo");
  Element nev = doc.createElement(tagName: "nev");
  Element vezeteknevElem = doc.createElement(tagName: "vezeteknev");
  vezeteknevElem.setTextContent(vezeteknev);
  Element keresztnevElem = doc.createElement(tagName: "keresztnev");
  keresztnevElem.setTextContent(keresztnev);
  nev.appendChild(vezeteknevElem);
  nev.appendChild(keresztnevElem);
  Element torzsvasarloElem = doc.createElement(tagName: "torzsvasarlo");
  torzsvasarloElem.setTextContent(torzsvasarlo ? "igen" : "nem");
  Element eletkorElem = doc.createElement(tagName: "eletkor");
  eletkorElem.setTextContent(Integer.toString(eletkor));
  Element nemElem = doc.createElement(tagName: "nem");
  nemElem.setTextContent(nem);
```

```java
  newVasarlo.appendChild(nev);
  newVasarlo.appendChild(torzsvasarloElem);
  newVasarlo.appendChild(eletkorElem);
  newVasarlo.appendChild(nemElem);

  // id inkrementálása
  Node lastVasarlo = vasarlok.item(vasarlok.getLength() - 1);

  String lastId = lastVasarlo
    .getAttributes()
    .getNamedItem(name: "v_id")
    .getTextContent();
  String newId =
    "v" + (Integer.parseInt(lastId.substring(lastId.length() - 1)) + 1);
  newVasarlo.setAttribute(name: "v_id", newId);

  // az előző vásárló után beillesztés
  root.insertBefore(newVasarlo, lastVasarlo.getNextSibling());

  // írás az outputfileba
  writeXml();
}
```

```java
public void addCukrasz(
  String vezeteknev,
  String keresztnev,
  int eletkor,
  String nem,
  String cu_c
) {
  Element root = doc.getDocumentElement();

  NodeList cukraszok = doc.getElementsByTagName(tagname: "cukrasz");
  // új Cukrász elem építés
  Element newCukrasz = doc.createElement(tagName: "cukrasz");

  Element nev = doc.createElement(tagName: "nev");
  Element vezeteknevElem = doc.createElement(tagName: "vezeteknev");
  vezeteknevElem.setTextContent(vezeteknev);
  Element keresztnevElem = doc.createElement(tagName: "keresztnev");
  keresztnevElem.setTextContent(keresztnev);
  nev.appendChild(vezeteknevElem);
  nev.appendChild(keresztnevElem);
  Element eletkorElem = doc.createElement(tagName: "eletkor");
  eletkorElem.setTextContent(Integer.toString(eletkor));
  Element nemElem = doc.createElement(tagName: "nem");
  nemElem.setTextContent(nem);

  newCukrasz.appendChild(nev);
  newCukrasz.appendChild(eletkorElem);
  newCukrasz.appendChild(nemElem);

  // id inkrementálása
  Node lastCukrasz = cukraszok.item(cukraszok.getLength() - 1);

  String lastId = lastCukrasz
```

```java
String lastId = lastCukrasz
  .getAttributes()
  .getNamedItem(name: "cu_id")
  .getTextContent();
String newId =
  "cu" + (Integer.parseInt(lastId.substring(lastId.length() - 1)) + 1);
newCukrasz.setAttribute(name: "cu_id", newId);

// foreign key
newCukrasz.setAttribute(name: "cu_c", cu_c);

// az előző cukrász után beillesztés
root.insertBefore(newCukrasz, lastCukrasz.getNextSibling());

// output fileba kiírás
writeXml();
```

```java
public void addCukraszda(
  String nev,
  String varos,
  String utca,
  String hazszam,
  String ertekeles,
  String tulajdonos,
  String weboldal
) {
  Element root = doc.getDocumentElement();

  NodeList cukraszdak = doc.getElementsByTagName(tagname: "cukraszda");
  Element newCukraszda = doc.createElement(tagName: "cukraszda");

  Element nevElem = doc.createElement(tagName: "nev");
  nevElem.setTextContent(nev);
  Element hely = doc.createElement(tagName: "hely");
  Element varosElem = doc.createElement(tagName: "varos");
  varosElem.setTextContent(varos);
  Element utcaElem = doc.createElement(tagName: "utca");
  utcaElem.setTextContent(utca);
  Element hazszamElem = doc.createElement(tagName: "hazszam");

  hely.appendChild(varosElem);
  hely.appendChild(utcaElem);
  hely.appendChild(hazszamElem);

  Element ertekelesElem = doc.createElement(tagName: "ertekeles");
  ertekelesElem.setTextContent(ertekeles);
  Element tulajdonosElem = doc.createElement(tagName: "tulajdonos");
  tulajdonosElem.setTextContent(tulajdonos);
  Element weboldalElem = doc.createElement(tagName: "weboldal");
  weboldalElem.setTextContent(weboldal);

  newCukraszda.appendChild(nevElem);
  newCukraszda.appendChild(hely);
  newCukraszda.appendChild(ertekelesElem);
  newCukraszda.appendChild(tulajdonosElem);
  newCukraszda.appendChild(weboldalElem);

  Node lastCukraszda = cukraszdak.item(cukraszdak.getLength() - 1);

  String lastId = lastCukraszda
    .getAttributes()
    .getNamedItem(name: "c_id")
    .getTextContent();
  String newId =
    "c" + (Integer.parseInt(lastId.substring(lastId.length() - 1)) + 1);
  newCukraszda.setAttribute(name: "c_id", newId);

  root.insertBefore(newCukraszda, lastCukraszda.getNextSibling());

  writeXml();
}
```

```
public void addSutemeny(
  String nev,
  int ar,
  int kaloria,
  String[] tulajdonsagok,
  String s_sl
) {
  Element root = doc.getDocumentElement();

  NodeList sutemenyek = doc.getElementsByTagName(tagname: "sutemeny");
  Element newSutemeny = doc.createElement(tagName: "sutemeny");

  Element nevElem = doc.createElement(tagName: "nev");
  nevElem.setTextContent(nev);
  Element arElem = doc.createElement(tagName: "ar");
  arElem.setTextContent(Integer.toString(ar));
  Element kaloriaElem = doc.createElement(tagName: "kaloria");
  kaloriaElem.setTextContent(Integer.toString(kaloria));

  newSutemeny.appendChild(nevElem);
  newSutemeny.appendChild(arElem);
  newSutemeny.appendChild(kaloriaElem);

  for (String tul : tulajdonsagok) {
    Element tulajdonsagElem = doc.createElement(tagName: "tulajdonsag");
    tulajdonsagElem.setTextContent(tul);
    newSutemeny.appendChild(tulajdonsagElem);
  }

  Node lastSutemeny = sutemenyek.item(sutemenyek.getLength() - 1);
```

```
  String lastId = lastSutemeny
    .getAttributes()
    .getNamedItem(name: "s_id")
    .getTextContent();
  String newId =
    "s" + (Integer.parseInt(lastId.substring(lastId.length() - 1)) + 1);
  newSutemeny.setAttribute(name: "s_id", newId);

  //foreign key
  newSutemeny.setAttribute(name: "s_sl", s_sl);

  root.insertBefore(newSutemeny, lastSutemeny.getNextSibling());

  writeXml();
}
```

```java
/**
 * Sütemény lista hozzáadása
 * @param legnepszerubb
 * @param legnepszerutlenebb
 * @param legutobbFrissitve
 * @param sl_c
 */
public void addSutemenyLista(
  String legnepszerubb,
  String legnepszerutlenebb,
  String legutobbFrissitve,
  String sl_c
) {
  Element root = doc.getDocumentElement();

  NodeList sutemenyListak = doc.getElementsByTagName(tagname: "sutemeny_lista");
  Element newSutemenyLista = doc.createElement(tagName: "sutemeny_lista");

  Element legnepszerubbElem = doc.createElement(tagName: "legnepszerubb");
  legnepszerubbElem.setTextContent(legnepszerubb);
  Element legnepszerutlenebbElem = doc.createElement(tagName: "legnepszerutlenebb");
  legnepszerutlenebbElem.setTextContent(legnepszerutlenebb);
  Element legutobbFrissitveElem = doc.createElement(tagName: "legutobb_frissitve");
  legutobbFrissitveElem.setTextContent(legutobbFrissitve);

  newSutemenyLista.appendChild(legnepszerubbElem);
  newSutemenyLista.appendChild(legnepszerutlenebbElem);
  newSutemenyLista.appendChild(legutobbFrissitveElem);

  Node lastSutemenyLista = sutemenyListak.item(
    sutemenyListak.getLength() - 1
  );
```

```java
  String lastId = lastSutemenyLista
    .getAttributes()
    .getNamedItem(name: "sl_id")
    .getTextContent();
  String newId =
    "sl" + (Integer.parseInt(lastId.substring(lastId.length() - 1)) + 1);
  newSutemenyLista.setAttribute(name: "sl_id", newId);

  //foreign key
  newSutemenyLista.setAttribute(name: "sl_c", sl_c);

  root.insertBefore(newSutemenyLista, lastSutemenyLista.getNextSibling());

  writeXml();
}
```

```java
/**
 * NodeLista törlése
 * @param nodeList
 */
public void removeByQuery(NodeList nodeList) {
  if (nodeList != null) {
    for (int i = 0; i < nodeList.getLength(); i++) {
      Node node = nodeList.item(i);
      node.getParentNode().removeChild(node);
    }
  }

  writeXml();
}

/**
 * Elem törlése id alapján
 * @param nodeType
 * @param id
 */
public void removeById(String nodeType, String id) {
  DomQueryJmdrgg dq = new DomQueryJmdrgg(doc);
  Node node = dq.queryById(nodeType, id);

  if (node != null) {
    node.getParentNode().removeChild(node);
  }

  writeXml();
}
```

```java
/**
 * Elemek módosítása tag alapján
 * @param nodeList
 * @param tagName
 * @param newValue
 */
public void modifyNode(NodeList nodeList, String tagName, String newValue) {
  if (nodeList != null) {
    for (int i = 0; i < nodeList.getLength(); i++) {
      Node node = nodeList.item(i);
      Element elem = (Element) node;
      try {
        elem.getElementsByTagName(tagName).item(index: 0).setTextContent(newValue);
      } catch (DOMException e) {
        e.printStackTrace();
      }
    }
  }

  writeXml();
}
```

```java
/**
 * Elemek módosítása tag alapján
 * @param node
 * @param tagName
 * @param newValue
 */
public void modifyNode(Node node, String tagName, String newValue) {
  if (node != null) {
    Element elem = (Element) node;
    try {
      elem.getElementsByTagName(tagName).item(index: 0).setTextContent(newValue);
    } catch (DOMException e) {
      e.printStackTrace();
    }
  }

  writeXml();
}
```

```java
/**
 * Módosítások kiírása az outputfileba
 */
private void writeXml() {
  try (FileOutputStream output = new FileOutputStream(outputFilename)) {
    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer(
      new StreamSource(new File(pathname: "./stylesheet.xslt"))
    );
    DOMSource source = new DOMSource(doc);
    StreamResult result = new StreamResult(output);

    transformer.transform(source, result);
  } catch (TransformerException | IOException e) {
    e.printStackTrace();
  }
}
```

Módosítások és eredményeik:

```java
dm.addVasarlo(vezeteknev: "Pista", keresztnev: "Laci", torzsvasarlo: false, eletkor: 40, nem: "férfi");
dq.queryVasarloByName(vezeteknev: "Pista", keresztnev: "Laci");
```

```
Vásárlók:
id: v4
Név: Pista Laci
Törzsvásárló: nem
Életkor: 40
Nem: férfi
```

```java
dm.removeById(nodeType: "vasarlo", id: "v4");
dq.queryVasarloByName(vezeteknev: "Pista", keresztnev: "Laci");
```

```
Vásárlók:
```

```java
String[] tuls = { "klasszikus", "édes" };
System.out.println(x: "elotte: ");
dr.printNodes(dq.querySutemenyByTulajdonsag(tuls));
dm.removeByQuery(dq.querySutemenyByTulajdonsag(tuls));
System.out.println(x: "utanna: ");
dr.printNodes(dq.querySutemenyByTulajdonsag(tuls));
```

```
elotte:
Sütemények:
id: s1
Név: Almás pite
?r: 740
Kalória: 296
tulajdonságok:
klasszikus
fahéjas
almás
pite

id: s4
Név: Mignon
?r: 720
Kalória: 120
tulajdonságok:
cukormáz
édes
Hagyományos

id: s5
Név: Piramis szelet
```

```
utanna:
Sütemények:
```

```
System.out.println(x: "elotte: ");
dq.queryById(nodeName: "vasarlo", value: "v1");
dm.modifyNode(dq.queryById(nodeName: "vasarlo", value: "v1"), tagName: "keresztnev", newValue: "Pista");
System.out.println(x: "utanna ");
dq.queryById(nodeName: "vasarlo", value: "v1");
```

```
elotte:
Vásárlók:
id: v1
Név: Nagy Róbert
Törzsvásárló: igen
Életkor: 21
Nem: férfi
```

```
utanna
Vásárlók:
id: v1
Név: Nagy Pista
Törzsvásárló: igen
Életkor: 21
Nem: férfi
```

```
Random r = new Random();
dm.modifyNode(
    dq.querySutemenyByKaloria(kaloria: 200),
    tagName: "kaloria",
    Integer.toString((r.nextInt(bound: 100) + 200))
);
```

```
Sütemények:
id: s4
Név: Mignon
?r: 720
Kalória: 120
tulajdonságok:
cukormáz
édes
Hagyományos

id: s5
Név: Piramis szelet
?r: 850
Kalória: 110
tulajdonságok:
krém
Csokoládés
édes
Klasszikus
krémes
piskóta

id: s7
Név: Piskótatekercs
?r: 710
Kalória: 110
tulajdonságok:
baracklekvár
édes
Klasszikus
piskóta
tekercs
```

```
id: s4
Név: Mignon
?r: 720
Kalória: 232
tulajdonságok:
cukormáz
édes
Hagyományos
```

```
id: s7
Név: Piskótatekercs
?r: 710
Kalória: 232
tulajdonságok:
baracklekvár
édes
Klasszikus
piskóta
tekercs
```

```
id: s5
Név: Piramis szelet
?r: 850
Kalória: 232
tulajdonságok:
krém
Csokoládés
édes
Klasszikus
krémes
piskóta
```

2c) adatlekérdezés

DomQueryJmdrgg.java

Lekérdezések név, id vagy tulajdonságok alapján. 2 darab privát változóval rendelkezik, Document és XPath változókkal.

```java
package hu.domparse.jmdrgg;

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

public class DomQueryJmdrgg {

  private Document doc;
  private XPath xPath;

  /**
   * Cukrász xml dokumentum lekérdezések kezeléséhez használt objektum
   * @param doc
   */
  public DomQueryJmdrgg(Document doc) {
    this.doc = doc;
    this.xPath = XPathFactory.newInstance().newXPath();
  }
```

```java
/**
 * Tetszőleges node lekérdezése id alapján
 * @param nodeName
 * @param value
 * @return
 */
public Node queryById(String nodeName, String value) {
  String id = null;
  switch (nodeName) {
    case "cukraszda":
      id = "c_id";
      break;
    case "cukrasz":
      id = "cu_id";
      break;
    case "sutemeny_lista":
      id = "sl_id";
      break;
    case "sutemeny":
      id = "s_id";
      break;
    case "vasarlo":
      id = "v_id";
      break;
  }

  if (id ≠ null) {
    String expression = String.format(
      format: "/NR_beadando/%s[@%s = '%s']",
      nodeName,
      id,
      value
    );
```

```java
    try {
      NodeList nodeList = (NodeList) xPath
        .compile(expression)
        .evaluate(doc, XPathConstants.NODESET);

      DomReadJmdrgg dr = new DomReadJmdrgg(doc);
      dr.printNodes(nodeList);

      return nodeList.item(index: 0);
    } catch (XPathExpressionException e) {
      e.printStackTrace();
    }
  }

  return null;
}
```

```java
/**
 * Vásárlók keresése név alapján
 * @param vezeteknev
 * @param keresztnev
 * @return
 */
public NodeList queryVasarloByName(String vezeteknev, String keresztnev) {
  String expression = String.format(
    format: "/NR_beadando/vasarlo[nev/vezeteknev='%s' and nev/keresztnev='%s']",
    vezeteknev,
    keresztnev
  );
  try {
    NodeList nodeList = (NodeList) xPath
      .compile(expression)
      .evaluate(doc, XPathConstants.NODESET);

    DomReadJmdrgg dr = new DomReadJmdrgg(doc);
    dr.printVasarlo(nodeList);

    return nodeList;
  } catch (XPathExpressionException e) {
    e.printStackTrace();
  }

  return null;
}
```

```java
/**
 * Cukrászok lekérdezése név alapján
 * @param vezeteknev
 * @param keresztnev
 * @return
 */
public NodeList queryCukraszByName(String vezeteknev, String keresztnev) {
  String expression = String.format(
    format: "/NR_beadando/cukrasz[nev/vezeteknev='%s' and nev/keresztnev='%s']",
    vezeteknev,
    keresztnev
  );
  try {
    NodeList nodeList = (NodeList) xPath
      .compile(expression)
      .evaluate(doc, XPathConstants.NODESET);

    // DomReadJmdrgg.printCukrasz(nodeList);

    return nodeList;
  } catch (XPathExpressionException e) {
    e.printStackTrace();
  }

  return null;
}
```

```java
/**
 * Cukrászda lekérdezése név alapján
 * @param name
 * @return
 */
public NodeList queryCukraszdaByName(String name) {
  String expression = String.format(format: "/NR_beadando/cukraszda[nev='%s']", name);

  try {
    NodeList nodeList = (NodeList) xPath
      .compile(expression)
      .evaluate(doc, XPathConstants.NODESET);

    DomReadJmdrgg dr = new DomReadJmdrgg(doc);
    dr.printCukraszdak(nodeList);

    return nodeList;
  } catch (XPathExpressionException e) {
    e.printStackTrace();
  }

  return null;
}
```

```java
public NodeList querySutemenyByTulajdonsag(String[] tulajdonsagok) {
  StringBuilder sb = new StringBuilder();
  if (tulajdonsagok.length == 1) {
    sb.append(str: "tulajdonsag='");
    sb.append(tulajdonsagok[0] + "'");
  } else {
    for (int i = 0; i < tulajdonsagok.length - 1; i++) {
      sb.append(str: "tulajdonsag='");
      sb.append(tulajdonsagok[i]);
      sb.append(str: "' or ");
    }
    sb.append(str: "tulajdonsag='");
    sb.append(tulajdonsagok[tulajdonsagok.length - 1] + "'");
  }

  String expression = String.format(
    format: "/NR_beadando/sutemeny[%s]",
    sb.toString()
  );
  try {
    NodeList nodeList = (NodeList) xPath
      .compile(expression)
      .evaluate(doc, XPathConstants.NODESET);

    DomReadJmdrgg dr = new DomReadJmdrgg(doc);
    dr.printSutemeny(nodeList);

    return nodeList;
  } catch (XPathExpressionException e) {
    e.printStackTrace();
  }
```

```java
/**
 * Sütemények lekérdezése kalória alapján
 * @param kaloria
 * @return
 */
public NodeList querySutemenyByKaloria(int kaloria) {
  String expression = String.format(
    format: "/NR_beadando/sutemeny[kaloria < '%s']",
    Integer.toString(kaloria)
  );
  try {
    NodeList nodeList = (NodeList) xPath
      .compile(expression)
      .evaluate(doc, XPathConstants.NODESET);

    DomReadJmdrgg dr = new DomReadJmdrgg(doc);
    dr.printSutemeny(nodeList);

    return nodeList;
  } catch (XPathExpressionException e) {
    e.printStackTrace();
  }

  return null;
}
```

Lekérdezések és eredményeik:

```
dq.queryById(nodeName: "vasarlo", value: "v1");
```

```
Vásárlók:
id: v1
Név: Nagy Róbert
Törzsvásárló: igen
Életkor: 21
Nem: férfi
```

```
dq.queryCukraszByName(vezeteknev: "Tompa", keresztnev: "Tamás");
```

```
Cukrászok:
id: cu1
Cukrász neve: Tompa Tamás
Életkora: 57
Neme: férfi
```

```
dq.querySutemenyByKaloria(kaloria: 400);
```

```
Sütemények:
id: s1
Név: Almás pite
?r: 740
Kalória: 296
tulajdonságok:
klasszikus
fahéjas
almás
pite

id: s2
Név: Mandarinos túrótorta szelet
?r: 990
Kalória: 360
tulajdonságok:
csokoládés piskóta
gyümölcsös
klasszikus tortaszelet
tejszínes
torta szelet
túrós

id: s3
Név: Cukormentes málnatorta szelet
```

```
String[] tuls = { "klasszikus", "édes" };
dq.querySutemenyByTulajdonsag(tuls);
```

```
Sütemények:
id: s1
Név: Almás pite
?r: 740
Kalória: 296
tulajdonságok:
klasszikus
fahéjas
almás
pite

id: s4
Név: Mignon
?r: 720
Kalória: 232
tulajdonságok:
cukormáz
édes
Hagyományos

id: s5
Név: Piramis szelet
?r: 850
Kalória: 232
tulajdonságok:
krém
Csokoládés
```

```
dq.queryCukraszdaByName(name: "asd");
```

```
Cukrászdák:
```