Automatizált adatbázis-frissítés

Szakdolgozat beszámoló

Írta: Tobik János – SEVOCP

Témavezető: Dr. Vassányi István

Pannon Egyetem

Villamosmérnöki- és Információs Rendszerek Tanszék

Tartalom

1. Bevezetés

Magyarországon nagyon sokan szenvednek olyan betegségektől, amelyek okozója az egészségtelen életmód. Ezen betegségek sokszor halálhoz is vezethetnek. Ilyen életmód-társult betegségek az elhízás, a metabolikus szindróma, a 2. típusú cukorbetegség, a stroke, a magas vérnyomás, a szívinfarktus, rosszindulatú daganatok, depresszió. A betegségek kialakulásának valószínűségét jelentősen lecsökkenthetjük, ha egészséges életvitelt folytatunk. Az egészséges életmód egyik alapja a tudatos táplálkozás. A tudatos táplálkozást segítik az olyan informatikai rendszerek, amelyekkel számon tarthatjuk a bevitt táplálékok tápanyagtartalmát, és saját testünk főbb adatait (pl. testtömeg, vérnyomás, vércukorszint).

A Pannon Egyetem Villamosmérnöki- és Informatikai Rendszerek tanszékén hosszú ideje folyik egy életmód tanácsadó szoftver fejlesztése, a Lavinia Életmód-tükör. 2013-2015 folyamán több vizsgálatot is végeztek az életmód-támogatás hatékonyságának ellenőrzésére. A Lavinia segíti a felhasználónak a mindennapi táplálékbevitel nyilvántartását. Gyors kereső felületeinek köszönhetően a táplálkozási naplózás időszükséglete akár napi 5 percre is lecsökkenthető, még a mobil technológiában nem jártas idősebb betegek esetén is. Az egyszerű kezelőfelületet az alábbi 1.1. ábra mutatja.

1.1. ábra Lavinia képernyőkép

Dolgozatom témája a Lavinia által használt adatbázis táplálékainak és azok tápanyagainak automatikus frissítése. Az adatbázis felhasználja az US Department of Agriculture National Nutrient Database for Standard Reference adatbázist. Az USDA egyik legfőbb tudományos kutatócsoportja az ARS (Agricultural Research Service), mely az Amerikai Egyesült

Államokban található meg. Munkájuk, hogy megoldást találjanak a mezőgazdasági problémákra, amelyek hatással vannak az amerikai emberek mindennapjaira. Közel hétszáz projektben több ezer embert foglalkoztatnak, ebből kétezer kutató. Az ARS vezeti a kutatást, hogy kifejlesszék és továbbítsák a megoldásokat a magas prioritású nemzeti problémákról. Információ hozzáférést biztosítanak a kiváló minőségű, veszélytelen ételek és más mezőgazdasági termékek vizsgálatainak eredményeiről. Az információkból felállított adatbázishoz bárki szabadon hozzáférhet.

A Lavinia adatbázisa jelenleg a 2007-ben közzétett SR20-as verziójú USDA adatbázist használja, amely jelentősen le van maradva a jelenlegi 2015-ös SR28-as verziótól. Feladatom, hogy az SR20-as verzió után kiadott frissítések alapján közzétett fájlok segítségével, szoftveresen frissítsem a Lavinia meglévő adatbázisát. Nagyon fontos szerepe van az adatbázis frissítésének, hiszen naprakésznek kell lennie az élelmiszerek terén. Az USDA egyre több és több élelmiszert vizsgál meg, és van, hogy akár több vizsgálatot is végeznek rajta a minél pontosabb eredmény érdekében. A pontos értékek elengedhetetlenek az egészséges életmódhoz.

A két adatbázis szerkezete jelentősen eltér egymástól. Ki kell olvasnom a külső adatbázisból a megfelelő adatokat, majd olyan formátumúra alakítani, hogy megfeleljen a Lavinia adatbázisának.

2. Specifikáció

A Lavinia jelenleg relációs adatbázist használ a receptek, az ételek és a tápanyagok tárolására, melyet a PostgreSQL relációsadatbázis-kezelő rendszerrel valósít meg. Az USDA adatbázis új verziói viszont szöveges fájlok formájában érhetőek el. Vannak fájlok amik a verziók közötti változásokat írják le, és vannak olyanok, amelyek az egész adatbázist tartalmazzák egy-egy verzióhoz. Tehát meg kell valósítani, hogy a szöveges fájlokból a megfelelő adatok alapján napra kész legyen a frissíteni kívánt adatbázis.

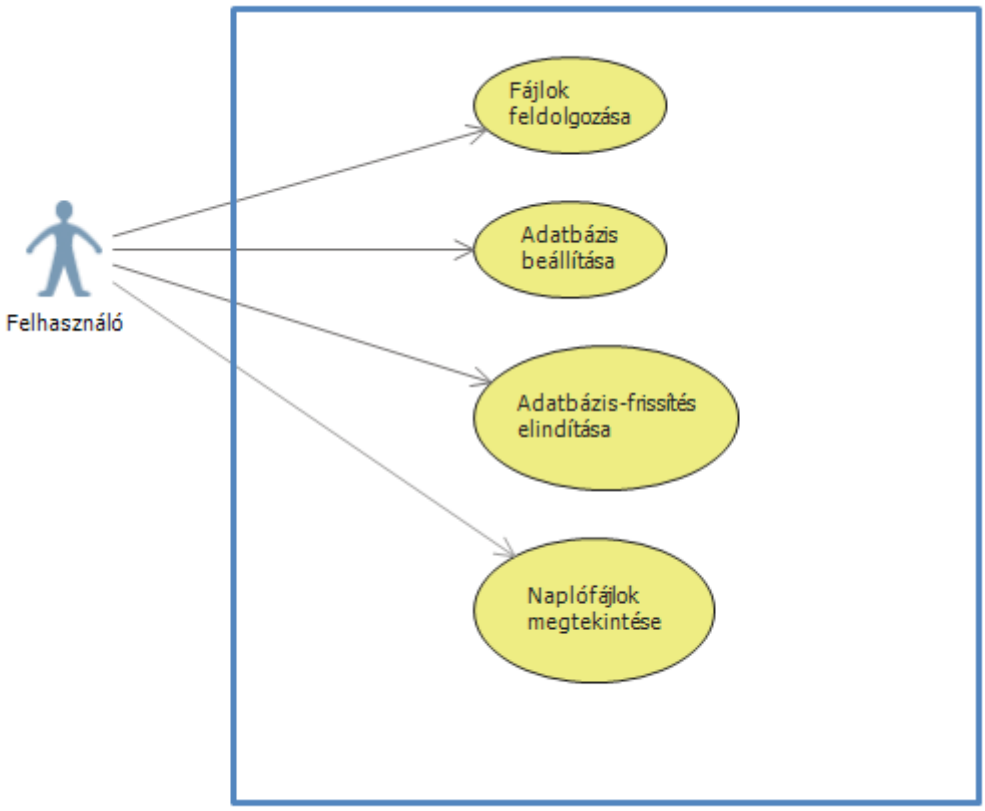
A felhasználóbarát működés érdekében egy könnyen kezelhető és átlátható grafikus megjelenítés szolgál arra, hogy tájékoztassa a felhasználót a műveletek eredményéről. A grafikus felület fő feladata, hogy közölni kell a felhasználóval a szöveges fájlok feldolgozásának eredményéről illetve az adatbázis frissítés eredményéről, mindezt jól látható és érthető megjelenítési formában.

Figyelembe kell venni, hogy egy ilyen adatbázis-frissítésnek függetlennek kell lennie a végrehajtás időpontjától és számától. Bármikor és bármennyiszer lefuttathatjuk. Célszerű azokat a fájlokat felhasználni, amelyek a verziók közötti változásokat definiálják, hiszen jelentősen megnövelné a folyamat időtartamát, ha kiürítenénk az adatbázist és az új verzió adataival töltenénk fel újra.

Az aktuális tranzakció visszagörgetése nagyon fontos része a rendszernek. Ha bármilyen hiba lép fel az adatbázis frissítése közben, a hiba előtt végrehajtott műveleteket vissza kell állítani a korábbi állapotra. Ez azt jelenti, hogy vagy az egész folyamat megvalósul, vagy egyik sem.

Fontos, hogy az adatbázis-frissítés folyamatát a rendszer valamilyen formában naplózza, jelentést készítsen szöveges formátumban úgy, hogy könnyen értelmezhető legyen a felhasználó számára. Sikeres és sikertelen műveletekről is egyaránt készülnie kell jelentésnek, hogy egy esetleges későbbi hibakeresés alkalmával könnyű dolgunk legyen annak kijavítására. Legegyszerűbb megoldás, ha egy közismert karakterkódolással (UTF-8) ellátott szöveges fájlokba mentjük a naplózást. A naplófájl felépítésének a lehető legegyszerűbbnek kell lennie a könnyű olvashatóság és értelmezhetőség miatt.

Az alábbi használati eset diagram (2.1. ábra) tartalmazza a főbb funkciókat, amelyeket a felhasználó el tud érni a rendszer felületén keresztül. A rendszer csak egyetlen, az ábrán látható aktort tartalmazza.



2.1. ábra Használati eset diagram

3. Rendszerterv

A szoftver implementálása előtt nagy hangsúlyt kell fektetni a tervezésre. Jól meg kell tervezni a szoftver komponenseket és az adatbázis oldali terveket. A rendszer több modult fog tartalmazni. A specifikációban említett funkciókat külön szegmensenként implementálom. A szoftver főbb funkcióit megvalósító alkotórészei a következők: adatbázis interfész, modell osztályok, grafikus felhasználói interfész, vezérlő osztályok, naplózás vezérlő. Java nyelven nem okoz gondot elkülöníteni a különböző feladatokat ellátó modulokat. Az azonos feladatért felelős osztályokat egy könyvtárba (Java package) lehet szervezni. Ily módon könnyebben átlátható az implementálás folyamata.

Az adatbázis interfész egy gyakran használt komponens a rendszerben. Újra és újra kapcsolatot kell kiépíteni a megadott adatbázissal való kommunikáció érdekében. Az adatbázis frissítését ez a modul végzi. A folyamatot az adatbázis-szerver oldali tárolt eljárásokkal hajtom végre, amelyeket a szoftver különböző paraméterekkel hív meg. Minden egyes beolvasott fájl adataival különböző műveleteket kell végrehajtani. Három típusú fájlt kell felhasználnom: *FOOD, NUTR, WGT*. Ezek a fájlok tartalmazzák az élelmiszereket, a tápanyagokat és az egységnyi súlyokat. Ezen fájlok tartalmazhatnak új adatokat, amelyeket hozzá kell adni az adatbázishoz (ADD_FOOD, ADD_NUTR, ADD_WGT), tartalmazhatnak olyanokat, amelyek adatait meg kell változtatni (CHG_FOOD, CHG_NUTR, CHG_WGT) és tartalmazhatnak olyanokat, amelyeket törölni kell (DEL_FOOD, DEL_NUTR, DEL_WGT). Így szám szerint kilenc tárolt eljárást kell implementálni az adatbázis megfelelő frissítéséhez. Ezen komponens megtervezése bizonyult a legnehezebbnek, hiszen a legfontosabb feladatot látja el a rendszerben.

Modell osztályok segítségével a beolvasott szöveges fájlok adatainak megfelelő leíró osztályokat valósítok meg az implementálás alatt. Ezekkel az osztályokkal könnyű eltárolni és átadni az adatokat a különböző vezérlő osztályoknak (pl. adatbázis interfész, naplózás vezérlő). Az egymáshoz tartozó adatokat reprezentálják ezen osztályok szerkezetei. Meglétük segíti a megfelelő adatok átadását az adatbázis tábláknak.

A grafikus felhasználó interfész teremti meg a kapcsolatot a szoftver folyamatai és a felhasználó között. Ezt a komponenst az egyik legmodernebb technológiával, a JavaFX-el valósítom meg. Könnyű kezelhetősége megkönnyíti az implementálás menetét. A JavaFX egy alapértelmezett stabil könyvtára a Java Standard Edition 8-as verziójának. Ezt a technológiát a dokumentum későbbi fejezetében ismertetem is bővebben.

A vezérlő osztályok fogják össze az egész rendszert. Összeköttetést létesítenek a modell osztályok és az interfészek között. Segítségükkel valósulhat meg a megfelelő adatáramlás a rendszeren belül. A naplózást is ilyen vezérlővel implementálom. Az adatbázison végrehajtott műveletek mindegyikéről készül jelentés. A modell osztályokat felhasználva az adatbázis-frissítés művelete megkapja a megfelelő paramétereket és a grafikus felület megjeleníti a hibás illetve sikeres lefutás adatait.

Fontos kérdés, hogy az adatbázis frissítése csak akkor legyen végrehajtva, ha az hibamentesen le tud futni, vagy tárolja el a frissítésre alkalmas adatokat miközben kihagyja a hibásokat. Ezt mind a szoftver oldalon és mind az adatbázis oldalon meg lehet valósítani tárolt eljárások formájában. Manapság szinte az összes adatbázis-kezelő rendszer támogatja a tárolt eljárások használatát. Ezt tekinthetjük egy függvénynek, amelyet az adatbázis szerver fordít le, tárol és hajt végre. Használatának előnye, hogy jelentős mértékben lecsökkenti az adatforgalmat az alkalmazás és az adatbázis szerver között, ezáltal gyorsabb lesz a végrehajtani kívánt művelet. Teljes mértékben független a hívó környezet programnyelvétől, egységes felületet biztosít. Így ha bárhonnan meghívjuk a tárolt eljárást, ugyanazt a funkcionalitást érjük el.

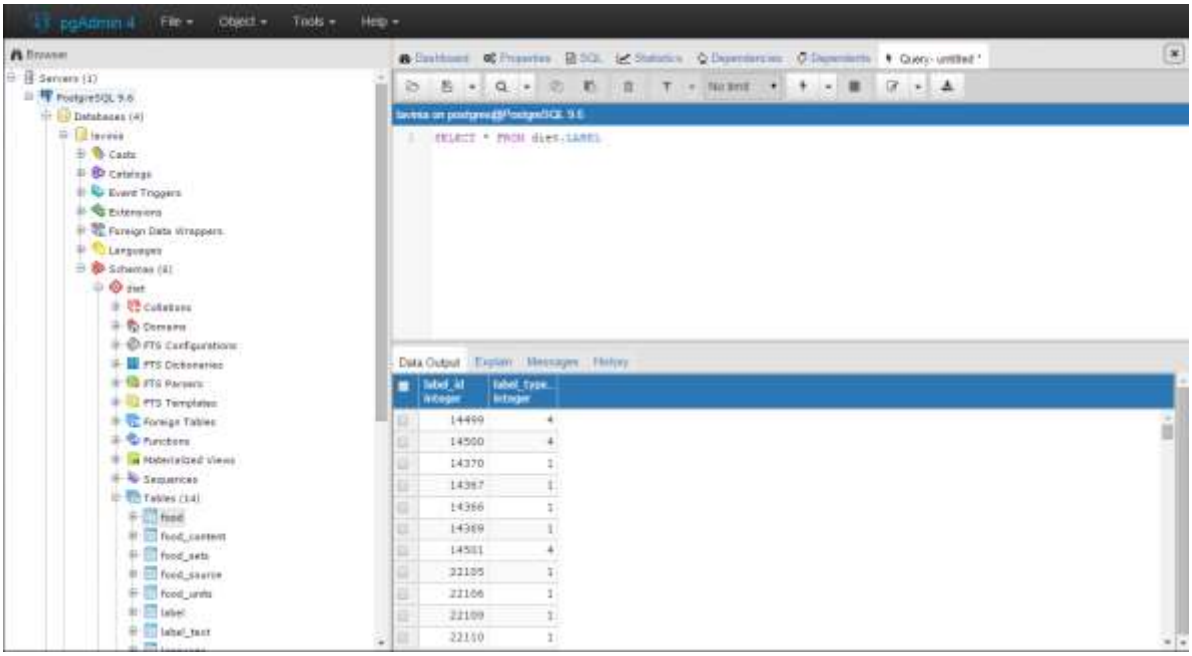
4. Technológiák

4.1 PostgreSQL

A PostgreSQL, más néven Postgres egy relációsadatbázis-kezelő rendszer, amelyet a Lavinia rendszere is használ. Szabad szoftver, melynek fejlesztését önkéntesek végzik közösségi alapon. A munka elsődleges koordináló oldala a postgresql.org. Kezdetben a Berkeley Egyetemen indult meg a fejlesztése a nyolcvanas években, majd a kilencvenes évek közepére elhagyta az egyetem falait és nyílt forráskódúvá vált.

A relációsadatbázis-kezelő rendszer (RDBMS) egy olyan adatbázis-kezelő rendszer, amelynek logikai adatbázisát szoftverkomponensei kizárólag a relációs adatmodellek elvén épülnek fel, illetve kérdezhetőek le. Kizárólag a relációs adatmodell alapú megközelítést támogatja. A relációsadatbázis-kezelő rendszerek szabványos adat hozzáférési nyelve az SQL (Structured Query Language). Az SQL segítségével könnyen és érthetően leírhatók akár az összetettebb CRUD (Create, Read, Update, Delete) funkciók is.

Az adatbázis adataihoz való hozzáférést, manipulációt, valamint az adatszerkezet tanulmányozásához a pgAdmin 4 programot használtam. Ez egy ingyenesen elérhető szoftver a PostgreSQL fejlesztőitől. E program melletti választásomat indokolta az, hogy biztosítja az egyszerű kezelőfelület és a szükséges funkciókat a feladatom során. A 4.1. ábra az említett program könnyen átlátható kezelőfelületet mutatja be.



4.1. ábra pgAdmin 4 felhasználói felülete

4.2 Java

A letölthető fájlok kezelésére, az adatbázis elérésére és manipulálására olyan szoftver kell, amely vezérli az adatátvitelt és közben erről tájékoztatást nyújt a felhasználó számára a folyamatról. Ezért döntöttem a Java nyelven történő implementálásról a feladat során.

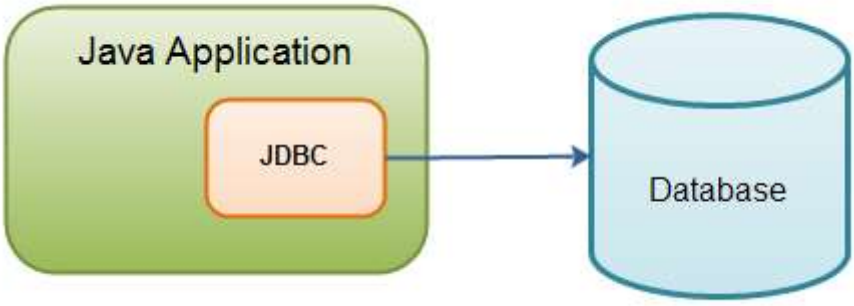
A Java egy általános célú, objektumorientált programozási nyelv, amelyet a Sun Microsystems fejlesztett a kilencvenes évek elejétől. Jelenleg az Oracle gondozásában áll. A Java alkalmazásokat jellemzően bájtkód formátumra alakítják. A bájtkód futtatását a Java virtuális gép (Java Virtual Machine) végzi, ami vagy interpretálja a bájtkódot, vagy natív gépi kódot készít belőle, és azt futtatja.

A program fejlesztéséhez a NetBeans IDE fejlesztőkörnyezetet választottam. A Java 8-as környezet szükséges a program futtatásához.

A feladat implementálásához a Java nyelvet választottam. A specifikáció során említett funkciókat könnyű vele megvalósítani. A fájlkezelést, az adatbázis kapcsolatot, a grafikus megjelenítést és az ezek közötti adatátvitelt egyaránt egyszerű felépíteni. Habár a Java Virtual Machine gyorsasága nem éri el a hardware közeli nyelvekét, ez nem jelent számottevő hátrányt a működésben.

4.2.1 Java Database Connectivity

A Java Database Connectivity, röviden JDBC egy API a Java programozási nyelvhez, amely az adatbázishozzáférést támogatja. A JDBC definiálja az adatbázisok lekérdezéséhez és módosításához szükséges osztályokat és metódusokat, miközben igazodik a relációs adatmodellhez. Működési elvét a 4.2. ábra mutatja be.



4.2. ábra JDBC működési elve

Az említett osztályok a *java.sql* csomagban találhatóak. Adatbáziskapcsolatot a csomag *Connection* osztály példányával lehet létrehozni. Egy ilyen objektumot a *DriverManager.getConnection()* metódus segítségével adhatunk meg. A metódus paramétereként megadhatjuk az adatbázis elérési útvonalát (IP cím, port szám), nevét, valamint a hozzá tartozó felhasználó nevet és jelszót.

```
Connection conn = DriverManager.getConnection(
"jdbc:postgresql://localhost:5432/testdb",
"username",
"password");
```

Az adatbázishoz való hozzáférés után SQL parancsot a *Statement* vagy *PreparedStatement* osztályok valamelyikével alkothatunk. A parancsot saját magunknak kell megadnunk szöveges formátumban úgy, hogy megfeleljen az SQL szintaktikának.

```
PreparedStatement pstmt = conn.preparedStatement("SELECT * FROM
MyTable");

pstmt.execute();
```

A végrehajtott művelet után fontos, hogy bezárjuk az adatbáziskapcsolatot. Fontos, hogy az operációs rendszer ne tartsa fent feleslegesen a kapcsolatot és ne foglalja a memóriát, ha nem szükséges. Ezt a *close()* metódussal tehetjük meg, amit a *Connection* objektumunkkal tudunk meghívni.

Ha bármilyen hiba adódna az adatbázis művelet végrehajtása során, úgy egy *SQLException* kivételt dob a program.

4.2.2 JavaFx

A JavaFX egy szoftver platform az asztali alkalmazások létrehozásához, amely a Swing mellett a Java Standard Edition alapértelmezett GUI könyvtára. A JavaFX applikációkat bármilyen asztali, mobil eszközön vagy böngészőben lehet futtatni. A grafikus felületet egy XML fájl definiálja, amelyet FXML fájlban tárolunk. A grafikus felület a Scene Builder

program segítségével egyszerűen összeállítható és nem kell aggódnunk az XML fájl összeállításában, mert automatikusan legenerálja azt.

4.2.3 Apache Commons IO

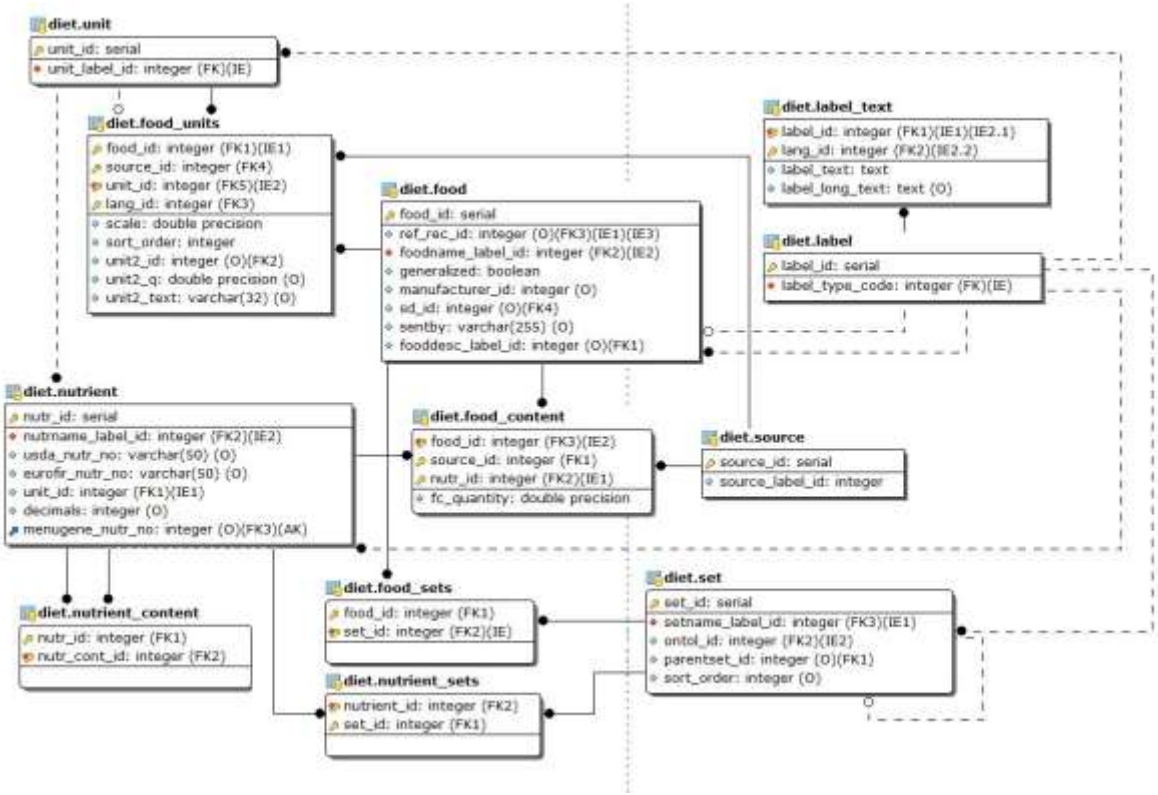
Az Apache Commons IO egy Java könyvtár fájlműveletekhez, melyet az Apache Foundation felügyelete alatt fejlesztettek. Osztályok sokaságát biztosítja a fejlesztők számára, hogy egyszerűbb, rövidebb és érthetőbb kód íródhasson a fájlok kezelésére. Habár a feladatom során kevés műveletet kell végre hajtanom fájlokkal, mégis fontos része a programnak, hiszen fájlok által beolvasott tartalommal kell frissíteni az adott adatbázist. A feladatom során használhattam volna az alapértelmezett Java osztályokat a fájlműveletez, de ez a könyvtár jelentős mértékben megkönnyíti a munkát implementálás közben.

4.3 Git

A Git egy nyílt forráskódú, elosztott verziókezelő szoftver. Feladata, hogy a projekt fájljainak különböző verzióit tárolja. Azért döntöttem a Git használata mellett, mert implementálás közben nagyon hasznos, ha egy fájl korábbi verziójához szeretnék hozzáférni és használni. Véleményem szerint kevés olyan rendszer van, amely felveheti a versenyt a Gittel mind a hatékonyságban és mind az egyszerű kezelésben.

5. Meglévő adatbázis

Feladatom során már egy meglévő adatbázissal kell dolgoznom. Ennek szerkezetén és felépítésén nem változtathatok, egyedül adatmanipulációt hajthatok végre. Ebben a fejezetben mutatom be az adatbázis felépítését és főbb pontjait, amelyet feladatom során kell felhasználnom. Az alábbi képen látható az adatbázis táblái és az azok közötti kapcsolatok.



5.1. ábra Az adatbázis ER modelje

Az adatbázis főbb törzsállományai a *food* és a *nutrient* tábla mely az élelmiszereket illetve a tápanyagokat reprezentálja. Ezt a két táblát a *food_content* köti össze közvetlenül. Itt kerül tárolásra, hogy melyik élelmiszer melyik tápanyagokat tartalmazza milyen mennyiségben. Ez a mennyiség 100g élelmiszerben lévő tápanyag mennyiségét mutatja meg. Egy egységnyi étel súlyát definiálja dekagrammban a *food_units* tábla. Feladatom során ezek a fontosabb táblák, amelyekben adat bevitelt, módosítást illetve törlést kell alkalmaznom. A sok kapcsolat miatt ezeket a műveleteket jól meg kell tervezni.