Debreceni Egyetem Informatikai Kar

Információs rendszerek tervezése, fejlesztése

Órarendszerkesztő program

Szakdolgozat

Témavezető:
Dr. L. Nagy Éva
Számítástechnikai munkatárs

Készítette:
Csipkés László
programozó matematikus

Debrecen, 2007.

Tartalomjegyzék

1. Bevezetés

A mai világban a számítógépek egyre inkább átveszik a humán munkaerő szerepét az egyes munkaterületeken. Egyre intelligensebb és precízebb programok jelennek meg a szakágakban, melyek használata nagyban megkönnyíti és felgyorsítja az egyes munkameneteket.
Szakdolgozatom témája az oktatás egy kényes pontját, az órarendek szerkesztését hivatott segíteni. Szakdolgozatomban egy konkrét információs rendszer megtervezését és fejlesztését mutatom be. Programom szerkezete a kazincbarcikai Kazinczy Ferenc Kertvárosi Általános Iskolában jelenleg is használt órarendi tematikára épül. Az iskola fennállása óta tradicionálisan két vagy három oktató feladata volt félévről félévre a felsőbb évfolyamos osztályok órarendjének összeállítása. Ez a folyamat manapság a sok éves rutin ellenére is 3-4 napot vesz igénybe. Célom programommal az ő munkájuk leegyszerűsítése és segítése egy minden igényt kielégítő, ütközésmentes órarendszerkesztő program megalkotásával. Iskolaválasztásom elsődleges szempontja az a tény volt, hogy általános iskolás tanulmányaim 8 évét ezen intézmény falai között töltöttem. Az itt tanító oktatók jelentős részét a mai napig ismerem, és jó viszonyt ápolok velük.
További célom egy könnyen átlátható és kezelhető felületet nyújtó, mégis minden követelménynek eleget tevő program megalkotása volt.

Adatbázis-kezelő rendszerként egy teljes mértékben Javahoz tervezett és Javaban írt relációs adatbázis-kezelőt használtam, melynek neve HSQLDB. A HSQLDB rendelkezik egy JDBC (Java DataBase Connectivity) vezérlővel, mely segítségével Java nyelven végezhetünk műveleteket az adatbázis egyedein.
A forráskód Java nyelven került implementálásra a Sun cég J2SE 5.0 fejlesztőkörnyezetével. A Java kód és az adatbázis-kezelő közti kapcsolatot a Hibernate Core 3.2.2 GA Java kiegészítő program valósította meg, mely futása során a háttérben JDBC segítségével kezeli a HSQLDB adatbázist.

2. Információs rendszerek

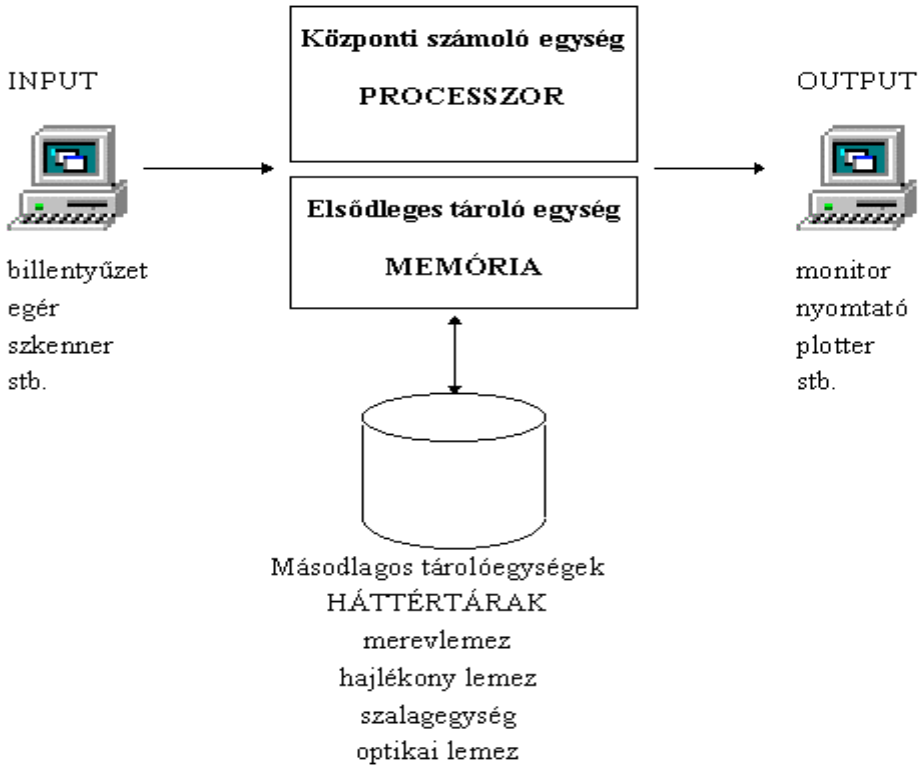
2.1. A rendszer

A rendszerek elemekből tevődnek össze, melyek együttesen egy közös cél érdekében működnek.

A rendszerekre jellemző paraméterek:

- bemenet (input)
- kimenet (output)

A rendszer a működése során bemenetet fogad, majd egy átalakítási folyamat során kimenetet hoz létre.



2.2. Az információs rendszer

Egy információs rendszer információknak, az ezen információkhoz kapcsolódó információs eseményeknek, a rajtuk végrehajtott információs tevékenységeknek, a kapcsolatos erőforrásoknak, az információk felhasználóinak és a mindezeket szabályozó eljárásoknak együttese.

Az információs rendszerek tervezésének és létrehozásának szakaszai:

- A felhasználói igények, valamint a rendelkezésre álló erőforrások felmérése
- Az igényeket és a lehetőségeket egyeztetni kell
- A feldolgozáshoz szükséges adatok összegyűjtése
- Az adatok legfontosabb jellemzőinek a meghatározása

- Az adatok közötti kapcsolatok meghatározása
- Logikai adatbázis megtervezése
- Fizikai adatbázis megtervezése
- Az adatokat feldolgozó folyamatok megtervezése
- A kritériumok meghatározása
- Az adatokat feldolgozó folyamat implementálása
- A kész rendszer próbája, tesztelése, javítása

Az igények és a lehetőségek egyeztetése után logikai adatbázis tervek jönnek létre, melyek alapján fizikai adatbázisok kerülnek implementálásra. Az információs rendszer feldolgozó folyamatai a fizikai adatbázis egyedein hajtja végre a műveleteit, majd az eredményt a rendszer outputjára továbbítja.

3. Az órarend rendszere

3.1. Az órarend eddigi készítéséről

A kazincbarcikai Kazinczy Ferenc Kertvárosi Általános Iskolában jelenleg az órarendkészítés két külön részre van bontva. Ahogy az általános iskolákban vannak alsós- és felsős osztályok, úgy az órarendek is külön készülnek az alsósoknak és a felsősöknek. Szakdolgozatom tárgyát képező program a komplexebb felsős évfolyamok órarendjét fogja elkészíteni, melyek az ötödik, hatodik, hetedik, és nyolcadik évfolyamokat tartalmazzák. Az órarendet félévente kell generálni, mivel az iskolában tavaszi és őszi órarend is létezik.
Az eddigi munkamenet a következő volt: a tanárok összeültek és manuális módszerrel, „pötyi-módszerrel”, egy erre a célra kialakított fatáblára tették ki az osztályok órarendjeit. Minden tanár

saját színű „pötyi”-vel lett ellátva, melyek egyértelműen jelzik, hogy mikor melyik osztállyal van órája az adott oktatónak.
Ha egy kezdeti verzió elkészült és néhány órának még nem volt helye a táblában, akkor azokat addig cserélgették a már bent lévő elemekkel, amíg végül egy komplex, minden elemet tartalmazó rendszert nem kaptak. Természetesen a tantárgyaknak megvannak a maguk prioritásai, amik alapján eldönthető, mely tantárgyakat érdemes átmozgatni úgy, hogy a lehető legkevesebb változásokat eszközöljük.

3.2. A rendszer feladata

Minden félév előtt az iskola vezetősége elkészít egy, az aktuális félévre érvényes tantárgyfelosztást, mely tartalmazza, hogy az adott félévben egy adott osztálynak melyik tantárgyat melyik oktató fogja tanítani, heti hány órában. Ez a felosztás félévről félévre változik. A rendszer bemenetként fogja megkapni az aktuális félév tantárgyfelosztását, melyet alapul véve generál egy órarendet, amit kimenetként fog visszaadni.
A rendszernek első fázisként el fog kelleni készítenie a tanárok, a termek és az osztályok kapcsolatait, majd az így kapott sémához hozzá kell rendelnie az osztálytermeket. Természetesen a hozzárendelések során figyelnie kell az esetleges ütközéseket, illetve, hogy a megadott input paraméterekkel generálható-e a kritériumoknak eleget tevő output.

4. Az órarendet generáló rendszer elkészítésének lépései

4.1. Követelmények

4.1.1. A feladat tömör összefoglalása

Egy általános iskola felsős tagozatos osztályai számára órarend generálása.
Az órarend a fejlesztendő program input paramétereiként megadott elemekből kell, hogy generálódjon. Minden újrafuttatásnál a programnak a feleslegessé vált adatokat ürítenie fog kelleni.

Ezek az input elemek a következők lesznek:

- Oktatók
- Termek

- Osztályok
- Tantárgyfelosztás

A kapott elemeket egy közös táblában fogjuk tárolni, melyből a futás végén egyértelműen kiolvasható lesz az aktuálisan generált órarend.

4.1.2. Adatok változtatása

Mai világunkban minden rendszer értéke a sokszínűségében rejlik. A lehető legnagyobb szabadságot kell nyújtani a felhasználónak a programok működésének keretein belül.
A felhasználónak lehetőséget kell nyújtani adatok felvételére, módosítására, törlésére, hogy a készítendő program elősegítse, és megkönnyítse a felhasználó munkáját.

4.1.2.1. Adatok rögzítése

A fejlesztendő program egyik legfontosabb feladata a külvilággal történő kommunikáció. Amennyiben erre nem lenne lehetőség, képtelenség lenne olyan újrahasznosítható programot írni, mellyel kielégíthetnénk a felhasználók mindig változó igényeit.
Készítendő programomban az adatok rögzítése grafikus felületek segítségével fog történni.
A felhasználó kitölti a kitöltendő mezőket, és a program a mezőkben megadott értékek alapján fog létrehozni egy új bejegyzést a hivatkozott táblában. Amennyiben a felhasználó valamilyen hibát követne el adatbevitel közben, például egy már létező adatot szeretne felvenni, vagy nem a megfelelő formában töltene ki egy mezőt, akkor a program felugró ablakkal fogja figyelmezteti a felhasználót az adott hibára.
Programom előreláthatóan 6 felvevő ablakot fog tartalmazni, melyek segítségével a felhasználó képes lehet az aktuális táblák manipulálására.
Felvehető adatok:

- Oktatók
- Termek
- Osztályok
- Tantárgyfelosztások
- Összevont tantárgyak tantárgyfelosztásai
- Dupla tantárgyak tantárgyfelosztásai

Mivel bizonyos oktatóknak lehetnek olyan hétköznapjai, mikor az aktuális félév adott napjain nem tud tanítást vállalni, a felvételnél biztosítani kell ezen lehetőség felvételét is.

4.1.2.2. Adatok törlése

Másik nagyon fontos tényező, az adatok törlési lehetőségei.
Mivel a felhasználó oldaláról nézve az adatbázissal csak a grafikus felületen keresztül tud kommunikálni fontos, hogy a már nem aktuális elemeket törölni tudja. Ha valamilyen oknál fogva csökkenteni kéne majd bármelyik tábla sorainak a számát, akkor ezt a megfelelő menüben kiválasztott ablak segítségével megteheti az illető. Előreláthatólag 4 tábla fog rendelkezésére állni, melyek adatok törlésére hivatottak:

- Oktató törlése
- Terem törlése
- Osztály törlése
- Tantárgyfelosztás törlése, itt fognak szerepelni a speciális tantárgyak is

Törlés esetén természetesen a táblára hivatkozó egyéb táblákban is törlődni fognak a megfelelő sorok.

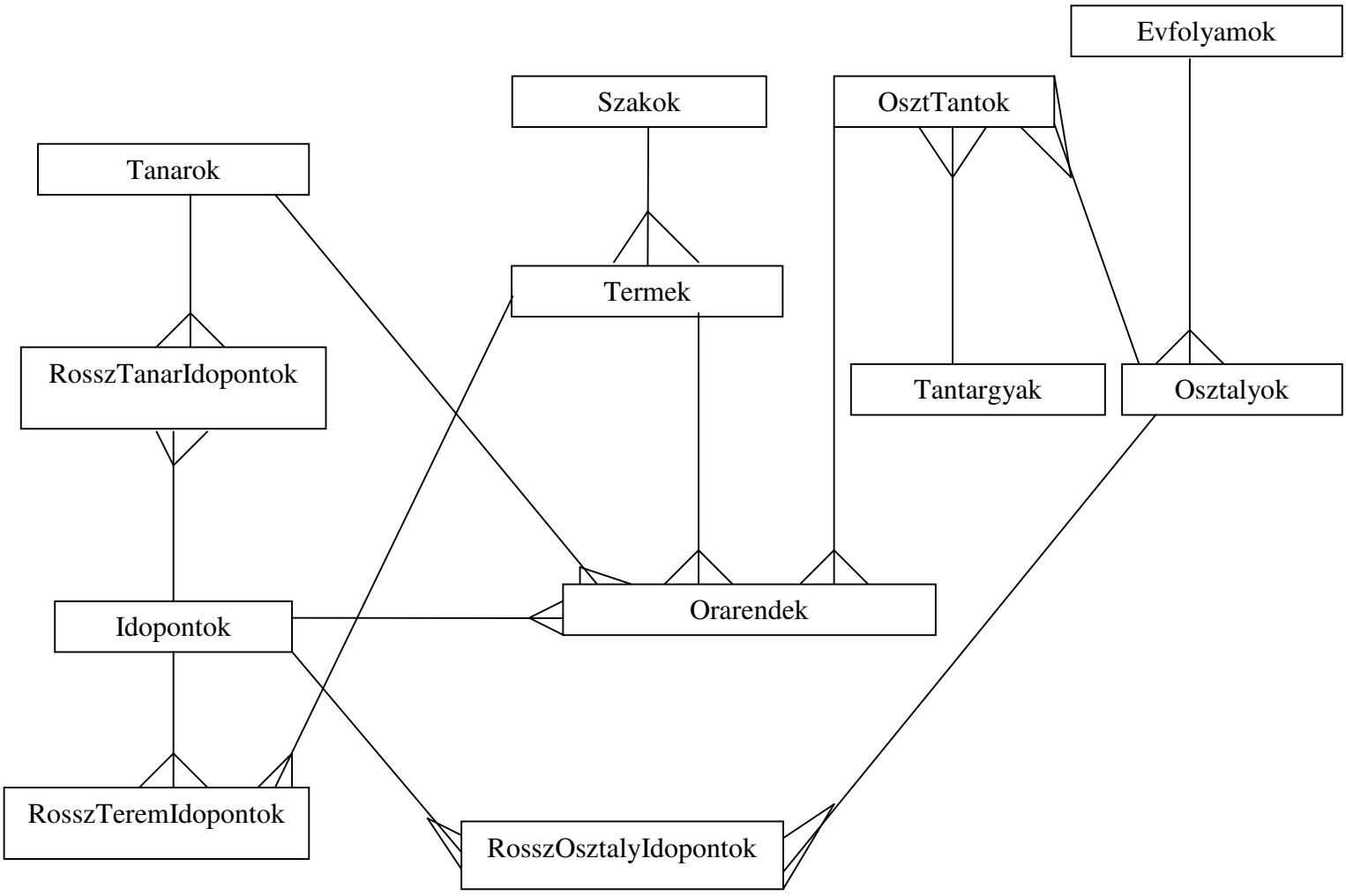
4.1.2.3. Adatok módosítása

Amennyiben a felhasználó igényei csak minimálisan változtatnák meg az adatbázis bejegyzéseit, úgy lehetőség fog nyílni csak a módosítandó adat változtatására, így megelőzvén, hogy a felhasználónak előbb törölnie, majd újra hozzáadnia kelljen az adatbázishoz.
A készítendő program futtatása során előreláthatóan 1 helyen lesz lehetőség a módosításra, a tantárgyfelosztásoknál.
Itt meg lehet majd adni, hogy ha egyik évről a másikra egy bizonyos tantárgyat ugyanaz az oktató tartana ugyanannak az osztálynak, de más heti óraszámban, akkor ne kelljen az aktuális tantárgyfelosztást előbb törölni, majd újra bejegyezni az új értékkel, hanem a heti óraszámát lehessen megváltoztatni.

4.2. Adatterv

Az adatterv a kulcsfeladatok átalakítása egy olyan formába, mely a stratégiai terv alapját képezi. Az adattervezés egy folyamat, amikor a kialakított logikai terv alapján létrehozzuk a fizikai tervet.

4.2.1. Logikai adatterv



A részvétel minden kapcsolat esetében a kapcsolat mindkét oldaláról nézve kötelező.

4.2.2. Fizikai adatterv

A relációs modell alapján az adatbázisban 12 táblát helyezünk el, melyek minimálisan szükségesek egy ütközésmentes órarend létrehozásához. A táblákat HSQLDB segítségével tároljuk, a tábla eleminek elérését a Hibernate Core 3.2.2 GA program segíti.

4.2.2.1. Táblák

Szakok	**Típus**	**Oszlopmegszorítás**
Szak_kod	String	Primary Key
Szak_nev	String	Not Null

Evfolyamok	**Típus**	**Oszlopmegszorítás**
Evfolyam	String	Primary Key CHECK ('5'<=Evfolyam<='8')

Tantargyak	**Típus**	**Oszlopmegszorítás**
Tantargy_kod	String	Primary Key
Tantargy_nev	String	Not Null

Osztalyok	**Típus**	**Oszlopmegszorítás**
Nev	String	Primary Key CHECK ('a'<=Nev<='z')
Evfolyam	String	Foreign Key Evfolyamok (Evfolyam)
Teremnev	String	Not Null References Termek (Teremnev)
Teremszak	String	Not Null References Termek (Szak_kod)
Okt_kod	Long	Not Null References Tanarok (Okt_kod)

Tanarok	**Típus**	**Oszlopmegszorítás**
Okt_kod	Long	Primary Key
Nev	String	Not Null
Szuletesi_ev	Integer	Not Null
Osztalya_neve	String	References Osztalyok (Nev)
Osztalya_evfolyama	String	References Osztalyok (Evfolyam)

Termek	**Típus**	**Oszlopmegszorítás**
Teremnev	String	Primary Key
Szak_kod	String	Foreign Key Szakok (Szak_kod)
Osztalyaneve	String	References Osztalyok (Nev)
Osztalyaevfolyama	String	References Osztalyok (Evfolyam)

OsztTantok	**Típus**	**Oszlopmegszorítás**
Tantargy_kod	String	Foreign Key Tantargyak (Tantargy_kod)
Osztalynev	String	Foreign Key Osztalyok (Nev)
Osztalyevfolyam	String	Foreign Key Osztalyok (Evfolyam)

Heti_oraszam	Integer	Not Null CHECK (1<=Evfolyam<=5)

Idopontok	**Típus**	**Oszlopmegszorítás**
Napnev	String	Primary Key
Hanyadikora	Integer	Primary Key

RosszTanarIdopontok	**Típus**	**Oszlopmegszorítás**
Okt_kod	Long	Foreing Key Tanarko(Okt_kod)
Napnev	String	Foreing Key Idopontok(Napnev)
Hanyadikora	Integer	Foreing Key Idopontok(Hanyadikora)

RosszTeremIdopontok	**Típus**	**Oszlopmegszorítás**
Terem_nev	String	Foreign Key Termek (Teremnev)
Szak_kod	String	Foreing Key Termek(Szak_kod)
Napnev	String	Foreing Key Idopontok(Napnev)
Hanyadikora	Integer	Foreing Key Idopontok(Hanyadikora)

RosszOsztalyIdopontok	**Típus**	**Oszlopmegszorítás**
Nev	String	Foreign Key Osztalyok(Nev)
Evfolyam	String	Foreing Key Osztalyok(Evfolyam)
Napnev	String	Foreing Key Idopontok(Napnev)

Hanyadikora	Integer	Foreing Key Idopontok(Hanyadikora)

Orarendek	**Típus**	**Oszlopmegszorítás**
Or_sorszam	Long	Primary Key
Tantargy_kod	String	References OsztTantok (Tantargy_kod)
Okt_kod	Long	References Tanarok (Okt_kod)
Teremnev	String	References Termek (Teremnev)
Szak_kod	String	References Termek (Szak_kod)
Nev	String	References OsztTantok(Osztalynev)
Evfolyam	String	References OsztTantok(Osztalyevfolyam)
Napnev	String	References Idopontok(Napnev)
Hanyadikora	Integer	References Idopontok(Hanyadikora)

4.2.2.2. Táblák jellemzése

-Szakok: Ebben a táblában tároljuk a szakokat. A szakok határozzák meg, hogy egy teremben milyen tantárgyakat lehet tartani. Jelen pillanatban az iskola rendszere alapján 3 fajta szak létezik, ezek közül egy teremhez csak egyféle rendelhető.
Szakfajták:

-TES (testnevelés) Testnevelés terem, az ilyen típusú termekben csak testnevelés órák tarthatóak.

-SZÁ (számítástechnika) Számítástechnika terem, az ilyen típusú teremben csak számítástechnika órák tarthatóak.

-ÁLT (általános) Általános terem, az ilyen teremben tartható bármilyen óra, ami nem testnevelés és számítástechnika. Oka: Az iskolában példának okáért se a kémia, se a fizika nincs teremhez kötve, ha esetleg valamilyen eszköz segítségére lenne mégis szükség az adott órán, azt a kémia-, fizikaszertárból könnyedén átvihetik az adott terembe.

-Evfolyamok: Ebben a táblában tároljuk az évfolyamokat. Mivel általános iskolának készül az órarend, és ezen belül is a felsős osztályoknak, a mai magyarországi tanulmányi rendszer alapján 4 évfolyamfajta jöhet szóba, ezek pedig a következők:

-5. Ötödikes évfolyam
-6. Hatodikos évfolyam
-7. Hetedikes évfolyam
-8. Nyolcadikos évfolyam

-Tantargyak: Ebben a táblában tároljuk a tantárgyakat. A tábla sorai az iskolában jelenleg érvényben lévő tantárgyakat tartalmazzák. Mivel a közelmúltban megjelent néhány új tantárgy, melyek még nincsenek egyértelműen eldöntve, hogy mely évfolyamoknak lesznek taníthatóak, a tantárgyakat nem kötjük évfolyamokhoz.
A Tantargyak táblában szereplő tantárgyak: matematika, biológia, történelem, irodalom, nyelvtan, kémi, fizika, biológia, testnevelés, … .

-Osztalyok: Ebben a táblában tároljuk az osztályokat. Egy osztályt egyértelműen azonosít a neve, ami egy betű, valamint az aktuális évhez tartozó évfolyama együttesen.
Az osztályok létszámának eltárolása felesleges, ugyanis minden tanévben csak olyan létszámú osztályok indulnak, melyek bármelyik teremben elférnek.

-Tanarok: Ebben a táblában tároljuk a tanárokat. Mivel megeshet, hogy két oktató azonos névvel rendelkezik és azonos évben született, az oktatókat hozzájuk rendelt oktatói kódok alapján különböztetjük meg. Az oktatókhoz tartozhat egy osztály is, melynek az adott oktató az osztályfőnöke. Az oktatóhoz nem szükséges továbbá szakot rendelni, ugyanis gyakorta előforduló eset, hogy egy oktató egy, nem a szakjainak megfelelő órát tart.
Amennyiben az illetékes tanár osztálya 8. Évfolyamos és évléptetést hajtunk végre, akkor az oktató adatainál a név mező értéke null érték lesz, ezzel jelezve, hogy az oktatónak ebben az évben még nincs osztálya.

-Termek: Ebben a táblában tároljuk a termeket. Egy termet egyértelműen azonosít a neve és a hozzá rendelt szak kód együttesen. Minden teremhez csak egy szak kód rendelhető. A termeknél szintén felesleges a befogadási létszám értékének szerepeltetése, ugyanis a már korábban is

megemlített tények alapján csak olyan létszámú osztályok indulnak az intézményben, melyek bármelyik terembe befférnek.

-OsztTantok: Ez a tábla egy közvetítő tábla az Osztályok és a Tantárgyak között. Benne a két tábla elsődleges kulcsai, mint külső kulcsok alkotják a tábla összetett kulcsát, valamint egy heti_oraszam mező, mely azt mondja meg, hogy az adott tantárgy egy héten hányszot tartható az adott osztálynak. A táblára azért van szükség, mert az Osztalyok és a Tantargyak táblák között N:M kapcsolat áll fenn, és mivel a legtöbb adatbázis kezelő program nem tudja ezt a kapcsolatfajtát kezelni, szükséges egy közvetítő tábla felvétele, melyhez mindkét tábla 1:N kapcsolattípussal csatlakozik.

-Idopontok: Ebben a táblában tároljuk az időpontokat. Egy időpontnak két fontos tulajdonsága van. Az egyik az adott nap neve, míg a másik egy óraszám. Ezen két attribútum együttesen alkotja a tábla elsődleges kulcsát. Mivel az iskolákban tanítási napok a munkanapokkal esnek egybe, a napok által felvehető értékek: hétfő, kedd, szerda, csütörtök, péntek. Egy tanítási napon egy osztálynak maximálisan 7 órája lehet. Ezek alapján minden naphoz hozzá kell rendelni mind a 7 órát, melyeknek értékei: 1, 2, 3, 4, 5, 6, 7. Az Idopontok tábla már a készítendő program első futtatása előtt fel van töltve, hozzáférés és módosítása nem megengedett.

-RosszOsztalyIdopontok: Ez a tábla egy közvetítő tábla az Idopontok és az Osztalyok táblák között. Mindkét tábla 1:N kapcsolattal csatlakozik ehhez a táblához. Benne a két tábla elsődleges kulcsai, mint külső kulcsok alkotják a tábla összetett kulcsát.
A készítendő program futása közben ha egy osztály az órarendben lehelyezésre került valahova, akkor a lehelyezés helyének időpontja és az osztály együttesen bekerülnek ebbe a táblába, amiből a későbbiekben kiolvasható, hogy az adott osztálynak mikorra van már órája.

-RosszTeremIdopontok: Ez a tábla egy közvetítő tábla az Idopontok és az Termek táblák között. Mindkét tábla 1:N kapcsolattal csatlakozik ehhez a táblához. Benne a két tábla elsődleges kulcsai, mint külső kulcsok alkotják a tábla összetett kulcsát.
A fejlesztendő program futása közben ha egy teremhez az órarendben időpont rendelődik, akkor a megadott időpont és a terem együttesen bekerülnek ebbe a táblába, amiből a későbbiekben kiolvasható, hogy az adott terem mikor foglalt.

-RosszTanarIdopontok: Ez a tábla egy közvetítő tábla az Idopontok és az Tanarok táblák között. Mindkét tábla 1:N kapcsolattal csatlakozik ehhez a táblához. Benne a két tábla elsődleges kulcsai, mint külső kulcsok alkotják a tábla összetett kulcsát.
Ha egy oktatónak minden héten egy adott napra más elfoglaltsága van és azon a napon nem tud tanítást vállalni, akkor az adott nap minden órája és az oktató együttesen, már az oktató rögzítésekor bekerül ebbe a táblába.
A kifejlesztendő program futása közben ha egy tanárhoz órát rendelünk az órarendben, akkor az óra időpontja és az oktató együttesen bekerülnek ebbe a táblába, amiből a későbbiekben kiolvasható, hogy az adott tanárnak mikor van már órája.

-Orarendek: Ebben a táblában tároljuk az órarendeket. Egy sort az or_sorszam ID azonosít. Ez a tábla 1:N kapcsolattal csatlakozik a következő táblákhoz: OsztTantok, Termek, Tanarok, Idopontok. Benne az összes tábla elsődleges kulcsai helyezkednek el.
Ha elkészül a programban az adott félévre vonatkozó tantárgyfelosztás bevitele, és a felhasználó elindítja az órarend generálását, akkor az első lépésként az OsztTantok tábla alapján feltöltődnek a tábla megfelelő attribútmai, míg a többi mezőbe Null érték kerül. Az OsztTantok táblából egy sor annyiszor jelenik meg az Orarendek táblában, ahány alkalommal kell tartani az adott tantárgyat az adott osztálynak. A futtatás során kicserélődnek a Null értékek tényleges egyedelőfordulásokra. A program akkor tekinthető sikeres végkimenetelünek, ha az Orarendek tábla egyik sora sem tartalmaz Null értéket.

4.2.2.3. Táblák közti kapcsolatok

A táblák között egységesen 1:N kapcsolatok vannak realizálva, mivel minden egymással kapcsolatban lévő tábla szülő-gyermek kapcsolatban van.
Ezen kapcsolattípusnál a kapcsolat bal oldalán elhelyezkedő táblának sorai a jobb oldali táblának több sorával is kapcsolatban lehetnek, míg a jobb oldali tábla minden sorához pontosan egy sor tartozhat a baloldali táblából.

Evfolyamok --- 1:N --- Osztalyok (Osztalyok references Evfolyamok(Evfolyam))

Osztalyok --- 1:N --- OsztTantok (OsztTantok references Osztalyokok(Nev, Evfolyam))

Tantargyak --- 1:N --- OsztTantok (OsztTantok references Tantargyak(Tantargy_kod))

Szakok --- 1:N --- Termek (Termek references Szakok(Szak_kod))

Idopontok --- 1:N --- RosszTanarIdopontok
(RosszTanarIdopontok references Idopontok(Napnev, Hanyadikora))

Tanarok --- 1:N --- RosszTanarIdopontok
(RosszTanarIdopontok references Tanarok(Okt_kod))

Idopontok --- 1:N --- RosszTeremIdopontok
(RosszTeremIdopontok references Idopontok(Napnev, Hanyadikora))

Termek --- 1:N --- RosszTeremIdopontok
(RosszTeremIdopontok references Termek(Teremnev, Szak_kod))

Idopontok --- 1:N --- RosszOsztalyIdopontok
(RosszOsztalyIdopontok references Idopontok(Napnev, Hanyadikora))

Osztalyok --- 1:N --- RosszOsztalyIdopontok
(RosszOsztalyIdopontok references Osztalyok(Nev, Evfolyam))

OsztTantok --- 1:N --- Orarendek
(Orarendek references OsztTantok(Tantargy_kod, Osztalynev, Osztalyevfolyam))

Termek --- 1:N --- Orarendek (Orarendek references Termek(Teremnev, Szak_kod))

Tanarok --- 1:N --- Orarendek (Orarendek references Tanarok(Okt_kod))

Idopontok --- 1:N --- Orarendek (Orarendek references Idopontok(Napnev, Hanyadikora))

4.3. Végrehajtási stratégia

Mivel a programnak sok mindent kell végrehajtania egy folyamaton belül, érdemes ezeket a folyamatokat részekre bontani, és őket egymás után végrehajtani úgy, hogy a korábban már végrehajtott folyamatok eredményei is befolyásolják a soron következő folyamat végkimenetelét.

A program feladata a tantárgyak, osztályok és termek összehozása egy egésszé úgy, hogy ugyanabban az időpontban egy oktató egy osztálynak egy tantárgyat csak egy teremben tartson. Ezen hármas összehozását fel lehet bontani 2 nagyobb folyamat egymásutánjára. Az első lépésben az osztályokhoz hozzárendeljük az oktatókat és a tantárgyakat, majd az így kapott eredményhez rendeljük hozzá a tantermeket. Azzal, hogy egy folyamatból kettőt csinálunk nagyban meggyorsítjuk programunk futási idejét, ugyanis ha például az osztály oldaláról tekintjük a program végrehajtását, akkor nem kell egyszerre két feltételnek teljesülnie (az adott időpontban a tantárgyat tanító oktatónak is szabad idejének kell lennie, valamint az épp hozzárendelésre szánt teremnek is), hanem mindkét folyamat során egynek-egynek egymástól függetlenül.

Az órarend készítésének alapjául a mesterséges intelligencia tantárgyból tanult állapottér reprezentáció szolgál. A kirakás alapjául pedig egy lépésszám figyelő BackTrack algoritmus szolgál.

4.3.1. Tantárgyfelosztások lehelyezése

A program egyik legkényesebb része az osztályokhoz rendelt tantárgyfelosztások megjelenítése az adatbázisban.
Egy osztályt egyértelműen a neve (egy darab kisbetű az angol abc-ből), valamint az évfolyama azonosít együttesen. Az iskolákban már évek óta használt technika, hogy az egyes évfolyamokon bizonyos órák egy időpontban tartandóak az évfolyam minden osztályának.
Az sem ritka, hogy ilyenkor az osztályokat több csoportra bontják, és minden osztályból csoportokat rendelnek össze, így alakítva ki összevont osztályokat az adott tantárgyakból.

Az adatbázis szempontjából az ilyen összevont osztályokhoz nem lehet korábban már felvett osztályokat rendelni, ugyanis egy osztálynak ilyenkor egyszerre több tanárral kéne órájának lennie több teremben. De egy oktatónak is több osztályt kéne tanítania egy időpillanatban, ahogy egy teremnek is több osztályt kéne befogadnia egyszerre.

Ennek a problémának a megoldására vezettem be az összevont osztály típusú órákat.
Vannak bizonyos tantárgyak, melyektől megköveteljük, hogy egymás után két darab kell, hogy tartva legyen belőlük, őket nevezzük dupla óráknak.

Amennyiben összevont, vagy dupla tantárgyakat tartalmazó tantárgyfelosztás letételére kerülne sor, a program generál egy véletlen napot és órát. Oda megpróbálja lehelyezni a tantárgyfelosztást.
Ha a tantárgyfelosztás osztályának nincs még órája az adott időpontban, és az oktatójának is szabad az adott időpont, akkor leteszi a tantárgyfelosztást, egyébként egy új, véletlenszerű időpontot generál.
Az általános osztályok letétele annyiban különbözik az előbbiektől, hogy itt csak egy napot generálunk véletlenszerűen, majd az adott nap első szabad órájára próbáljuk letenni az aktuális tantárgyfelosztást, ha nem sikerül egyik nap első órájára se letenni, akkor a sorban következő tantárgyat veszük, és azt próbáljuk meg letenni. Ezzel, hogy mindig az adott nap első szabad időpontjára próbálunk beszúrni kiküszöböljük a lyukas óra lehetőségét, mivel folyamatosan töltjük fel a napokat. Persze a lyukmentes órarendhez szükséges, hogy az előre rögzített órákat ne generálhassuk az adott napon túl későre, hogy előttük lyuk maradhasson.

4.3.1.1. Összevont órák

Hogy az olyan tárgyakat is feltudjuk venni adatbázisunkba, melyekhez egyszerre több osztály is tartozik egy egyedi megoldást alkalmazok. A tantárgyat nem osztályhoz, hanem évfolyamhoz rendeltetem, ugyanis az ilyen tantárgyakon az évfolyam összes osztálya képviselteti magát az osztályok egy-egy csoportjával.
Mivel megkövetelendő, hogy minden osztály neve egyetlen kis betű legyen az angol abc-ből, az összevont óra egy új osztályhoz fog hozzárendelődni, mely osztály neve az évfolyamra felvett osztályok neveiből összefűzött, majd a végére nagybetűkkel hozzáfűzött tantárgynév karaktersorozat, pl: abc1ANG. Ez annyit jelent, hogy az adott évfolyamon 3 osztály van

szerepeltetve, melyek nevei a, b, és c. Ebből a 3 osztályból hozunk össze egy angol 1-es csoportot, akik angolt fognak tanulni.

Kiértékeléskor elég az osztály nevét az 1-es számig vizsgálni, majd az addig kapott karaktersorozatban vizsgálni, hogy az adott osztály neve szerepel e benne. Ha igen, akkor az óra az adott osztályra is vonatkozik.
Ugyanezzel a technikával kezelendő a sávos matematika példája. Ebben az esetben az osztályok nincsenek összevonva, de a matematika órájuknak az évfolyamon egy időpontban kell lenniük. Mivel egy osztálynak egy újonnan felvett osztály feleltethető meg fontos, hogy ugyanannyi új osztályt hozzunk létre, ahány osztály van az aktuális évfolyamon.

Az órarend elkészítése szempontjából ezen tárgyak a legérzékenyebbek az áthelyezésre, ugyanis ilyenkor az nem 1, hanem egy egész évfolyamnyi osztály órarendjének a módosítását vonná maga után, ami előre beláthatatlan következményeket és ütközéseket vonna maga után.
Ezért a még üres órarendbe elhelyezzük az összevont és sávos tantárgyakat, és azokat a készítés további fázisában nem módosítjuk.

4.3.1.2. Dupla órák

Vannak órák, melyekből egy nap 2 darabot kell tartani, közvetlenül egymás után. Az általános tantárgyfelosztás felvétele során a dupla órákat nem lehet együtt tekinteni az általános órákkal, ezért külön felvételi lehetőséget kell nekik biztosítani.

Az órarend generálásakor ezen órák lehelyezése a következő lépés az összevont tantárgyak után, ugyanis a dupla órák későbbi letételéhez 2 egymást követő szabad időpontra lenne szükség, ami az órarend későbbi fázisában már nem biztos, hogy rendelkezésünkre áll. Ekkor előfordulhat, hogy az osztály egész órarendjét vissza kéne bontani a kezdetekig, hogy helyet találjunk óránknak. Ezt megelőzendő, az összevont tantárgyak lehelyezése után lehelyezzük a dupla tárgyakat, és azokat a futás további részében már nem módosítjuk.

4.3.1.3. Általános órák

Minden tantárgy, ami nem az előző két kategóriába tartozik általános tantárgynak tekintendő. Az ilyen tantárgyak alkotják az órarend tényleges kirakását, ugyanis a program csak ezen tantárgyakat mozgathatja. Ezen tárgyaknál egyetlen kikötés, hogy egy nap csak egy darab tartható belőlük. Ez alól kivételt képez, ha az adott tantárgy duplaóraként már fel van véve, ekkor az általános tárgyak közül ugyanaz a tantárgy az adott napon már nem szerepelhet.

4.3.2. Tantárgyfelosztás – Terem kapcsolata

Ha a tantárgyfelosztás kirakása sikeresen befejeződött, az órarendünkben már megjelentek az osztályok a hozzájuk rendelt tanárokkal és tantárgyakkal. Kiosztottuk hozzájuk az időpontokat.

Már csak egy feladatunk maradt, meghatározni, hogy melyik tantárgyfelosztás melyik teremben tartandó. Mivel a számítástechnika- és a testnevelés termeket kivéve az iskolában jelenleg használt rendszer alapján bármelyik tantárgy bármelyik osztályteremben tartható, érdemes a termek kiosztásának folyamatát is két részre bontani.

Az első lépésben a testnevelés- vagy számítástechnika órákat tartalmazó tantárgyfelosztásokhoz rendelnénk a tantermeket.

A második lépésben a maradék tantárgyfelosztáshoz rendelnénk a nem testnevelés-, vagy számítástechnika tantermeket, ezek közül is legelőször az összevont osztályokat megkövetelő tantárgyakat tartalmazó felosztásokhoz.

4.3.2.1. Testnevelés-, és számítástechnika termek

A testnevelés termekben csak testnevelés, míg a számítástechnika termekben csak számítástechnika órák tarthatóak. Az órarendkészítés ezen, második nagyobb fázisában a legfontosabb szempont, hogy az osztályok lehetőleg a saját osztálytermükben maradjanak a nap folyamán, és csak akkor hagyják el azt, ha a tantárgy megköveteli azt.

Nyilvánvalóan a számítástechnika- és a testnevelés órák ebbe a kategóriába esnek, mivel az ilyen termek nem lehetnek osztálytermek, de van egy harmadik lehetőség is, amiről említést kell tenni, ezek pedig az összevont osztályokat követelő összevont tantárgyak.

Példaként gondoljunk bele, hogy egy évfolyamnak adott évben 3 osztálya van, de a 3 osztályból osztályonként 4 csoportba sorolják a nyelvi órákra a diákokat, így 4 összevont osztályt hozva létre. Ekkor az első 3 összevont osztály elhelyezhető az alkotó osztályok osztálytermébe, viszont a fennmaradó 4. osztályt kénytelenek vagyunk egy olyan osztály osztálytermébe küldeni, akinek épp testnevelés, vagy számítástechnika órája van.

Ezen okfejtésből kiindulva fontos, hogy elsőként a testnevelés-, és számítástechnika tantermekhez rendeljünk tantárgyfelosztást. Így a későbbiekben, ha plusz osztályteremre lenne szükségünk egy osztály esetében, akkor elsőként az ilyen típusú termekben órán lévő osztályok osztálytermeit vizsgálhassuk.

4.3.2.2. Általános termek

A nem számítástechnika-, és testnevelés termek gyűjtő neve az általános terem kifejezés. Ezekben a termekben a testnevelés- és számítástechnika tantárgyak kivételével bármilyen óra megtartható. A termekhez szabadon rendelhetőek a tantárgyfelosztások, azonban a kikötések teljesülésének érdekében ezekhez a termekhez elsősorban azokat a felosztásokat rendeljük, amelyeknél a felosztás osztályának osztályterme az adott terem.

4.4. A program

Az órarendkészítő program egy grafikus felületen keresztül kommunikál a külvilággal. Menügombok lenyomásának hatására előtűnnek a megfelelő input ablakok, amelyek segítségével a megadott szabályok keretein belül tudja a felhasználó módosítani az adatbázis tartalmát.

4.4.1. Az órarend követelményei

A program legfontosabb feladata, hogy a korábbiakban megadott tantárgyfelosztásból egy olyan órarendet alkosson, mely a tantárgyfelosztás összes elemét tartalmazza ütközések nélkül. Egy osztálynak egy nap maximálisan 6 órája. Továbbá fontos kritérium, hogy egy tantárgy egy nap csak egyszer szerepelhessen, kivéve a dupla órákat. Az osztályok osztályfőnöki órájának az adott nap utolsó órájában kell lennie. Lehetőleg a kész órarend ne tartalmazzon lyukasórát az osztályok részéről, a tanárok oldaláról lehet. A nyelvi- és testnevelés óráknak kötelezően egyszerre kell lenniük, illetve ha az adott évfolyamon van sávos matematika, akkor annak is.
Mivel ez a rendszer igen komplex lesz, és sok helyen sérülhet, olyan kritériumokat fog kell lefektetni, melyek teljesülése nélkül nem hozható létre órarend.
A program alapértelmezetten az iskola jelenlegi paramétereinek megfelelően fogja tartalmazni a kritériumokat, melyek változtatására természetesen lehetőség van az újrafelhasználhatóság szelleme jegyében.

A kezdőkritériumok a következők:

- Egy osztálynak egy tantárgyfelosztásban maximálisan 28 órája lehet hetente egy félévre vonatkozóan .
- Egy tanár egy héten maximum 25 órát tarthat.
- Egy osztálynak egy adott félévre vonatkozóan maximum 17 féle tantárgya lehet.
- A jelenlegi helyzet alapján az órarend kirakásához legalább 35 terem szükséges, melyekből legalább 2-nek számítástechnika teremnek és legalább 5-nek testnevelés teremnek kell lennie.
- Egy tanárnak egy héten legalább 3 napot kell tanítani, tehát azon napok száma, amikor ő nem ér rá a tanításra egyéb elfoglaltságai miatt maximálisan 2 lehet.

Az iskolában a jelenlegi állapotok szerint egy évfolyamon 3 osztály indul minden évben, a kritériumokban feltüntetett adatok számolásánál ez is fontos feltétel volt.

Ezen adatok az órarendszerkesztésben segítő oktató számolásai alapján lettek rögzítve, mivel hivatalos szabály nincs ezen értékek beállítására.

Ha a korábban rögzített kritériumok bármelyike nem teljesül a program elindításakor, akkor a program működése leáll, mivel előreláthatóan az adott feltételek mellett egységes órarend nem generálható.
Amennyiben a feltöltött adatbázis eleget tesz a követelményeknek, a program megkezdheti az órarend generálását.

4.4.2. Rögzített táblák feltöltése

Vannak olyan táblák, melyek már a program első futása előtt feltöltött állapotban vannak a megfelelő sorokkal. Ezen adatokat tároló táblákhoz a felhasználó nem férhet hozzá, nem módosíthatja a tartalmaikat, csak megtekinthetőek

Az ilyen adatokat tároló táblák a következők:

Szakok: az iskola jelenlegi állapotát tükrözve 3 féle szakot különböztetünk meg:

- TES (testnevelés): az ilyen szakú termekben csak testnevelés tartható
- SZÁ (számítástechnika): az ilyen szakú termekben csak számítástechnika tartható
- ÁLT (általános): az ilyen szakú termekben tartható az összes többi tantárgy.

Idopontok: Minden munkanap minden óráját tartalmazza az elsőtől a hatodikig.

- Hétfő 1: hétfő első óra
- …
- Péntek 6: péntek hatodik óra

Tantargyak: Az iskolában tanított tantárgyak.

- BIO (biológia): Biológia tantárgy
- MAT (matematika): Matematika tantárgy
- …

Evfolyamok: Az iskola felsős évfolyamai.

- 5: Ötödikes évfolyam
- 6: Hatodikos évfolyam

- 7: Hetedikes évfolyam
- 8: Nyolcadikos évfolyam

4.4.3. A dinamikusan változó táblák feltöltése

Vannak táblák, melyek feltöltését a felhasználónak kell biztosítania. Ezen táblákba sorok felvétele, törlése, módosítása a program grafikus felületén keresztül történik. A fent említett táblák a következők: Tanarok, Termek, Osztalyok, OsztTantok.
Az OsztTantok tábla feltöltése csak az előbb helyes adatokkal ellátott előző három tábla megadása után végezhető el. Mivel ezen tábla félévről félévre változik lehetőség nyílik a tábla adatainak módosítására. Amennyiben az első három tábla bármelyikének bármely sorának adatai változnának, például a sor törlődne, akkor amennyiben ezek a változtatások érintenének más táblákat, a változások ezekben a táblákban automatikusan hajtódnának végre.
A végső órarend az Orarendek táblában tárolódik le. A felhasználó által felvitt tantárgyfelosztások ebben a táblában jelennek meg, mint egyedelőfordulások. Mivel a tantárgyfelosztás csak az oktató, tantárgy és osztály táblák kapcsolatát tartalmazza az Orarendek tábla soraiban a többi táblára hivatkozott oszlopoknál null értékek jelennek meg az előfordulásokban. Ezek értéket a program futása közben kapnak. Amikor a tábla egyik egyedelőfordulása sem tartalmaz null értékű attribútumot, az órarend elkészült.

4.4.4. A súgó

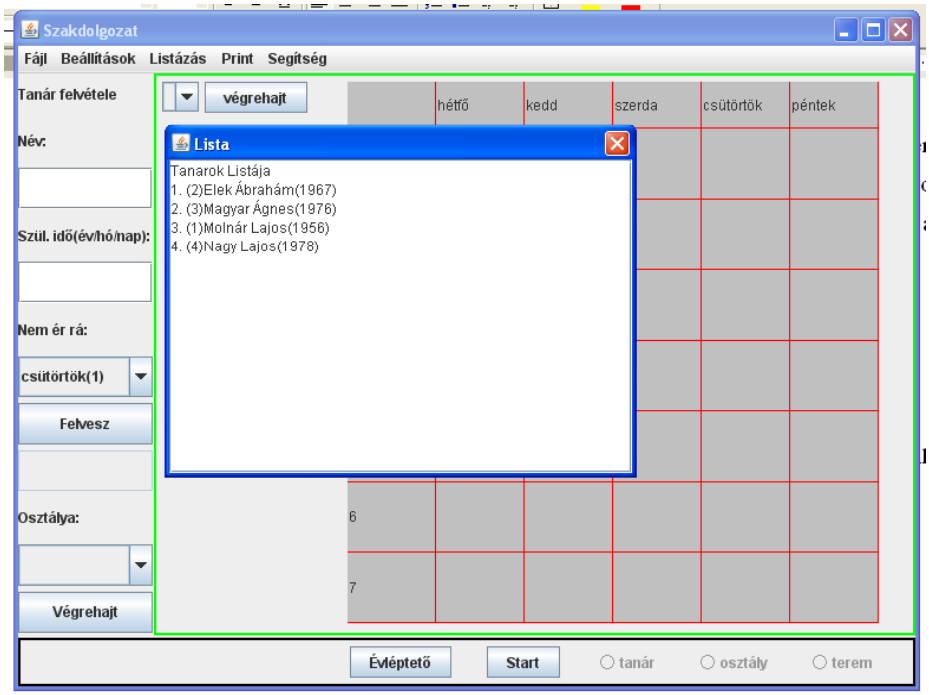
Menüből megnyitható ablak. A felhasználóban felvetődhető kérdésekre igyekszik válaszokat adni. Egy rádiógomb csoportból kiválasztható a kérdéses hiba, majd egy megjelenítő ablak szöveges mezőjében olvasható a felvetődött kérdésre a válasz.

4.4.5. Évléptetés

Az újrahasznosíthatóságot elősegítő lehetőség. Köztudott tény, hogy minden tanév végén elballagnak az aktuális 8.-os osztályok, és helyeiket az aktuálisan 7.-es osztályok töltik be. Ugyanez a procedúra megy végbe az 5.-es és 6.-os osztályok esetben is. Az aktuális 5.-es osztályok helyére pedig a program szemszögéből nézve új osztályok érkeznek.

Ennél a procedúránál a manuális törlések és újrafelvételek sok hiba lehetőségét rejtik magukban, ezért a programot használó személy szempontjából lényegesen felhasználóbarátabb megoldás, ha a fent említett procedúrát egyetlen gomb lenyomásával végrehajthatjuk. Ennek céljából került a felületre az „évléptető” feliratú nyomógomb.
A gomb egyszeri lenyomása után törlődnek a 8.-os osztályok és az összes rájuk történő hivatkozások, az alacsonyabb évfolyamú osztályok pedig évfolyamot lépnek.
Az évfolyamot lépett osztályok tantárgyfelosztásai megmaradnak, mindösszesen a hozzájuk kötődő évfolyam értékek emelkednek eggyel.

4.4.6. Listázás



Fontos szempont a program használata során, hogy ellenőrizni tudjuk az adatbázisban tárolt adatok jelenlegi állapotát, hiszen a program félévente egyszeri használatra készült, így nem lehet folyamatosan nyomon követni az adatok aktuális állapotát. Ennek a segítségére szolgál a menü Listázás menüpontja. Ezen belül kiválasztható, hogy a tanárok, az osztályok, esetleg a termek listáját szeretnénk megtekinteni. A megfelelő menü kiválasztása után egy előugró ablak kilistázza nekünk a kért adatokat, megjelenítve azok tulajdonságait is. Például ha az órarend generálásunk nem indulna el, mert valamelyik osztálynak nincs osztályfőnöke, vagy osztályterme, akkor a kilistázások segítségével megtekinthető, hogy melyik osztálynál kell módosításokat eszközölnünk.

4.4.7. Órarend készítése

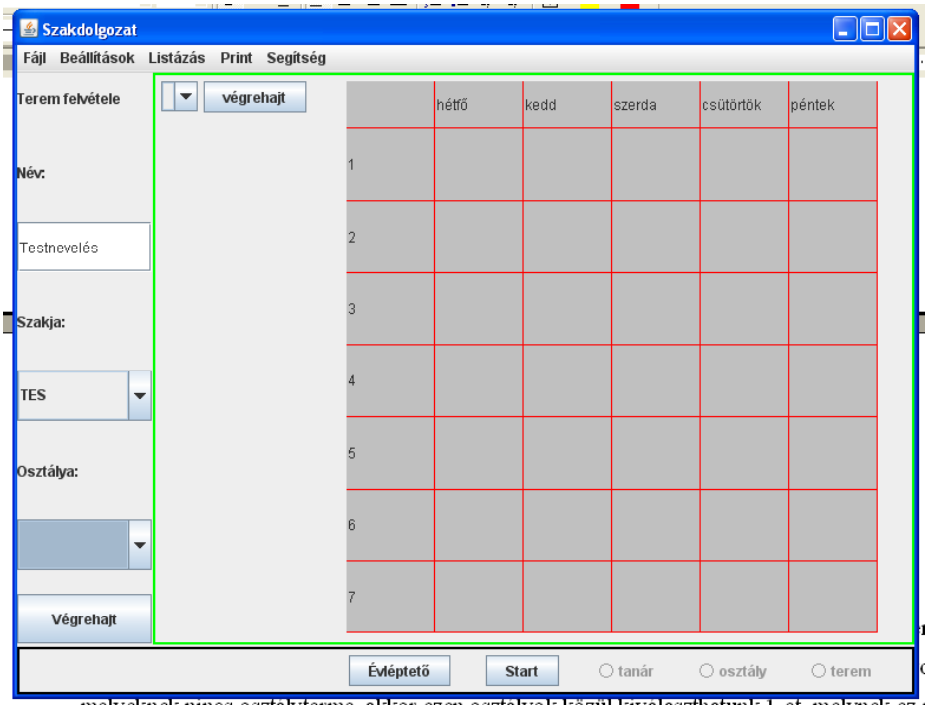
Az órarend készítése folyamán fontos szempont az adatok eltárolása. Vannak a program kódjában előre elhelyezett adatok, ezeket nem lehet módosítani, csak hozzájuk férni, és vannak a felhasználó által felvehető adatok, melyek módosítása szabadon megengedett.

4.4.7.1. Tanárok elhelyezése az órarendben

Az órarendkészítés első lépéseinek egyike a Tanarok tábla feltöltése. Ezt megtehetjük a menü Fájl/Módosít Tanár/felvesz menüpontja alatt, ahol megadhatjuk a felvenni kívánt oktató nevét, születési évét, esetleges elfoglaltságának időpontjait, mely időpontokhoz a program nem rendelhet oktatást neki, illetve ha az Osztalyok táblában szerepelnek olyan osztályok, melyeknek nincs osztályfőnöke, akkor ezen osztályok közül kiválaszthatunk 1-et, melynek az oktató osztályfőnöke lesz (nem kötelező egy oktatónak osztályfőnöknek lennie).

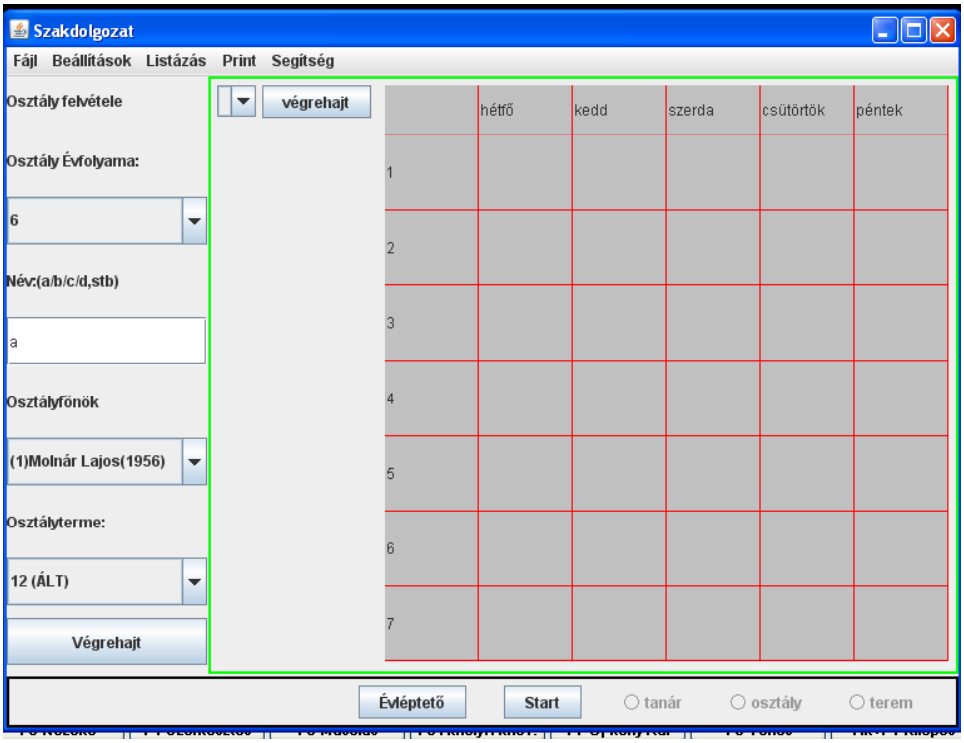
Amennyiben egy oktatót törölni szeretnénk, mert például nyugdíjazták, akkor a menü Fájl/Módosít Tanár/töröl menüpontja alatt törölhetjük a kiválasztott oktatót. Ekkor az összes olyan előfordulás törlődik a megfelelő táblákból, melyeknél az oktatóra hivatkozás volt.

4.4.7.2. Termek elhelyezése az órarendben



Az órarendkészítés másik kezdeti lépése a Termek tábla feltöltése. Ezt megtehetjük a menü Fájl/Módosít Terem/Felvesz menüpontja alatt, ahol megadhatjuk a felvenni kívánt terem nevét és kiválaszthatjuk a szakját, illetve ha az Osztalyok táblában szerepelnek olyan osztályok, melyeknek nincs osztályterme, akkor ezen osztályok közül kiválaszthatunk 1-et, melynek ez a terem lesz az osztályterme (nem kötelező egy teremnek osztályteremnek lennie).

Amennyiben egy termet törölni szeretnénk, akkor a menü Fájl/Módosít Terem/Töröl menüpontja alatt törölhetjük az kiválasztott termet. Ekkor az összes olyan előfordulás törlődik a megfelelő táblákból, melyeknél a teremre hivatkozás volt.



4.4.7.3. Osztályok elhelyezése az órarendben

Az órarendkészítés ajánlottan harmadik lépése az Osztalyok tábla feltöltése. Ezt megtehetjük a menü Fájl/Módosít Osztály/Felvesz menüpontja alatt, ahol megadhatjuk a felvenni kívánt osztály nevét és kiválaszthatjuk az évfolyamát. Ha a Tanarok táblában szerepelnek olyan oktatók, akik még nem osztályfőnökök, akkor ezen tanárok közül kiválaszthatunk 1-et, aki az adott osztály osztályfőnöke lesz. Ha a Termek táblában szerepelnek olyan termek, melyeknekhez még nincs osztály rendelve, akkor ezen termek közül kiválaszthatunk 1-et, mely az adott osztály osztályterme lesz.
Fontos, hogy míg az oktatóknál és a tantermeknél nem volt kikötés, hogy osztályfőnöknek, osztályteremnek kell lenniük, addig az osztályok esetében kötelezően kell, hogy legyen osztályfőnöke és osztályterme.

Amennyiben egy osztályt törölni szeretnénk, akkor a menü Fájl/Módosít Osztály/Töröl menüpontja alatt törölhetjük az kiválasztott osztályt. Ekkor az összes olyan előfordulás törlődik a

megfelelő táblákból, melyeknél az osztályra hivatkozás volt. Fontos, hogy egy osztály csak azután törölhető, vagy felvehető, miután az évfolyamból töröltük az összevont órákat. Ugyanis az Összevont órákhoz tartozó Összevont osztályok neveikben tartalmazzák az adott osztály nevét, ami az osztály törlése esetén a későbbiekben hibákhoz vezethet.

4.4.7.4. Tantárgyak elhelyezése az órarendben

Az órarendkészítés következő lépése a tantárgyfelosztás felvitele, és ezáltal az OsztTantok és az Orarendek táblák feltöltése.
A korábban említett 3 tantárgy kategória közül az Általános órákat tartalmazó tantárgyfelosztások felvehetőek a menü Beállítások/Tantargyfelosztas/Tantargyfelvétel menüpontja alatt. Itt kiválasztható egy oktató, egy osztály, egy tantárgy, valamint megadható, hogy az adott tantárgyat heti hány órában tanulja az adott osztály. Ezek az értékek új sorként bekerülnek az OsztTantok táblába, majd a heti óraszámnak megfelelő darab új sort kerül be az Orarendek táblába az adott egyedelőfordulás hivatkozásával.

Hasonló módon vehetőek fel a Dupla órákat tartalmazó tantárgyfelosztások. Ezen órák felvevő ablaka a menü Beállítások/Tantargyfelosztas/DuplaTantargyfelvetel menüpontja alól érhető el. Itt kiválasztható egy oktató, egy osztály, illetve egy, a Dupla órák közül. Ennek felvétele az előzőhöz hasonló módon történik, egyetlen eltérése, hogy a heti óraszáma automatikusan 2 lesz.

Speciális az Összevont órákat tartalmazó tantárgyfelosztások felvétele. Ezen órák felvevő ablaka a menü Beállítások/Tantargyfelosztas/ÖsszevontTantargyfelvetel menüpontja alól érhető el. Itt kiválasztható egy oktató, egy évfolyam, és egy, az Összevont órák közül, valamint megadható a heti óraszám. Ekkor a már korábban említettek alapján létrejön egy Összevont osztály, nevében tartalmazva az évfolyam osztályainak neveit összefűzve. Ezek után a felvétel ugyanúgy történik, mint az első esetben.

Lehetőség nyílik a tantárgyfelosztások módosítására is. Ez megtehető a menü Beállítások/Tantargyfelosztas/Tantárgymódosítás menüpontja alól. Természetesen az imént említett tantárgyfelosztások közül, csak az első és a harmadik heti óraszáma módosítható, ugyanis a dupla órák kötelezően két órában tartandóak.

A tantárgyfelosztások törlése a menü Beállítások/Tantargyfelosztas/Tantárgytörlés menüpontja alól érhető el. Itt kiválasztható a kívánt tantárgyfelosztás, illetve megadható, hogy mennyi legyen az új heti óraszáma. A végrehajtása hatására előbb az Orarendek táblából törlődnek a hivatkozott sorok, majd frissül az adott osztály-tantárgy kapcsolat az OsztTantok táblában, és bekerülnek az új egyedelőfordulások az Orarendek táblába az új heti óraszámnak megfelelően.

4.4.7.5. Órarend generálása

Az órarend generálás a Start gomb lenyomásával veszi kezdetét. Ekkor az évléptető gombot letiltjuk, majd kiértékelődnek a kritériumok, ha nincs ütközés a kivételeknél, akkor kinullázódnak a korábbi generálásból hátramaradt, immár felesleges értékek, majd a korábbi elveknek megfelelően kezdetét veszi az órarend generálása.

4.4.8. Eredmény megtekintése

Az órarend sikeres legenerálása után választhatunk, hogy tanár, osztály, vagy tanterem órarendjét szeretnénk megtekinteni. Ekkor előbb a lent elhelyezett rádió gombok segítségével ki kell választani a megjelenítés tárgyát, majd a legördülő menüből az aktuális egyedet. Miután nyugtáztuk választásunkat, a táblázatban megjelenik a választott egyedhez fűződő órarend.

4.4.9. Adatvédelem és Biztonság

A program legfontosabb védelmi feladata, hogy az adatbázisban a program készítője által előre tárolt adatok módosítását ne tegyük lehetővé a felhasználók részére.
Ennek érdekében a program kezdeti állapotában már tartalmazza ezeket az értékeket. Ezek módosítása, törlése nem lehetséges, mivel a felhasználónak nincs jogosultsága közvetlenül hozzáférni a táblákhoz, csak a grafikus felületen keresztül, bizonyos kereteken belül módosíthatja azokat.

Az adatok felvétele előtt a bevitelre szánt adatokat a program megvizsgálja, és ha valamelyik adat nem felelne meg az aktuális beviteli mezőre előírt szabályoknak, vagy a felvétel során hiba történik, például leáll a server, akkor a hibernate egy rollback-el visszagörgeti az adatbázist az előző, még helyes állapotába.
Amennyiben a későbbiek során szükségessé válna új értékek felvétele valamelyik táblába, melyek módosításai kiesnek a felhasználó hatásköréből, a program készítőjéhez kell fordulni.

4.5. Implementáció

A program fejlesztése során a Netbeans IDE 5.5-ös kezelőfelületét használtam.
A forráskód Java nyelven lett implementálva, a Java programnyelv JDK 6.1 nevű generációjában.
A Netbeans IDE5.5 és a JDK 6.1 Update együttesen ingyen letölthető a következő címről:
http://www.netbeans.info/downloads/index.php

Az adatbázist a HSQLDB relációs adatbázis-kezelő rendszer kezeli, mely ingyenesen letölthető a következő linkről:
http://sourceforge/project/showfiles.php?group_id=23316&package_id=16653

Az adatbázis tábláinak kezelését a Hibernate Core 3.2.2 GA végzi, mely ingyenesen letölthető a: http://www.hibernate.org/30.html címről.

4.5.1. Az adatbázis implementációja

4.5.1.1. Hibernate Core 3.2.2 GA

Ha Javaban egy adatbázis tábláit hibernate-el szeretnénk kezelni, akkor a legfontosabb, hogy a hibernaten tudja,hogy hogyan tudja betölteni és tárolni a perzisztens osztályok objektumait. Ezt a célt szolgálja a mapping fájl, mely megmondja a programnak, hogy melyik táblához kell, hogy hozzáférjen, és a táblákon belül melyik oszlopokat kell használnia.

4.5.1.2. Mapping fájlok

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
[…]
</hibernate-mapping>
```

AZ XML fájlok sémáját a DOCTYPE-ban definiált DTD határozza meg. A Hibernate DTDje elég bonyolult.
A hibernate ezt a DTD fájlt a program classpath-jából tölti be.
Arra használjuk, hogy automatikusan kiegészítsük azXML mapping fájl elemeit
és tulajdonságait.

A hibernate3.jar nevű fájl tartalmazza a DTD fájlt, mely csomag segítségével hivatkozhatunk a programunkban a hibernatere.

```
<hibernate-mapping>

        <class name =”csomagnév.osztálynév” table=”táblanév”>

        </class>

</hibernate-mapping>
```

hibernate-mapping tag-ek között helyezkednek el az osztály elemek. Miután megadtuk, hogy milyen formában kell az adott osztályt leképezni az adott táblára, minden egyed le lesz képezve egy sorra.

4.5.1.3. Adatbázist kezelő fájlok

Az adatbázis kezelése során minden táblához saját mapping fájlt hoztam létre, a jobb elkülöníthetőség érdekében, igaz, ezeket a mapping fájlokat egy fájlba is össze lehetett volna szerkeszteni.

A program 12 táblát kezel, így 12 különböző mapping fájlra volt szükségem a program készítése során. Mivel az összes fájlt felesleges lenne megjelenítenem, egy példán mutatom be a mapping fájl felépítését, mely tartalmazza az összes elemtípust, melyeket a többi fájlokban is használtam.

Termek tábla kezelését megvalósító mapping fájl:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping auto-import="false" schema="orarend">

    <class name="orarend.Terem" table="termek">

    <composite-id name="id" class="orarend.TeremId">
        <key-property name="teremnev" type="string" length="30"/>
      <key-many-to-one name="szakkod" class="orarend.Szak" column="szakkod"/>
    </composite-id>

    <many-to-one name="osztaly" class="orarend.Osztaly">
        <column name="osztnev"/>
        <column name="evf_kod"/>
        </many-to-one>

        </class>

</hibernate-mapping>
```

Haladjunk lépésről lépésre kintről befele

```
<hibernate-mapping auto-import="false" schema="orarend">
    […]
</hibernate-mapping>
```

A korábban már említett hibernate-mapping tag, melynél látható, hogy paraméterként, az automatikus import le van tiltva,illetve megmondjuk, hogy a tábla az orarend nevű sémában helyezkedik el.

```
<class name="orarend.Terem" table="termek">
    […]
</class>
```

Az orarend csomag Terem osztálytagja fogja kezelni a termek nevű táblát.

```
<composite-id name="id" class="orarend.TeremId">
        <key-property name="teremnev" type="string" length="30"/>
        <key-many-to-one name="szakkod" class="orarend.Szak" column="szakkod"/>
</composite-id>
```

Összetett kulcs definiálása. Az összetett kulcs neve az osztályban id, melynek osztálya az orarend csomagban található TeremId.

Elsődleges kulcs a teremnev, mely string típusú, és maximálisan 30 hosszú lehet.
Külső kulcshivatkozás a szakkod osztálytag, mely az orarend csomag Szak osztályából származtatható. Ennek az oszlopnak a neve a táblában szakkod lesz.
Amennyiben nem adunk meg oszlopnevet, a hivatkozott változó nevét kapja az oszlop, előző esetben teremnev.

```
        <many-to-one name="osztaly" class="orarend.Osztaly">
                <column name="osztnev"/>
                <column name="evf_kod"/>
        </many-to-one>
```

1:N típusú kapcsolat realizálható a many-to-one tag segítségével. Mivel a fizikai adatmodellben látható volt, hogy a táblák között csupa 1:N kapcsolat valósul meg, így feladatomban csak many-to-one tagek használatára volt szükség.
Az M:N kapcsolat típust a many-to-many, még az N:1 kapcsolat típust a one-to-one tagek valósítják meg.

A hibernatenek van még egy fontos tage, mellyel autómatikusan generáltathatunk elsődleges kulcsot egy egyedhez.

```
<id name="oktkod" type="long" unsaved-value="null">
                        <generator class="native"/>
</id>
```

A példában a Tanarok tábla elsődleges kulcsát létrehozó taget láthatjuk, mely a long típusú oktkod változóhoz rendel automatikusan elsődleges kulcsértéket, ezt a <generator class="native"/> tag valósítja meg.

A mapping fájlok létrehozásával csak az első lépést tettük meg tábláink kezelésének irányába. Ugyanis a táblák mapping fájlaihoz létezniük kell a hivatkozott osztályoknak, valamint azok hivatkozott változóinak is.

A Terem osztaly:

```
package orarend;
import java.util.*;

public class Terem extends TeremId{

   protected TeremId id;
   protected Osztaly osztaly = null;
   protected Szak szakkod;
   public Terem() {
   }



   public Terem(String ternev, Szak szkod){

      id = new TeremId(ternev,szkod);
      szakkod = szkod;
   }

   public TeremId getId(){
      return id;
   }

   public void  setId(TeremId _id){
      id =_id;
   }

   public Szak getSzakkod() {
      return szakkod;
   }

   public void setSzakkod(Szak sza) {
      szakkod=sza;
   }

   public Osztaly getOsztaly() {
                  return osztaly;
         }
```

```
    public void setOsztaly(Osztaly o) {
          osztaly = o;
    }

    public boolean equals(Object o){
       if(o==null)
          return false;
       if(o instanceof Terem){
          Terem t = (Terem)o;
          if(id.equals(t.getId()) && szakkod.equals(t.getSzakkod()))
             return true;
       }
       return false;
    }

    public String toString() {
       return getId().toString();
    }

}
```

A TeremId osztály:

```
package orarend;

public class TeremId  implements java.io.Serializable{


    protected String teremnev;
    protected Szak szakkod = null;


    public TeremId(){
    }


    public TeremId(String nev, Szak szak) {
       this.teremnev = nev;
       szakkod = szak;
    }

    public String getTeremnev(){
       return teremnev;
    }

    public void setTeremnev(String tn){
```

```
        teremnev = tn;
    }

    public Szak getSzakkod() {
                    return szakkod;
            }

    public void setSzakkod(Szak sza) {
            szakkod = sza;
    }


    public boolean equals(Object o){
        if(o==null)
            return false;
        if(o instanceof TeremId){
            TeremId t = (TeremId)o;
            if(szakkod.equals(t.getSzakkod()) && teremnev.equals(t.getTeremnev()))
                return true;
        }
        return false;
    }

    public String toString() {
        return getTeremnev()+" "+getSzakkod().toString();
    }


}
```

A példából elsőre is szembetűnik, hogy az osztályok változóihoz hozzá van rendelve egy get és egy set függvény, melyek neveinek folytatása a változó neve, nagybetűvel kezdve.

Például:

```
protected String teremnev;

public String getTeremnev(){
        return teremnev;
    }

public void setTeremnev(String tn){
        teremnev = tn;
    }
```

Ezeket a metódusokat gettereknek és settereknek nevezzük. A hibernate ezen metódusokon keresztül tud adatokat lekérdezni, módosítani, feltölteni a táblákban, hiányuk esetében a mapping fájl alapján a hibernate nem tudja kezelni az adott táblát.

Láttuk, hogy hogyan lehet táblákra hivatkozásokat létrehozni, nincs más hátra, mint a táblák kezelésének bemutatása.

```
        Session         session = null;
        Transaction     tx = null;
        Configuration cfg = new Configuration()
                        .addClass(orarend.Terem.class)
                        .addClass(orarend.OsztTant.class)
                        .addClass(orarend.Szak.class)
                        .addClass(orarend.Evfolyam.class)
                        .addClass(orarend.Tanar.class)
                        .addClass(orarend.Orarend.class)
                        .addClass(orarend.Idopont.class)
                        .addClass(orarend.Tantargy.class)
                        .addClass(orarend.RosszOsztalyIdopont.class)
                        .addClass(orarend.RosszTanarIdopont.class)
                        .addClass(orarend.RosszTeremIdopont.class)
                        .addClass(orarend.Osztaly.class);
   SessionFactory       sf = cfg.buildSessionFactory();
try {
                        session = sf.openSession();
                        tx = session.beginTransaction();

                        […]

                tx.commit();
            } catch(HibernateException e) {
                        if (tx != null) {
                                try {
                                        tx.rollback();
                                } catch(HibernateException ex) {}
                        }
                } finally {
                        if (session != null) {
                                try {
                                        session.close();
                                } catch(HibernateException e) {}
                        }
            }
```

A példában egy adatbázis-kapcsolat létrehozásának példáját láthatjuk.

A Session típusú változó egy „szessön”, mely lehetővé teszi az adatbázishoz való csatlakozást.

A Transaction típusú változó maga a tarnzakció, ami addig tart, míg egy commit-tal végre nem hajtatjuk a módosításokat, vagy a rollback vissza nem görgeti valamilyen hiba miatt.

Configuration : beállítást szolgál, hozzá kell adni az összes olyan mapping fájlt, melyre hivatkozás történhet az adott tranzakció keretein belül.

SessionFactory: az adatbázihoz való hozzáférést valósítja meg.

Mivel az adatokhoz való csatlakozást a Session valósítja meg, így az adatok módosítását is rajta keresztül végezhetjük.

A 4 legfontosabb módosítási fajta a betöltés, törlés, kilistázás, módosítás.

Ezek példái (a beillesztett kódrészletek a Transaction törzsében helyezhetőek el a […] rész helyébe):

Betöltés, Módosítás és Törlés

```
Tanar tanar = null;
                Long oktkod = Long.valueOf(”1”);
                tanar = (Tanar) session.load(orarend.Tanar.class,oktkod);
                tanar.setNev(”Molnár Lajos”);
               session.update(tanar);
               session.delete(tanar);
```

Betöltjük az 1-es oktató kóddal rendelkező oktatót, ha létezik ilyen, majd a nevét Molnár Lajosra változtatjuk, a módosítást elmentjük, majd töröljük az oktatót.

Listázás

```
Iteratori = session.createQuery("from orarend.Terem t order by t.id.teremnev").iterate();

                while (i.hasNext()) {
                                Terem terem = (Terem) i.next();
                }
```

A Termek tábla végigjárása teremnév szerint rendezve.

4.5.2. A felület implementációja

A program grafikus felhasználói felülete (GUI) a Java Swing osztálygyűjteménye segítségével lett megvalósítva. A Swing a Java régebbi osztálygyűjteményére, az AWT-re épül, ezt fejlesztették tovább. Az AWT-vel ellentétben a Swing komponensek már nem használják az operációs rendszer elemit, maguk rajzolják meg a komponenseket, így azok különböző operációs rendszer alatt is ugyanúgy néznek ki.

Maga a megjelenítés egy keretben, és azon belül 3 JPanel típusú konténerben történik.
A panel feladata, hogy a benne elhelyezett elemeket összefogja, azokat egy elrendezési stratégia alapján elhelyezze.

A baloldalra helyezett panel tartalma a menü függvényében változik, míg a középre és lentre helyezett panel tartalma a program futása során változatlan marad.

```
public class Frame extends JFrame implements ActionListener{
        […]
}
```

A Frame osztály kiterjeszti a JFrame osztályt, illetve implementálja az ActionListener interfészt. Így az osztályunk futtatáskor létrehoz egy keretet, melyen eseményfigyelést lehet alkalmazni.

```
JPanel jp1, jp2, jp3;
static JPanel paneltomb [] = {new JPanel(),…);

jp1 = paneltomb[0];
jp2 = new JPanel2();
jp3 = new JPanel3();

setLayout(new BorderLayout(1,2));
```

```
this.add(jp1,"West");
this.add(jp2,"Center");
this.add(jp3,"South");
```

Ha az osztály konstruktorába elhelyezzük a vezérlő utasításokat, akkor előbb meghatározódik egy elrendezési stratégia a kereten belül, mely jelen esetben BorderLayout, mely a konténer határain rendezi el a komponenseket 4+1 égtáj szerint.
Ezek lehetnek: "North", "South", "West", "East", "Center"

Miután létrehoztuk a konténereket és meghatároztuk az elrendezési stratégiáikat már nincs akadálya, hogy elemeket tegyünk beléjük.

```
JButton bvegrehajt = null;
JComboBox cbtantervek = null;
JText text = null;

add(new JLabel("Címke"));
add(cbtantervek = new JComboBox());
add(text = new Jtext());
add(bvegrehajt = new JButton("Végrehajt"));
bvegrehajt.setCursor(new Cursor(Cursor.HAND_CURSOR));
bvegrehajt.addActionListener(this);
```

Amennyiben egy konténer konstruktorában elhelyeznénk az add függvényhívásokat, akkor sorrendben előbb egy címke jönne létre, mely megjeleníti a komponensen a Címke feliratot, majd egy üres JcomboBox típusú legörgethető lista, ezt követően egy Jtext típusú szöveges mező, legvégül pedig egy Végrehajt feliratú nyomógomb, melyre ha az egérkurzorral rámegyünk, a kurzor kéz formájúra változik.

Nyomógombunkra eseményfigyelést állítottunk.
Ahhoz, hogy ez működjön is, az osztályban el kell helyeznünk az ActionListener interfész actionPerformed metódusának megvalósítását, melyben meg kell adni, hogy mit csináljon az esemény bekövetkeztekor a program.

```
public void actionPerformed (ActionEvent ae){
```

```
        if (ae.getSource()==bvegrehajt){
        […]
        }
}
```

A […] helyre lehet elhelyezni az esemény bekövetkeztekor végrehajtandó utasításokat.

Végül essék pár szó a program előugró ablakairól. Ezen ablakok Jdialog típusúak. Céljuk információközlés a felhasználóval a program futása közben. Ha futtatás alatt egy ilyen ablak kerül ki a képernyőre, a program futása félbeszakad, és addig nem folytatódik, amíg az ablakot be nem zárjuk.
Programomban tipikus példa erre a hibákra figyelmeztető ablak, mely példányosításkor egy Stringet vár paraméterként, majd ezt a Stringet jeleníti meg az ablakban, mint hibaüzenetet.

```
public class Hiba extends JDialog{

public Hiba(String hiba) {
      Toolkit tk = Toolkit.getDefaultToolkit();
      Dimension kepernyomeret = tk.getScreenSize();
      JTextArea text;
      setBounds(kepernyomeret.width/2-100 , kepernyomeret.height/2-40 , 200 , 80);
      setTitle("FIGYELEM");
      add(text = new JTextArea());
      text.setSize(30,30);
      text.setEditable(false);
      text.setText(hiba);

      setModal(true);
      setVisible(true);

   }
}
```

Az ablak elrendezés szempontjából a képernyő középpontjában helyezkedik el, 200 pixel szélesen és 80 pixel magasan.

5. Összefoglalás

Szakdolgozatomban egy információs rendszer tervezését és fejlesztését mutattam be.
A program készítése során figyelembe vettem az újrahasznosíthatóság és a továbbfejlesztés lehetőségeit.
A program fejlesztését nem sikerült teljesen befejezni. A jelenlegi állapotok szerint a program funkciójának megfelelően hibátlanul kezeli az adatbázist, kivételek bekövetkeztekor annak okaira figyelmeztet.
A 4.3. pontokban leírt stratégiának megfelelően a program az összevont, valamint a duplán tartandó órákat tárolja az adatbázisban.
Ami még hiányzik: az előbbi két letároláshoz hasonlóan az általános típusú tantárgyak elhelyezése az órarendben, valamint ugyancsak hasonló módon az így kapott, kezdeti órarendhez terembeosztás rendelése.

6. Felhasználói útmutató

A program futtatása 2 lépésben történik.

Első lépésként futtatni kell a CDn szereplő szakdolgozat/database/runDatabaseServer.bat nevű fájlt, mely elindítja az adatbázisunkat.

Második lépésben kerül futtatásra maga a program, melyet a szakdolgozat/orarend/run.bat fájl indít el.

A programban a fájl menüpont alatti lehetőségeknél tudunk tanárt, termet, osztályt felvenni az adatbázisba, a beállítások menüpont alatt vehető fel a tantárgyfelosztás.

A program futása a start gomb lenyomásával veszi kezdetét.

A program leállításához előbb zárjuk be a program ablakát, majd a még futó server abalkában nyomjunk ctrl+c billentyűkombinációt, majd a feltett kérdésre válaszként egy i betűt. Ezzel sikeresen leállítottuk az adatbázis-kezelő programot, melyben eltárolódtak a program futtatása közben eszközölt változtatások.

7. Irodalomjegyzék

Könyvek:

Nyékyné Gaizler Judit: Java 2 útikalauz programozóknak 1.3
Angster Erzsébet: Objektumorientált tervezés és programozás

Linkek:

http://hsqldb.org/
http://www.hibernate.org/hib_docs/v3/reference/html/tutorial.html
http://inf.unideb.hu/~jeszy/adatkezeles/index.html

8. Köszönet

Munkámat elsősorban témavezetőm, Dr. L. Nagy Éva, valamint a Kazinczy Ferenc Kertvárosi Általános Iskola órarendjének felelőse, Kövér Gábor segítette.

Továbbá köszönet illeti Jeszenszky Péter, egyetemi tanársegédet, aki a 2006/2007-es tanév őszi félévének felétől lehetőséget nyújtott nekem J2SE adatkezelés nevű órájának látogatására, ahol megismerkedhettem a Hibernate Core 3.2.2 GA alapjaival.