# 1. fejezet BEVEZETÉS

## 1.1. Témaválasztás

A témaválasztást két kiemelt szempont befolyásolta:

- egyrészt hosszú évek óta kedvenc területem az adatbázis-kezelő rendszerek tervezése és fejlesztése, ehhez a területhez kapcsolódó kliens-szerver architektúrák optimalizálása; ez a terület rendkívül összetett ismereteket és felkészültséget igényel, tehát nagy kihívás,
- másrészt munkahelyemen az Allianz Hungária Biztosítónál valós igény jelentkezik naprakész vezetői információs rendszerek elkészítésére.

A *VINFO 2007 (Vezetői INFOrmációk 2007)* elnevezésű rendszer is egy ilyen – a menedzsment részéről jelentkező – igény alapján készült el. A feladat összetettsége pontosan azt a kihívást jelenti, ami szükséges a szakmai fejlődéshez. Nem utolsó sorban egy éles rendszerről van szó, amelyben jelenleg 60 felhasználó végzi napi munkáját, ami nagy felelősség egy fejlesztő, rendszergazda számára, ugyanakkor – a ritka – sikerek forrása is. Ez szintén szükséges a nehéz órákban.

## 1.2. A diplomamunka célja

Olyan éles rendszer elkészítése, amely mindenben eleget tesz a felhasználói követelményeknek, sőt módszereiben, megoldásaiban irányt is mutat a felhasználóknak. További célom olyan elosztott rendszermodell kialakítása, amely mind a hardver-, mind a szoftver erőforrásokat optimálisan kihasználja, nem pazarló azokkal. Megítélésem szerint jelenleg két nagy irányzat egyszerre van jelen az adatbázis-kezelő rendszerek tervezésénél és implementálásánál:

- az egyik irányzat webes eszközökre alapozva készít un. háromrétegű kliens-szerver adatbázis-kezelő rendszereket, vékony kliens[1] réteggel,
- a másik irányzat a kétrétegű kliens-szerver architektúrát részesíti előnyben, ahol egy speciálisan elkészített kliens programot szükséges a kliens számítógépekre telepíteni.

Természetesen mindkét irányzatnak vannak előnyei és hátrányai. Diplomamunkámban megpróbálom a két rendszerfejlesztési irány előnyeit ötvözni, azaz elkészítek egy kettő- és háromrétegű kliens-szerver adatbázis-kezelő rendszert, amelyben mindkét irányzat azt valósítja meg, amiben jó. Ebből a szempontból talán újszerűnek is tekinthető ez az elosztott rendszermodell.

## 1.3. A diplomamunka felépítése

A diplomamunka hat fejezetre tagolódik. Az egyes fejezetek summáját a következő hat pontban lehet megvonni:

1. fejezet BEVEZETÉS: tisztázza az alapvető motivációkat, célokat, meghatározza a rendszerfejlesztés módszerét és eszközeit az újszerű rendszermodell felállításához és a teljes rendszer elkészítéséhez,

[1] A vékony kliens réteg technológia a kliens számítógépeken lényegében egy webes böngészőre alapoz. Ez a webes böngésző jelenti magát a kliens programot, amelyen keresztül történik a kommunikáció. Ebben a modellben többnyire egy http kiszolgálót és egy adatbázis-szervert üzemeltetnek. A http szerveren helyezik el magát a webes alkalmazást, amelynek segítségével az egyes felhasználók konkurens módon csatlakoznak az adatbázishoz. Tehát a http host a kliensréteg és az adatbázis-szerver között helyezkedik el topológiai szempontból, így alakul ki a három réteg.

2. fejezet FELADATSPECIFIKÁCIÓ: definiálja az alapvető feladatokat, amelyeket a rendszernek meg kell valósítani, meghatározza az elemzés, konceptuális tervezés fő szempontjait és a kitűzött feladatokhoz hozzárendeli a megfelelő rendszermodellt, ugyanakkor meghatározza a rendszerkörnyezetet a rendszerhez kapcsolódó platformokat, konfigurációs- és telepítési paramétereket,
3. fejezet LOGIKAI RENDSZERTERVEZÉS: magában foglalja a tervezés teljes egészét, úgy mint:
    a. az adatbázis-tervezést,
    b. az adatbázis objektumainak és eljárásainak a tervezését,
    c. a tranzakció kezelést mind az adatbázis-, mind a kliens program szintjén,
    d. az adatbiztonság, adatvédelem módszereinek a tervezését,
4. fejezet RENDSZERDOKUMENTÁCIÓ: definiálja – első sorban szakemberek, fejlesztők számára – a rendszerkomponenseket, azok helyét, szerepét, fontos attribútumait a teljes rendszeren belül, megadja a telepítés pontos módszerét, tisztázza a rendszerkomponensekhez rendelt jogosultságokat, pontos leírást ad a projekt fontos elemeiről, objektumairól, összegző elemzést nyújt a rendszerállományokról, forrásokról, script-ekről,
5. fejezet FELHASZNÁLÓI DOKUMENTÁCIÓ: a felhasználók számára készített útmutató, melyben konkrét példákon keresztül történik a program használatának, beállításainak a bemutatása,
6. fejezet ÖSSZEGZÉS: melyben a rendszerfejlesztés, tesztelés és bevezetés tapasztalatait összegzem, és ezek alapján vonok le alapvető következtetéseket a projektre vonatkozóan, illetve a jövőbeni fejlesztésekre nézve.

### 1.4. A rendszerfejlesztés módszerei és eszközei

Alapvetően Objektum Orientált (a továbbiakban OO) rendszerfejlesztési módszertan szerint történt a fejlesztés. A tervezés teljes folyamatában az UML[2]-t, mint szabványos modellező nyelvet használtam fel. Az UML alapvető filozófiája az OO szemléletű fejlesztés elveire támaszkodik, így osztályokat, objektumokat, metódusokat kezel, tehát kiválóan alkalmas egy teljesen OO fejlesztési környezetben implementált rendszer modellezésére.

Mivel a teljes elosztott rendszer két alapvető rendszer-architektúra egysége, ezért a különböző architektúrák fejlesztői eszközeit külön-külön adom meg:

- a kétrétegű kliens-szerver program fejlesztői eszközei:
    - Borland Delphi Enterprise Edition 5.0 (Build 6.18, Update Pack 1),
        - Oracle Data Access Components 6.05 Standard Edition komponenscsalád,
        - Developer Express komponensgyűjtemény,
    - InstallShield Express 2.12 (a telepítő-rendszerhez),
- a háromrétegű kliens-szerver web-alkalmazás fejlesztéséhez használt szoftverkomponensek:
    - HTML 3
    - PHP 4.4.4 Release
    - Apache 2.0 HTTP Server

[2] **U**nified **M**odelling **L**anguage = Egyesített (szabványos) Modellező Nyelv.

Mindkét architektúra tervezése és fejlesztése projekt-alapú és a termékorientált életciklust alkalmazza, mint alapvető modellt. Az ORACLE 10gXE relációs adatbázis-kezelő rendszerben történt (történik) az adatbázis implementálása és menedzselése.

Az adatbázis programozásához – PLSQL eszközként – az **SQLDeveloper** 1.0.0.15.57 (Build MAIN-15.57) verziót használom. CASE eszközként a **DELPHI IDE**[3] beépített lehetőségeit és a **Visio 2000** 6.0.1245v rendszert alkalmaztam.

### 1.5. A rendszer továbbfejlesztésének lehetőségei

A teljes elosztott rendszerben elsősorban a webes szegmens továbbfejlesztése lehet jó célkitűzés. A dinamikus oldalak számának- és az ilyen oldalak intelligenciájának növelésében látok további lehetőségeket. Konkrétan ez azt jelenti, hogy tervezek például egy HelpDesk[4] szolgáltatást, melynek alapvető feladata a felhasználók és a fejlesztés közötti gyors és pontosan dokumentált kommunikáció megteremtése. A következő releváns feladatokat emelem ki a szolgáltatáson belül:

1. az adott felhasználó e-mail címének megadásával választ egy kategóriát,
    a. segítségkérés,
    b. észrevétel, javaslat,
    c. hibabejelentés,
2. megadja a bejelentés tárgyát, azaz röviden leírja bejelentése lényegét, majd
3. egy hosszabb szöveges leírásban kifejti mondandóját.

Az adatokat rögzítjük az adatbázisban és a felhasználó kap egy nyugtázást bejelentéséről a megadott levelezési címre. A bejelentés elemzése után szöveges formában tájékoztatást kap az adott probléma kezeléséről, lehetséges megoldási módjairól.

A bejelentés végleges lezárására akkor kerülhet sor, amikor az adott probléma végleges megoldást nyer. Ekkor újabb tájékoztatást kap a bejelentő és amennyiben szükséges a probléma megoldása felkerül a webes felület külön erre a célra létrehozott szekciójába, (VINFO 2007)[5], ahol a rendszer újdonságai és dokumentációja is megtalálható.

Megjegyzés: *a felhasználóknak az Intranetes felületen jelenleg is módjuk van a rendszergazdának közvetlenül e-mailt küldeni, amelyben leírhatják észrevételeiket, segítséget kérhetnek.*

A kétrétegű kliens-szerver program továbbfejlesztési lehetőségei között a vezetői modul bővítését emelem ki. Ebben a modulban a régió felső vezetése számára vannak alapvető statisztikai információk. Ez az un. VIM (Vezetői Információs Modul) olyan technológiával készült – a teljes rendszerrel összhangban – amely magában hordozza a bővítés lehetőségét. Ilyen bővítési lehetőségek például:

- a meglévő elemzések szofisztikált statisztikai eszközökkel történő kiegészítése[6], vagy
- az adatokhoz kapcsolódó grafikonok körének bővítése.

---

[3] IDE = **I**ntegrated **D**evelopment **E**nviroment, aza Integrált Fejlesztői Környezet.

[4] A rendszer használata közben felmerült kérdések, javaslatok és esetleges hibák bejelentésének és kezelésének az eszköze. Egyfajta segítségnyújtás a felhasználók számára, melynek gyors és jól dokumentált módja ez a webes (Intranetes) megoldás.

[5] Ez a szekció a főoldalról is elérhető a VINFO 2007 linkre kattintva, de a felhasználók közvetlenül is hozzáférhetnek a kétrétegű kliens-szerver program súgó menüjéből is.

[6] Ehhez az alkalmazott ORACLE 10gXE RDBMS is tartalmaz beépített statisztikai függvényeket.

# 2. fejezet FELADATSPECIFIKÁCIÓ

## 2.1 Egy valós probléma és annak háttere

Az Allianz Hungária Biztosító Észak-alföldi Igazgatóságának menedzsmentje részéről már a régió megalakulását[7] követően felmerült az igény, hogy a gépjármű kárrendezés területén szükséges lenne a területet legjobban jellemző adatok napi változásainak követése.

Arra is komoly igény mutatkozott, hogy ezen a területen dolgozó munkatársak munkahatékonyságát a korábbiaknál pontosabban megismerhessék és a szükséges változtatásokat az új ismeretek tükrében meghozzák.

Mivel a kárrendezés hatékonysága jelentős tényező az üzleti hatékonyságban és az ügyfél orientációban is, ezért a rendszer adatbázisára épülő naprakész adatok, statisztikák mindkét említett tényezőt pozitívan kell hogy befolyásolják.

## 2.2 Megoldandó feladatok definiálása (követelményspecifikáció)

A modellszemléletű vállalatcentrikus rendszerfejlesztési folyamat első lépéseként a valós rendszer működését kell megérteni. Ezen ismeretek birtokában állítható fel a CIM[8]-modell. Ennek a modellnek a specifikációja a felhasználókkal és a vezetőkkel történt egyeztetések írásos dokumentációja.

Megjegyzés: *A korszerű rendszerfejlesztési módszertanok középpontjában a FELHASZNÁLÓ- és az igényei állnak. Ennek szellemében törekedtem arra, hogy minden egyes vezető és beosztott munkatárs elmondhassa véleményét az elkészítendő rendszerrel kapcsolatban. Ennek formális megvalósítása projektértekezletek megtartása volt. Minden egyes összejövetelről egy projektdokumentum készült, melynek tartalma szolgáltatta a követelményspecifikáció pontos tartalmának az alapját.*

*A projektértekezletek alkalmasak – az igényfelmérésen túl – a következő szempontok érvényesítésére:*

- *egyrészt növeli a vezetők elkötelezettségét a rendszer iránt, másrészt*
- *csökkenti a felhasználói kör ellenállását az új rendszer bevezetésével szemben.*

A CIM-modell termékeként tehát létrejött egy írásos anyag, amelyben az elkészítendő rendszerrel kapcsolatos követelményeket foglaltam össze. A követelményelemzés fundamentális eleme a jó kérdésekben és az azokra adott tömör válaszokban rejlik. A következő bekezdésben egyetlen tömör kérdésre válaszolva definiálom a rendszerrel szemben támasztott alapvető követelményeket.

Milyen alapadatok jellemzik a kárrendezési területet a legjobban? A kérdés megválaszolását első közelítésben globális szempontok alapján szegmentáltuk, és a következő – funkciókhoz is közvetlenül hozzárendelhető – igényeket fogalmaztunk meg:

[7] Az Igazgatóság 10 évvel ezelőtt jött létre három megyét integrálva. Ez a három megye: Hajdú-Bihar, Szabolcs-Szatmár-Bereg és Jász-Nagykun-Szolnok. A kárrendezés területén az idén egy kilenc megyére kiterjedő kárrendezési centrum felállítása van folyamatban. Ez a rendszer kiterjesztését is jelenti kilenc megyére!

[8] Computational Independent Model. Ez a modell még teljesen független a rendszer-környezettől, a különböző platformoktól.

adjon a rendszer tömör és pontos információkat a kárügyekről, a kárügyekben megképzett tartalékokról[9] tehát,…

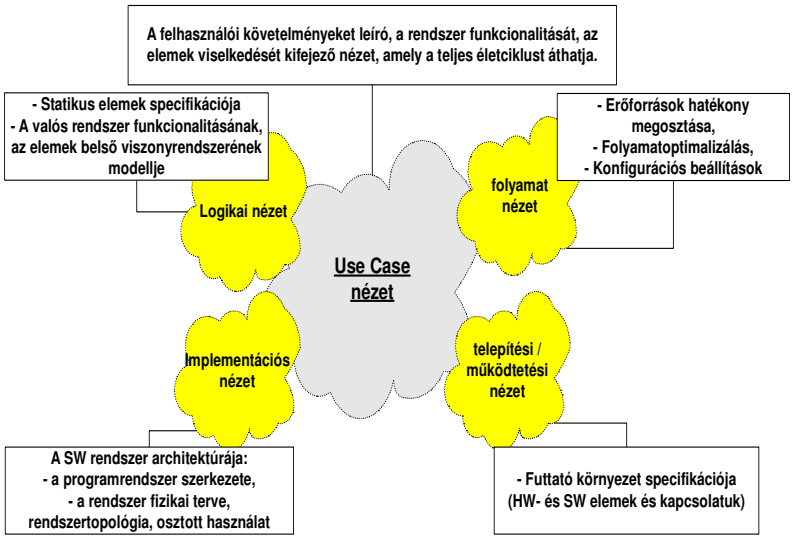
a. a rendszer csak a legszükségesebb – az adott kárhoz és nem az ügyfélhez kapcsolódó – általános adatokat rögzítse az adatbázisban (kárszám, rendszám, módozat),
b. szolgáltasson információt arról, hogy milyen pontosan történt a tartalékképzés (túlképzés, alulképzés),
c. mely szakértő képezte a tartalékot, és mely kárügyintézőhöz tartozik az adott kárügy (beazonosítható felelősség),
d. legyen vizsgálható egy adott időszak összes kárügye,
e. teremtse meg a rendszer a kártípusonkénti (töréskár, lopáskár, totálkár, stb.) kárelemzés lehetőségét,
f. adjon olyan – az egyes kárügyintézők munkáját – mind kvantitatív, mind kvalitatív szempontból jól jellemző információkat, amelyek alapján megítélhető az adott munkatárs leterheltsége és munkavégzésének minősége,
g. legyen naprakészen nyomon követhető minden egyes kárügy, azaz milyen intézkedések történtek az adott kárban, milyen állapotban van (nyitott, zárásra vár vagy lezárt) és kapjunk tájékoztatást arról, hogy mennyi idő alatt történt meg a kár teljes rendezése, lezárása

2. szükséges a kifizetésekről tételes (partnerenkénti) és globális (megyénkénti és régiós) adatelemzés elkészítése, azaz…
   a. a partnerenkénti kifizetéseknél legyen az átlagos- és kiugróan magas (maximális) kifizetés is megtekinthető,
   b. a megyénkénti- és régiós kifizetés-elemzésnél jelenjenek meg az összes kifizetések mellet az átlagok és a kiugróan magas összegek,
   c. fontos kritérium, hogy a kifizetések partnerenkénti csoportosításban is legyenek elkészítve, illetve legyen lehetőség egy adott időszak vizsgálatára,
3. legyen a rendszerbe integrálva szerződött partnereink díjbevételi teljesítménye, azaz legyen összevethető ezen partnereknek (autószalonoknak, javítóknak) tett kifizetések és az általuk kötött biztosítások díjbevételeinek az összege, vagyis…
   a. fontos szempont, hogy a megkötött alapmódozatokhoz (CASCO, GFB) külön legyen rögzítve a hozzá tartozó kiegészítő biztosítások köre,
   b. itt is alapvető követelmény a díjbevételek adott dátumtartományon belüli vizsgálata, illetve az, hogy a kiegészítő módozatok díjbevételei külön is elemezhetőek legyenek,
4. teremtse meg a rendszer a vezetői információk gyors közzétételének a lehetőségét, azaz…
   a. a régió vezetése az első három pontban definiált funkciókhoz rendelt adatokat könnyen kezelhető, áttekinthető formában tekinthesse meg,
   b. ehhez kapcsolódóan biztosítani kell a vezetők által szükségesnek tartott intézkedések, utasítások közzétételét is,
5. biztosítsa a program-rendszer azt, hogy minden felhasználó csak a rá tartozó adatokhoz férjen hozzá, csak ezeken legyen képes műveleteket végezni és ezen műveletek köre is legyen pontosan meghatározva.

[9] Minden egyes kárügyben kötelező egy pénzügyi tartalékot képezni, melyet elkülönülten kell kezelni és ez képezi az adott kár fedezetét.

## 2.3. Konceptuális tervezés

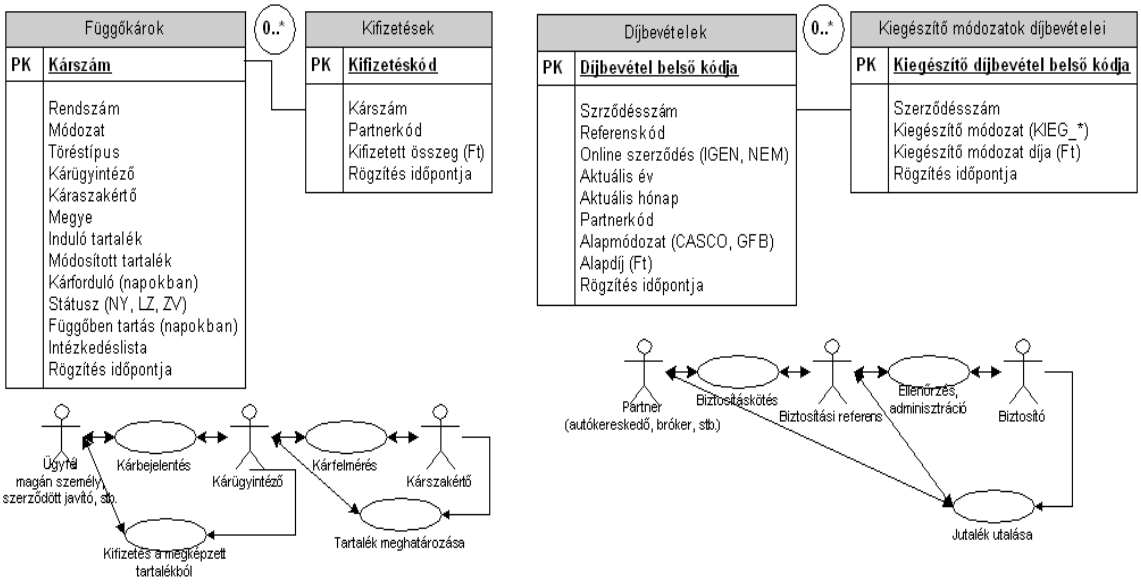
A konceptuális tervezés fő szegmenseit az UML nyelv négy alapvető nézetén keresztül mutatom be.




**2.1. ábra. Az UML nyelv alapvető nézetei.**

- **Logikai nézet (Logical view):**

A követelményspecifikáció elemzése kapcsán jól látható, hogy a rendszernek négy alapvető aktív osztálya van, melyeket a 2.2. ábra szemléltet.




**2.2. ábra. Függőkárok, Kifizetések, Díjbevételek és Kiegészítő díjak aktív osztályai.**

A 2.2. ábrából leolvasható az alapvető osztályokhoz kapcsolódó szereplők és feladatok (Use case) nézete. Ez a nézet a rendszer alapvető viselkedését, funkcionalitását fejezi ki, ahogyan azt a felhasználók, vezetők, a validálást végző munkatársak és a rendszerfejlesztő látja, azaz olyan elemeket ábrázol, amelyek a funkcionalitás megvalósításához szükségesek.

Az Ügyfél, Kárügyintéző, Kárszakértő, Partner, Biztosítási referens vagy a Biztosító olyan személy vagy elem, amely kapcsolatban van a rendszerrel, és generálja, kiváltja vagy végrehajtja annak funkcióit. Milyen – átfogóan is érvényes funkciók – valósítják meg a rendszerrel szemben támasztott követelményeket? Erre a kérdésre a folyamat nézet elemzésével lehet megadni a választ.

- **Folyamat nézet (Process view):**

Ebben az absztrakt nézetben a rendszert folyamataira (metódusaira) és annak végrehajtó elemeire bontom. Ez lényegében a teljes elosztott-rendszer funkcionális sajátosságainak a feltárását jelenti azzal a céllal, hogy a paralel végezhető műveletek ismeretében hatékony erőforrás-gazdálkodás legyen megvalósítható.

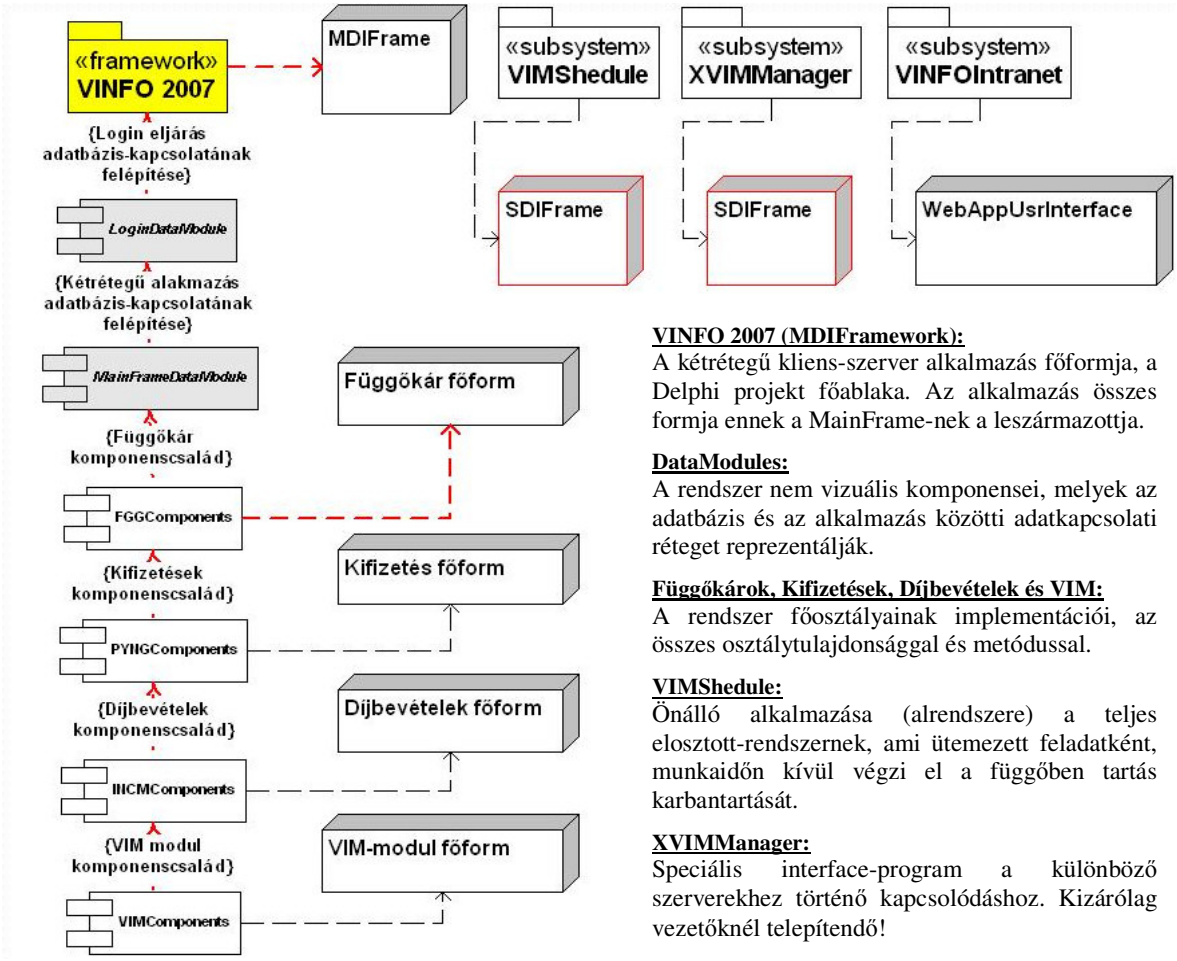
Csak a Függőkár osztály elemzését végzem el, mert a további alapvető osztályokra örökíthető[10] a metódusok többsége. Az OO rendszerfejlesztésben ugyanakkor érvényes egy másik lényeges tulajdonság a polimorfizmus. Ennek a tulajdonságnak köszönhető, hogy a származtatott objektumban új – az ősben nem szereplő – tulajdonságok és metódusok jelenhetnek meg. Ennek szellemében a többi objektum esetében csak az új tulajdonságokat, metódusokat definiálom.

| Folyamat (metódus) megnevezése | Végrehajtó elem | Paralel végrehajtás |
|---|---|---|
| 1. **Felvitel** (módosított tartalék = induló tartalék, függőben tartás = 30 nap, kárforduló = 0 nap) | Tárolt eljárás | IGEN |
| 2. **Módosítás** (függőben tartás = 30 nap intézkedés esetén) | Tárolt eljárás | IGEN |
| 3. **Törlés** (csak akkor, ha nem sérti meg a referenciális integritást!) | Tárolt eljárás | IGEN |
| 4. **Tartalékmódosítás** (automatikus, vagy manuális) | Tárolt eljárás | IGEN |
| 5. **Lezárás** (módosított tartalék = 0, függőben tartás = 0 nap, kárforduló számítása a rögzítés dátumából és a rendszerdátumból) | Tárolt eljárás | IGEN |
| 6. **Megnyitás** (módosított tartalék = induló tartalék, függőben tartás = 30 nap) | Tárolt eljárás | IGEN |
| 7. **Kifizetés** (módosított tartalék – kifizetett összeg, kevés tartalék esetén automatikus vagy manuális tartalékmódosítás meghívása) | Tárolt eljárás | IGEN |
| 8. **Kifizetés módosítása** (módosított tartalék megfelelő karbantartása) | Tárolt eljárás | IGEN |
| 9. **Kifizetés törlése** (módosított tartalék + kifizetett összeg) | Tárolt eljárás | IGEN |
| 10. **Gyorskeresés** (adatok gyorskereső metódusa) | Tárolt eljárás | IGEN |
| 11. **Rendezés** (adatok rendezése növekvő vagy csökkenő sorrendben, egymásba ágyazható módon) | Tárolt eljárás | IGEN |
| 12. **Csoportképzés** (egymásba ágyazható módon!) | Tárolt eljárás | IGEN |
| 13. **Szűrés** (több – egymással logikai ÉS kapcsolatban lévő – feltétel szerint!) | Tárolt eljárás | IGEN |
| 14. **Dátumtartomány szűrés** (a rögzítés időpontja szerint, csak az adott tartományba eső rekordok láthatók) | Tárolt eljárás | IGEN |
| 15. **Dinamikus listakészítés** (a felhasználó által a program futása közben rendezett, megfelelően szűrt és szelektált adatok listázása, szerkeszthető nyomtatási képpel) | Tárolt eljárás | IGEN |
| 16. **Statisztikai elemzések** (tartalékelemzés, kifizetés-elemzés, leterheltség vizsgálat) | Tárolt eljárás | IGEN |
| 17. **Adatexport** (.XLS, .HTML és .XML formátumban, VIM[11] garfikonok esetén meta-fájl és .BMP formátumban is!) | Tárolt eljárás | IGEN |
| 18. **Exportált adatok közzététele az intraneten** | Globális jogokkal mentés | IGEN |

[10] Az OO fejlesztés egyik alapvető tulajdonsága az öröklődés (inheritance), azaz a szülő- vagy ősobjektum tulajdonságai és metódusai megjelennek a gyerekobjektumban.
[11] VIM = **V**ezetői **I**nformációs **M**odul.

A Kifizetések aktív osztályra a 4-6 pontok kivételével minden definíció érvényes. A díjbevételeknél a 4-9 pontok értelemszerűen nem szükségesek, a többi metódust viszont beépítettem a rendszerbe.

- **Implementációs vagy komponens nézet (Implementation view):**




**VINFO 2007 (MDIFramework):**
A kétrétegű kliens-szerver alkalmazás főformja, a Delphi projekt főablaka. Az alkalmazás összes formja ennek a MainFrame-nek a leszármazottja.

**DataModules:**
A rendszer nem vizuális komponensei, melyek az adatbázis és az alkalmazás közötti adatkapcsolati réteget reprezentálják.

**Függőkárok, Kifizetések, Díjbevételek és VIM:**
A rendszer főosztályainak implementációi, az összes osztálytulajdonsággal és metódussal.

**VIMShedule:**
Önálló alkalmazása (alrendszere) a teljes elosztott-rendszernek, ami ütemezett feladatként, munkaidőn kívül végzi el a függőben tartás karbantartását.

**XVIMManager:**
Speciális interface-program a különböző szerverekhez történő kapcsolódáshoz. Kizárólag vezetőknél telepítendő!

**2.3. ábra. A VINFO 2007 elosztott rendszer komponensdiagramja.**

A **VINFOIntranet (subsystem)** a teljes elosztott-rendszer egy alrendszere, amely lényegében egy háromrétegű webes alkalmazás. Ennek az alrendszernek a következő feladatokat kell ellátni:

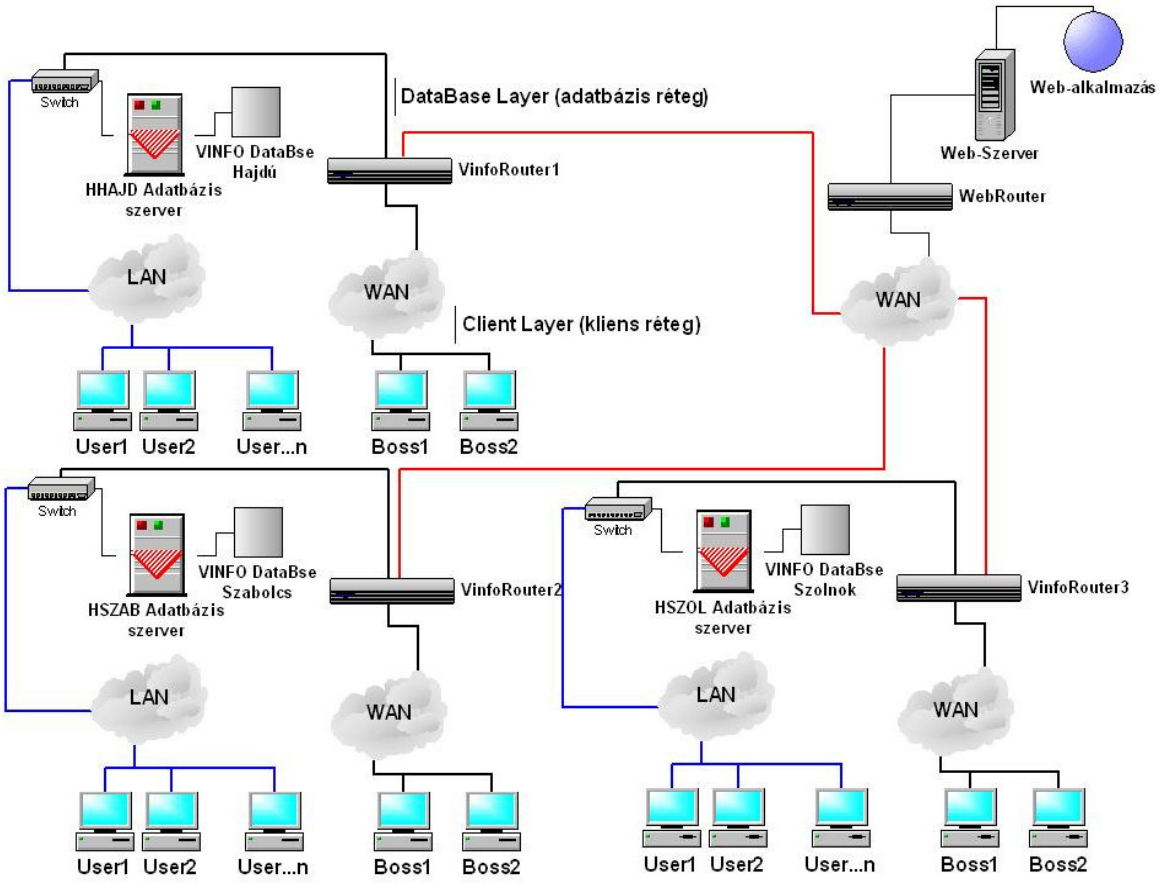
- biztonságos és gyors adatmegjelenítés, ami elsősorban az adatbázisból – a kétrétegű kliens-szerver alkalmazáson keresztül – kinyert adatok gyors közzétételét szolgálja,
- az adminisztrációs és dokumentációs tevékenységeket is összefogja, és
- a további fejlesztések eredményeként szabványos[12] felületet nyújt a felhasználók számára, hogy a kérdéseikkel, javaslataikkal esetleges hibabejelentéseikkel közvetlenül a fejlesztővel vegyék fel a kapcsolatot.

[12] A HelpDesk komponens beépítéséről van itt szó. Ez a funkció meggyorsítja a rendszerrel kapcsolatos kommunikációt és ezzel végső soron hozzájárul a hatékony munkavégzéshez.

## 2.4. Rendszerkörnyezet és rendszer-architektúra tervezés

- **Telepítési vagy működtetési nézet (Implementation view):**

Ebben a nézetben elsőként a hálózati- és hardvertopológiát fejtem ki, ami meghatározza az egyes rendszerkomponensek helyét és szerepét, kapcsolatuk módját.




**2.4. ábra. A VINFO 2007 elosztott-rendszer hálózati topológiája.**

### 2.4.1. Elosztott rendszermodell

A rendszerrel szemben támasztott követelmények között szerepel, hogy az egyes megyék adatait elkülönülten kell kezelni és egy adott megyén belüli kárügyintézők, helyi vezetők ne láthassák a többi megye adatait. Az is elvárás, hogy minden kárügyintéző csak a saját káradatait láthassa és kezelhesse.

Külön problémát jelentett az, hogy sem MainFrame, sem MiniFrame szerverek vásárlására nem volt lehetőség. Ilyen feltételek mellett kellet megoldani, hogy a rendszer a szerverekkel történő kommunikáció tekintetében gyors legyen, de a vezetés is megfelelő információkhoz jusson az egyes megyék adataiból, illetve a teljes régióra vonatkozóan is pontos képet lásson.

A 2.4. ábrán vázolt elosztott-rendszermodell kialakítása az adott premisszák mellett minden felhasználó számára azt tudja nyújtani, amit elvár a rendszertől. Tehát, milyen elvárásokat kell ennek a modellnek teljesíteni?

Az elosztott-rendszerek fő sajátossága, hogy a felhasználói kör számára a teljes rendszer bonyolultsága, a teljes működési mechanizmus el van fedve, mindenki egyetlen rendszerként, egyetlen programként érzékeli a teljes rendszert. Hogyan valósul ez meg a gyakorlatban?

1. Az egyes megyék kárügyintézői és helyi vezetői a saját szerverük adatbázisába viszik fel az adatokat. Ezt a szervert a helyi számítógép-hálózaton (LAN[13]-on) keresztül érik el. A 2.4. ábrán kék színnel kiemelt hálózati szegmensben a kommunikáció rendkívül gyors,
    a. hiszen a szerverrel történő adatcsere során nincs routing, azaz nem kell egy távoli szervert – esetleg több router közbeiktatásával – elérni,
    b. nem jelentkeznek az adatkommunikációs vonalak sávszélesség problémái,
    c. sokkal kisebb az esélye bármilyen hálózati problémának, ami gátolná a folyamatos munkavégzést,
    d. a szerverre bekonnektált felhasználók száma optimalizálható,
    e. ugyanakkor az a vezetői elvárás, miszerint az egyes megyék adatait szeparáltan kell kezelni teljesen megoldott.
2. A 2.4. ábrán fekete színnel kiemelt hálózati (WAN[14]) szegmensben a felhasználók száma korlátozott. Az esetek többségében ezt a hálózati szegmenset csak a régiós vezetők, elemzők (néhány fő), illetve a rendszergazda használja. Ennek következtében a helyi szerverek távoli elérése nem jelent lassulást a lokális felhasználók számára és megfelelően gyors a vezetőknek, elemzőknek.
3. A Web-Szerver a régióközpontban egy 20 Mbps- sávszélességgel rendelkező optikai adatkommunikációs vonal végén csatlakozik a hálózatra. Ezt a hálózati szegmenset piros színnel jelöltem a 2.4. ábrában. Látható, hogy a lokális felhasználók két routeren (WAN-on) keresztül kommunikálnak a Web-Szerverrel, viszont ott csak korlátozott számú lapot tölthetnek le. Elsősorban a felhasználói dokumentációt érhetik el és a rendszerrel kapcsolatos aktuális információkat, utasításokat. A komolyabb adatforgalmat generáló adatexportokat, grafikonokat nem tudják letölteni, ezekhez az adatokhoz csak a régiós vezetésnek van hozzáférése. A régiós vezetők elemzők viszont a Web-Szerverrel azonos alhálózaton vannak így itt szintén optimális a kommunikáció szerver és kliens számítógép között. Mivel a régiós vezetés elsősorban az Intranet-re kiexportált információkra kíváncsi, mind a VINFO szerverek, mind a Web-Szerver kihasználása optimális.

### 2.4.2. A kétrétegű kliens-szerver modell szerepe

A lokális felhasználók ezen a kliens programon keresztül látják a teljes rendszert, természetesen csak olyan funkcionalitással, ahogyan azt a jogosultságuk lehetővé teszi. Az Intranet megfelelő oldalait is ebből a programból érik el, tehát az Ő számukra ez a program jeleníti meg a teljes rendszert, elfedve a különböző hálózati szegmenseket és szervereket. Elmondható tehát, hogy egyetlen felhasználói felületet GUI[15]-t kezelnek, így számukra lényegesen leegyszerűsödik a program használata.

[13] LAN = **L**ocal **A**rray **N**etwork, azaz helyi számítógép-hálózat.
[14] WAN = **W**ide **A**rray **N**etwork, azaz kiterjedt számítógép-hálózat. Például különböző városokban lévő telephelyek alhálózatainak összekapcsolásával hozható létre ilyen topológia.
[15] GUI = **G**raphical **U**ser **I**nterface, azaz grafikus felhasználói felület.

### 2.4.3. A háromrétegű kliens-szerver modell funkciója

A háromrétegű webes modell szerkezetileg a következő módon épül fel:

- maga az Apache2 http szerver és a webes alkalmazás egy számítógépen, a 2.4. ábrán Web-Szerver névvel ellátott szervergépen van telepítve,
- a modell legalsó rétege az un. vékony kliens réteg a felhasználói számítógépeken telepített web-böngésző program.

A webes modell lényegében egy saját Intranet, azaz olyan Internetes eszközökön és módszereken alapuló fejlesztés, ami kívülről, a biztosító ügyfeleinek az oldaláról nem érhető el. Ez érthető is, hiszen ennek a rendszernek az alapfeladatai között az is szerepel, hogy a kétrétegű kliens-szerver alkalmazásból kiexportált adatokat megjelenítse, tartalmazza a belső utasításokat, kezelje a belső kommunikációt. Ezek az adatok nem publikusak a biztosító ügyfelei számára.

### 2.4.4. A kettő- és háromrétegű architektúra kapcsolata

Minden felhasználó a kétrétegű kliens-szerver programon keresztül képes elérni és letölteni az Intranet keresett oldalait. Az ötszintű jogosultsági rendszer gondoskodik arról, hogy minden felhasználó a jogosultságának megfelelő hozzáféréssel rendelkezzen. A megfelelő menüpont kiválasztásával a program az alapértelmezett web-böngészőt indítja el, abban pontosan az a lap töltődik le a Web-Szerverről, amit megenged a felhasználó jogosultságkódja.

## 2.5. Platformok, környezeti paraméterek, rendszer-telepítés tervezés

A Biztosító Társaságnál általánosan bevezetett operációs rendszer a felhasználói számítógépeken a Windows XP Professional SP2 verzió. Szerver oldalon és így a fejlesztői számítógépek többségén ennek megfelelően a Windows 2003 szerver OP-rendszer használata terjedt el. A 32 bites OP-rendszereknek megfelelően a kétrétegű kliens-szerver program is 32 bites. Ezt a kétrétegű modellt egyetlen speciálisan elkészített **VNF7.EXE** program reprezentálja. A programhoz összesen egy alapértelmezett **SYSTEM** könyvtár tartozik, melyben a következő inicializációs állományok vannak elhelyezve:

- APPSYS.INI: a rendszer főablakának elmentett paramétereit tartalmazza,
- DBPRM.INI: a rendszer adatbázishoz történő kapcsolódásának legfontosabb paraméterei mellett tartalmazza az alapvető rendszerparaméterek szekcióját, valamint az adatexporthoz, és a Web-Szerver eléréséhez szükséges alapértelmezett útvonalakat; ebben a belső struktúrával rendelkező fájlban a fontosabb szekciók:
  - [DataBase]: az adatbázishoz történő kapcsolódás paramétereit definiálja
  - [SystemParameters]: rendszerparaméterek definíciója,
  - [SaveFggData]: alapértelmezett útvonalak, URL-ek,
- SYSPRM.INI: a 2.3. ábrán megadott főformok paramétereinek a mentését tartalmazza szekciókba rendezett módon.

Az .INI fájlok esetleges sérülése, véletlen törlése esetén maga a program újra generálja azokat és rendre létrehozza azok belső struktúráját. A VNF7.EXE program futás közben – az adott funkciók használatának megfelelően – olvassa vagy felülírja, létrehozza a megfelelő adatokat, természetesen szekció specifikus módon. A SYSTEM könyvtárra a felhasználói csoportnak módosítási joggal kell rendelkezni (hogy a program írni is tudjon azokba)!

Opcionálisan tartalmaz a rendszer egy SAVE könyvtárat a VNF7.EXE-vel azonos szinten. A rendszer alapértelmezésben ezt a könyvtárat ajánlja fel az exportálni kívánt adatok mentéséhez. Ez – a rendszer telepítése után – egy üres könyvtár. Az adatexportokat természetesen más útvonalon lévő könyvtárba is lehet menteni.

A VNF7.EXE program speciális technológiával készült, melynek következtében az adatbázishoz történő kapcsolódáshoz és a teljes hálózati kommunikáció lebonyolításához nem szükséges az ORACLE kliens program telepítése. Ez azt jelenti, hogy minden felhasználó számítógépén egyetlen program és egyetlen könyvtár (benne három .INI fájllal) telepíthető.

A rendszer karbantarthatóságát tehát a speciális fejlesztési technológia alkalmazása lényegesen leegyszerűsíti, hiszen egyetlen program frissítését kell szükség esetén elvégezni. Ez a feladat egy un. telepítő gateway[16] számítógép segítségével csoportosan végezhető el a számítógép-hálózaton keresztül, azaz nem szükséges helyi rendszergazda a frissítések menedzseléséhez.

A telepítő rendszer szintén egyetlen **VINFOINST.EXE** állományból áll. Ez a telepítő program a rendszer telepítő CD-jén van elhelyezve. Futtatásával a megadott rendszerkomponenseket a \Program Files\Vinfo7\ útvonalra telepíti fel. Célszerű ezt az útvonalat választani, de nem kötelező, ti. a VNF7.EXE bármely meghajtó bármely könyvtárából indítható. A program indítására szolgáló ikont a telepítő elhelyezi az adott felhasználó munkaasztalán.

A vezetői gépeken az említett VNF7.EXE programon és a SYTEM könyvtáron felül még egy program telepítése szükséges. A 2.3. ábrán XVIMManager névvel ábrázolt speciális konzol program, melynek futtatható állománya az XVIMManager.EXE.


**2.5. ábra. Az XVIMManager-konzol.**


**2.6. ábra. Konzol program GUI.**

A konzol programot célszerűen az alapértelmezett útvonalon – a VNF.EXE szintjén – lehet elhelyezni. A program indításának automatizálására célszerű az indító ikont a Windows Startmenüjének az Indítópultjában elhelyezni. Ezzel elérhető, hogy az OP-rendszer indulásakor a konzol program is elinduljon

A konzol program feladata, hogy előkészítse a kiválasztott szerverhez történő kapcsolódást. Ez alapvetően a DBPRM.INI fájl adatbázis-kapcsolódáshoz kötődő szekcióiban a szükséges paraméterek beállítását jelenti. A szerver kiválasztását követően a Start gombra kattintással a VNF.EXE indul el és kapcsolódik a megadott szerverhez.

Ebből következik, hogy a vezetők és elemzők is a kétrétegű kliens-szerver GUI-t kezelik, innen érnek el minden – jogosultságuknak megfelelő – funkciót. Tehát számukra látszólag csak a program indításában van különbség a lokális munkatársakkal szemben.

A 2.3. ábrán feltüntetett **VIMShedule** alrendszer egy (32 bites) **VIMShedule.EXE** nevű futtatható állomány, melynek telepítése kizárólag az adatbázis-szervereken szükséges! Ez a program ütemezett feladatként – munkaidőn kívül – indul el. Feladata a függőkárok (kezdetben 30 napos) függőben tartási paraméterének a karbantartása. A szervereken is a korábban ismertetett telepítési eljárások érvényesek azzal az eltéréssel, hogy itt VIMShedule.EXE is telepítésre kerül. A szerverek karbantartása viszont kizárólag az adatbázis adminisztrátor feladata!

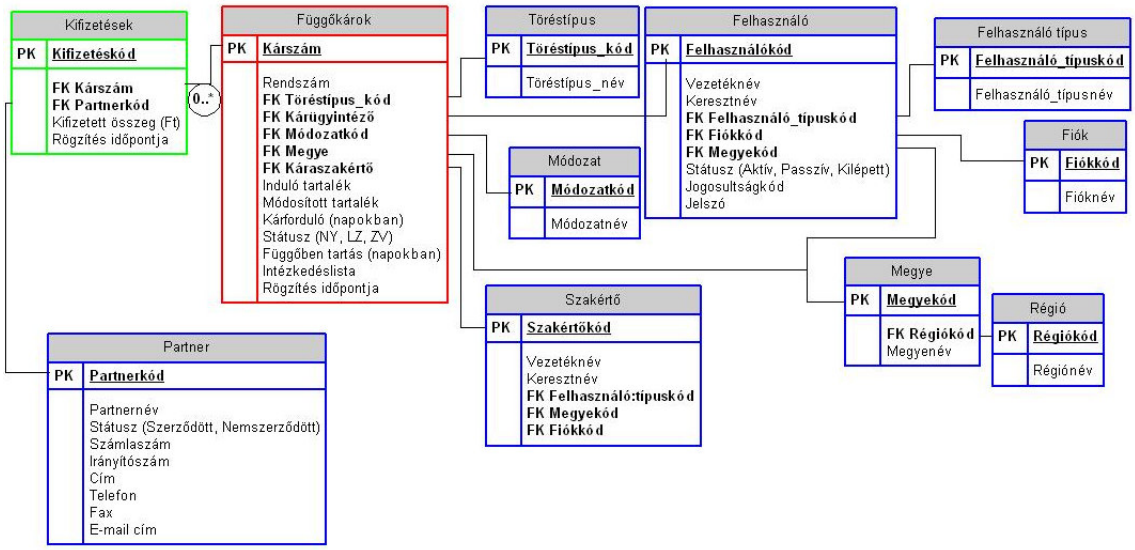
[16] A telepítő gateway egy olyan számítógép, amely a régió központban a nagy sávszélességű optikai kábel végén csatlakozik a számítógép-hálózatra. Ezen a számítógépen elhelyezve a program új verzióját az képes csoportosan, azaz egyszerre több felhasználói PC-n telepíteni a frissítéseket.

# 3. fejezet LOGIKAI RENDSZERTERVEZÉS

## 3.1. Az adatbázis globális architektúrája, RDBMS

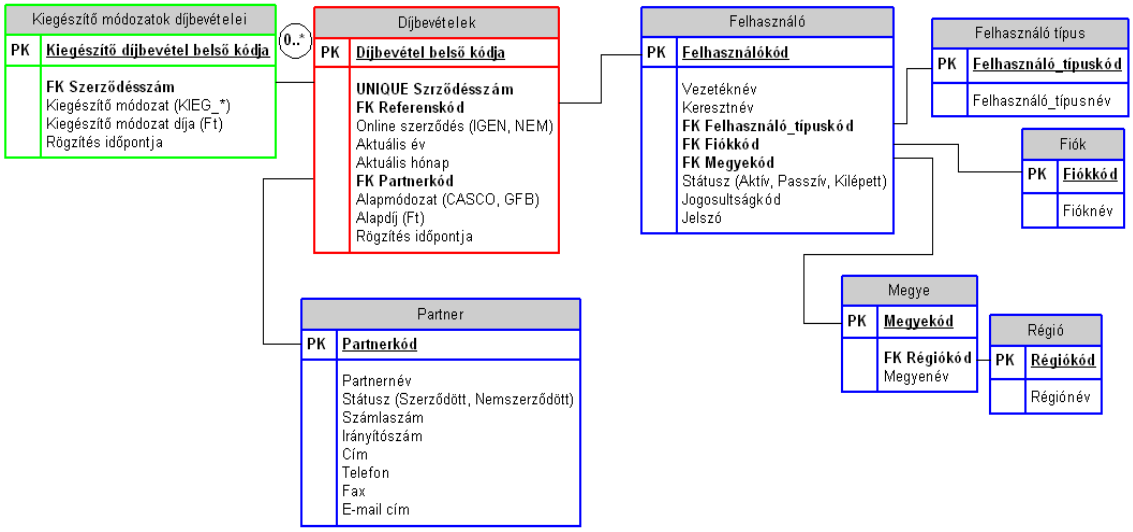
Az adatbázis struktúrájának, a benne definiált objektumok tulajdonságainak és kapcsolatainak az ismertetése során a négy fő objektumra, azaz a négy fő adatbázis táblára helyezem a hangsúlyt. Az adatbázisban létrehozott objektumok összes tulajdonsága, metódusa és kapcsolatrendszere SQL scriptek formájában készült el. Ez a módszer – többek között – lehetővé teszi, hogy a PLSQL nyelven implementált objektumokat illetve a teljes üzleti logikát, megfelelő csoportosításban elemezni lehessen. A teljes adatbázis-definíciót a következő SQL scriptek tartalmazzák:

- TABLE.SQL: állomány tartalmazza az adatbázis táblák definícióit, a táblákhoz kapcsolódó szekvenciákkal, megszorításokkal ( constraint-ekkel) és kommentekkel,
- TRIGGER.SQL: fájlban a triggerek SQL implementációi vannak,
- STRDPRC.SQL: fájl tartalmazza az adatbázis összes tárolt eljárását; ezen eljárások elnevezésénél a következő konvenciókat alkalmaztam:
  - INS_ prefixummal kezdődnek (és a kapcsolódó tábla nevével folytatódnak) azok az eljárások, amelyek az egyes táblákba új rekordot visznek fel,
  - UPD_ prefixummal kezdődnek (és a kapcsolódó tábla nevével folytatódnak) azok az eljárások, amelyek az egyes táblák megadott rekordját módosítják,
  - DEL_ prefixummal kezdődnek (és a kapcsolódó tábla nevével folytatódnak) azok az eljárások, amelyek az egyes táblák adott rekordját törlik,
  - ST1_, ST2, ST…_ prefixummal kezdődnek (és a kapcsolódó tábla nevével folytatódnak) azok az eljárások, amelyek a különböző statisztikákat állítják elő,
- VIEW.SQL: állományban helyeztem el az adatbázis nézeteinek a forrását, és az
- INSERT.SQL: scriptben azok az SQL-ben elkészített insert-ek vannak definiálva, amelyek már az adatbázis telepítése után felviszik a szükséges (alapértelmezett) rekordokat a megfelelő adatbázis táblákba.



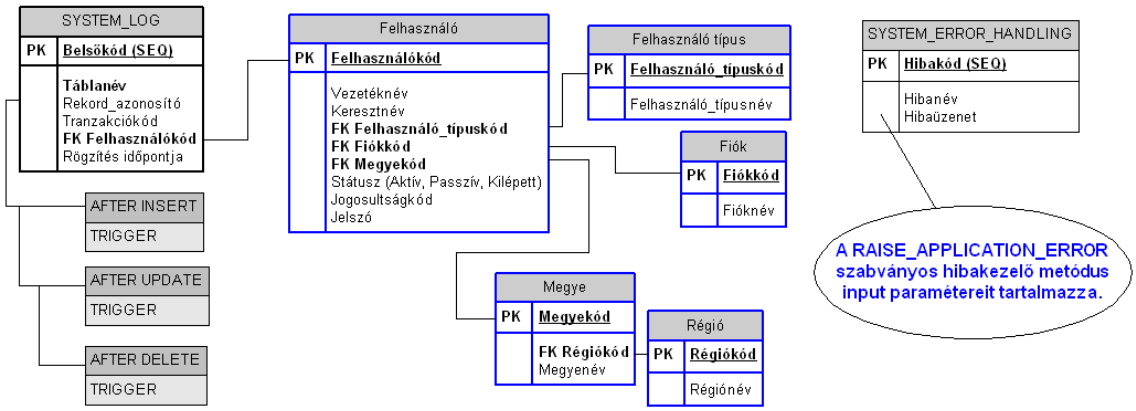

**3.1. ábra. A „Függőkárok" és „Kifizetések" táblák szerkezete és kapcsolatai.**

A 3.1. ábrán látható az adatbázis összes törzsadat-táblája (kék színnel kiemelve) és a „Függőkárok”- valamint a „Kifizetések” táblák és azok Master-Detail[17] kapcsolata.




**3.2. ábra. A „Díjbevételek” és „Kiegészítő módozatok díjbevételei” táblák szerkezete és kapcsolatai.**

A 3.2. ábra szemlélteti az adatbázis „Díjbevételek” és „Kiegészítő módozatok díjbevételei” táblákat, illetve azok kapcsolatát. A két tábla közötti „**0..*”** jel az opcionális többes kapcsolat UML specifikus ábrázolása, aminek a konkrét esetben az a jelentése, hogy a „Díjbevételek” tábla minden egyes rekordjához egy vagy több „Kiegészítő módozatok díjbevételei” rekord kapcsolódhat. Ez az összefüggés a „Függőkárok” és „Kifizetések” között is fennáll.




**3.3. ábra. Rendszertáblák szerkezete és kapcsolatai.**

A 3.3. ábrán feltüntetett „SYSTEM_LOG” táblából három példány van implementálva az adatbázisban. Ezek a rendszertáblák rendre a „Függőkárok”, a „Kifizetések” és a „Díjbevételek” főtáblákhoz tartoznak. Ezeknek a rendszertábláknak a segítségével valósul meg a tranzakcionált adatbázis-kezelés.

[17] Ebben az esetben a „Függőkárok” (Master) táblához kapcsolódik a „Kifizetések” (Detail) táblája.

Mindhárom főtáblához három speciálisan elkészített trigger[18] kapcsolódik, amelyek a rendszertáblákban rögzítik azt, hogy az adott főtáblán műveletet végrehajtó felhasználó, milyen módosításokat (felvitelt, módosítást, törlést) hajtott végre. Ezek a speciális metódusok, un. sorszintű triggerek, tehát a megfelelő tábla rekordjaira érvényesek.

A sorszintű triggerek alkalmazásával a rendszertáblákban nyomon követhető minden egyes felhasználó munkája, azaz látható, hogy mely főtáblákon milyen műveleteket végzett, ezek a műveletek (felvitel, módosítás, törlés) milyen rekordokhoz kapcsolódtak és pontosan rögzítve van az adott művelet elvégzésének időpontja is. A „SYSTEM_LOG" táblák bevezetésével például elérhető, hogy bármely művelet visszafejthető legyen, amennyiben az szükséges, de alkalmas a munkavégzéssel kapcsolatos vitás esetek egyértelmű tisztázására is. Minden rendszertábla kizárólag adatbázis-adminisztrátori joggal érhető el!

Az adatbázisban a „Függőkárok", a „Kifizetések" és a „Díjbevételek" főtáblákhoz a sorszintű triggereken kívül hozzá vannak rendelve un. táblaszintű triggerek, amelyek szerepe az, hogy ezen főtáblákat csak munkaidőben lehessen módosítani. A táblaszintű triggerek tehát, úgy vannak elkészítve, hogy – a rendszerdátum függvényében – csak $6^{00}$-$18^{00}$ időintervallumban engedélyezik a főtáblák bármilyen módosítását. Ennek a szabályozásnak a bevezetésére azért volt szükség, hogy a munkaidőn kívüli időszakban el lehessen végezni az adatok karbantartását és mentését. Ilyenkor fut le például a VIMShedule.EXE (lásd 2.3. ábra) ütemezett feladat, melynek funkciója a függőkárok adatainak karbantartása.

Az adatbázis része további két temporális tábla. Mindkét táblában – átmeneti jelleggel – statisztikai adatok tárolása történik. A statisztikai adatokat generáló tárolt eljárások használják ezt a két táblát. Ezeknek a tábláknak a tartalmát csak a vezetők és elemzők érhetik el, tekinthetik meg.

### 3.1.1. Az adatbázis tervezés fő szegmensei

Az adatbázis tervezés – a követelményspecifikáció alapján – a következő három szakaszra bontható:

- funkcionális függőségek elemzése,
- redundancia eliminálása, azaz normalizálás (legalább 3NF elérése),
- a teljes adatbázis integritását biztosító megoldások áttekintése.

Az adatbázis fő objektumai a „Függőkárok", és „Kifizetések" valamint a „Díjadatok" és „Kiegészítő módozatok díjadatai" táblák, és a hozzájuk kapcsolódó, azokat input adatokkal kiszolgáló adatbázis objektumok:

- a törzsadatokat[19] tartalmazó adatbázis táblák.

Az adatbázis tervezés folyamatában elsőként a funkcionális függőségek meghatározására koncentráltam. Az adatok közötti funkcionális függőségek, azok természetéből következnek, tehát a kiinduló (még normalizálatlan) adatokat első lépésben alapos elemzésnek kell alávetni.

---

[18] Az adott adatbázis tábla INSERT, UPDATE és DELETE eseményeinek bekövetkezéséhez kötött SQL eljárások. Ezek az ütemezett végrehajtású metódusok a fenti három esemény előtt (BEFORE) és utána (AFTER) is végrehajthatódnak, amennyiben megfelelő módon definiálva vannak.

[19] A törzsadatok a normalizálás során definiálható szegmensét képezik az adatbázisnak. Ezek az adatok a normalizálás folyamatában jól definiált táblákba szervezhetők és a főtábláknak szolgáltatnak bemenő adatokat egy új rekord felvitele-, illetve egy kiválasztott rekord módosítása során. A törzsadat-táblák közös jellemzője, hogy tartalmuk ritkán igényel karbantartást.

Az adatbázis logikai tervezése egy folyamat, amelyben az egyes lépések egymással szoros összefüggésben állnak. Így a funkcionális függőségek meghatározásával lényegében párhuzamosan a relációk kulcsairól is dönteni lehet.

A következő feladat a redundancia kiküszöbölése, ami ráépül a függőségek és reláció kulcsok meghatározása közben szerzett ismeretekre. A redundancia eliminációja természetesen egy újabb folyamat, amelynek jól definiált lépései vannak.

### 3.1.2. Funkcionális függőségek

Az adatbázis logikai tervezését konkrét példákon keresztül mutatom be. Az adatelemzés első lépése tehát a funkcionális függőségek feltárásával kezdődik. Az adatok között akkor áll fenn funkcionális függőség, ha egy vagy több adat konkrét értékéből más adatok egyértelműen következnek. Nézzük meg ezt a relációs algebra formalizmusával leírva:

FÜGGŐKÁROK = {Kárszám, Rendszám, Módozat, …, Töréstípus, Módosított tartalék}.[20]

Megjegyzés: *A „Kárszám" attribútum értéke egy központilag elkészített teljesen egyedi adat. Ezzel az egyedi attribútummal történik minden függőkár azonosítása. Ezeket a kárszámokat egy központi rendszer automatikusan generálja.*

A „Kárszám" tehát funkcionálisan meghatározza a többi attribútumot. Ez a függőség ráadásul teljes[21]. Ebből következik, hogy a „FÜGGŐKÁROK" relációnak a „Kárszám" egy kulcsjelöltje.

### 3.1.3. Relációk kulcsai

A reláció – definíció szerint – nem tartalmazhat két azonos sort, ezért minden relációban létezik kulcs. A reláció kulcsa a reláció attribútumainak nem üres halmaza. Milyen szerepe van ennek a kulcsnak?

A reláció kulcsa a reláció egy sorát azonosítja egyértelműen. A következő feltételeknek kell, hogy eleget tegyen a reláció kulcs:

- az attribútumok egy olyan csoportja, amelyek együttesen csak egy sort azonosítanak, ez az egyértelműségi kritérium,
- a kulcsban szereplő attribútumok egyetlen részhalmaza sem alkothat kulcsot,
- a kulcsban szereplő bármely attribútum értéke nem lehet definiálatlan (NULL).

A reláció kulcsára vonatkozó kritériumokból nyilvánvaló, hogy a relációnak lehet un. összetett- (több attribútumból álló) és egyszerű (egy attribútumból álló) kulcsa. A „FÜGGŐKÁROK" relációnak egyszerű kulcsa van.

Megjegyzés: *Általában is törekedtem arra, hogy egyszerű kulcsokat definiáljak minden relációhoz. Ennek a megoldásnak konkrét előnye például, hogy az így előállított reláció kulcs indexe is egyszerű, ami a kereső funkciókat jelentősen gyorsíthatja.*

Azokban a relációkban, ahol csak több attribútumból álló összetett kulcsjelöltek voltak a reláció egy sorának kulcsaként egy un. szekvenciát definiáltam. Erre a megoldásra jó példa:

KIFIZETÉSEK = {Kifizetéskód, Kárszám, Partnerkód, Kifizetett összeg, Rögzítés időpontja}.

---

[20] A funkcionális függőség baloldalát a függőség meghatározójának nevezzük. Nem áll fenn a funkcionális függőség, ha a meghatározó egy értékét a jobboldalon több attribútummal hozhatjuk összefüggésbe.

[21] A funkcionális függőség bal oldalán több attribútum is megjelenhet, melyek együttesen határozzák meg a jobb oldalon szereplő attribútum(ok) értékét. Abban az esetben viszont, amikor a baloldal egyetlen attribútuma a jobboldal minden attribútumát meghatározza, teljes funkcionális függőségről beszélünk.

A „KIFIZETÉSEK” relációban a „Kárszám” nem lehet önmagában kulcs, mert egy kárügyben több kifizetés is történhet ugyanarra a kárszámra, a „Partnerkód” sem megfelelő, hiszen ugyanannak a partnernek egy kárügyben szintén lehet több kifizetése, és ugyanezen okok miatt a két attribútum összetett kulcsként sem alkalmazható.

Ebben az esetben a reláció minden sorához meg lehet generálni egy sorszámot, melynek értékét az adatbázis motor egy új sor beszúrásakor automatikusan állítja elő az előre definiált intervallumból, meghatározott lépésközzel.

### 3.1.4. Redundancia

Az adatbázis logikai tervezésének egyik fő célja a redundanciák megszüntetése.

Redundanciáról akkor beszélünk, ha valamely adatot (tényt), vagy a többi adatból levezethető (kalkulált) mennyiséget ismételten (többszörösen) tároljuk az adatbázisban.

A redundancia, a fölöslegesen lefoglalt tárolókapacitás mellett, komplikált adatbázis frissítési és karbantartási műveletekhez vezet, melyek könnyen az adatbázis inkonzisztenciáját okozhatják. Egy adatbázis akkor inkonzisztens, ha egymásnak ellentmondó adatokat tartalmaz.

Megjegyezés: *A gyakorlatban már a tervezés során az adatbázis műveletek gyorsítása érdekében esetleg redundáns attribútumokat is be szoktunk vezetni.*

A logikai tervezés célja egy redundancia mentes reláció-rendszer, azaz egy relációs adatbázis létrehozása. A reláció elmélet kidolgozott módszereket tartalmaz a redundancia megszüntetésére, az úgynevezett normálformák alkalmazásával.

A normálformák tervezése során a funkcionális függőség, valamint a reláció kulcs korábban definiált fogalmait fel kell használni. A normálformák elkészítése során leegyszerűsítve, olyan relációk létrehozása a cél, melyekben csak az adott reláció kulcsára vonatkozó megszorításokat teszünk.

Öt normálformát különböztetünk meg. Az egyes normálformák egymásra épülnek, a második normál formában lévő reláció első normál formában is van és a 3NF-ban lévő már 2NF-ban van, stb. A tervezés során a legmagasabb normálforma elérése a cél. A gyakorlatban azonban túlnyomó többségében megelégszünk a 3NF elérésével.

Az első három normálforma a funkcionális függőségekben található redundanciák megszüntetésére koncentrál, míg a negyedik és ötödik a többértékű függőségekből adódó redundanciák kiküszöbölését célozza.

A következő példa a redundancia megszüntetésének a folyamatát mutatja be a „Kifizetések” és „Partnerek” relációk felhasználásával. Ebben az esetben a relációt táblázatos formában adom meg.

| Kifizetéskód | Kárszám | Kifizetett összeg | **Partner** |
|---|---|---|---|
| 1 | 100000000 | 128.000 Ft | **Partnerkód** / **Partnernév**<br>*PRTNR1* / *Partner1 Kft.*<br>*PRTNR2* / *Partner2 Kft.* |
| 2 | 110000000 | 258.000 Ft | **Partnerkód** / **Partnernév**<br>*PRTNR3* / *Partner3 Kft.*<br>*PRTNR4* / *Partner4 Kft.* |

**3.4. ábra. A „Kifizetések” reláció normalizálatlan állapotban.**

A normálformák részletezését megelőzően két új fogalom bevezetésére van szükség a normalizálás teljes folyamatának megértéséhez:

1. elsődleges attribútumnak nevezzük a reláció azon attribútumait, melyek legalább egy reláció kulcsban szerepelnek, ugyanakkor
2. a többi attribútumot nem elsődlegesnek (másodlagosnak) nevezzük.

### 3.1.5. Normalizálás, normálformák

Egy reláció első normálformában van, ha minden attribútuma egyszerű, azaz nem összetett adat. A 3.3. ábra „Kifizetések" relációja nincs első normálformában, mert a „Partner" attribútuma nem egyszerű, hiszen maga is egy reláció.

| Kifizetéskód | Kárszám | Kifizetett összeg | **Partnerkód** | **Partnernév** |
|---|---|---|---|---|
| 1 | 100000000 | 128.000 Ft | *PRTNR1* | *Partner1 Kft.* |
| 2 | 110000000 | 258.000 Ft | *PRTNR2* | *Partner2 Kft.* |

**3.5. ábra. A „Kifizetések" reláció első normálformában.**

A 3.4. ábra már 1NF-ban mutatja a „Kifizetések relációt. Az első normál forma azonban nem elegendő feltétele a redundanciák megszüntetésének.

Egy reláció második normálformájában nem tartalmazhat függőséget a reláció kulcs egy részére vonatkozóan. A második normálforma definíciója a következő két feltétellel írható le:

- a reláció első normálformában van, és
- a reláció minden nem elsődleges attribútuma teljes funkcionális függőségben van az összes reláció kulccsal.

Ebben az esetben tehát a „Kifizetések" reláció már 1NF-ban van. Elsődleges kulcsa a „Kifizetéskód". Azok a relációk, melyek reláció kulcsa csak egy attribútumból áll, mindig másodiknormál formában vannak, ekkor ugyanis nem lehetséges, hogy csak a reláció kulcs egy részétől függjön egy nem elsődleges attribútum. Ebből az is következik, hogy a „Kifizetések" reláció 3.4. ábrán megadott alakja rögtön második normálformában is van.

Itt jelentkezik az egyszerű reláció kulcsok alkalmazásának egy újabb előnye, nevezetesen az, hogy alkalmazásukkal csökkenthető a redundancia.

| Kifizetéskód | Kárszám | Kifizetett összeg | **Partnerkód** |
|---|---|---|---|
| 1 | 100000000 | 128.000 Ft | *PRTNR1* |
| 2 | 110000000 | 258.000 Ft | *PRTNR2* |

| **Partnerkód** | **Partnernév** |
|---|---|
| *PRTNR1* | *Partner1 Kft.* |
| *PRTNR2* | *Partner1 Kft.* |

**3.6. ábra. „Kifizetések" reláció harmadik normálformában.**

Ahhoz, hogy egy reláció harmadik normálformában legyen a következő kritériumok teljesítése szükséges:

- a reláció második normálformában legyen, és
- nem tartalmazhat funkcionális függőséget a nem elsődleges attribútumok között.

A 3.4. ábrán világosan látható, hogy a „Partnerkód" mint nem elsődleges attribútum funkcionális függőségben van a „Partnernév" attribútummal.

A végső megoldást a 3NF eléréséhez a 3.5. ábra mutatja meg. Itt egyben az is megfigyelhető, hogy a normalizálás az adatbázis táblák számának a növekedésével jár. A redundancia eliminációja tehát (általában) azzal jár, hogy több táblát kell kezelni az adatbázisban. Jó tervezéssel megtalálható egy egyensúly a redundancia megfelelő szinten tartása és a táblák számának növekedése között. Létrejött tehát egy új „Partnerek”reláció (egy új „Partnerek” adatbázis tábla), melyben törzsadatokat, a partner cégek adatait tároljuk.

## 3.2. Az adatbázis integritási kérdései

Az adatbázis integritását számos módszer együttes alkalmazásával lehet optimalizálni. Mely tényezők azok, amelyek legjobban szolgálják az adatbázis integritásának a biztosítását? Természetesen az előző bekezdésekben ismertetett összes adatbázis-tervezési módszer is ezt a célt szolgálja. Ebben a szakaszban azokról a technológiákról ejtek szót, amelyek ugyancsak hozzájárulnak az integritás fenntartásához. Melyek ezek a konkrét módszerek?

### 3.2.1. Táblákhoz kapcsolódó constraint-ek

Alapvetően három constraint[22] elemzését végzem el konkrét példákon keresztül.

1. PRIMARY KEY CONSTRAINT:
   a. ***CONSTRAINT "PK_FGG" PRIMARY KEY ("FGG_CODE") ENABLE*** *példa a „Függőkár” tábla (reláció) „Kárszám” attribútumát a tábla kulcsmezőjeként definiálja. Ebben az esetben az RDBMS[23] létrehoz egy indexet is a tábla elsődleges kulcsára. „PK_FGG” a megszorítás neve, „FGG_CODE” a „Kárszám” attribútum adatbázisbeli neve, az ENABLE opció azt jelöli, hogy a megszorítás engedélyezve van. Lehetőség van a constraint letiltására is, ebben az esetben a DISABLE klauzula alkalmazható.*
2. CHECK CONSTRAINT:
   a. ***CONSTRAINT "CHK_FGGSTATUS" CHECK ("FGG_STATUS" IN ('NY', 'LZ', 'ZV')) ENABLE*** *példa a „Függőkár” tábla státuszkódjára definiál egy megszorítást. Ezzel a constraint-tel elérhető, hogy az adott mező csak a három megadott érték valamelyikét vehesse fel.*
   b. ***"FGG_STATUS" VARCHAR2(2) DEFAULT 'NY' NOT NULL ENABLE*** *példa azt is szemlélteti, hogy az „FGG_STATUS” attribútuma a táblának új függőkár-rekord rögzítésekor alapértelmezésben az ’NY’, azaz nyitott állapotot veszi fel. A mező a három érték valamelyikét kötelezően fel kell hogy vegye, ezt jelzi a „NOT NULL” klauzula!*
3. FOREIGN KEY CONSTRAINT:
   a. ***CONSTRAINT "FK_FGG_MODTYPECODE" FOREIGN KEY ("MODTYPE_CODE") REFERENCES "VF"."MODTYPE" ("MODTYPE_CODE") ON DELETE CASCADE ENABLE*** *példa egy un. referenciális integritási megszorítás. A függőkár rekordok felvitele vagy módosítása során a biztosítási módozat típuskódját a „Módozatok” törzstáblából kell vennie a rendszernek. Csak az ott szereplő értékek használhatóak. Amennyiben egy adott módozatkódot már rögzítettünk egy vagy több függőkárhoz, annak törlése a törzstáblából csak akkor lehetséges, ha a kapcsolódó összes függőkár-rekordot is töröljük (ON DELETE CASCADE klauzula). Függőkárokhoz kapcsolt módozatkódok módosítása a törzstáblában természetesen maga után vonja az összes olyan függőkár-rekord módosítását, melyben a módosítandó módozatkód szerepel. Ez a kaszkád referenciális integritás lényege.*

[22] Constraint = megszorítás, azaz olyan szabály, amelynek definíciója az adatbázisban kódolva van és betartásáról maga az adatbázis-kezelő rendszer gondoskodik. Ezen szabályok megsértése speciális hibaüzeneteket generál, melyeket megfelelő módon kezelve, tájékoztatni lehet a felhasználókat is, hogy milyen jellegű hiba keletkezett. Ezen szabályok megsértése kizárja a megkezdett művelet végrehajtását.
[23] RDBMS = Relációs Adatbázis-kezelő Rendszer.

Összegezve: a különböző megszorítások, alapértelmezett értékek használatával és bizonyos attribútumok értékeinek kötelező megadásával az adatbázis integritása jelentősen befolyásolható, javítható. Kiemelem a referenciális integritási megszorítás jelentőségét, hiszen elképzelhető, hogy ennek pontos definiálása hiányában megtörténhetne, hogy például a törzstáblában törölnénk egy módozatkódot, amellyel már 12000 függőkár-rekord rögzítve lett az adatbázisban. Azonnal létrehoznánk egy 12000 rekordot érintő integritási problémát, ugyanis ezekben a rekordokban olyan módozatkód szerepelne, ami már nincs meg az adatbázisban. A módosítási anomália is hasonló problémákat vet fel.

### 3.2.2. Triggerek és tárolt eljárások a művelet-végrehajtásban

A triggerek és tárolt eljárások elkészítése során természetesen ügyelni kell a definiált megszorítások következetes betartására. Az előző bekezdésekben rávilágítottam arra, hogy milyen anomáliákat vet fel például, a referenciális integritási megszorítások megszegése. Éppen ezért az adatbázis programozása során minden olyan triggerbe és tárolt eljárásba, ahol ez indokolt volt, beépítettem azokat az ellenőrző algoritmusokat, melyek biztosítják a megszorítások betartását. Mivel minden adatkarbantartás tárolt eljárások végrehajtásához kötött, az adatbázis integritásának megsértése nem lehetséges. Nem lehet például olyan függőkár-rekordot törölni, amelyhez már kapcsolódik kifizetés, de lezárt függőkárban sem lehet például kifizetni, stb.

## 3.3. Tranzakció-kezelés a teljes rendszerben

Minden tranzakció tárolt eljárásokhoz, illetve triggerekhez kötött. Milyen előnyei vannak ennek a rendszerfejlesztési technológiának? A tárolt alprogramok az adatbázis-szerveren vannak definiálva, ott futnak le, ezért a futtatásuk minden szempontból optimálisnak tekinthető. A tárolt eljárások megfelelő paraméterezésével elérhető, hogy a kliens és szerver számítógép között csak a szükséges input-output adatok mozogjanak. Ez a számítógép-hálózatnak mint kommunikációs csatornának a forgalmát is optimalizálja.

### 3.3.1. A kliens program és az adatbázis közötti kommunikáció

A VNF7.EXE kliens alkalmazás számára nem szükséges az ORACLE kliens program telepítése a kliens számítógépeken. Ez nem csak a rendszer karbantarthatóságát egyszerűsíti le. A kommunikációt is gyorsítja, hiszen a program közvetlenül a szerverre konnektál be, ott (helyben) futtatja azokat a rendszer-komponenseket, amelyek biztosítják a gyors művelet végrehajtást. AVNF7.EXE program elkészítése során speciálisan az ORACLE adatbázisokhoz kifejlesztett ODAC 6.05v komponenscsaládot használtam. Az ORACLE specifikus komponensek használata szintén optimális kommunikációt és tranzakció-kezelést eredményez.

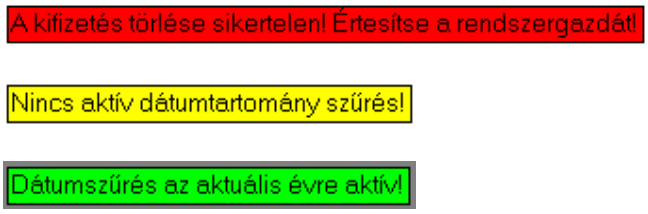
### 3.3.2. Hibakezelés az adatbázis szintjén és a kliens programban

Mivel minden fontos tranzakció a megfelelő tárolt eljárások végrehajtásához kötött, ezért mind az adatbázis-, mind a kliens program szintjén kidolgoztam egy egységes hibakezelő módszert. Ennek a rendszernek tehát van egy adatbázis szintű és egy kliens program szintű vetülete.

Az adatbázis szintjén minden tárolt alprogramba implementáltam egy un. ERROR_HANDLING (azaz hibakezelő) metódust, melynek központi eleme a RAISE_APPLICATION_ERROR beépített függvény. Ennek a függvénynek a bemenő paramétereit (a hibakódot, és magát a hibaüzenetet is) az adatbázisban létrehozott tábla megfelelő rekordja szolgáltatja. Amennyiben az adott tárolt eljárás a futása közben például egy megszorítási hibát észlel egy – már a felhasználó számára is érthető – hibaüzenetet küld a kliens program szintjére.

A kliens programban minden tárolt eljárás egy un. hibakezelő blokkban fut. Így maga a kliens program is azonnal érzékeli, ha a megkezdett tranzakció bármely okból félbeszakadna. Ilyenkor a teljes elkezdett tranzakció visszagörgetődik (ROLLBACK) a futás előtti pontig. A hibakezelő blokk egy értelmes hibaüzenetet küld a felhasználónak a hiba okáról és az esetleges tennivalókról. Erre a hibakezelő rendszerre épül rá egy üzenetküldő metódus, melynek háttere a „rendőrlámpa” néven ismert metafora.

Az üzenetek a főablak jobb alsó harmadában jelennek meg. Az egyes üzenetek három osztályba rendezhetők. A piros háttérszínnel megjelenő üzenetek valamilyen hibára (ERROR) figyelmeztetnek. A sárga háttérrel megjelenők (WARNING) tájékoztató, figyelemfelhívó üzenetek és a zöld hátterűek (MESSAGE) valamely funkció engedélyezéséről tájékoztatnak.




**3.7. ábra. Az üzenetküldő rendszer három üzenettípusa.**

Az üzenetküldő metódus az üzenet szövegét egy speciálisan elkészített ablakban jeleníti meg, a paraméterként megadott háttérszínnel. Input adata ennek az eljárásnak még az időtartam paraméter, amellyel szabályozni lehet, hogy az üzenetablak mennyi ideig legyen látható.

Az üzenetablak tehát a paraméterként megadott időtartam leteltével automatikusan bezáródik. A felhasználók nagy többsége számára idegesítő statikus üzeneteket (amelyeket nyugtázni kell) felváltja a rendszerben ez a dinamikus üzenetküldő metódus. Tapasztalataim szerint a megoldás hatékony, a mögöttes metafora is jól működik. Az üzenet megjelenésének időtartama szabályozható, azaz több időintervallum is rendelkezésre áll, annak érdekében, hogy az esetleg hangsúlyosabb üzenetek tovább legyenek láthatóak és a kevésbé fontosak rövidebb ideig jelenjenek meg.

## 3.4. Adatbiztonság, adatvédelem

Az adatbiztonságnak, adatvédelemnek különös jelentősége van egy vezetői információs rendszerben, hiszen bizalmas adatok, statisztikák tömegével dolgozik. A kiélezett versenyben nagyon fontos, hogy a piac napi eseményeire is – szükség esetén – adekvát válaszokat lehessen adni. Ebben a piaci környezetben a Biztosító Társaságon belüli nagy régiók között is egy – tudatosan beépített – versenyhelyzet áll fenn. Az egyes régiók között tehát van egyfajta üzleti titkot képező ismeretanyag, tudás, ami az Észak-alföldi igazgatóság esetében A VINFO 2007 rendszerben is realizálódik.

### 3.4.1. Az adatbázis-szerver telepítése, konfigurálása

Az adatbázis szerver telepítése és konfigurálása speciális rendszergazdai- és adatbázis-adminisztrátori tudást igényel. A rendszer adatbázisplatformja az ORACLE 10g EXPRESS EDITION verzió. Az RDBMS rendszernek külön telepítő (SRVINST) cd-t készítettem. A 32 bites Windows verzió installálása erről a cd-ről az OracleXeUniv.EXE program futtatásával indítható el. Az installációval kapcsolatban a következő fontos megjegyzéseket teszem:

- a telepítő program a teljes csomagot (a szervert és klienst is) tartalmazza, így nincs szükség az OracleXeClient.EXE telepítő programra a szerver telepítésekor sem, mert ez csak a klienst tudja telepíteni,
- az OracleXe.EXE program egy un. standard (nem teljes funkcionalitású) verziót tartalmaz, a VINFO 2007 RDBMS rendszer viszont a teljes funkcionalitású verziót használja.

A szerver konfigurációs beállításai közül a következőket emelem ki:

- Destination Folder: C:\OracleXE\: alapértelmezett telepítési útvonal, amelytől az installáció során el lehet térni,
- Port for 'Oracle Database Listener': 1521: speciális – a Windows alatt service-ként futó alkalmazás – portjának a sorszáma, alapértelmezésben ezt használja az adatbázis-kezelő rendszer és maga a VNF7.EXE kliens program is,
- Port for 'Oracle Services for Microsoft Transaction Server': 2030: szintén service-ként futó program, a Microsoft Tranzakciós Szerver eléréséhez szükséges portnak a számával,
- Port for HTTP Listener: 8080: a HTTP protokoll által (alapértelmezetten használt) portszám; az adatbázis-kezelő rendszer képes http protokollon keresztül is kommunikálni, ehhez szükséges a protokoll portjának sorszámát megadni, de ezt az alapértelmezett portot használja a rendszer HTTP kiszolgálója is.

### 3.4.2. Bejelentkezés az RDBMS szerveren

Az adatbázis szerveren történő bejelentkezéshez szükséges (SYS vagy SYSTEM) adatbázis-adminisztrátori jelszó megadása is az installálás során történik. Ez egy legalább nyolc karakterből álló (kis- és nagybetűket, valamint számjegyeket is tartalmazó) teljesen egyedi, máshol nem alkalmazott jelszó. Ezt a jelszót csak az adatbázis-adminisztrátor ismeri!

### 3.4.3. A Web-szerver telepítése és konfigurálása

A Web-szerver telepítő készlete is a „SRVINST” elnevezésű telepítő cd-n kapott helyet. Az Apache2 http kiszolgáló 32 bites verziójának installációt indító állománya:

- **Apache_2.0.59-win32-x86-no_ssl.msi**, ez a fájl a Windows alatt futtatható telepítőkészlet.

A Web-szerver telepítő csomagja a WebDevInst mappában, az RDBMS telepítő készlettől elkülönítve kapott helyet. Ennek a mappának egy alkönyvtára a WebRoot könyvtár, amely a következő alkönyvtárakat tartalmazza:

- DownLoad: ebbe a könyvtárba kerülnek azok a fájlok, amelyeket a http kiszolgáló segítségével le lehet tölteni,
- Html: könyvtár a web-szerver un. root (gyökér vagy fő) könyvtára, ez tartalmazza a rendszer háromrétegű webes alkalmazásának az állományait, a következő csoportosításban:
  - a Vinfo7 mappa a root könyvtárban van és ez tartalmazza a webes alrendszer összes állományát, azaz HTM vagy HTML kiterjesztésű fájlokat, a PHP kiterjesztésű forrásokat, a különböző bittérkép állományokat és Java scripteket,
  - ennek a könyvtárszerkezetnek megfelelően a webes alrendszer a http://localhost/vinfo7/index.html URL segítségével közvetlenül is elérhető, amennyiben a http kiszolgáló böngészőjét használjuk, ellenkező esetben a „localhost” helyére a Web-szerver IP-címe írható be, ez az útvonal azonban a felhasználók számára nem publikus, maga a rendszer is úgy van fejlesztve, hogy teljesen elfedje ezt az információt,
- Php: ez a mappa tartalmazza a CGI[24]-ként futó PHP 4.4.4 alkalmazás összes szükséges állományát,
- Upload: az esetleges feltöltések célmappája.

[24] CGI = **C**ommon **G**ateway **I**nterface.

A PHP egy szerver oldali script nyelv, amelyet elsősorban webes alkalmazások fejlesztésére terveztek és valóban elterjedten használják világszerte. Ez egy HTML forrásba beépülő programozási nyelv, amelynek forrásállományai a http kiszolgálón vannak elhelyezve. A PHP-ban megírt programblokkok a http kiszolgálón futnak, nem töltődnek le a kliens számítógépre. Ez volt az egyik fő szempont a webes alkalmazás fejlesztésben, hiszen ez a megoldás az adatbiztonságot lényegesen javító tényező.

Megjegyzés: *Tudatos – szintén az adatbiztonságot növelő – megoldás, hogy a Php könyvtár nem a Web-szerver root mappája, a Html könyvtár alá került, ti., ha ott helyeztem volna el, akkor magán a http kiszolgálón keresztül elérhető lett volna, ami természetesen nem kívánatos.*

A Web-szerver minden konfigurációs beállítását a **httpd.conf** állomány tartalmazza, melynek alapértelmezett útvonala: **C:\Program Files\Apache Group\Apache2\conf.**

A PHP 4.4.4 rendszer konfigurációs beállításait a **php.ini** fájl tartalmazza. Ennek az állománynak az elérési útvonala (az előző bekezdésekben leírtakkal összhangban): **C:\WebRoot\php.**

A Web-szerver és a PHP konfigurációs állományai egymással szinkronizált beállításokat tartalmaznak, így a teljes http kiszolgáló installáció a következő lépésekből áll:

1. **Apache_2.0.59-win32-x86-no_ssl.msi** telepítő csomag futtatása, alapértelmezett opciókkal, (csak „Next” funkciókkal), majd a http szerver leállítása a konzolprogram segítségével,
2. a **httpd.conf** állomány lecserélése a megadott útvonalon,
3. a **WebRoot** mappa bemásolása a C:\ útvonalra, és végül
4. az Apache2 szerver újraindítása.

### 3.4.4. RDBMS rendszeradminisztráció

A rendszeradminisztráció eszközeként az SQL Developer első fejezetben megadott (ORACLE specifikus fejlesztésű) rendszert használom. Ebben történik PLSQL programozási nyelven az adatbázis minden objektumának a fejlesztése és szükség esetén a módosítása is. Ennek a fejlesztő- és egyben rendszeradminisztrációs programnak a telepítése a rendszeradminisztrátor feladata. Elegendő ezt a szoftvert egy gépen (a rendszeradminisztrátor) gépén telepíteni, hiszen a program képes a távoli szervereket elérni. Ehhez természetesen ismerni kell a szerverek telepítési paramétereit és IP-címét, valamint a bejelentkezéshez szükséges összes adatot.

Beépített lehetőséget nyújt ez a program az adatbázis adatainak a mentésére. Ezt a lehetőséget elsősorban a törzsadatok mentése során alkalmazom. Az adatok több formátumban történő exportjához is alkalmazható a rendszer, ezt a lehetőséget azonban szintén csak a kisebb rekordszámú táblákon célszerű alkalmazni.

### 3.4.5. Az adatok mentése, archiválása

A teljes adatbázis mentése egy – a lokális szerverektől független – központi szerverre történik. Erről a központi szerverről napi szalagos mentés készül, tehát az adatok duplikált mentése történik egymástól független háttértároló eszközökkel. A mentést a lokális szervereken elhelyezett ütemezett feladatként futó BATCH[25] script végzi, a rendszer karbantartási időszakában.

[25] BATCH = kötegelt művelet végrehajtás. Ez egy olyan script állomány, amelyet a Windows OP-rendszer alatt ütemezett feladatként is futtatni lehet. A BATCH scriptek olyan utasítások kötegét tartalmazzák, amelyeket az OP-rendszer szekvenciálisan végrehajt.

## 3.6. Az ötszintű jogosultsági rendszer

A rendszerben a felhasználói igények alapján egy ötszintű jogosultsági rendszert alakítottam ki. Ennek a jogosultsági rendszernek az érvényessége a teljes elosztott rendszerre kiterjed. Ez azt jelenti, hogy minden egyes felhasználója a rendszernek egy un. jogosultsági kódot kap, amelyhez konkrét felhasználói típus van rendelve. Minden felhasználó típushoz hozzákapcsoltam az általa végezhető műveletetek halmazát, így tulajdonképpen egy jól definiált jogosultsági osztályzást készítettem el.

Felhasználói típusok

| Felhasználótípus kódja | Felhasználótípus neve |
|---|---|
| LOCAL_BOSS | Helyi vezető |
| REGION_BOSS | Igazgatósági vezető |
| RFRNS_USER | Referens |
| SMPL_USER | Ügyintéző |
| SYS_ADMIN | Rendszer-adminisztrátor |

**3.8. ábra. A rendszer felhasználói típusai.**

A 3.7. ábra. Az öt felhasználói osztályt mutatja. Milyen jogosultságokkal rendelkeznek az egyes osztályok?

1. SYS_ADMIN: teljes jogosultság a rendszer egészében.
2. REGION_BOOS: a törzsadatok és a felhasználói adatok kivételével minden adathoz hozzáfér és azokon a folyamat nézet táblázatában foglalt mind a 18 műveletet használhatja.
3. LOCAL_BOSS: sem a törzsadatokhoz, sem a felhasználói adatokhoz nincs jogosultsága, le van tiltva a hozzáférése a díjadatokhoz és a globális statisztikákat tartalmazó VIM modulhoz. A lokális felhasználók összes adatát viszont láthatja és azokon a folyamat nézet táblázatában foglalt műveleteket elvégezheti a 3. (törlési funkció) kivételével, azaz semmilyen adatot nem törölhet.
4. SMPL_USER: csak a káradatok és kifizetések rögzítéséhez és módosításához van jogosultsága. Kizárólag a saját kárait és kifizetéseit kezelheti. A káraktát lezárhatja, de megnyitni csak a helyi vezető tudja a lezárt károkat.
5. RFRNS_USER: Kizárólag a díjadatok rögzítéséhez és módosításához van jogosultsága. Csak a saját díjadataira és azok kiegészítő díjaira lát rá.

A kárügyintézők saját adataikon használhatják a 10-15 funkciókat és ugyanez a referensekről is elmondható.

A rendszer indulásakor egy elemző algoritmus megvizsgálja az adott felhasználó jogosultság kódját és annak megfelelően testre szabja a teljes GUI-t. Minden olyan funkciót letilt, amelyhez nincs jogosultsága a bejelentkezett felhasználónak.

A szabadságok, betegségek, tartós távollétek, stb. kezelésére a rendszerbe beépítettem egy un. státuszkódot, amelynek három lehetséges állapota van:

1. A: aktív felhasználó,
2. P: passzív felhasználó; ezt az állapotot akkor használjuk, amikor egy munkatárs például tartósan távol van,
3. K: kilépett.

A távollévő munkatársak állapotkódját „**P**asszívra” állítva senki nem képes bejelentkezni az Ő adataival, a felettesei természetesen rálátnak minden adatára. A kilépette dolgozók végleg kitilthatók a rendszerből.

# 4. fejezet RENDSZERDOKUMENTÁCIÓ

## 4.1. A rendszer telepítése

Ebben a szakaszban a rendszertelepítés gyakorlati lépéseit taglalom, a rendszertelepítés tervezése során meghatározott elvek alapján. Kiemelem azt a törekvést, miszerint a kliens oldalon telepített rendszerkomponensek számát minimalizálni kell annak érdekében, hogy a karbantartás a lehető legegyszerűbb legyen. A karbantartás az első telepítés után lényegében távolról – a telepítéstervezésnél megfogalmazott módszerrel – legyen elvégezhető.

### 4.1.1. Rendszerparaméterek, környezeti változók

A kliens oldali komponensnél a fejlesztés során alkalmazott speciális eszközöknek és módszereknek köszönhetően nincs szükség az operációs-rendszerben új környezeti változók definiálására. A Windows XP Professional SP2 verzió standard feltelepítésekor létrehozott (alapértelmezett beállítások) megfelelőek, semmilyen különleges paraméter definiálására nincs szükség.

A kliens program saját dátumformátumot[26] használ. Ez az adat már az adatbázisban is olyan (konvertált) formátumban van eltárolva, aminek a felhasználói interfészen további konverziója nem szükséges. A VNF7.EXE program tehát saját dátumformátumát használja, nem veszi át a Windows OP-rendszer beállításait. Így biztosítható az, hogy a felhasználók által szabadon állítható dátumformátumok befolyásolják a kliens programban megjelenített adatokat, azok formátumát. Ez egységes adatmegjelenítést eredményez a GUI szintjén.

A rendszer kiterjedten alkalmaz különböző szűréseket dátumadatokra. Ez is indokolja az egységes dátumformátumok használatát. Az alkalmazott formátum:

- YYYY.MM.DD. HH24:MI:SS: megfelel a szabványos un. hosszú dátum-idő formátumnak, amelynek komoly szerepe van ügyviteli szempontból is a rendszerben, hiszen a tranzakcionált adatkezelésnél annak is jelentősége van, hogy az egyes tranzakciók végrehajtása óra, perc, másodperc pontossággal követhető legyen.

A Windows pénznem formátumát néhány ritka esetben átveszi a kliens program. Ezek elsősorban a dinamikus listák pénznem adatainak a kirajzolásakor történnek. Az összes többi esetben szintén saját formátumokat használok.

### 4.1.2. Rendszerkönyvtárak, rendszerállományok

A telepítő program, két könyvtárat hoz létre a kliens számítógép merevlemezén. Sem a meghajtóra, sem a tartalmazó könyvtárra nincs megkötés a telepítés során. A lokális felhasználók gépein kizárólag a VNF7.EXE és három inicializáló állomány telepítése szükséges. Vezetői számítógépeken az XVIMManager.EXE konzolprogram felmásolására is szükség van.

[26] A Windows dátum- és pénznem formátumának beállítása a **Startmenü\Beállítások\Vezérlőpult\Területi és nyelvi beállítások** útvonalon érhető el. A „Területi beállítások” fülön a „Testreszabás…” gombra kattintva jelennek meg azok a lehetőségek, amelyekkel ezeket a formátumadatokat meg lehet változtatni.

A telepítéstervezés során alapértelmezettként tekintett C:\Program Files\Vinfo7 útvonalat célszerű használni. Semmi nem indokolja az ettől való eltérést, bár ez lehetséges. Egyetlen kikötés, hogy a VNF7.EXE programmal azonos szinten létezzen a SYSTEM könyvtár és benne három inicializáló állomány:

- APPSYS.INI: a rendszer főablakának elmentett paramétereit tartalmazza,
- DBPRM.INI: a rendszer adatbázishoz történő kapcsolódásának legfontosabb paraméterei mellett tartalmazza az alapvető rendszerparaméterek szekcióját, valamint az adatexporthoz, és a Web-Szerver eléréséhez szükséges alapértelmezett útvonalakat; ebben a belső struktúrával rendelkező fájlban a fontosabb szekciók:
  - [DataBase]: az adatbázishoz történő kapcsolódás paramétereit definiálja
  - [SystemParameters]: rendszerparaméterek definíciója,
  - [SaveFggData]: alapértelmezett útvonalak, URL-ek,
- SYSPRM.INI: a 2.3. ábrán megadott főformok paramétereinek a mentését tartalmazza szekciókba rendezett módon.

Az első telepítéshez lokális rendszergazdaként kell bejelentkezni és a telepítő cd-ről el kell indítani a VINFOINST.EXE telepítő programot. Az installáció során csak azt kell megadni, hogy lokális felhasználó gépére történik a telepítés, vagy vezetői számítógépre. Más teendő a telepítés befejezéséig nincs.

### 4.1.3. Jogosultságok a telepített rendszerkomponenseken

A Biztosító Társaságnál érvényben van a Windows operációs rendszerekhez egy POLICY[27], amelynek alkalmazásával a felhasználók OP-rendszerbeli műveleteit és bizonyos hardver- és szoftver erőforrásokhoz való hozzáférésüket korlátozzák. A kliens számítógépen biztosítani kell, hogy a Vinfo7 könyvtárra és annak alkönyvtáraira, illetve azok tartalmára a felhasználói csoportnak módosítási joga legyen. Erre azért van szükség, mert a VNF7.EXE program futás közben – az éppen futó folyamatoktól függően – írhatja, módosíthatja az inicializáló állományok tartalmát. Másik fontos szempont a Vinfo7 könyvtár módosíthatóságának a beállítására, hogy ellenkező esetben nem volna lehetséges a távkarbantartás.

## 4.2. Globális UNIT-ok

A rendszerfejlesztés teljesen objektum orientált módszerekkel és eszközökkel történt. A Borland Delphi Enterprise Edition fejlesztői környezetben projektalapú fejlesztés és dokumentálás valósítható meg. A fejlesztői IDE által nyújtott lehetőségekkel törekedtem széles körűen élni.

A fejlesztés során néhány globális unitot hoztam létre, melyek alapvető feladatai a következő pontokban foglalhatók össze:

- tartalmazzák a teljes rendszer globális konstansait, típusait és változóit,
- itt definiáltam a rendszerben globálisan használt un. erőforrás sztringeket (például a felhasználónak szánt szöveges üzeneteket), és
- a rendszer globális függvényeit és eljárásait.

Ezeknek a globális UNIT[28]-oknak az alkalmazásával nem csak a rendszer jól strukturált szerkezetét lehet kialakítani, hanem adatbiztonsági szempontból is fontos szerepük van, hiszen nélkülük a teljes projekt használhatatlan.

---

[27] POLICY = házirend, amely korlátozhatja a felhasználók műveletvégző képességét, bizonyos erőforrásokhoz történő hozzáférését.

[28] A UNIT a PASCAL programozási nyelvben egy strukturált programozási egység, modul. Ezekből a modulokból építjük fel a teljes rendszert.

### 4.2.1. A DELPHI projekt fő programozási egységei

```
Program VNF7;

Uses
  Forms,
  ATOM In 'ATOM.PAS',
  GLOBALAPPID In 'GLOBALAPPID.pas',
  RUNFIRST In 'RUNFIRST.PAS',
  RCONSTNS In 'RCONSTNS.PAS',
  GLOBALS In 'GLOBALS.pas',
  dxGlobalUtils In 'dxGlobalUtils.pas',
  dxGridMenus In 'dxGridMenus.pas',
  Splash In 'Splash.pas' {SplashForm},
  VNFMain In 'VNFMain.pas' {MDIFrame},
  DataModule1 In 'DataModule1.pas' {DM1: TDataModule},
  DataModule2 In 'DataModule2.pas' {DM2: TDataModule},
  About In 'About.pas' {AboutBox},
  Ssetup In 'Ssetup.pas' {SystemSetupForm},
  SrvPwd In 'SrvPwd.pas' {ServicePwdForm},
  FLogin In 'FLogin.pas' {LoginForm},
  BodyData In 'BodyData.pas' {BodyDataForm},
  UserData In 'UserData.pas' {UserDataForm},
  VNFFgg In 'VNFFgg.pas' {FggForm},
  SrvFggPyng In 'SrvFggPyng.pas' {SrvFggPyngForm},
  VNFPyng In 'VNFPyng.pas' {PyngForm},
  VNFViewPyng In 'VNFViewPyng.pas' {ViewPyngForm},
  Picdrqry In 'PICDRQRY.PAS' {DateRangeQueryForm},
  VNFIncm In 'VNFIncm.pas' {IncmForm},
  PicNewRsrvM In 'PicNewRsrvM.PAS' {NewFggRsrvmdfyForm},
  SrvIncm In 'SrvIncm.pas' {SrvIncmForm},
  VNFIncmExp In 'VNFIncmExp.pas' {VNFIncmExpForm},
  VNFViewIncmExp In 'VNFViewIncmExp.pas' {VNFViewIncmExpForm},
  SrvIncmExp In 'SrvIncmExp.pas' {SrvIncmExpForm},
  VNFDataMining1 In 'VNFDataMining1.pas' {VNFDMiningForm},
  VNFDataMining2 In 'VNFDataMining2.pas' {VNFDataMiningForm2};
```

**4.1. ábra. A VNF7 Delphi projekt unitjai, formjai.**

A 4.1 ábra szemlélteti a VNF7 nevű projekt fő UNIT-jait, a következő csoportosításban:

- globális UNIT-ok:
  - ATOM.PAS: speciális rendszerindítási metódusok,
  - GLOBALAPPID.PAS: globális rendszerazonosítók,
  - RUNFIRST.PAS: a rendszerindítás globális paraméterei,
  - RCONSTNS.PAS: globális konstansok, típusok, változók, erőforrás sztringek,
  - GLOBALS.PAS: globális függvények és eljárások,
  - DXGLOBALUTILS.PAS: globális függvények és eljárások,
  - DXGRIDMENUS.PAS: globális függvények és eljárások,

- olyan UNIT-ok, amelyekhez vizuális komponensek FORMOK is tartoznak.

### 4.2.2. Globális metódusok

A globális modulok közül az ATOM.PAS UNIT metódusainak a működését emelem ki, mert egy adatbázis-kezelő rendszer elkészítésében az itt alkalmazott eljárások alapvető szerepet játszanak. Milyen feladatot kell megoldania az egységnek?

Ennek a kérdésnek a megválaszolásához a kliens-szerver kapcsolat felépítését szükséges elemezni. Ismeretes az a tulajdonsága az ORACLE RDBMS-rendszernek, hogy a szerverre egy időben konnektáló (csatlakozó) felhasználók száma korlátozott. Minden egyes kliens program indítás, egyben egy konnekt is a szerveren. Az is köztudott, hogy a Windows operációs rendszerben egy program több példányban is elindítható, hiszen a központi tár[29] szegmentált és minden egyes alkalmazás külön szegmensben fut.

Meg kell tehát akadályozni, hogy a felhasználók a kliens programot egyszerre több példányban is futtathassák, hiszen ebben az esetben egy felhasználó több kapcsolatot is felhasznál a szerveren. Ez fölösleges terhelést okoz a szerveren, lassítja annak működését és végső soron megakadályozhat más felhasználókat a szerverhez történő csatlakozásban.

Hogyan oldja meg ezt a problémát az ATOM.PAS-ban implementált kód?

Az ellenőrzéshez felhasználja a WINDOWS ATOM táblázatát, amely egy állandóan elérhető, globális sztring-lista. Az alkalmazás élettartamát két fázisra bontja:

1. alkalmazás betöltési fázisára, és az
2. alkalmazás futási fázisára.

Mindkét fázishoz egyedi azonosító sztringet rendel és az adott fázisnak megfelelő sztringet beregisztrálja az ATOM táblázatba, de egyszerre mindig csak az egyiket. Mivel az ATOM.PAS globális UNIT a project USES listájának az első helyén szerepel (lásd 4.1. ábra), ennek a modulnak az INITIALIZATION szekciója fut le először. Ekkor az algoritmus megkezdi az alkalmazás azonosítását.

Megjegyzés: *A futtatást is több részre kell bontani, mert mint fejlesztő magából a DELPHI-ből is futtathatom az alkalmazást. Ebben az esetben Először ellenőrzi, hogy fut-e a Delphi, ha igen átlépi az ellenőrzést. Ezt követően megvizsgálja, hogy az alkalmazás "/d" vagy "-d" paraméterrel lett-e indítva. Ha igen, törli az esetleg beragadt azonosító sztringeket a Windows ATOM táblázatából, majd HALT utasítással kilép.*

Amennyiben nem a fejlesztői környezetből, hanem az OP-rendszerből történik a program indítása, akkor ellenőrzi, hogy az alkalmazás "futó" fázisához rendelt sztring be van-e regisztrálva az ATOM táblázatba,

- ha igen már fut az alkalmazás egy példánya és HALT utasítással kilép,
- ha nem, megnézi, hogy az alkalmazás "betöltése" fázisát azonosító sztring be van-e regisztrálva az ATOM táblázatba,
    - ha igen, megnézi, hogy a "TApplication" nevű ablak létezik-e a Windows-ban. ha igen, HALT utasítással kilép, mert az alkalmazásnak már fut egy példánya.
    - ha nem, az alkalmazásnak még egyetlen példánya sem fut, a beregisztrált sztring "beragadt" az ATOM táblázatba, mert az alkalmazást előzőleg feladatkezelővel állították le. Ekkor a "beragadt" sztring-et törli, majd a "StartingApplication" globális változót True értékre állítja, jelezve, hogy megkezdődött az alkalmazás teljes betöltése, ezt követően beregisztrálja a "futó" alkalmazást azonosító sztring-et és megkezdi az alkalmazás teljes betöltését.

  A betöltés eljut a project fő blokkjáig. Itt meghívódik az ApplicationStart.Execute metódus és elkezdődhet a program teljes betöltése a központi tárba.

[29] Központi tár, azaz közvetlen elérésű memória (RAM = Random Access Memory).

A kliens program betöltődésének utolsó fázisaként "StartingApplication" globális változót False értékre állítja a metódus, ezzel jelezve, hogy az alkalmazás teljes mértékben betöltődött. Végül meghívja az Application.Run metódust.

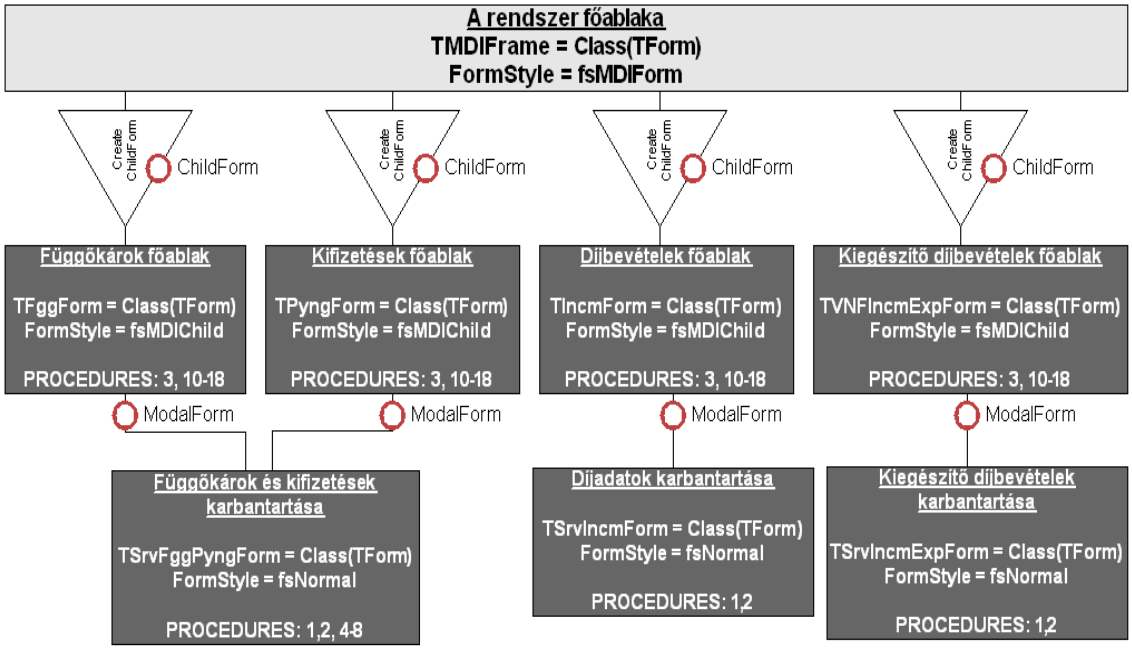
Az alkalmazásból történő kilépéskor törlődik az ATOM táblázatba beregisztrált sztring. Az ATOM unit hivatkozik egy GlobalAppID nevű modulra, amit alkalmazásonként el kell készíteni. Ebbe az egységbe egy egyedi (az alkalmazásra jellemző) azonosítót kell elhelyezni.

Az alkalmazás fejlesztése során számtalan ilyen, vagy ehhez jellegében hasonló probléma megoldásával kell szembenézni. Adott egy egyszerűen megfogalmazható feladat: meg kell gátolni a kliens program több példányban történő futtatását! Látható, hogy a végső megoldás nem egyszerű. Természetesen nincs arra mód, hogy az összes fontos metódust bemutassam, éppen ezért csak a rendszerfejlesztést jól jellemző (és érdekes, de kiragadott) példákat mutatok be.

## 4.3. Az MDI projekt-fejlesztés

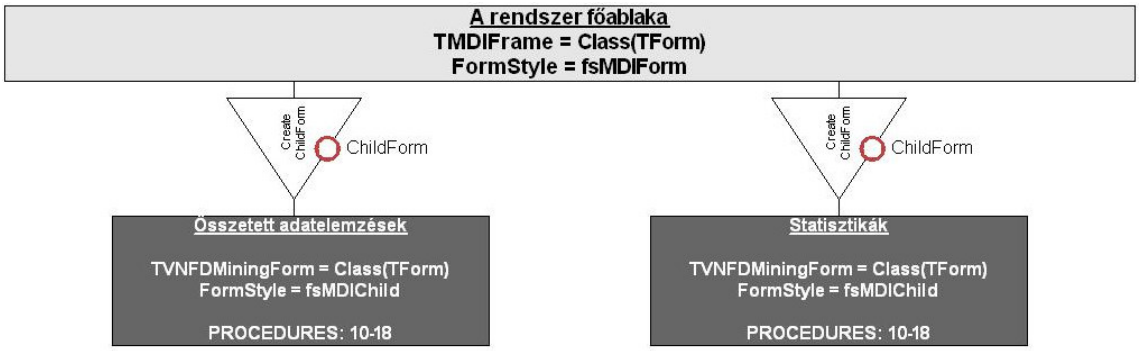
Az MDI projekt jellemzője, hogy az alkalmazásnak van egy főablaka (MDIFrame). Minden további ablaka a programnak ebben MainFrame-ben nyitható meg. A főablak leszármazottjai (a gyerekablakok) ebben a keretben logikusan elrendezhetők, csoportosíthatók, dinamikusan átméretezhetők. Ez a fejlesztési módszer talán nehezebb feladatok elé állítja a fejlesztőt, mint egy SDI alkalmazás elkészítése, viszont a felhasználó számára egy jól kezelhető GUI-t eredményez, amelyben a testreszabhatóság, az intelligencia és egyfajta rendszerezettség nagyobb teret kap.

### 4.3.1. GUI bemutatása, Formok, Osztályok, Öröklődés




**4.2. ábra. GUI fő objektumai, osztályok, öröklődés**

A 4.2. ábrán nem csak a rendszer főformjai és azok osztály-, illetve típusdefiníciói láthatók, hanem az egyes formokban érvényes metódusok sorszámai is. A metódus definíciókat a folyamat nézet (Process view) táblázatban adtam meg. Ebben a táblázatban beazonosítható az egyes kódokhoz tartozó metódusok jelentése. Ebből az ábrából az egyes ablakok létrehozásának a módja (ChildForm, ModalForm) is leolvasható.




**4.3. ábra. GUI Vezetői Információs Modul formjai, osztályok, öröklődés.**

A 4.3. ábra a rendszer VIM moduljának a formjait ábrázolja. Itt is megtekinthető a MainFrame-hez tartozó két form összes jellemző tulajdonsága.

A 4.2. ábra és a 4.3. ábra is az adatbázis fő objektumainak a grafikus felhasználói interfészen történő leképezését mutatja meg. Ez a GUI fejlesztése során nem csupán egy vizuális megjelenítés, hanem a relációkon (az adatbázis egyes tábláin) értelmezett metódusok leképezése is. Az alkalmazott fejlesztői környezetben a formok vizuális tervezése és az elvárt funkciók implementálása párhuzamosan zajlik.

## 4.4. Projektkönyvtárak

A DELPHI projekt fejlesztése során a telepítéstervezésnél megadott (alapértelmezett) könyvtárstruktúrát használtam, azaz a projekt főkönyvtára a C:\Program Files\Vinfo7 útvonalon található. Ebben a mappában van a SYSTEM és a SAVE alkönyvtár. A SYSTEM könyvtárban természetesen a fejlesztés során is ott kell lennie a három INI fájlnak.

### 4.4.1. A projekt állományai

A projekt főkönyvtára tulajdonképpen az XVIMManager.EXE konzol program és a VIMShedule.EXE ütemezett adatbázis karbantartó alkalmazás projektkönyvtára is. A Vinfo7 mappában tehát három DELPHI projekt összes állománya van elhelyezve. Ezek projektállományok, forráskód fájlok, a vizuális objektumok (formok) kódját tartalmazó fájlok, erőforrás állományok és természetesen itt vannak a lefordított programkódok is. A fejlesztői környezet komponens alapú vizuális tervezésre készült, így a telepített komponens-könyvtárakkal együtt a projekthez kapcsolódó állományok száma rendkívül nagy.

Mindhárom projektre érvényes, hogy speciális komponenskönyvtárakat használ. Ezeket a komponenskönyvtárakat a bennük található komponensek szerint két osztályba lehet sorolni:

- vizuális (VCL[30]) komponens könyvtárakra, és
- vizuális komponenseket nem tartalmazó könyvtárakra.

Amennyiben egy adott komponenskönyvtár komponensét vagy metódusát alkalmazom a projektfejlesztés során, annak vizuális és forráskód szintű elemei is beépülnek a projektbe és természetesen a bináris állománynak is részei lesznek a projekt lefordítása után. Éppen ezért a fejlesztett projektek IDE specifikusak. Más fejlesztői környezetben többnyire meg sem nyithatóak hibák nélkül. Ez magának a projektnek a hordozhatóságát annyiban befolyásolja, hogy az alkalmazott komponenskönyvtárakat is csatolni kell a teljes projekthez. Ebben az esetben újraépíthető az a fejlesztői környezet, amiben hibátlanul kezelhető a projekt.

[30] VCL = **V**isual **C**omponent **L**ibrary, azaz vizuális komponenseket tartalmazó könyvtár.

A teljes projekt, jelenleg 253.616 sort tartalmaz. Ebben azonban a komponenskönyvtárak összes érintett forrásállományainak a sorai is benne vannak. A komponenskönyvtárak több ezer állományt foglalnak magukban és az általam fejlesztett három projekthez is több száz fájl tartozik. Ezért a jelenlegi projektfejlesztési technológiák mellett célszerű csak a legfontosabb könyvtárakat és állományokat dokumentálni.

### 4.4.2. SQL script-ek, Java script-ek és PHP források

Az SQL scriptek mentett változatait a projektmappa SAVE alkönyvtárában helyeztem el. Ezek a PLSQL-ben megírt programkódok a következő állományokban vannak:

- TABLE.SQL: állomány tartalmazza az adatbázis táblák definícióit, a táblákhoz kapcsolódó szekvenciákkal, megszorításokkal ( constraint-ekkel) és kommentekkel,
- TRIGGER.SQL: fájlban a triggerek SQL implementációi vannak,
- STRDPRC.SQL: fájl tartalmazza az adatbázis összes tárolt eljárását,
- VIEW.SQL: állományban helyeztem el az adatbázis nézeteinek a forrását, és az
- INSERT.SQL: scriptben azok az SQL-ben elkészített insert-ek vannak definiálva, amelyek már az adatbázis telepítése után felviszik a szükséges (alapértelmezett) rekordokat a megfelelő adatbázis táblákba.

A Java scriptek és PHP források a webes alkalmazás részei. A Java scripteket a *WebRoot\Html\Vinfo7\Scrpts* útvonalon lehet megtalálni, míg a PHP forrásokat a *WebRoot\Html\Vinfo7\Libraries* könyvtárban helyeztem el.

A webes rendszerhez tartozó bittérkép állományok a *WebRoot\Html\Vinfo7\Images* útvonalon vannak. A Vinfo7 mappa tartalmaz még három további könyvtárat HHAJD, HSZAB és HSZOL nevekkel, ezekbe a mappákba lehet menteni az egyes megyék kiexportált adatait. Így az Intraneten azok azonnal láthatóak. Ezekhez a mappákhoz létrehoztam két globális csoportot, melyek közül az egyiknek csak olvasási joga van, a másiknak pedig módosítási jogosultsága is. Ezzel a módszerrel még a vezetői szinten belül is jól szabályozható az adathozzáférés módja.

A *WebRoot\Html\Vinfo7* mappában vannak a webes alkalmazás legfontosabb állományai:

- hhajd.htm, hszab.htm, hszol.htm fájlok az egyes megyék önálló oldalai,
- ifrmleft.htm, ifrmmiddle.htm, ifrmright.htm állományok a főoldal baloldali, középső és jobboldali kereteit készítik el,
- ig.htm, az igazgatóság főoldala,
- infostore.htm, a VINFO 2007 újdonságait, dokumentációit tartalmazó oldal; ezt érik el a lokális felhasználók,
- tocframe.htm, a három keret integrációját végző oldal.

A webes alkalmazásban a frame-technológia alkalmazása mellett nem csak azért döntöttem, mert jól áttekinthető site-okat lehet készíteni velük, hanem azért is, mert használatukkal a fontos URL-ek is elrejthetőek szükség esetén. Tehát ez a fejlesztési megoldás is része az adatvédelemnek.

## 4.5. A rendszerteszt módszerei és eredményei

A rendszertesztelés folyamatát alapvetően négy szakaszra osztottam fel. Ezek a szakaszok a következő pontokban összegezhetők:

1. Alfa teszt: ebben a szakaszban egy un. validátor munkatárs segítségével közösen teszteltük azt, hogy a követelményspecifikációban megfogalmazottak hogyan teljesülnek a rendszer egészében. Itt lényegében a konceptuális tervezésnél definiált nézetekben leírtakat ellenőriztük, azaz…
    a. a leképzett funkciók teljességét és kapcsolatrendszerét ellenőriztük,
    b. a leképzett folyamatok (metódusok) szemantikai helyességét az adatbázis- és a kliens program szintjén is vizsgáltuk (a szintaktikai ellenőrzés a rendszerfejlesztés folyamatának permanens része, ezt az integrált fejlesztői környezetben végeztem el), és
    c. teszteltük azt, hogy a felhasználói inteface hogyan integrálja az összes funkciót.
2. Béta teszt: ennek az ellenőrzési folyamatnak a keretében mindhárom megyében megtörtént a szerverek telepítése. Ebben a fázisban a validátor munkatárssal együtt ellenőriztük magának a telepítésnek a menetét is. A szerverek telepítése és konfigurációja után az adatbázis szerveren egy tesztadatbázist telepítettem fel. Ebben az adatbázisban a szükséges törzsadatok már benne voltak. A törzsadatok összeállítását korábban, egy vezetőkből és beosztott munkatársakból álló csoporttal végeztük el, a normalizálás után kialakított relációs modell alapján. A törzsadatok mindhárom megyében egységesek!
    a. A tesztelésnek ebben a fázisában minden megyében egy lokális felhasználói csoportot képeztünk, akik egy félévre visszamenőleg berögzítették az összes függőkárt és azok kifizetéseit, valamint a díjadatokat és azok kiegészítő díjait. Ellenőriztek minden olyan funkciót, ami a folyamat nézet táblázatban szerepel. Ebbe a tesztelési szakaszba bevontuk a controlling munkatársat, aki elsősorban a berögzített adatokra épülő vezetői statisztikák, adatelemzések helyességét vizsgálta.
    b. A felhasználói csoportok vezetői összegezték az észlelt hiányosságokat, javaslatokat és írásos formában megküldték azokat.
3. Éles verzió kibocsátása. A beérkezett listák alapján elvégeztem a szükséges javításokat. Ezt követően került sor az első éles programverzió kibocsátására. Ekkor a helyi szervereken a tesztadatbázis helyett az éles verziót telepítettem fel és minden felhasználói gépen (távkarbantartással) installáltam az új (éles) verziót.
4. Éles verzió utóellenőrzése. Ez egy újabb féléves éles adatmennyiség berögzítése után történt meg. Egy újabb projektértekezlet keretében tekintettük át a rendszer működését. Ezen az értekezleten összegeztük az addig felhalmozott tapasztalatokat és megbeszéltük a továbbfejlesztés irányait.

A továbbfejlesztés fő irányait már az első fejezetben tisztáztam. Alapvetően pozitív visszajelzéseket kaptam a vezetőktől és a beosztott munkatársaktól. Az elosztott rendszermodell jól vizsgázott, ti. a rendszerben végzett műveletek válaszideje jó, azaz megfelelően gyors a rendszer. Minden felhasználó úgy érzi, hogy a beépített funkciók jól működnek, gyorsan kezelhetőek. Ezeket a pozitív visszacsatolásokat a felhasználói igények maximális figyelembevételének tulajdonítom. Ez magyarázza, hogy a rendszer bevezetése során nem tapasztaltam érdemi ellenállást a felhasználói körben!

# 5. fejezet FELHASZNÁLÓI DOKUMENTÁCIÓ

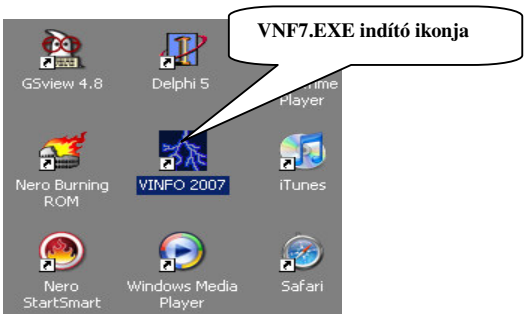
Megjegyzés: *A program használata során a különböző menüpontok, illetve az eszközsoron elhelyezett nyomógombok alkalmazásakor alapértelmezésben mindig az egér bal gombjával történő szimpla kattintást kell érteni! Amennyiben ettől eltérő egérműveletet szükséges végezni, az a dokumentációban külön fel van tüntetve!*

## 5.1. Programindítás

A programindítás attól függően, hogy beosztott felhasználóról, vagy vezető, elemző munkatársról van szó, kétféle módon történik.

- Az első esetben közvetlenül a VNF7.EXE programot kell indítani.
- A második esetben pedig az XVIMManager.EXE konzolt szükséges futtatni.

### 5.1.1. A kliens program indítása




**5.1. ábra. Kliens program indítása.**

A kétrétegű kliens-szerver adatbázis-kezelő rendszer kliens programjának (VNF7.EXE) az indítása az első telepítés alkalmával a Windows munkaasztalára kihelyezett ikon segítségével történik (lásd 5.1. ábra).

Az ikonra az egér bal gombjával duplát kattintva történhet a program futtatása. A program elindulását követően elsőként a bejelentkezés ablaka jelenik meg.

### 5.1.2. Az XVimManager konzol indítása kiemelt felhasználói gépeken

A vezetői számítógépeken – szintén az első telepítés alkalmával – az XVIMManager.EXE konzolprogram is telepítésre kerül. Ennek a programnak az indító ikonját az adott felhasználó Startmenüjében a \Programok\Indítópult\ útvonalra elhelyezi a telepítő program. Így elérhető, hogy az operációs rendszer indulásakor a konzol program automatikusan elinduljon és bekerüljön az ikongyűjtőbe.

A vezetői jogosultsággal rendelkező felhasználók tehát a konzol program segítségével indítják el a kliens programot. Ez megtehető az ikonon történő dupla kattintással vagy az indító ikonra a jobb egérgombbal kattintva a „VIM menedzser…” menüpont kiválasztásával.


**5.2. ábra. Konzol program használata.**

A konzol program elindítása után az 5.2. ábrán bemutatott ablak jelenik meg.

Ebben az ablakban elsőként annak az adatbázisnak a kiválasztását kell elvégezni, amelyhez csatlakozni kívánunk. Ez a szelekció az egyes adatbázisokat leíró szövegekre vagy a szöveg előtt található rádiógombra történő kattintással lehetséges.

Az adatbázis kiválasztását követően a Startgomb aktív állapotba kerül, és erre kattintva elindul a kliens program.

A kliens program elindulását követően elsőként a rendszerbe történő bejelentkezést kell végrehajtani.


**5.3. ábra. Bejelentkezés a rendszerbe.**

A bejelentkezés során minden felhasználó a rendszergazdától kapott azonosítót és jelszót gépeli be a megfelelő adatbeviteli mezőbe.

Az adatok helyes begépelése után válik aktívvá a „Bejelentkezés” gomb. Erre kattintva megjelenik a rendszer főablaka.

## 5.2. A rendszer főablaka

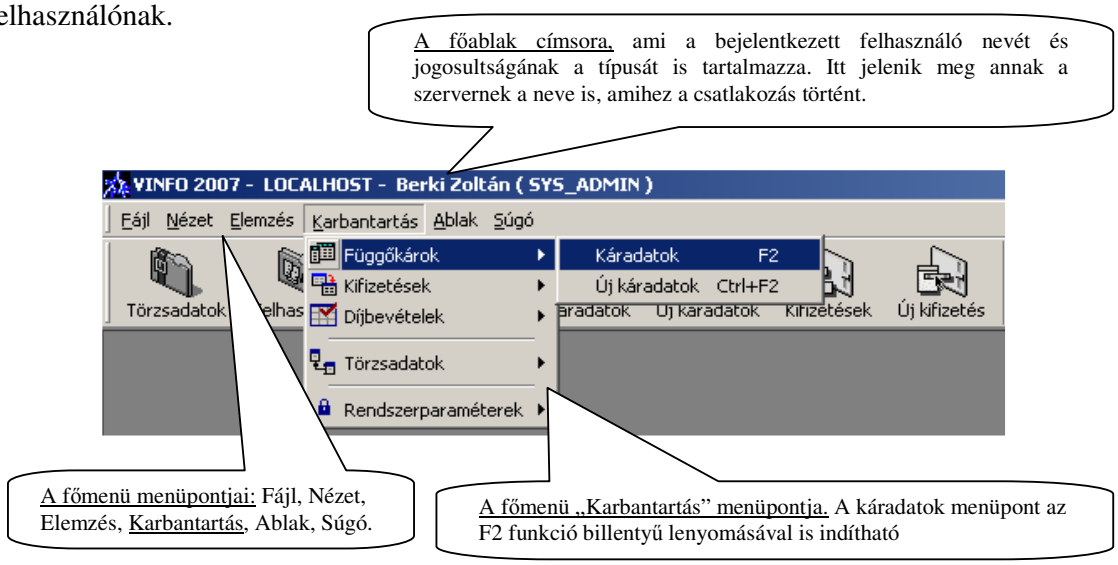
A rendszer főablakának öt alapvető eleme van:

- Címsor.
- Főmenü.
- Eszközkészletek.[31]
- Státuszsor.
- Munkaterület.

### 5.2.1. A főmenü, az eszközkészletek és a státusz sor bemutatása

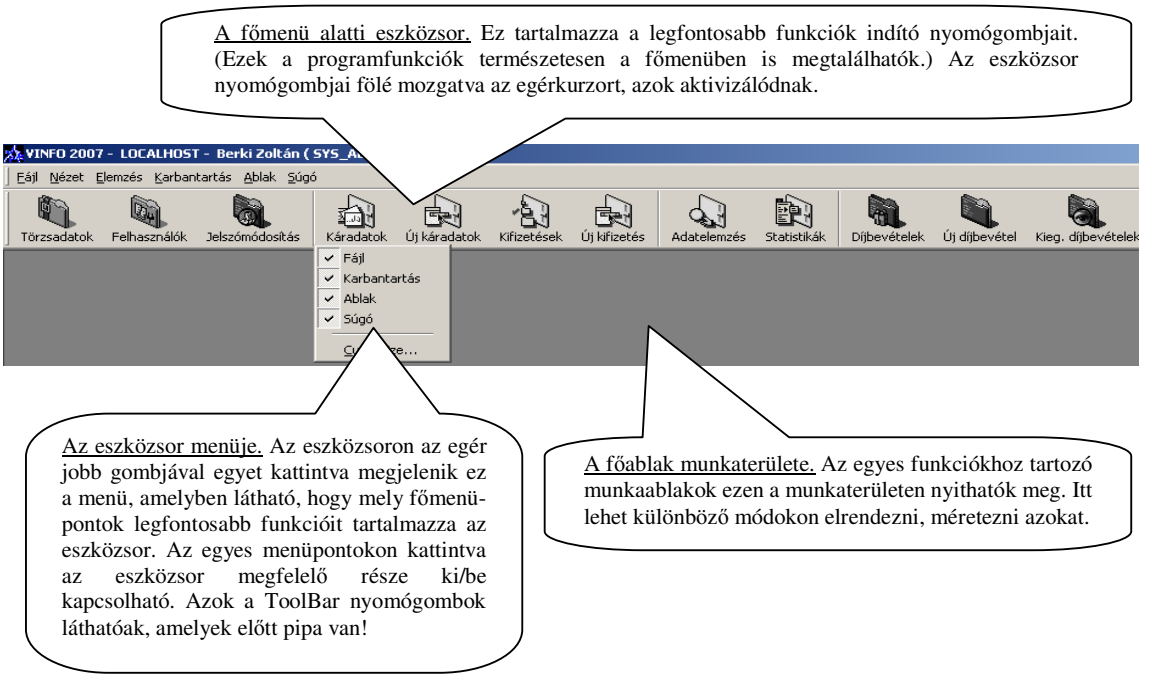
A „Főmenü” tartalmazza az összes indítható funkciót. Ez a menü a bejelentkezett felhasználó jogosultságának megfelelően, csak azokat a menüpontokat tartalmazza, amelyeket az adott jogosultsággal használni lehet, tehát eltérő jogosultságú felhasználókhoz más-más főmenü tartozik. A program főmenüje tehát az adott felhasználó jogosultságához igazodik (testre szabható).

A főmenü feladata nem csak az, hogy abból a megfelelő programfunkciók indíthatók legyenek, hanem az is, hogy ezeket a funkciókat logikus csoportosításban kínálják a felhasználónak.



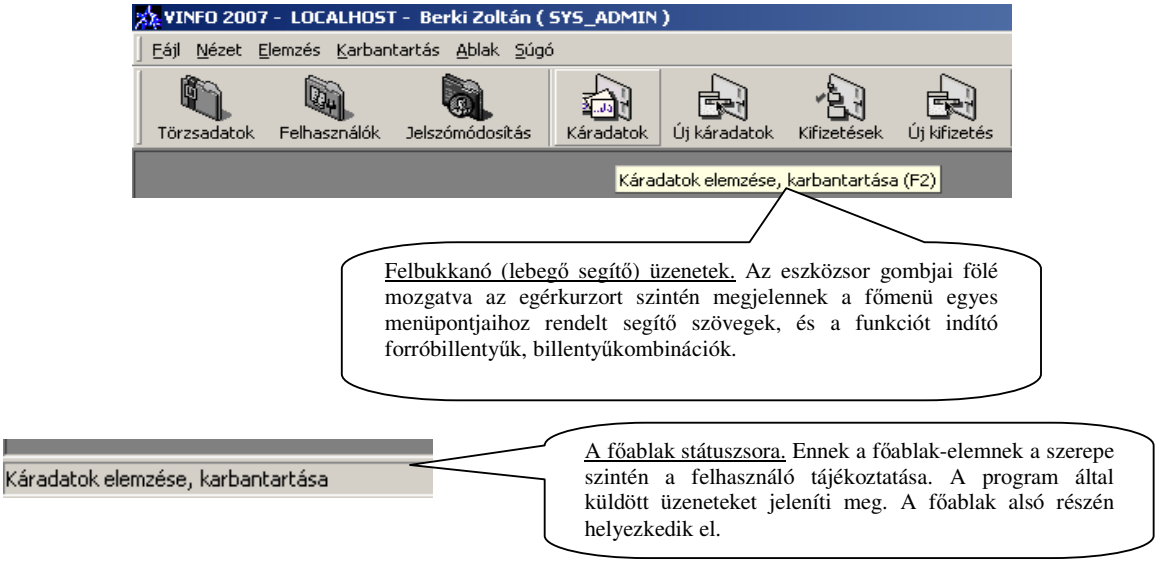

**5.4. ábra. A rendszer főablakának címsora és főmenüje.**

[31] Az eszközkészletnek több elnevezése is van. A gyakorlatban használják még a ToolBar, illetve az eszközsor elnevezéseket is.




**5.5. ábra. A rendszer főablakának eszközsora és munkaterülete.**

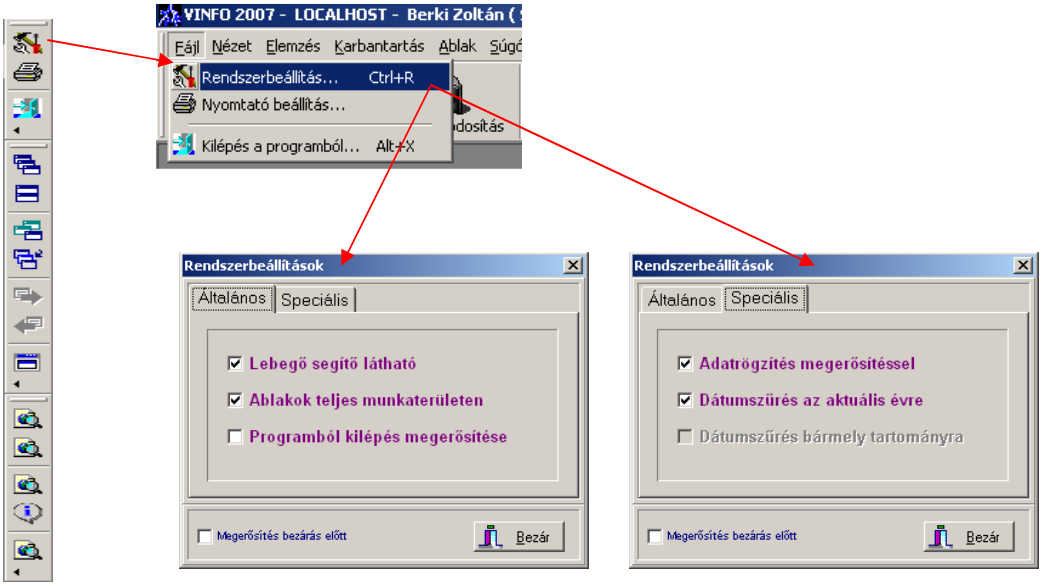
A főmenü legfontosabb menüpontjaihoz úgynevezett forró billentyűk vannak rendelve . Ezek segítségével a billentyűzetről közvetlenül – az egér használata nélkül – lehet indítani a hozzájuk rendelt funkciókat. Minden fontos menüponthoz magyarázó szövegek is tartoznak, amelyek tájékoztatnak arról, hogy az adott menüpont milyen funkciót indít el, hogyan lehet gyorsbillentyűk segítségével is aktiválni ezt a funkciót.




**5.6. ábra. A főablak üzenetei a státuszsoron és lebegő segítő formájában.**

A főablak jobb alsó részén a programfunkciók használata során szintén megjelenhetnek felbukkanó üzenetek. Ezek a szöveges információk **zöld** háttérszínnel rendelkeznek, ha pl. valamely funkció engedélyezése történt meg, **sárga** háttérszín esetén tájékoztató jellegű az üzenet tartalma és **piros** háttér esetén valamilyen hibára utaló szöveg bukkan fel.

### 5.2.2. Testre szabható GUI




**5.7. ábra. A rendszerbeállítások ablak megnyitása, rendszerbeállítások.**

A főmenü „Fájl”menüpontjával, vagy a főablak jobboldalára rendezett eszközkészlet 5.7. ábrán megjelölt nyomógombjával nyitható meg a rendszerbeállítások ablak. Ebben az ablakban az „Általános” fülön a következő funkciókat lehet ki/bekapcsolni:

- Lebegő segítő látható: az 5.6. ábrán szemléltetett felbukkanó segítő megjelenítése a szöveg előtt lévő jelölőnégyzetre történő kattintással letiltható vagy engedélyezhető. Egyes felhasználókat zavar ez a felbukkanó üzenet. Számukra ez a funkció kikapcsolható.

- Ablakok teljes munkaterületen: beállítás a gyerekablakok megnyitásának a módjára utal. Amennyiben ez a funkció engedélyezve van, ezek az ablakok teljes méretben jelennek meg, azaz kitöltik a főablak teljes munkaterületét.

- Programból kilépés megerősítése: funkció beállításával a rendszerből történő kilépés, azaz a főablak bezárása előtt egy megerősítő üzenet jelenik meg. Ennek az opciónak az alkalmazása elsősorban a program használatának a betanulási szakaszában hasznos, amikor egy nem szándékolt kilépést vissza lehet vonni, azaz nem kell újra elindítani a rendszert és újból bejelentkezni.

Megjegyzés: *A rendszerbeállításokat a program elmenti! A megadott beállítások egészen addig érvényben maradnak, amíg azokat újból nem módosítjuk.*

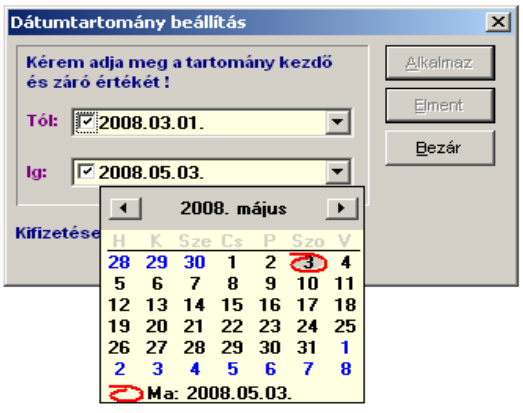
A „Speciális” fülön a következő funkciókat lehet ki/bekapcsolni:

- Adatrögzítés megerősítéssel: opció engedélyezése szintén a betanulási időszakban hasznos. Ekkor minden adatfelvitel és adatmódosítás végén (de még az adatbázisba történő végső rögzítés előtt) egy megerősítő üzenet jelenik meg, amelynek segítségével visszavonható a nem szándékolt művelet-végrehajtás.

- Dátumszűrés az aktuális évre: opció engedélyezésekor a program következő indulásakor a „Függőkárok” és „Díjadatok„ a rögzítés dátuma szerint meg vannak szűrve, azaz csak azok a rekordok láthatók, amelyeket az aktuális évben rögzítettek. Ez jelentősen csökkentheti a szerver leterheltségét, ezzel gyorsítja minden felhasználó munkavégzését és optimalizálja a számítógép-hálózat adatforgalmát, hiszen mindenki számára csak a megszűrt adatokat kell a hálózaton átküldeni. Törekedni kell ennek az opciónak az engedélyezésére! Az esetek többségében ritkán van arra szükség, hogy az egyes felhasználók számára szükséges legyen több év visszamenőleges adathalmaza.

- Dátumszűrés bármely tartományra: opciónak hasonló szerepe van, mint az előzőnek, csak itt a felhasználónak lehetősége van egyedi dátumtartományokat definiálni a függőkárokra, a kifizetésekre, és a díjbevételekre. A függőkárok és díjbevételek dátumtartomány szűrése szinkronizálva van, azaz ennek az opciónak az engedélyezésekor a program úgy indul el, hogy mind a függőkárok, mind a díjadatok ugyanarra a (megadott) dátumtartományra vannak megszűrve. Ez logikus, hiszen az így megszűrt függőkárokhoz tartozó összes kifizetések csak ebben az esetben vethetők érdemben össze a díjbevételekkel. Természetesen mód van arra, hogy ettől a szinkronizált állapottól eltérjünk. Ez úgy tehető meg, hogy a dátumtartomány szűrést a függőkárok ablakból, vagy a díjadatok ablakból indítjuk el.

De hogyan lehet ilyen egyedi dátumtartományokat megadni?




**5.8. ábra. Egyedi dátumtartomány beállítása.**

A tetszőleges dátumtartomány beállítás ablakát szemlélteti az 5.8. ábra. Ebben az ablakban lehet megadni a kívánt tartományt.

Elsőként a dátumtartomány kezdetét (Tól:) célszerű megadni, majd a záró értéket (Ig:). Az ábrán látható naptár megnyitásához mindkét esetben a dátummező jobb oldalán lévő (csúcsával lefelé mutató háromszöget ábrázoló) gombra kell kattintani. Ügyelni kell a dátumtartomány értékeinek helyes megválasztására! Az „Alkalmaz” vagy „Elment” nyomógomb csak akkor lesz aktív, ha a beállítások helyesek.

A dátumtartomány helytelen megadását a program ellenőrzi. Ilyen hiba lehet például, hogy a tartomány kezdete nagyobb, mint a záró érték. Ez természetesen helytelen, hiszen a kezdő dátum nem lehet nagyobb, mint a záró érték.

Az esetleges hibás beállításokról hibaüzenetet küld a rendszer. Helyes tartománymegadás esetén két lehetőség közül választhatunk:

1. az alkalmaz gombbal azonnal aktiváljuk a szűrést, vagy
2. elmentjük és a beállított tartományszűrés a program következő indulásakor lesz aktív.

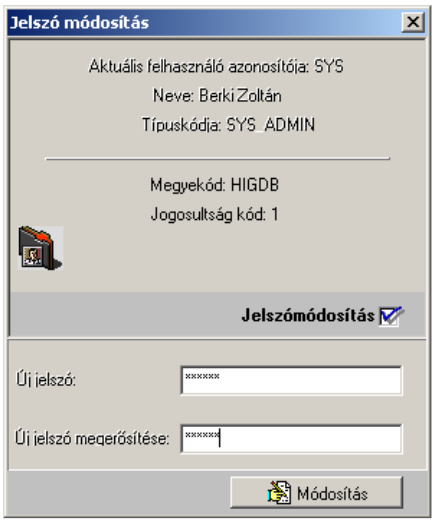
A dátumtartomány szűréséhez kapcsolódó opciók együttes engedélyezése nem lehetséges! Ezek a műveletek egymást kizárják, azaz vagy az aktuális évre történik szűrés vagy egyedi tartományra!

Az természetesen lehetséges, hogy egyedi tartományszűrést választunk és ott az aktuális év kezdő- és záró dátumát adjuk meg. Ez a szűrés is az aktuális év adatait fogja eredményezni, ugyanakkor az egyedi tartományszűrés lesz az aktív funkció mindaddig, amíg ezen nem változtatunk.

Megjegyzés: *A rendszerbeállítások mentésének része a főablak méreteinek, a Windows munkaasztalán elfoglalt pozíciójának a mentése is. Ez azt jelenti, hogy a főablak méreteit és pozícióját elegendő egy alkalommal beállítani és ezek után minden programindítás után ugyanazokkal a méretekkel és pozícióval fog megjelenni. Ezek a beállítások egészen addig érvényesek lesznek, amíg azokat nem módosítjuk. A főablak egy bizonyos (optimális) méretnél kisebbre nem állítható! Erre azért van szükség, hogy a benne megnyitott ablakok adattartalma mindig értelmezhető legyen!*

### 5.2.3. Felhasználó azonosítása a rendszerben, jelszómódosítás

Minden felhasználónak lehetősége van saját adatainak a megtekintésére és jelszavának a módosítására. A bejelentkezés után az eszközsor „Jelszómódosítás” nyomógombjának használatával vagy a CTRL+F12 billentyűkombinációval lehet megnyitni a bejelentkezett felhasználó rendszerbeli adatait tartalmazó ablakot. A program főmenüben ez a funkció a *Karbantartás**Rendszerparaméterek**Jelszómódosítás*\ útvonalon érhető el.

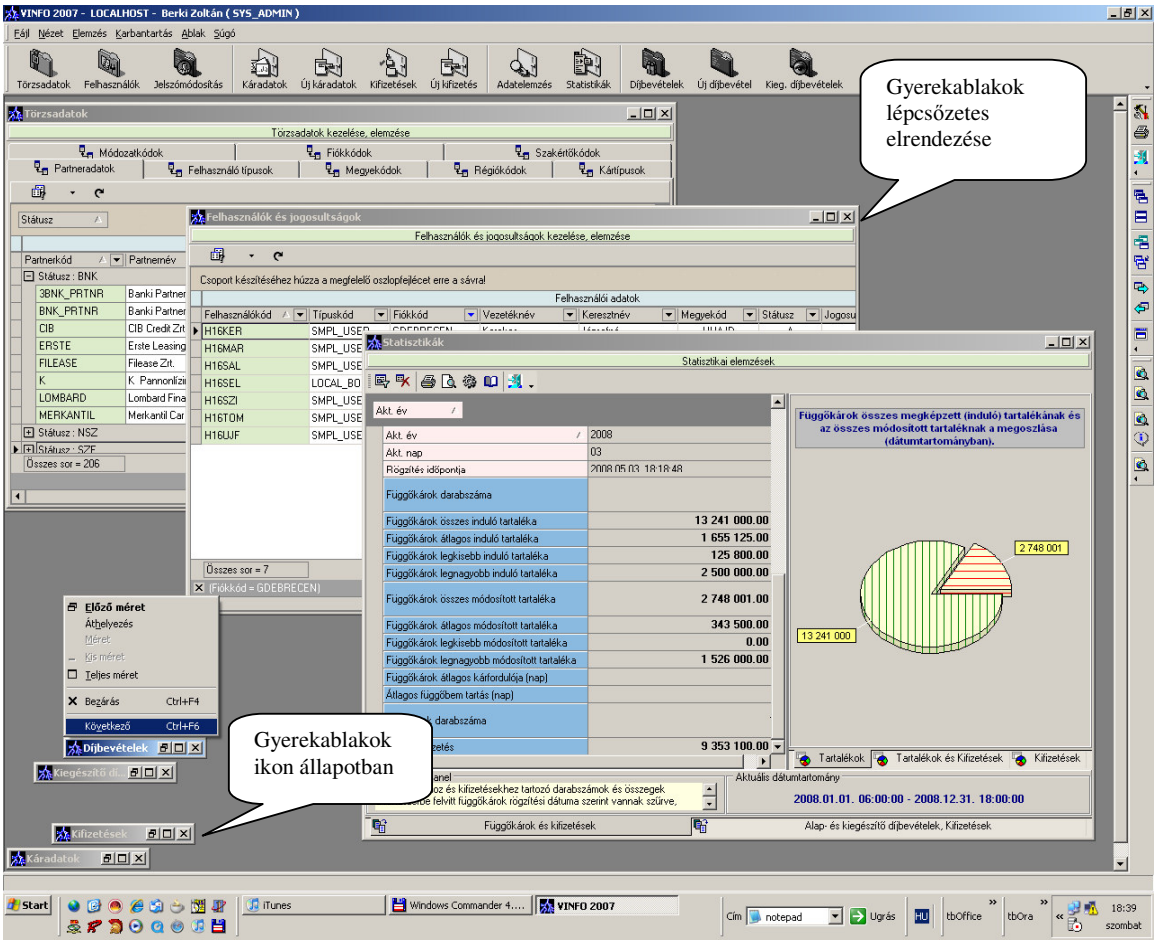



**5.9. ábra. Jelszómódosítás.**

Az 5.9. ábrán bemutatott „Jelszómódosítás” ablakban megjelennek az aktuális felhasználó publikus rendszerbeli adatai. Ebben az ablakban van mód az éppen érvényes jelszó megváltoztatására. A jelszó módosításához első lépésként a „Jelszómódosítás” jelölő négyzetre kell kattintani. Ekkor az új jelszó megadására szolgáló adatbeviteli mezők aktivizálódnak és be lehet gépelni az új jelszót.

A jelszónak pontosan hat karakter hosszúnak kell lennie! Kerülni kell a speciális karakterek használatát! Az új jelszó megerősítése után, amennyiben a két begépelt jelszó megegyezik, és eleget tesz a rendszer által elvárt feltételeknek, a „Módosítás” nyomógomb aktívvá válik és rögzíthető az új jelszó. A következő bejelentkezésnél már ezt az új jelszót kell használni!

### 5.2.4. Adatmegjelenítés a főablakban, gyerek-ablakok elhelyezése, kezelése




**5.10 ábra. A rendszer főablaka és gyerekablakai.**

A rendszer fő munkaablakai (gyerekablakai):

- Törzsadatokat kezelő ablak,
- Felhasználói adatokat kezelő ablak.
- Káradatokat kezelő ablak.
- Kifizetéseket kezelő ablak.
- Adatelemzéseket kezelő ablak (a **VIM** = **V**ezetői **I**nformációs **M**odul része)
- Statisztikákat kezelő ablak . (a **VIM** = **V**ezetői **I**nformációs **M**odul része)
- Díjbevételeket kezelő ablak.
- Kiegészítő díjbevételeket kezelő ablak.

A felsorolt munkaablakok közös tulajdonsága, hogy a főablakban nyílnak meg, méretezhetőek, lépcsőzetesen és mozaikszerűen (egymás alá igazítva) is elrendezhetők. Ikon állapotban a főablak munkaterületén tetszőlegesen helyre mozgathatók és onnan szükség esetén újra normál vagy teljes méretben megnyithatók. Az 5.9. ábra szemléletesen mutatja ennek az ablaktípusnak az elrendezési, méretezési lehetőségeit.

A gyerekablakok kezelésével kapcsolatos összes funkció az ablak menüben található meg. Ennek a főmenünek az egyes menüpontjai akkor aktivizálódnak, amikor legalább egy ilyen munkaablakot megnyitunk. Az 5.9. ábrán bemutatott esetben, amikor egyszerre több ilyen típusú ablak meg van nyitva, az egyes ablakok közötti váltáshoz jól használható az F6 billentyű az előre mozgáshoz és a Ctrl+F6 billentyű kombináció a visszafelé történő mozgáshoz. Ezeknek a típusú ablakoknak a kezeléséhez szükséges összes funkciók az 5.9. ábrán látható főablak jobb oldalán lévő eszközkészletről is közvetlenül elérhetők.

A munkaablakok fontos tulajdonsága, hogy egy bizonyos (optimális) méretnél kisebbre nem méretezhetőek. Erre a megoldásra azért van szükség, hogy az ablakokban megjelenített adatok mindenkor értelmezhetőek legyenek. Amennyiben egy ilyen típusú ablak már meg van nyitva, újból nem nyitja meg a program, csak ráhelyezi az input fókuszt, azaz a többi ablak elé kerül, ez lesz az aktív ablak.

Fontos tulajdonsága a programnak, hogy ezekből a munkaablakokból minden művelet közvetlenül elvégezhető azokon az adatokon, amelyeket megjelenítenek. Mit jelent ez konkrétan?

Például a függőkárokat megjelenítő „Káradatok” elnevezésű ablakba minden olyan funkció be van építve, amellyel a káradatokra érvényes összes műveletet el lehet végezni. A felhasználói munka kényelmesebbé és egyszerűbbé tétele érdekében történik mindez, hiszen így egyetlen ablakból minden elérhető, minden művelet végrehajtható; meg kell jegyezni, hogy ez természetesen az adott felhasználó jogosultságának a keretein belül érvényes.

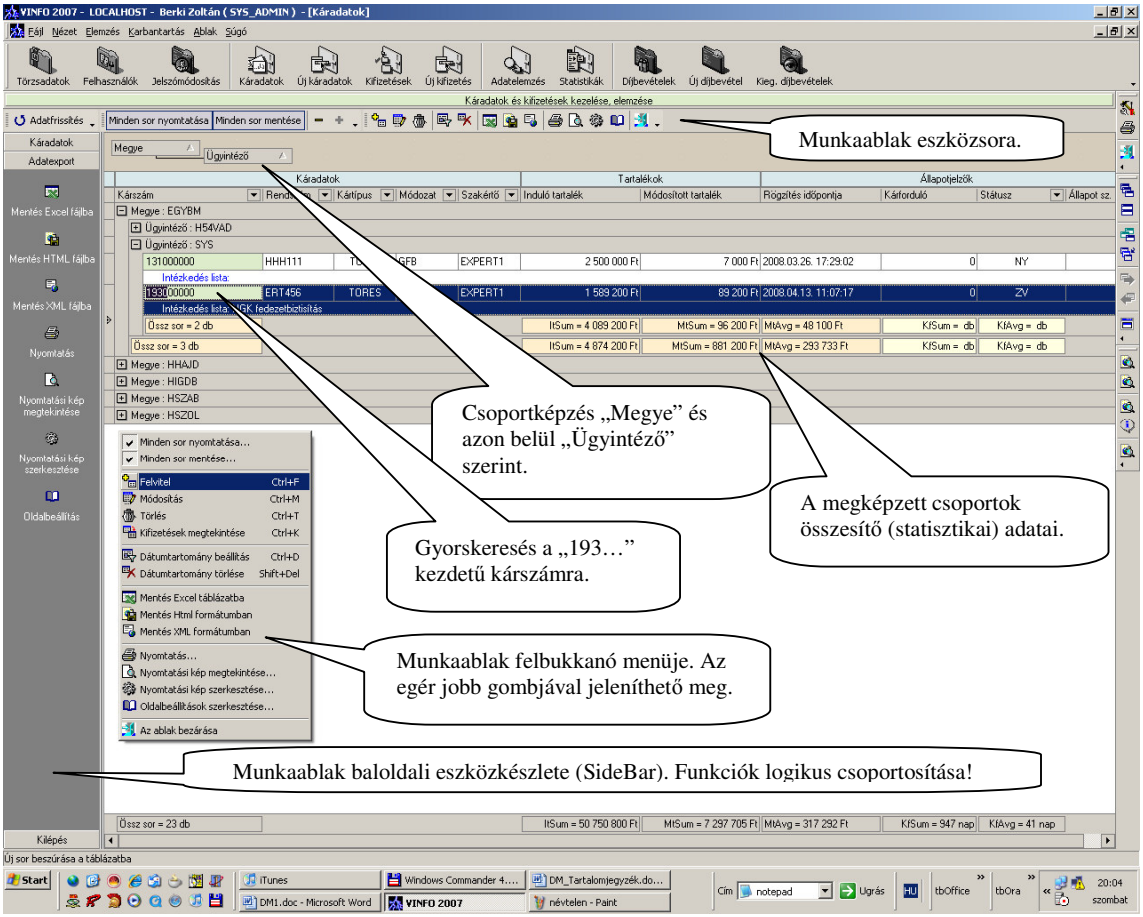
Milyen műveleteket lehet végezni egy ilyen munkaablakban? Minden fontos művelet a „Káradatok” példáján keresztül kerül bemutatásra. A következő táblázat a káradatokra (is) érvényes műveletek listáját tartalmazza.

| **Műveletek a káradatokon** |
|---|
| 1. **Felvitel** (módosított tartalék = induló tartalék, függőben tartás = 30 nap, kárforduló = 0 nap) |
| 2. **Módosítás** (függőben tartás = 30 nap intézkedés esetén) |
| 3. **Törlés** (csak akkor, ha nem sérti meg a referenciális integritást!) |
| 4. **Tartalékmódosítás** (automatikus, vagy manuális) |
| 5. **Lezárás** (módosított tartalék = 0, függőben tartás = 0 nap, kárforduló számítása a rögzítés dátumából és a rendszerdátumból) |
| 6. **Megnyitás** (módosított tartalék = induló tartalék, függőben tartás = 30 nap) |
| 7. **Kifizetés** (módosított tartalék – kifizetett összeg, kevés tartalék esetén automatikus vagy manuális tartalékmódosítás meghívása) |
| 8. **Kifizetés módosítása** (módosított tartalék megfelelő karbantartása) |
| 9. **Kifizetés törlése** (módosított tartalék + kifizetett összeg) |
| 10. **Gyorskeresés** (adatok gyorskereső metódusa) |
| 11. **Rendezés** (adatok rendezése növekvő vagy csökkenő sorrendben, egymásba ágyazható módon) |
| 12. **Csoportképzés** (egymásba ágyazható módon!) |
| 13. **Szűrés** (több – egymással logikai ÉS kapcsolatban lévő – feltétel szerint!) |
| 14. **Dátumtartomány szűrés** (a rögzítés időpontja szerint, csak az adott tartományba eső rekordok láthatók) |
| 15. **Dinamikus listakészítés** (a felhasználó által a program futása közben rendezett, megfelelően szűrt és szelektált adatok listázása, szerkeszthető nyomtatási képpel) |
| **16. Statisztikai elemzések** (tartalékelemzés, kifizetés-elemzés, leterheltség vizsgálat) |
| **17. Adatexport** (.XLS, .HTML és .XML formátumban, VIM[32] garfikonok esetén meta-fájl és .BMP formátumban is!) |
| **18. Exportált adatok közzététele az intraneten** |

[32] VIM = **V**ezetői **I**nformációs **M**odul.

### 5.2.5. Munkaablakok tulajdonságai, kereső és rendező algoritmusok használata

Tekintsük először a 10-18 funkciókat!




**5.11 ábra. Műveletek a káradatokon.**

10. Gyorskeresés. A táblázat megfelelő oszlopára mozgatva a sávkurzort a keresett érték folyamatos begépelésével a program folyamatosan pozícionál a begépelt értékkel egyező sorára a táblázatnak. Például a kárszám oszlopon állva egy kárügy megkereséséhez folyamatosan elkezdjük begépelni az adott kárügyhöz rendelt kárszámot. Minden egyes karakter begépelésekor a program rögtön arra a sorára pozícionál a táblázatnak, ahol az eddig begépelt karakterekkel egyező értéket talál. Ha a begépelés során nincs további egyezés, akkor a keresés megáll. Az alkalmazott gyorskereső algoritmus előnye, hogy a táblázat minden oszlopának az értékeire képes keresni, csak rá kell pozícionálni a megfelelőre, és el kell kezdeni a keresendő érték begépelését.

11. Rendezés. A táblázat oszlopainak a fejlécére (ahol az oszlopok neve szerepel) egyszerűen rá kell kattintani. Ekkor a fejlécben megjelenik egy csúcsával felfelé (vagy lefelé) mutató háromszög, ami jelzi, hogy az adott oszlop értékei növekvő (vagy csökkenő) sorrendbe vannak rendezve. Újabb kattintással a rendezés iránya megváltoztatható. A rendezés egymásba ágyazható művelet. Ez azt jelenti, hogy egyszerre több oszlop értékei szerint is történhet rendezés. Ilyenkor megadjuk az első oszlop megfelelő rendezését, majd a Shift billentyű lenyomása mellett rákattintunk annak az oszlopnak a fejlécére, aminek az értékei a rendezésben a másodlagos feltételt adják.

### 5.2.6. Az adatok csoportosítása és szűrése

12. Csoportképzés. Az 5.10. ábrán először megyék szerint képeztünk egy csoportot, majd a megyéken belül még a kárügyintézők szerint is csoportosítottuk a káradatokat. A művelet elvégzéséhez először a „Megye" oszlop fejlécét kell a táblázat feletti sávra vonszolni,[33] majd ugyanezt a műveletet el kell végezni az „Ügyintéző" oszlop fejlécével. Az egyes csoportok előtt megjelenő + vagy – jelekre kattintva a csoportok becsukhatók vagy kinyithatók. Az ablak eszközsorán megtalálható egy + és egy – ikonnal ellátott nyomógomb. Ezek az összes csoportra érvényesek, azaz minden csoportot képesek kinyitni vagy becsukni. Az ablak bármely belső pontján a jobb egérgombbal kattintva az 5.1. ábrán látható un. felbukkanó menü is elérhető. Ebben is minden funkció megtalálható és elindítható. Az egyes csoportok utolsó sorában (soraiban) az adott csoporthoz tartozó összesítő adatok is megtekinthetők. A táblázat legalsó sorában pedig az összes csoporthoz tartozó statisztikák láthatók.

13. Szűrés. A táblázatnak vannak olyan oszlopfejlécei, amelyekben listadoboz található. Ennek a listának a legörgetésével kiválasztható az a listaelem, ami szerint meg akarjuk szűrni a táblázat sorait. A szűrőfeltételek is egymásba ágyazhatók, azaz egyszerre több szűrőfeltétel is definiálható. Ezek a feltételek logikai „ÉS" kapcsolatban vannak egymással. Például, ha elsőként megszűrjük az adatokat a totálkárokra (TOTAL listaelem), majd a CASCO módozatokra, akkor csak a CASCO biztosítással rendelkező és totálkáros kárügyeket fogjuk látni a táblázatban. A szűrőfeltételek ilyenkor a táblázat alsó sorában kiíródnak és itt is meg lehet szüntetni a beállított szűrőfeltételek hatását.

A dátumtartomány szűrésének minden fontos lépése az 5.8. ábra funkcióinak a bemutatása kapcsán ismertetve volt.

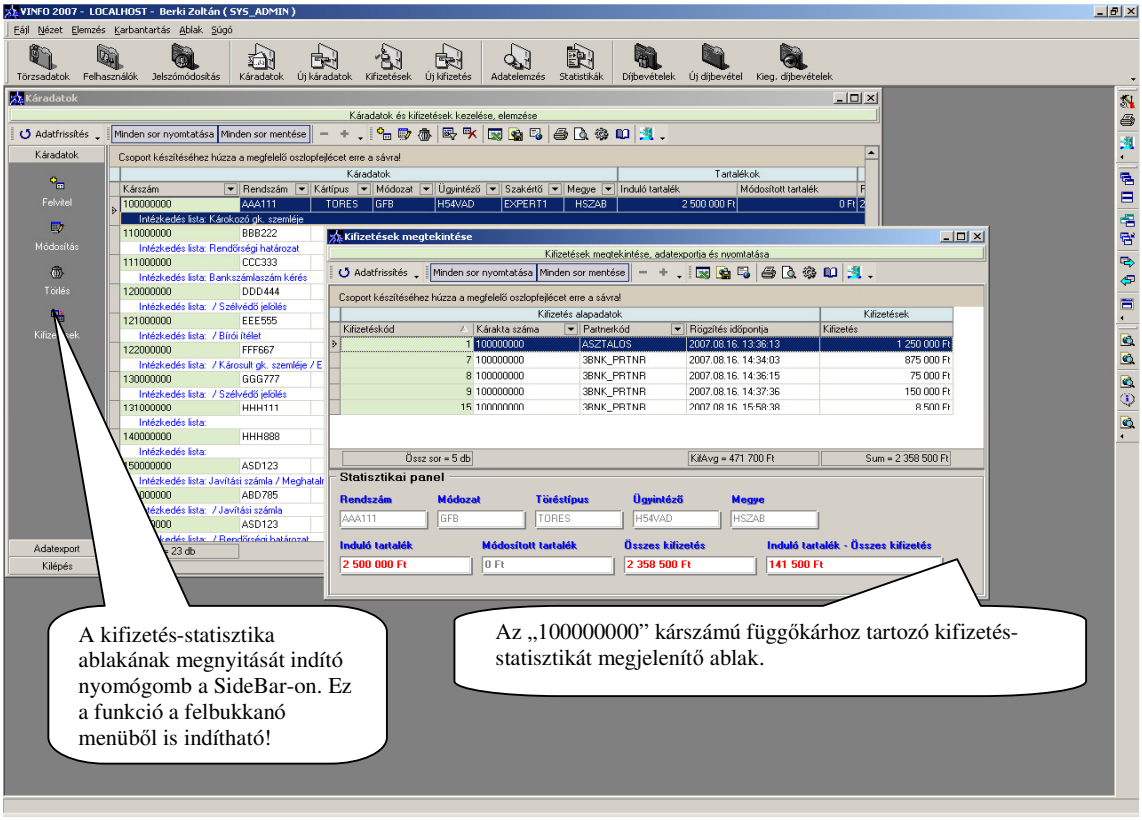
### 5.2.7 Dinamikus listakészítés, nyomtatási kép megjelenítése, nyomtatás

15. Dinamikus listakészítés. Ennek a műveletnek az alkalmazása rendkívül egyszerű. Először elvégezzük az adatok rendezését, csoportosítását, szűrését annak megfelelően, hogy mit szeretnénk a listában látni, majd alkalmazzuk a nyomtatás funkciót vagy nyomtatás előtt a „Nyomtatási kép megtekintése" funkcióval megnézzük az elkészült listát. A dinamikus listakészítés azt jelenti, hogy a program futása közben a felhasználó aktuális igényei alapján készül el a lista. Tetszés szerint rendezheti, csoportosíthatja, szűrheti az adatokat és a megjelenő eredményt azonnal képes listázni. Például megszűrhetjük az adatokat úgy, hogy csak a CASCO módozatok- és azon belül csak a totálkárok legyenek láthatóak és az eredményt abban a formában, ahogyan az megjelenik a táblázatban, azonnal listázni lehet. Nincsenek előre legyártott (módosíthatatlan) listák! A dinamikus listák nyomtatási képét szerkeszteni is lehet. A listában színes kiemelések és formázások is lehetnek az esetleges színes nyomtatáshoz.

### 5.2.8 Statisztikai adatok elemzése

16. Statisztikai elemzések. Minden felhasználó elkészítheti a saját jogosultságának megfelelő statisztikákat. Az előző pontokban ismertetett műveletek nélkül is látja például a függőkár darabszámait. Megtekinthető az egyes kárügyekhez tartozó kifizetések statisztikája; ehhez a megfelelő kárügy során állva alkalmazhatjuk a Ctrl+K billentyűkombinációt.

[33] Az egér bal gombját az adott oszlopon lenyomva tartva ráhúzzuk a csoportképző sávra. Ez egy un. Drag End Drop technológia.




**5.12. ábra. Kifizetés statisztika elemzése.**

A kifizetés-statisztikákra is minden korábban megadott (keresési, rendezési, listakészítési, stb.) művelet érvényes. Az 5.11. ábrán szereplő kifizetéseket megjelenítő munkaablak az egyes megyékhez tartozó kárügyintéző munkatársak számára jelent eszközt a kifizetések elemzéséhez, ti. ezek a kollégák a globális elemzéseket nem tekinthetik meg!

Milyen szolgáltatásokkal segíti még a káradatok munkaablak a felhasználókat?

1. A rendszer külön színnel kiemeli számára azokat a kárügyeket, amelyekben feltétlenül intézkednie kell. Azok a kárakták, amelyekben a függőben tartás un. állapotszámlálója eléri a 7 napot, feltétlenül döntést kell hozni! Ezeknek a kárügyeknek a státuszát ZV = Zárásra Vár állapotúra állítja a rendszer. Célszerű tehát napi szinten elvégezni egy szűrést a ZV állapotú aktákra és azonnal látható, hogy aznap mennyi feltétlenül elvégzendő munka van.

2. A kárforduló átlagából megítélhető bizonyos szinten a munkavégzés minősége. A túlságosan magas átlagértékek hosszú kárügyintézést jelentenek, ami sem az ügyfélnek, sem a Biztosító Társaságnak nem jó!

3. A tartalékok megjelenő összesítő adataiból észrevehető például, hogy van-e egy bizonyos időszakon belül kiugróan magas tartalék megképezve.

Ezeket az egyszerű statisztikákat a függőkárokat kezelő ablak táblázatának alsó sorában is meg lehet tekinteni. A kárrendezési vezetők rálátnak az adott terület összes kifizetési adataira is. Ezek elemzése a fenti szempontok alapján időszakosan elvégzendő feladat!

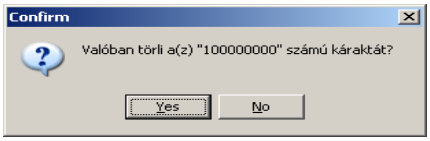
### 5.2.9. Adatkarbantartás (felvitel, módosítás, törlés)

Tekintsük az 1-9 funkciókat!

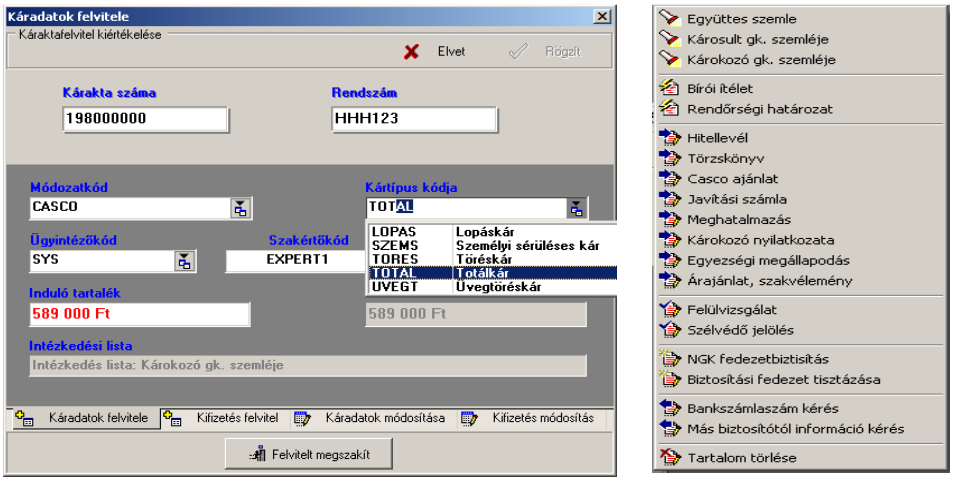
A függőkárok és kifizetések felvitele és módosítása egy munkaablakban (különböző füleken) végezhető el. Minden kárrendezési vezető és kárügyintéző rögzíthet függőkárokat az adatbázisban, a felrögzített függőkárokhoz kifizetéseket is felvihet. Ezek a munkatársak módosíthatják a káradatokat, lezárhatják a kárügyet, de a kifizetések módosítása és a lezárt károk megnyitása számukra nem engedélyezett. Szintén tiltott funkció a törlés. Ezzel a jogosultsággal csak a régió vezetői és a rendszergazda rendelkezik. A „Felvitel", „Módosítás", „Törlés" műveletek a korábban ismertetett módszerek (a státuszsorból, felbukkanó menüből, SideBar-ról) bármelyikével indíthatók, van azonban a programban ezeknek a funkcióknak a használatához egy egységesen bevezetett billentyűkombináció csoport definiálva:

- Ctrl+F: Felvitel, azaz új sor beszúrása a táblázatba.
- Ctrl+M: Módosítás, azaz egy kiválasztott sor adatainak a módosítása.
- Ctrl+T: Törlés, azaz egy konkrét sor törlése a táblázatból.

A törlési műveletet minden esetben megelőzi egy megerősítő kérdés (5.12. ábra)! A törlés indítása előtt meg kell keresni (pl. a keresési funkcióval) azt a sort, amelyet törölni szeretnénk, majd el kell indítania a törlés funkciót. Az adott sor törölhetőségének az összes feltételét megvizsgálja a rendszer és csak ezek teljesülése esetén kerül végrehajtásra a művelet! Ellenkező esetben nem törlődik a kijelölt sor és hibaüzenetet kap a felhasználó erről a tényről!




**5.13. ábra. A törlési művelet megerősítése.**




**5.14. ábra. Káradatok felvitele és az intézkedési lista menüje.**

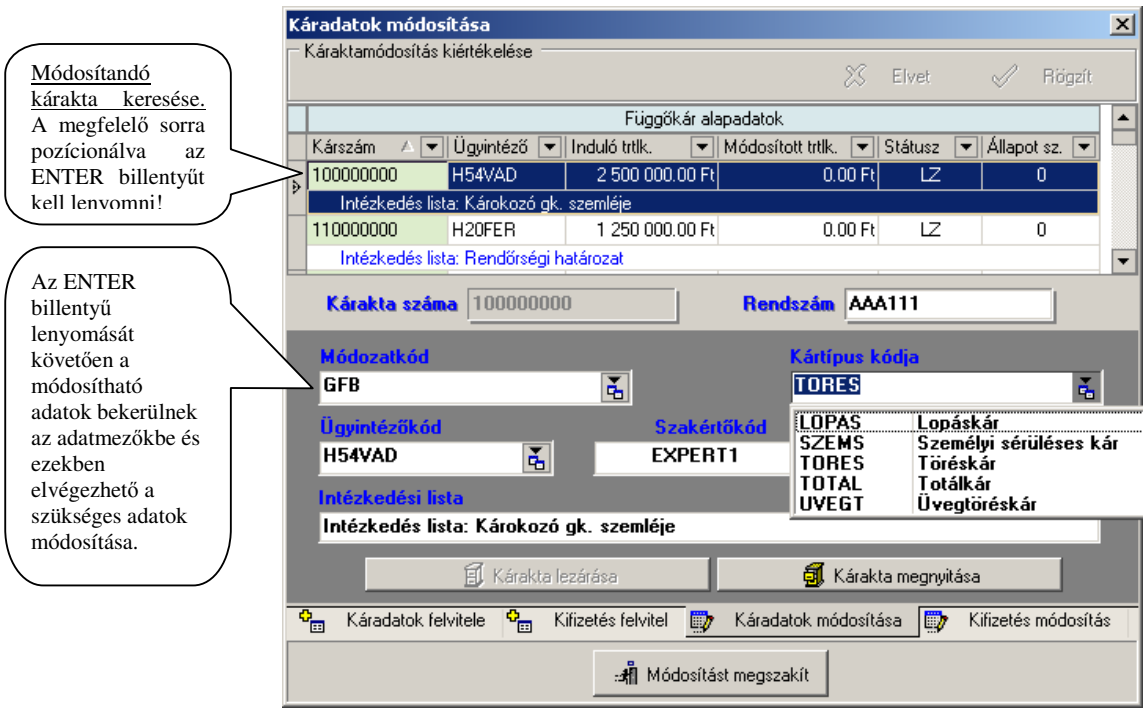
A káradatok felvitelét szemlélteti az 5.13. ábra. Az egyes adatok begépelése után az ENTER billentyű lenyomásával is tovább lehet lépni a következő adatbeviteli mezőre.

A törzsadatokat megjelenítő listadobozokban (lásd 5.13. ábra) is érvényes a gyorskereső funkció, azaz például a „Kártípus kódja” nevű mezőben a „TOTAL” kódú listaelem kiválasztásához elegendő a „tota” szórészlet begépelése a keresett értékre történő pozícionáláshoz. Az ENTER billentyű lenyomásával el lehet fogadni a megkeresett értéket. A Shift+Del billentyűkombinációval törölhető a megadott érték és újrakezdhető a keresés! Minden törzsadat ilyen (a példában bemutatott) listadobozban jelenik meg. Ezeknek az adatoknak a begépelésére tehát nincs szükség, csak a megfelelő listaelem kiválasztása szükséges!

Csak a függőkárokra jellemző intézkedési lista elemeinek a kiválasztása az 5.13. ábra jobb oldalán látható felbukkanó menü segítségével történik. A menü az ablak területén bárhol megjeleníthető a jobb egérgomb használatával. Ebben a menüben a megfelelő menüpont kiválasztásával bekerül az „Intézkedési lista” elnevezésű mezőbe a szükséges szöveges megjegyzés. A menü többszöri alkalmazásával egyszerre több intézkedés is összefűzhető az adatmezőben.

A „Káradatok felvitele” művelet során az ablak jobb felső részében található „Rögzít” nyomógomb csak akkor aktivizálódik, ha minden szükséges adat a megfelelő formában lett megadva. Ekkor rögzíthetők a megadott adatok (egy kárügyként) az adatbázisban. Ez a „Káradatok” táblában azonnal egy új sorként jelenik meg.

Az adatrögzítés folyamata és módszerei a „Kifizetések”, a „Díjadatok” és a „Kiegészítő díjak” felvitelekor is teljesen azonos, természetesen az adott táblára specifikus input adatok megadásával!




**5.15. ábra. Káradatok módosítása.**

A „Káradatok” módosítása az 5.14. ábrán szemléltetett módon végezhető el. Itt lehet az adott kárügy lezárását vagy megnyitását is megcsinálni.

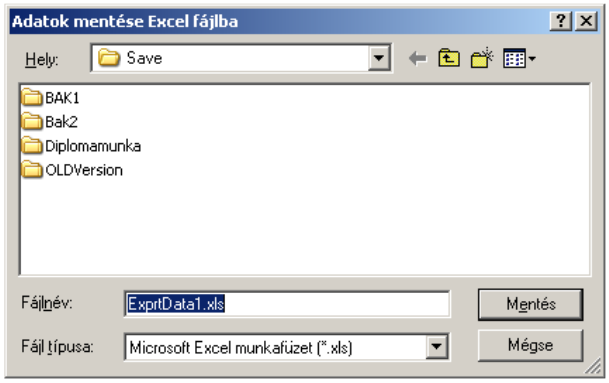
A „Kárakta lezárása” és a „Kárakta megnyitása” nyomógombok attól függően aktívak vagy inaktívak, hogy a „Függőkár alapadatok” táblázatban éppen milyen állapotú kárügy során áll a sávkurzor, azaz éppen melyik az aktuális sora ennek a táblázatnak. Az 5.14. ábrán egy éppen lezárt (LZ) státusszal rendelkező sor az aktuális. Ebből következik, hogy a „Kárakta megnyitása” nyomógomb aktív. Ennek a konkrét kárügynek a módosítása csak akkor végezhető el, ha először megnyitjuk, ti. lezárt kárügyben semmilyen intézkedés nem tehető!

A kárakta megnyitásakor a módosított tartalék mezőbe bekerül az induló tartalék összege, az akta státuszának értéke újra nyitott (NY) lesz és a függőben tartás állapotszámlálója az induló 30 napos értéket kapja meg.

Az adatmódosítás módszere a „Kifizetések” és a „Díjadatok” tekintetében is hasonló!

### 5.2.10 Az adatok exportja három formátumban

A káradatok exportja XLS, HTML és XML formátumban is lehetséges. Ezeknek a műveleteknek az indítása szintén több módon lehetséges.




**5.16. ábra. Adatexport Excel formátumban.**

Az adatexport funkció indítása előtt célszerű a táblázat rekordjait úgy rendezni, csoportosítani stb., ahogyan az exportálás után azokat látni szeretnénk és ezután kell elvégezni a mentést az 5.15. ábra mentési dialógusablakának a segítségével.

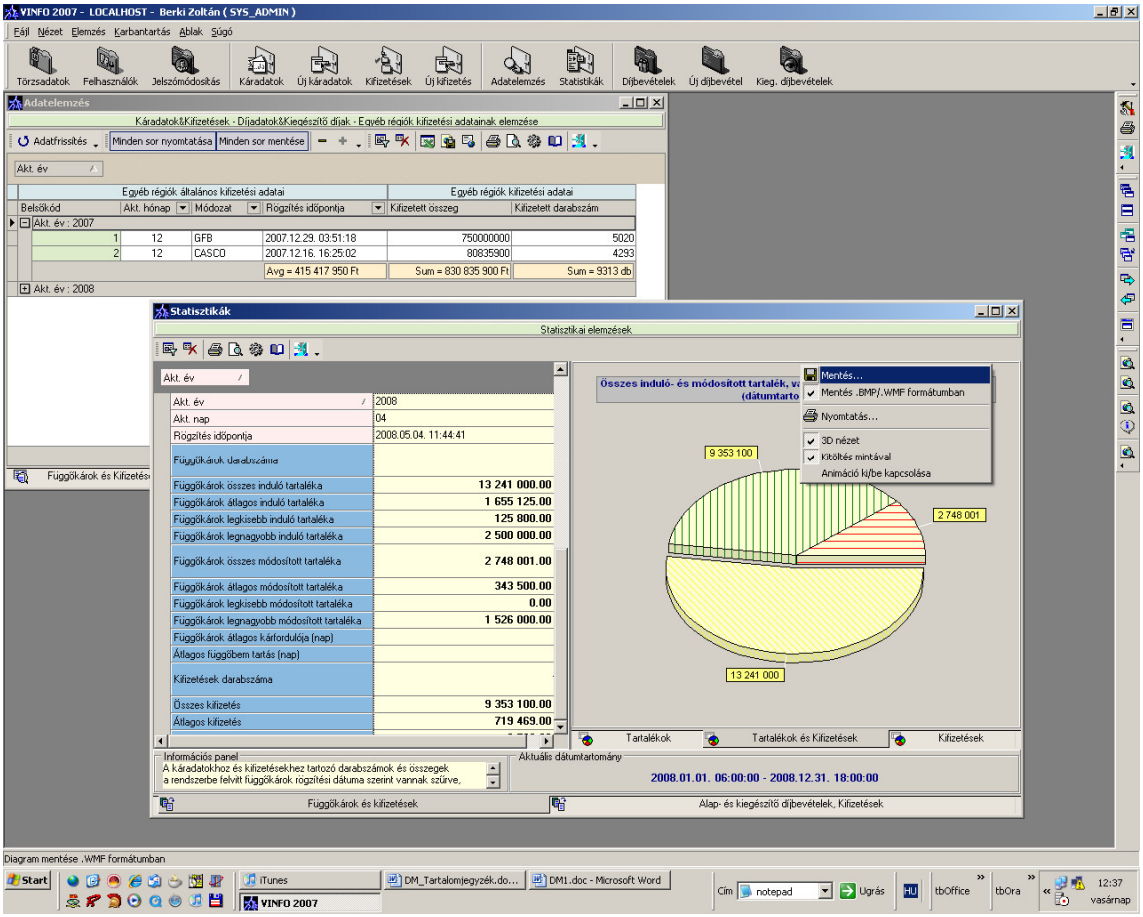
### 5.2.11 Vezetői Információs Modul

A VIM = Vezetői Információs Modulhoz két munkaablak tartozik.

1. Adatelemzés: károkkal, kifizetésekkel és díjadatokkal kapcsolatos globális elemzési felület, melyhez csak a vezetői szintnek van hozzáférése.

2. Statisztikák: a káradatok, kifizetések és díjadatok globális statisztikáit lehet ebben a munkaablakban megtekinteni. Ennek az ablaknak jellemzője, hogy az adatokat grafikonok formájában is ábrázolja.

Mindkét ablakban a korábban ismertetett műveletek érvényesek! Ezek az adatok azonban nem publikusak, így az elkészített listákkal, mentésekkel rendkívül körültekintően szükséges bánni!




**5.17. ábra. A VIM-modul munkaablakai.**

A VIM-modul ablakai a főmenüből és a főmenü alatti eszközkészletről is megnyithatóak. Legegyszerűbben azonban az F11 („Adatelemzések”) és a CTRL+F11 („Statisztikák”) gyorsító billentyűkkel lehet a funkciókat indítani.

## 5.3. A programból történő kilépés

A programból történő kilépés akkor lehetséges, ha az adatkarbantartáshoz kapcsolódó ablakok valamelyike nincs megnyitva, ti. ezek az ablakok úgy vannak elkészítve, hogy bezárásukig csak azok a funkciók végezhetőek el, amelyek az adott ablakban definiálva vannak. Erre azért van szükség, mert a felviteli és módosítási műveletek esetében nincs köztes fázis, azaz vagy teljesen befejezzük a megkezdett adatrögzítést, adatmódosítást vagy teljesen elvetjük.

A többi – az 5.2.4. szakaszban megadott – munkaablak (azaz a gyerekablakok) esetén nem szükséges mindegyiket bezárni a kilépés előtt, azokat a program ellenőrzi és bezárja, mielőtt a végső kilépés megtörténne.

A program a munka befejezésekor megszünteti az adatbázis kapcsolatot is. Fontos tehát, hogy aki huzamosabb ideig nem végez a rendszerben semmilyen munkát, az lépjen ki a programból, mert ezzel a szerveren nem foglal le feleslegesen erőforrásokat!

# 6. fejezet ÖSSZEGZÉS

## 6.1. A rendszerfejlesztés tapasztalatainak összefoglalása

Legfontosabb tapasztalata a rendszer bevezetésének, hogy a rendszertervezés folyamatába be kell vonni a programrendszer leendő felhasználóit. Valóban – nem csak a szándékok, vagy csak a szavak szintjén – az Ő igényeiknek kell a középpontban állni.

A rendkívül gyorsan változó gazdasági, piaci, stb. (egyszóval környezeti) feltételek mellett, kizárólag olyan rendszerfejlesztési módszerek és eszközök alkalmazását látom relevánsnak, amelyek lehetővé teszik, hogy a rendszeresen érkező módosítási igényeket kezelni lehessen.

Kiemelem a hardver és szoftver erőforrások kihasználásának, optimalizációjának a tervezését. Manapság bővelkedünk az IT erőforrásokban, minden tárolókapacitásnál van nagyobb, minden sávszélességet meg lehet növelni, félévenként jelennek meg új fejlesztői platformok, stb. Megítélésem szerint ezen a területen is törekedni kell az optimális megoldások megtalálására, és ennek megfelelően adekvát megoldásokat kell alkalmazni, egyébként nagy eséllyel pazarló erőforrás-kihasználás lesz az eredmény.

## 6.2. Köszönetnyilvánítások

Köszönöm témavezetőmnek, Márton Ágnes tanárnőnek a diplomamunka elkészítése során végzett áldozatos munkáját.

Köszönöm a Matematikai Intézetben dolgozó minden tanáromnak, hogy értékes tudást szerezhettem tőlük. Legfőképpen azonban azt köszönöm, hogy távlatokat nyitottak meg előttem.

Köszönöm feleségemnek, egyben mentoromnak, tanáromnak az egyetlen igaz barátomnak, hogy mellettem áll. Köszönöm gyermekeimnek, Zoltánnak, Péternek, Balázsnak és Gábor Jenőnek a türelmet és kitartást a nehéz órákban. Biztos vagyok benne, hogy megérte.

# IRODALOMJEGYZÉK

[1] Raffai Mária: UML2 Modellező Nyelvi Kézikönyv
Platina Nyomda és Kiadó, 2005

[2] Angster Erzsébet: Az objektumorientált tervezés és programozás alapjai
Unified Modelling Language
Angster Erzsébet, 1997

[3] Ian Sommerville: Szoftver-rendszerek fejlesztése
Panem, 2002

[4] Andrew S. Tanenbaum – Maarten van Steen
Elosztott rendszerek
Alapelvek és paradigmák
Panem, 2004

[5] Gábor András – Juhász István
PL/SQL programozás
Alkalmazásfejlesztés Oracle 10g-ben
Panem, 2007

[6] Kende Mária – Nagy István
ORACLE példatár
SQL, PL/SQL
Panem, 2005

[7] Kende Mária – Kotsis Domokos – Nagy István
Adatbázis-kezelés az ORACLE rendszerben
Panem, 2002

[8] Marco Cantu
Delphi 5
Kiskapu, 2000

[9] Borland Delphi for Windows: User's Guide
Borland™, 1995

[10] Borland Delphi for Windows: Database Application User's Guide
Borland™, 1995

[11] Ben Lurie – Peter Laurie
Apache
Kossuth, 2001

[12] Peter Moulding
PHP haladóknak
Fekete Könyv
Perfact-pro, 2001

# IRODALOMJEGYZÉK

[13] World Wide Web
HTML 3
Dr. Füstös János
Szak Kiadó, 1996

[14] Glenn Rowe
JAVA programozás
Panem, 2002

[15] Christian MacAuley – Paul Jobson
JavaScript programozói referencia
Panem, 2003

[16] Gábor András– Gunda Lénárd – Juhász István
Kollár Lajos – Mohai Gábor – Vágner Anikó
Az ORACLE és a web
Panem, 2003

# ÁBRAJEGYZÉK

## 2. fejezet FELADATSPECIFIKÁCIÓ



## 3. fejezet LOGIKAI RENDSZERTERVEZÉS



## 4. fejezet RENDSZERDOKUMENTÁCIÓ



## 5. fejezet FELHASZNÁLÓI DOKUMENTÁCIÓ