JEGYZŐKÖNYV

Adatkezelés XML-ben

Féléves feladat

Képregény kiadó

**Készítette: Györki Ádám**

**Neptunkód: A7TIJX**

**Dátum: 12.05**

***Feladat leírása:***

Feladatom témáját foglalkozásom miatt a képregény piac, azon belül is a világ legnagyobb képregénypiaca, a japán. Ezt a képregénykiadók, a képregények (mangák) készítői és a könyvesboltok közötti kapcsolattal lehet megfelelő módon bemutatni. Mivel egy borzasztó nagy gépezetről van szó így egy, a feladathoz illő modellt lehet belőle szerkeszteni, azon keresztül pedig nagyon jól be lehet mutatni mi hogyan működik ezen a rendszeren belül. Mivel elemenként vizsgáljuk a szereplőket így a ránézésre bonyolult szisztémát részekre tudjuk bontani és ha így figyeljük meg az egészet akkor könnyen érthetővé válik a működés. Persze akár tovább is ki lehetne egészíteni ezt például a nyomdával is vagy más szereplőkkel, mint például a szerkesztők, vagy akár a teljes marketing gépezetet ide lehetne felsorakoztatni, ennél a modellénél viszont csak a leglényegesebb részeket vizsgáljuk, hogy hogyan jut el a képregény alkotótól az olvasóig a legújabb történet. Ahogy máskor is, most is fontosnak tartom, hogy autentikus legyen a feladat ezért valós adatokkal dolgoztam, ezeknek megfelelően utána néztem

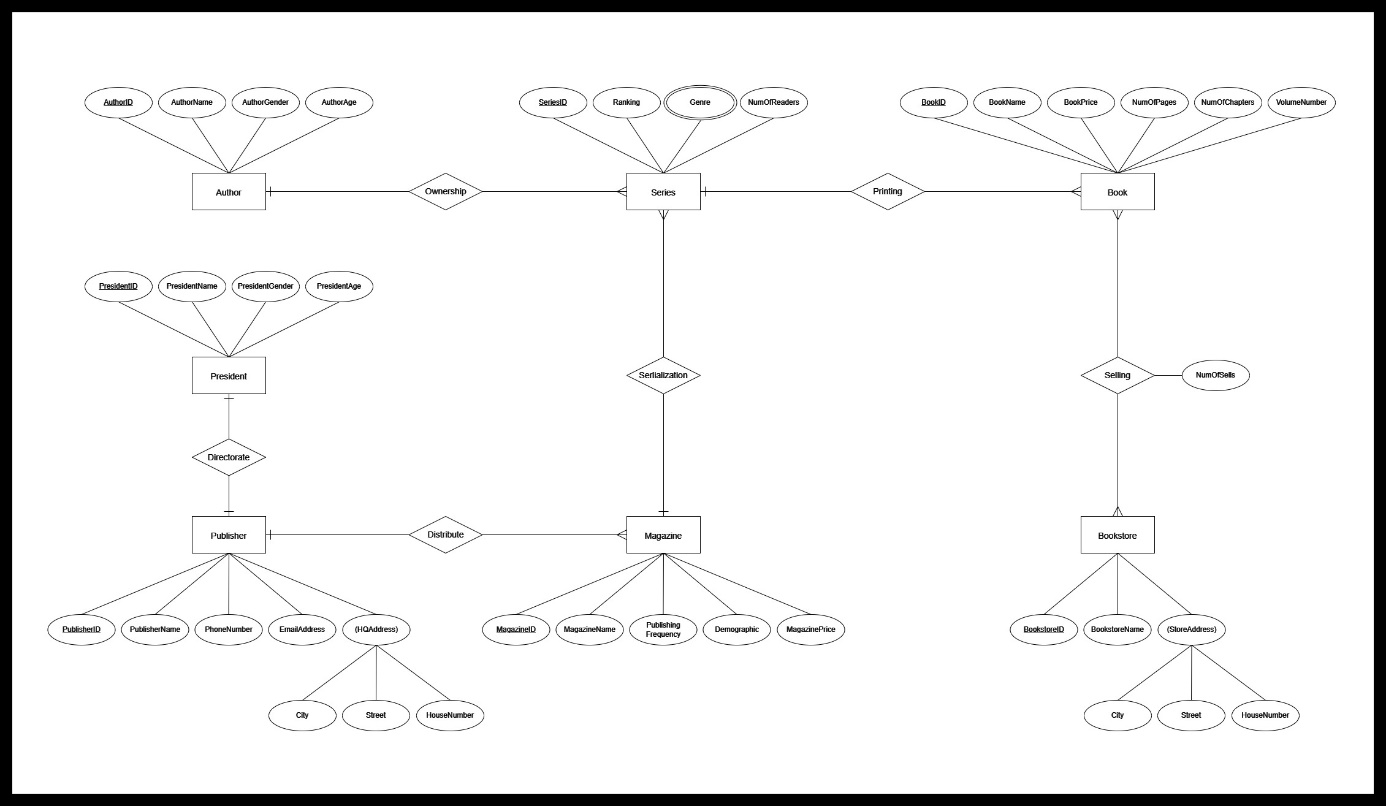
Maga a japán képregény piac a következőképpen működik: a képregénykészítők/mangakák elkészítik a fejezeteket, amiknek oldalszáma függ attól, hogy milyen megjelenési gyakoriságú magazinnál dolgoznak. Miután kész vannak ezt leadják és ez kikerül a magazinba, ami megjelenik a hét/hónap/negyed év elején. Ebben a magazinban mindig a legfrissebb fejezetek vannak. Ezeket tömeg gyártják és ezt adják el újságosoknál, könyvesboltokban vagy egyszerű vegyesboltokban. A magazin egy adott kiadóhoz tartozik, akik több magazint is futtatnak amelyek különböző megjelenési gyakorisággal rendelkeznek és más korosztálynak készülnek. A sounen tipikusan fiatal fiúknak, a seinen felnőtt férfiaknak, a shojo fiatal lányoknak és a josei felnőtt nőknek készül. Minden kiadónak van egy elnöke aki főként hozza a döntéseket a kiadóval kapcsolatban. Miután összegyűlt egy adag (általában úgy 200-250 oldalnyi) fejezet, ezeket könyvekbe gyűjtik és így kiadják. Ezeket tudják venni az olvasók gyűjtésre és a szerző támogatására. Ezekben a magazinnal ellentétben csak egy szerző munkája van természetesen. Ezeket könyveket szórják ki különböző könyvesboltokban, ahol eladják őket a vevők számára.

**1.feladat**

***1a) Az adatbázis ER modell tervezése***

Az ER modell leírja a képregényiparban lévő szereplők közti kapcsolatokat, hogy hogyan viszonyulnak egymáshoz a felek. Ezeknek a feleknek vannak különféle adataik, amik fel vannak itt tüntetve valamint a kapcsolati tulajdonságuk is.

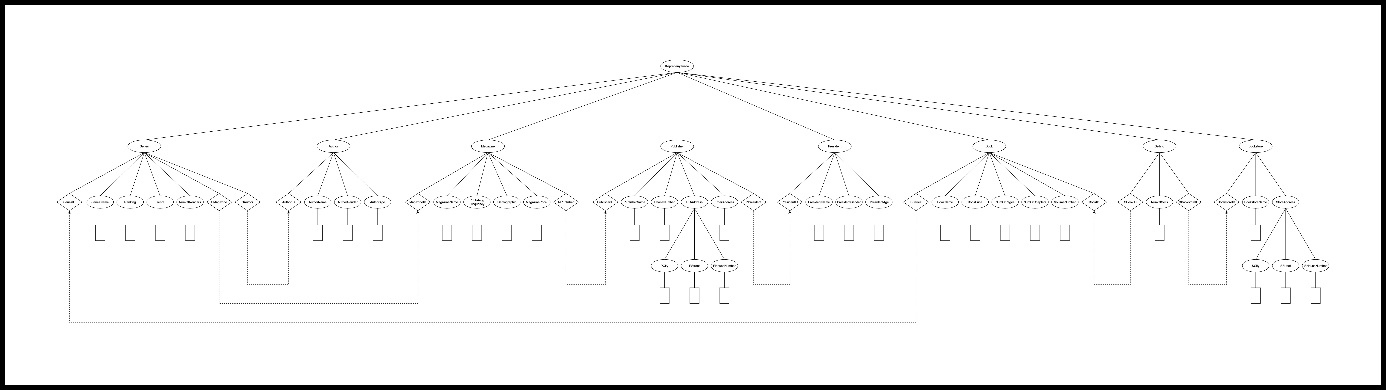
Az Author a képregényszerző, a Series a képregényszériák amiket az szerzők alkotnak és a magazinokban jelennek meg. A magazin (Magazine) a kiadó által vezetett nyomtatott képregények gyűjteménye ami adott időközönként jelenik meg. A kiadó (Publisher) amely üzemelteti a magazint, valamint ennek van egy igazgatója/vezetője. A könyvek (Book) azok a képregényszériák fejezeteiből összegyűjtött nyomtatott képregények amelyeket a könyvesboltokban (Bookstore) árulnak.



*ERA7TIJX.jpg*

***1b) Az adatbázis konvertálása XDM modellre***

Az XDM modellnél a legnagyobb hangsúlyt a a fa struktúra adja, erre kell lényegében átkonvertálnunk az eredeti relációs modellt. A különböző elemek alkotják a korábban említett szereplőket és itt is különbözőfél tulajdonságokkal látjuk el amik az elemek gyerekelemei lesznek. Ami még nehéz lehet ennél a folyamatnál azok az attribútumok és a közöttük fennálló kapcsolatok kialakítása. Eleve nem mindegy hogy milyen kapcsolat van az elemek között mert például ha több:több, akkor mindjárt be kell hoznunk egy újabb elemet (ez történt a Book és a Bookstore között. Ahogy az ER modellnél itt és érdemes figyelni a szépen kidolgozott küllemre és az átláthatóságra.



*XDMA7TIJX.jpg*

***1c) Az XDM modell alapján XML dokumentum készítése***

Ha az XDM modellt megfelelően elkészítettük akkor sok feladatunk nem is lesz az XML-el mivel lényegében ezzel megadtunk egy 1:1 sémát az XML kialakításához. A fa struktúrának megfelelően kialakítjuk az elemeket, hozzájuk írjuk a gyerekeiket és az attribútumokat. A többször előforduló elemeket próbálom apró változtatásokkal elnevezni, mint ahogy megkülönböztetem például az igazgató és az alkotó nemét vagy a kiadó és a könyvesbolt címét. Továbbra is valós adatokkal dolgozunk ezért nézhet ki furcsán elsőre a telefonszám, az email és a címek formátuma. A magazinokat yenben számoljuk azonban a könyveket dollárban, ez azért van így mivel a könyveket később lefordítják japánról angolra és különböző nyelvekre ami a világ minden pontjára elérést biztosít és a dollár egy jó számérték ha általánosságban akarjuk mérni világszerte az árát a könyvnek.

A Seriesben láthatjuk gyerekelemként a széria nevét, jelenlegi rankját a többihez képest, a műfaját és hogy mennyien olvassák. Attribútumoknál ki van emelve az ID-je valamint hogy melyik számú szerzőhöz és magazinhoz tartozik.

Az Authornál a szerző nevét, nemét és életkorát láthatjuk, valamint a saját IDját amire például hivatkozik a Series elem.

A Magazine-nál a magazin nevét látjuk, azt hogy milyen gyakran jelenik meg, hogy milyen korosztálynak szánták és hogy mennyibe is kerül egy átlag japán boltban. Ezen felül neki is van saját IDje és a tárol egy számot még ami megmutatja melyik kiadóhoz tartozik.

A Publishernél fel van tüntetve a kiadó neve, elérhetősége, mint email és telefon, valamint a címe, ami a városból, utcából és házszámból áll (ez az egész egy igen különös rendszert követ, amibe most nem mennék bele, a japán címek nagyon máshogy néznek ki mint itt Európában például, de a lényeg ugyan az. Ez is tartalmazza a saját IDjét és a kiadó vezetőjének, az elnöknek a számát.

A Bookban a könyvek általános adatait láthatjuk, mi a nevük, mi az áruk, hány fejezetből állnak, hány oldalból állnak és hányadik számú kötete a szériának a sorozatban.

A Bookstore a könyvesbolt nevét, és címét tartalmazza és utoljára marad a Selling ami összeköti a könyvet és a könyvesboltot ID tekintetében valamint tartalmaz egy eladási számot is a relációk fényében.

*(XMLA7TIJX.xml) ---*

A képen szöveg, képernyőkép, menü látható

Automatikusan generált leírás

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

A képen szöveg, képernyőkép látható

Automatikusan generált leírásA képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás

***1d) Az XML dokumentum alapján XMLSchema készítése***

Az XSD fájl az XML fájl létrehozásának segítségéül szolgál, előírja hogy mit, hogyan írhatunk bele a fájlba, akárcsak egy interface. Itt először egyszerű típusokat hoztam létre amelyek a legalapabb, legegyszerűbb elemeket képzik le, ezután ezeknek a referálásával komplex típusokat csináltam, majd saját típusokat. Ezeket végül szintén referálással felhasználom a végleges ábrázolásnál, ezért néz ki nagyon egyszerűen a felépítése a fájlnak. Fontos volt még hogy különböző korlátozásokat írjunk elő bizonyos elemekhez és az attribútumokhoz, például hogy mi legyen a típusa (int, string, saját), hogy kötelező szerepelnie e vagy hogy abból az adott elemből hány hozható létre az XML fájlban.

*(XMLSchemaA7TIJX.xsd) ---*

A képen szöveg, képernyőkép látható

Automatikusan generált leírásA képen szöveg, képernyőkép, menü látható

Automatikusan generált leírásA képen szöveg, képernyőkép, menü látható

Automatikusan generált leírásA képen szöveg, képernyőkép, menü látható

Automatikusan generált leírásA képen szöveg, képernyőkép látható

Automatikusan generált leírás

**2. feladat**

***2a) adatolvasás***

Az adatolvasónál létrehoztam az XML readert amivel betápláltam a programba az XML fájlt majd ezt Node-ok és NodeList-ek formájában feldaraboltam, majd ezeket eltároltam előre elkészített osztályok szerint az osztályokhoz tartozó metódusokkal. Azért maradtam a Listeknél mivel így jól átlátható így hogy kerültek tárolásra az adatok valamint a Listnek vannak hasznos funkciói amit később tudunk kamatoztatni. Az osztályoknál természetesen az adattagok az elemek és az attribútumok, az alattuk megtalálható metódusok pedig az előre létrehozott lista feltöltését szolgálják. Miután el lettek tárolva a Node-ok a listákba nagyon könnyen, egyszerű hivatkozásokkal ki tudjuk őket íratni konzolra. A feltöltő metódus úgy működik, hogy a NodeList-ből kiolvassuk a Nodeokat (az elemeket, gyerekelemeket) egy-egy ciklusban és a megfelelő tag-el rendelkező elemeket belerakjuk az újonnan létrehozott osztályokba. Egy következő metódus pedig azért felelős, hogy ezeket az elemeket List-be szedje.

*(DomReadA7TIJX.java).xsd* ---

A képen szöveg, képernyőkép látható

Automatikusan generált leírásA képen képernyőkép, szöveg, szoftver látható

Automatikusan generált leírásA képen képernyőkép, szöveg látható

Automatikusan generált leírás

*Az itt listákhoz felhasznált osztályok a következőket tartalmazzák:*

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírásA képen szöveg, képernyőkép látható

Automatikusan generált leírásA képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírásA képen szöveg, képernyőkép látható

Automatikusan generált leírásA képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírásA képen szöveg, képernyőkép látható

Automatikusan generált leírásA képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírásA képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

*Ezt kapjuk kiírásként a konzolon:*

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

***2b) adatmódosítás***

Adatmódosításnál miután létrehoztuk a kódban a dokumentumot és beolvastuk az adatokat, elkezdjük a belső módosításokat külön erre a célra megírt függvényekkel. Ezeknek a függvényeknek meg kell adni hogy melyik főbb elem hányadik példányának melyik gyerekelemét kívánjuk módosítani és hogy mire. Miután megtörtént a dokumentumon belül a módosítás egy Transformer és a FileOutput segítségével frissítjük a korábban beolvasott XML-t segítségével (ebben az esetben új fájlt hoztam létre, hogy látszódjon a különbség, de rá is lehetne menteni).

*(DomModifyA7TIJX.java) ---*

A képen szöveg, képernyőkép, tervezés látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás

*Ezek a módosított elemek:*

*A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás*

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás *A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás*   A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

***2c) adatlekérdezés***

Az adatok lekérdezése az osztályokból származó előnyök miatt igen egyszerűen megvalósítható. Sima ciklusokkal át lehet futni és meg lehet keresni a kívánt elemeket amit további keresésekhez használhatunk, így összetett lekérdezéseket is megvalósíthatunk.

*(DomQueryA7TIJX.java) ---*

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

A képen szöveg, képernyőkép látható

Automatikusan generált leírás A képen szöveg, képernyőkép látható

Automatikusan generált leírás

*A lekérdezések eredménye:*

*A képen szöveg, képernyőkép látható

Automatikusan generált leírás*

***2d) adatírás***

Az adatok kiírása szintén a Transformerrel történik egyszerű módon, akárcsak a modify esetében. Betápláltam a dokumentumot majd az felhasználva létrehoztam egy új fájlt.

*(DomWriteA7TIJX.java) ---*

A képen szöveg, képernyőkép látható

Automatikusan generált leírás