Jegyzőkönyv

Webes adatkezelő környezetek

Féléves feladat

Autóalkatrész Webáruház

Készítette: **Borbély Ádám**

Neptunkód: **R41N5H**

[A feladat leírása: 2](#_Toc184484870)

[1.Feladat 3](#_Toc184484871)

[1.1 Az adatbázis ER modelle 3](#_Toc184484872)

[1.2 Az adatbázis XDM modelle 4](#_Toc184484873)

[1.3 Az XDM modell alapján XML dokumentum készítése 4](#_Toc184484874)

[1.4 Az XML dokumentum alapján XMLSchema készítése 8](#_Toc184484875)

[2. feladat 14](#_Toc184484876)

[2.1 adatolvasás 15](#_Toc184484877)

[2.2 adatmódositás 16](#_Toc184484878)

[2.3 adatlekérdezés 17](#_Toc184484879)

[2.4 adatmódosítás 18](#_Toc184484880)

# A feladat leírása:

A feladatomban egy „v9Engine” fantázia névre keresztelt autóalkatrészt árusító webshop adatkezelő rendszerének megtervezését valósítom meg. Az ER modell 6 entitásból áll melyek elnevezésük szerint a következőek: Costumers, CreditCard, Car, Product, Picture és Supplier.

A következőkben vizsgáljuk meg az egyedeket egyesével.

**Costumers**

A Costumers egyed elsődleges kulcsa a CID azaz a Costumer ID mely minden vásárlónál más értéket vesz fel. Az email a vásárló email címét tárolja. A passwd a bejelentkezéshez szükséges jelszavát, a UserName pedig a felhasználó egyedi, az oldalon látható nevét tárolja. Ezeken felül a felhasználó címét és nevét is ez az entitás tartalmazza. Ez két összetett tulajdonság, a Name mely a FirstName vagyis a keresztnév és a LastName, a családnév komponensből áll össze. Valamint az address mely 6 különböző komponensből áll: Country, County, City, ZIP, Street és StreetNo. Az előző sorrendet követve ezek a komponensek a vásárló Országát, régióját, városát, irányítószámát, utcáját és házszámát tárolják.

**CreditCard**

Ez az egyed értelem szerűen a vásárlóval kapcsolatos hisz ez tárolja az adott vásárló bankés/vagy hitelkártyáinak adatait. A legfontosabb a kártyának a száma ami természetesen egyedi és állandó hosszúságú így ez lett az elsődleges kulcs is, ez a mező a CardNo. Mivel különféle fajtájú kártyák lehetnek így szükséges letárolni a kártya típust is amit a Type tulajdonság valósít meg. Az Expire az adott kártya lejárati dátumát tárolja míg a Bank a kártyát kibocsájtó bank nevét.

**Car**

Ez az egyed az adott vásárlónak a regisztrált autóit tartalmazza. Ennek az elsődleges kulcsa az adott autó rendszáma, a PlateNo. Tároljuk még az autó márkáját, Brand és az évjárata az Produced mezőben valamint szükségünk van az autó produced Modell-jére is.

**Product**

A product vagyis a termék egyed elsődleges kulcsa a PID. Tároljuk a termék nevét, Name. A Price a termék árát tartalmazza. Rendelkezik a termék leírással is ami a Description. Valamint a CarType ami tartalmazza, hogy a termék milyen autóhoz, autókhoz és persze évjáratához felel meg. Mivel egy alkatrész lehet, hogy az adott típusból több évjárathoz is megfelel vagy akár több típushoz is megfelelő így ez egy többértékű tulajdonság, valamint összetett is mivel két komponensből áll, a Brand-ből ami az autó márkáját, a Modell, ami az adott modellt és a Produced-ből ami az évjáratát tárolja. Ezen felül a termék rendelkezik érékeléssel is ami a felhasználói tetszéstől ,megelégedésétől függ. A Rates egy összetett tulajdonság, tárolja, hogy ki, Who ami egy vásárló nevét tartalmazza és, hogy milyenre értékelte, Rate ,ami egy szám 1 és 5 között, értelem szerűen az 1 a legkevésbé megelégedett és az 5 a tökéletesen megelégedettséget jelzi.

**Picture**

A Picture egyed tartalmazza a megadott kép eléréséi útvonalát, filePath. A kép címét, Title. Valamint az esetlegesen képhez tartozó leírást, Caption. Valamint egy egyedi elsődleges kulcsot tartalmaz ami a PicID.

**Supplier**

A beszállító vagyis a Supplier entitás tulajdonságai a beszállító neve, Name. A címe, Address ami egy összetett tulajdonság, ami tartalmazza az adott cég telephelyének országát, régióját, városát, annak irányítószámát, utcáját valamint házszámát, melyek a Country, County, City, ZIP, Street és StreetNo tulajdonságok.

# 1.Feladat

## 1.1 Az adatbázis ER modelle

A képen képernyőkép, Színesség, diagram, kör látható

Automatikusan generált leírás

## 1.2 Az adatbázis XDM modelle

ER modellt XDM modellre úgy konvertálhatunk, hogy három féle jelölést használunk. Van az elem, amely az XDM modellben ellipszissel lehet ábrázolni. Az attribútumot rombusszal lehet jelölni, illetve a szöveget a téglalapban. Ezeken felül van még a dupla ellipszis, amely a többszörös előfordulási lehetőséget határozza meg. Ezenkívül minden egyedhez és N:M kapcsolathoz felvesszük az egyed többesszám alakját is.

A képen fekete, képernyőkép, tér, természet látható

Automatikusan generált leírás

## 1.3 Az XDM modell alapján XML dokumentum készítése

*<?*xml version="1.0" encoding="UTF-8"*?>*<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="XMLSchemaR41N5H.xsd">  
 <cars>  
 <car PlateNo="asd123" CID="001">  
 <Brand>Aygo</Brand>  
 <Modell>Toyota</Modell>  
 <Produced>2010</Produced>  
 </car>  
 <car PlateNo="cfg456" CID="002">  
 <Brand>Aygo</Brand>  
 <Modell>Toyota</Modell>  
 <Produced>2010</Produced>  
 </car>  
 <car PlateNo="mnb932" CID="003">  
 <Brand>Swift</Brand>  
 <Modell>Suzuki</Modell>  
 <Produced>2008</Produced>  
 </car>  
 <car PlateNo="kli673" CID="004">  
 <Brand>E60</Brand>  
 <Modell>BMW</Modell>  
 <Produced>2015</Produced>  
 </car>  
 </cars>  
 <CreditCards>  
 <CreditCard CardNO="12344567893" CID="001">  
 <Type>Visa</Type>  
 <Expire>23-10</Expire>  
 <Bank>K and H</Bank>  
 </CreditCard>  
 <CreditCard CardNO="45612378934" CID="002">  
 <Type>Amex</Type>  
 <Expire>23-07</Expire>  
 <Bank>OTP</Bank>  
 </CreditCard>  
 <CreditCard CardNO="98712345634" CID="003">  
 <Type>Visa</Type>  
 <Expire>23-08</Expire>  
 <Bank>ABG</Bank>  
 </CreditCard>  
 <CreditCard CardNO="12344567893" CID="004">  
 <Type>Visa</Type>  
 <Expire>23-02</Expire>  
 <Bank>CA</Bank>  
 </CreditCard>  
 </CreditCards>  
 <Costumers>  
 <Costumer CID="001">  
 <email>dean@gmail.com</email>  
 <passwd>hangpass</passwd>  
 <UserName>Dean</UserName>  
 <Name>  
 <FirstName>Dean</FirstName>  
 <LastName>Winchester</LastName>  
 </Name>  
 <Address>  
 <country>Magyarorszag</country>  
 <County>BAZ</County>  
 <Zip>3530</Zip>  
 <City>Miskolc</City>  
 <Street>Elet utca</Street>  
 <StreetNo>5</StreetNo>  
 </Address>  
 </Costumer>  
 <Costumer CID="002">  
 <email>alal1@yahoo.com</email>  
 <passwd>hunpass</passwd>  
 <UserName>Belika</UserName>  
 <Name>  
 <FirstName>Bela</FirstName>  
 <LastName>Nagy</LastName>  
 </Name>  
 <Address>  
 <country>Magyarorszag</country>  
 <County>BAZ</County>  
 <Zip>3530</Zip>  
 <City>Miskolc</City>  
 <Street>Boldog utca</Street>  
 <StreetNo>90</StreetNo>  
 </Address>  
 </Costumer>  
 <Costumer CID="003">  
 <email>hamar1@freemail.hu</email>  
 <passwd>alapass</passwd>  
 <UserName>UgraBug</UserName>  
 <Name>  
 <FirstName>Alfred</FirstName>  
 <LastName>Gonzalez</LastName>  
 </Name>  
 <Address>  
 <country>Magyarorszag</country>  
 <County>BAZ</County>  
 <Zip>3530</Zip>  
 <City>Miskolc</City>  
 <Street>Hatalom utca</Street>  
 <StreetNo>10</StreetNo>  
 </Address>  
 </Costumer>  
 <Costumer CID="004">  
 <email>nGerg@gmail.com</email>  
 <passwd>nempass</passwd>  
 <UserName>Geri78</UserName>  
 <Name>  
 <FirstName>Gergely</FirstName>  
 <LastName>Németh</LastName>  
 </Name>  
 <Address>  
 <country>Magyarorszag</country>  
 <County>Pest Megye</County>  
 <Zip>1082</Zip>  
 <City>Budapest VIII</City>  
 <Street>Hock Jánós út</Street>  
 <StreetNo>2</StreetNo>  
 </Address>  
 </Costumer>  
 </Costumers>  
 <Requests>  
 <Request ReqID="001" CID="001" PID="001">  
 <ShipMode>Személyes</ShipMode>  
 <Price>10000</Price>  
 <ShipPrice>0</ShipPrice>  
 <PayMode>KP</PayMode>  
 </Request>  
 <Request ReqID="002" CID="004" PID="003">  
 <ShipMode>Futár</ShipMode>  
 <Price>12000</Price>  
 <ShipPrice>10</ShipPrice>  
 <PayMode>BK</PayMode>  
 </Request>  
 <Request ReqID="003" CID="002" PID="002">  
 <ShipMode>Posta</ShipMode>  
 <Price>33</Price>  
 <ShipPrice>2</ShipPrice>  
 <PayMode>UV</PayMode>  
 </Request>  
 </Requests>  
 <Rates>  
 <Rate RiD="001" PID="001">  
 <Who>Belika</Who>  
 <Rate>1</Rate>  
 </Rate>  
 <Rate RiD="002" PID="003">  
 <Who>Dean</Who>  
 <Rate>4</Rate>  
 </Rate>  
 <Rate RiD="003" PID="002">  
 <Who>UgraBug</Who>  
 <Rate>6</Rate>  
 </Rate>  
 </Rates>  
 <Products>  
 <Product Pid="001">  
 <Price>12000</Price>  
 <Descreption>Komplett motor blokk</Descreption>  
 <Name>Motor</Name>  
 </Product>  
 <Product Pid="002">  
 <Price>500</Price>  
 <Descreption>Vezérműszíj</Descreption>  
 <Name>Szíj</Name>  
 </Product>  
 <Product Pid="003">  
 <Price>6000</Price>  
 <Descreption>Toronycsapágy</Descreption>  
 <Name>csapágy</Name>  
 </Product>  
 <Product Pid="004">  
 <Price>3000</Price>  
 <Descreption>Jobb normál saru</Descreption>  
 <Name>Akkumlátor</Name>  
 </Product>  
 </Products>  
 <CarTypes>  
 <CarType CTID="001" PID="001">  
 <Produced>2014</Produced>  
 <Brand>Toyota</Brand>  
 </CarType>  
 <CarType CTID="002" PID="002">  
 <Produced>2008</Produced>  
 <Brand>Toyota</Brand>  
 </CarType>  
 <CarType CTID="003" PID="003">  
 <Produced>2009</Produced>  
 <Brand>Suzuki</Brand>  
 </CarType>  
 <CarType CTID="004" PID="004">  
 <Produced>2011</Produced>  
 <Brand>BMW</Brand>  
 </CarType>  
 </CarTypes>  
 <Pictures>  
 <Picture PicID="001" PID="001">  
 <Title>Enng.jpg</Title>  
 <filePath>/home/pic/toyota</filePath>  
 <Caption>Toy</Caption>  
 </Picture>  
 <Picture PicID="002" PID="002">  
 <Title>Enng.jpg</Title>  
 <filePath>/home/pic/Toyota</filePath>  
 <Caption>Toyota</Caption>  
 </Picture>  
 <Picture PicID="003" PID="003">  
 <Title>Enng.jpg</Title>  
 <filePath>/home/pic/suz</filePath>  
 <Caption>Suz</Caption>  
 </Picture>  
 <Picture PicID="004" PID="004">  
 <Title>Enng.jpg</Title>  
 <filePath>/home/pic/BMW</filePath>  
 <Caption>BMW</Caption>  
 </Picture>  
 </Pictures>  
 <Orders>  
 <Order OrID="001" PID="001" SiD="001">  
 <PartName>Kipufogócső</PartName>  
 <CarType>Toyota</CarType>  
 <Price>3</Price>  
 </Order>  
 <Order OrID="002" PID="002" SiD="002">  
 <PartName>Komplett motor</PartName>  
 <CarType>Ferrari</CarType>  
 <Price>9000</Price>  
 </Order>  
 <Order OrID="003" PID="003" SiD="003">  
 <PartName>Stabilizátor</PartName>  
 <CarType>Audi</CarType>  
 <Price>15</Price>  
 </Order>  
 <Order OrID="004" PID="004" SiD="004">  
 <PartName>Féktárcsa</PartName>  
 <CarType>Chevrolet</CarType>  
 <Price>6</Price>  
 </Order>  
 </Orders>  
 <Suppliers>  
 <Supplier SID="001">  
 <Name>BOSCH</Name>  
 <Address>  
 <country>Magyarorszag</country>  
 <County>BAZ</County>  
 <Zip>3526</Zip>  
 <City>Miskolc</City>  
 <Street>Robert Bosch Park</Street>  
 <StreetNo>1</StreetNo>  
 </Address>  
 </Supplier>  
 <Supplier SID="002">  
 <Name>AUTOMOTIVE</Name>  
 <Address>  
 <country>England</country>  
 <County>AB10</County>  
 <Zip>1JB</Zip>  
 <City>Aberdeen</City>  
 <Street>Albyn Place</Street>  
 <StreetNo>>37</StreetNo>  
 </Address>  
 </Supplier>  
 <Supplier SID="003">  
 <Name>SEMA</Name>  
 <Address>  
 <country>Germany</country>  
 <County>D77</County>  
 <Zip>87499</Zip>  
 <City>Wildpoldsried</City>  
 <Street>Dorfmuehlstr</Street>  
 <StreetNo>>11</StreetNo>  
 </Address>  
 </Supplier>  
 </Suppliers>  
</root>

## 1.4 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="root">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="cars">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="car" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Brand" type="xs:string"/>  
 <xs:element name="Modell" type="xs:string"/>  
 <xs:element name="Produced" type="xs:string"/>  
 </xs:sequence>  
 <xs:attribute name="PlateNo" type="xs:string"/>  
 <xs:attribute name="CID" type="xs:string"/>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <xs:element name="CreditCards">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="CreditCard" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Type" type="xs:string"/>  
 <xs:element name="Expire" type="LejaratiDatum"/>  
 <xs:element name="Bank" type="xs:string"/>  
 </xs:sequence>  
 <xs:attribute name="CardNO" type="Cardtip"/>  
 <xs:attribute name="CID" type="cidnum"/>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <xs:element name="Costumers">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Costumer" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="email" type="xs:string"/>  
 <xs:element name="passwd" type="xs:string"/>  
 <xs:element name="UserName" type="xs:string"/>  
 <xs:element name="Name">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="FirstName" type="xs:string"/>  
 <xs:element name="LastName" type="xs:string"/>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <xs:element name="Address">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="country" type="xs:string"/>  
 <xs:element name="County" type="xs:string"/>  
 <xs:element name="Zip" type="xs:integer"/>  
 <xs:element name="City" type="xs:string"/>  
 <xs:element name="Street" type="xs:string"/>  
 <xs:element name="StreetNo" type="xs:string"/>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 <xs:attribute name="CID" type="xs:integer"/>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <xs:element name="Requests">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Request" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="ShipMode" type="xs:string"/>  
 <xs:element name="Price" type="xs:string"/>  
 <xs:element name="ShipPrice" type="xs:string"/>  
 <xs:element name="PayMode" type="xs:string"/>  
 </xs:sequence>  
 <xs:attribute name="ReqID" type="xs:integer"/>  
 <xs:attribute name="CID" type="xs:integer"/>  
 <xs:attribute name="PID" type="xs:integer"/>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <xs:element name="Rates">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Rate" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Who" type="xs:string"/>  
 <xs:element name="Rate" type="xs:string"/>  
 </xs:sequence>  
 <xs:attribute name="RiD" type="xs:string"/>  
 <xs:attribute name="PID" type="xs:string"/>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <xs:element name="Products">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Product" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Price" type="xs:integer"/>  
 <xs:element name="Descreption" type="xs:string"/>  
 <xs:element name="Name" type="xs:string"/>  
 </xs:sequence>  
 <xs:attribute name="Pid" type="xs:integer"/>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <xs:element name="CarTypes">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="CarType" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Produced" type="xs:integer"/>  
 <xs:element name="Brand" type="xs:string"/>  
 </xs:sequence>  
 <xs:attribute name="CTID" type="xs:integer"/>  
 <xs:attribute name="PID" type="xs:integer"/>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <xs:element name="Pictures">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Picture" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Title" type="xs:string"/>  
 <xs:element name="filePath" type="xs:string"/>  
 <xs:element name="Caption" type="xs:string"/>  
 </xs:sequence>  
 <xs:attribute name="PicID" type="xs:integer"/>  
 <xs:attribute name="PID" type="xs:integer"/>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <xs:element name="Orders">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Order" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="PartName" type="xs:string"/>  
 <xs:element name="CarType" type="xs:string"/>  
 <xs:element name="Price" type="xs:integer"/>  
 </xs:sequence>  
 <xs:attribute name="OrID" type="xs:integer"/>  
 <xs:attribute name="PID" type="xs:integer"/>  
 <xs:attribute name="SiD" type="xs:integer"/>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <xs:element name="Suppliers">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Supplier" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="Name" type="xs:string"/>  
 <xs:element name="Address">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="country" type="xs:string"/>  
 <xs:element name="County" type="xs:string"/>  
 <xs:element name="Zip" type="xs:string"/>  
 <xs:element name="City" type="xs:string"/>  
 <xs:element name="Street" type="xs:string"/>  
 <xs:element name="StreetNo" type="xs:string"/>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 <xs:attribute name="SID" type="xs:integer"/>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 <xs:key name="Car\_Key">  
 <xs:selector xpath="car"/>  
 <xs:field xpath="@PlateNo"/>  
 </xs:key>  
 <xs:key name="CreditCard\_Key">  
 <xs:selector xpath="CreditCard"/>  
 <xs:field xpath="@CardNO"/>  
 </xs:key>  
 <xs:key name="Costumer\_Key">  
 <xs:selector xpath="Costumer"/>  
 <xs:field xpath="@CID"/>  
 </xs:key>  
 <xs:key name="Request\_Key">  
 <xs:selector xpath="Request"/>  
 <xs:field xpath="@ReqID"/>  
 </xs:key>  
 <xs:key name="Rate\_key">  
 <xs:selector xpath="Rate"/>  
 <xs:field xpath="@RiD"/>  
 </xs:key>  
 <xs:key name="Product\_key">  
 <xs:selector xpath="Product"/>  
 <xs:field xpath="@PiD"/>  
 </xs:key>  
 <xs:key name="CarTypes\_key">  
 <xs:selector xpath="CarType"/>  
 <xs:field xpath="@CTID"/>  
 </xs:key>  
 <xs:key name="Picture\_key">  
 <xs:selector xpath="Picture"/>  
 <xs:field xpath="@PicID"/>  
 </xs:key>  
 <xs:key name="Order\_key">  
 <xs:selector xpath="Order"/>  
 <xs:field xpath="@OrID"/>  
 </xs:key>  
 <xs:key name="Supplier\_key">  
 <xs:selector xpath="Supplier"/>  
 <xs:field xpath="@SID"/>  
 </xs:key>  
 <xs:keyref name="Fkey\_Car\_Costumer" refer="Costumer\_Key">  
 <xs:selector xpath="car"/>  
 <xs:field xpath="@CID"/>  
 </xs:keyref>  
 <xs:keyref name="Fkey\_CreditCard\_Costumer" refer="Costumer\_Key">  
 <xs:selector xpath="CreditCard"/>  
 <xs:field xpath="@CID"/>  
 </xs:keyref>  
 <xs:keyref name="Fkey\_Request\_Costumer" refer="Costumer\_Key">  
 <xs:selector xpath="Request"/>  
 <xs:field xpath="@CID"/>  
 </xs:keyref>  
 <xs:keyref name="Fkey\_Request\_Product" refer="Product\_key">  
 <xs:selector xpath="Request"/>  
 <xs:field xpath="@PID"/>  
 </xs:keyref>  
 <xs:keyref name="Fkey\_CarType\_Product" refer="Product\_key">  
 <xs:selector xpath="CarType"/>  
 <xs:field xpath="@PID"/>  
 </xs:keyref>  
 <xs:keyref name="Fkey\_Rates\_Product" refer="Product\_key">  
 <xs:selector xpath="Rate"/>  
 <xs:field xpath="@PID"/>  
 </xs:keyref>  
 <xs:keyref name="Fkey\_Product\_Order" refer="Product\_key">  
 <xs:selector xpath="Order"/>  
 <xs:field xpath="@PID"/>  
 </xs:keyref>  
 <xs:keyref name="Fkey\_Picture\_Product" refer="Product\_key">  
 <xs:selector xpath="Picture"/>  
 <xs:field xpath="@PID"/>  
 </xs:keyref>  
 <xs:keyref name="Fkey\_Order\_Supplier" refer="Supplier\_key">  
 <xs:selector xpath="Order"/>  
 <xs:field xpath="@SID"/>  
 </xs:keyref>  
 </xs:element>  
 <xs:simpleType name="LejaratiDatum">  
 <xs:restriction base="xs:string">  
 <xs:pattern value="[0-9]{1}[0-9]{1}-[0-9]{1}[0-9]{1}" />  
 </xs:restriction>  
 </xs:simpleType>  
 <xs:simpleType name="Cardtip">  
 <xs:restriction base="xs:string">  
 <xs:pattern value="[0-9]{11}" />  
 </xs:restriction>  
 </xs:simpleType>  
 <xs:simpleType name="cidnum">  
 <xs:restriction base="xs:string">  
 <xs:pattern value="[0-9]{3}" />  
 </xs:restriction>  
 </xs:simpleType>  
</xs:schema>

# 2. feladat

## 2.1 adatolvasás

A DomWrite kódomból kiemelnék néhány érdekesebb megoldást.

Lista elemek kinyerése:

list=document.getElementsByTagName("car");

for (int i=0;i<list.getLength();i++) {

Node node=list.item(i);

System.***out***.println("\nAktuális elem: " + node.getNodeName());

***getElementsByTagName***: Az XML fájl összes olyan elemét visszaadja, amelynek neve megegyezik a megadott tag névvel (jelen esetben car).

**Iterálás**: A ***list.getLength*()** adja meg, hogy hány elem található a listában, amelyen egy egyszerű for-ciklussal iterálunk.

***node.getNodeName***: Az aktuális csomópont nevét adja vissza, ami segíthet az XML elem azonosításában.

Elem és attribútok feldolgozása:

if(node.getNodeType() == Node.***ELEMENT\_NODE***) {

Element element=(Element) node;

System.***out***.println("Plate Number: " + element.getAttribute("PlateNo"));

System.***out***.println("CID: " + element.getAttribute("CID"));

System.***out***.println("Brand: "

+ element.getElementsByTagName("Brand").item(0).getTextContent());

System.***out***.println("Modell: "

+ element.getElementsByTagName("Modell").item(0).getTextContent());

System.***out***.println("Produced: "

+ element.getElementsByTagName("Produced").item(0).getTextContent());

}

**Elem típusának ellenőrzése**: A ***getNodeType()*** metódus segítségével ellenőrizzük, hogy a csomópont típusa ***ELEMENT\_NODE***, vagyis egy XML elem. Csak ekkor dolgozzuk fel az elem tartalmát.

**Attribútumok lekérése**: Az ***element.getAttribute()*** metódussal közvetlenül kinyerhetjük az elem attribútumait (pl. PlateNo, CID).

**Gyerekelemek tartalmának lekérése**: Az ***getElementsByTagName()*** metódus visszaadja a gyerekelemek listáját, és az ***item(0).getTextContent()*** az első elem szöveges tartalmát adja vissza.

Többszintű elemek kezelése:

Node nodenev=list.item(i);

if(nodenev.getNodeType()==Node.***ELEMENT\_NODE***){

Element elementnev=(Element) node;

System.***out***.println("Születési idő: " + elementnev.getElementsByTagName("FirstName").item(0).getTextContent());

System.***out***.println("Születési hely: " + elementnev.getElementsByTagName("LastName").item(0).getTextContent());

**Többszintű elemek**: Itt a Costumer elem alatti Name gyerekelemek (pl. FirstName, LastName) tartalmát nyerjük ki.

**Hierarchikus feldolgozás**: Ez a technika hasznos a mélyebb XML struktúrák bejárásához.

## 2.2 adatmódositás

A Hitelkártya táblából megkeresi azt a hitelkártyát akinek a kártyaszámja-12344567893

majd a lejárati dátumot módosítja a kódban megadottra majd egy új xml fájlba kiírja a módosított xml fájlt.

package hu.domparse.R41N5H;  
  
import org.w3c.dom.Document;  
import org.w3c.dom.Element;  
import org.w3c.dom.Node;  
import org.w3c.dom.NodeList;  
  
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.ParserConfigurationException;  
import javax.xml.transform.Result;  
import javax.xml.transform.Source;  
import javax.xml.transform.Transformer;  
import javax.xml.transform.TransformerFactory;  
import javax.xml.transform.dom.DOMSource;  
import javax.xml.transform.stream.StreamResult;  
import java.io.File;  
  
public class DomModifyR41N5H {  
 public static void main(String[] args) {  
 NodeList nodeList;  
 try{  
 DocumentBuilderFactory factory =DocumentBuilderFactory.*newInstance*();  
 DocumentBuilder builder=factory.newDocumentBuilder();  
 *//Fájl beolvasása* Document document=builder.parse(new File("XMLR41N5H.xml"));  
 document.getDocumentElement().normalize();  
 *//Aktuális elem meghatározása* nodeList=document.getElementsByTagName("CreditCard");  
 for(int i =0;i<nodeList.getLength();i++){  
 Node node=nodeList.item(i);  
 System.*out*.println("\nAktuális elem: " + node.getNodeName());  
 *//Kártya adatainak kiirása* if(node.getNodeType()==Node.*ELEMENT\_NODE*){  
 Element element=(Element) node;  
 *//Kártyaszámának eltárolása* String id= element.getAttribute("CardNO");  
  
 System.*out*.println("Card Number: " + element.getAttribute("CardNO"));  
 System.*out*.println("Type: " + element.getElementsByTagName("Type").item(0).getTextContent());  
 System.*out*.println("Expire: "  
 + element.getElementsByTagName("Expire").item(0).getTextContent());  
 System.*out*.println("Bank: " + element.getElementsByTagName("Bank").item(0).getTextContent());  
  
 if (id.equals("12344567893")){  
 ((Element) node).getElementsByTagName("Expire").item(0).setTextContent("24-10");  
 }  
 }  
 }  
 *//létrehozza az xml fájlt módositva* Transformer transformer= TransformerFactory.*newInstance*().newTransformer();  
 Source input=new DOMSource(document);  
 Result output=new StreamResult(new File("XMLR41N5HModify.xml"));  
 transformer.transform(input,output);  
 }catch (ParserConfigurationException e){  
 e.printStackTrace();  
 } catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
}

## 2.3 adatlekérdezés

A termék táblából kiírja ki listázza azokat, amelyeknek az ára több mint 3000, és kiírja a consolera.

package hu.domparse.R41N5H;  
  
import org.w3c.dom.Document;  
import org.w3c.dom.Element;  
import org.w3c.dom.Node;  
import org.w3c.dom.NodeList;  
import org.xml.sax.SAXException;  
  
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.ParserConfigurationException;  
import java.io.File;  
import java.io.IOException;  
import static java.lang.Integer.*parseInt*;  
  
public class DomQueryR41N5H {  
 public static void main(String[] args) {  
 try {  
 NodeList nodeList;  
  
 DocumentBuilderFactory factory =DocumentBuilderFactory.*newInstance*();  
 DocumentBuilder builder=factory.newDocumentBuilder();  
 *//Fájl beolvasása* Document document=builder.parse(new File("XMLR41N5H.xml"));  
 document.getDocumentElement().normalize();  
  
 *//Aktuális elem meghatározása* nodeList=document.getElementsByTagName("Product");  
  
 for(int i =0;i<nodeList.getLength();i++){  
 Node node=nodeList.item(i);  
  
 *//Product adatainak kiirása* if(node.getNodeType()==Node.*ELEMENT\_NODE*){  
 Element element=(Element) node;  
 int price= *parseInt*(element.getElementsByTagName("Price").item(0).getTextContent());  
  
 if(price>3000)  
 {  
 System.*out*.println("\nAktuális elem: " + node.getNodeName());  
 System.*out*.println("-------");  
 System.*out*.println("");  
 System.*out*.println("Product id: " + element.getAttribute("Pid"));  
 System.*out*.println("Price: " + element.getElementsByTagName("Price").item(0).getTextContent());  
 System.*out*.println("Descreption: " + element.getElementsByTagName("Descreption").item(0).getTextContent());  
 System.*out*.println("Name: " + element.getElementsByTagName("Name").item(0).getTextContent());  
  
 }  
 }  
 }  
 } catch (ParserConfigurationException | IOException | SAXException e){  
 e.printStackTrace();  
 }  
 }  
}

## 2.4 adatmódosítás

Az importálás után inicializájuk a DOM objektumot és betöltjük az XML fájlt

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.Result;

import javax.xml.transform.Source;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import java.io.File;

public class DomModifyR41N5H {

public static void main(String[] args) {

NodeList nodeList;

try{

DocumentBuilderFactory factory =DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder=factory.newDocumentBuilder();

Document document=builder.parse(new File("XMLR41N5H.xml"));

document.getDocumentElement().normalize();

Ezután bejárjuk az elemeket és módosítjuk:

**Elemek lekérése**: Az ***document.getElementsByTagName("CreditCard")*** minden CreditCard nevű elemet visszaad az XML fájlból.

**Iterálás**: A nodeList listán végig haladunk, és minden CreditCard elem feldolgozásra kerül.

**Attribútum ellenőrzés**: Az adott CreditCard elem attribútumainak (pl. CardNO) értékét ellenőrizzük.

**Módosítás**: Ha a CardNO attribútum értéke 12344567893, akkor az adott Expire elem szöveges tartalmát "24-10" értékre változtatjuk.

nodeList=document.getElementsByTagName("CreditCard");

for(int i =0;i<nodeList.getLength();i++){

Node node=nodeList.item(i);

System.***out***.println("\nAktuális elem: " + node.getNodeName());

if(node.getNodeType()==Node.***ELEMENT\_NODE***){

Element element=(Element) node;

String id= element.getAttribute("CardNO");

System.***out***.println("Card Number: " + element.getAttribute("CardNO"));

System.***out***.println("Type: " + element.getElementsByTagName("Type").item(0).getTextContent());

System.***out***.println("Expire: "

+ element.getElementsByTagName("Expire").item(0).getTextContent());

System.***out***.println("Bank: " + element.getElementsByTagName("Bank").item(0).getTextContent());

if (id.equals("12344567893")){

((Element) node).getElementsByTagName("Expire").item(0).setTextContent("24-10");

}

}

}

Majd elmentjük az XML fájlt:

**Transformer létrehozása**: A ***TransformerFactory*** segítségével létrehozunk egy Transformer objektumot, amely az XML átalakításáért felelős.

**Forrás és kimenet**: A módosított Document objektumot (az XML fájl DOM reprezentációja) bemenetként adjuk meg, és egy új fájlba ("XMLR41N5HModify.xml") mentjük az eredményt.

**Mentés**: A ***transformer.transform()*** metódus elvégzi az XML módosításainak mentését az új fájlba.

Transformer transformer= TransformerFactory.*newInstance*().newTransformer();

Source input=new DOMSource(document);

Result output=new StreamResult(new File("XMLR41N5HModify.xml"));

transformer.transform(input,output);

Legvégül kezeljük a kivételeket:

***ParserConfigurationException***: Ha probléma merül fel az XML parser konfigurációjával.

***Exception***: Általános hibakezelés, beleértve az XML fájl elérését vagy a DOM manipulációval kapcsolatos hibákat.

}catch (ParserConfigurationException e){

e.printStackTrace();

} catch (Exception e){

e.printStackTrace();

}

}

}