JEGYZŐKÖNYV

Webes adatkezelő környezetek

Féléves feladat

Egyetem nyilvántartó

Készítette: **Mihály Gergő**

Neptunkód: **OY6ICJ**

Dátum: **2025. november**

Miskolc, 2025

Tartalomjegyzék

A feladat leírása

A feladat két részből áll.

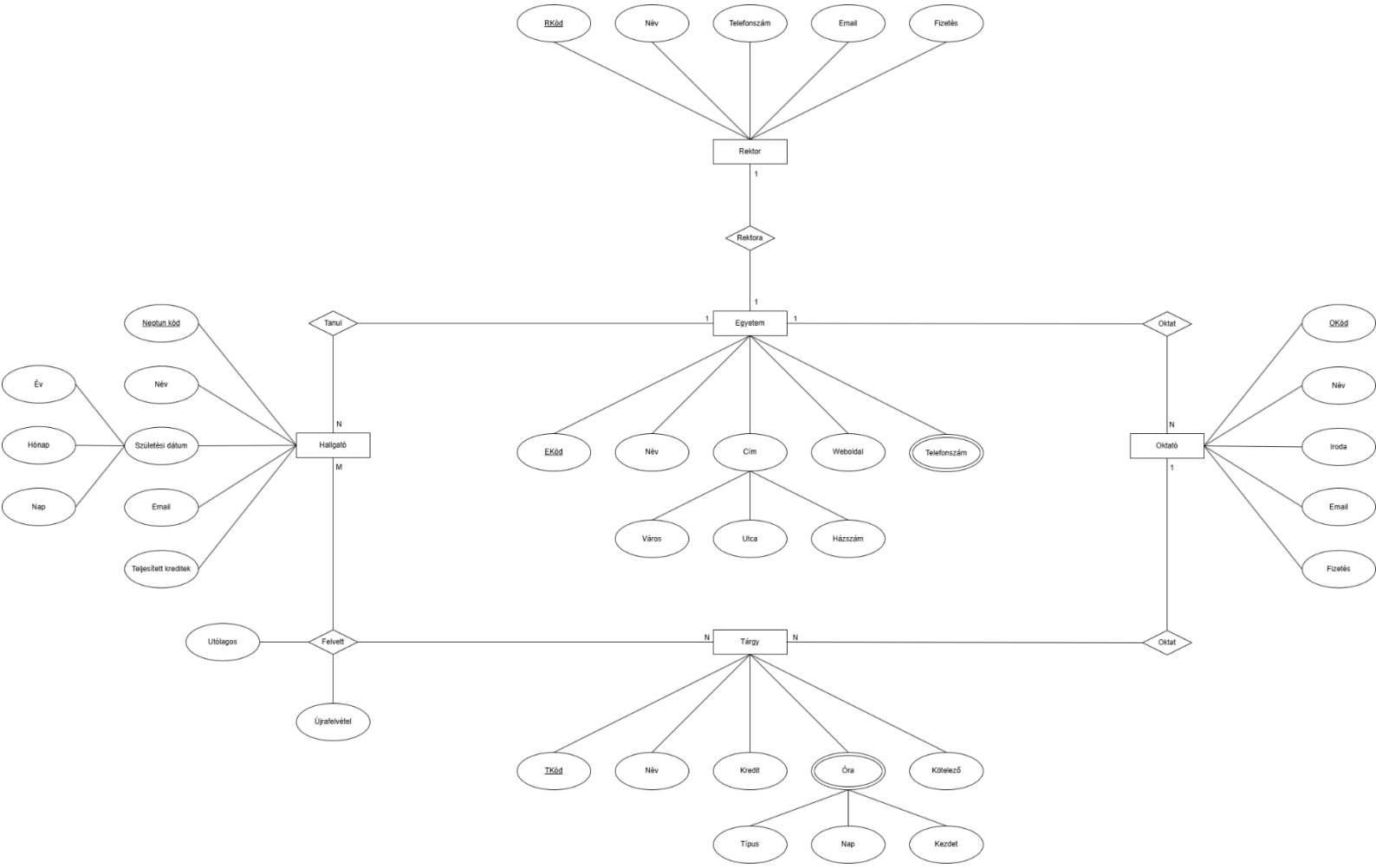
1. feladat: Egy XML adatkezelő készítése, amelynek lépései:
    - egy egyed-kapcsolat modell (ER-modell) elkészítése
    - konvertálás egy XML Document Model-re (XDM)
    - a tényleges XML elkészítése, feltöltése adatokkal
    - XMLSchema készítése az XML dokumentumra, a validáltatás elvégzése az XML dokumentumon

2. feladat: Egy Document Object Model (DOM) program készítése az XML dokumentum feldolgozására Java nyelven. A program részei:
    - egy adatolvasó elem
    - egy adatokat lekérdező elem
    - egy adatokat módosító elem

1. Feladat

1.1 Az ER-modell

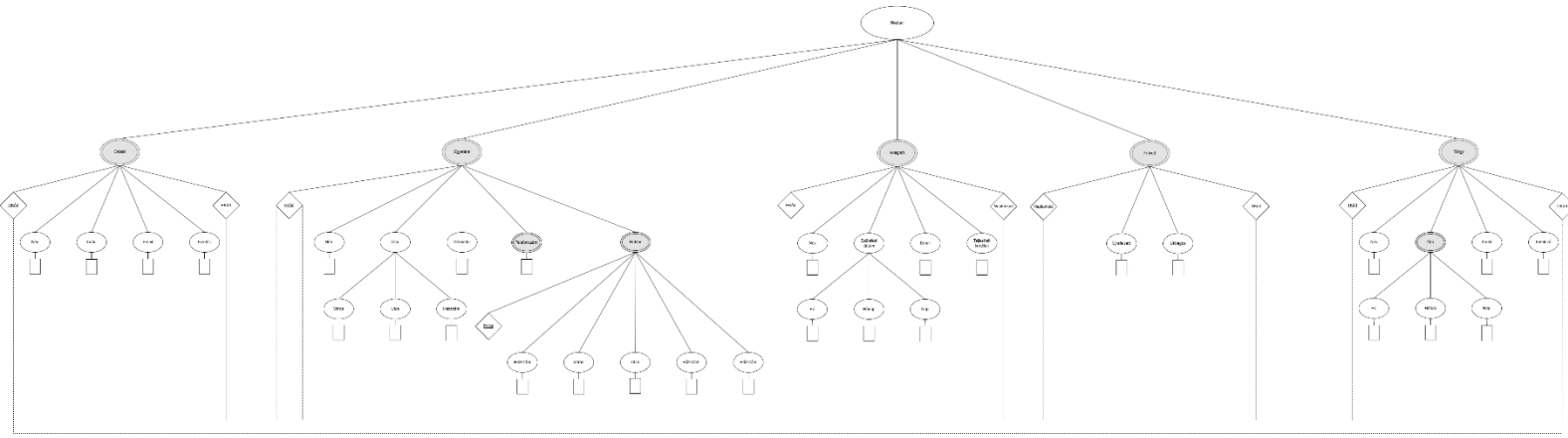
Az ER-modell a Neptun egyetemkezelő rendszert modellezi. Öt darab egyedből áll: Egyetem, Rektor, Hallgató, Oktató, Tárgy. Minden egyednek van egyedi azonosítója, és még négy darab tulajdonsága. A tulajdonságok között vannak összetettek (pl.: a Hallgató születési dátuma), többértékűek (pl.: az Egyetem telefonszáma), és egy olyan összetett tulajdonság, ami egyben többértékű is (a Tárgy órája). Az egyedek között van 1:1 (Egyetem - Rektor), van 1:N (pl.: Oktató - Tárgy) és van N:M (Hallgató - Tárgy) kapcsolat. Az N:M kapcsolatnak a tulajdonságai a felvétel típusát jellemzik.



(1.ábra: Az ER-modell)

1.2 Az XDM

Az XDM-nél megjelenik a Felvett kapcsolótábla, mint elem, a két kulccsal és a tulajdonságaival. Emellett a Rektor elem az 1:1 kapcsolat miatt az Egyetem alá rendelődik (helyes lenne fordítva is, tehát hogy a Rektor elem alá rendelődik az Egyetem). Az idegen kulcs, elsődleges kulcs kapcsolatokat a szaggatott vonal jelöli.



(2. ábra: Az XDM)

1.3 A tényleges XML dokumentum

Az XML dokumentum szerkezete az XDM-ből triviálisan leképezhető. Ezután a dokumentum fel lett töltve adatokkal. A többértékű összetett tulajdonság (a Tárgy órája) a következő módon jelenik meg:

```
<ora>
   <tipus>Előadás</tipus>
   <nap>Csütörtök</nap>
   <kezdet>10</kezdet>
</ora>
<ora>
   <tipus>Gyakorlat</tipus>
   <nap>Kedd</nap>
   <kezdet>8</kezdet>
</ora>
```

A Felvett kapcsolótábla egyik eleme pedig:

```
<felvett neptun_kod="U6SP8X" tkod="01">
    <utolagos>Nem</utolagos>
    <ujrafelvetel>Nem</ujrafelvetel>
</felvett>
```

1.4 Az XMLSchema és a validálás

Az XMLSchema is egyszerűen képezhető az XDM-ből. Először a fő elemek saját típusai lettek definiálva, majd a gyerek elemeik saját típusai. Ahol szükséges, ott a szöveges adatok is ellenőrizve vannak, pl.: weboldalnál a saját típus regex kifejezéssel ellenőrzi, hogy helyes-e a weboldal formátuma:

```
<xs:simpleType name="weboldalTipus">
    <xs:restriction base="xs:string">
        <xs:pattern value="https:\/\/[a-zA-Z0-9\-.]*.hu\/"></xs:pattern>
    </xs:restriction>
</xs:simpleType>
```

Egy másik gyakori ellenőrzés, hogy a szöveges adat csak bizonyos értékeket vehet fel, pl.: a nap elem csak a hét napjait veheti fel értékül:

```
<xs:simpleType name="napTipus">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Hétfő"/>
        <xs:enumeration value="Kedd"/>
        <xs:enumeration value="Szerda"/>
        <xs:enumeration value="Csütörtök"/>
        <xs:enumeration value="Péntek"/>
    </xs:restriction>
</xs:simpleType>
```

Ezt követi az XML dokumentum szerkezetének leírása, felhasználva a definiált típusokat, majd a kulcs definíciók, ahol minden elsődleges kulcs és idegen kulcs megtalálható.

A Felvett kapcsolótábla kulcs definíciói:

```
<xs:keyref name="felvettHallgatoFK" refer="hallgatoPK">
   <xs:selector xpath="felvett"/>
   <xs:field xpath="@neptun_kod"/>
</xs:keyref>

<xs:keyref name="felvettTargyFK" refer="targyPK">
   <xs:selector xpath="felvett"/>
   <xs:field xpath="@tkod"/>
</xs:keyref>
```

A hallgatoPK és targyPK előtte definiált elsődleges kulcsok, a Hallgatóra és a Tárgyra.

2. Feladat

A Java program a Maven projektmenedzsert használja, egy darab függősége van, a Lombok, ami automatikusan ír sablon kódot (pl.: getter, setter, toString). A programban egy kezdetleges API van megvalósítva az XML dokumentum közvetlen Java osztályba olvasására, onnan közvetlen írásra. Bár egy DOM program, megpróbálja ezt absztraktálni. A különböző műveletek között konzolos menürendszerben lehet navigálni. A programban minden függvény Javadoc dokumentációval van ellátva. Minden kiírási művelet, mindig, vagy opcionálisan a konzolon kívül fájlba is kiírja az eredményét. Ezekhez az írásokhoz elérhető két kiíró függvény, blokk, ill. lekérdezés kiírására. A program kimeneti és bemeneti fájljai a 2. feladat\OY6ICJDomParse\res mappában érhetőek el.

2.1 Adatolvasás

Az olvasáshoz szükséges egy olyan Java osztályszerkezet, ami megfelel az XML dokumentum szerkezetének. Minden ilyen osztálynak implementálnia kell a *Parsable* interfészt, aminek két függvénye van:

```
/**
 * Feldolgozza az implementáló osztálynak megfelelő XML <code>Element</code>-et.
 * @param parentElement az implementáló osztálynak megfelelő XML elem szülő eleme
 * @throws XMLParseException az XML olyan címkét tartalmaz, amit az implementáló osztály nem tudott feldolgozni
 */
public void parse(Element parentElement) throws XMLParseException;
/**
 * Kiír a konzolra, opcionálisan fájlba.
 * @param writer opcionális <code>PrintWriter</code>, ha nincs a konzolra, ha van akkor fájlba is ír
 * @param indent a kiírás tartalmazzon-e behúzást (egymásba ágyazott <code>Parsable</code> osztályok
 * kiírásakor az alosztályok mindenképp tartalmaznak behúzást, a hierarchia szemléltetése miatt)
 */
public void print(Optional<PrintWriter> writer, boolean indent);
```

Ezután elvégezhető a beolvasás, ami a *parse* metóduson keresztül érhető el. Mivel a beolvasás (és a kiírás is) osztály szinten történik, csak a szülő elem szükséges, ezért bármilyen struktúrájú és bonyolultságú XML dokumentum feldolgozható. A tényleges *parse* implementáció DOM-al működik. A beolvasás után opcionálisan kiírható a teljes tartalom fájlba (out.txt).

2.2 Adat lekérdezés

Mivel a beolvasás Java osztályt eredményez, ezért a lekérdezések is ezen az osztályon dolgoznak, lista műveletekkel, szűrésekkel, segéd lekérdezésekkel. Az osztályszerkezet gyökerében (*Neptun* osztály) megtalálható néhány segéd lekérdezés. 4 darab elérhető lekérdezés lett megvalósítva:

- Egy egyetem minden hallgatója
- Egy egyetem minden tárgya és a kredit értékük
- Egy egyetem minden dolgozója és a beosztásuk, fizetésre szűrve
- Egy hallgató tárgyai és hogy kötelezőek-e, újrafelvettre szűrve

A legfrissebb lekérdezés eredménye a konzolon és a queryResults.txt fájlban látszódik.

2.3 Adat módosítás

A módosítás is a Java osztályon keresztül történik. Setterekkel kell beállítani a kívánt értékeket, majd lehetőség van kiírni konzolra, vagy fájlba a módosított XML dokumentumot. A fájlba íráshoz szükséges implementálnia az osztálynak *Writable* interfészt:

```
public interface Writable {

    /**
     * Az implementáló osztálynak megfelelő XML elemet hozza létre.
     * @param doc az írandó dokumentum
     * @param parentElement az implementáló osztálynak megfelelő XML elem szülője,
     * ehhez helyben hozzá is kell fűzni a létrehozott elemet
     */
    public void writeElement(Document doc, Element parentElement);
}
```

És a gyökér osztálynak implementálnia kell a *RootWritable* interfészt:

```
public interface RootWritable extends Writable{

    /**
     * A <code>Document</code>-et és a gyökérelemet hozza létre, majd hívja a tényleges fájl kiírást.
     * @param name az XML fájl neve
     * @throws ParserConfigurationException
     * @throws TransformerException
     */
    public void write(String name) throws ParserConfigurationException, TransformerException;
}
```

Ha egymásba ágyazott írandó osztályok vannak, a gyökér elem felel azért, hogy elkezdődjön az írási lánc.

4 darab módosítás lett megvalósítva:

- Egy oktató email címének megváltoztatása
- Új telefonszám hozzáadása egy egyetemhez
- Egy egyetem minden dolgozójának fizetésének százalékos megemelése
- Egy tárgy elvégzése (minden hallgatójának a teljesített kreditjéhez hozzáadódik a tárgy kredit értéke)

A legutóbbi módosítás a konzolon látszik, emelett minden módosítás külön ír XML fájlba (modify1-4.xml)