Jegyzőkönyv

Adatkezelés XML környezetben

Féléves feladat

Készítette: **Bolyki Balázs**
Neptunkód: **N5IF3V**

A feladat leírása:

A feladat témája bolygók, illetve égitestek adatainak összefoglalása. Esetünkben, amit legfőképp lényegesnek tartottam, azok a **bolygók**. A bolygókhoz közvetlenül tárolandó információ a bolygó azonosítója, neve, sugara, tömege és típusa (ahol a típus lehet szilárd, gáz és törpe).

A bolygók külső adatai először is a bolygó **anyaga**. Az anyaggal kapcsolatban olyan információkkal szeretnénk rendelkezni, mint az anyag neve, azonosítója, molekuláris súlya és a használhatóságai. A használhatóság több értékű tulajdonság. Egy anyagot lehet például életfeltételként megjelölni használhatóság alapján, de például a víz üzemanyag is, továbbá a szilícium elektronikában használatos. Egy bolygónak lehet több anyaga, és egy anyag több bolygóhoz is tartozhat.

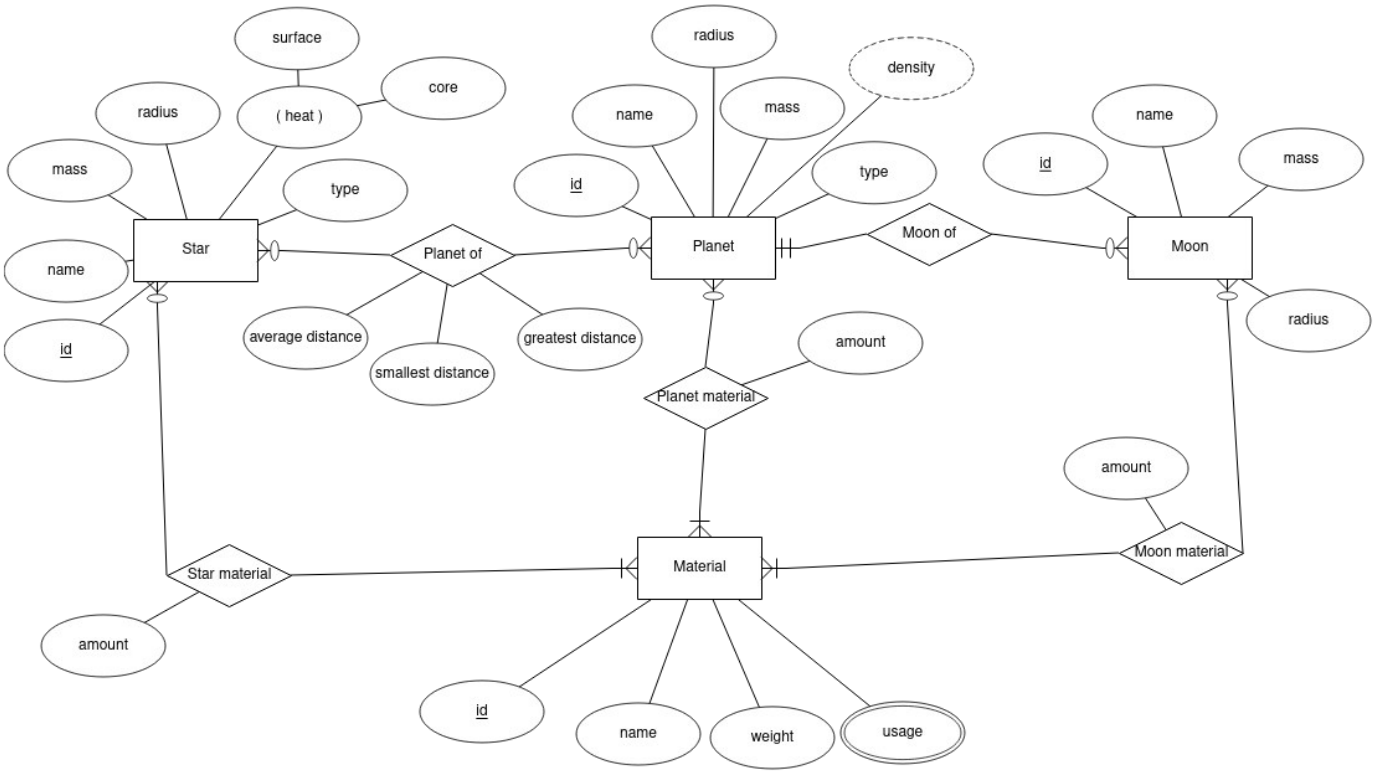
A következő lényeges külső adat a **csillag**, ami körül a bolygó kering. A csillaghoz hasonló módon tároljuk az azonosítóját, a nevét, az anyagát (ami külső kapcsolat a bolygó anyagához hasonlóan), továbbá tömegét, sugarát, típusát (típus például fehér törpe, barna törpe, vörös óriás, stb.), végül pedig összetett tulajdonságként a hőmérsékletét. Mivel egy csillag hőmérséklete drasztikusan különbözik a magjában és a felszínén, ezért ezeket külön tároljuk. Egy bolygó tartozhat több csillaghoz, és egy csillaghoz tartozhat több bolygó is; szemléltetendő ennek a lehetőségét megemlíthetjük, hogy a Földhöz legközelebb eső másik naprendszer 2 egymáshoz közel keringő, és 1 harmadik távolabbi napból áll, ez utóbbi harmadik pedig rendelkezik egy bolygóval.

Az utolsó lényeges külső adat a **hold**, amely hasonló adatokkal rendelkezik, mint a nap és a bolygó:

azonosító, név, sugár, tömeg, tovább az anyag. Az anyagot továbbra is több-több kapcsolatként kezeljük. Egy hold tárgyalásunkban csak egy bolygóhoz tartozhat. Azt az esetet, hogy talán több bolygó van és több hold, kizárjuk. Erre indok lehet az egyszerűbb tárgyalás, és az, hogy például a Plútó törpebolygó holdja, a Kharón is pusztán feleakkora, mint a Plútó; keringés szempontjából könnyen lehetne őket egyenrangúnak tekinteni, az egyik mégis bolygónak van kinevezve. Tehát az adatmodellünkben esetleges több bolygós rendszer esetén kinevezhetünk egy főbolygót, a többit pedig holdként kezelhetjük.

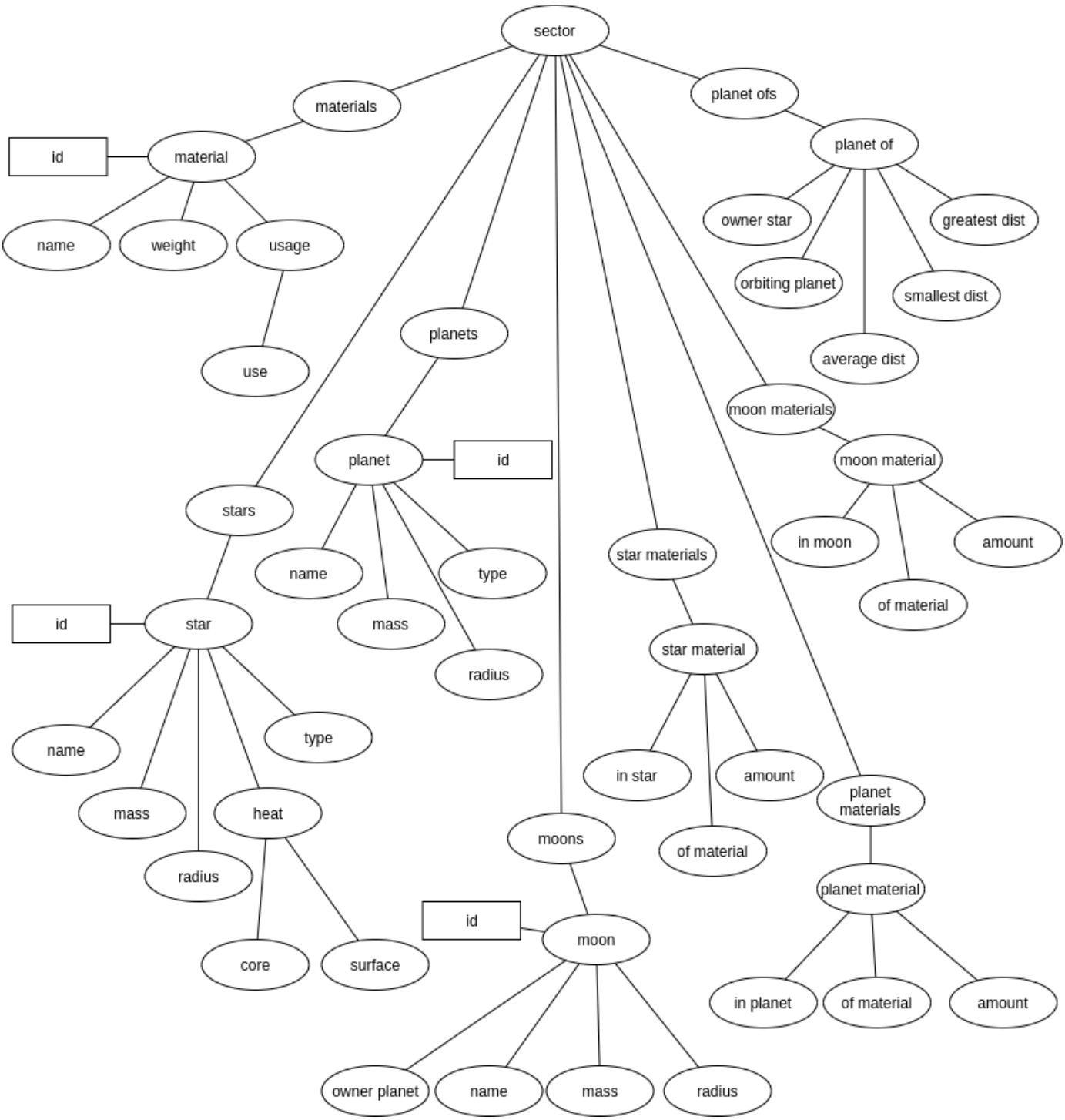
1. feladat

1a) Az adatbázis ER modell



1b) Az adatbázis konvertálása XDM modellre

A konvertálás XDM modellre egyszerűen adódik. A könnyebb kezelhetőség érdekében minden egyedhez és több-több kapcsolathoz felvesszük annak a többet számú alakját is (ez jól jöhet, ha beillesztéseket is akarnánk végezni). A több-több kapcsolatokat külön XDM összetett elemként kezeljük. A ‘usage’ többértékű tulajdonságot szintén külön XDM összetett elemnek vesszük (ez egy sequence-t fog tartalmazni). A készült ábra alább látható.



1c) Az XDM modell alapján XML dokumentum készítése

```
<?xml version="1.0" encoding="UTF-8"?>

<sector xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="XMLSchemaN5IF3V.xsd">

      <materials>
            <material id="H">
                  <name>hydrogen</name>
                  <weight>1</weight>
                  <!--in gramms-s-->
                  <usage>
```

```
                  <use>fuel</use>
            </usage>
      </material>
      <material id="O">
            <name>oxygen</name>
            <weight>16</weight>
            <usage>
                  <use>life condition</use>
            </usage>
      </material>
      <material id="H2O">
            <name>water</name>
            <weight>18</weight>
            <usage>
                  <use>life condition</use>
                  <use>fuel</use>
                  <use>machinery part</use>
            </usage>
      </material>
      <material id="Si">
            <name>silicon</name>
            <weight>28</weight>
            <usage>
                  <use>electronics</use>
            </usage>
      </material>
</materials>

<stars>
      <star id="0E4J4">
            <name>Sun</name>
            <mass>1.989e30</mass>
            <!--in kg-s-->
            <radius>696342</radius>
            <!--in km-s-->
            <heat>
                  <surface>3800</surface>
                  <core>15700000</core>
            </heat>
            <type>yellow dwarf</type>
      </star>
</stars>

<planets>
      <planet id="1">
            <name>Earth</name>
            <radius>6371</radius>
            <mass>5.97237e24</mass>
            <!--in kg-s-->
            <type>solid</type>
      </planet>
</planets>

<moons>
      <moon id="1">
            <owner_planet>1</owner_planet>
            <name>Luna</name>
            <mass>7.342e22</mass>
            <!--in kg-s-->
            <radius>1737</radius>
            <!--in km-s-->
      </moon>
```

```
	</moons>

	<star_materials>
		<star_material>
			<in_star>0E4J4</in_star>
			<of_material>H2O</of_material>
			<amount>1.123e10</amount>
			<!--in kg-s-->
		</star_material>
	</star_materials>

	<planet_materials>
		<planet_material>
			<in_planet>1</in_planet>
			<of_material>H2O</of_material>
			<amount>1.194474e21</amount>
			<!--in kg-s-->
		</planet_material>
	</planet_materials>

	<moon_materials>
		<moon_material>
			<in_moon>1</in_moon>
			<of_material>Si</of_material>
			<amount>1.4684e22</amount>
			<!--in kg-s-->
		</moon_material>
		<moon_material>
			<in_moon>1</in_moon>
			<of_material>O</of_material>
			<amount>3.15706e22</amount>
		</moon_material>
	</moon_materials>

	<planet_ofs>
		<planet_of>
			<owner_star>0E4J4</owner_star>
			<orbiting_planet>1</orbiting_planet>
			<average_dist>1.49e8</average_dist>
			<!--in km-s-->
			<smallest_dist>1.46e8</smallest_dist>
			<!--in km-s-->
			<greatest_dist>1.52e8</greatest_dist>
			<!--in km-s-->
		</planet_of>
	</planet_ofs>
</sector>
```

1d) Az XML dokumentum alapján XMLSchema készítése

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="sector">
	<xs:complexType>
		<xs:sequence>
			<xs:element name="materials" type="materials_type"/>
			<xs:element name="stars" type="stars_type"/>
			<xs:element name="planets" type="planets_type"/>
			<xs:element name="moons" type="moons_type"/>
			<xs:element name="star_materials" type="star_materials_type"/>
			<xs:element name="planet_materials"
type="planet_materials_type"/>
```

```
                  <xs:element name="moon_materials" type="moon_materials_type"/>
                  <xs:element name="planet_ofs" type="planet_ofs_type"/>
            </xs:sequence>
      </xs:complexType>
      <xs:key name="K_material">
            <xs:selector xpath="material"></xs:selector>
            <xs:field xpath="@id"></xs:field>
      </xs:key>
      <xs:key name="K_star">
            <xs:selector xpath="star"></xs:selector>
            <xs:field xpath="@id"></xs:field>
      </xs:key>
      <xs:key name="K_planet">
            <xs:selector xpath="planet"></xs:selector>
            <xs:field xpath="@id"></xs:field>
      </xs:key>
      <xs:key name="K_moon">
            <xs:selector xpath="moon"></xs:selector>
            <xs:field xpath="@id"></xs:field>
      </xs:key>
      <xs:keyref name="FK_star_material_star" refer="K_star">
            <xs:selector xpath="star_material"></xs:selector>
            <xs:field xpath="in_star"></xs:field>
      </xs:keyref>
      <xs:keyref name="FK_star_material_material" refer="K_material">
            <xs:selector xpath="star_material"></xs:selector>
            <xs:field xpath="of_material"></xs:field>
      </xs:keyref>
      <xs:keyref name="FK_planet_material_planet" refer="K_planet">
            <xs:selector xpath="planet_material"></xs:selector>
            <xs:field xpath="in_planet"></xs:field>
      </xs:keyref>
      <xs:keyref name="FK_planet_material_material" refer="K_material">
            <xs:selector xpath="planet_material"></xs:selector>
            <xs:field xpath="of_material"></xs:field>
      </xs:keyref>
      <xs:keyref name="FK_moon_material_moon" refer="K_moon">
            <xs:selector xpath="moon_material"></xs:selector>
            <xs:field xpath="in_moon"></xs:field>
      </xs:keyref>
      <xs:keyref name="FK_moon_material_material" refer="K_material">
            <xs:selector xpath="moon_material"></xs:selector>
            <xs:field xpath="of_material"></xs:field>
      </xs:keyref>
      <xs:keyref name="FK_planet_of_star" refer="K_star">
            <xs:selector xpath="planet_of"></xs:selector>
            <xs:field xpath="owner_star"></xs:field>
      </xs:keyref>
      <xs:keyref name="FK_planet_of_planet" refer="K_planet">
            <xs:selector xpath="planet_of"></xs:selector>
            <xs:field xpath="orbiting_planet"></xs:field>
      </xs:keyref>
      <xs:keyref name="FK_moon_owner_planet" refer="K_planet">
            <xs:selector xpath="moon"></xs:selector>
            <xs:field xpath="owner_planet"></xs:field>
      </xs:keyref>
</xs:element>

<xs:complexType name="materials_type">
      <xs:sequence>
            <xs:element name="material" maxOccurs="unbounded"
type="material_type"/>
```

```
      </xs:sequence>
</xs:complexType>
<xs:complexType name="stars_type">
      <xs:sequence>
            <xs:element name="star" maxOccurs="unbounded" type="star_type"/>
      </xs:sequence>
</xs:complexType>
<xs:complexType name="planets_type">
      <xs:sequence>
            <xs:element name="planet" maxOccurs="unbounded" type="planet_type"/>
      </xs:sequence>
</xs:complexType>
<xs:complexType name="moons_type">
      <xs:sequence>
            <xs:element name="moon" maxOccurs="unbounded" type="moon_type"/>
      </xs:sequence>
</xs:complexType>
<xs:complexType name="star_materials_type">
      <xs:sequence>
            <xs:element name="star_material" maxOccurs="unbounded"
type="star_material_type"/>
      </xs:sequence>
</xs:complexType>
<xs:complexType name="planet_materials_type">
      <xs:sequence>
            <xs:element name="planet_material" maxOccurs="unbounded"
type="planet_material_type"/>
      </xs:sequence>
</xs:complexType>
<xs:complexType name="moon_materials_type">
      <xs:sequence>
            <xs:element name="moon_material" maxOccurs="unbounded"
type="moon_material_type"/>
      </xs:sequence>
</xs:complexType>
<xs:complexType name="planet_ofs_type">
      <xs:sequence>
            <xs:element name="planet_of" maxOccurs="unbounded"
type="planet_of_type"/>
      </xs:sequence>
</xs:complexType>

<xs:complexType name="material_type">
      <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="weight" type="xs:decimal"/>
            <xs:element name="usage" type="usage_type"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="usage_type">
      <xs:sequence>
            <xs:element name="use" maxOccurs="unbounded" type="use_type"/>
      </xs:sequence>
</xs:complexType>

<xs:simpleType name="use_type">
      <xs:restriction base="xs:string">
            <xs:enumeration value="fuel"></xs:enumeration>
            <xs:enumeration value="life condition"></xs:enumeration>
            <xs:enumeration value="machinery part"></xs:enumeration>
```

```
            <xs:enumeration value="electronics"></xs:enumeration>
      </xs:restriction>
</xs:simpleType>

<xs:complexType name="star_type">
      <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="mass" type="xs:double"/>
            <xs:element name="radius" type="xs:double"/>
            <xs:element name="heat" type="heat_type"/>
            <xs:element name="type" type="type_of_star"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="heat_type">
      <xs:sequence>
            <xs:element name="surface" type="xs:double"/>
            <xs:element name="core" type="xs:double"/>
      </xs:sequence>
</xs:complexType>

<xs:simpleType name="type_of_star">
      <xs:restriction base="xs:string">
            <xs:enumeration value="yellow dwarf"></xs:enumeration>
            <xs:enumeration value="red dwarf"></xs:enumeration>
            <xs:enumeration value="white dwarf"></xs:enumeration>
            <xs:enumeration value="red giant"></xs:enumeration>
            <xs:enumeration value="blue giant"></xs:enumeration>
            <xs:enumeration value="brown dwarf"></xs:enumeration>
            <xs:enumeration value="neutron star"></xs:enumeration>
            <xs:enumeration value="pulsar"></xs:enumeration>
            <xs:enumeration value="supergiant"></xs:enumeration>
      </xs:restriction>
</xs:simpleType>

<xs:complexType name="planet_type">
      <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="radius" type="xs:double"/>
            <xs:element name="mass" type="xs:double"/>
            <xs:element name="type" type="type_of_planet"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
</xs:complexType>

<xs:simpleType name="type_of_planet">
      <xs:restriction base="xs:string">
            <xs:enumeration value="solid"></xs:enumeration>
            <xs:enumeration value="gas"></xs:enumeration>
            <xs:enumeration value="dwarf"></xs:enumeration>
      </xs:restriction>
</xs:simpleType>

<xs:complexType name="moon_type">
      <xs:sequence>
            <xs:element name="owner_planet" type="xs:string"/>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="mass" type="xs:double"/>
            <xs:element name="radius" type="xs:double"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
```

```
</xs:complexType>

<xs:complexType name="star_material_type">
	<xs:sequence>
		<xs:element name="in_star" type="xs:string"/>
		<xs:element name="of_material" type="xs:string"/>
		<xs:element name="amount" type="xs:double"/>
	</xs:sequence>
</xs:complexType>

<xs:complexType name="planet_material_type">
	<xs:sequence>
		<xs:element name="in_planet" type="xs:string"/>
		<xs:element name="of_material" type="xs:string"/>
		<xs:element name="amount" type="xs:double"/>
	</xs:sequence>
</xs:complexType>

<xs:complexType name="moon_material_type">
	<xs:sequence>
		<xs:element name="in_moon" type="xs:string"/>
		<xs:element name="of_material" type="xs:string"/>
		<xs:element name="amount" type="xs:double"/>
	</xs:sequence>
</xs:complexType>

<xs:complexType name="planet_of_type">
	<xs:sequence>
		<xs:element name="owner_star" type="xs:string"/>
		<xs:element name="orbiting_planet" type="xs:string"/>
		<xs:element name="average_dist" type="xs:double"/>
		<xs:element name="smallest_dist" minOccurs="0" type="xs:double"/>
		<xs:element name="greatest_dist" minOccurs="0" type="xs:double"/>
	</xs:sequence>
</xs:complexType>

</xs:schema>
```

2. feladat

2a) Adatolvasás – DOMReadN5IF3V.java

```
package hu.domparse.n5if3v;

import java.io.*;

import javax.xml.parsers.*;

import org.w3c.dom.*;
import org.xml.sax.SAXException;

import hu.domparse.n5if3v.utils.MyAppException;

public class DOMReadN5IF3V {

	private Document document;

	/*Document Object Model gets read from file in the constructor.*/
	public DOMReadN5IF3V() throws IOException, ParserConfigurationException, SAXException{
		DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
```

```
		DocumentBuilder db = dbf.newDocumentBuilder();
		document = db.parse(new File("XMLN5IF3V.xml"));
	}

	/*Run the class. The class instantiates itself in its main method.*/
	public static void main(String[] args) {
		try {
			DOMReadN5IF3V ownInstance = new DOMReadN5IF3V();
			/*Print all planets.*/
			ownInstance.printAllPlanets();
		}
		catch(Exception exc) {
			exc.printStackTrace();
		}
	}

	/*Function to read all planets.*/
	public void printAllPlanets() throws MyAppException{
		NodeList planets = document.getElementsByTagName("planet");

		/*Go through all planets*/
		for(int i = 0; i < planets.getLength(); i++) {
			/*Get id of planet*/
			NamedNodeMap attributes = planets.item(i).getAttributes();
			String planetId = attributes.getNamedItem("id").getTextContent();
			System.out.println("id: "+planetId);
			Element planetElement = (Element) planets.item(i);
			/*Get local data of planets*/
			String planetName =
planetElement.getElementsByTagName("name").item(0).getTextContent();
			String planetRadius =
planetElement.getElementsByTagName("radius").item(0).getTextContent();
			String planetMass =
planetElement.getElementsByTagName("mass").item(0).getTextContent();
			String planetType =
planetElement.getElementsByTagName("type").item(0).getTextContent();
			double radius = Double.parseDouble(planetRadius);
			double mass = Double.parseDouble(planetMass);
			double planetDensity = mass/(radius*radius*radius*Math.PI*4/3)/1000000000;
			System.out.println("name: "+planetName);
			System.out.println("radius: "+planetRadius+" km");
			System.out.println("mass: "+planetMass+ " kg");
			System.out.println("type: "+planetType);
			System.out.println("density: "+planetDensity+" kg per m3");
			/*Print the name of the planet's sun(s)*/
			NodeList planetOfList = document.getElementsByTagName("planet_of");
			for(int j = 0; j < planetOfList.getLength(); j++) {
				Node planetOf = planetOfList.item(j);
				if(planetOf.getNodeType() == 1) {
					Element planetOfElement = (Element)planetOf;

	if(planetOfElement.getElementsByTagName("orbiting_planet").item(0).getTextContent().equals(planetId)) {
						String starId =
planetOfElement.getElementsByTagName("owner_star").item(0).getTextContent();
						System.out.println("star: "+this.getStarNameById(starId));
					}
				}
			}
			/*Print the name of the planet's moon(s)*/
```

```
                        NodeList moons = document.getElementsByTagName("moon");
                        for(int j = 0; j < moons.getLength(); j++) {
                                Node moon = moons.item(j);
                                if(moon.getAttributes().getNamedItem("id").getTextContent().equals(planetId)) {
                                        if(moon.getNodeType() == 1) {
                                                Element moonElement = (Element) moon;
                                                String moonName =
moonElement.getElementsByTagName("name").item(0).getTextContent();
                                                System.out.println("moon: "+moonName);
                                        }
                                }
                        }
                        /*Print the name and the amount of the material(s) the planet is composed of*/
                        NodeList planetMaterials = document.getElementsByTagName("planet_material");
                        for(int j = 0; j < planetMaterials.getLength(); j++) {
                                Element planetMaterialElement = (Element) planetMaterials.item(j);

        if(planetMaterialElement.getElementsByTagName("in_planet").item(0).getTextContent().equals(planetId)) {
                                        String materialId =
planetMaterialElement.getElementsByTagName("of_material").item(0).getTextContent();
                                        String amount =
planetMaterialElement.getElementsByTagName("amount").item(0).getTextContent();
                                        System.out.println("material: "+amount+" kg of
"+this.getMaterialNameById(materialId));
                                }
                        }
                        System.out.println();
                }
        }

        /*Get the name for the star with the given id*/
        public String getStarNameById(String starId) throws MyAppException{
                NodeList stars = document.getElementsByTagName("star");
                for(int i = 0; i < stars.getLength(); i++) {
                        Element starElement = (Element) stars.item(i);
                        if(starElement.getAttributes().getNamedItem("id").getTextContent().equals(starId)) {
                                String name =
starElement.getElementsByTagName("name").item(0).getTextContent();
                                if(name != null && name.length() > 0) {
                                        return name;
                                }
                                else {
                                        return null;
                                }
                        }
                }
                throw new MyAppException("FATAL: Star not found! Foreign key violated (or coding error)!");
        }

        /*Get the name for the material with the given id*/
        public String getMaterialNameById(String materialId) throws MyAppException{
                NodeList materials = document.getElementsByTagName("material");
                for(int i = 0; i < materials.getLength(); i++) {
                        Element materialElement = (Element) materials.item(i);
                        if(materialElement.getAttributes().getNamedItem("id").getTextContent().equals(materialId))
{
                                String name =
materialElement.getElementsByTagName("name").item(0).getTextContent();
                                if(name != null && name.length() > 0) {
```

```
                                        return name;
                                }
                                else {
                                        return null;
                                }
                        }
                }
                throw new MyAppException("FATAL: Material not found! Foreign key violated (or coding error)!");
        }

}
```

2b) Adatmódosítás – DOMModifyN5IF3V.java

```
package hu.domparse.n5if3v;

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

import javax.xml.XMLConstants;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import javax.xml.validation.*;

import org.w3c.dom.*;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

import hu.domparse.n5if3v.utils.MyAppException;

import java.util.Scanner;

public class DOMModifyN5IF3V {

        private Document document;

        /*Document Object Model gets initialized in the constructor.*/
        public DOMModifyN5IF3V() throws IOException, ParserConfigurationException, SAXException{
                DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
                /*Has to be set so the namespace declaration part of the xml
                 * document won't cause an error at validation.*/
                dbf.setNamespaceAware(true);
                DocumentBuilder db = dbf.newDocumentBuilder();
                document = db.parse(new File("XMLN5IF3V.xml"));
        }

        /*When the class is run, it instantiates itself in the main method.*/
        public static void main(String[] args) {
                try {
                        DOMModifyN5IF3V ownInstance = new DOMModifyN5IF3V();
                        /*Modify the planet.*/
                        ownInstance.consoleModifySession();
                }
                catch(Exception exc) {
                        exc.printStackTrace();
                }
```

```
	}

	/*A modification loop, which uses a console to modify XML objects.*/
	public void consoleModifySession() throws TransformerConfigurationException, TransformerException,
IOException, SAXException{
		Scanner input = new Scanner(System.in);
		String choice;
		do {
			System.out.println("Please, type a letter to continue:\nq to quit, m to modify a planet");
			choice = input.nextLine();
			switch(choice) {
			case "m":
				try {
					this.modifyPlanet();
				}
				catch(SAXParseException exc) {
					System.out.println("Modification failed: "+exc.getMessage());
				}
				catch(MyAppException exc) {
					System.out.println(exc.getMessage());
				}
				break;
			}
		}
		while(!choice.equals("q"));
		System.out.println("Thank you for being the part of this universe! Goodbye!");
	}

	/*Function to modify the local data of the planets.*/
	public void modifyPlanet() throws MyAppException, TransformerConfigurationException,
TransformerException, IOException, SAXException{
		System.out.println("Please, type the id of the planet you want to modify. Available:\n");
		NodeList planets = document.getElementsByTagName("planet");
		for(int i = 0; i < planets.getLength(); i++) {
			System.out.println(planets.item(i).getAttributes().getNamedItem("id").getTextContent());
		}
		Scanner input = new Scanner(System.in);

		/*Choose the planet*/
		String planetId = input.nextLine();
		Element planet = null;
		for(int i = 0; i < planets.getLength(); i++) {
			if(planets.item(i).getAttributes().getNamedItem("id").getTextContent().equals(planetId)) {
				planet = (Element) planets.item(i);
				break;
			}
		}
		if(planet == null) {
			throw new MyAppException("No planet by such id.");
		}

		/*Get the new data of the planet.*/
		System.out.println("Please, write the new NAME of the planet, then press Enter.");
		String name = input.nextLine();
		System.out.println("Please, write the new RADIUS of the planet, then press Enter.");
		String radius = input.nextLine();
		System.out.println("Please, write the new MASS of the planet, then press Enter.");
		String mass = input.nextLine();
		System.out.println("Please, write the new TYPE of the planet (solid, gas, dwarf), then press Enter.");
```

```
                String type = input.nextLine();
                Node planetName = planet.getElementsByTagName("name").item(0);
                planetName.setTextContent(name);
                Node planetRadius = planet.getElementsByTagName("radius").item(0);
                planetRadius.setTextContent(radius);
                Node planetMass = planet.getElementsByTagName("mass").item(0);
                planetMass.setTextContent(mass);
                Node planetType = planet.getElementsByTagName("type").item(0);
                planetType.setTextContent(type);

                /*Validate the new data of the planet*/
                String language = XMLConstants.W3C_XML_SCHEMA_NS_URI;
                SchemaFactory factory = SchemaFactory.newInstance(language);
                Schema schema = factory.newSchema(new File("XMLSchemaN5IF3V.xsd"));
                Validator validator = schema.newValidator();
                DOMSource domsource = new DOMSource(document);
                validator.validate(domsource);

                /*Save the new data of the planet into the file.
                 * First calibrate the transformer.*/
                Transformer tr = TransformerFactory.newInstance().newTransformer();
                tr.setOutputProperty(OutputKeys.INDENT, "yes");
                tr.setOutputProperty(OutputKeys.METHOD, "xml");
                tr.setOutputProperty(OutputKeys.ENCODING, "ISO-8859-2");
                tr.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
                /*Write the result into the file.
                 * XMLN5IF3V_mod.xml is a support file, which exist so the original data don't get
                 * messed up.*/
                tr.transform(domsource, new StreamResult(new FileOutputStream("XMLN5IF3V_mod.xml")));
                System.out.println("Modification succesful!\n");
        }

}
```