

# JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Banki tranzakciós adatbázis

Készítette:	<b>Répási Gábor</b>
Neptunkód:	<b>D51MXC</b>
Dátum:	<b>2023.12.04</b>

# Bevezetés

A feladatom egy banki adatbázis menedzselése. Mint minden banknak vannak ügyfelei. Az ügyfelek általában többen vannak, és a számlákból is több van. Ami a hétköznapi életben is előszokott fordulni, hogy egy természetes személy egyszer fordul elő egy banknál, de ennek a személynek lehet több számlája is, ez valósítja meg az **1:N** kapcsolatot.

Szintén egy ügyfélnek kell lennie egy hivatalos elérhetőségnek, amelyen mindig mindenkor utolérhető, ez is megvalósítja az **1:N** kapcsolatot. Opcionális, lehet nulla, egy vagy több is.

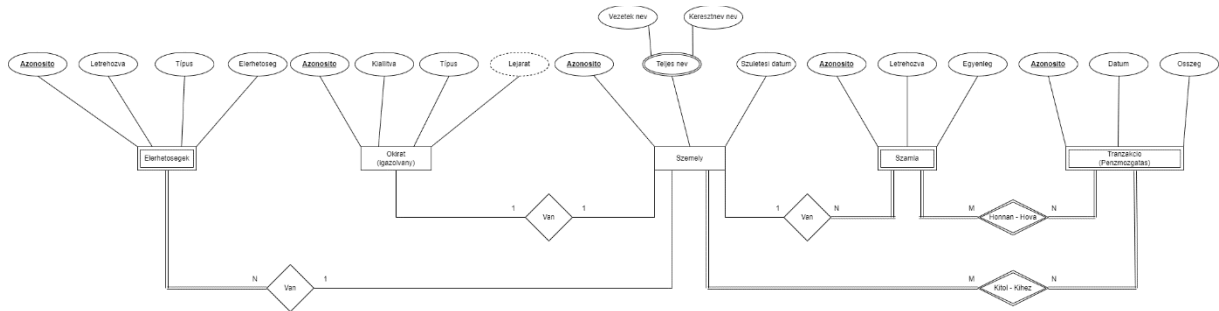
Egy személynek csak egyféle okirata lehet, például nem lehet több személyi igazolványa, csak egy, amely csak egy személyhez tartozhat, ez lesz az **1:1** kapcsolat. A lejárat mély opcionális, ugyanis nem minden okiratnak van lejárat ideje, ez jelöli a rajz is.

Nem utolsó sorban pedig maga a tranzakciót megvalósító, azt dokumentáló, illetve naplózó bejegyzések, ahol több személyhez is kapcsolódhat, attól függően, hogy kitől és kihez kerül a pénz, valamint melyik számláról melyikre, amely így több-több **N:M** kapcsolat.

A pénzmozgások nyomon követhetőek, ezáltal kimutatásokat is lehet belőle készíteni, amelyet szemléltetni is fogok a második feladat lekérdezési és szűrési feltételével.

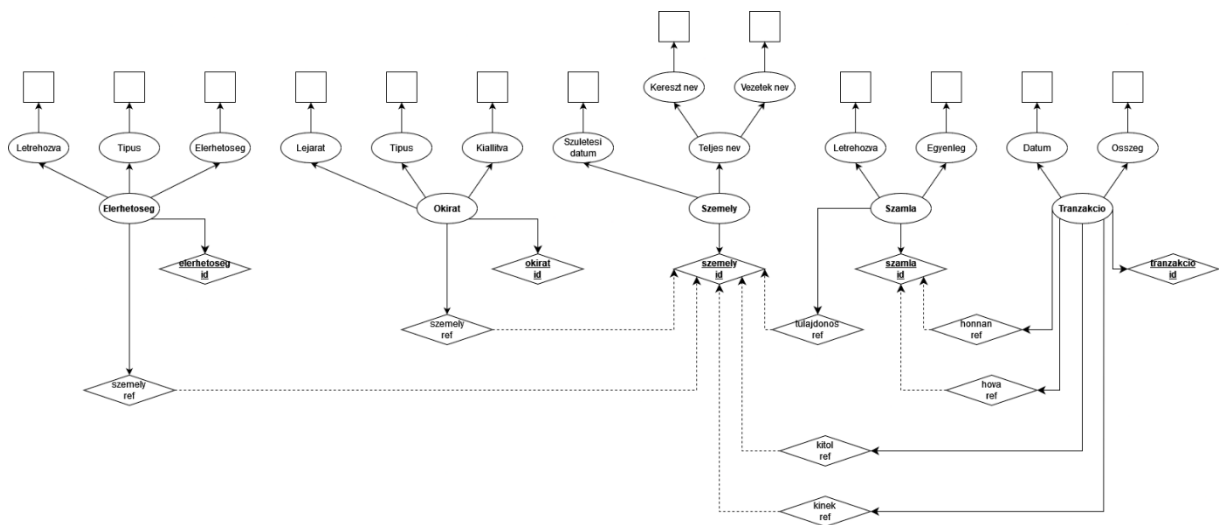
## 1. feladat

A Draw.IO ingyenes rajzolóprogram segítségével elkészítettem az ER modell-t:



(Kép és szerkeszthető terv-sablon mellékelve.)

Az ER modellből megrajoltam az XDM modell-t:



(Kép és szerkeszthető terv-sablon mellékelve.)

## 2. feladat

Microsoft Visual Studio Code programot használok a JAVA állományok indirekt fordítására és futtatására.

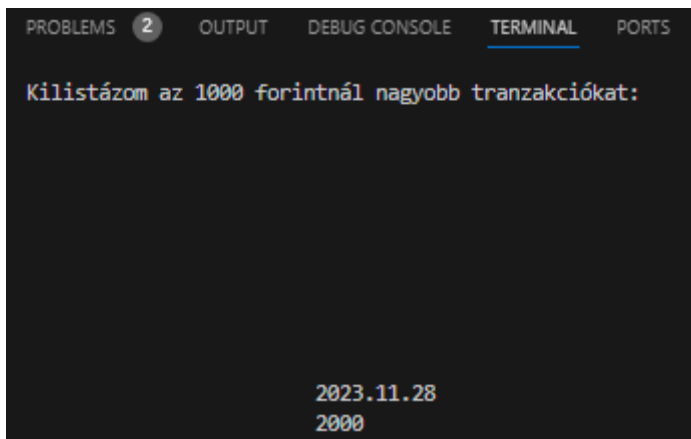
DOM Query (lekérdezés előkészítés, a fájl megnyitása):

```
public static void main(String[] args) {  
    try {  
        // Beolvasom az adatbázisfájlt a projekt "resource" mappájából  
        URL url = DomQueryD51mxc.class.getClassLoader().getResource("XMLD51MXC.xml");
```

DOM Read (beolvasás):

```
System.out.println(x:"Kilistázom az 1000 forintnál nagyobb tranzakciókat:");  
DomReadD51mxc.printnode(document, expression: "/database/tranzakciok/tranzakcio[osszeg>1000]");
```

Eredmény:



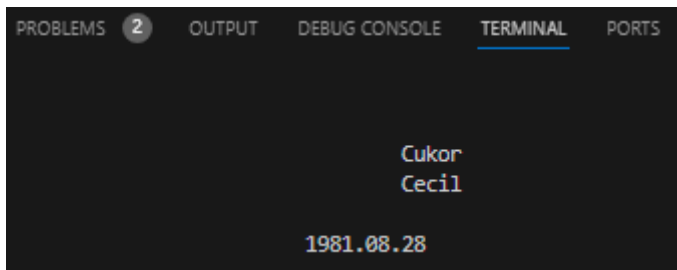
The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab selected. The output of the first query is displayed as follows:

```
Kilistázom az 1000 forintnál nagyobb tranzakciókat:  
  
2023.11.28  
2000
```

DOM Read (beolvasás):

```
System.out.println(x:"Kiválasztom az utolsó személyt (függvényel):");  
DomReadD51mxc.printnode(document, expression: "/database/szemelyek/szemely[last()]");
```

Eredmény:



The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab selected. The output of the second query is displayed as follows:

```
Cukor  
Cecil  
  
1981.08.28
```

A fájl megnyitása sikeresen megtörtént, mind az öt lekérdezésem az XPATH szabványos kifejezéssel valós és legitim eredményekkel szolgált. A szűrési feltételnek megfelelően.

DOM Modify (módosítás):

Betöltöm a lekérdezésnél is használt fő forrásfájlt:

```
public static void main(String[] args) {  
    try {  
        // Beolvasom az adatbázisfájlt a projekt "resource" mappájából  
        URL url = DomQueryD51mxc.class.getClassLoader().getResource("XMLD51MXC.xml");
```

Végrehajtok egy törlő és egy hozzáadó műveletet:

```
// Törölök egy elemet  
removeElement(document, expression: "/database/szemelyek/szemely[id=1]", elementName: "szuletesi_datum");  
  
// Hozzáadok egy elemet  
Node nodeAdd = document.createElement(tagName: "szuletesi_datum");  
nodeAdd.appendChild(document.createTextNode(data: "1999.09.19"));  
addElement(document, expression: "/database/szemelyek/szemely[id=1]", nodeAdd);
```

Végrehajtok egy felülíró műveletet:

```
// Létrehozom a szülő elemet  
Node nev = document.createElement(tagName: "teljes_nev");  
  
Node adat1 = document.createElement(tagName: "vezetek_nev");  
adat1.appendChild(document.createTextNode(data: "Átok"));  
  
Node adat2 = document.createElement(tagName: "kereszt_nev");  
adat2.appendChild(document.createTextNode(data: "Áron"));  
  
// Összefűzöm a gyermek elemeket  
nev.appendChild(adat1);  
nev.appendChild(adat2);  
  
// Felülírom a meglévőt  
replaceElement(document, expression: "/database/szemelyek/szemely[id=1]", nev);
```

Végrehajtom a mentést és kiíratás fa struktúrába:

```
// Kiíratom egy új fájlba  
DomWriteD51mxc.write(document);  
  
// Kiíratom a képernyőre  
DomWriteD51mxc.printNode(new DOMSource(document), depth: 0);
```

Eredmény:

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
  
Elemek törölve!  
Elemek hozzáadva!  
Elemek törölve!  
Elemek hozzáadva!  
Az XML fájl sikeresen kiíródott!  
XML fa struktúra:  
<database xmlns="d51mxc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="d51mxc XMLSchemaD51MXC.xsd">
```

A megvalósító függvényeim az „addElement”, „removeElement”, és a „replaceElement”. Mindkettő tud sima gyöker elemet, egy adattal, illetve szülő-gyerek többadatos Node-ot is fogadni, illetve már meglévőket felülírni.

Az új adatbázisom egy külön fájlba kerül: „XMLD51MXC\_uj.xml”.