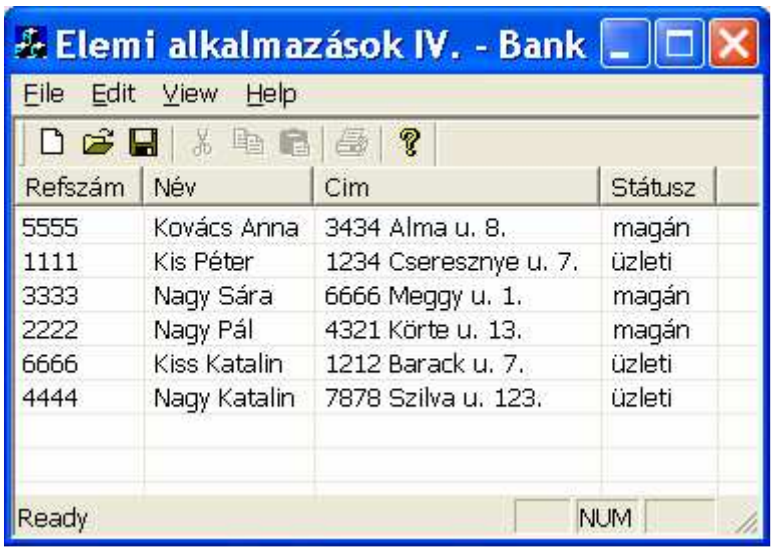
ADATBÁZISKEZELÉS KÖZVETLEN ELÉRÉSSEL – I. 2

1. FELADAT: 2
ELŐKÉSZÜLETEK: A MYSQL ADATBÁZISKEZELŐ HASZNÁLATA 2
PROJEKT LÉTREHOZÁSA 3
AZ ÜGYFÉL OSZTÁLY (UGYFEL) 4
A PROJEKT FELKÉSZÍTÉSE A MYSQL ADATBÁZISKEZELŐ HASZNÁLATÁRA 5
Illesszük be a mysql++ könyvtárat a projektbe 5
Legyen futtatható a LIBMYSQL.dll könyvtár! 5
A DOKUMETUM OSZTÁLY (CBANKDOC) 5
Előkészületek a kapcsolat kiépítéséhez 5
Adatok lekérdezése a kapcsolódás után – általános ismertető 6
A dokumentum osztály adatbáziskezelést támogató adattagjai 7
Result res; 7
Connection* con; 7
Result::iterator iResult 7
A dokumentum osztály adatbáziskezelést támogató metódusai 7
void CBankDoc::ConnectDataBase(CString s) 7
void CBankDoc::ReleaseDataBase() 7
bool CBankDoc::ExecuteQuery(CString q) 7
int CBankDoc::Count() 7
bool CBankDoc::MoveFirst() 7
bool CBankDoc::MoveNext() 7
A NÉZET OSZTÁLY (CBANKVIEW) 9
Lista típus beállítása 9
A lista kitöltését támogató metódusok 10
void CBankView::FillHeader() 10
void CBankView::FillData() 11
„Rákapcsolódás” az adatbázisra 11
void CBankView::OnInitialUpdate() 11
„Lekapcsolódás” az adatbázisról 12
void CBankView::OnFinalRelease() 12

Adatbáziskezelés közvetlen eléréssel – I.

1. Feladat: Lista

Készítsünk el egy olyan egy dokumentumos (SDI) alkalmazást, amely alkalmas a mysql adatbáziskezelővel létrehozott **BANK** adatbázisunk **UGYFEL** táblájának megjelentetésére.



Az ügyféltábla felépitése:

mysql>**use bank;**
mysql>**describe ugyfel;**

Field	Type	Null	Key	Default	Extra
refszam	int(4)		PRI	0	
nev	varchar(20)				
cim	varchar(30)				
status	varchar(8)				

Előkészületek: a mysql adatbáziskezelő használata

Ha Ön olyan gépnél dolgozik, amelyre még nem installálták fel a mysql-t, akkor hajtsa végre az 1-5 lépéseket, egyébként a 4-5. lépéseket.

1. A **mysql.com** site-ról töltese le a **mysql-3.23.52-win.zip** és a **mysql++-1.7.1-1-win32-vc++.zip** fájlokat.
2. A **mysql-3.23.51-win.zip** kicsomagolása után inditsa el a setup.exe programot. Legyen a célkönyvtár neve mysql.
3. Csomagolja ki a **mysql++-1.7.1-1-win32-vc++.zip** fájlt. Legyen a célkönyvtár neve szintén mysql.
4. Server elinditása - Inditsa el a a **mysql\bin\winmysqladmin.exe** programot.
5. Parancsablakból indítsa el a **mysql\bin\mysql.exe** programot. Ezután parancsokat adhat ki a mysql adatbáziskezelőnek, amely segítségünkre lehet programunk futása közben adataink megtekintéséhez.

Megjegyzés: *Installálás után a **mysql\docs** alkönyvtárban hasznos tudnivalókat találhat az SQL nyelvre és a mysql adatbáziskezelőre vonatkozóan.*

Projekt létrehozása

A projekt egyik érdekessége, hogy nézet osztály bázisosztályaként a **CListView** osztályt fogjuk kiválasztani. Így azonnal egy jól kezelhető, „listás” alkalmazást kapunk.

File/New/ Projects	MFC AppWizard.exe Project name: **Bank** Platforms: ☑ **Win32** **Next**	✍ *Saját jegyzeteim*
Step1:	⊙ **Single Document** ☑ Document/View architecture support? **Next**	
Step 2:	What database support would you like to import? ⊙ none **Next**	
Step 3:	*What compound document support would you like to include?* ⊙ none *What othr support would you like to include?* ☐ Automation ☐ ActiveX controls **Next**	
Step 4:	*What features would you like to include?* ☑ Docking toolbars ☑ Initial status bar ☐ Printing and print preview ☑ 3D controls *How do you want your toolbars to look?* ⊙ normal *How many files would you like on your recent file list?* **4** **Advanced:** File extension: bnk **Next**	
Step 5:	⊙ MFC Standards *style* ⊙ Yes please *comments* ⊙ As a shared DLL *library* **Next**	✍ *Nekünk kell a legördülő listából kiválasztani!!!*
Step 6:	**CBankView** Class name: CBankView **Base class**: **CListView** Header file bankView.h Implementation file: bankView.cpp **CBankDoc** Class name: CBankDoc Base class: CDocument Header file bankDoc.h Implementation file: bankDoc.cpp **Finish**	

Forditás/Futtatás

Az ügyfél osztály (UGYFEL)

Az ügyfél adatok kényelmes kezeléséhez hozzuk létre az Ugyfel osztályt!

Ugyfel.h

```
class Ugyfel
{
public:
        void SetStatus(CString s);
        void SetCim(CString s);
        void SetNev(CString s);
        void SetRefszam(int i);
        CString Status() const;
        CString Cim() const;
        CString Nev() const;
        int Refszam() const;
        Ugyfel();
        virtual ~Ugyfel();

protected:
        CString status;
        CString cim;
        CString nev;
        int refszam;
};
```

Az osztályvarázsló és/vagy gépelés segítségével önállóan deklarálja a fenti adattagokat és metódusokat.

Saját jegyzeteim

A projekt felkészítése a MySql adatbáziskezelő használatára

Illesszük be a mysql++ könyvtárat a projektbe

Workspace/File view/Bank files/(jobb egérfül)/**New Folder**
Name of the new folder: **Library Files**
File extensions:

Workspace/File view/Library Files/(jobb egérfül)/**Add files to folder**
(elérési út: c:\mysql\lib\) **mysql++**

Project(menü)/**Settings…**/”**C/C++” fül/**
Category: **Preprocessor**
Additional include directories: **c:/mysql/include,c:/mysql/mysql/include**

Legyen futtatható a LIBMYSQL.dll könyvtár!

Kétféle libmysql.dll fájl van. Az egyik a mysql\lib\debug alkönyvtárban található és ezt abban az esetben használjuk, ha nyomkövetéssel szeretnénk programunkat futtatni, a másik ugyanilyen nevű fájl a mysql\lib\opt\ alkönyvtárban van és ez akkor használatis, ha nyomkövetés nélkül futtatjuk programunkat. A megfelelő a **LIBMYSQL.dll** fájlt programunk futtatásához át kell másolnunk:
vagy: az alkalmazásunk alkönyvtárába**.**
vagy a windows/system alkönyvtárba.

A dokumetum osztály (CBankDoc)

Az adatbázissal a dokumentumosztály tartja a kapcsolatot.

Minthogy a MySQL szolgáltatásait a dokumentum osztályban használjuk, ezért a **mysqll++** header fájlt be kell illesztenünk **BankDoc.h** header fájlba. (*Természetesen, ha hivatkozni szeretnénk az Ugyfel osztályra is, akkor azt is be kell illeszteni.*)

BankDoc.h

```
// BankDoc.h : interface of the CBankDoc class

#endif // _MSC_VER > 1000
#include <mysql++>
#include "Ugyfel.h"
```

Előkészületek a kapcsolat kiépítéséhez

Egy **Connection** tipusú objektumon keresztül tartjuk a kapcsolatot az adatbázissal.

*A **CBankDoc** osztályban a **Connection** tipusú con adattag segítségével „építjük ki” a kapcsolatot az adatbázisunkkal. A **ConnectDataBase(…)** metódusban definiálunk egy kapcsolatot (dinamikus helyfoglalással) és a **ReleaseDataBase()** metódusban szüntetjük meg azt.*

Workspace/ClassView/**CBankDoc**/(jobb egérfül)/**Add member variable** …
Variable type: **Connection***
Variable name: **con**
Access: **Protected**

Workspace/ClassView/**CBankDoc**/(jobb egérfül)/**Add member function** …
Function type: **void**
Function declaration: **ConnectDataBase(CString s))**
Access: **Public**

Workspace/ClassView/**CBankDoc**/(jobb egérfül)/**Add member function** …
Function type: **void**
Function declaration: **ReleaseDataBase()**
Access: **Public**

BankDoc.h

```
class CBankDoc : public CDocument
{

// Implementation
public:
        void ReleaseDataBase();
        void ConnectDataBase();
        virtual ~CBankDoc();
protected:
        Connection* con;

};
```

BankDoc.cpp

```
// BankDoc.cpp : implementation of the CBankDoc class// CBankDoc commands
void CBankDoc::ConnectDataBase()
{
        con = new Connection(s);
}

void CBankDoc::ReleaseDataBase()
{
        if (con>0) delete con;
}

CBankDoc::~CBankDoc()
{
        con=NULL;
}
```

Adatok lekérdezése a kapcsolódás után – általános ismertető

Megjegyzés: Itt csak az általános elveket mutatjuk be, a tényleges megvalósítást **később részletesebben is megadjuk.**

1. Létrehozunk Query tipusú query objektumot

   Query query = con->query();

2. A query objektumba átirányítjuk az SQL parancsot. (mintha egy ostreambe írnánk)

   query << "select * from ugyfel order by refszam";

3. A query.store() metódussal végrehajtatjuk a parancsot és a visszaadott eredményt egy Result típusú res objektumban tároljuk

   Result res = query.store();

4. A Result osztály iterátorával res-ből rendre kiolvashatjuk a lekérdezés eredményét.

```
CListCtrl list;
Row row;
Result::iterator i;
int j=0;
for (i = res.begin(); i != res.end(); i++) {
        row = *i;
        list.InsertItem(j,row[0]);
        list.SetItemText(j,1,row[1]);
        list.SetItemText(j,2,row[2]);
        list.SetItemText(j,3,row[3]);
        j++;
}
```

A dokumentum osztály adatbáziskezelést támogató adattagjai

Result res; // az eredmény
Connection* con; // a kapcsolat
Result::iterator iResult; // az eredményt bejáró iterátor

A dokumentum osztály adatbáziskezelést támogató metódusai

void CBankDoc::ConnectDataBase(CString s)
„Összekapcsolja” programunkat a paraméterben megadott nevű adatbázissal
void CBankDoc::ReleaseDataBase()
Felszabadítja a dokumentumosztályhoz definiált adatbázist.
bool CBankDoc::ExecuteQuery(CString q)
Végrehajtja a q stringben megadott lekérdezést és tárolja az eredményt. Ha a lekérdezés kivételt dobott, akkor false egyébként true a visszatérési érték.
int CBankDoc::Count()
Visszaadja az eredményül kapott rekordok számát.
void CBankDoc::GetCurrentItem(Ugyfel &uf)
Visszaadja az aktuális ügyfél adatait.
bool CBankDoc::MoveFirst()
„Rááll” az eredmény első rekordjára. A visszatérési érték true, ha van ilyen, false egyébként.
bool CBankDoc::MoveNext()
„Rááll” az aktuális ügyfélt követő rekordra. A visszatérési érték true, ha van ilyen, false egyébként.

Az osztályvarázsló és/vagy gépelés segítségével deklarálja önállóan a fenti adattagokat és metódusokat.

CBankDoc.h

```
class CBankDoc : public CDocument
{

// Implementation
public:
        bool MoveNext();
        bool MoveFirst();
        void GetCurrentItem(Ugyfel &uf);
        int  Count();
        bool ExecuteQuery(CString st);
        void ReleaseDataBase();
        void ConnectDataBase(CString s);
        virtual ~CBankDoc();

protected:

// Generated message map functions
protected:
        Result res;
        Connection* con;
        Result::iterator iResult;
        //{{AFX_MSG(CBankDoc)
                // NOTE - the ClassWizard will add and remove member functions here.
                //    DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

CBankDoc.cpp

```
// BankDoc.cpp : implementation of the CBankDoc class// CBankDoc commands

void CBankDoc::ConnectDataBase(CString s)
{
        con = new Connection(s);
}
void CBankDoc::ReleaseDataBase()
{
                if (!con) delete con;
}
bool CBankDoc::ExecuteQuery(CString q)
{
        try {
                Query query = con->query();
                query << q;
                res = query.store();
        } catch (BadQuery er){
                cerr << "Error: " << er.error <<  endl;
                return false;
        }
        iResult=res.begin();
        return true;
}
int CBankDoc::Count()
{
        return res.end() - res.begin();
}
void CBankDoc::GetCurrentItem(Ugyfel &uf)
{
        Row row;
        row=*iResult;
        uf.SetRefszam((int) row[0]);
        uf.SetNev((CString)row[1]);
        uf.SetCim((CString)row[2]);
        uf.SetStatus((CString)row[3]);
}
bool CBankDoc::MoveFirst()
{
        iResult = res.begin();
        if ( iResult == res.end() )
                return(false);
        else    return(true);
}
bool CBankDoc::MoveNext()
{
bool CBankDoc::MoveNext()
{
        if (iResult == res.end()) return (false);
        iResult++;
        if (iResult == res.end()) return (false);
        return true;
}
}
```

A nézet osztály (CBankView)

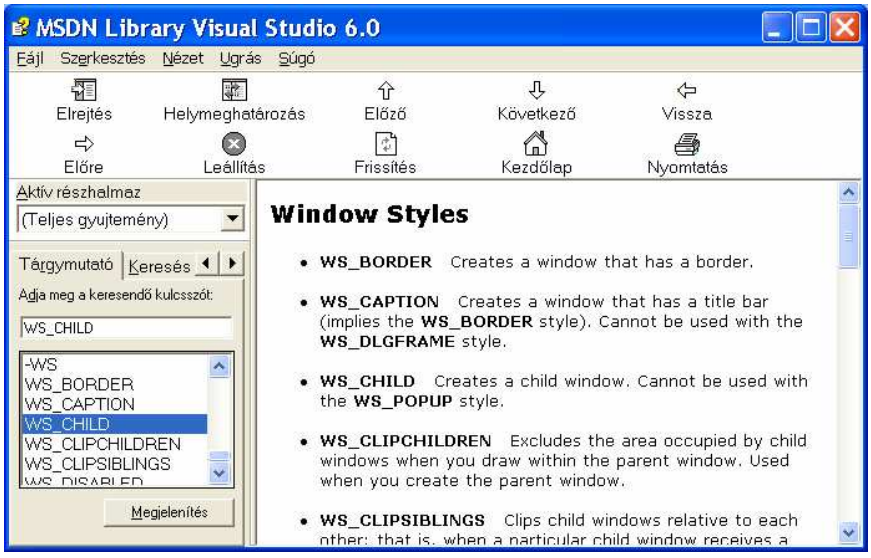
Lista típus beállítása

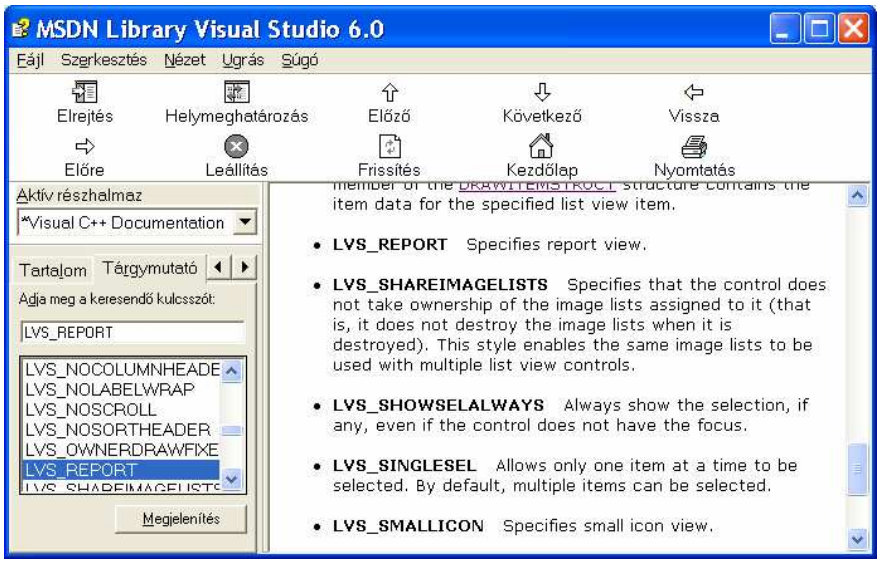
CBankView.cpp

```
BOOL CBankView::PreCreateWindow(CREATESTRUCT& cs)
{
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs
        cs.style=WS_VISIBLE|WS_CHILD|LVS_REPORT|LVS_SINGLESEL;
        return CListView::PreCreateWindow(cs);
}
```

WS_VISIBLE: Az ablak azonnal legyen látható
WS_CHILD: Az ablak legyen gyerekablak
LVS_REPORT: A lista legyen táblázat
LVS_SINGLESEL: Egyszerre csak egy listaelemet lehet kiválasztani

Gyakorlás: *Használja bátran az MSDN könyvtárat. A könyvtárban további stílusokat találhat.*





A lista kitöltését támogató metódusok

Készitsünk egy saját **– FillHeader()** – metódust, amely kitölti a listánk fejlécét!

Workspace/ClassView/BankView/(jobb egérfül)/**Add member function** …

Function type**: void**
Function declaration: **FillHeader()**
Access: **Public**

BankView.h

```
class CBankView : public CListView
{
// Implementation
public:
        void FillHeader();
        virtual ~CBankView();
}
```

Gépelje be a „tennivalókat” az alábbiak szerint!

BankView.cpp

```
void CBankView::FillHeader()
{
        CListCtrl* pList = &GetListCtrl();
        pList->SetExtendedStyle (LVS_EX_FULLROWSELECT
                |LVS_EX_GRIDLINES|LVS_EX_HEADERDRAGDROP);
        pList->InsertColumn(1,"Refszám",LVCFMT_LEFT,100);
        pList->InsertColumn(2,"Név",LVCFMT_LEFT,100);
        pList->InsertColumn(3,"Cim",LVCFMT_LEFT,100);
        pList->InsertColumn(4,"Státusz",LVCFMT_LEFT,100);
}
```

LVS_EX_FULLROWSELECT: Kiválasztásnál a teljes sort kijelöli
LVS_EX_HEADERDRAGDROP: Az oszlop címkéket „megragadva” átrendezhetjük az oszlopok sorrendjét.
LVS_EX_GRIDLINES: Legyen „vonalkázott” a táblázatunk.

Megjegyzés:

*Projektünk létrehozásakor a **CBankView** bázisosztályának a **CListView** osztályt választottuk. Az osztályhoz tartozó listát (melynek tipusa CListCtrl) a **GetListCtrl()** metóduson keresztül érhetjük el. A GetListCtrl() egy CListCtrl tipusú objektumot ad vissza, melyre már kiadhatjuk az eddig megismert üzeneteket. (InsertItem, SetItemText, stb)*

A CListCtrl osztály InsertItem(), SetItemText() metódusait az előző félév anyagaiban találhatja meg.

Gyakorlás: *Keressen további metódusokat az MSDN könyvtárban.*

Saját jegyzeteim

Készitsünk egy saját **– FillData()** – metódust, amely kitölti a listánkat az adatbázis adataival!

Workspace/ClassView/BankView/(jobb egérfül)/**Add member function** …

Function type**: void**
Function declaration: **FillData()**
Access: **Public**

BankView.h

```
class CBankView : public CListView
{
// Implementation
public:
        void FillData();
        void FillHeader();
        virtual ~CBankView();
}
```

BankView.cpp

```
void CBankView::FillData()
{
        Ugyfel uf;
        CListCtrl* pList= &GetListCtrl();
        CBankDoc* pDoc = GetDocument();
        pDoc->ConnectDataBase("Bank");
        pDoc->ExecuteQuery("select * from ugyfel;");
        pDoc->MoveFirst();
        for (int i=0; i<pDoc->Count(); i++){
                pDoc->GetCurrentItem(uf);
                CString str;
                str.Format("%d",uf.Refszam());
                pList->InsertItem(i,str);
                pList->SetItemText(i,1,uf.Nev());
                pList->SetItemText(i,2,uf.Cim());
                pList->SetItemText(i,3,uf.Status());
                pDoc->MoveNext();
        }
}
```

Megjegyzés:

A GetDocument() egy, a dokumentumunkra mutató pointert ad vissza, így a dokumentu osztályban definiált metódusainkat ezen pointeren keresztül hívhatjuk meg.

„Rákapcsolódás” az adatbázisra

Módosítsuk a nézet osztály **OnInitialUpdate()** metódusát az alábbiak szerint

BankView.cpp

```
void CBankView::OnInitialUpdate()
{
        CListView::OnInitialUpdate();

        // TODO: You may populate your ListView with items by directly accessing
        //  its list control through a call to GetListCtrl().

        GetDocument()->ConnectDataBase("Bank");
        FillHeader();
        FillData();
}
```

Megjegyzés:

A GetDocument() egy, a dokumentumunkra mutató pointert ad vissza, így a dokumentu osztályban definiált metódusainkat ezen pointeren keresztül hívhatjuk meg.

„Lekapcsolódás” az adatbázisról

BankView.cpp

```
void CBankView::OnFinalRelease()
{
        // TODO: Add your specialized code here and/or call the base class

        GetDocument()->ReleaseDataBase();

        CListView::OnFinalRelease();
}
```

Forditás/Futtatás


Saját jegyzeteim