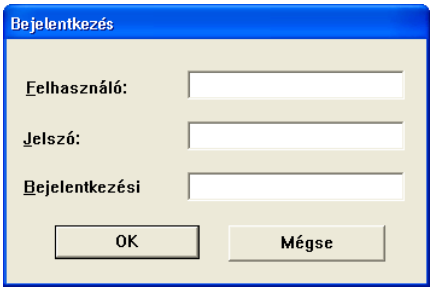
Oracle adatbázis elérése

A gyakorlat célja az, hogy a hallgató tapasztalatot szerezzen egy szerver oldali adatbázis kezelő rendszer elérésében, gyakorolja a vizuális eszközök és a kapcsolat nélküli (Disconnected Access) adatelérési modell alkalmazását. Ennek érdekében létrehozunk egy táblát az Oracle kliens programja segítségével, majd készítünk egy Windows Forms alkalmazást, amelynek segítségével bővíthetjük módosíthatjuk és böngészhetjük a tábla tartalmát. A feladat megoldásának fontosabb lépései a következők:

1. az adatbázis-tábla létrehozása és két rekord felvitele
2. kapcsolódás az adatbázishoz/adatkapcsolat létrehozása
3. kliens alkalmazás készítése
    - felhasználói felület elkészítése: fő-,adatrögzítés- és adatmódosítás ablak
    - adatok elérésének és helyi tárolásának beállítása
    - adatok böngészésének megoldása
    - új adatok felvitelének (adatrögzítés) megoldása
    - adatok módosítása
    - szinkronizálás

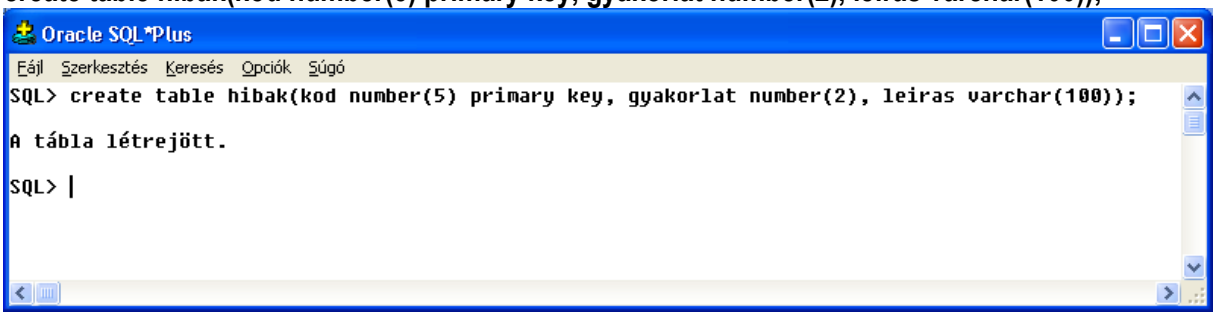
1. Az adatbázis-tábla létrehozása

Ez a részfeladat nem oldható meg Visual Studioban. Két lehetőség között választhatunk vagy webes felületen egy böngészőprogram segítségével vagy az Oracle által szállított kliens alkalmazással dolgozunk. Az alábbiakban a második megoldást tekintjük át.



Indítsuk el az Oracle SQL*Plus alkalmazást. Ez a Start menü/Programok/Oracle – OraClient10g_home1/Application Development/SQL Plus útvonalon érhető el. A megjelenő Bejelentkezés ablakban adjuk meg felhasználói azonosítónkat és jelszavunkat, a Bejelentkezési részbe pedig írjuk be az INFO_ORACLE-t. Ezután egy parancs ablak jelenik meg. Ebben hozzuk létre a táblát az alábbi SQL utasítással:

```
create table hibak(kod number(5) primary key, gyakorlat number(2), leiras varchar(100));
```



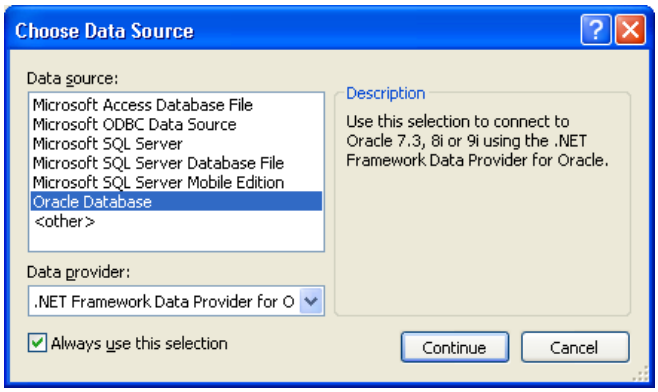
Hozzunk létre két rekordot a táblában, majd véglegesítsünk (commit):

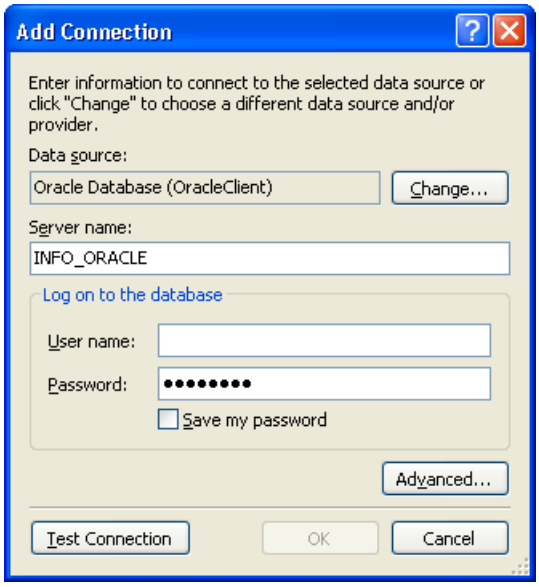
```
insert into hibak values(12456,3,'Csak egyik relációs operátort definiálta át');
insert into hibak values(32541,4,'Elírta a metódus nevét');
commit;
```

Ezután lépjünk ki a kliens programból.

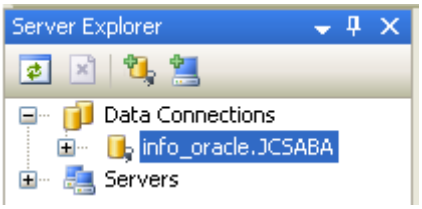
2. Kapcsolódás az adatbázishoz/Adatkapcsolat létrehozása

Indítsuk el a Visual Studio 2008-at, és a Tools menüben válasszuk ki a Connect to Database…-t.





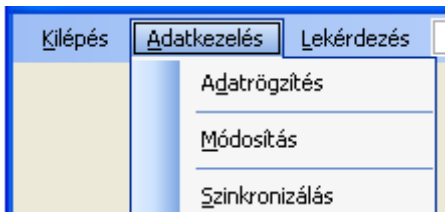
A Choose Data Source ablakban válasszuk ki az Oracle Database-t, majd Continue. Az Add Connection ablakban a Server Name-nél adjuk meg az INFO_ORACLE-t, majd töltsük ki a felhasználónév és jelszó részt. Ezután ellenőrizzük a kapcsolatot a Test Connection-nel, majd OK. Siker esetén a Server Explorer ablakban új kapcsolat jelenik meg. A kiemelten megjelenő kapcsolatnév utolsó része mindenkinél az Oracle rendszerbeli saját azonosító.



3. Kliens alkalmazás készítése

3.1. Felhasználói felület elkészítése

3.1.1. Főablak



Hozzunk létre egy Windows Application típusú új C# alkalmazást OracleProba néven. A formot tartalmazó állományt nevezzük át frmFoablak.cs-re. A form osztályát nevezzük át frmFőablak-ra, az ablak fejlécében jelenjen meg a Hibák nyilvántartása szöveg. Készítsünk menüt (msFőmenü) az alkalmazásunkhoz az alábbi menüpontokkal: „&Kilépés” (tsmiKilépés), „&Adatkezelés” (tsmiAdatKezelés), „&Lekérdezés” (tsmiLekérdezés). Az adatkezelés menü három pontot tartalmazzon egymástól vízszintes vonallal elválasztva. Ezek az „A&datrögzítés” (tsmiAdatRögzítés), a „&Módosítás” (tsmiMódosítás) és a „&Szinkronizálás” (tsmiSzinkronizálás). A Lekérdezés menü egy menüpontot tartalmazzon „Ada&trács” (tsmiAdatRács) néven.

Hozzunk létre egy új formot frmAdatRács (frmAdatRacs.cs) néven. Az Adatrács menüponthoz fogjuk majd a későbbiekben felhasználni. Az ablak fejlécében helyezzük el az „Adatok böngészése” feliratot.
Hozzuk létre a projekt osztálydiagramját.
Készítsünk egy eseménykezelőt a Kilépés menüponthoz, amelyben kilépünk az alkalmazásból.

```
private void tsmiKilépés_Click(object sender, EventArgs e)
{ Application.Exit();
}
```

3.1.2. Adatrögzítés ablak

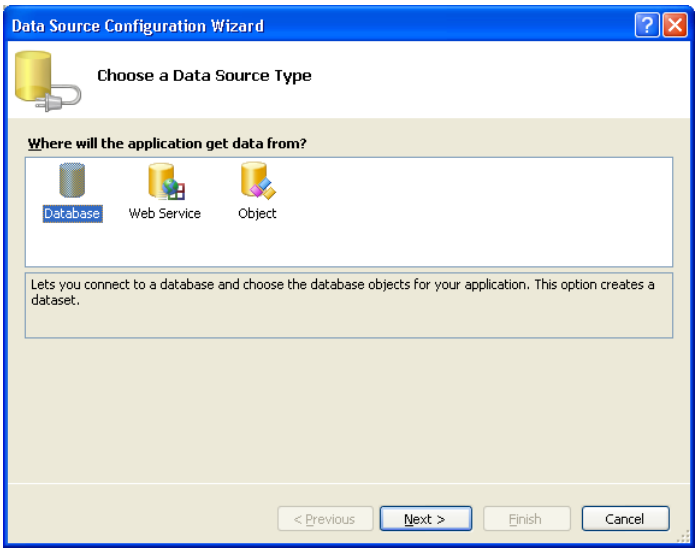
Készítsünk egy formot az adatrögzítéshez frmAdatRögzítés (frmAdatRogzites.cs) néven. Fejlécében helyezzük el az „Adatok rögzítése” feliratot. Alakítsuk ki a felületet a mellékelt ábrának megfelelően. Az első szerkesztőmező neve tbKód. A numerikus forgatómező (NumericUpDown) neve nudGyakorlat, minimális értéke 1, maximális értéke 12. A második szerkesztőmező neve tbLeírás és többsoros (Multiline=True). A Mentés nyomógomb neve btMentés. A Mégsem nyomógomb neve btMégse és DialogResult tulajdonsága Cancel.

3.1.3. Adatmódosítás ablak

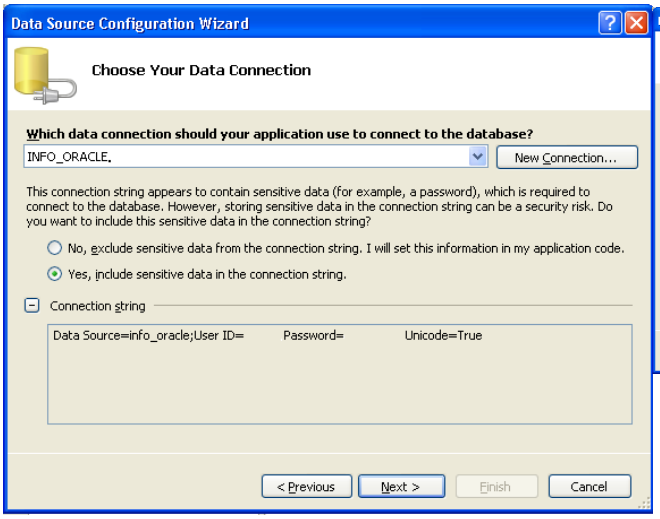
Hozzunk létre egy új formot az adatok módosításához frmMódosítás (frmModositas.cs) néven. Fejlécében helyezzük el az „Adatok módosítása” feliratot. A felület további részeinek kialakítására a 3.5. szakaszban kerül sor.

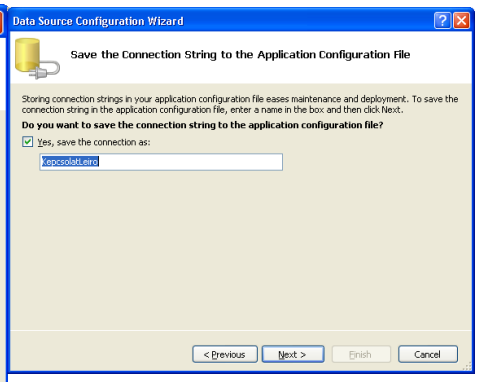
3.2. Adatok elérése és helyi tárolásának beállítása

A feladat megoldásához a kapcsolat nélküli modellt választottuk, azaz tárolni fogjuk a teljes adattáblát lokálisan egy DataSet objektumban, amit a főablak egy adattagján keresztül fogunk elérni. Ehhez az alábbi lépéseket kell végrehajtani.

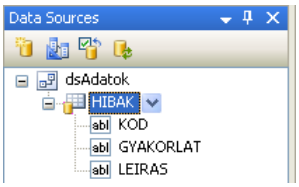


A Data menüben válasszuk ki a Show Data Sources menüpontot. A megjelenő ablakban kattintsunk az Add New Data Source… feliratra. Adatforrásként válasszunk adatbázist (Database), majd Next. Válasszuk az INFO_ORACLE.*sajátazonosító* kapcsolatot. Nyissuk ki a Connection string-et, és válasszuk a Yes, include …-t, majd Next. A mellékelt képen a felhasználó azonosító és a jelszó ki lett törölve.
Állítsuk be a kapcsolati sztring lementését KapcsolatLeíró néven, majd Next.

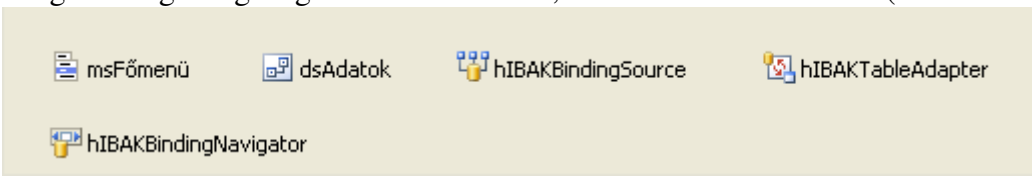




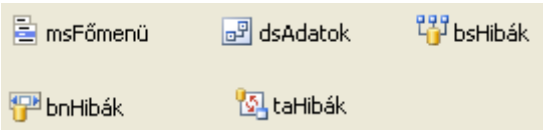
Következő lépésként az adatbázis objektumot (itt táblát) kell kiválasztani. Nyissuk ki a Tables csoportot, és a lista végén válasszuk ki a Hibak táblát. Ezután a DataSet Name mezőbe írjuk be a dsAdatok nevet, majd kattintsunk a Finish gombon. A fejlesztőrendszer elkészíti az adatbázis sémáját (dsAdatok.xsd), és generál hozzá egy DataSet osztályt dsAdatok néven. Az ebből létrehozott objektum szolgál az adatok helyi tárolására. A Data Sources ablakban megjelenik az adatforrás.



Fogjuk meg itt az egér segítségével a Hibak táblát, és húzzuk át a főablakra (frmFőablak).

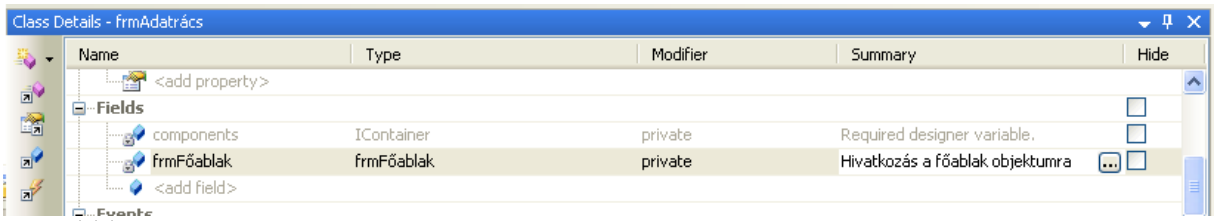


A dsAdatok objektum fog az adattábla lokális tárolására szolgálni. Tegyük ezt hozzáférhetővé a szerelvény többi része számára. Ehhez kijelöljük, majd a Properties ablakban Modifiers=Internal. A hIBAKBindingSource objektumot nevezzük át bsHibák-ra, a hIBAKTableAdapter-t taHibák-ra, és hIBAKBindingNavigator-t bnHibák-ra. Mindegyiknél végezzük el az Internal hozzáférésre történő módosítást. A főablakon levő adatrácsot nevezzük át dgvHibák-ra.

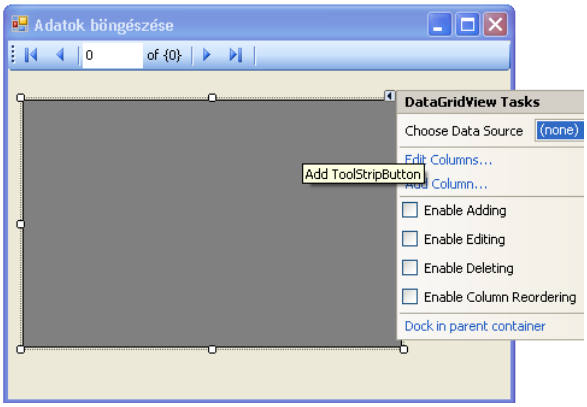


3.3. Adatok böngészésének megoldása

Az adatokat az frmAdatRács ablak segítségével tesszük böngészhetővé egy adatrács és egy navigáló sáv segítésével. Ebből az osztályból el kell érnünk a főablakot, pontosabban annak bsHibák adattagját, ami lokális adatbázisunk referenciáját tárolja. Ezért az frmAdatRács osztályon belül hozzunk létre egy frmFőablak típusú és frmFőablak nevű adattagot, majd módosítsuk a konstruktort úgy, hogy vegyen át egy hivatkozást a főablakra, és ezzel inicializálja a nevezett adattagot.




```
    /// <summary>
    /// Hivatkozás a főablak
objektumra
    /// </summary>
    private frmFőablak frmFőablak;
    /// <summary>
    /// Konstruktor
    /// </summary>
    public frmAdatrács(frmFőablak fa)
    { InitializeComponent();
      frmFőablak = fa;
    }
```



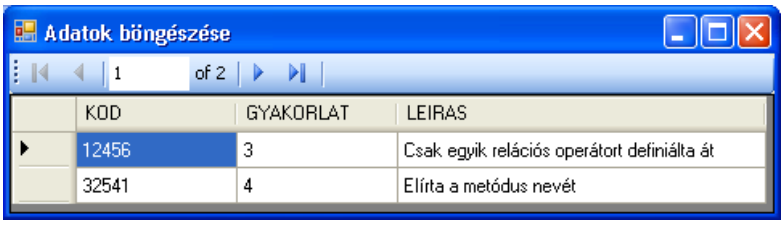
A vágólapon keresztül mozgassuk át a főablakról az adatrácsot (dgvHibák) és a bnHibák komponenst az frmAdatRács formra.
Az adatrácsnál tiltsuk le a hozzáadást, szerkesztést és törlést. Távolítsuk el a navigáló sávról a hozzáadást, törlést és mentést lehetővé tévő három nyomógombot. Állítsuk be a szülő tároló kitöltését (Dock=Fill).
A form konstruktorában kössük az adatrácsot és a navigáló sávot a bsHibák adatforráshoz az alábbi két utasítással

```
    dgvHibák.DataSource = frmFőablak.bsHibák;
    bnHibák.BindingSource = frmFőablak.bsHibák;
```

Ezek hatására a program futása során az ablak megjelenésekor láthatóvá válik az adattábla tartalma.
Készítsünk egy eseménykezelőt a főablak Adatrács menüpontjához, amelyben megjelenítjük az *Adatok böngészése* ablakot.

```
    private void tsmiAdatrács_Click(object sender, EventArgs e)
    { frmAdatrács frmAdatrács = new frmAdatrács(this);
      frmAdatrács.ShowDialog();
    }
```

Futtassuk le a programot, és próbáljuk ki az adatok böngészését. A mellékelt ábra kell megjelenjen.



3.4. Adatrögzítés megoldása

Az új adatok felviteléhez (adatrögzítés) az frmAdatrögzítés ablakot használjuk. Készítsünk egy érvényességellenőrző metódust ebben az osztályban a Kód szerkesztőmezőhöz, amelyben ellenőrizzük, hogy a bevitt kód pozitív egész és legfeljebb ötjegyű szám-e.

```
    private void tbKód_Validating(object sender, CancelEventArgs e)
    { int a=0;
      try
      { a=int.Parse(tbKód.Text);
        if (a <= 0 || a>99999) throw new Exception();
      }
```

```
      catch
      { MessageBox.Show("A kód csak pozitív egész és legfeljebb" +
          " ötjegyű szám lehet!", "Hibás kód",
          MessageBoxButtons.OK, MessageBoxIcon.Error);
        e.Cancel = true;
      }
    }
```

Készítsünk mindhárom adatbeviteli komponenshez egy-egy nyilvános csak olvasható tulajdonságot, ami lehetővé teszi a bennük megadott értékek lekérdezését a formon kívülről.

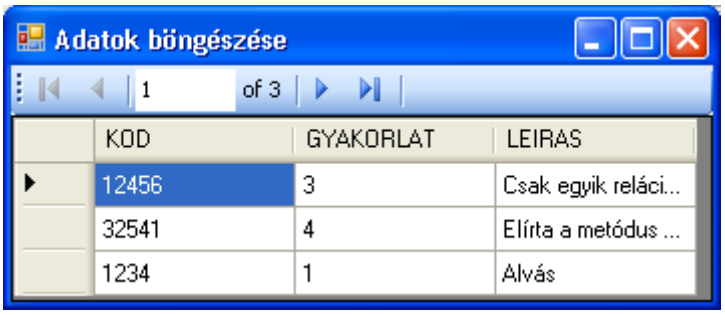
```
    public decimal Kód
    { get
      { return int.Parse(tbKód.Text);
      }
    }
    public decimal Gyakorlat
    { get
      { return nudGyakorlat.Value;
      }
    }
    public string Leírás
    { get
      { return tbLeírás.Text;
      }
    }
```

Készítsünk egy eseménykezelőt a Mentés nyomógombhoz, amelyben ellenőrizzük, hogy ki van-e töltve a Kód mező. Ennek hiányát hiba-üzenetablakkal jelezzük a felhasználó felé, majd a tbKód-hoz irányítsuk az input fókuszt. Készítsünk hasonló ellenőrzést a leírás szerkesztőmezőhöz is. Állítsuk a form DialogResult tulajdonságát OK-ra, majd zárjuk be a formot.

```
    private void btMentés_Click(object sender, EventArgs e)
    { if (tbKód.Text.Length == 0)
      { MessageBox.Show("A Kód mező kitöltése kötelező!",
          "Hiba", MessageBoxButtons.OK, MessageBoxIcon.Error);
        tbKód.Focus();
        return;
      }
      if (tbLeírás.Text.Length == 0)
      { MessageBox.Show("A Leírás mező kitöltése kötelező!",
          "Hiba", MessageBoxButtons.OK, MessageBoxIcon.Error);
        tbLeírás.Focus();
        return;
      }
      DialogResult = DialogResult.OK;
      Close();
    }
```

A főablakban készítsünk egy eseménykezelőt az Adatrögzítés menüponthoz. Ebben jelenítsük meg modálisan az adatrögzítési formot. Majd ha a Mentés gombbal (DialogResult.OK) zárta be azt a felhasználó, akkor másoljuk át az adatokat a helyi adathalmazba.

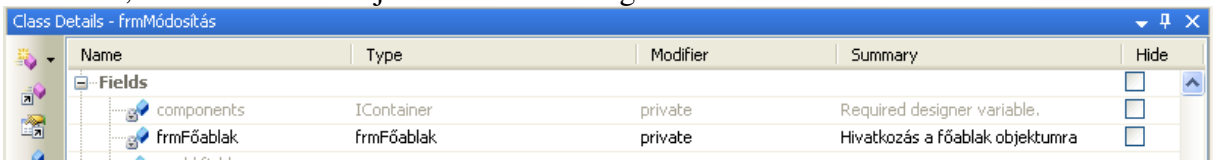
```
    private void tsmiAdatRögzítés_Click(object sender, EventArgs e)
    { frmAdatRögzítes frmAdatRögzítés = new frmAdatRögzítes();
      DialogResult Dr=frmAdatRögzítés.ShowDialog();
      if (Dr == DialogResult.OK)
      { dsAdatok.HIBAK.AddHIBAKRow(frmAdatRögzítés.Kód,
          frmAdatRögzítés.Gyakorlat, frmAdatRögzítés.Leírás);
      }
    }
```



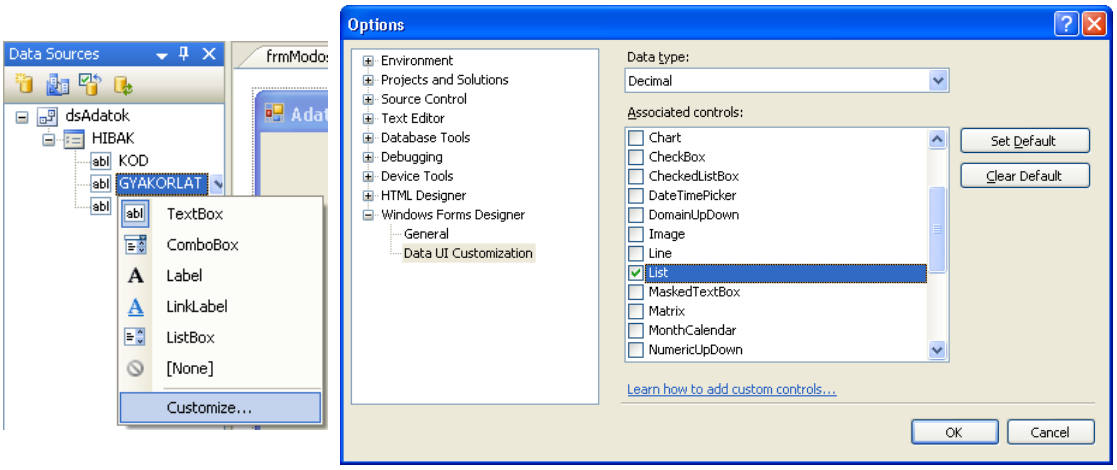
Futtassuk le a programot. Próbáljuk ki az adatrögzítést, majd a böngészést. Lépjünk ki a programból, majd indítsuk újra, és válasszuk ki az Adatrács menüpontot.
Miért nem jelennek meg az előzőekben felvitt adatok?

3.5. Adatok módosítása

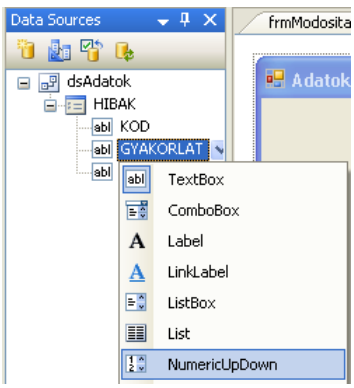
A frmMódosítás form osztályán belül hozzunk létre egy frmFőablak típusú és frmFőablak nevű adattagot, majd módosítsuk a konstruktort úgy, hogy vegyen át egy hivatkozást a főablakra, és ezzel inicializálja a nevezett adattagot.

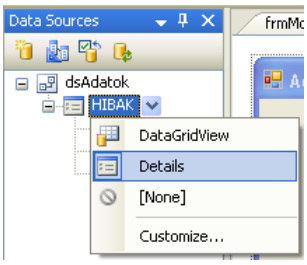



```
/// <summary>
/// Hivatkozás a főablak objektumra
/// </summary>
private frmFőablak frmFőablak;
/// <summary>
/// Konstruktor
/// </summary>
public frmMódosítás(frmFőablak fa)
{ InitializeComponent();
  frmFőablak = fa;
}
```

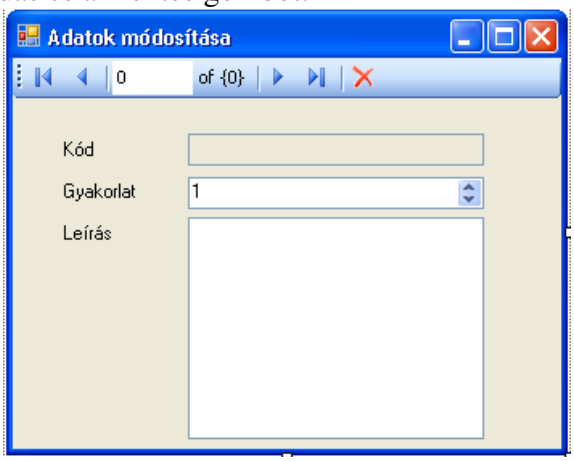


Váltsunk át tervezési (Design) nézetre, majd a Data Sources ablakban válasszuk ki a GYAKORLAT mezőt. Ezután a lenyíló listában válasszuk a Customize… menüpontot.
Az Options ablak Associated controls listájában jelöljük be a NumericUpDown-t., majd OK. Nyissuk le még egyszer a GYAKORLAT listát és válasszuk ki a NumericUpDown-t.

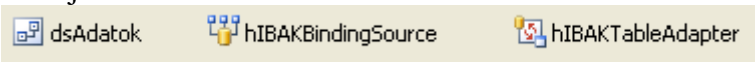




Nyissuk le a HIBAK táblához tartozó legördülő listát, és válasszuk a Details pontot.
Fogjuk meg az egérrel a HIBAK táblát és húzzuk át a formra. Írjuk át a címkéket az alábbi ábrának megfelelően. A kód szövegmezőt nevezzük át tbKódra, tegyük csak olvashatóvá és állítsuk TabStop tulajdonságát False-ra.
A Gyakorlat forgató gombot nevezzük át nudGyakorlat-ra és állítsuk be a minimális 1 és a maximális 12-es értéket.
A leírás szerkesztőmezőt nevezzük át tbLeírás-ra és tegyük többsorossá. Távolítsuk el a navigáló sávról a hozzáadás és a mentés gombot.



Töröljük le az adatmódosítás formhoz létrehozott dsAdatok, hIBAKBindingSource és hIBAKTableAdapter objektumokat. Figyelem, ha most lefordítjuk az alkalmazást, akkor a fejlesztőrendszer hibát jelez!



Nevezzük át a hIBAKBindingNavigator-t bnHibák-ra. Kössük a navigáló sávot a bsHibák adatforráshoz a form konstruktorában. Készítsük el az adatkötést a szerkesztőmezők és a forgató számára a konstruktorban az alábbiak szerint.

```
public frmMódosítás(frmFőablak ffa)
{ InitializeComponent();
  frmFőablak = ffa;
  bnHibák.BindingSource = frmFőablak.bsHibák;
  tbKód.DataBindings.Add("Text", frmFőablak.bsHibák, "KOD");
  nudGyakorlat.DataBindings.Add("Value", frmFőablak.bsHibák,
    "GYAKORLAT");
  tbLeírás.DataBindings.Add("Text", frmFőablak.bsHibák,
    "LEIRAS");
}
```

Válasszuk ki a formot, majd a Properties ablak Events gombján kattintsunk. Az eseménykezelők listájából töröljük ki a form betöltődéséhez (Load) kapcsolódó kezelőt. Ezután kódnézetben töröljük ki az eseménykezelő metódust.
Kódnézetben keressük ki azt a metódust, ami a navigáló sáv Mentés (Save) gombján történő kattintásra reagál (`hIBAKBindingNavigatorSaveItem_Click`). Tartalmát módosítsuk az alábbi kódmintának megfelelően.

```
private void hIBAKBindingNavigatorSaveItem_Click(object sender,
  EventArgs e)
{ this.Validate();
}
```

Készítsünk egy eseménykezelőt a főablak Módosítás menüpontjához, amelyben létrehozunk egy adatmódosítás formot és modálisan megjelenítjük azt.

```
private void tsmiMódosítás_Click(object sender, EventArgs e)
{ frmMódosítás frmMódosítás = new frmMódosítás(this);
  frmMódosítás.ShowDialog();
}
```

3.6. Szinkronizálás

Utolsó lépésként írjuk meg azt a kódrészt, amely gondoskodik a helyileg tárolt tábla (lokális adatbázis cache) és az adatbázisrendszerben tárolt adatok szinkronizálásáról. Ehhez készítsünk egy eseménykezelőt a főablak Adatkezelés menüjének Szinkronizálás menüpontjához (frmFőablak).

```
private void tsmiSzinkronizálás_Click(object sender, EventArgs e)
{ this.bsHibák.EndEdit();
  this.taHibák.Update(this.dsAdatok.HIBAK);
}
```