Szakdolgozat

Balogh Árpád

Debrecen

2008

Debreceni Egyetem

Informatika Kar

Adatbázis-kezelés Delphiben

Középiskolai nyilvántartás

Témavezető:

Dr. Bajalinov Erik

Tudományos főmunkatárs

Készítette:

Balogh Árpád

Informatika tanár szak

Debrecen

2008

Tartalom

Bevezetés

Témaválasztás

Egy közelmúltban indult szakközépiskolában dolgozom. Felmerült az igény az adtok elektronikus tárolására, ennek több oka van. Az időszakos adatszolgáltatások, statisztikák, jelentős adminisztratív terhet jelentenek. A Közoktatási Információs Iroda illetve a társadalombiztosítás rendszerébe be kell jelenteni a diákokat, így ha rendelkezünk egy minden adatot tartalmazó adatbázissal, akkor elkerülhető a többszörös adatrögzítés, csökken a hiba lehetőség. Ezen kívül a diákok, szülők informálása is egyszerűbbé válik a hiányzásokról, tanulmányi eredményről. Az elkészült program oktatási célokra is alkalmas a programozás, adatbázis-kezelés és a szervezés tantárgyak esetében. A negyedik fejezet a tanári szakdolgozat, amiben a programozás és adatbázis kezelés oktatásának és motiválásának problémájáról szól. Ennek segítségével egy olyan rendszert tanulmányozhatnak, amely valós feladatokat lát el. A rendszer tesztelése, nagy mennyiségű tényleges adatok segítségével történik, a hiányosságok hamar kiderülnek.

Eszközválasztás

A cél megvalósításához számos eszköz áll rendelkezésre. Az adatbázis kezelők terén a választásom a Microsoft SQL Server 2005 verziójára esett. A kiszolgáló Standard változata ingyenesen használható a közoktatási feladatokat ellátó iskolák számára a tiszta szoftver program keretében, de ha ez a lehetőség később nem állna fenn akkor a program Express verziója továbbra is rendelkezésre áll térítésmentesen. Az ingyenes SQL szerverek terén a legtöbb szolgáltatást nyújtja, folyamatosan fejlesztik, a fejlesztők támogatás megoldott. Kellő mennyiségű és minőségű szakirodalom áll rendelkezésre. Egyszerűen kezelhető az SQL Server Management Studio segítségével, számos funkciót grafikusan meg lehet oldani. Az iskolában informatikai szakok oktatása is folyik ahol az adatbázis-kezelés tantárgy gyakorlati oktatása során ezt a szoftvert is használjuk. A hallgatók, szülők tájékoztatását Web-es felületen is meglehet oldani a szerver ezt is támogatja.

A fejlesztőeszközök terén a Borland Delphi 2005 változatát választottam. Ennek indokai hasonlóak, mint az adatbázis-kezelő választásánál. A Turbo Delphi for Win32 rendszerrel teljesen kompatibilis, amely ugyancsak ingyenesen használható, ezért nem jelent

költséget az intézmény számára, ennek ellenére egy olyan eszköz segítségével oktathatjuk a programozási ismereteket, amely használható tudást biztosít a munkaerőpiacon végzett diákjaink számára. Az elkészült rendszer így teljes mértékben felhasználható oktatási célokra is a már említett gyakorlati felhasználás mellett. Ugyan csak a Delphi mellett szólt az is, hogy ez a rendszer már ismert volt számomra ezért a program fejlesztés során nagyobb figyelmet szentelhettem a részletek kidolgozására. Nem szükséges más fejlesztő eszköz a vékony kliens elkészítéséhez sem, ez az eszköz ennek megvalósítását is lehetővé teszi. A Delphi egyik legnagyobb erőssége a különböző típusú adatbázisokhoz való hozzáférési lehetőségek széleskörű támogatása. Ha szükséges az áttérés egy másik szerverre az a kód jelentősrészének megváltoztatása nélkül is megvalósítható. Ennek a szoftvernek a hibáit is folyamatosan korrigálják. A szakirodalom bőséges, magyar nyelven is sok szakkönyv elérhető. Külön kiadványok jelentek meg a Delphi és az adatbázis-kezelés témakörében.

Fejlesztő eszközök

A Microsoft SQL2005 Server

Az SQL Server átfogó, integrált, teljes körű adatkezelési megoldás, amely biztonságosabb, megbízhatóbb, produktívabb platformot biztosít az intézményi adat- és intelligencia-alkalmazások számára, kiterjesztve ezzel a szervezet minden felhasználójának a lehetőségeit. Az SQL Server 2005 kiváló és ismerős eszközöket nyújt mind az informatikusoknak, mind az adatokkal munkát végzők számára. Leegyszerűsíti a intézményi adatkezelő és elemző alkalmazások elkészítését, felügyeletét és használatát minden platformon, a mobileszközöktől kezdve egészen a nagyvállalati adatkezelő rendszerekig. Az SQL Server 2005 szolgáltatásai mindenre kiterjednek, képes együttműködni a meglévő rendszerekkel, és automatizálja a rutinfeladatokat, így mérettől függetlenül minden intézmény számára teljes körű megoldást biztosít.

Az 1. ábrán az SQL Server 2005 adatplatform szerkezete látható.

1. ábra

Az SQL Server adatplatform a következő eszközöket foglalja magában:

Relációs adatbázis-kezelő. Biztonságosabb, megbízható, skálázható, nagy rendelkezésre állású adatbázismotor, amelynek elődjéhez képest jelentősen javult a teljesítménye, és egyaránt támogatja a strukturált és a strukturálatlan (XML) adatokat.

Replikációs szolgáltatások. Adatreplikáció elosztott és mobil adatfeldolgozó alkalmazások számára, nagy rendelkezésre állás, az egyidejű funkciók skálázhatók a nagyvállalati adatkészítő megoldásokhoz használható másodlagos adattárakkal, integráció a heterogén rendszerekkel, köztük akár a meglévő Oracle adatbázisokkal.

Értesítési szolgáltatások (Notification Services). A fejlett értesítési szolgáltatások segítségével olyan skálázható alkalmazások készíthetők és működtethetők, amelyekkel személyre szabott, időben kézbesített tájékoztatást lehet eljuttatni a különféle hálózati és mobileszközökre.

Integrált szolgáltatások (Integration Services). Adatkinyerési, átalakítási és betöltési (ETL) funkciók adattárházakhoz és az egész intézményot átfogó adatintegráció céljára.

Elemzési szolgáltatások. Online analitikus feldolgozó (OLAP) funkciók, amelyek révén többdimenziós tárolással, gyorsan és összetett szempontok szerint elemezhetők a nagy és bonyolult adathalmazok.

Jelentéskészítő szolgáltatások. Teljes körű megoldás a hagyományos, papírra nyomtatható és az interaktív, webes jelentések elkészítésére, kezelésére és küldésére.

Felügyeleti eszközök. Az SQL Server integrált felügyeleti eszközöket tartalmaz, amelyek sokoldalú adatbázis-felügyeletre és hangolásra adnak módot, illetve szorosan integrálhatók más eszközökkel, például a Microsoft Operations Managerrel (MOM) és a Microsoft Systems Management Serverrel (SMS). A szabványos adat-hozzáférési protokollok jelentősen csökkentik az SQL Serverben tárolt adatok és a meglévő rendszerek integrációjához szükséges időt. Ezenkívül az SQL Server a webszolgáltatások beépített támogatását is tartalmazza, ami garantálja a más alkalmazásokkal és platformokkal való együttműködést.

Fejlesztőeszközök. Az SQL Server olyan fejlesztőeszközöket tartalmaz az adatbázismotorhoz, az adatkinyerési, -átalakítási és betöltési szolgáltatáshoz, az adatbányászathoz, az OLAP-hoz és a jelentéskészítéshez, amelyek szorosan integrálva vannak a Microsoft Visual Studióval, így teljes körű alkalmazásfejlesztési szolgáltatásokat nyújtanak. Az SQL Server minden fontosabb alrendszeréhez mellékelve van saját objektummodellje, illetve

alkalmazásprogramozási interfészei (API), amelyekkel az adott intézmény bármely sajátos szükségletének megfelelő irányban ki lehet bővíteni az adatbázis-kezelő rendszert.

Az SQL Server 2005 adatplatform mérettől függetlenül minden szervezetnek a következő előnyöket nyújtja:

Az adatvagyon kihasználása. Az SQL Server 2005 nemcsak megbízható és biztonságos adatbázis-kezelési funkciókat kínál az üzleti és az elemző alkalmazások számára, hanem beépített funkciói (jelentéskészítés, elemzés, adatbányászat) révén további érték megszerzésére is lehetőséget nyújt felhasználóinak, így a szervezet minden zugába eljuttathatók az adatok ezzel a kivételesen jól teljesítő és rugalmas platformmal.

A termelékenység növelése. Az SQL Server 2005 teljes körű üzleti intelligencia funkcióinak és a Microsoft Office System-hez hasonló ismerős eszközökkel való integrációjának köszönhetően a szervezet minden infomunkása idejében hozzájuthat az elengedhetetlen üzleti információkhoz, a sajátos igényeinek megfelelő tartalommal és formában. Az a cél, hogy az üzleti intelligencia a szervezet minden felhasználójához eljusson, és végső soron a szervezet minden szintjén jobb üzleti döntéseket lehessen hozni a vagyon legfontosabb elemei, az adatok alapján.

Az informatikai rendszer leegyszerűsítése. Az SQL Server 2005 segítségével könnyebbé válik az üzleti és az elemző alkalmazások fejlesztése, bevezetése és felügyelete, mivel a fejlesztőket rugalmas fejlesztőkörnyezettel, az adatbázis-rendszergazdákat pedig integrált, automatikus felügyeleti eszközökkel segíti.

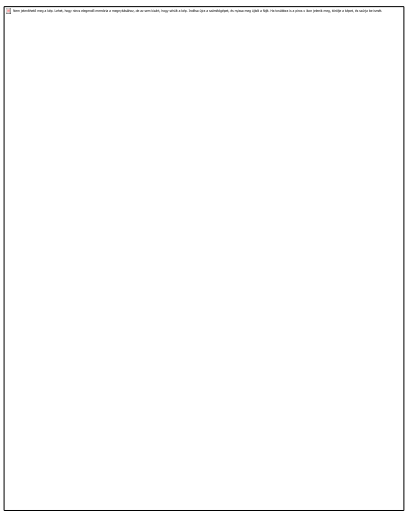
Kisebb birtoklási összköltség (TCO). Az SQL Server 2005 integrált szemléletmódja, a könnyű használat és bevezetés kiemelt szerepe azt eredményezi, hogy ez a legolcsóbban bevezethető és karbantartható adatbáziskezelő-beruházás. Ez különösen igaz azoknál az intézményeknél, amelyek jogosultjai a Tiszta szoftver licensz-szerződésnek.

Az SQL Server 2005 nyújtotta technológiára és szolgáltatásokra minden szervezet mindig számíthat. Az SQL Server 2005-re történő áttérés számottevő előnyökkel jár, mert jelentős előrelépést hoz az adatkezelés, a fejlesztői hatékonyság és az üzleti intelligencia kiemelt területén.

Delphi 2005 For Win32

A Delphi rövid története

2005.februájában ünnepelte a Borland egyik legnagyobb sikerterméke, a Delphi fejleszőrendszer és programozási nyelv 10. születésnapját. Az anno forradalmi újdonságokat bemutató programozási nyelv 1.0-s változatát 1995. február 14-én mutatta be a cég kaliforniai fejlesztői konferenciáján



Delphi 1.0

A Delphi legnagyobb előnye a villámgyors fordítóval párosult teljesen vizuális fejlesztőkörnyezet volt, amely pillanatok alatt tette lehetővé az olyan különböző beviteli ablakok és képernyők megtervezését, valamint bizonyos nem-vizuális feladatok megvalósítását is, amelyek leprogramozása korábban napokat vagy akár heteket raboltak el a fejlesztőktől.

Szintén a gyors és tiszta fejlesztést segítette elő az Object Pascal-ban gyökerező, de jelentősen fejlesztett objektum-orientált nyelv, valamint a Pascal-fordítóktól megszokottakhoz képest hatalmas méretű standard rutin- és osztály-könyvtár is, amelyek drasztikus mértékben csökkentették a leggyakoribb feladatok leprogramozását az új rendszerben.

Bár a Windows 3.1 alá készült Delphi 1 még csak 16-bites futtatható állományok készítésére volt képes, egy évvel később megjelent utódja, a Delphi 2 számára már a 32-bites programok készítése sem okozott gondot. Sajnos a 16-ról 32-bitesre ugrás túl nagy falatnak bizonyult, amelynek köszönhetően a rendszer igen csak megbízhatatlanul működött. Erre az orvosságot az 1997 közepén bemutatott, teljesen újraírt kódgenerátorral rendelkező Delphi 3 jelentette.

A sort 1998-ban a Delphi 4, majd később az 5, 6 és 7 követte, amelyek mindegyike egyre fejlettebb nyelvet, és egyre több komponenst kínált a fejlesztők számára. A Delphi 8-cal a Borland 2003 végén bemutatta első .NET programkód készítésére képes fordítóját, amelynek sikere azonban jóval elmaradt az előző változatokétól. A csorbát a Borland a jelenlegi

legújabb, Delphi 2005 változat piacra dobásával orvosolta, amely immár ismét magában foglalta a natív Windows-alkalmazások készítésének képességét is.

A program írásához használt verzió

A Borland Software (NASDAQ: NM: BORL) a 2004 októberében jelentette be korábban "Diamondback" kódnéven ismert Borland® Delphi™ 2005, valamint a Borland Windows® és .NET gyors alkalmazásfejlesztői (Rapid Application Development, RAD) környezetének megjelenését. A Delphi 2005 a Win32, .NET, Delphi és C# támogatást egyetlen környezetben kombinálja, jelentősen megnöveli a fejlesztők és a csapat hatékonyságát, és integráltan működik együtt a Borland vezető alkalmazás-életciklus menedzsment (Application Lifecycle Management, ALM) megoldásaival. “A Delphi 2005 a Delphi évek óta kiadott legjelentősebb változata, és a piacon ma elérhető egyik legteljesebb Windows IDE és ALM fejlesztőrendszer" - mondta George Paolini, a Borland alelnöke és fejlesztői eszközökért felelős igazgatója. “Ugyanakkor tökéletesen illeszkedik a Borland Szoftverszállítási optimalizálásra vonatkozó jövőképéhez, segít a fejlesztőcsapatoknak, hogy megnöveljék a szoftverprojektek előreláthatóságát és sikerét, hogy azok szállítása időben és költségvetésen belül történhessen, az üzleti érték maximalizálásával."

A Delphi 2005 több nyelv, valamint mind a Win32, mind pedig a .NET SDK-k támogatásán felül az innovatív fejlesztői és csapatszintű hatékonyságnövelő funkciók tárházát kínálja, mint például a kód újrafaktorizálás, egységtesztelés és az új ECO II (Enterprise Core Objects, vállalati alapobjektumok) modell által megvalósított .NET vállalati üzleti alkalmazás-keretrendszer. Ezzel egy időben lehetővé teszi a fejlesztőcsapatok számára, hogy a meglévő Windows alkalmazásokat fenntartsák és továbbfejlesszék, miközben kihasználják az új technológiákat és lehetőségeket.

“A Borland megérti, hogy a mai Windows fejlesztőcsapatokra nagy nyomás nehezedik, mivel úgy kell támogatniuk a meglévő alkalmazásokat, hogy közben új technológiákkal törnek előre, és mindezt a rövidülő leszállítási ciklusok és gyakran csökkenő erőforrások környezetében kell véghezvinniük" - mondta Michael Swindell, a Borland fejlesztői eszközök termékmenedzsmentjének igazgatója. “A Delphi 2005 olyan funkciókat kínál, amelyekre a fejlesztőknek szüksége van ahhoz, hogy mind a meglévő, mind pedig a következő generációs Windows alkalmazások fejlesztési és karbantartási folyamatát lerövidítsék."

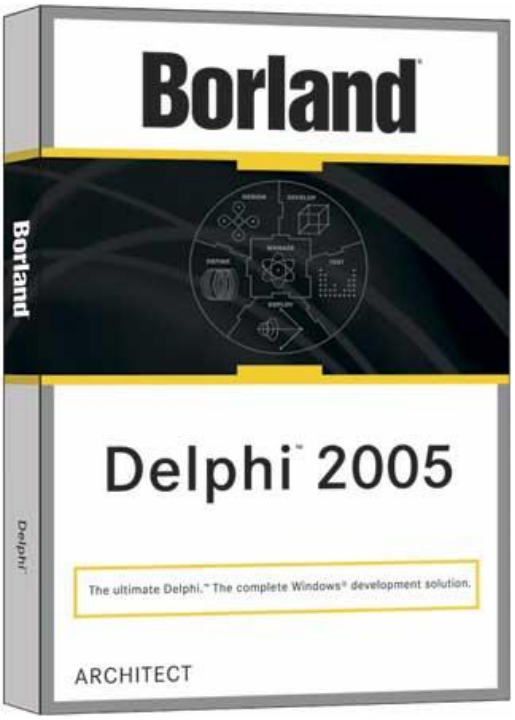
A Delphi 2005 biztosítja azokat a Windows nyelveket, Win32 és .NET SDK eszközöket, ALM integrációt és fejlesztői hatékonyságnövelő funkciókat, amelyekre a modern Windows fejlesztőknek manapság szüksége van. A Delphi 2005 segítségével a fejlesztők mind a Win32 alkalmazások továbbvitelére, mind pedig a .NET és ASP.NET továbbfejlesztésére képesek, anélkül, hogy a meglévő befektetéseket maguk mögött hagynák.

Az Delphi 2005 néhány legfontosabb továbbfejlesztése:

- Több nyelv és Windows SDK támogatása - A Delphi 2005 biztosítja a modern Windows fejlesztéshez szükséges nyelveket és SDK támogatást. Mivel mind a Delphi, mind pedig a C# fejlesztést támogatja, ez az egyetlen igazi olyan Windows termék, amely ugyanazon eszközből és ugyanazon nyelv alapján (Delphi) támogatja a natív

Win32 és .NET fejlesztést. Ugyanakkor az ASP.NET, ADO.NET, VCL.NET és VCL for Win32 megoldásokat is támogatja.

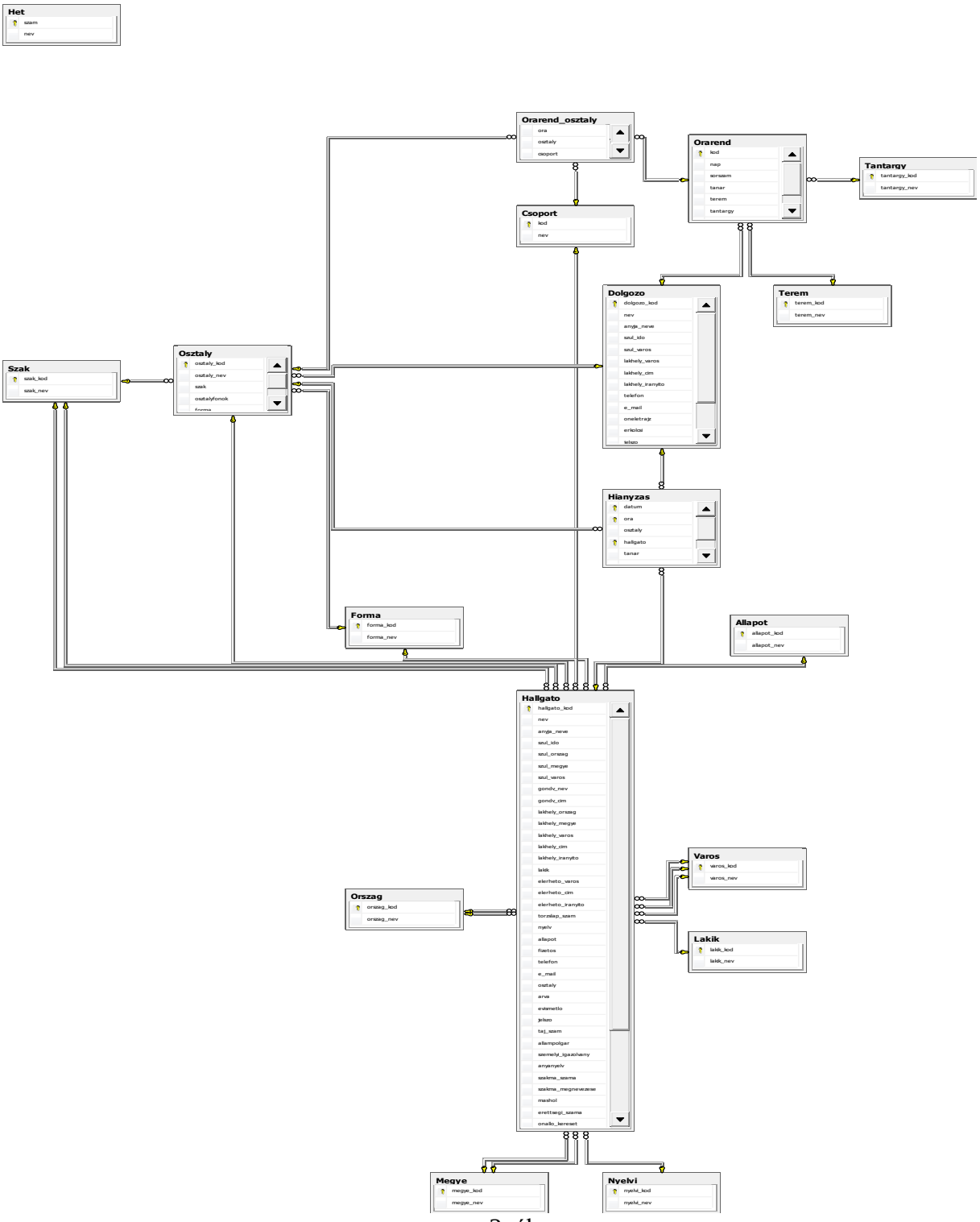
- ALM megoldások integrációja - A Delphi 2005 célja, hogy a fejlesztőknek a StarTeam® és az Optimizeit™ integrációjával rálátást biztosítson az alkalmazás-életciklus különböző fázisaira. A StarTeam integráció célja, hogy leegyszerűsítse a forráskód erőforrások menedzsmentjét és növelje a csapat kommunikációját, míg a mellékelt Optimizeit Profiler for .NET segít az egységtesztek automatizálásában, valamint az alkalmazás minőségének és teljesítményének általános továbbfejlesztésében.
- Gyors vállalati MDA fejlesztést tesz lehetővé - A Delphi 2005 ECO II megoldása vállalati szintű gyors modell alapú architektúra (Model Driven Architecture, MDA) megoldást biztosít a .NET-hez, amely lerövidíti a bonyolult alkalmazások fejlesztését, javítja minőségüket, és megnöveli karbantarthatóságukat. Az ECO II az objektumok önműködő diagrammszerű ábrázolásának, valamint létrehozatalának teljes megoldása, amely rugalmasan méretezhető, fejlett vállalati objektum funkciókkal (pl. visszavonás/ismétlés, verziókezelés és tranzakciók) ellátott .NET objektum gyorsítótárat kínál.
- Leegyszerűsíti és lerövidíti a Windows fejlesztést - A Delphi 2005 számos innovatív IDE funkciót kínál, amely hozzájárul a napi fejlesztői munka megkönnyítéséhez, megnöveli a hatékonyságot, és leegyszerűsíti a kód karbantartását. Olyan funkciókat kínál, mint a fejlett kód refaktorizálás, Help Insights és Error Insights (súgó és hiba vizsgálat), SyncEdit (szinkronizált szerkesztés), History Management (régi elemek menedzsmentje), és a Delphi nyelv új továbbfejlesztései. A Delphi Advantage for ADO.NET célja, hogy az adatbázisokhoz kapcsolódó .NET alkalmazások fejlesztését minden szempontból lerövidítse és leegyszerűsítse mind Delphi, mid pedig C# alatt.



2.ábra

A program informatikai leírása

Az adatbázis



3. ábra

A fenti diagramot az Sql Management Studio New Database Diagram funkciójával készítettem. Az ábrán jól látható a táblák kapcsolat rendszere. Az adatbázis negyedik normálformában van, így megfelel a relációs adatmodell követelményeinek. A táblák szerkezetét, a mezők típusát a generáló szkript segítségével mutatom be:

```
USE [master]
GO
CREATE DATABASE [Debrecen] ON  PRIMARY
( NAME = N'Debrecen', FILENAME = N'D:\Fejlesztés\Iskola\Data\Debrecen.mdf' , SIZE =
3072KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'Debrecen_log', FILENAME = N'D:\Fejlesztés\Iskola\Data\Debrecen.ldf' , SIZE =
1024KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)
 COLLATE SQL_Hungarian_CP1250_CS_AS
GO
EXEC dbo.sp_dbcmptlevel @dbname=N'Debrecen', @new_cmptlevel=90
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Debrecen].[dbo].[sp_fulltext_database] @action = 'disable'
end
GO
ALTER DATABASE [Debrecen] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [Debrecen] SET ANSI_NULLS OFF
GO
ALTER DATABASE [Debrecen] SET ANSI_PADDING OFF
GO
ALTER DATABASE [Debrecen] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [Debrecen] SET ARITHABORT OFF
GO
ALTER DATABASE [Debrecen] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [Debrecen] SET AUTO_CREATE_STATISTICS ON
GO
ALTER DATABASE [Debrecen] SET AUTO_SHRINK OFF
```

```
GO
ALTER DATABASE [Debrecen] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [Debrecen] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [Debrecen] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [Debrecen] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [Debrecen] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [Debrecen] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [Debrecen] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [Debrecen] SET  DISABLE_BROKER
GO
ALTER DATABASE [Debrecen] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [Debrecen] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [Debrecen] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [Debrecen] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [Debrecen] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [Debrecen] SET  READ_WRITE
GO
ALTER DATABASE [Debrecen] SET RECOVERY FULL
GO
ALTER DATABASE [Debrecen] SET  MULTI_USER
GO
ALTER DATABASE [Debrecen] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [Debrecen] SET DB_CHAINING OFF
```

```
USE [Debrecen]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Allapot](
	[allapot_kod] [int] IDENTITY(1,1) NOT NULL,
	[allapot_nev] [nchar](15) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
 CONSTRAINT [PK_Allapot] PRIMARY KEY CLUSTERED
(
	[allapot_kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [IX_Allapot] UNIQUE NONCLUSTERED
(
	[allapot_nev] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
CREATE TABLE [dbo].[Csoport](
	[kod] [int] NOT NULL,
	[nev] [nchar](15) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
 CONSTRAINT [PK_Csoport] PRIMARY KEY CLUSTERED
(
	[kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
CREATE TABLE [dbo].[Dolgozo](
	[dolgozo_kod] [int] IDENTITY(1,1) NOT NULL,
	[nev] [nchar](30) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
	[anyja_neve] [nvarchar](30) COLLATE SQL_Hungarian_CP1250_CS_AS NULL,
	[szul_ido] [datetime] NULL,
	[szul_varos] [int] NULL,
	[lakhely_varos] [int] NULL,
```

```
	[lakhely_cim] [nvarchar](30) COLLATE SQL_Hungarian_CP1250_CS_AS NULL,
	[lakhely_iranyito] [nchar](4) COLLATE SQL_Hungarian_CP1250_CS_AS NULL,
	[telefon] [nvarchar](40) COLLATE SQL_Hungarian_CP1250_CS_AS NULL,
	[e_mail] [nvarchar](30) COLLATE SQL_Hungarian_CP1250_CS_AS NULL,
	[oneletrajz] [bit] NULL,
	[erkolcsi] [bit] NULL,
	[jelszo] [nchar](10) COLLATE SQL_Hungarian_CP1250_CS_AS NULL,
	[elmeleti] [int] NULL CONSTRAINT [DF_Dolgozo_elmeleti]  DEFAULT ((0)),
	[gyakorlati] [int] NULL CONSTRAINT [DF_Dolgozo_gyakorlati]  DEFAULT ((0)),
 CONSTRAINT [PK_Dolgozo] PRIMARY KEY CLUSTERED
(
	[dolgozo_kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
CREATE TABLE [dbo].[Forma](
	[forma_kod] [int] IDENTITY(1,1) NOT NULL,
	[forma_nev] [nchar](20) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
 CONSTRAINT [PK_Forma] PRIMARY KEY CLUSTERED
(
	[forma_kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [IX_Forma] UNIQUE NONCLUSTERED
(
	[forma_nev] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
CREATE TABLE [dbo].[Het](
	[szam] [int] NOT NULL,
	[nev] [nchar](10) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
 CONSTRAINT [PK_Het] PRIMARY KEY CLUSTERED
(
	[szam] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
```

```
) ON [PRIMARY]

GO
CREATE TABLE [dbo].[Lakik](
	[lakik_kod] [int] IDENTITY(1,1) NOT NULL,
	[lakik_nev] [nchar](15) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
 CONSTRAINT [PK_lakik] PRIMARY KEY CLUSTERED
(
	[lakik_kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [IX_Lakik] UNIQUE NONCLUSTERED
(
	[lakik_nev] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
CREATE TABLE [dbo].[Megye](
	[megye_kod] [int] IDENTITY(1,1) NOT NULL,
	[megye_nev] [nchar](25) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
 CONSTRAINT [PK_Megye] PRIMARY KEY CLUSTERED
(
	[megye_kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [IX_Megye] UNIQUE NONCLUSTERED
(
	[megye_nev] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Nyelvi](
	[nyelvi_kod] [int] IDENTITY(1,1) NOT NULL,
	[nyelvi_nev] [nchar](10) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
 CONSTRAINT [PK_Nyelvi] PRIMARY KEY CLUSTERED
```

```
(
	[nyelvi_kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [IX_Nyelvi] UNIQUE NONCLUSTERED
(
	[nyelvi_nev] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
CREATE TABLE [dbo].[Orszag](
	[orszag_kod] [int] IDENTITY(1,1) NOT NULL,
	[orszag_nev] [nchar](15) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
 CONSTRAINT [PK_Orszag] PRIMARY KEY CLUSTERED
(
	[orszag_kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [IX_Orszag] UNIQUE NONCLUSTERED
(
	[orszag_nev] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
CREATE TABLE [dbo].[Szak](
	[szak_kod] [int] IDENTITY(1,1) NOT NULL,
	[szak_nev] [nchar](30) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
 CONSTRAINT [PK_Szak_1] PRIMARY KEY CLUSTERED
(
	[szak_kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [IX_Szak] UNIQUE NONCLUSTERED
(
	[szak_nev] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

```
GO
CREATE TABLE [dbo].[Tantargy](
	[tantargy_kod] [int] IDENTITY(1,1) NOT NULL,
	[tantargy_nev] [nchar](25) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
 CONSTRAINT [PK_Tantargy] PRIMARY KEY CLUSTERED
(
	[tantargy_kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [IX_Tantargy] UNIQUE NONCLUSTERED
(
	[tantargy_nev] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
CREATE TABLE [dbo].[Terem](
	[terem_kod] [int] IDENTITY(1,1) NOT NULL,
	[terem_nev] [nchar](20) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
 CONSTRAINT [PK_Terem] PRIMARY KEY CLUSTERED
(
	[terem_kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
CREATE TABLE [dbo].[Varos](
	[varos_kod] [int] IDENTITY(1,1) NOT NULL,
	[varos_nev] [nchar](15) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
 CONSTRAINT [PK_Varos] PRIMARY KEY CLUSTERED
(
	[varos_kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [IX_Varos] UNIQUE NONCLUSTERED
(
	[varos_nev] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

```
GO
CREATE TABLE [dbo].[Orarend](
	[kod] [int] IDENTITY(1,1) NOT NULL,
	[nap] [nchar](10) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
	[sorszam] [int] NOT NULL,
	[tanar] [int] NULL,
	[terem] [int] NULL,
	[tantargy] [int] NULL,
	[tipus] [nchar](10) COLLATE SQL_Hungarian_CP1250_CS_AS NULL,
 CONSTRAINT [PK_Orarend] PRIMARY KEY CLUSTERED
(
	[kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Orarend]  WITH CHECK ADD  CONSTRAINT [FK_Orarend_Dolgozo]
FOREIGN KEY([tanar])
REFERENCES [dbo].[Dolgozo] ([dolgozo_kod])
GO
ALTER TABLE [dbo].[Orarend]  WITH NOCHECK ADD  CONSTRAINT [FK_Orarend_Tantargy]
FOREIGN KEY([tantargy])
REFERENCES [dbo].[Tantargy] ([tantargy_kod])
GO
ALTER TABLE [dbo].[Orarend] CHECK CONSTRAINT [FK_Orarend_Tantargy]
GO
ALTER TABLE [dbo].[Orarend]  WITH CHECK ADD  CONSTRAINT [FK_Orarend_Terem]
FOREIGN KEY([terem])
REFERENCES [dbo].[Terem] ([terem_kod])

GO
CREATE TABLE [dbo].[Osztaly](
	[osztaly_kod] [int] IDENTITY(1,1) NOT NULL,
	[osztaly_nev] [nchar](10) COLLATE SQL_Hungarian_CP1250_CS_AS NOT NULL,
	[szak] [int] NOT NULL,
	[osztalyfonok] [int] NULL,
```

```
	[forma] [int] NOT NULL,
 CONSTRAINT [PK_Osztaly] PRIMARY KEY CLUSTERED
(
	[osztaly_kod] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [IX_Osztaly] UNIQUE NONCLUSTERED
(
	[osztaly_nev] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Osztaly]  WITH NOCHECK ADD  CONSTRAINT [FK_Osztaly_Dolgozo]
FOREIGN KEY([osztalyfonok])
REFERENCES [dbo].[Dolgozo] ([dolgozo_kod])
GO
ALTER TABLE [dbo].[Osztaly] CHECK CONSTRAINT [FK_Osztaly_Dolgozo]
GO
ALTER TABLE [dbo].[Osztaly]  WITH CHECK ADD  CONSTRAINT [FK_Osztaly_Forma]
FOREIGN KEY([forma])
REFERENCES [dbo].[Forma] ([forma_kod])
GO
ALTER TABLE [dbo].[Osztaly]  WITH NOCHECK ADD  CONSTRAINT [FK_Osztaly_Szak]
FOREIGN KEY([szak])
REFERENCES [dbo].[Szak] ([szak_kod])
GO
ALTER TABLE [dbo].[Osztaly] CHECK CONSTRAINT [FK_Osztaly_Szak]

GO
CREATE TABLE [dbo].[Hallgato](
	[hallgato_kod] [int] IDENTITY(1,1) NOT NULL,
	[nev] [nchar](30) NOT NULL,
	[anyja_neve] [nvarchar](30) NULL,
	[szul_ido] [datetime] NULL,
	[szul_orszag] [int] NULL,
	[szul_megye] [int] NULL,
	[szul_varos] [int] NULL,
```

[gondv_nev] [nvarchar](30) NULL,
[gondv_cim] [nchar](40) NULL,
[lakhely_orszag] [int] NULL,
[lakhely_megye] [int] NULL,
[lakhely_varos] [int] NULL,
[lakhely_cim] [nvarchar](30) NULL,
[lakhely_iranyito] [nchar](4) NULL,
[lakik] [int] NULL,
[elerheto_varos] [int] NULL,
[elerheto_cim] [nvarchar](30) NULL,
[elerheto_iranyito] [nchar](4) NULL,
[torzslap_szam] [nchar](10) NULL,
[nyelv] [int] NULL,
[allapot] [int] NULL,
[fizetos] [bit] NULL CONSTRAINT [DF_Hallgato_fizetos] DEFAULT ((0)),
[telefon] [nvarchar](40) NULL,
[e_mail] [nvarchar](30) NULL,
[osztaly] [int] NULL,
[arva] [bit] NULL CONSTRAINT [DF_Hallgato_arva] DEFAULT ((0)),
[evismetlo] [bit] NULL CONSTRAINT [DF_Hallgato_evismetlo] DEFAULT ((0)),
[jelszo] [nchar](10) NULL,
[taj_szam] [nvarchar](20) NULL,
[allampolgar] [nvarchar](20) NULL,
[szemelyi_igazolvany] [nvarchar](20) NULL,
[anyanyelv] [nvarchar](20) NULL,
[szakma_szama] [smallint] NULL,
[szakma_megnevezese] [nvarchar](50) NULL,
[mashol] [bit] NULL,
[erettsegi_szama] [nchar](15) NULL,
[onallo_kereset] [bit] NULL,
[szulovel] [bit] NULL,
[nem] [bit] NOT NULL CONSTRAINT [DF_Hallgato_nem] DEFAULT ((0)),
[diak_ig_szam] [nchar](15) NULL,
[csoport] [int] NULL,
[szak1] [int] NULL,
[szak2] [int] NULL,
[forma] [int] NULL,

```
	[kir] [nchar](15) NULL,
 CONSTRAINT [PK_Hallgato] PRIMARY KEY CLUSTERED
(
	[hallgato_kod] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY],
 CONSTRAINT [IX_Hallgato] UNIQUE NONCLUSTERED
(
	[nev] ASC,
	[osztaly] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Hallgato]  WITH NOCHECK ADD  CONSTRAINT [FK_Hallgato_Allapot]
FOREIGN KEY([allapot])
REFERENCES [dbo].[Allapot] ([allapot_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Allapot]
GO
ALTER TABLE [dbo].[Hallgato]  WITH CHECK ADD  CONSTRAINT [FK_Hallgato_Csoport]
FOREIGN KEY([csoport])
REFERENCES [dbo].[Csoport] ([kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Csoport]
GO
ALTER TABLE [dbo].[Hallgato]  WITH CHECK ADD  CONSTRAINT [FK_Hallgato_Forma]
FOREIGN KEY([forma])
REFERENCES [dbo].[Forma] ([forma_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Forma]
GO
ALTER TABLE [dbo].[Hallgato]  WITH NOCHECK ADD  CONSTRAINT [FK_Hallgato_Lakik]
FOREIGN KEY([lakik])
REFERENCES [dbo].[Lakik] ([lakik_kod])
GO
```

```
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Lakik]
GO
ALTER TABLE [dbo].[Hallgato]  WITH NOCHECK ADD  CONSTRAINT [FK_Hallgato_Megye]
FOREIGN KEY([lakhely_megye])
REFERENCES [dbo].[Megye] ([megye_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Megye]
GO
ALTER TABLE [dbo].[Hallgato]  WITH NOCHECK ADD  CONSTRAINT [FK_Hallgato_Megye1]
FOREIGN KEY([szul_megye])
REFERENCES [dbo].[Megye] ([megye_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Megye1]
GO
ALTER TABLE [dbo].[Hallgato]  WITH NOCHECK ADD  CONSTRAINT [FK_Hallgato_Nyelvi]
FOREIGN KEY([nyelv])
REFERENCES [dbo].[Nyelvi] ([nyelvi_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Nyelvi]
GO
ALTER TABLE [dbo].[Hallgato]  WITH NOCHECK ADD  CONSTRAINT [FK_Hallgato_Orszag]
FOREIGN KEY([lakhely_orszag])
REFERENCES [dbo].[Orszag] ([orszag_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Orszag]
GO
ALTER TABLE [dbo].[Hallgato]  WITH NOCHECK ADD  CONSTRAINT [FK_Hallgato_Orszag1]
FOREIGN KEY([szul_orszag])
REFERENCES [dbo].[Orszag] ([orszag_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Orszag1]
GO
ALTER TABLE [dbo].[Hallgato]  WITH NOCHECK ADD  CONSTRAINT [FK_Hallgato_Osztaly]
FOREIGN KEY([osztaly])
REFERENCES [dbo].[Osztaly] ([osztaly_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Osztaly]
```

```
GO
ALTER TABLE [dbo].[Hallgato]  WITH CHECK ADD  CONSTRAINT [FK_Hallgato_Szak]
FOREIGN KEY([szak1])
REFERENCES [dbo].[Szak] ([szak_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Szak]
GO
ALTER TABLE [dbo].[Hallgato]  WITH CHECK ADD  CONSTRAINT [FK_Hallgato_Szak1]
FOREIGN KEY([szak2])
REFERENCES [dbo].[Szak] ([szak_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Szak1]
GO
ALTER TABLE [dbo].[Hallgato]  WITH NOCHECK ADD  CONSTRAINT [FK_Hallgato_Varos]
FOREIGN KEY([lakhely_varos])
REFERENCES [dbo].[Varos] ([varos_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Varos]
GO
ALTER TABLE [dbo].[Hallgato]  WITH NOCHECK ADD  CONSTRAINT [FK_Hallgato_Varos1]
FOREIGN KEY([szul_varos])
REFERENCES [dbo].[Varos] ([varos_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Varos1]
GO
ALTER TABLE [dbo].[Hallgato]  WITH NOCHECK ADD  CONSTRAINT [FK_Hallgato_Varos2]
FOREIGN KEY([elerheto_varos])
REFERENCES [dbo].[Varos] ([varos_kod])
GO
ALTER TABLE [dbo].[Hallgato] CHECK CONSTRAINT [FK_Hallgato_Varos2]

GO
CREATE TABLE [dbo].[Orarend_osztaly](
        [ora] [int] NOT NULL,
        [osztaly] [int] NOT NULL,
        [csoport] [int] NULL
) ON [PRIMARY]
```

```
GO
ALTER TABLE [dbo].[Orarend_osztaly]  WITH CHECK ADD  CONSTRAINT
[FK_Orarend_osztaly_Csoport] FOREIGN KEY([csoport])
REFERENCES [dbo].[Csoport] ([kod])
GO
ALTER TABLE [dbo].[Orarend_osztaly]  WITH CHECK ADD  CONSTRAINT
[FK_Orarend_osztaly_Orarend] FOREIGN KEY([ora])
REFERENCES [dbo].[Orarend] ([kod])
GO
ALTER TABLE [dbo].[Orarend_osztaly]  WITH CHECK ADD  CONSTRAINT
[FK_Orarend_osztaly_Osztaly] FOREIGN KEY([osztaly])
REFERENCES [dbo].[Osztaly] ([osztaly_kod])

GO
CREATE TABLE [dbo].[Hianyzas](
        [datum] [datetime] NOT NULL,
        [ora] [int] NOT NULL,
        [osztaly] [int] NOT NULL,
        [hallgato] [int] NOT NULL,
        [tanar] [int] NULL,
        [igazolt] [bit] NULL,
 CONSTRAINT [PK_Hianyzas] PRIMARY KEY CLUSTERED
(
        [datum] ASC,
        [ora] ASC,
        [hallgato] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[Hianyzas]  WITH CHECK ADD  CONSTRAINT [FK_Hianyzas_Dolgozo]
FOREIGN KEY([tanar])
REFERENCES [dbo].[Dolgozo] ([dolgozo_kod])
GO
ALTER TABLE [dbo].[Hianyzas]  WITH CHECK ADD  CONSTRAINT [FK_Hianyzas_Hallgato]
FOREIGN KEY([hallgato])
```

```
REFERENCES [dbo].[Hallgato] ([hallgato_kod])
GO
ALTER TABLE [dbo].[Hianyzas]  WITH CHECK ADD  CONSTRAINT [FK_Hianyzas_Osztaly]
FOREIGN KEY([osztaly])
REFERENCES [dbo].[Osztaly] ([osztaly_kod])
```

Az SQL nyelv ismertetése nem témája a dolgozatnak ezért csak azokat az utasításokat emelem ki amelyek nem részei a szabvány SQL-nek de szükségesek a Transact-SQL-szkript írásához.

1. USE[adatbázis név] beállítja az aktuális adatbázist.
2. EXEC dinamikus kódgenerálást lehet végrehajtani. Futatási időben adhatjuk meg a kódot.
3. GO nem T-SQL utasítás de a legtöbb SQL Servert támogató eszköz felismeri, a köteg végét jelzi. Az addigi kódot egyetlen egységként értelmezi és elküldi a kiszolgálónak. A táblák létrehozásánál azért van erre szükség, mert a hivatkozások (külső kulcsok) csak így adhatók meg. Egy 1-N kapcsolatban csak akkor hivatkozhatok a master táblára ha az már létezik, például a Hallgato táblában a hallgato_varos mező mint idegen kulcs akkor állítható be ha már létezik a Varos tábla.

Az adatbázis megadásánál az adatokra vonatkozó megszorítások is szerepelnek, ezek ellenőrzése automatikusan megtörténik. Az adatok rendezése a magyar ékezeteknek megfelelően történik.

A program

A feladat

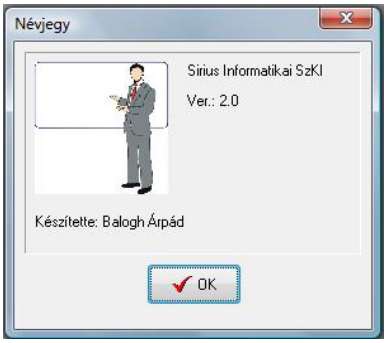
A program feladata az oktatási tevékenységhez kapcsolódó adminisztráció támogatása. A jelenlegi törvényi szabályozás megköveteli a papír alapú dokumentumok vezetését, azonban ezen a területen várhatók módosítások. Ha ez megtörténik és lehetővé teszik, hogy az adatokat digitális formában tárolhassák az iskolák azzal jelentős munkaerő megtakarításra nyílik lehetőség és a nyilvántartások is pontosabbak, naprakészebbek lehetnek. Amíg ez bekövetkezik addig is jelentős segítséget jelent egy megfelelően kialakított adatbázis használata. A program ezen verziója a képzési alap adatok (tanárok, tanulók, osztályok) mellett az órarendet is kezeli. A tanuló kapcsán nagyon sok információra van szükség. A szokásos információk mellett a Közoktatási Információs Rendszer és az Országos

Egészségbiztosítási Pénztár által megkövetelt adatokat is nyilván kell tárolni. A program segítségen nyújt az említett rendszerekkel történő kapcsolattartásra. A hiányzások kezelése egy bonyolult feladat, amihez jelentős támogatást biztosít az alkalmazás.

A felépítés

A program felépítése a Multi Document Interface modellt követi, ami azt jelenti, hogy a főablak területén jelennek meg a gyermek ablakok. Az adatbázis kezelésben a kétrétegű kliens-szerver architektúrát valósítja meg, ami jelen esetben a kliens oldalt megvalósítását jelenti. A szerver oldalon egy MS SQL Server 2005 alkalmazás működik. A részletes bemutatást az ablakok szerint ismertetem.

Splash képernyő

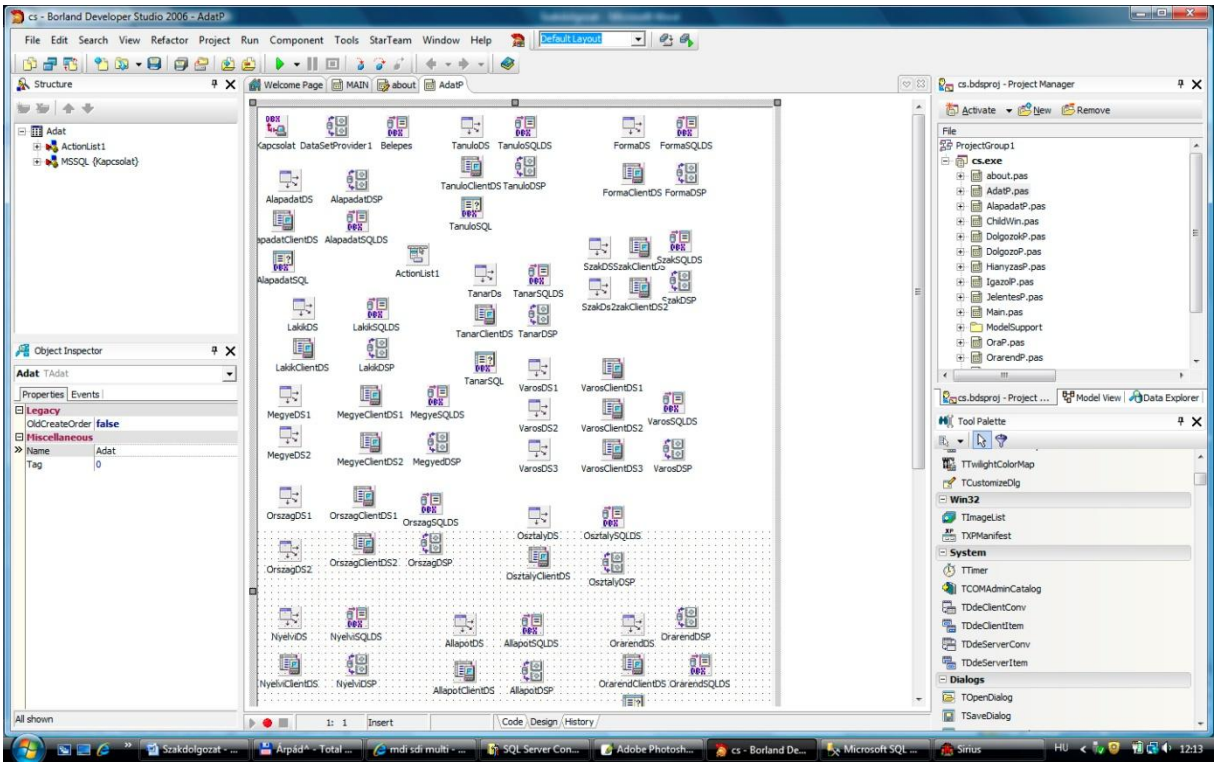


Ez az első képernyő amivel a felhasználó találkozik. Kettős szerepet tölt be. Egyrészt a programba való belépést teszi lehetővé. ComboBox-ok segítségével lehet kiválasztani a SQL szervert és a adatbázis nevét. zzel a megoldással egy programban több intézmény, tagintézmény adatait tudjuk kezelni egymás mellett. Az azonosításhoz felhasználói névre ami a Dolgozo táblában a Nev mezőben található a jelszó pedig a Jelszo-ban. A gombok TBitBtn típusú vezérlőkkel vannak megoldva. A „Mégsem” gomb egyszerűen kilép a programból, az „OK” pedig elvégzi csatlakozást a szerverhez, ellenőrzi a felhasználó adatait és ha minden rendben megjeleníti a főablakot. A csatlakozás elvégzését egy try-except blokkban végzi el, így ha hiba lép fel, akkor ezt kezeli. Másrészt a „Segítség” menü „Névjegy…” pontjában megjelenő ablak is ez. A funkciótól függően az ablakot átméretezem és csak a megfelelő vezérlők látható. A bejelentkezés alatt a keret sem jelenik meg.

Adatmodul

Az adatbázissal kapcsolatos komponenseket az alkalmazás legtöbb ablakából el kell érni. Ez nem jelent nehézséget, hiszen a Delphiben minden form egy unit-ban kerül

megvalósításra ezért a uses utasítás segítségével bármelyik másik unit-ban felhasználhatjuk azt. Célszerű az összes ilyen komponenst egy form-ra összegyűjteni így könnyebben átláthatóbb lesz a szerkezet. A Delphiben van egy speciális típusú form ennek a megvalósításra. Ennek az az oka, hogy ezek nem látható a komponensek tehát nincs szükség olyan erőforrásokra, amelyek a megjelenítéshez szükségesek. Ez a TDataModule, ami csak tervező nézetben látható.



Az alkalmazott komponensek:

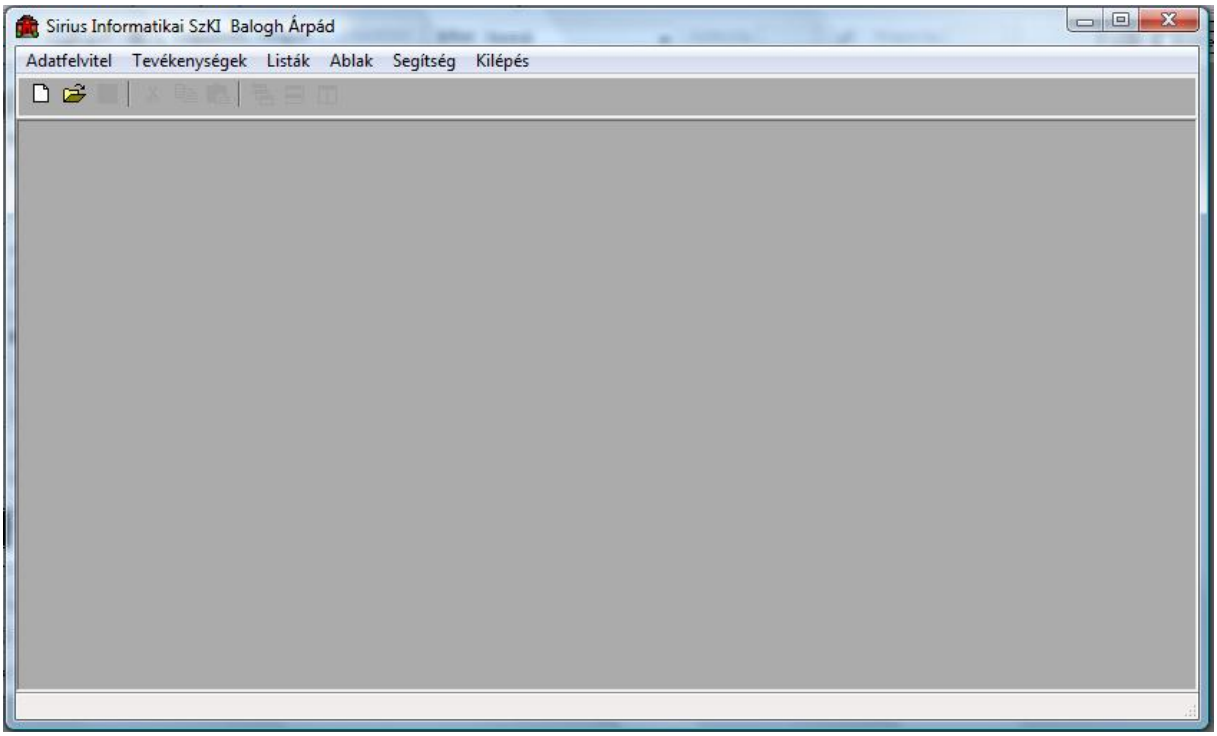
-  TSQLConnection az adatbázis szerverrel történő kapcsolattartás eszköze. Ebben kell megadnia használt drivert, a szervert futtató hoszt nevét, az adatbázis nevét, és a login információkat.
-  TDataSetProvider adatokat szolgáltat egy adatkészletből, illetve elvégzi a módosítások frissítését is az adatkészletben.
-  TSQLDataSet egyirányú adathalmaz a szervertől származó információk tárolására.

-  TClentDataSet ez olyan adathalmaz, amelyben tetszés szerint mozoghatunk ezzel lehetővé téve például a tartalom adatrácsban történő megjelenítését.
-  TDataSource segítségével az adataink forrás függetlenül használhatók fel adatmegjelenítési komponensekkel.
-  TSQLQuery az adatbázis irányába történő adatmozgatás eszköze.

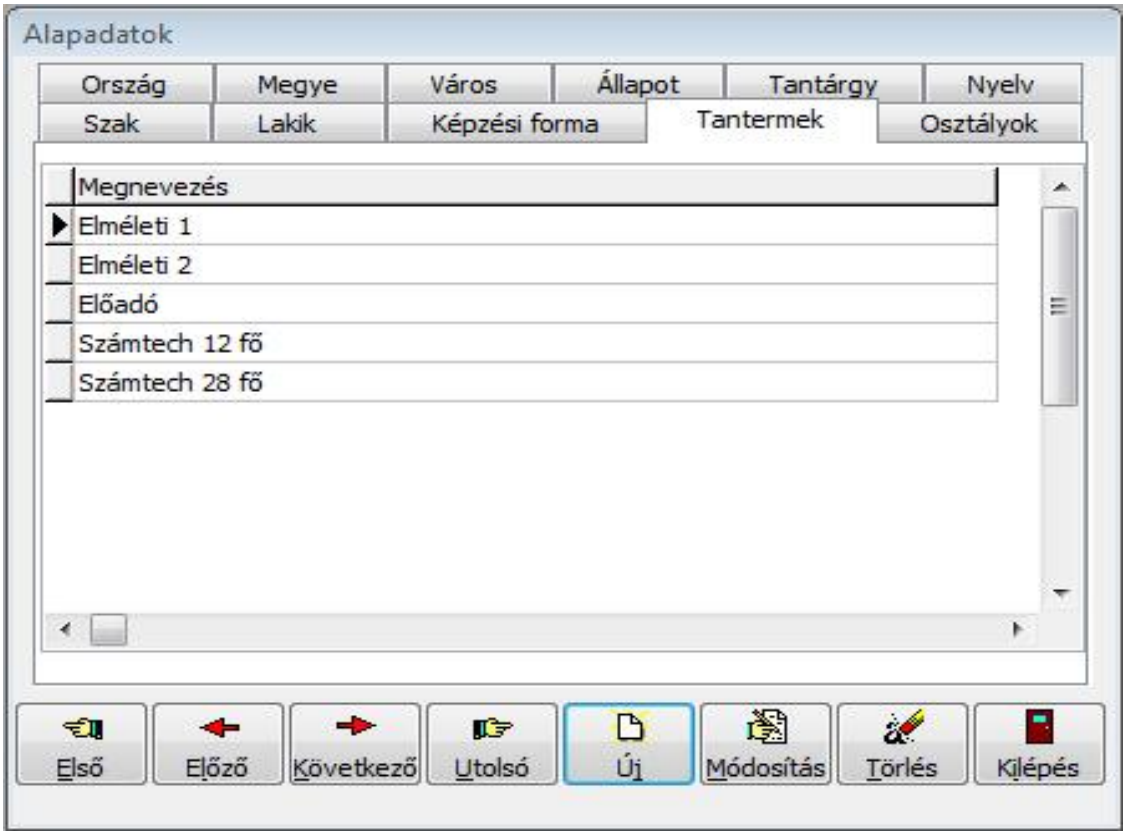
Ezeket az alkotó elemeket használom adat forrásonként alkalmazva. A felhasználók a formokon kiadott utasítási ezeken keresztül lesznek megvalósítva. Található még egy ActionList elem is az adatmodulon, amely egy Action-t tartalmaz, amely az adatok frissítését oldja meg. Erre minden módosítást követően szükség van.

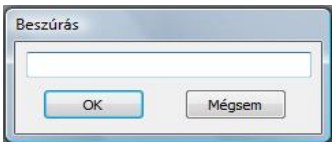
Főablak

Az alkalmazás főablaka a funkciók indítására szolgál. Ehhez egy TMainMenu típusú menüt és a gyakori feladato elvégzéséhez egy eszköz palettát tartalmaz. A form típusa ftMDIForm. A Microsoft Windows XP és Vista operációs rendszerekben lehetőség van az asztalon ún. témák beállítására. Ez egy sajátos megjelenési formát eredményez. Ez alkalmazható a saját programjainkban is ha a program főablakán elhelyezünk egy TXPManifest nevű komponenst.

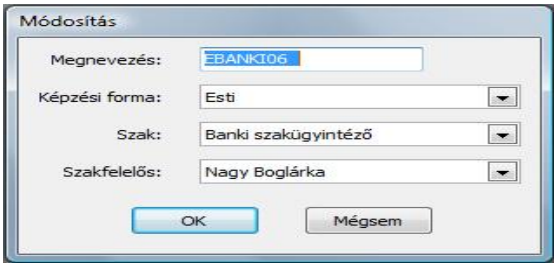


Alapadatok

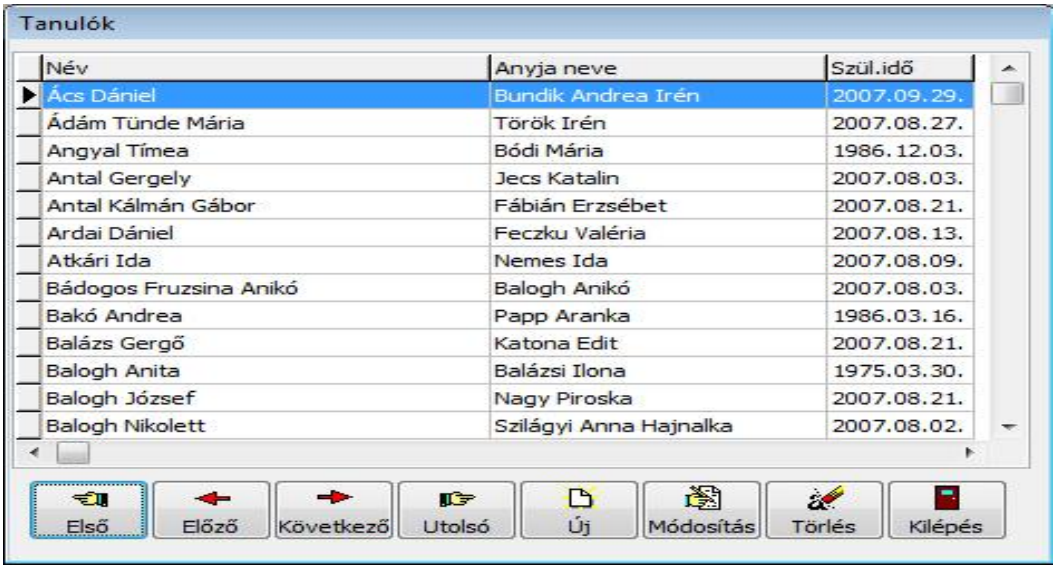


Tizenegy különböző tábla adataik kell megjeleníteni és a karbantartó műveleteket elindítani (új elem felvitel, módisítás, törlés…). Hogy ne kelljen ennyi ablakot elhelyezni a menüben ezért egy ablakon egy TTabControl-t használok. Az adatrácsban pedig dinamikusan változtatom a megjelenített tartalmat a kiválasztott fültől függően. A műveletek indítását TBitBtn gombokkal oldottam meg. Az adatok felvitelét, módosítását az osztályoktól eltekintve egy egyszerű dialógusablak valósítja meg. 

Az osztályok esetében több adatra van szükség. Ezért külön ablak kell hozzá. Ahol a megnevezés DBEdit a többi adat pedig TDBLookUpComboBox-ban tölthető ki. A listás vezérlők esetében az adatokat az adatbázis tábláiból olvassa fel a program.



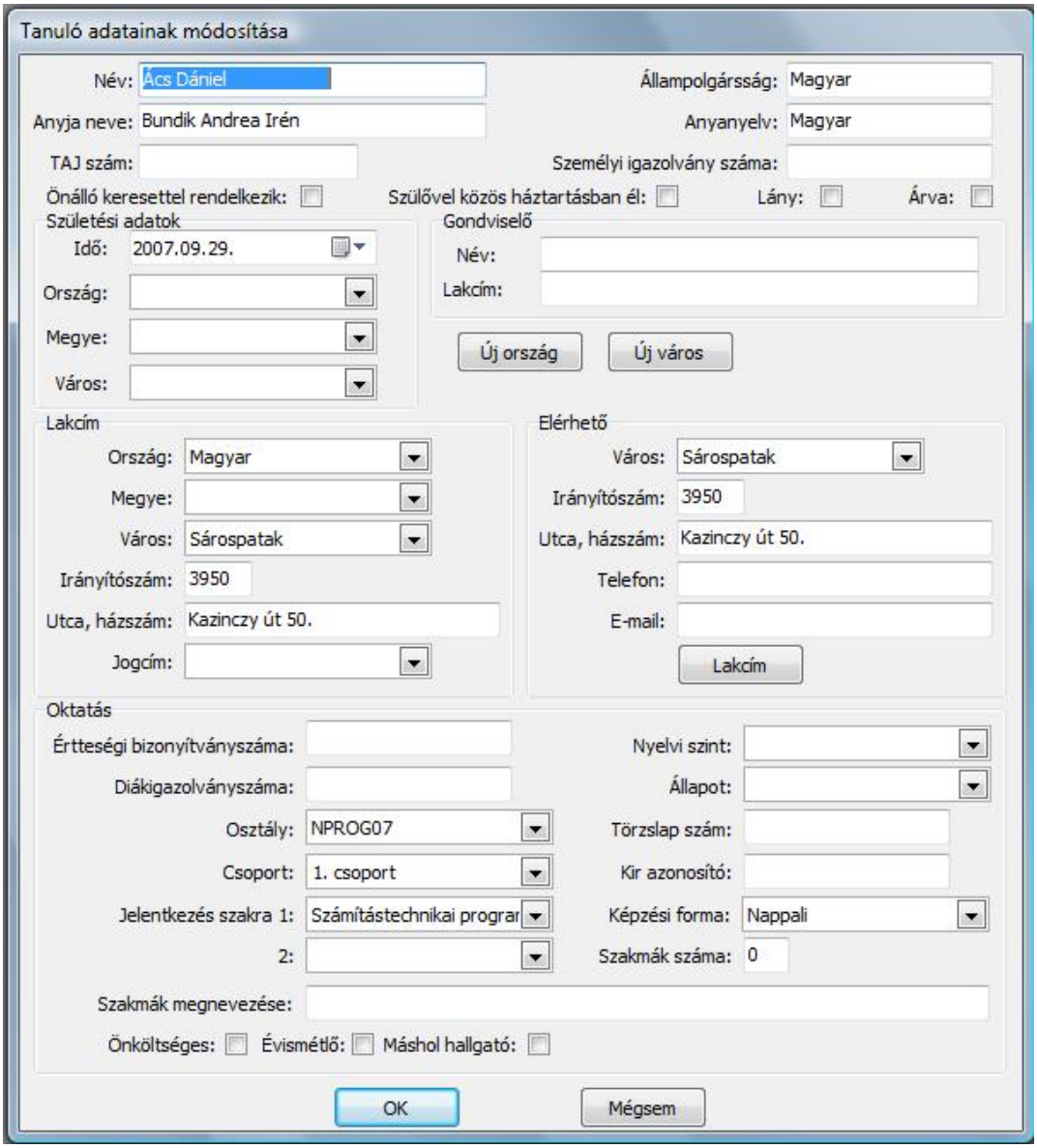
Tanulók



Amint látszik a feladat ugyan az azonban több oszlopra van szükség hiszen a név nem azonosít egyértelműen egy tanulót, lehetnek azonos nevűek is. Itt az anyja neve és a születési idő is szerepel. A gombsor ugyan az mint az alapadatok esetében.

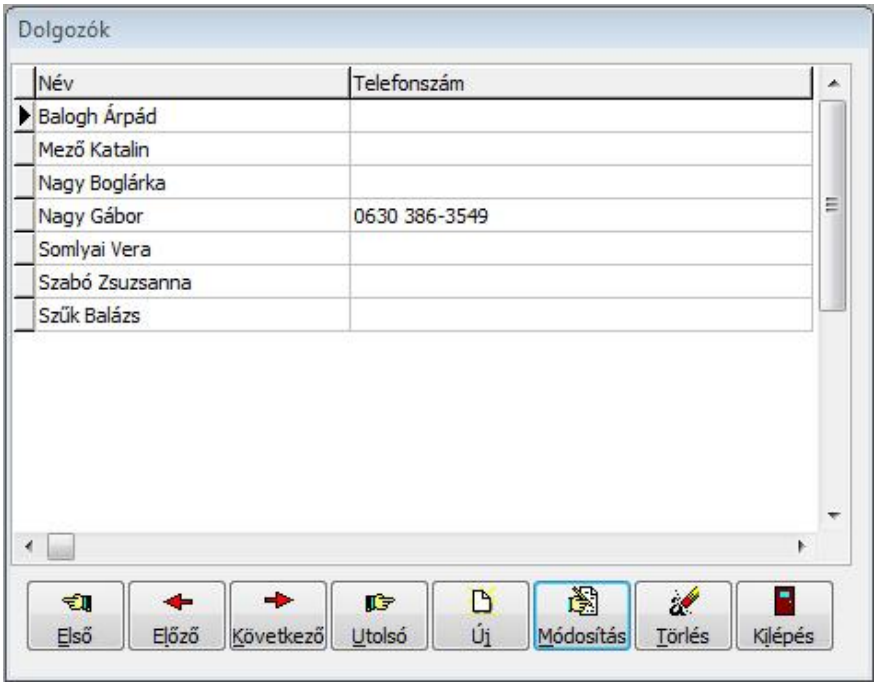
Tanuló adatai

Ez a legnagyobb méretű és legtöbb komponenst tartalmazó ablak. A dátumok kezelése a TDateTimePicker a legalkalmasabb, így nem kell külön ellenőríni, hogy valóban valós dátumot gépeltekbe és a felhasználó is megmenekül a gépeléstől. A logikai értékek TCheckBox-okkal egyszerűen kezelhető. Azok az adatok amelyek már rendelkezésre állnak TDBLookUpCombox-okkal kiválaszthatók, így nem csak gépelni nem kell a felhasználónak de csak a felkínált lehetőségek közül választhat, nem okozhat integritási problémát. Ha olyan tanuló adatait kell rögzíteni aki olyan helyen lakik, amely még nincs rögzítve akkor a lakhely közvetlenül felvihető nem kell kilépni. Az elérhetőség az esetek többségében egybeesik a lakcímmel. Az adatok másolását egy nyomógombbal meg lehet oldani.

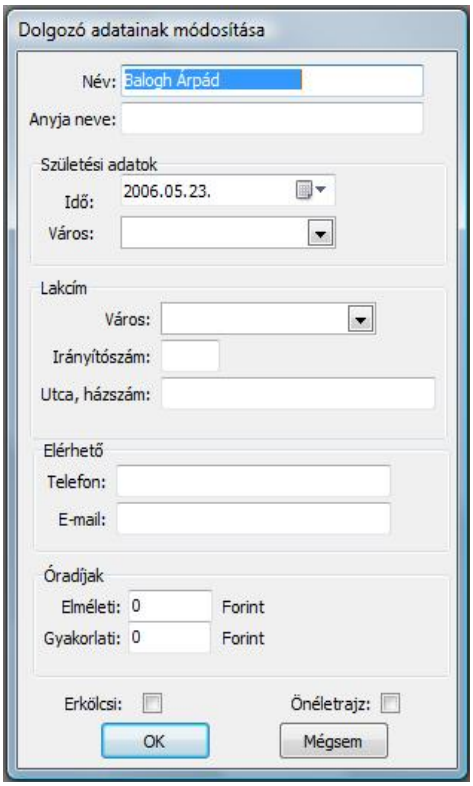

Tanuló adatainak módosítása
Név: Ács Dániel
Állampolgárság: Magyar
Anyja neve: Bundik Andrea Irén
Anyanyelv: Magyar
TAJ szám:
Személyi igazolvány száma:
Önálló keresettel rendelkezik:
Szülővel közös háztartásban él:
Lány:
Árva:
Születési adatok
Idő: 2007.09.29.
Ország:
Megye:
Város:
Gondviselő
Név:
Lakcím:
Új ország
Új város
Lakcím
Ország: Magyar
Megye:
Város: Sárospatak
Irányítószám: 3950
Utca, házszám: Kazinczy út 50.
Jogcím:
Elérhető
Város: Sárospatak
Irányítószám: 3950
Utca, házszám: Kazinczy út 50.
Telefon:
E-mail:
Lakcím
Oktatás
Érettségi bizonyítványszáma:
Diákigazolványszáma:
Osztály: NPROG07
Csoport: 1. csoport
Jelentkezés szakra 1: Számítástechnikai prograr
2:
Nyelvi szint:
Állapot:
Törzslap szám:
Kir azonosító:
Képzési forma: Nappali
Szakmák száma: 0
Szakmák megnevezése:
Önköltséges:
Évismétlő:
Máshol hallgató:
OK
Mégsem

Dolgozók

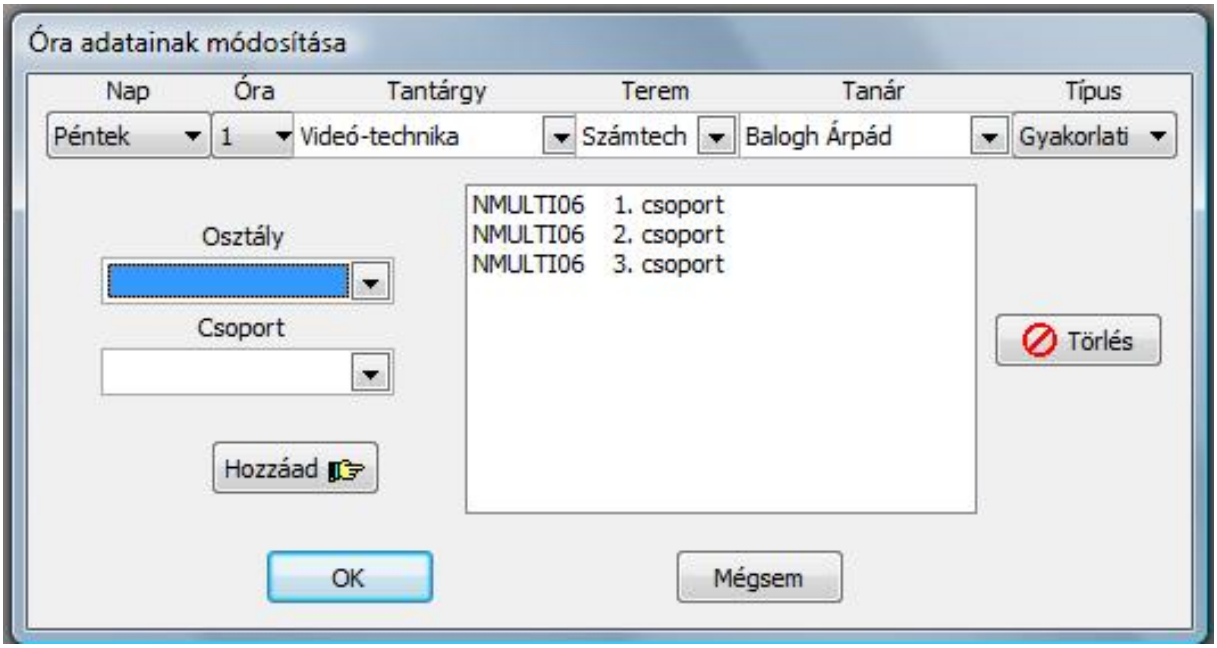
Az ablak hasonló mint a tanulók ablak de itt a nevek mellett a telefonszámok vannak feltüntetve, ez az az információ amely leggyakrabban szükséges. A gombsor úa. mit a többi hasonló ablaknál.



Az egyes dolgozók adatainál az elméleti és gyakorlati óradíjak is szerepelnek amelyek egy később megvalósítandó modulban kerülnek felhasználásra modulban



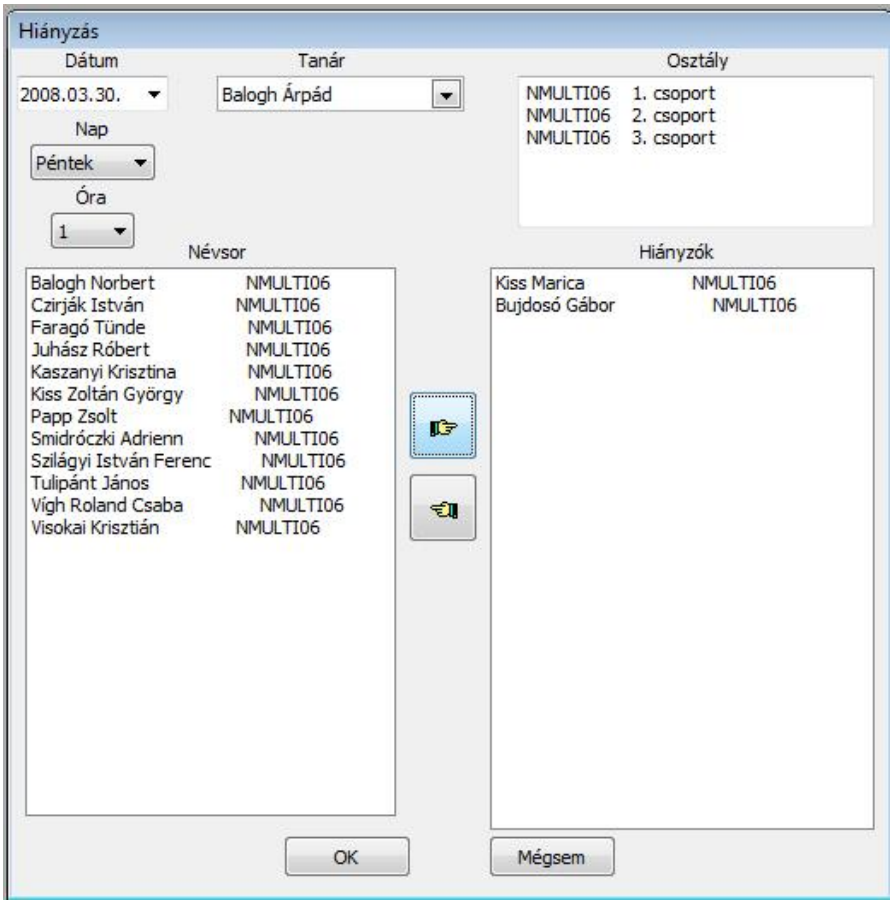
Órarend



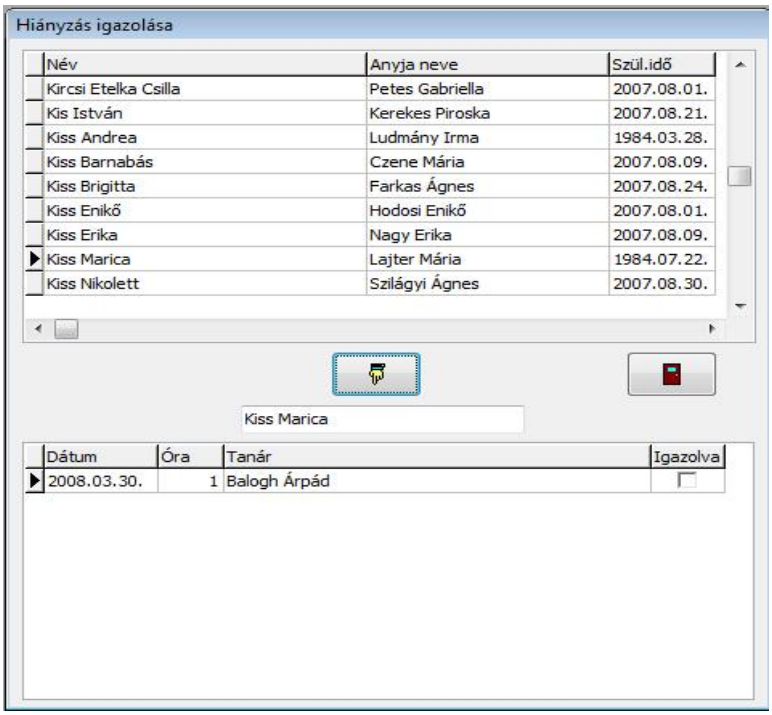
Az órarend megadásakor nem az adatok mennyisége, hanem az összetettség jelenti a nehézséget. A napok nevei és a óra sorszáma egy ComboBox-ból választható ki. A tantárgy, tanterem, tanár, osztály és csoport adatok pedig az alapadatok és a dolgozók menüpontban már nyilvántartásba vett adatok közül kerülhet ki. Ezeket az információkat TDBLookUpComboBox-ok szolgáltatják. Az órarend megadásakor figyelembe kell venni, hogy bizonyos tantárgyak csoportbontásban, míg mások összevonva kerülnek megtartásra. Ha egy óra fölöslegesen vagy tévesen lett bejegyezve a „Törlés” gombbal távolítható el.

Hiányzások rögzítése

A szükséges adatok: a napi dátum, az óra sorszáma és a tanár neve. Ezen információk alapján a program feltölt egy ListBox-ot azon tanulók nevével, akiknek jelen kell lenniük az órán a csoportbontásokat és összevonásokat is figyelembe véve. Lehetőség van a nap megváltoztatására, erre munkanapok áthelyezésekor lehet szükség. A hiányzókat a gombok vagy drag and dropp technikával mozgathatjuk a ListBoxok között. Lehetőség van egy vagy akár egyszerre több sor mozgatására is. A megvalósítás két dinamikus tömbbel történik. A hiányzások feltöltésére az órarendek megadása után nyílik lehetőség. A számítógéppel felszerelt termekben azonnal elvégezhető a feladat.



A hiányzások igazolására is lehetőség van. Csak a tanuló nevét kell kiválasztani és a rendszer automatikusan megjeleníti az összes hiányzását. A dátum az óra sorszáma és a tanár mellett egy CheckBox jelenik meg. Ha pipa szerepel benne igazolt a hiányzás.



Listák

A jelentések Rave Designer-rel készültek. Ez a program plug-in ként érhető el a Delphi-ben. Ez a program hasonlóan működik, mint a hasonló alkalmazások mint például a Cristal Reports vagy akár az Access jelentés tervezője. Előnye, hogy közvetlenül képes a lekérdezések eredményeinek átvételére a Delphiből. A jelentések használatához szükséges a Rave által generált .rav kiterjesztésű fájlokra futási időben is. Ezek nem túl nagyméretű állományok jellemzően 3-4 kilobájtosak, ezért akár az exe fájlba is beépíthetők. Ekkor azonban a jelentésekben történt változtatások az alkalmazás módosítását vonják maguk után. A Rave ismertetése nem tárgya a szakdolgozatnak. A programban jelenleg három lista érhető el:

- Összesítés a normatív támogatásra jogosult tanulókról.

Normatív támogatás

Képzésiforma	Szak	Létszám
Esti	Banki szakügyintéző	60
Esti	Gazdasági informatikus I.	9
Esti	Intézményi kommunikátor	21
Esti	Médiatechnológus asszisztens	6
Esti	Multimédia-fejlesztő	24
Esti	Számítástechnikai programozó	21
		141
Nappali	Banki szakügyintéző	86
Nappali	Gazdasági informatikus I.	13
Nappali	Intézményi kommunikátor	31
Nappali	Médiatechnológus asszisztens	20
Nappali	Multimédia-fejlesztő	37
Nappali	Reklámszervező szakmenedzse	26
Nappali	Számítástechnikai programozó	41
		254
		395

- Összesítés a költségtérítést fizető tanulókról, amely lista szerkezete megegyezik az előbbivel.
- Boríték címzés a diákoknak

Sirius Informatikai SzKI
Debrecen
Salétrom u. 1
4025

Ács Dániel
Sárospatak
Kazinczy út 50.
3950

- A listák menü utolsó pontja a tanulók adataiból egy EXCEL táblázatot hoz létre melynek szerkezetét a KIR rendszerben adták meg. Ezt az .xls kiterjesztésű állományt használva automatikusan regisztrálhatók az új tanulók a közoktatási rendszerben.

Tesztelés

A program tesztelésének környezete: A kiszolgáló MS SQL 2005 Server Standard Edition. A szervergép 2.8 GH Intel CPU, 1 Giga bájt ram, az operációs rendszer MS Windows 2003 Server. A kliens program mind Windows XP mind Vista operációs rendszereken működik. A tesztelés több változatos kiépítésű hardveren is megtörtént. Különösebb konfigurációs igény nincs, az operációs rendszer követelményei elegendőek. Az adatok az iskola tanulóinak adatai.

Programozás és az adatbázis kezelés oktatása középiskolában számítástechnikai programozó szakon

A programozás tanítása

A szakdolgozathoz kapcsolódó ismeretek oktatása a 14. évfolyamon történik. Azonban a tantárgy már része a 13. évfolyam anyagának is. Azok a diákok, akik jelentkeznek, a szakképzésre már rendelkeznek számítástechnikai előképzettséggel. A NAT által előírt tematika mind az általános mind a középiskolában meghatározza a tanítandó tananyagot ennek része a programozási ismeret is (algoritmusok). Az öt éves tapasztalat, amit ebben a képzésben szereztem azonban azt mutatja, hogy ezt a terület elhanyagolják. Nincsenek meg a legalapvetőbb tudásuk sem, ezért erre nem lehet építeni. Természetesen vannak olyan tanulók, akik rendelkeznek akár a követelményi szintet meghaladó tudással is. A jelentkezés feltétele

csak az érettségi bizonyítvány megléte, nem előfeltétel a számítástechnikából történt vizsga. A jelentkezők között jelentő a tudás különbség, ez különösen az első időszakban jelent problémát. A szintet azokhoz kell igazítani akik semmilyen előismerettel nem rendelkeznek ez viszont a többiek számára unalmas lehet. Ezt a szakaszt ezért célszerű a lehető leggyorsabban átvenni. A túlzott sietség viszont ahhoz vezethet, hogy a kezdők számára nem lesznek meg a szükséges alapok és később emiatt lemorzsolódnak.

A legnehezebb feladat az algoritmus szemlélet kialakítása. Ennek elsajátításában hasznos segédeszköz a folyamatábra vagy más folyamat leíró rendszer pl. stuktogram. Más ismeret területen megszerzett ismereteket felhasználva, mint például a matematika, el lehet sajátítatni a lépésekre bontás, a lépések megfelelő sorrendbe állításának a képességét. Ilyen feladat lehet például a másodfokú egyenletek megoldása. A legtöbb hiba a szelekció és az iteráció tévesztéséből szokott adódni. Általában azok a diákok akik ezt az anyagrészt teljesen eltudják sajátítani, a későbbiekben is jobb eredményeket érnek el. A lemorzsolódók számára általában ez megoldhatatlan feladatot jelent.

Mire már tisztában vannak az alapvető lépésekkel akkora a számítástechnikai alapismeretek tárgyban már foglalkoztak a számítógép belső ábrázolási módszereivel. Ezt felhasználva könnyebben megismertethetjük a típusok fogalmát. Példákon keresztül kialakítható az adatmodell alkotás képessége. Egy adott feladathoz önállóan megtudják határozni a szükséges változókat, konstansokat és ezek típusait.

Ezen ismeretek birtokában egy konkrét programozási nyelv megismertetése nem is olyan nehéz feladat. Fontos a kezdő nyelv kiválasztása. Mivel a többség még nem találkozott programozással így a Pascal nyelv egy járható út, ráadásul később ez a tudás a Delphi elterjedtsége miatt piacképes tudást is jelent. A C típusú nyelvek megértése komolyabb feladatot jelent, ezért ezt csak később második programnyelvként tanulják. Az első programok az algoritmizálási feladatok megvalósításai. Egyszerű konzol programok segítségével megismerkednek az input-, outpututasításokkal és egyre összetettebb feladatokat oldanak meg, például legkisebb közös többszörös, legnagyobb közös osztó. Ennek a szakasznak a célja, hogy a tanuló egy egyszerű feladatot önállóan meg tudjon valósítani: algoritmizálás, adatok meghatározása, kódolás. Fontos, hogy sikerélményhez jussanak, ne csak unalmas sablon feladatokat kapjanak, az alap faladatok is feldobhatók ötletekkel. Elegendő időt kell biztosítani az alapok elsajátítására. Ha már kellő magabiztossággal oldják meg a feladatokat a következő lépcső az alprogramok és modulok megismertetése. Az absztrakciós szint

növekedésével egyre nagyobb problémák megoldását tűzhetjük ki. Itt már a csoportos munka is megjelenik. Egy komplex feladatot fel lehet bontani olyan diszjunkt részekre, amelyeket a csoportok önállóan megvalósíthatnak és végül összeállítanak egy tejes programot. A team munka megismerése egyébként is fontos feladat, a végzett programozók a legtöbb esetben egy munkacsoport tagjaként kapnak munkát. A nehézséget a figyelem fenntartása jelenti. A gyerekek a mindennapi életükben számos alkalmazással találkoznak, amelyek grafikus felületen futnak és sokkal tetszetősebbek a saját maguk írta programok, ilyenek például s játékprogramok. A szokásos menetrendet célszerű néha megszakítani egy-egy demonstrációval és bemutatni, hogy a tetszetős külső mögött ugyan olyan kód található, mint az ő alkalmazásaikban. A modulok bevezetésével tovább bővítjük a kód újra hasznosításának tárházát. Az itt szerzett a tudást Delphi-ben a formok és komponensek megismertetésekor felhasználhatjuk.

A következő nagyfeladat az objektumorientált programozás tanítása. A nehézséget az okozza, hogy sok összetett fogalommal kell megismertetni a tanulókat. Ekkor az már tisztában vannak az alapvető algoritmizálási, programozási lépésekkel, de az objektum-orientáltság egy magasabb fokú absztrakciót valósít meg amelynek megértése még a jobb képességű tanulók számára is több energiát igényel. Szemléletes példákon keresztül lehet elmagyarázni a modell lényegét. A funkcionális modell és az adatmodell már ismert fogalmak. Ezek összekapcsolását és az ennek folyományaként megjelenő új elemeket kell megismertetni. Az első fogalom az egységbezárás. Ez az alapja az egész rendszernek. Az ismertetésben párhuzamot vonhatunk a rekord típussal, különösen a C++ programozási nyelvvel kapcsolatban ahol is a valóban megvalósítható így is az objektum. Nem szabad azonban elfelejteni, felhívni a különbségekre a figyelmet. Érdemes több időt szánni a magyarázatra és sok gyakorlati példát hozni, majd önálló vagy csoport munkával objektumokat alkotatni a diákokkal. Ha már tisztában vannak az alapokkal bele mehetünk a részletekbe. Mindig az előnyök szemszögéből mutassuk be az aktuális fogalmat, mert az a tapasztalatom, hogy még így is nehéz a sok és komplikált ismeret elfogadtatása, mivel már összetettebb programok írásával is elboldogulnak, nem értik miért hasznos ez számukra. A másik probléma, hogy sokan úgy vélik, hogy az eddigi tudásuk értéktelen, ezért tudatosítani kell, hogy amit eddig tanultak az fontos és erre épül az új tudásuk. A programírási készséget egyébként is karban kell tartani, programozási tételeken alapuló kisebb-nagyobb feladatok megoldásával. A záróvizsgán az írásbeli feladatsorban mindig vannak ilyen feladatok is. Az

adatrejtés megértésével ne szokott gond lenni, néhány feladaton keresztül könnyen bemutathatjuk a különböző láthatósági direktívák hatásait. Ami lényegesen összetettebb probléma az az öröklés. Ennek szemléltetésére is kiválóan alkalmas a Delphi. Egyszerűben valósítja meg a mechanizmust, mint például a C++, mivel nincs többszörös öröklés. Nem szabad túl bonyolult példákat hozni, hiszen az alapgondolat feldolgozása is nehéz feladat. Olyan szemléletes feladatokon keresztül, mint például: jármű, gépjármű, személygépkocsi, tehergépkocsi, kerékpár, stb.. vagy síkidom, sokszög, háromszög, négyzet, ellipszis, kör el lehet érni azt, hogy mindenki számára világos legyen a lényeg. A gyakorlásra itt is nayg szükség van. Addig nem érdemes tovább menni, amíg az ilyen típusú feladatokat nem tuják készség szinten megoldani a tanulók. A metódusok szerepét bemutatni sem jelenthet nehéz feladatot. Itt kamatoztathatók a már meglévő ismeretek. Külön ki kell emelni a konstruktorok és destruktorok szerepét. Ebben a témakörben a legnehezebb az osztály metódusok működésének tisztázása. Nem szokták érteni azt a helyzetet, hogy hogyan működhet valami amihez még nem hoztunk létre példányt. A példányosítás és felszabadítás párhuzamba állítható a dinamikus memória kezelés témakörével, ott is hasonló feladatokat kell végezni. A tulajdonságok megismertetése lesz az alapja a komponens alapú fejlesztésnek. Ezeknek a feladatoknak a bemutatására véleményem szerint a karakteres felület alkalmasabb mint a grafikus. Túl sok információt kellene feldolgozni rövid idő alatt ha még ezen ismeretek átadását is ebben a szakaszban szeretnék megvalósítani. A tananyagban ez a megcélzott szint amit az objektumorientált módszerrel kapcsolatban a tanulóknak ismerniük kell. Gazdagítható az anyag például az interfészekkel, virtuális illetve dinamikus metódusokkal, ha az osztály tudás szintje ezt megengedi.

Az objektumokról szerzett tudást felhasználva a következő témakör a grafikus felület megismertetése. Ezt általában lelkesen fogadják a tanulók. Itt találkoznak először a mindennapi életben, sőt más tantárgyak esetében (Weblap-tervezés, operációs-rendszerek, …) eleve megszokott felülettel. Az eddigi tananyaghoz képest sokkal látványosabbá válik a munka, viszonylagosan kis energia befektetéssel nagyon látványos eredmény érhető el. Természetesen itt is meg vannak az elméleti részek. A legnagyobb különbség az eseményvezérelt programozási módszer. Új eszközöket kell megismerniük a tanulóknak, mint az Object Inspector, Structure ablak, Project Manager. Az ablak tulajdonságainak, eseményeinek megismertetésével lehetőség van a témakör elméletének kifejtésére. Például az ablak létrejöttének, lebontásának folyamata szépen szemléltethető a háttérszín változtatásával.

A grafikus komponensek feldolgozásának sorendjéhez meghatározásához jó támpontot nyújt a ComputerBooks kiadó gondozásában megjelent Programozzunk Delphi 7 rendszerben című könyv. Az adatok megjelenítés és bekérés kapcsán külön foglalkozni kell a típuskonverziós függvényekkel, eljárásokkal, hiszen a Windows csak szöveges a megjelenítést és bekérést támogatja. Az ismert komponensek tárházának bővülésével nő a variációs lehetőségek száma. Könnyű olyan feladatokat adni, amelyek szórakoztatók, szemben a konzol programokkal ahol a típus feladatok sokszor bizony egysíkúak. Olyan egyszerű faladatok segítségével, mint például egy olyan nyomógomb elhelyezése az ablakon, amelyre nem lehet rákattintani, mert elmozog, nagyon sok hasznos tudás átadható és a szívesen is dolgoznak a diákok. Ebben a fázisban sok egyéni és csoportos feladat adható akár házi feladat formában is. Különösen hasznosak a csoport feladatok, ezzel a módszerrel a lemaradók is bevonhatók a munkába és teret enged a fantáziának is. Ennek a veszélye az, hogy sokan túlbecsülik tudásukat és olyan feladatokat szeretnének megvalósítani, amelyekre nem képesek. Nem szabad függőben hagyni a felmerült problémákat. Amennyire lehet, az aktuális tudásszinten megoldást kell találni a problémára illetve a megoldás bemutatásával ösztönözni lehet a mélyebb tudás megszerzésére. Lehetőséget kell biztosítani a gyorsabban haladók számára plusz feladatok adásával, versenyekre történő nevezésekkel. Azonban tekintettel kell lenni azokra is akiknek a törzsanyag feldolgozása is nehéz. Fennáll a veszélye annak, hogy többen elveszítik a fonalat és nem vesznek részt a közös munkában. A feladatokat lehetőség szerint úgy kell kialakítani, hogy több szinten megvalósíthatók legyenek. Ezen a ponton tárgyalható a kapcsolattartás az operációs-rendszerrel, a telepítőkészlet létrehozása és a súgó készítése is.

A szakdolgozatom témájához szorosan kapcsolódó terület az adatbázis kezelés megvalósítása programszinten. A témakör megkezdésének előfeltétele az eddig leírt programozói ismeretek mellett az adatbázis kezelésben megszerzett előismeretek megléte is. Erre a szakaszra a 14. évfolyamban kerül sor. A diákok ekkorra már teljesítettek egy félévet az adatbázis kezelés elméletéből és gyakorlatból is. Ismerik a relációs modellt és az SQL nyelvet is. A két tantárgynak ettől a ponttól szinkronban kell haladniuk, előnyös lehet, ha a két tantárgyat egy személy oktatja, hiszen a két téma gyakran összemosódik. A téma feldolgozását egy kis ismétléssel kell bevezetni. A fájlokkal végezhető műveletek részét képezik az elsajátított programozási anyagnak. Most egy kicsit más szemszögből kell megvizsgálni. Azokra a problémákra kell felhívni a figyelmet, amelyeket a CODASYL DBTG ajánlása is felvet. Előnyös, ha hálózatba kapcsolt gépeket használhatunk ezzel a

konkurens hozzáférést gyakorlatban szemléltethetjük, ha erre nincs lehetőség, akkor az alkalmazás több példányban történő elindítása is megoldás lehet. Egy adott feladatot több csoporttal megoldatva különböző rekordformátumok jönnek létre, hogyan lehet a megvalósítani, hogy ezek együtt tudjanak működni. Hogyan valósítható meg a konzisztencia és az adatok biztonsága? Ilyen és ehhez hasonló kérdésekkel tehetjük szemléletessé az adatbázis kezelés fontosságát. A Delphi számos lehetőséget kínál a megvalósításra. A szakirodalomban általában három témakörben osztják a lehetőségeket. Ezt a felosztást célszerű megtartani.

Első lehetőség a helyi adattárolás. Kisebb feladatok megoldására kiválóan alkalmas megoldás. A Delphi szinte minden ismert formátumot támogat. Érdemes ezeknek egy kis időt eltölteni ezen lehetőségek tanulmányozására. A Delphi telepítésekor a lemezre kerül egy segédprogram, (dbd32.exe) amellyel például Dbase, Paradox formátumú adattárak hozhatók létre. Ismeretek szerezhetők az index fájlokról, lehetőség van mezőszintű integritási feltételek megadására, kapcsolatok definiálására. A fejlesztő eszközben új komponens családokkal kell megismerkedni. Az adatbázishoz kapcsolódás, adatokhoz való hozzáférés komponensei eddig teljesen ismeretlenek voltak. Az felhasználókkal történő kapcsolattartáshoz használt vezérlőknek azonban van ismert párja pl. Tedit →TDbEdit, ezért ezek esetében csak az új tulajdonságokra kell felhívni a figyelmet (pl. DataSet, DataField, stb.) Ez a módszer lehetőséget ad Alias-ok használatára és végső esetben ha olyan formátumot használunk amit a Borland adatbázis motor (BDE) nem támogat natív módon akkor Open DataBase Connection (ODBC) bejegyzésekkel is dolgozhatunk. Ezekhez azonban rendszergazdai jogosultság szükséges, ami azzal a veszéllyel, hogy a diákok bármit telepíthetnek, törölhetnek a gépekről ezért vannak olyan iskolák ahol ezt nem vállalják fel. Az első feladat az adatok megjelenítése, ehhez a már említett komponensek mellett a DBGrid és a DBNavigtor komponenseket használjuk. Adatkapcsolat létrehozása, bontása adatok sorrendjének változtatása, a rács testre szabása jelentik és hasonló feladatokkal kezdhetjük az ismerkedést. Az adatfelvitelt először egyszerűen az adatrács felhasználásával majd DataControls paletta komponenseivel végezhetjük el. Fel kell hívni a figyelmet, hogy bizonyos adatok nem vagy korlátozottan jeleníthetők meg a rács segítségével ilyenek a képek vagy a feljegyzés típusú értékek. Érdemes rámutatni, hogy nem érdemes mindig adatbázis szervert alkalmazni, egyszerűbb feladatok esetében előnyösebb az ilyen megoldás viszont az adatbázis motornak vagy az ODBC –nek telepítve kell lenni a felhasználás helyén. Nagyon fontos, hogy kialakuljon a

megfelelő szemlélet. A komponensekből képesek legyenek felépíteni a „csővezetéket” ami összeköti a fizikai adattárat az adat megjelenítési komponensekkel. Fontos, hogy tisztában legyenek azzal, hogy a memóriában lévő tartalom nem feltétlenül egyezik meg a fizikai fil e tartalommal. Tudniuk kell, hogy az adattár aktuális állapotától (closed, browse, insert, edit) függően milyen művelet hajtható végre, hogyan lehet egyik állapotból a másikba juttatni az adattárat.

Egy másik módszer az Activex Data Objects alkalmazása. Az ADO a Microsoft legújabb és legnagyobb teljesítményű adathozzáférési modellje, amely az úgynevezett OLE DB platformra épült. Mára ez lett a legelterjedtebb adatelérési interfész a Windows alatt. Az ADO olyan programozási objektumokat jelent, amelyek az adatbázist és a benne tárolt adatok felépítését jelentik. Segítségükkel táblák, jelentések hozhatók létre illetve módosíthatók. Megvalósítható az adatok védelme, valamint minden olyan adaterőforrás elérhető amely az ODBC illesztő programok segítségével elérhető, azonban ez gyorsabb hozzáférést biztosít valamint nincs szükség operációsrendszer szintű beavatkozásra a felhasználók gépén. Ez a módszer könnyebben alkalmazható osztálytermi követelmények között. Nem csak oktatási szempontból fontos, hogy ez az adatbázis-architektúra hatékony támogatást nyújt hálózati és internetes hozzáférés megvalósítására is. Ez kapcsolódási pont a hálózati ismeretek tantárgyhoz. Lehetőség nyílik a másik tárgy elméleti tudásának a gyakorlatban történő prezentálására. Az ADO és az OLE DB együttesen alkotják az úgynevezett Universal Data Access stratégia alapjait, melynek célja valamennyi adatformátumhoz történő egységes hozzáférés biztosítása. Az ADO tulajdonképpen egy COM felületre épülő szolgáltatásszerver, melyet egy dinamikus csatolású könyvtárban (DLL-ben) található, ami a kliensgépen fut. Ez a fájl része a Windows rendszereknek a Windows2000 óta része az operációs-rendszernek ezért telepíteni sem kell. A megvalósítás előnye, hogy magyarított változata is létezik tehát például a hibaüzenetek magyarul jelennek meg, ami kezdő programozók esetében nagy előny. A komponensek nagyon hasonlóak az előző tananyagban megismert BDE komponensekhez ezért nem szükséges sok időt vesztegetni az ismertetésükre, inkább a különbségekre kell felhívni a figyelmet. Az adatbázis gyakorlatokon először az Office csomag MS ACCESS programjával ismerkedtek meg a diákok, így kézenfekvő első adatbázis kezelőnek ezt választani. A minta adatbázisok így egyrészt már rendelkezésre állnak, de ha újakra van szükség, akkor sem okoz gondot az elkészítésük, mert az ehhez szükséges ismereteknek ekkorra már birtokában vannak. Ez jelentős idő megtakarítást jelent és az új ismeretekre lehet

fókuszálni. Az első lépés itt is a kapcsolatfelvétel az adatforrással. Szemben a helyi megoldásokkal kiaknázhatjuk a hálózat adta lehetőségeket. Könnyedén szemléltethetjük, hogy milyen problémákat okoz a konkurens hozzáférés. Nyittassuk meg ugyanazt a táblát mindenkivel és hajtassunk végre egy módosító műveletet. A tapasztalatokat közösen beszéljük meg és vonjuk le a következtetéseket. Az alkalmazások készítésekor figyelmet kell fordítani a designra is. Tudatosítani kell a tanulókkal, hogy nem elég ha egy program helyesen működik fontos, hogy a felhasználók számára könnyen áttekinthető, logikus felépítésű legyen. Az értékelésnél legyen szempont a külalak, jó ha már ilyen egyszerű felületek kialakításakor rögzül, hogy legyen igényes a megjelenésre is.

Az ADO komponensek alkalmasak nem csak adatbázisok, hanem Excel táblák és Word dokumentumok kezelésére is. Ennek jelentős gyakorlati haszna van. A gazdasági életben mindenütt ezeket a formátumokat használják, ezért ennek a tudásnak nagy hasznát veszik később a munkájuk során. A képzés felépítéséből adódóan a szövegszerkesztés és táblázatkezelés már ismert a tanulók számára, az elméleti tudással (cella, cellaformázás, függvény, szakasz bekezdés, betűtípus, stb.) már rendelkeznek és a konkrét szoftvereket is használták a gyakorlat során. Van néhány olyan tulajdonság, amelynek ismerete feltétlenül szükséges és erre az előismeretek nem terjednek ki: az Excel-táblázatok esetében a szerkezeti felépítésnek egy relációhoz kell hasonlítani, a kezelni kívánt tartományokat el kell nevezni, a cellaformátumokat (számok, dátumok) mindig be kell állítani. Az adatbázisokban tárolt adatok megjelenítésére kiválóan alkalmas a Word dokumentum. Amennyiben a Delphi Explorer (ingyenes) verzióját tudjuk csak használni akkor nincs lehetőség egy jelentés generátor alkalmazását megtanítani így ez lehet az egyik megoldás a problémára.

Az ADO komponensosztály segítségével a következő témakör az SQL alapú adatbázis kezelés ismeretit is megalapozhatjuk. Ha nem egy tanár tanítja a programozás és az adatbázis gyakorlat tárgyakat, akkor fontos az egyeztetés ennél a pontnál. Amíg az SQL ismerete nem ér el egy adott szintet – adatdefiníciós utasítások, egyszerű lekérdezések – addig a programozásban sem lehet előre lépni. Ebben a szakaszban a TADOCommand komponens lehetőségeit tárgyaljuk. Látványos eredményt érhetünk el az eredményhalmaz HTML-formátumra konvertálásával. Ha az idő engedi és a hallgatóság befogadó késsége megengedi, kitérhetünk az XML technológiára és lehetőségeire. Ezek az ismeretek nem tartoznak ugyan a törzsanyaghoz, de jelentős kereslet mutatkozik erre a tudásra. Színesíthetők az alkalmazások

bináris állományok kezelésével. A képek, hosszabb szövegek alkalmazása általában növeli a lelkesedést az órákon.

Az adatbázis programozás utolsó témaköre az SQL alapú adatelérés. Ebben a részben tulajdonképpen két stratégiát kell ismertetni. A gyakorlatban az ügyfél-kiszolgáló struktúra oktatására van lehetőség. A többrétegű architektúra megvalósítása meghaladja az iskola infrastrukturális lehetőségeit és a munkájuk során sem találkoznak vele a végzett tanulók. Ezért ennek ismertetésére csak elméletben kerül sor.

Az ügyfél-kiszolgáló struktúrára példa a fentebb ismertetett program. Az eddigi példák alkalmasak voltak az otthoni gyakorlásra esetleg beadandó feladatokat lehetett adni. Azonban adatbázis-kiszolgáló szerver nincs telepítve az otthoni gépeken, sok esetben nem is alkalmasak erre. Ha a hardver megfelelő, akkor is olyan adatbázis adminisztrátori ismeretek szükségesek, amelyek nem részei a szak követelményeinek és sajnos idő sincs az elsajátítására. Ezt figyelembe véve ebben a szakaszban több gyakorlási lehetőséget kell biztosítani és nagyobb az igény a személyes segítségnyújtásra. A tananyag ezen részének feldolgozásághoz elkerülhetetlen az SQL nyelv készségszintű ismerete. Szemben az eddigi technikákkal ahol lehetőség volt a procedurális feldolgozásra itt a halmazorientált feldolgozáson van a hangsúly. A modellben lehetőség van az alkalmazáslogikát a szerver oldalon, azaz az adatbázisban megvalósítani ez a másik tárgy témaköre illetve kliens oldalon az alkalmazásban. Ha a rendelkezésre álló időbe belefér érdemes mid két megoldásra példát mutatni és a különbségeket kiértékelni. A megelőző technikáknál megismert komponensek nem alkalmazható az ilyen feladatokhoz, de nagyrészüknek meg van a modellben alkalmazható párja. Az eddig alkalmazott TDataBase komponensnek a kapcsolat kialakításához szükséges egy ún. Session azaz környezet. Ez egy új elem és nem egyszerű a szerepét bemutatni sem. Ehhez egy olyan példát célszerű mutatni, amelyik több különböző DBMS-sel tartják a kapcsolatot. Az adatbázis kezelő kiválasztásánál több szempontot kell mérlegelni. Mindenképpen „eladható” tudást kell adnunk ezért az olyan rendszerek, amelyek ugyan kiválóan alkalmasak, de környezetünkben nem terjedtek el (pl. DB2, InterBase) nem jó választások. A Web-es környezetben gyakran alkalmazott MYSQL mellett az Oracle és a Microsoft SQL Server 2005 mellett döntöttünk. Ezek egyrészt piacvezető alkalmazások másrészt létezik belőlük ingyenesen használható változat ezért akár szakdolgozatokban is alkalmazható. Ezeket a kiszolgálókat a Delphi kiemelten támogatja. Az egyik újdonság a kapcsolat felépítésében rejlik. Bonyolultabb feladat mint eddig, speciális paramétereket kell

megadni. Más tantárgyakból származó ismeretek felelevenítésére van szükség –hálózatok, operációs rendszerek. Az adatbázis kezelés olyan szegmensei kerülnek előtérbe amelyekkel eddig nem találkozhattak a tanulók. Ilyen például a többszintű jogosultság kezelés, amelyet csak az elméleti tanulmányaikból ismertek eddig. Ez is egy olyan pont ahol feltétlenül szükséges a szinkronizáció az adatbázis órákkal. A témakör megkezdése előtt feltétlenül szükséges, hogy az előbb említett szerverek közül legalább az egyiket felhasználói szinten ismerjék a tanulók. Legyen érvényes felhasználói nevük és legyenek általuk ismert adatbázisok. A legtöbb problémát az adatok egyirányúságának megértése okozza. Többször előforduló probléma, hogy nem értik meg azt, hogy két alkalmazásnak kell együtt működni és amit a képernyőn látnak az csak a gépük memóriájában létezik. A távoli gépen tárolt adatbázis tartalma már régen megváltozott. Az SQL gyakorlatban történő alkalmazása az előző feladatokban az adat lekérdezésre szorítkozott az adatmanipulációs utasításokat most alkalmazzák először ami különösen egy paraméteres megvalósítás esetén sok hiba forrása lehet. Ezek egyediek mindenkinél máshol fordulnak elő, ezeket a hibákat csak egyedileg lehet javítani. Ez jelentősen lassítja az óra tempóját. Ha vannak gyorsabban haladók, akkor olyan feladatokat kell kitalálni amelyek tartalmaznak számukra önállóan megoldandó részeket. A szakdolgozatban szereplő mintapélda az adatbázis kezelés záró példája. Nem csak erre a témakörre mutat egy átfogó példát, hanem egy komplex alkalmazás megvalósításra is.

A tematikában szerepel grafikaprogramozás. Ez az a témakör amelyet legnehezebb feldolgozni. A nehézség nem a programozási ismeretek hiánya, hanem a matematikai alapok elégtelensége. A számítástechnikai programozó szakon kevés a matematika tanítására meghatározott óraszám ahhoz, hogy erre a grafika tárgyat alapozni lehetne. Amit így el lehet érni az inkrementális vonal- és körrajzolás, síkidomok megjelenítése és a beépített függvények használata. A görbék nem beszélve a felületek szimulációját csak elméletben tárgyaljuk.

A fennmaradó időt a programokhoz kapcsolódó tevékenységek bemutatására és a szakdolgozatok során felmerülő problémák megoldására lehet felhasználni. A súgó megvalósítható a klasszikus Windows stílusban vagy Web-es felülettel. A kész alkalmazásokhoz telepítőkészleteket lehet létrehozni stb…

A második programozási nyelv oktatása párhuzamosan zajlik. Az elérendő cél az, hogy az elsődleges nyelven megírt programot a tanuló képes legyen átfordítani. A Delphi mellett ajánlott, hogy a másik nyelv valamelyik C típusú legyen. Így jelentősen szélesebb lesz

a diákok látóköre. Ezek a nyelvek a hatékonyságot helyezik előtérbe a programozás klasszikus elveivel szemben így jól kiegészíti a Pascal alapokra épülő Delphit. Nem cél és nem is lehet egy két éves képzésben egy másik teljes értékű fejlesztői eszköz megismertetése. Amennyire lehet a nyelv speciális lehetőségeit kell előtérbe helyezni, hogy ne C nyelven megírt Pascal programok szülessenek. Mindkét nyelv esetében a hangsúly a gyakorlati tudás minél magasabb szintű elsajátításán van a szükséges elmélet mellett.

Az adatbázis-kezelés tanítása

Ennek a tárgynak az oktatása két részre bomlik elméletre és gyakorlatra. Az elméleti rész az általános ismeretek mellett tárgyalja a hierarchikus, hálós és a relációs adatmodellt is, a gyakorlatban csak a szemantikus modell és relációs modell kerül tárgyalásra.

Elmélet

A programozási ismeretek esetében említett előképzettség hiánya ennél a tárgynál is jelentkezik. A kerettanterv tartalmaz adatbázis-kezelés témakört, de alapvető hiányosságok tapasztalhatók. Ezért a tantárgy oktatását csak a 13. évfolyam második félévében tudjuk kezdeni. Az első félévben a számítástechnikai alapismeretek, programozás, operációs-rendszerek tárgyban kerülnek pótlásra az ismeretek.

Mielőtt az adatbázis-kezeléssel foglalkoznánk az úgynevezett hagyományos fájlkezeléssel és annak nehézségeivel foglalkozunk. Alap szinten megismerkednek az ismert fájlszervezési módszerekkel úgymint soros, szekvenciális, indexelt, hasing. Ezek után az alapfogalmakat kell tisztázni. A záró vizsgán is előfordulnak ebből az anyagrészből kérdések és gyakoriak az olyan hibák, amikor az adatbázis, adatbázis-kezelő fogalmai keverednek.

Az elmélet oktatásakor az adatbázisok tervezésének sorrendjét követjük. Előbb a szemantikai modellel foglalkozunk és csak utána a tényleges megvalósítással. A szemantikai modellek megértése az alapja az adatbázis szemléletben gondolkodás kialakulásának. Nem véletlen, hiszen a célja pontosan az, hogy számítástechnikai ismeretekkel nem vagy csak korlátozott mértékben rendelkező személyek számára is értelmezhető legyen a modell segítségével leírt szerkezet. Az legelterjedtebb az ER (EER) modell, ennek megtanulása nem nehéz feladat. Érdemes előbb egyszerű, köznapi példákon keresztül összetettebb feladatokat is kitűzni, mert ez jól megalapozza a relációs ismeretek bevezetését. Ezeket a példákat később felhasználhatjuk a leképzési szabályok alkalmazásához. Egy-egy feladat önálló vagy kiscsoportos munkában is kidolgozható, ezzel jól szemléltethető, hogy ugyan annak a

feladatnak több megoldása is lehetséges. Ezek a megoldások összevethetők, előny-hátrány szempontjából. Mivel programozó szakosokról van szó érdemes az UML modellel is foglalkozni, ez az objektumorientáltság megértésében segítség lehet.

A tananyag tartalmazza a hierarchikus modellt. Ez mára teljesen háttérbeszorult, inkább csak mint a kialakulás egy állomása jelentős. Alapvetően a mágnesszalagos tárolás korlátait lehet kiemelni. Befogadó nyelvű rendszer lévén a példákat Pascal nyelvbe ágyazva egyszerűen magyarázhatók. Túl sok időt nem érdemes ráfordítani, elég egy maximum két óra.

A hálós modell ugyan csak egy régebbi elgondolás azonban az Internet és az XML térhódításával ismét előtérbe kerül. Ez a legbonyolultabb struktúra amelyről szó esik. Sokkal magasabb absztrakciós szintet valósít meg mint a hierarchikus modell, de azért az alapvető elemekben hivatkozhatunk rá. Mindegyik adatmodell esetében az elemeket kössük az ER-ben megtanult fogalmakhoz. A hangsúly a rekordtípusok közötti kapcsolatok megvalósításán a SET -en van. Ez is befogadó nyelvű a kezelőnyelv azonban elég bonyolult, ez volt elsősorban az oka annak, hogy nem terjedt el szélesebb körben. Nem cél a nyelv megtanítása, elég ha a működési mechanizmust megértik.

A tantárgy legfontosabb témaköre a relációs adatmodell. Ennek tárgyalására több idő kell fordítani. A gyakorlatokon a szemantikai modell mellett csak erről esik szó, ezért az elmélettel csak később kerül szinkronba. Először a modell alapfogalmait kell tisztázni. Ezek az attribútum, domain, relációséma, reláció, adatbázis. Külön ki emelendő a háromszintű integritás feltétel ellenőrzési rendszer. Rögzíteni kell, hogy milyen szabályok adhatók meg mező, reláció illetve adatbázis szinten. Nagyon fontos a kulcsok szerepének a tisztázása. A tanult ER szemantikai modellt le kell tudni képezni a relációs sémára. Gyakorlás ként elővehetők az akkori példák. A témakör legnehezebben elsajátítható része a relációs modell műveleti része. Ennek oka a már említett matematikai hiányosságokban és az elvontabb megközelítésben rejlik. A műveletek megközelítése két úton történhet a relációs kalkulus és a relációs algebra felől. Tapasztalataim szerint az első elsajátítása a nehezebb, ráadásul az algebra közelebb visz a célhoz az SQL megalapozásához. Mind kettőt ismerniük kell a tanulóknak, de a kalkulust csak érintőlegesen. Az algebrai ismeretket felhasználva ismertetjük az SQL nyelvet. Ennek elve eltér az eddig tanult imperatív nyelvek gondolkodás módjától. Fontos, hogy megértsék a deklaratív filozófiát. Ezen a ponton kell kapcsolódni a gyakorlathoz. A tantárgy utolsó témaköre a relációs adatstruktúra helyességének a vizsgálata. Ez gyakorlathoz kötődő ismeretanyagot takar. Ennek legfontosabb eleme a normalizálás.

Több olyan szakdolgozattal találkoztam ahol ezt nem végezték el helyesen ezért gyakorlatilag újra kellett tervezni a struktúrát és jelentősen módosítani a ráépülő programot. Ilyen példák segítségével jelezni kell a művelet fontosságát. Az írásbeli feladatsor visszatérő kérdései között is szerepel valamelyik úgynevezett normálforma ismertetése.

A számonkérések írásban történnek, rövid esszékérdések, teszt és feladat megoldás formájában.

Gyakorlat

A gyakorlatokon a cél az eszköztudás elérése. Az elmélet és gyakorlat tartalma nem egyezik meg egymással. Az elmélet esetében történeti visszatekintés és matematikai háttér ismeretek is szerepelnek. A relációs modell műveleti részének taglalásakor találkozik a két óra anyaga. Az adatbázisokkal való ismerkedést szemantikai modellek alkotásával kezdjük. Ehhez az ER modellt használjuk eszközként. Az egyed, tulajdonság, kapcsolat hármasság megismerése közelebb vezet a relációs elemekhez is. A számítógépes megvalósításokkal történő ismerkedést a Microsoft Office Access programjával kezdjük. Ennek megismerése lefedi az első féléves ismereteket. Szemben az elméleti órával, amely csak egy gyakorlat három féléven keresztül tart. Az Accessben típusok megismerését követően táblákat hozunk létre. Összetettebb feladatok megvalósítások kapcsán megismerkedünk a kapcsolatok kezelésének lehetőségeivel. A lekérdezéseket az ún. QBE rács segítségével kezdjük el megismerni. Érdemes időt szakítani az Access kényelmi szolgáltatásainak megismertetésére, mint például a beviteli maszkok alkalmazása, űrlapok, stb. hiszen az irodai szoftver csomagot széleskörűen alkalmazzák. Akár kis kitérő is tehető az alkalmazás létrehozása felé, ami a Visual Basic for Application programozási nyelvvel történő ismerkedést is magába foglalhatja. Mivel programozási ismeretekkel rendelkeznek ez nem jelent nagyobb problémát. A jelentés készítés megismerése fontos feladat. A report generátorok részét képezik a mai adatbázis-kezelő rendszerek szolgáltatásainak. Az első félévet az SQL nyelv alapjainak megismertetése zárja. A lekérdezések, amelyeket menet közben létrehozta a tanulók adják az alapot. Ezeket ugyan „kattintgató” módszerrel hozták létre, de a program generálja a szükséges SQL utasítást. Előbb csak ezeket módosítgassuk például az ORDER BY –t alkalmazva rendezzük sorba az eredményeket, bővítsük-szűkítsük a mezők számát. A félév végére az elérendő cél az önálló utasítás írás maximum két tábla összekapcsolásával.

A további félévek feladata valamelyik vagy akár több olyan DBMS használatának készségszintű elsajátíttatása, amelyekre van munkaerő piaci kereslet. Ez lehet az ORACLE,

MS SQL Server, de akár MySQL is. Amennyire lehetséges adatbázis adminisztrátori ismeretekkel kiegészítve.

Motiváció

Ebben a fejezetben az Országos Képzési Jegyzékben szereplő érettségi után szerezhető számítástechnikai programozó szakmáknak (emelt szintű végzettséget adó szak) nappali rendszerű képzésben történő oktatás motivációs problémáit szeretném körüljárni. A dolgozat megírásában ezen a téren szerzett öt éves tapasztalataimra támaszkodom.

A problémát három irányból szeretném megközelíteni. Az első feladat a diákok meggyőzése, hogy vegyenek részt a képzésben. A beiratkozott tanulók esetében két szempont köré lehet csoportosítani az ösztönző tevékenységet. Egyrészt a lemorzsolódás csökkentése a lehetséges minimumra, másrészt a diákok késztetése a tananyag tökéletesebb elsajátítására, jobb tanulmányi eredmény elérésére.

A képzésre jelentkezők összetétele nagyon heterogén. A többséget a gyengébb tanulmányi eredményt elért, gyakran esti vagy levelező szakon érettségizett jelentkezők teszik ki. Számukra a felsőoktatásba bekerülés szinte lehetetlen, a megszerzett iskolai végzettségükkel nem tudnak munkahelyet találni önmaguk számára. Jelentős számban találkozunk olyanokkal is, akiknek vannak esélyeik a tanulmányaikat főiskolán vagy egyetemen folytatni, de az adott évben nem sikerült a felvételijük (különösen kiemelkedő számban találkoztunk velük a kétszintű érettségi vizsga rendszer bevezetését követően). Ezen kívül azok száma sem elhanyagolható, akik ugyan bekerültek a felsőoktatásba, de nem voltak képesek a követelményrendszernek megfelelni. Rendszerint néhány a normatív támogatásra nem jogosult önköltséges diák is csatlakozik a képzéshez, számuk nem túljelentős, ők általában munkahelyük megtartása illetve elhelyezkedési esélyeiket javító szakma megszerzése miatt döntenek a tanulás mellett. Lakóhelyük szerint megoszlást tekintve a Debrecen vonzáskörzetében lévő kistelepülésen élők vannak a legtöbben, sokan helybeliek és néhányan a környező városokból érkeznek. Sajnos általánosan elmondható a diákokról, az hogy nehéz anyagi körülmények között élnek és rossz a szociális helyzetük. Sok csonka családban élő, rendszeres szociális támogatásban, árvasági ellátásban részesülő tanuló.

Az első kapcsolat a diák és az iskola között akkor alakul ki, amikor személyesen vagy telefonon érdeklődik a jelentkező (sok esetben a szülő) a képzéssel kapcsolatban. A feltett kérdések egy része a szociális juttatásokra vonatkozik. (Sajnos szép számmal kizárólag emiatt kezdik meg tanulmányaikat, ők jellemzően nem tesznek záróvizsgát.) Ezen a téren ösztönözni kollégiumi elhelyezés segítésével, (eredményfüggő) ösztöndíj bevezetésével valamint diákmunka ajánlatok közvetítésével tudunk. A kérdések másik csoportja az oktatott tárgyakra vonatkozik. Ezért nagyon fontos, hogy aki a tájékoztatást nyújtja, legyen a megfelelő ismeretek birtokában. Azokra a kérdésekre, hogy az itt szerzett végzettséggel az elhelyezkedési esélyei hogyan nőnek nem tudunk kellően motiváló választ adni. Pozitív példaként a sikeresen elhelyezkedett tanulókat lehet említeni, valamint a továbbtanulási szándékkal rendelkezőknek a plusz felvételi pontok megszerzése lehet ösztönző.

Nagyon komoly probléma a lemorzsolódás a képzés során. Ennek okai összetettek. A diákok egy része a következő tanévben újra megpróbálja a felvételit. Természetesen, ha ez sikeres, akkor a felsőoktatásban folytatja tanulmányait. Vannak, akik nem képesek a minimális szintet teljesíteni. Számukra felzárkózási lehetőséget kell biztosítani, illetve ha ez lehetséges még a képzés elején megoldhatóvá kell tenni számukra, hogy egy másik szakra átiratkozhasson. Többen nem tájékozódnak kellően egy adott szakmával kapcsolatban és csak az első órákon jönnek rá, hogy rosszul választottak. A felzárkóztatás különösen fontos az informatika és az idegen nyelv területén. A tapasztalatok azt mutatják, hogy ezeken, a területen vannak a legnagyobb különbséget a tudásszintek között, ezekből, a tárgyakból a legnagyobb a bukási arány. Ennek egyik oka a már említett heterogén összetétel. Volt olyan tanuló, aki esti tagozaton érettségizett és gyakorlatilag itt találkozott először számítástechnikával. Ezzel szemben ugyan abban a számítás technikai programozó osztályban olyan is volt, aki két évig programozó matematikus szakot hallgatott az egyetemen. Nagy különbség mutatkozik azok között is, akik rendelkeznek számítógéppel, illetve akik nem. A tanterv kialakításakor biztosítani kell a hátrányos helyzetű diákok folyamatos felzárkóztatásának lehetőségét. Ezt az órarendben szereplő tanóra keretében kell megvalósítani erre a kerettanterv lehetőséget, biztosít. Ezeken, az órákon lehetőség van az egyéni problémák megoldására, nem okoz szorongást a diáknak azokat a kérdéseket sem feltenni, amit az osztályközösségben esetleg a mások megjegyzései miatt szégyellt volna. Nagyon hasznos és fontos, hogy a felzárkóztatást végző tanár és a tantárgyat tanító tanár folyamatosan egyeztessen a témakörök és a felmerült problémák témakörében, így lehet ez a

tevékenység nagyon hasznos. A felzárkóztatásnak nem csak azok látják hasznát, akik részt vesznek benne, hanem a többiek is. A tanórák így gördülékenyebbek, ritkábban kell megszakítani olyan magyarázatokkal, kitérőkkel, amelyek a többség számára nem szükségesek. Természetesen ez gyakorlatilag minden tantárgy esetében alkalmazható. Az idegen nyelv oktatása még nehezebb problémákat vet fel. A diákok tudása az adott nyelvből nagyon széles skálán mozog. A szélsőségek; volt olyan diák, aki angol nyelvterületen érettségizett és olyanok is, akik soha nem tanulták a nyelvet. A nyelvtudás és szak nincs szoros összefüggésben egymással. A tanulócsoportok kialakításának sem ezt kell követni. A kezdeti nyelvi szintfelmérők alapján a szaktanárok a tényleges tudás alapján alakíthatják ki a csoportokat. Ez az órarendkészítésben nehézséget, de a tanulásban, tanításban nagy könnyebbséget jelent a diáknak és a tanárnak is.

A lemorzsolódás egy másik oka a sok hiányzás. A diákok életkorukból is adódóan (19-25 év) hajlamosak az iskola helyett más programokat szervezni maguknak. Ennek egyik oka, az hogy nem köti le őket a tanóra. A tanórákat lehetőség szerint érdekesebbé kell tenni modern taneszközök használatával. Sokkal szemléletesebben be lehet mutatni egy adott problémát audiovizuális eszközök használatával. Ha alkalom adódik rá az adott témakörhöz kapcsolódó előadásokra, kiállításokra, stb. el kell vinni a diákokat. Ez a mindennapok monotonitásának megszakítása mellett az ismeretek szélesebb körű bemutatásával is segíti az oktatást. Sokszor az váltja ki az érdektelenséget, hogy a tanár mondandója nem teljesen érthető, követhető a túl sok szakkifejezés alkalmazása miatt. Természetesen ez nem azt jelenti, hogy mindig a „konyhanyelv” alkalmazandó, de mindig meg kell győződni arról, hogy a hallgatóság tudja, hogy miről beszélünk. Az óráknak nem előadás jellegűeknek kell lenniük, néha a megértést sokban segíti, ha a diákok kérdezhetnek óraközben, amikor valami nem világos a számukra. A szorosan értelmezett tananyag mellett szükség van a diákokra zúduló információ áradatban eligazodni. A különböző médiákból és az Internetről származó adathalmaz nagyon sok ismeretanyagot közvetít, de nem mindig „fogyasztható” formában, sok esetben elsősorban az Internetről származó anyagok nem ellenőrzöttek, hibákat, tévedéseket tartalmaznak. Nagyon fontos tehát segíteni ezeknek, az adatoknak a feldolgozásában, értelmezésében ekkor valóban hasznos információk birtokába jutnak a diákok és nem terjednek el tévhitek. Találkoztam dolgozatokban visszatérő hibákkal, amelyek mint később kiderült egy Internetről letöltött segédanyagból származnak. Ezeknek a hibáknak, tévedéseknek a korrigálása csak hosszadalmas munkával lehetséges. Az osztályfőnökök

szerepe lényegesen csökken a középiskolával szemben, hisz a tanuló nem tanköteles és már nem áll gyámság alatt. Azonban gyakran előfordul, hogy a hiányzások oka egy olyan probléma (családi, szociális, anyagi) amelyet nem tud feldolgozni, megoldani egyedül a diák. Ebben a fiatal felnőtt korban amúgy sem könnyű eligazodni a megváltozott elvárási szabályok között, elfogadni az új helyzetét a társadalomban. A szakképzésben nagyon sok hátrányos, sőt fokozottan hátrányos helyzetű, sérült személy vesz rész, akik a megszokottnál sokkal érzékenyen reagálnak az őket ért eseményekre. Gyakran szükségük van külső segítségre, amit máshol nem kapnak meg. Ebben lehet segítségére az osztályfőnök, aki a többi szaktanárnál jobban ismeri mind a személyiségét mind a körülményeit a problémákkal küzdő diáknak. Sokszor már az is elég, ha csak valaki meghallgatja őket valaki.

Az utolsó a szempontok közül a tanulásra ösztönzés. Ennek a motivációját megtalálni talán a legnehezebb. A tanulmányi eredmény az osztályzatokban tükröződik. Ebben a képzésben viszont a jegyeknek nincs túl nagy jelentőségük. Amíg az általános iskolában, középiskolában fontos volt, a tovább tanulás szempontjából az eredmény addig a tanulók nagy részének szempontjából itt teljesen mellékes. „Úgy is azt a szakmai bizonyítványt kapom, ha szín kettes vagyok, mint ha kitűnő” vagy „ugyan annyi plusz pontot ér a felvételin a kettes, is mint az ötös”. A képzési cél nem, az hogy a diákokat így vagy úgy bizonyítványhoz juttassuk. A társadalmi érdek azt kívánja, hogy minél jobban képzett szakemberek kerüljenek ki az intézményekből, az iskola érdeke, pedig az hogy jó híre legyen, a mai diákokért versengő időszakban ez rendkívül fontos. Ezek a célok csak úgy érhetőek el, ha minden eszközzel motiváljuk a tanulóinkat a tanulásra.

A motiváció egyik eszköze a tanulmányi eredménytől függő ösztöndíj. Az ösztöndíj bevezetése csak egy réteg számára ad ösztönző erőt és ez a réteg meglehetősen szűk. Ráadásul nem is ők azok, akiknek a legnagyobb szükségük van az ösztönzésre. Ez csak az úgymond jó tanulókat érinti, akik amúgy is motiváltak voltak valamilyen szinten. A többség számára más eszközökre van szükség, amelyeket a tanár képes alkalmazni a tanórán.

Az értékelés konkrétan az érdemjegyekben jelenik meg. Ez az ötfokú skála nem ad túl nagy mozgásteret a tanár számára. Gyakran nehéz összehasonlítani a különböző tanulók teljesítményeit. Egy adott felelet, dolgozat ugyan olyan szintre történő teljesítése nagyon eltérő mennyiségű befektetett energiát tükrözhet. Dilemmát okoz a tanárnak, hogy egy szerényebb képességű, de szorgalmas diák, akiről tudja, hogy nagyon sokat készült ugyan azt a jegyet kapja, mint egy jó képességű tanuló, aki tanulás nélkül képes ugyan arra a

produkcióra. Egyrészt elkedvetleníti a gyengébb tanulót, ha nem látja erőfeszítésének eredményét – nincs értelme tanulni, súlyosabb esetben kisebbségi érzései is lehetnek amiatt, hogy ő csak ennyire képes. Ez a másik tanuló számára sem biztos, hogy a meg felelő tanulságokkal szolgál; ő így is képes teljesíteni – nincs értelme tanulni. Másrészt, ha az azonos teljesítményt különböző képen értékeli a tanár, akkor az egész osztály számára küldhet negatív üzeneteket. Nincs egységes mérce, elfogult az osztályozás. A gyengébb képességű tanulók általában nehezen fejezik ki magukat, gyakran ez az oka annak, hogy a tudásukról nem tudnak hiteles képet adni. Szóbeli feletetés alkalmával több türelemre és célirányos segítségre van szükség, ahhoz hogy ne a verbális képességen múljon az eredmény. Ha egy kifejezés (általában olyan szakkifejezés, amelyet nem használ a hétköznapi kommunikáció során) nem jut eszébe akkor azt körül lehet íratni esetleg kisegíteni az adott szóval, ez lökést adhat a folytatáshoz. Az írásbeli számonkérés esetén az ilyen beavatkozás nem lehetséges. Az értékelés során azonban az osztály előtt rá lehet kérdezni azokra a pontokra ahol érezhető, hogy csak nem tudta megfogalmazni a mondanivalóját és a végleges érdemjegyet ennek függvényében kialakítani. A jól összeállított tesztek esetében ilyen problémák nem merülnek fel. Az azonos teljesítmények személyfüggő megítélése minden képen hibás döntés. Amit a jegy nem fejez ki az egy szóbeli kiegészítő értékelés megtehet.

A gyakorlatokon a versenyhelyzet teremtése általában inspiráló hatású. Különösen, ha valamilyen tétje van (egy jeles osztályzat vagy megajánlott jegy a következő témazárón annak aki elsőnek oldja meg a feladatot). A módszer elsősorban a jó tanulók számára jelent kihívást, viszont közben a gyengébbek segítésére több idő jut. Ha egy probléma megoldása közben megakad egy-egy diák, akkor is célszerű személyes segítséget nyújtani neki, ha ő ezt nem kéri. Ha nem tudja folytatni a feladatot, akkor elveszíti az érdeklődését és nem vesz részt a további munkában. Nem mindegy hogy milyen példákat választ a tanár. A képzéseknél általában nincs előfeltétel a választott szakkal kapcsolatban az érettségin kívül. Vannak olyanok is akik már (akár már több alkalommal) tanulták az adott témakört (pl. szövegszerkesztés, táblázatkezelés több képzésben is szerepel (ECDL, stb.)). Ezért a kiválasztott feladatoknak olyanoknak kell lenniük amely az ő érdeklődésüket is felkelti de a teljesen kezdők számára nem jelentenek plusz nehézséget. Ilyen lehet például egy olyan szöveg formázása, amely napi aktualitásokat, a diákok érdeklődésére számot tartó információkat tartalmaz. Vannak olyan tárgyak, amelyek hosszabb bevezetést igényelnek, mielőtt látványos eredményre lehet jutni benne. Ilyen például a programozás. A tantárgy neve

hallatán a dákokban kialakul egy kép, ami nem vág egybe az első néhány óra (sőt teljes első félév) gyakorlatával. Ilyenkor célszerű egy rövid bemutató során megmutatni azt, hogy hová fogunk eljutni a képzés során. És amennyire a megszerzett tudás engedi érdekes, játékos feladatokkal szemléltetni az anyagot. Például egy egyszerű képernyőre írási feladat is sokkal nagyobb érdeklődést vált ki ha színeket használunk esetleg valamilyen egyszerű animációt alkalmazunk. Ez általában azok fantáziáját is megmozgatja, akik számára a tananyag ismert és így szívesen vesznek részt az amúgy számukra rutin feladatnak tetsző munkában.

Nem csak a gyakorlaton fontos de ott nélkülözhetetlen a tanulók reakcióinak figyelése. Az óra nem lehet előadás ahol a tanár elmondja az anyagot a tanulók hallgatják és jobb esetben jegyzetelnek. Mindenképpen lehetővé kell tenni a hallgatók számára a kérdezési lehetőséget. A szakképző évfolyamokon a tárgyak nagy része olyan, amelynek ismertetése viszonylagosan sok idegen vagy szakszó használatát indokolja. Természetesen amennyire lehet kerülni kell az indokolatlan a tartalmi lényeget elfedő szakzsargon használatát, de egy szakembernek ismerni kell a munkájához szükséges kifejezéseket. A kérdezés lehetősége azért fontos, mert akár egy ismeretlen szó, akár az előadó számára teljesen logikus de a hallgató számára érthetetlen gondolatmenet meggátolja a hallottak értelmezését akkor a tanuló számára az előadás csak üres szónoklatnak fog tűnni. Ekkor unalmassá, érdektelenné válik az óra. Az a jellemző, hogy ha valami nem érthető egy valaki számára akkor ennek értelmezése másoknak is problémát jelent. Ha a tanárnak sikerül kialakítani egy olyan légkört az osztályban, melyben a kérdezési lehetőséggel élnek (és nem visszaélnek) a tanítványai akkor ez mind az órai fegyelemre (aki unatkozik előbb utóbb valamivel elszórakoztatja magát, esetleg másokat is) mind a tanítás hatásfokára nagyon jó befolyást gyakorolhat. Sokkal vonzóbb egy olyan órára bemenni a diákoknak ahol jó légkör uralkodik, lehetősége van a kérdéseire választ kapni, mint egy olyanra ahol az idő jó része a fegyelmezésre megy el és az elmondottak sem teljesen világosak számára. Az oktatott tárgyak közül sokhoz nem létezik igazán jó tankönyv sem (pl. multimédiafejlesztő, banki szaküggyintéző) így az egyetlen információforrás az órán leadott anyag, ha ez zavaros vagy érthetetlen, akkor esély sincs arra hogy ezt valaki elsajátítsa.

Kötelező házi feladatok kiadása tapasztalataim szerint nem vezet eredményre. Viszont önkéntes alapon a tananyaghoz kapcsolódó témák egyéni feldolgozása és ezekből kis előadások tartása pozitív fogadtatásra számíthat. A gyengébb tanulók számára ez egy lehetőség a jegyeinek javítására, a tehetségesebbek pedig arról beszélhetnek, ami őket a

leginkább érdekli az adott témakörben. Nem baj ha néha elkalandoznak más irányba, ezzel is színesítik a tananyagot. Ez mindenképpen a mélyebb megismerést, a kutatási kedv felébresztését szolgálja. Ez a beszámolási mód a beszédkészségüket is fejleszti, könnyebb így előre felkészülve beszélni, mint egy felelés alkalmával, amikor a feltett kérdésre azonnal kell produkálni a választ. Modern technikai eszközök használata (kivetítő, audiovizuális eszközök) még sikeresebbé teheti az előadást. Egy jól sikerült szereplés önbizalmat adhat a későbbi megszólalásokhoz is.

A gyakorlati órákon, amikor önállóan oldanak meg feladatokat a tanulók akkor nem az a cél, hogy megtaláljunk egy lehetséges megoldást. Elsősorban a természettudomány területén a matematika, számítástechnika oktatásban fontos a gondolkodás fejlesztése. Nem jó az a szemlélet, amikor a kiadott feladatot megoldottnak tekintjük az első eredmény bemondásakor. Az értékelhető, ha valaki rendszeresen a többiek előtt helyes megoldásra jut (dicséret formájában mindenképen) de itt nem szabad a feladatot befejezettnek tekinteni. A leggyakrabban egy feladatnak több megoldása is létezik. Az osztály tagjai nem valószínű, hogy mindannyian ugyanabba az irányba indultak el. Az első helyes megoldás után hagyni kell dolgozni a többieket, lehet, hogy valaki más egy hosszabb, de jó úton jár, illetve vannak, akiknek ugyan az a műveletsor végrehajtása több időt vesz igénybe. Ha a nagytöbbség végzett, akkor ki lehet elemezni a megoldásokat. Annak a megtanítása a fontos, hogy milyen módokon lehet az adott feladatcsoportot megoldani. Azok számára utat mutatni, akiknek nem sikerült ezt a feladatot megoldani, ha valaki rendszeresen kudarcot vall, akkor előbb utóbb elbizonytalanodik. Össze lehet vetni a helyes megoldásokat különböző szempontok szerint (előnyök, hátrányok a többi megoldáshoz képest). Így ösztönözhető az optimális megoldásra törekvés, ne elégedjen meg a diák, azzal hogy van egy megoldása, gondolja át többször a feladatot, keressenek analógiákat más feladatokkal. Nem szabad elfeledkezni azokról, akik nem tudtak helyes eredményt produkálni. Ők is mondják el a gondolatmenetüket, amely mentén elindultak. Lehet, hogy az elgondolás helyes csak valahol kisebb hibát vétettek és ezért nem találták meg a helyes eredményt ennek felismerése csökkenti a kudarcélményt. Azok számára, akiknek eleve hibás az elgondolásuk nagyon fontos ezt be is bizonyítani azért, hogy a hibás megoldási algoritmus ne rögzüljön. Meg kell próbálni meg találni a hiba okát és végigvezetni az eredeti elgondolást.

Az elvont fogalmak, eljárások ismertetését célszerű a hétköznapi életben előforduló analógiákhoz kötni. Sokan megrettennek egy összetettebb levezetéstől vagy egy algoritmus

bonyolultságától, pedig az alapgondolat egyszerű. Ezt egy példán keresztül szeretném bemutatni. A programozás oktatása során a különböző rendező algoritmusok működése rendszeresen problémát okoz. A feladat megoldását cédulákra írt számok sorrendbe rakásával kezdjük. Néhány diákot megkérek, hogy mondja el, hogyan végezte el a rendezést. Az alapvető rendezési algoritmusok így már rendelkezésre állnak. Ekkor az azonos megoldásokat választókból csoportokat alkotva meg lehet írni az algoritmusokat megvalósító programokat. Tanulói csoportok alkalmazása általában hasznosnak bizonyul az összetettebb feladatok megoldásánál. Azok akik egyébként kisebb lelkesedéssel vetik bele magukat az önálló munkába egy csoport munkájában szívesen részt vesznek. Ez a módszer azok számára is kedvező, akik önmagukban képtelenek lennének a feladatot végrehajtani, hiszen így létrehozás teljes folyamatában közre működnek, még ha kevésbé aktívan is mint mások.

Amint ez a fentiekből látszik az ösztönzés nem egyszerű feladat. A kitűzött célok is egészen mások lehetnek egy adott osztály egyes tagjai számára. Van olyan, akinél már az is sikernek számít, ha egyáltalán bejár az órákra és nem kallódik el. Különböző képességű diákok esetében az elégséges vagy a jeles osztályzat megszerzése jelentheti az elérendő célt. Az ők esetében egészen más a feladatot kell megoldani a pedagógusnak.

Irodalomjegyzék

Robert Vieira Kezdőlönyv az Sql Server 2005 Programozásához Szak kiadó 2006

Marcu Cantu Delphi 7 Mesteri szinten Kiskapu kiadó 2003

Kovács László Adatbázisok tervezésének és kezelésének módszertana ComputerBooks kiadó 2004

Kuzmina Jekatyerina, Dr Tamás Péter, Tóth Bertalan Programozzunk Delphi 7 rendszerben ComputerBooks 2003

Internet

http://msportal.hu/files/folders/lersok/entry190.aspx 2007.11.22

http://pcforum.hu/hirek/9365/10+eves+a+Delphi.html 2007.11.22

http://pcforum.hu/hirek/8929/Delphi+2005+reszletek.html 2007.11.12

A szakdolgozatban felhasználtam a didaktika tantárgyhoz benyújtott házi dolgozatomat.