A movies.db adatbázis Movies táblájában filmek, és a hozzájuk tartozó különböző adatok szerepelnek. 1 sor 1 filmet tartalmaz, illetve annak adatait. Viszont a tulajdonságok között több olyan mező is van, amely vesszővel elválasztott felsorolásokat tartalmaz.

Ezeknek a mezőknek a tartalmát ebben az állapotban csak egyben lehet felhasználni, ez pedig nem előnyös. Ha a filmeket össze szeretnénk hasonlítani, esetleg csoportosítani, akkor ez esetben arra van szükség, hogy a közös tulajdonságaikat vizsgáljuk meg: vannak-e egyáltalán, ha igen, akkor melyek azok? Amennyiben egy olyan tulajdonságot vizsgálunk, amelyhez tartozó mező több adatot is tartalmazhat (pl. egy film kulcsszavainak vizsgálata esetén a film nem csak egy kulcsszóval rendelkezhet, hanem felsorolás szerűen többel is), akkor arra rendkívül kicsi esély van, hogy az adott tulajdonságot vizsgálva a 2 film 100%-osan megegyezzen. Viszont ettől még lehet számos egyezés, azonban ha az adatokat egyben vizsgáljuk akkor ezeket nem tudjuk megtalálni, ezért szükséges ezen mezők lebontása, hogy a tartalmukban szereplő különböző adatokat külön-külön tudjuk kezelni, ne pedig egyben.

Tehát a mezőket le kell bontani, hogy kezelhetőek legyenek. Ebben az esetben szükség van egy külön táblára, amiben az éppen lebontott mező összes adata szerepel ismétlődések nélkül úgy, hogy minden egyes adat egy új sor. Tehát ahol eddig vesszők voltak, azok az adatok külön-külön új sorba kerülnek. Ahhoz, hogy az így kapott adatokat hozzá tudjuk rendelni a filmekhez, mivel több-több kapcsolat van az adatok között, így szükség lesz annyi kapcsolótáblára is, ahány új táblánk keletkezett az adatok lebontása során. A TableInserts nevű C# program olyan egyszerű konzolos alkalmazás, amely ezen táblák, illetve kapcsolótáblák feltöltését végzi el.

**A lebontás folyamatának általános lépései**

1. Az adott lebontandó mező összes adatának kigyűjtése egy listába
2. A listából az ismétlések kiszűrése
3. A lista maradék elemeihez egyedi id-t rendelni, így megkapjuk az összes lehetséges adatot ami hozzá lehet rendelve a filmekhez
4. Az így kapott adatok feltöltése az adatbázis megfelelő táblájába
5. A filmek még le nem bontott tulajdonságainak összevetése a már kiszűrtekkel
6. Ha az összevetés során egyezést találunk, akkor a film id-jához hozzárendeljük annak az elemnek az id-ját, amelynél az egyezést találtuk, ezek lesznek a kapcsolótábla adatai
7. A kapott adatokkal a kapcsolótábla feltöltése

A lebontandó mezők a Keywords, Languages, Directors, Countries és Genres. A program az adatbázis movie tábláját használja bemeneti adatként, és az előbb említett táblákat, illetve az azokhoz kapcsolódó kapcsolótáblákat tölti fel. Mivel egy SQLite adatbázisból olvas, és abba is ír, ezért kapcsolatot kell teremteni a program és az adatbázis között, ez az SqlConnector nevű osztályban valósul meg.

**SqlConnector osztály**

Az SQL kapcsolat létrehozása mellett ebbe az osztályba kerültek az SQL kapcsolatot igénylő lekérdezéseket, táblamódosításokat tartalmazó metódusok is. Az osztály tartalmaz egy collectMoviedata nevű metódust, amely a filmek szükséges adatait kérdezi le és tárolja el a Program.cs-ben definiált Movie típusú movies listában, tehát itt kerülnek felhasználásra a Movies osztályban megadott változók.

A prepareKeywords, prepareLanguages, prepareDirectors, prepareCountries, és prepareGenres metódusok a film adott jellemzőit kigyűjtik egy listába, ahol minden egyes elem, ami az adatbázis movie táblájában vesszővel volt elválasztva, az ide új listaelemként kerül be. Ezek a metódusok szolgálnak a Keywords, Languages, Directors, Countries és Genres táblák alapjául. Az összes filmre vonatkozó adat egy listába kerül bele, hiszen a lényeg, hogy egy helyen legyen az összes lehetséges adat, ami majd később hozzárendelésre kerül a filmekhez. A listában ezen a ponton még sok ismétlődés lesz, viszont ezek a későbbiekben kiszűrésre kerülnek, és a lista minden eleméhez egyedi azonosítót, id-t rendelünk. A Program.cs osztályban létrehozott data listában kerülnek majd eltárolásra a metódusokban kigyűjtött adatok, a különböző prepare metódusok a Table osztályban kerülnek meghívásra. Erre a célra elég egyetlen lista is, hiszen egyszerre mindig csak egy táblát töltünk fel, így csak az ahhoz szükséges adatokat kell eltárolnunk.

Az osztály tartalmaz még egy removeWhitespace metódust is. Erre azért van szükség, mivel az előbbiekben említett metódusoknál a program a vesszőket figyeli mint határolópont, viszont az adatbázis eredeti mezőiben ahonnan dolgozik, ott vesszővel és szóközzel vannak határolva az adatok, így a data listába kigyűjtve a legtöbb listaelem egy szóközzel fog kezdődni, hiszen így került kigyűjtésre. Ez a metódus végigmegy az adott lista minden elemén amire meghívjuk és amennyiben az adott elem szóközzel kezdődik, akkor törli az első karakterét, tehát a szóközt a szöveg elejéről.

Végül az osztály egy executeInserts nevű metódussal zárul, ami bemeneti paraméterként egy listát kap meg, amelynek minden egyes eleme egy-egy táblafrissítő parancs, amelyet a Program.cs osztályban állítunk össze. A metódus végigmegy a lista elemein és sorra végrehajtja az SQL parancsokat. Hiba esetén kiírja, hogy „Hiba történt a tábla feltöltése során”.

**Program.cs osztály**

A Main metódusban felépítésre kerül az adatbázis eléréséhez szükséges út, az SqlConnector, FillUps, Algorithms, és Table osztályok példányosítása, illetve a program működéséhez szükséges változók:

* Egy int típusú id lista, amelyben a Keywords, Languages, Directors, Countries és Genres táblák feltöltéséhez használt id-k kerülnek majd tárolásra
* A korábban már említett string típusú data lista
* További két int típusú lista, a finalMovieID és a finalDataID, melyekre majd a kapcsolótáblák feltöltésére szolgáló parancsoknál lesz szükség,
* A string típusú cmds, melybe a végrehajtandó SQL utasítások kerülnek bele, ezeket fogja végrehajtani a korábban említett ExecuteInserts metódus

A definiált változók után meghívásra kerül a már bemutatott collectMoviedata metódus, tehát a movies listában eltároljuk a filmek lebontandó adatait mindenféle változás nélkül.

Ezután meghívásra kerül a Table osztály Options metódusa.

**Table.cs osztály**

A Table osztály az Algorithms, SqlConnector és FillUps osztályok metódusaival dolgozikj.

A Table osztály tartalmazza az Options metódust. A felhasználó itt tudja kiválasztani, hogy a Keywords, Movie\_Keywords, Languages, Movie\_Languages, Directors, Movies\_Directors, Countries, Movies\_Countries, Genres és Movies\_Genres táblák közül melyiket szeretné feltölteni. Az opciók 0-9-ig vannak megszámozva, a felhasználó a megfelelő szám beírásával tud választani közülük. A metódus további része egy switch case szerkezet. A Table.cs osztály minden feltölthető táblához tartalmaz egy a tábla nevével megegyező nevű metódust, amelyben az adott tábla feltöltéséhez szükséges metódushívások vannak elrendezve. A switch case szerkezettel a kiválasztott szám alapján ezek közül a metódusok közül hívunk meg egyet.

Ezeknek a metódusoknak két típusa van:

* Táblát tölt fel: genre, keyword, language, director, country
* Kapcsolótáblát tölt fel

Táblát tölt fel

Amelyek táblát töltenek fel, azok 6 darab metódushívást tartalmaznak. Az alábbiakban a keywords metódus tartalma kerül bemutatásra, de a genre, language, director, country metódusok is ugyanígy néznek ki, természetesen ahol szükséges ott az adott táblához igazítva:

1. Az adott táblához kapcsolódó, már korábban részletezett prepare metódus meghívása.

A keywords esetén a prepareKeywords. A metódushíváskor a data listát adjuk át neki referencia szerint, hogy a metódusban a listán végzett módosítások a metódus futásának befejeztével ne vesszenek el. A futása végén megkapjuk a data listában az összes kulcsszót, ömlesztve.

1. Az Algorithms osztály Matches metódusának meghívása.

A metódusnak átadjuk az előzőleg feltöltött data listát, illetve létrehozunk benne egy átmeneti string típusú listát (list\_ok). Végigmegyünk a data minden elemén, és ami még nem szerepel a list\_ok-ban, azt hozzáadjuk, ha pedig már szerepel benne az adott elem akkor megyünk a következőre. A metódus visszatérési értéke egy string típusú lista, így a list\_ok-ot fogjuk belőle visszaadni. A metódushívás helyén a data listát egyenlővé tesszük magával a metódushívással, tehát gyakorlatilag a list\_ok lista tartalmával.

1. Az Algorithms osztály idGenerate metódusának hívása.

A metódus arra szolgál, hogy az előző lépésben már ismétlődésmentessé szűrt data lista elemeit egyedi azonosítóval, kulccsal lássa el. Paraméterként megkapja a data listát, illetve referencia szerint a main függvényben már létrehozott int típusú id listát. A kulcs generálás egy for ciklus segítségével történik, 1-től indul és egyesével növekszik.

1. Az adott táblához kapcsolódó FillUp metódus meghívása a FillUps osztályból.

A keywords esetén FillUps.K. Ezzel a metódussal történik az adatbázis adott táblájának feltöltéséhez szükséges utasítások összeállítása, jelen esetben a Keywords feltöltéséhez szükséges SQL táblafeltöltő utasítások összeállítása. A tábla feltöltéséhez felhasználjuk az előzőleg összekészített data és id listát, ezeket megkapja a metódus paraméterként, továbbá referencia szerint átadjuk neki a korábban már említett string típusú cmds listát, mivel ebben kerülnek majd tárolásra a parancsok. Egy for ciklussal végigmegyünk az id és a data listán (elég egy darab for ciklus, hiszen mindkettőnek ugyanannyi eleme van), és az összetartozó indexű adatokat belerakjuk ugyanabba a parancsba.

1. A korábban már említett SqlConnector osztályban található ExecuteInserts metódus meghívása.

Az előző lépésben összekészített cmds listát átadjuk a metódusnak, és egy for ciklus segítségével mindegyik összekészített utasítás végrehajtásra kerül.

1. Végül a cmds listát a cmds.Clear() használatával kiürítjük.

Azért szükséges, mert ha egy futtatáson belül szeretnénk több táblát is feltölteni, akkor a korábban már végrehajtott utasítások a következő tábla feltöltésénél újra megpróbálnának végrehajtódni, viszont hibát kapnánk, hiszen ezek az adatok léteznek már a táblában.

Kapcsolótáblát tölt fel

Azok a metódusok, amelyek kapcsolótáblát töltenek fel, azok tartalmazzák azokat a metódushívásokat is, amelyek az előbbiekben, a sima tábláknál kerültek bemutatásra. Az alábbiakban a keyword\_movie metódus tartalma kerül részletezésre, de a genre\_movie, language\_movie, director\_movie, és country\_movie metódusok is ugyanígy néznek ki, természetesen ahol szükséges ott az adott táblához igazítva:

1. mode változó beállítása.

A mode változó segítségével fogjuk tudni később az Algorithms osztály createData metódusánál eldönteni, hogy a filmek melyik adatával dolgozunk, így ez az érték mindegyik kapcsolótábla esetén más.

1. prepareKeywords meghívása
2. Az Algorithms osztály Matches metódus hívása
3. Az Algorithms osztály idGenerate meghívása
4. Az Algorithms osztály createData függvény meghívása.

A metódus megkapja paraméterként az id, data, movies listát és a mode változót, továbbá referencia szerint átadjuk neki a finalMovieID és a finalDataID int típusú listákat. Ezekbe kerülnek majd bele azok az adatok, amelyekkel a kapcsolótáblát feltöltjük, tehát ezeket adjuk majd át az adott táblához tartozó FillUp metódushoz, mint az a sima tábláknál is történt.

A függvényben létrehozásra kerül egy string típusú temp lista, ezután az Algorithms osztály copy nevű metódusát hívjuk meg, átadva neki a movies listát, referencia szerint pedig a temp listát, illetve a mode változó értékét. A copy függvény egy switch case szerkezetet tartalmaz, a mode változó értéke alapján dönti el, hogy a temp listát a movies lista melyik adataival töltse fel. (Például a keyword\_movie kapcsolótábla feltöltésekor a program futásának ezen a pontján értelemszerűen a movies listából a keyword adatokra van szükség. A keyword\_movie függvény elején a mode változót 0-ra állítottuk be, ezért a copy metódus ha a mode értéke 0, akkor a temp listát a movies lista keyword adataival tölti fel, ami az adatbázisunk movie táblájának keyword mezőinek értékét tartalmazza, minden egyes filmhez a vesszővel elválasztott értékeket amiket eredetileg elkezdtünk lebontani).

A copy metódusból a createData-ba visszatérve végigmegyünk a temp listán. Két egymásba ágyazott for ciklus segítségével minden egyes elemét összehasonlítjuk a data lista összes elemével (a data lista a kigyűjtött, ismétlődésmentes kulcsszavakat tartalmazza, illetve az id lista az ezekhez rendelt egyedi kulcsot). Amennyiben egyezés van, akkor a finalMovieID listához hozzáadjuk az aktuálisan vizsgált film azonosítóját, (movies[index].Id), a finalDataID listához pedig annak a kulcsszónak az azonosítóját az id listából, amellyel az egyezés történt. Így létrejönnek a kapcsolótábla feltöltéséhez szükséges adatok.

1. FillUp\_MK meghívása
2. ExecuteInserts metódussal a parancsok végrehajtása
3. cmds.Clear() használatával a lista kiürítése