

7.3. Lokális adatbázisok kezelése Delphi alkalmazásokból

1. Halkatalógus megjelenítése [Fishes](#)
2. Képgaléria **DBGrid** segítségével [Galeria](#)
3. Grafikon a **DBChart** segítségével [Grafikon](#)
4. A **Table** vezérlő használata [Table_Peldak](#)
5. Adattábla SQL lekérdezése másik tábla mezőértéke alapján [Query_Params](#)
6. Alaptábla-részletező tábla kapcsolata (*master-detail*) [Master](#)
7. Példák SQL lekérdezésekre [SQL_Peldak](#)
8. Jelentéskészítés a **QReport** segítségével [Beszamolok](#)
9. Telefonkönyv [Telefon_Konyv](#)
10. Az ADO vezérlők használata [ADO_Peldak](#)

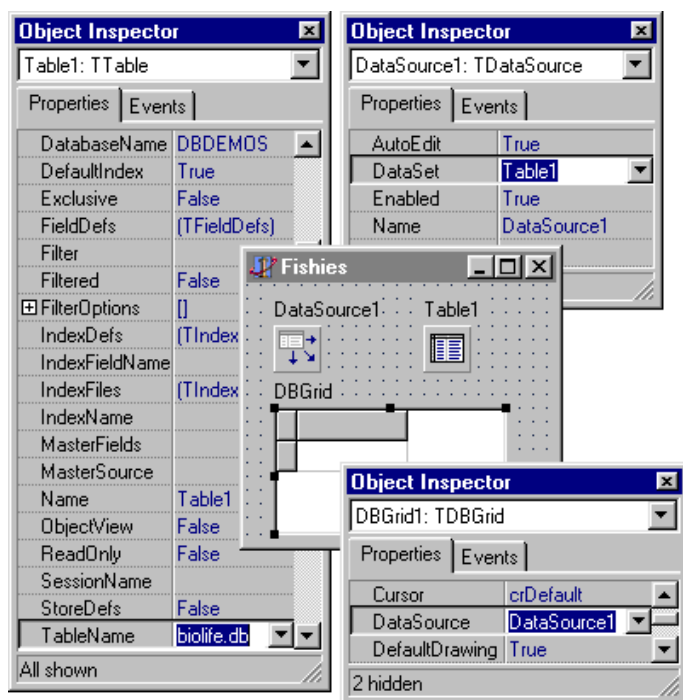


Ha az adattáblák tartalmának megjelenítésén, illetve a már létező rekordok értékeinek megváltoztatásán kívül semmi más kívánnivalónk nincs egy adatbáziskezelő alkalmazással szemben, akkor az összes munka egyetlen programsor megírása nélkül is elvégezhető.

Először is helyezzünk el a formon egy **Table** vezérlőelemet (**DataAccess** palettalapon), és állítsuk be a **DatabaseName** tulajdonságának értékét az adatbázis logikai nevére (a példában a Delphi **DBDEMOS** nevű gyűjteményét fogjuk használni, amelyben kétféle lokális adattáblát találunk: **dBASE** és **Paradox** táblákat)! Ezek után állítsuk be a vezérlőelem **TableName** tulajdonságának értékét a *biolife.db*-re!

```
Table1.DatabaseName:= DBDEMOS;  
Table1.TableName:= biolife.db;
```

Az adatkészletező vezérlőelem, azaz a **Table** megadása után következik az ún. adatforrás, vagy inkább adatközvetítő **DataSource** (**Data Access** palettalapon) formon való elhelyezése. Ahhoz, hogy a **DataSource** vezérlőelemen keresztül a következő lépésben elhelyezendő adatmegjelenítő vezérlőelemek megkapják a **Table** által összeállított rekordhalmazt, be kell állítanunk a **DataSource** vezérlőelem **DataSet** tulajdonságának értékét az adatkészletező vezérlőelem nevére:



```
DataSource1.DataSet:= Table1;
```

Az utolsó szem az adatátadó láncban egy vagy több adatmegjelenítő vezérlőelem (**Data Controls** palettalapon) lehet.

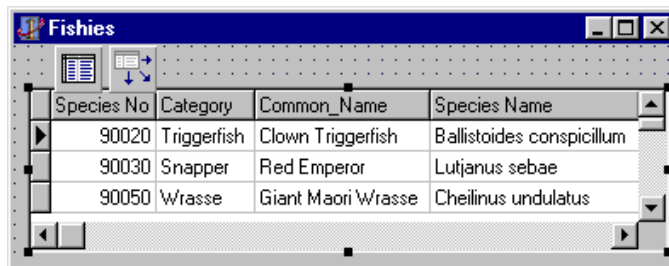
Az adatmegjelenítők közül a **DBGrid** az, amely az adattábla tartalmát szöveges táblaként jeleníti meg. Ehhez először kapcsolatot kell teremteni az adatmegjelenítő és az adatközvetítő vezérlőelemek között azzal, hogy megadjuk a közvetítő nevét az adatmegjelenítő komponens **DataSource** tulajdonságának értékeként. Ahogy az ábra is szemlélteti, a példánkban ez a következő beállítást jelenti:

```
DBGrid1.DataSource:= DataSource1;
```

Csak egy lépés szükséges még ahhoz, hogy az alkalmazásunk számára elérhetővé tegyük az adattáblában tárolt adatokat (amelyek aztán megjelennek az adatmegjelenítő vezérlőelemekben) - az adattáblát (**Table**) „meg kell nyitni”, az **Active** tulajdonság **true** értékre való állításával, vagy az **Open** metódus hívásával. Megjegyezzük, hogy a **Table** vezérlőelem **TableName**, vagy **DatabaseName** tulajdonságainak megváltoztatása automatikusan deaktiválja („lezárja”) az adattáblát. Gyakorlati szempontból ez annyit jelent, hogy a tulajdonságok megváltoztatása után újra el kell végezni az alábbi műveletek egyikét:

```
Table1.Active:=true; // fejlesztés alatt is megadható  
// vagy  
Table1.Open;
```

Az adattábla aktiválása után az adatok megjelennek a **DBGrid** rácsban.



Species No	Category	Common_Name	Species Name
90020	Triggerfish	Clown Triggerfish	Ballistoides conspicillum
90030	Snapper	Red Emperor	Lutjanus sebae
90050	Wrasse	Giant Maori Wrasse	Cheilinus undulatus

Ha most elindítva az alkalmazást, rákattintunk a rács valamelyik cellájára, átírhatjuk a benne megjelenő adatot. Az új érték azonban nem csak a rácsban lesz látható, hiszen a változtatás az adatbázisra is hatással van.

Az átírhatóságot úgy lehet megtiltani, hogy a **Table** vezérlőelem **ReadOnly** tulajdonságának értékét **true**-ra állítjuk. Amennyiben több adatmegjelenítő vezérlőelemet helyezünk el a formon, és csupán a **DBGrid** számára szeretnénk megtiltani a szerkeszthetőséget, a **DBGrid Options** halmaztulajdonságából ki kell vennünk a **dgEditing** elemet.

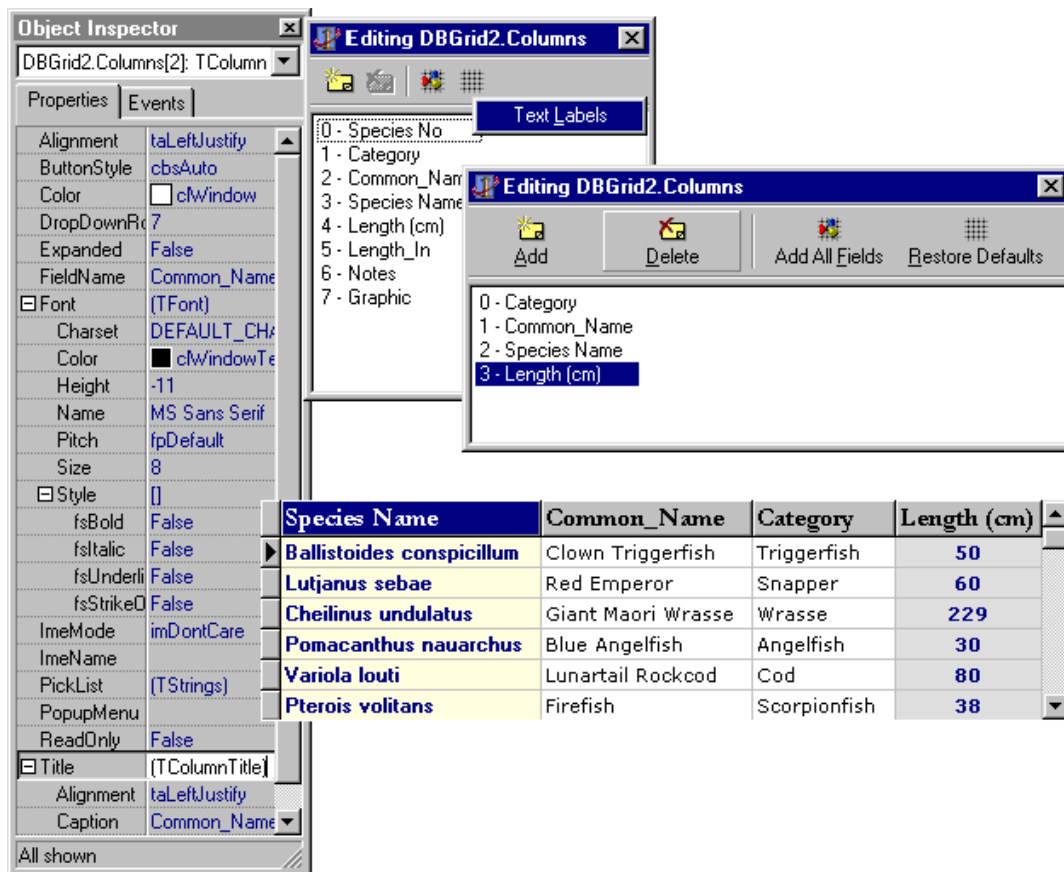
Az **Options** tulajdonságon keresztül számos más beállítást is elvégezhetünk, többek között olyanokat, amelyekkel megváltoztathatjuk a rács megjelenési jellemzőit. A **dgTitles** elemmel például be- és kikapcsolhatjuk a mezőneveket tartalmazó címsor megjelenítését, a **dgIndicator** a háromszög alakú rekordmutató, a **dgColLines** és a **dgRowLines** pedig a rács függőleges és vízszintes vonalainak láthatóságát szabályozzák.

A rács megjelenítését több más is eszközzel szabályozhatjuk, például a **DBGrid** vezérlőelem **TitleFont**, **FixedColor**, **Color** tulajdonságaival. A **Columns** jellemzővel egyes oszlopokra vonatkozó megjelenítési tulajdonságokat is beállíthatunk (**Alignment**, **Color**, **Font**, **Title**, **Width**).

A **DBGrid** vezérlőelem **Columns** tulajdonságának segítségével megadhatjuk, hogy mely adatmezők (oszlopok), milyen sorrendben jelenjenek meg a rácsban. E feladat elvégzését a Delphi egy ún. oszlop-szerkesztővel (**Columns Editor**) segíti, melyet az objektum-szerkesztő **Columns** tulajdonságából (☰), illetve a **DBGrid** felbukkanó menüjéből indíthatunk el.

A „**Columns Editor**” ablakában található gombok segítségével új mezőt hozhatunk létre (**Add**), mezőket törölhetünk a listából (**Delete**), hozzáadhatjuk a listához az adattábla összes mezőjét (**Add All Fields**) vagy visszaállíthatjuk az alapértelmezés szerinti beállításokat (**Restore Defaults**). (A gombok feliratainak be- és kikapcsolását a gombsoron jobb oldali egérbillentyűvel kattintva végezhetjük el). A listabejegyzések – és így a **DBGrid** vezérlőelemben megjelenő oszlopok – sorrendjét az elemek egymás fölé-alá való áthúzásával változtathatjuk meg. Az oszlopok beállításait - a megfelelő listabejegyzés kijelölése után - az objektum-szerkesztőben módosíthatjuk.

A példában - a következő ábrán is látható módon - megváltoztattuk a rácsban megjelenő oszlopok sorrendjét és megjelenését (ez az utóbbit az oszlopok **Alignment**, **Color**, **Font**, **Title** és **Width** tulajdonságainak segítségével). A rácsból hiányzó mezők megjelenítését pedig egy **DBEdit**, egy **DBText**, egy **DBMemo** és egy **DBImage** vezérlőelem segítségével oldottuk meg. A felsorolt vezérlőelemek **DataSource** tulajdonságának értékét a formon elhelyezkedő adatközvetítő nevére (**DataSource1**), a **DataField** tulajdonságukat pedig rendre a következő adatmezőkre állítottuk: **Species No**, **Length_In**, **Notes** és **Graphic**.



Abban az esetben, ha nem helyezünk a formra **DBGrid** vezérlőelemet, amelynek csuszkasorát használva lépkedhetünk a rekordok között, szükségünk van egy **DBNavigator** vezérlőelemre, melynek látható gombjait a **VisibleButtons** tulajdonságon keresztül szabályozhatjuk:



A **DBNavigator** vezérlőelem mindegyik gombjához tartozik egy sugóbuborék-karakterlánc (**Hints**), amely a futó alkalmazásban akkor jelenik meg, amikor egy kicsit elidőzünk az egérmutatóval a gombon. (Ehhez azonban előbb engedélyezni kell a sugóbuborék megjelenését a **ShowHint** tulajdonság **true** értékre való beállításával). A következő táblázatban a gombok nevét és a sugóbuborékjaik alapértelmezés szerinti angol nyelvű szövegét láthatjuk (amit az alkalmazásunkban természetesen át is írhatunk), a gombok segítségével elvégezhető műveletek magyarázatával együtt:

VisibleButtons	Hints	Magyarázat
<i>nbFirst</i>	<i>First record</i>	Pozicionálás az adattábla első bejegyzésére
<i>nbPrior</i>	<i>Prior record</i>	Pozicionálás az adattábla előző bejegyzésére
<i>nbNext</i>	<i>Next record</i>	Pozicionálás az adattábla következő bejegyzésére
<i>nbLast</i>	<i>Last record</i>	Pozicionálás az adattábla utolsó bejegyzésére
<i>nbInsert</i>	<i>Insert record</i>	Üres rekord beillesztése
<i>nbDelete</i>	<i>Delete record</i>	Aktuális rekord törlése
<i>nbEdit</i>	<i>Edit record</i>	Aktuális rekord szerkeszthetőségének engedélyezése
<i>nbPost</i>	<i>Post edit</i>	Aktuális rekord elküldése
<i>nbCancel</i>	<i>Cancel edit</i>	Szerkesztési művelet visszavonása
<i>nbRefresh</i>	<i>Refresh data</i>	Adatkészlet tartalmának frissítése


A beállítások eredménye fejlesztés és futás közben az alábbiakban látható:

Fishies

Species No: 90020 Length (Inch): 19.6850393700787

Notes: Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and mollusks by crushing them with powerful teeth. They are voracious eaters, and divers report seeing the clown triggerfish devour beds of pearl oysters.

Do not eat this fish. According to an 1878 account, "the poisonous flesh acts primarily upon the nervous tissue of the stomach, occasioning violent spasms of that organ, and shortly afterwards all the muscles of the body. The frame becomes rocked with



Species Name Common_Name Category Length (cm)

Ballistoides conspicillum	Clown Triggerfish	Triggerfish	50
Lutjanus sebae	Red Emperor	Snapper	60
Cheilinus undulatus	Giant Maori Wrasse	Wrasse	229
Pomacanthus nauarchus	Blue Angelfish	Angelfish	30
Variola louti	Lunartail Rockcod	Cod	80


Fishies

Species No: 90210 Length (Inch): 59.0551181102362

Notes: Young barracuda live in inshore seagrass beds, while adults range from inshore channels to the open ocean. The barracuda feeds on a wide variety of fishes.

It frequently drifts just below the surface and is known to approach divers at very close range. The long underslung jaw with its very sharp teeth can be disconcerting. Attacks on humans have reportedly been in cloudy water when the victim is wearing bright diving gear or attempting to spear the fish.

Edibility is good for small specimens, but



Species Name Common_Name Category Length (cm)

Sparisoma Aurofrenatum	Redband Parrotfish	Parrotfish	28
Sphyraena barracuda	Great Barracuda	Barracuda	150
Haemulon flavolineatum	French Grunt	Grunt	30
Lutjanus jocu	Dog Snapper	Snapper	90
Epinephelus striatus	Nassau Grouper	Grouper	91



A feladat megoldását kezdjük azzal, hogy elhelyezünk a formon egy **Table**, egy **DataSource** (*Data Access* palettalap) és egy **DBCtrlGrid** (*Data Controls* palettalap) vezérlőelemet.

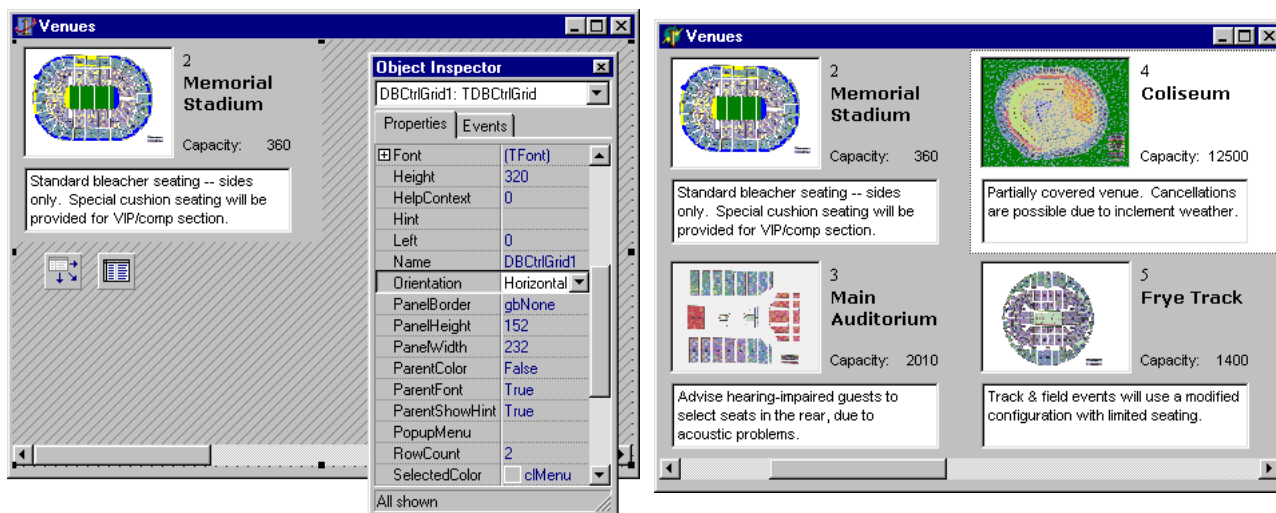
A **Table** vezérlőelem **DatabaseName** tulajdonságának értékét állítsuk be *DBDEMOS*-ra, a **TableName** tulajdonságát pedig *venues.db*-re! A **DataSource** vezérlőelemet - **DataSet** tulajdonságán keresztül – kapcsoljuk hozzá a **Table** (*Table1*) adatkészletezőhöz. A kapcsolatlánc befejező elemeként, a **DBCtrlGrid** vezérlőelem **DataSource** tulajdonságát az adatközlítő vezérlőelemre állítjuk (*DataSource1*).

A **DBCtrlGrid** segítségével tetszőleges sor- és oszlopszámú rács alakítható ki az adattáblák tartalmának megjelenítésére, amihez azonban be kell állítanunk a vezérlőelem **RowCount** és **ColCount** tulajdonságait a megfelelő értékekre (a példában 2 sort és 2 oszlopot használunk). A program futása során, az így kialakított cella adattáblák értékeit jeleníti meg, azonos elrendezésében. Az elrendezés adatainak meghatározására a **DBCtrlGrid** bal felső cellát használjuk.

Ebben a vonalmintával nem befedett cellában el kell helyeznünk azokat az adatmegjelenítő vezérlőelemeket (*Data Controls* palettalap), amelyekkel megvalósítható az adattábla mezőinek megjelenítése. Példánkban szükségünk lesz egy **DBImage**, három **DBText**, egy **DBMemo**, illetve egy egyszerű, az összes cellában egyforma „Capacity:” feliratra (**Label**).

Az adatmegjelenítő vezérlőelemek **DataSource** tulajdonságának értékeként adjuk meg az adatközlítő vezérlőelem nevét (*DataSource1*)! A **DataField** tulajdonság értéke pedig a **DBImage** esetén a *Venue_Map*, a **DBMemo** esetén a *Remarks*, a három **DBText** vezérlőelem esetén pedig rendre a *VenueNo*, a *Venue*, illetve a *Capacity* mezőnév lesz.

Végül aktiváljuk (**Active** ← *true*) a **Table** adatkészletező vezérlőelemet! Azt pedig, hogy a rácson belül a csúszkásor melyik oldalon jelenjen meg, látszódnak-e a cellákat elválasztó vonalak, milyen színű legyen a kijelölt cella, a **DBCtrlGrid** vezérlőelem **Orientation**, **PanelBorder**, illetve **SelectedColor** tulajdonságán keresztül szabályozhatjuk.





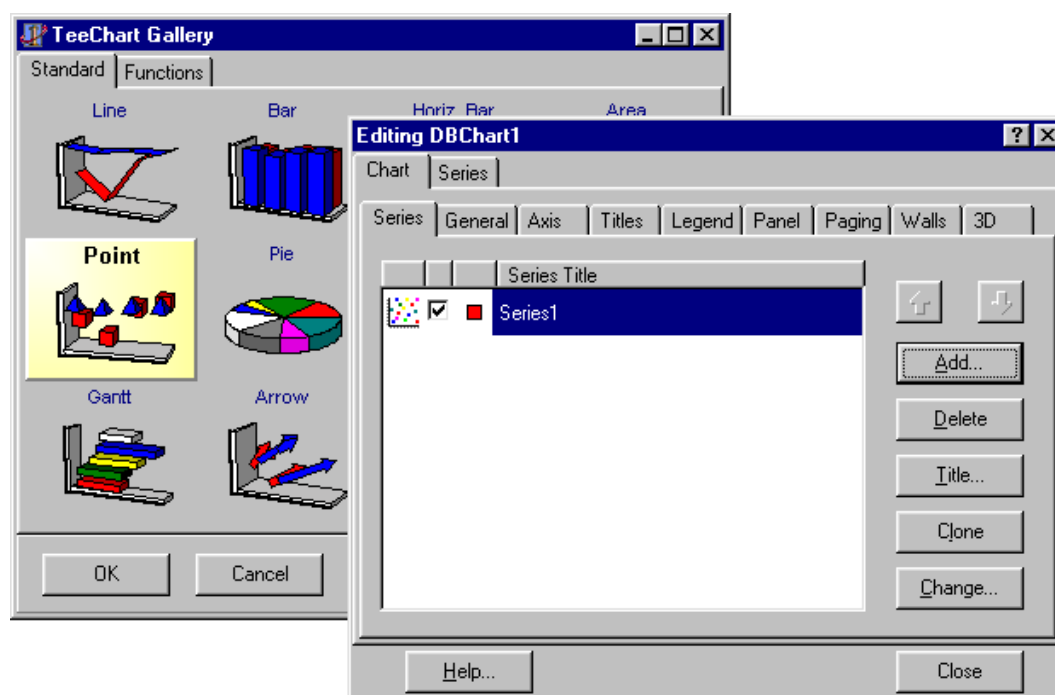
Legyen feladatunk annak grafikus megjelenítése, hogy vevőink mennyit vásároltak nálunk! A vásárlásokat a már átutalt pénzmennyiségben mérjük. Az adatforrásnak a Delphi példaadatbázisának (*DBDEMOS*) *orders.db* nevű adattábláját fogjuk használni, amelyben a *CustNo* mező a vásárlók azonosítószámait, az *AmountPaid* pedig a kifizetett összegeket jelöli.

Az adatbázissal való kapcsolat megteremtéséhez helyezzünk el a formon egy **Table**, egy **DataSource** és egy **DBChart** vezérlőelemet! Állítsuk be a **Table** vezérlőelem **DatabaseName** tulajdonságának értékét *DBDEMOS*-ra, a **TableName** tulajdonság értékét pedig *orders.db*-re! Mivel a táblázat elég széles vevőkör adatait tartalmazza, egy kis grafikonon aligha lehetne az összes vevő adatát egyszerre megjeleníteni. Ezért szűkítsük le a megjelenítendő adatok körét csak az 1300 és 1500 közé eső azonosítóval rendelkező vevőkre - adjuk meg a **Table** vezérlőelem **Filter** tulajdonságában a következőt feltételt:

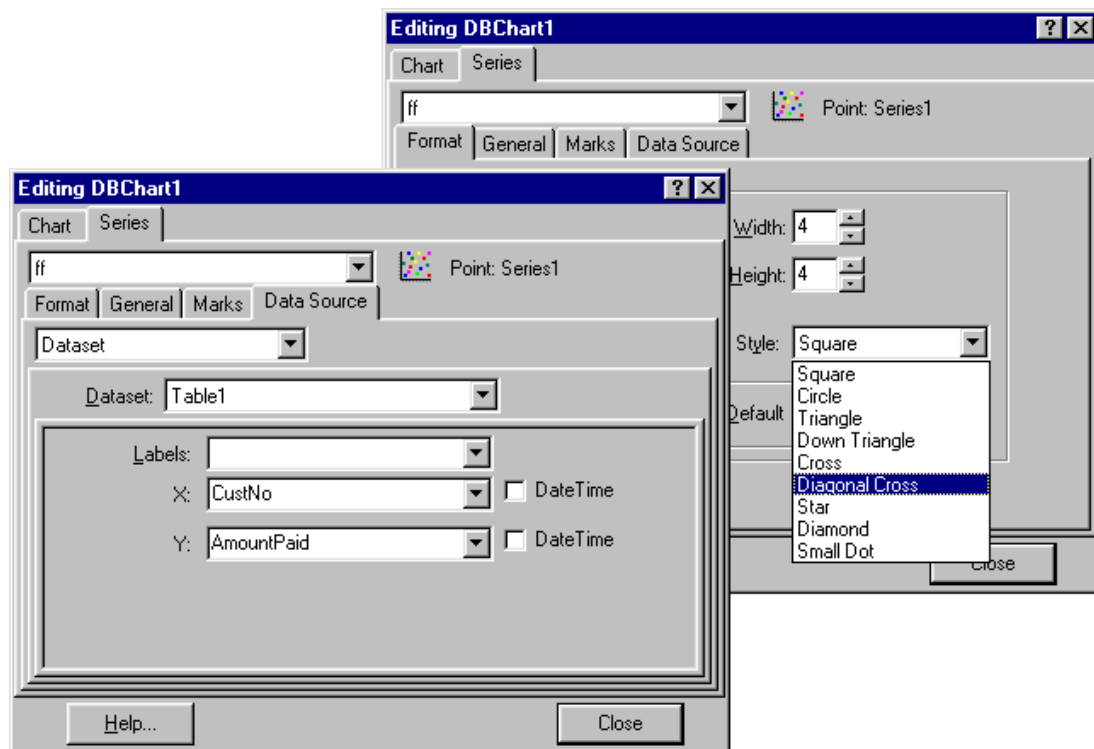
```
(CustNo > 1300) and (CustNo < 1500)
```

Ezek után állítsuk be a vezérlőelem **Filtered** tulajdonságának értékét *true*-ra! Kapcsoljuk össze a **Table** és a **DataSource** vezérlőelemet a **Table** objektum nevének (a példában *Table1*) megadásával a **DataSource** vezérlőelem **DataSet** tulajdonságán keresztül, és aktiváljuk a **Table** adattáblát (**Active** = *true*)!

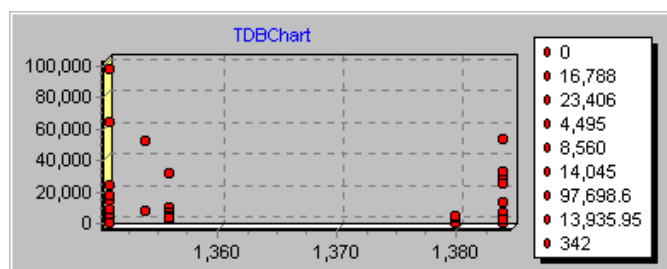
A grafikon megszerkesztéséhez kattintsunk kétszer a **DBChart** vezérlőelemen (vagy válasszuk ki a felbukkanó menüből az „**Edit Chart**” menüpontot)! A megjelenő diagramszerkesztő ablak **Chart** paneljéről válasszuk ki a diagram megjelenítési típusát, a **Series** lap **Add** gombján kattintva. (A példában a jobb áttekinthetőség érdekében a *Point* típust használjuk).



A szerkesztőablak **Series** paneljének **DataSource** lapján válasszuk ki a listából a „*Data Set*” bejegyzést! A megjelenő **Dataset** mezőben pedig adjuk meg az adatkészletező objektumunk nevét (*Table1*)! A grafikon X- és Y-tengelyén megjelenő adatsorokat a megfelelő feliratú mezőkből választhatjuk ki – a példánkban az X értékek a vevők azonosítószámai (*CustNo*), az Y értékek pedig az általuk kifizetett összegek (*AmountPaid*) lesznek. (A grafikszerkesztő ablak többi lapján módosíthatjuk a grafikon azon beállításait, amelyek a megjelenítésre vonatkoznak. Például a **Series|Format|Style** mezőből kiválaszthatjuk a pontok megjelenítésére alkalmazott alakzatot. Az alapértelmezés szerinti kocka helyett, a **Series|Marks|Visible** jelölőnégyzet segítségével a változók értékét is megjeleníthetjük)



Ezek után a grafikonunknak:

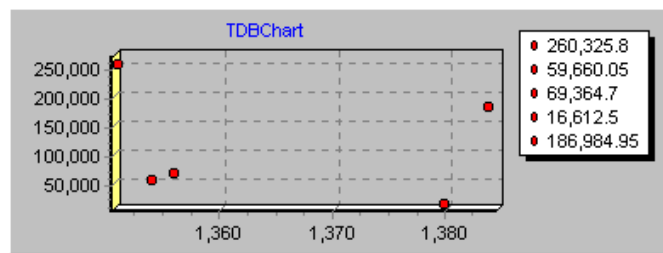


Probléma az, hogy az egyes vevők által költött pénz nem egy összegben, hanem az egyes vásárlásokra vonatkozóan jelenik meg. Ahhoz, hogy mindegyik vevőt az összes vásárlása során költött pénzmennyiséggel tudjuk azonosítani a grafikonon, a **Table** vezérlőelem helyett az SQL lekérdezések lebonyolítását támogató **Query** vezérlőelemet kell használnunk.

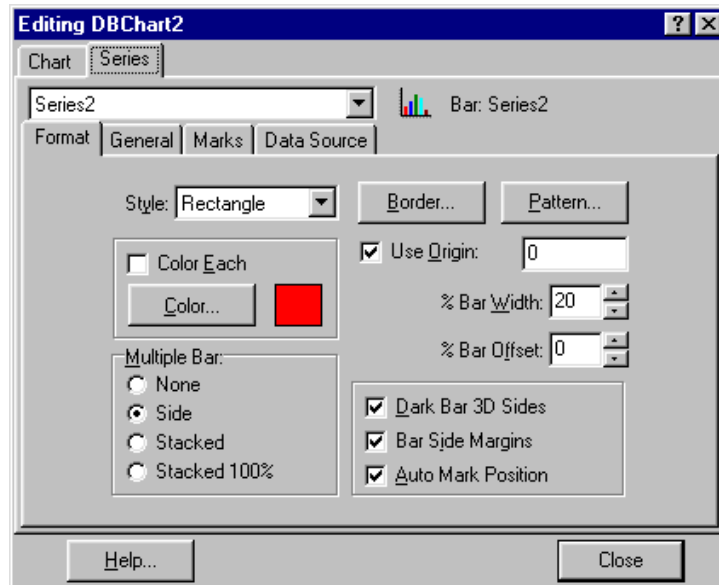
Az adatelérő objektumok láncolatában csak annyi lesz a változás, hogy a **Table** vezérlőelem szerepét a **Query** vezérlőelem veszi át, amelynek a **DatabaseName** tulajdonságán kívül csak az SQL tulajdonságát is be kell állítani:

```
SELECT CustNo, sum(AmountPaid) FROM orders WHERE CustNo BETWEEN 1300 AND 1500
GROUP BY CustNo
```

Ezzel lekérjük az *orders* nevű táblából a *CustNo* mező egyforma értékei szerint (1300 és 1500 között) csoportosított a *sum(AmountPaid)* értékeket, azaz az egyes csoportokra (vevőkre) vonatkozó *AmountPaid*-összegeket. A **Query** vezérlőelem **Active** tulajdonságának *true*-ra való beállítása után, a **DBChart** vezérlőelemen a következő grafikon jelenik meg:



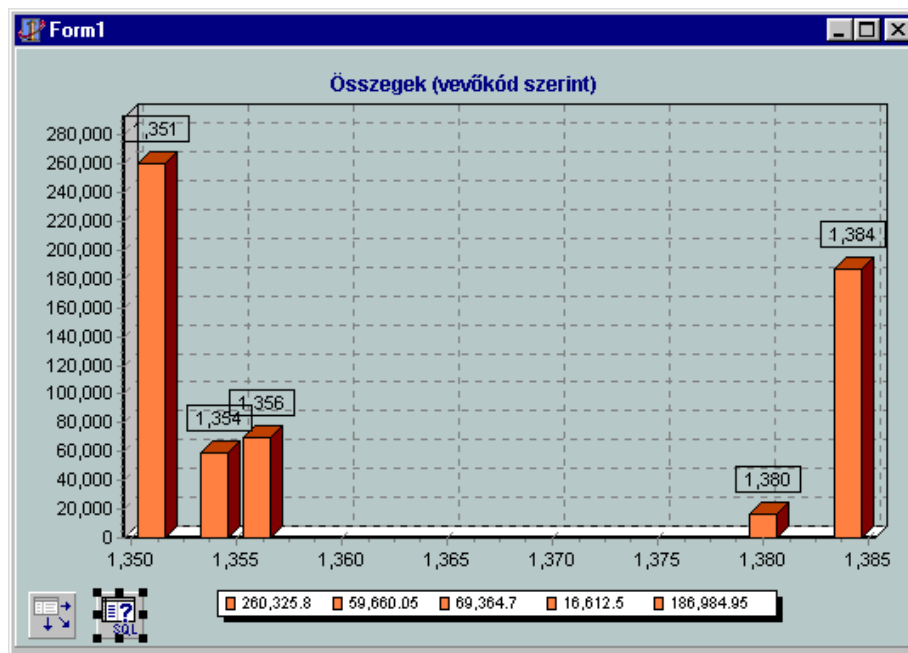
Mivel ebben az esetben összegekről van szó, a szemléltetést sokkal jobban szolgálja egy oszlopdiagram. Az átállításhoz kattintsunk kétszer a **DBChart** vezérlőelemen, majd kattintsunk a **Chart** panel **Series** lapján található **Change** gombon, és állítsuk át a típust! Az oszlopok színét, szélességét stb. a **Series** panel **Format** lapján változtathatjuk meg:



Ahhoz, hogy az oszlopok mellett a vevők neve is látható legyen, jelenítsük meg az egyes oszlopok által képviselt vevőazonosítókat is. Ehhez a **Series** panel **Marks** lapján jelöljük be a **Visible** négyzetet, a feliratok átlátszóságához a **Transparent** jelölőnégyzetet, és válasszunk ki megfelelő stílust a **Style** választógombok segítségével! (A példában **X Value**, azaz az X-tengely menti értékek.) Ha azt szeretnénk, hogy a feliratmezőket vonal kapcsolja össze az oszlopokkal, írjuk be a kívánt vonalhosszat a **Length** mezőbe!



A végeredmény – az elvégzett beállításoktól függően – például ilyen lesz:





A program megírása előtt hozzunk létre egy dBASE adattáblát a „**Database Desktop**” (**Start | Programok | Borland Delphi 5 | Database Desktop**) alkalmazás segítségével. A tábla struktúráját és adatait a következő ábra mutatja:

The screenshot shows two windows from the Database Desktop application. The top window, titled 'Structure Information dBASE III+ Table: aruraktar.dbf', displays the table's structure. The bottom window, titled 'Table : C:\...\aruraktar.dbf', displays the data records.

Field roster:

	Field Name	Type	Size	Dec
1	ARU_NEVE	C	15	
2	DARAB	N	8	0
3	AR	N	15	2
4	CEGSORSZAM	N	2	0

Table properties:

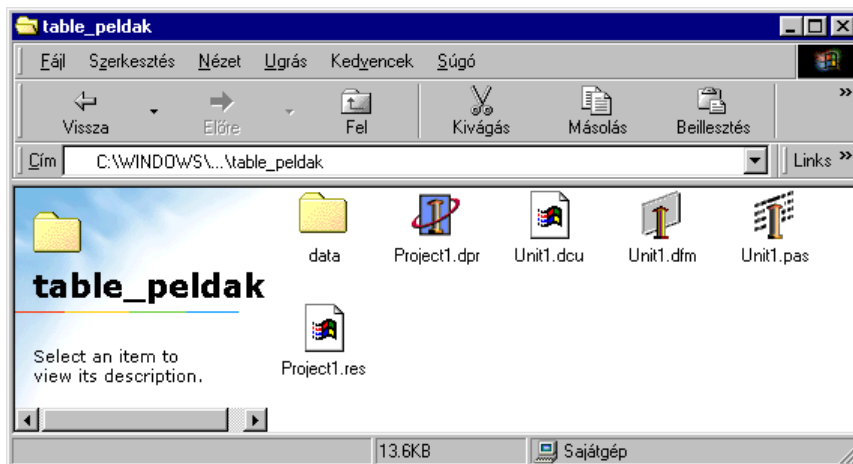
Indexes: [Dropdown]
Detail Info: [Button]

Record lock:
☐ Info size [Dropdown]

Table : C:\...\aruraktar.dbf

	aruraktar	ARU_NEVE	DARAB	AR	CEGSORSZAM
1		asztal	5	5,200.00	1
2		szek	12	2,500.00	1
3		szekreny	4	8,500.00	1
4		agy	3	12,000.00	2
5		lampa	4	2,000.00	2
6		szonyeg	7	3,400.00	3
7		tukor	2	2,300.00	3
8		fotel	4	4,100.00	1
9		allolampa	2	3,500.00	2

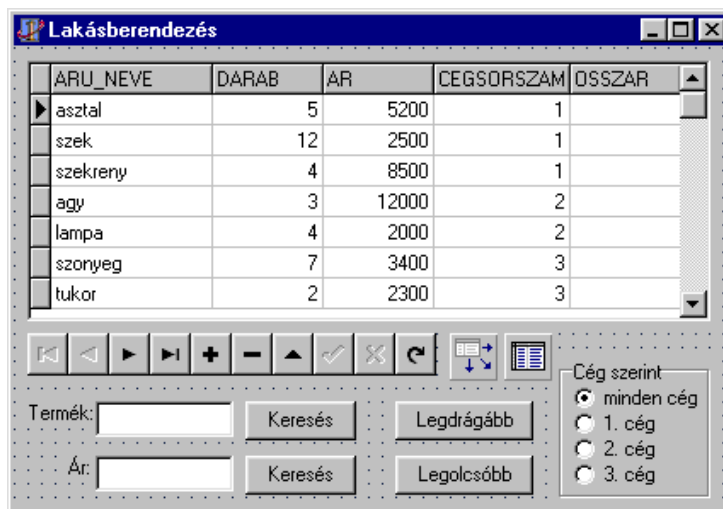
A munkát az adattábla struktúrájának kialakításával kezdjük (**File | New | Table | dBASE III+**).



A struktúra létrehozása után mentjük el a táblát (**Save As...**) *aruraktar.dbf* névvel egy *Data* nevű alkönyvtárba, amelyet előzőleg a készülő alkalmazásunk mappájában hoztunk létre!

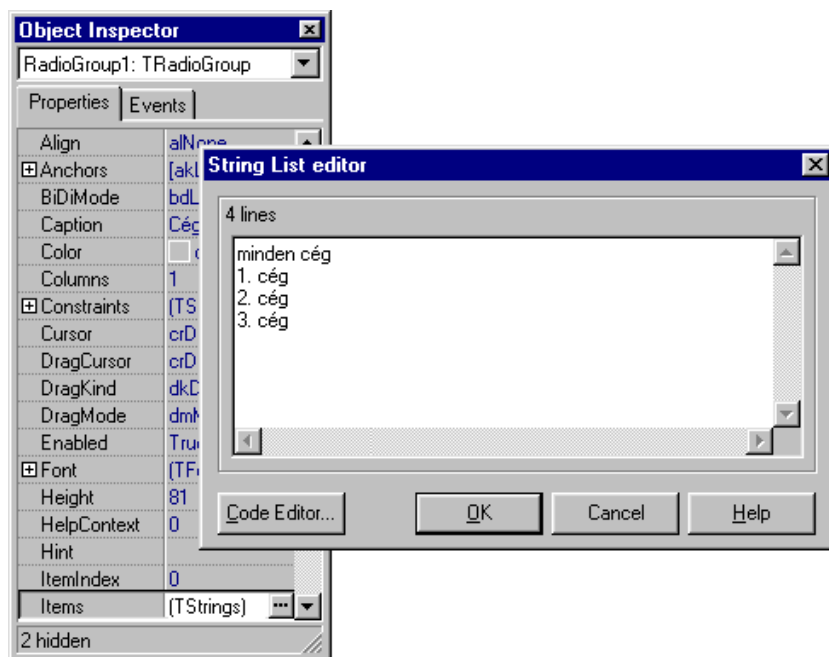
Ezek után nyissuk meg az adattáblát (**File|Open|Table**), és a „**Table|Edit Data**” menüpont bejelölése után töltjük fel adatokkal!

Készítsünk a Delphi-ben egy új alkalmazást, és helyezzük el a formon az alábbi ábrán is látható vezérlőelemeket: egy **Table**, egy **DataSet**, egy **DBGrid**, egy **DBNavigator**, két-két **Label**, illetve **Edit**, négy **Button** és egy **RadioGroup** vezérlőelemet!



Állítsuk be a **DBNavigator**, illetve a **DBGrid** vezérlőelem **DataSource** tulajdonságának értékét a **DataSource** vezérlőelem nevére (*DataSource1*), a **DataSource** vezérlőelem **DataSet** tulajdonságát pedig a **Table** vezérlőelemre (*Table1*)!

A **Table** vezérlőelem **DatabaseName** tulajdonsága értékeként állítsuk be az *aruraktar.dbf* táblánk elérési útját, illetve nevét (*..\table_peldak\data*), és állítsuk át az **Active** tulajdonságot *true*-ra!



A többi vezérlőelem **Caption** tulajdonságának értékét a fenti ábráról is leolvashatjuk.

A **RadioGroup** vezérlőelem esetén még be kell állítani az **Items** tulajdonság értékét a megjelenő választógombok felirataira, majd válasszuk ki az első gombot, megadva az **ItemIndex** tulajdonságmezőben a 0 értéket.

A **RadioGroup** vezérlőelem segítségével lehetőséget adunk a felhasználónak arra, hogy csak egy bizonyos cégre vonatkozó adatokat jelenítsen meg a **DBGrid** vezérlőelem adatrácsában.

Ezt a **Table** vezérlőelem **Filter** tulajdonságának átállításával érhetjük el, amelyben meg kell adni a szűrési feltételeket, például:

```
State <> 'CA' or State = NULL
```

vagy pedig – ha a mezőnevek több szóból állnak:

```
[Home State] = 'CA' or [Home State] = 'MA'
```

A szűrés engedélyezéséhez át kell állítani a **Table** vezérlőelem **Filtered** tulajdonságát *true* értékűre.

Példaprogramunkban a szűrő – az első cég adatainak megjelenítésékor – következőképpen néz ki:

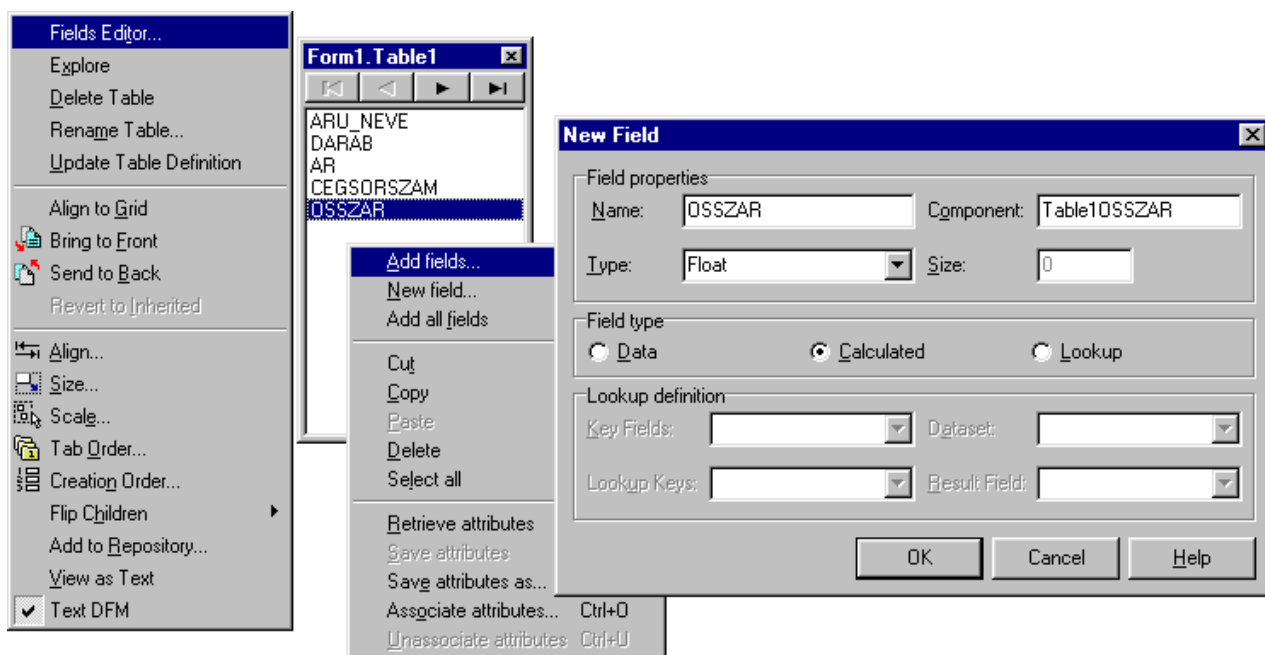
```
CEGSORSZAM = 1
```

A szűrőváltás pedig a **RadioGroup** vezérlőelemen történő kattintás esetén történik:

```
procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
    Table1.Filter:= 'CEGSORSZAM = ' + IntToStr(RadioGroup1.ItemIndex);
    Table1.Filtered:= (RadioGroup1.ItemIndex>0);
end;
```

A fenti példában alkalmazott megoldásnak köszönhetően a **Filtered** tulajdonság értéke csak akkor lesz *false* (így az adattábla össze rekordja megjelenik, szűrés nélkül), ha a **RadioGroup** vezérlőelemen belül a legelső (**ItemIndex** = 0), a „minden cég” feliratú választógombra kattintunk. Az **ItemIndex** tulajdonság összes többi lehetséges értéke bekapcsolja a szűrést.

Ahhoz, hogy az adattáblában definiált mezőkön kívül a **DBGrid** vezérlőelem segítségével megjeleníthessük az ún. kiszámított mezőket is (a példánkban ez az **OSSZAR** mező, amely az adott rekord **DARAB**, illetve **AR** mezőiben található értékek szorzatát jeleníti meg), szerkesztési időben az ún. mezőszerkesztőt (**Fields Editor**) kell használnunk. A mezőszerkesztő megjelenítéséhez kattintsunk a jobb oldali egérgombbal a **Table** vezérlőelemen, és válasszuk ki a megjelenő menüből a „**Fields Editor...**” menüpontot! A kattintást ismételjük meg a **Form1.Table1** ablakban is!



A megjelenő menü „**Add all fields**” menüpontja segítségével az adattábla összes mezőjét lekérhetjük a **Table** vezérlőelemen keresztül. A „**New field**” menüponttal az adattábla mezőitől eltérő elemeket is definiálhatunk. A példánkban használt kiszámított (*Calculated*) **OSSZAR** mező *Float* típusú lesz. (A **Component** mezőben automatikusan megjelenő nevet ugyanúgy átírhatjuk másra, mint a vezérlőelemek **Name** tulajdonsága esetén.) A **Fields Editor** segítségével történő mezőmegadás azt eredményezi, hogy ezek a mezők a **Form** osztály elemei lesznek (*TxxxField*), így a programban közvetlenül elérhetjük őket:

```

type
  TForm1 = class(TForm)
    DataSource1: TDataSource;
    Table1: TTable;
    DBNavigator1: TDBNavigator;
    DBGrid1: TDBGrid;
    Label1: TLabel;
    ...
    Table1ARU_NEVE: TStringField;
    Table1DARAB: TFloatField;
    Table1AR: TFloatField;
    Table1CEGSORSZAM: TSmallintField;
    Table1OSSZAR: TFloatField;
    ...
    procedure RadioGroup1Click(Sender: TObject);
    ...
  private { Private declarations }
  public { Public declarations }
  end;

```

Például, az *OSSZAR* kiszámított mező értékeit a következő módon határozhatjuk meg (az értékek átszámítását a **Table** vezérlőelem külön eseménykezelője végzi, amelynek nevét az **OnCalcFields** mezőben kell megadni az objektum-szerkesztő **Events** lapján):

```

procedure TForm1.Table1CalcFields(DataSet: TDataSet);
begin
  Table1OSSZAR.AsFloat := Table1DARAB.AsInteger * Table1AR.AsFloat;
end;

```

ARU_NEVE	DARAB	AR	CEGSORSZAM	OSSZAR
agy	3	12000	2	36000
lampa	4	2000	2	8000
allolampa	2	3500	2	7000
csillar	3	4000	2	12000

A két *Keresés* gomb segítségével az *ARU_NEVE* (*Termék*) és az *AR* (*Ár*) mezőértékek szerint pozícionálhatunk az adattáblában:

```

procedure TForm1.NevSzerintClick(Sender: TObject);
begin
  if Edit1.Text <> '' then Table1.Locate('ARU_NEVE', Edit1.Text, [loPartialKey]);
end;

procedure TForm1.ArSzerintClick(Sender: TObject);
begin
  if Edit2.Text <> '' then Table1.Locate('AR', Edit2.Text, [loCaseInsensitive]);
end;

```

A **Table** vezérlőelem **Locate** metódusának első két paramétere a pozícionáláshoz használt (keresendő) mező neve és értéke, a harmadik paraméter értéke pedig *loCaseInsensitive*, vagy *loPartialKey* lehet. Az elsővel betűnagyságra nem érzékeny keresést végzünk, a második használata esetén a **Locate** metódus második paraméterében a mező teljes értéke helyett elegendő csak egy részt megadni (például *lam* karaktersorozatot *lampa* helyett).

A *Legdrágább* és a *Legolcsóbb* nyomógombok segítségével az adattáblában megkeressük a legnagyobb, illetve a legkisebb árat, a **Table** vezérlőelem **Fields** tulajdonságát használva (*Fields[index]*, ahol az *index* a mező rekordon belüli sorsszámát határozza meg, 0-tól kezdve). A maximális ár keresését következőképpen valósíthatjuk meg:

```
procedure TForm1.MaxArClick(Sender: TObject);
var max :real;
begin
  max:=0;
  Table1.First;           // pozicionálás az adattábla első rekordjára
  while not Table1.EOF do // amíg nem értünk el a tábla végét
  begin
    if max < Table1.Fields[2].AsFloat then max := Table1.Fields[2].AsFloat;
    Table1.Next;           // pozicionálás az adattábla következő rekordjára
  end;
  MessageDlg('Legdrágább termék ára: ' + Format('%f',[max]), mtInformation, [mbOK], 0);
end;
```

A fenti megoldás hátránya, hogy a lépkedések következtében az utasítások végrehajtása után a táblakurzor nem arra a rekordra mutat, amilyen előzőleg állt, hanem az utolsó bejegyzésre:



A legalacsonyabb ár keresését már úgy oldjuk meg, hogy a keresés után a kurzor visszakерüljön az eredeti helyére. Ehhez a keresés előtti pozíciót könyvjelzővel (*TBookmark*) jelöljük meg (*GetBookmark* metódus), amelyre visszatérünk a keresés után (*GotoBookmark* metódus, utána pedig *FreeBookmark* metódus):

```
procedure TForm1.Button1Click(Sender: TObject);
var min: real; bookmark: TBookmark;
begin
  bookmark:= Table1.GetBookmark; // könyvjelző létrehozása
  Table1.DisableControls;        // a táblával összekapcsolt adatmegjelenítő
                                // vezérlőelemek frissítésének tiltása

  Table1.First;
  min:=Table1.Fields[2].AsFloat;
  Table1.Next;

  while not Table1.EOF do
  begin
    if min > Table1.Fields[2].AsFloat then min:= Table1.Fields[2].AsFloat;
    Table1.Next;
  end;

  if Table1.BookmarkValid(bookmark) then // ha érvényes a könyvjelző (azaz a mutató)
  begin
    Table1.GotoBookmark(bookmark); // könyvjelzőre való pozicionálás
    Table1.FreeBookmark(bookmark); // könyvjelző felszabadítása
  end;

  Table1.EnableControls; // a táblával összekapcsolt adatmegjelenítő vezérlőelemek
                        // frissítésének engedélyezése

  MessageDlg('Legolcsóbb termék ára: ' + Format('%f',[min]), mtInformation, [mbOK], 0);
end;
```


A fenti példában a keresési műveletek gyorsítására, a keresés előtt megtiltjuk az adatmegjelenítő vezérlőelemek frissítését a ***DisableControls*** metódus hívásával (így elkerülhetjük a ***DBGrid*** keresés közbeni „ugrálását”). A keresés után újra engedélyezni kell az adatok megjelenítési frissíthetőségét, amire a ***Table*** vezérlőelem ***EnableControls*** metódusa szolgál.



Készítsünk alkalmazást, amelyben SQL-lekérdezés segítségével megkereshetjük az összes olyan bejegyzést egy adattáblában, amelynek egyik mezőértéke megegyezik egy másik táblázatból kiválasztott értékkel (például az ügyfelek adatait, az ügyfelet azonosító szám szerint)! (*Query_Params*)

Ha két táblával dolgozunk, az egyikből a másikban is megtalálható értékekkel rendelkező rekordokat kell kiválasztanunk, általában a következőhöz hasonló SQL-utasítással oldhatjuk meg a feladatot:

```
SELECT elsoTabla.* FROM orders elsoTabla, customer masodikTabla
WHERE elsoTabla.CustNo = masodikTabla.CustNo
```

A fenti példában az első *orders* táblából - amelynek az SQL-utasításban *elsoTabla* nevet adtuk - az összes olyan rekordot kiválasztottuk (*elsoTabla.**), amelynél *CustNo* mező értéke megegyezik a *customer* (*masodikTabla*) táblában található rekordok *CustNo* mező értékével.

A példa teszteléséhez helyezzünk el a formon egy **Query**, egy **DataSource** és egy **DBGrid** vezérlőelemet, és végezzük el a következő beállításokat:

Vezérlőelem neve	Tulajdonság	Érték
<i>Query1</i>	DatabaseName	<i>DBDEMOS</i>
	SQL	lásd a fenti példában
	Active	<i>True</i>
<i>DataSource1</i>	DataSet	<i>Query1</i>
<i>DBGrid1</i>	DataSource	<i>DataSource1</i>

A fenti SQL-utasítás azonban a két értékhalmoz teljes metszetét adja vissza (azaz az összes „párosítható” *CustNo* értékkel rendelkező bejegyzést). Ha csupán egy bizonyos értékre kell szűkíteni a lekérdezést, akkor ezt legegyszerűbben a **WHERE** feltételrész megadásával tehetjük meg:

```
SELECT elsoTabla.*
FROM orders elsoTabla, customer masodikTabla
WHERE (elsoTabla.CustNo =
masodikTabla.CustNo)
AND (CustNo = 1221)
```

OrderNo	CustNo	SaleDate	ShipDate	EmpNo
1023	1221	01/07/88	02/07/88	5
1076	1221	16/12/94	26/04/89	9
1123	1221	24/08/93	24/08/93	121
1169	1221	06/07/94	06/07/94	12
1176	1221	26/07/94	26/07/94	52
1269	1221	16/12/94	16/12/94	28

Ha a konkrét értéket a felhasználó határozhatja meg – beírással egy szövegbeviteli mezőbe, kiválasztással egy lista-, kombinált ablakból stb. – az SQL-utasítás összeállítását és végrehajtását futási időben kell elvégeznünk:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with Query2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('SELECT elsoTabla.* FROM orders elsoTabla, customer masodikTabla ');
      SQL.Add('WHERE (elsoTabla.CustNo = masodikTabla.CustNo) AND (elsoTabla.CustNo = ');
      SQL.Add(ComboBox1.Text + ')');
      Open;
    end;
end;
```

A fenti példában szereplő kombinált listát (*ComboBox1*) a form betöltésekor kell kitöltenünk a megfelelő értékekkel, a *customer* táblázatból (azért használjuk a **ComboBox** vezérlőt, nem pedig a **DBComboBox**-ot, mivel nem kívánjuk megváltoztatni az adattábla értékeit kombinált lista vezérlőelemen keresztül). A kombinált listán kívül helyezzünk el a formon egy **Table** és egy újabb **DataSource** vezérlőelemet, majd végezzük el a következő beállításokat!

Vezérlőelem neve	Tulajdonság	Érték
Table1	DatabaseName	DBDEMOS
	TableName	customer.db
	Active	true
DataSource2	DataSet	Table1

A kombinált lista feltöltése adatokkal:

```

procedure TForm1.FormCreate(Sender: TObject);
begin
  while not Table1.Eof do                                // amíg nincs vége a táblának
  begin
    ComboBox1.Items.Add(Table1.FieldName('CustNo').AsString);
    Table1.Next;                                           // léptetés a tábla következő bejegyzésére
  end;
  // a kombinált lista beviteli mezőjében megjelenő érték megadása:
  ComboBox1.Text:=ComboBox1.Items[0];
end;

```

Ha a *customer* táblát teljes egészében megjelenítjük a formon egy **DBGrid** vezérlőelemben (a *DBGrid2*-öt a már létező *Table1* és *DataSource2* vezérlőelemek alkotta adattovábbító láncához kapcsolhatjuk a **DataSource** tulajdonságán keresztül), a felhasználó kattintással is jelezheti, mely *CustNo* értékkel rendelkező rekordokra kíváncsi. Ehhez a működéshez azonban meg kell írunk a *DBGrid2* objektum cellakattintási eseményét (**OnCellClick**) kezelő eljárást.

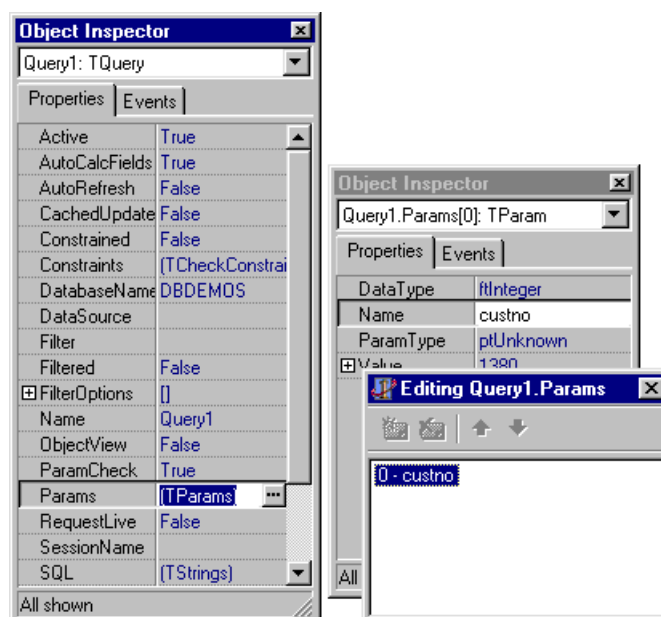
Az eseménykezelőben történő SQL-utasítás összeállítás helyett, oldjuk meg a feladatot az ún. paraméterezett SQL-utasítás segítségével, amelyet a fejlesztés során adhatunk meg a *Query1* vezérlőelem **SQL** tulajdonsága értékeként:

```

SELECT * FROM orders WHERE
    custno=:custno

```

A példában látható SQL-utasításban a *:custno* (*:paraméternév*) a **Query** vezérlőelem **Params** tulajdonságán keresztül elérhető paramétert jelöl. A fejlesztés során megadott, SQL-utasításban szereplő paraméterek automatikusan bekerülnek a **Params** listába.



Az elmondottak alapján a *DBGrid2* objektum cellakattintási eseménykezelőjében nem kell átírnunk a *Query1 SQL* tulajdonságának értékét, hanem elég beállítani a benne szereplő paraméter aktuális értékét az adatrácsban kijelölt cellában szereplő értékre (*Query1.Params* tömb - *DBGrid1.SelectedField.DisplayText*). (Természetesen ügyelve a megfelelő típuskonverzióra.)

```
procedure TForm1.DBGrid1CellClick(Column: TColumn);
begin
// ha a kijelölt cella (SelectedField) mezőneve (FieldName) a CustNo:
if DBGrid1.SelectedField.FieldName = 'CustNo' then
  with Query1 do
    begin
      Close;
      // állítsuk be a Query1 vezérlőelem első paraméterének (Params[0]) értékét
      // a DBGrid1 vezérlőelem kiválasztott cellájában (SelectedField) látható
      // szövegre (DisplayText), megfelelő típuskonverziók alkalmazásával:
      Params[0].AsInteger:=StrToInt(DBGrid1.SelectedField.DisplayText);
      Open;
    end;
end;
```

Az alkalmazásunkban mind a **ComboBox**, mind pedig a **DBGrid** segítségével kiválasztható mezőértékeket alkalmazó megoldást valósítottunk meg. A szemléletesség kedvéért az egyik **DBGrid** vezérlőelem háttérszínét (**Color**) szürkére állítottuk, a másik kettőben azonban csak a *CustNo* oszlopot színeztük ki.

A színezéshez először a **DBGrid** vezérlő **Columns** tulajdonság-szerkesztőjének „Add All Fields” gomjára kell kattintanunk, majd a megjelenő listában a *CustNo* mezőre. Ezt követően az objektum-szerkesztő ablakában átállíthatjuk a mezők megjelenítésére vonatkozó tulajdonságokat (**Color** – háttérszín, **Font** – betűk típusa, mérete, színe stb.)

A kész alkalmazás futás közbeni ablaka:

CustNo --> CustNo reláció

Kattintson az alábbi táblázat CustNo oszlopa valamelyik bejegyzésén, vagy válasszon értéket a CustNo listából (kattintás esetén figyelje meg az alsó táblázatban megjelenő értékeket)!

customer.db

CustNo	Company	Adj
2984	Professional Divers, Ltd.	473
3041	Divers of Blue-green	634
3042	Gold Coast Supply	223
3051	San Pablo Dive Center	170
3052	Underwater Sports Co.	351
3053	American SCUBA Supply	173

CustNo: 1231 Rendeléslista

OrderNo	CustNo	SaleDate	ShipDate
1060	1231	28/02/89	01/03/89
1073	1231	15/04/89	16/04/89
1102	1231	06/06/92	06/06/92
1160	1231	01/06/94	01/06/94
1173	1231	16/07/94	16/07/94

orders.db

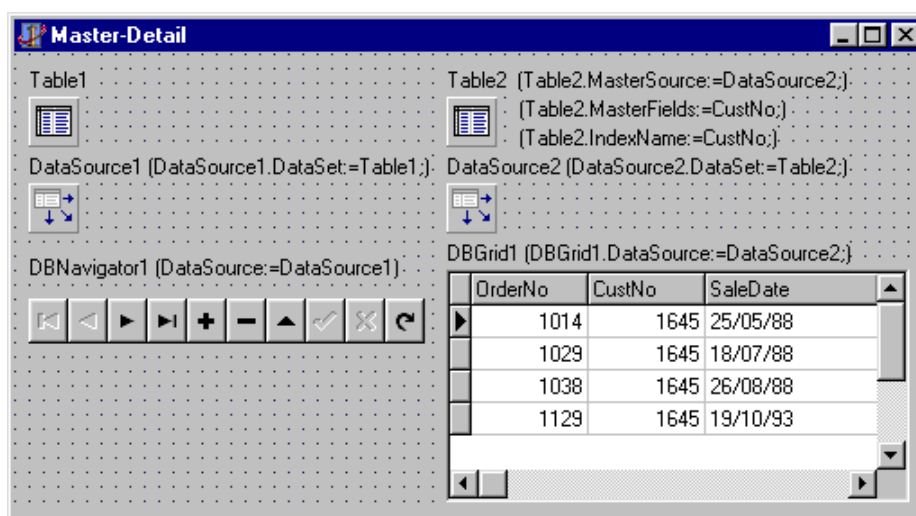
OrderNo	CustNo	SaleDate	ShipDate	EmpNo	ShipToContact	ShipToAddr1	ShipToAddr2
1050	3052	24/12/88	25/12/88	94			
1150	3052	25/03/94	25/03/94	44			
1250	3052	24/11/94	24/11/94	61			
1350	3052	02/02/95	02/02/95	71			



Oldjuk meg az előző pontban kiírt feladatot az alaptábla (*master*) – részletező (*detail*) tábla kapcsolat kialakításával! (*Master*)

Azokban az esetekben, amikor egy adattáblában és egy másik táblában szereplő rekordok között információs kapcsolat áll fenn (például az iskolai tantárgyak felsorolása és a tanárok listája között, ahol például az egyes tanárok által oktatott tárgyak alkotják a kapcsolatot) a fentiekől különböző megoldást is választhatunk a táblák összekapcsolására. Az egyik tábla konkrét mezőértékére való lekérdezést nem csak a **Query** vezérlőelem segítségével, hanem két **Table** vezérlőelem közötti alap-részletező kapcsolat kialakításával is megoldhatjuk, amennyiben léteznek indextáblák a kapcsolódó adatmezőkhöz.


Ha a fenti megoldásokban is használt adattáblánál (*DBDEMOS*) maradunk, a következő ábrából leolvasható vezérlőelemekre, illetve beállításokra lesz szükség:

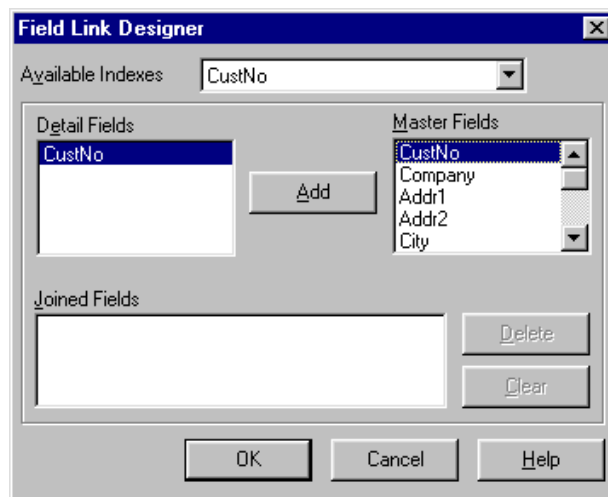


Vezérlő	Leírás
Table1	az alaptáblát elérő készletező vezérlőelem,
DataSource1	a Table1-hez kapcsolódó adatközetítő vezérlőelem,
DBNavigator1	az alaptáblában való léptetést megvalósító vezérlőelem,
Table2	a részletező táblát elérő készletező vezérlőelem,
DataSource2	a Table2-höz kapcsolódó adatközetítő vezérlőelem,
DBGrid1	a részletező táblában megtalált, az alaptábla aktuális rekordjának megfelelő bejegyzéseket megjelenítő vezérlőelem.

Mind a két **Table** vezérlő **DatabaseName** tulajdonságának értékét állítsuk be *DBDEMOS*-ra, a **Table1** **TableName** tulajdonságának értéke legyen *customer.db*, a **Table2**-e pedig *orders.db*! A táblák **Active** tulajdonságának értékét kapcsoljuk át *true*-ra!

Meg kell jegyeznünk, hogy a *customer.db* adattáblában a *Custno* kulcsmezőként (elsődleges indexként) szerepel, míg az *orders.db* adattáblához a *CustNo* mint másodlagos index definiált. Ez a két indexelés teszi lehetővé a alap-részletező tábla kapcsolat kialakítását.

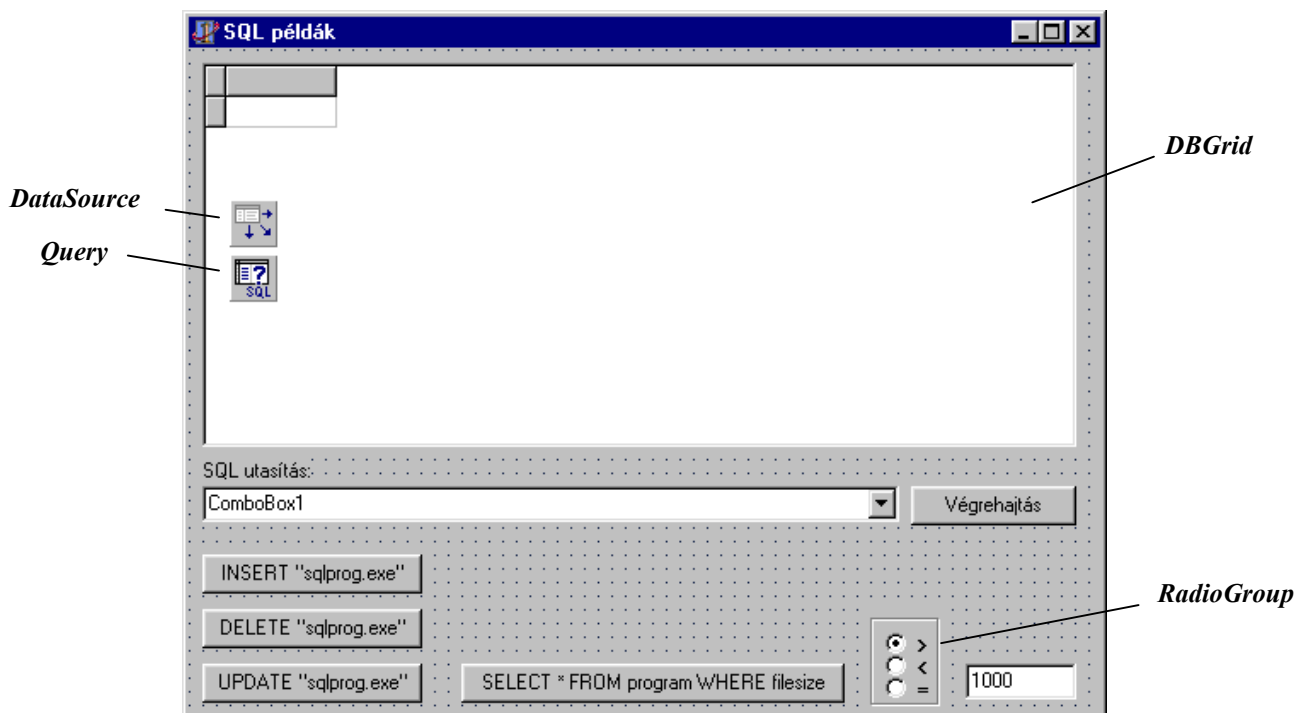
A részletező tábla **MasterFields** tulajdonságának beállításához először az értékmezőben látható  gombon kell kattintanunk. A megjelenő „Field Link Designer” párbeszédablakban először az „Available Indexes” mezőből, majd pedig a „Detail Fields” és a „Master Fields” mezőkből is kiválasztjuk a *CustNo* mezőnevet, és megnyomjuk az **Add** gombot:



Ebben az esetben a **DBGrid** vezérlőelemben megjelenő rekordok csak az alaptáblában (*Table1*) aktuálisan elért rekordnak megfelelő bejegyzéseket jelenítik meg a részletező táblából. Pontosabban azokat a rekordokat, amelyek *CustNo* mezőjének értéke megegyezik az alaptábla azonos nevű mezőjében tárolt aktuális értékkel. Példánkban ez annyit jelent, hogy egyszerre csak azon megrendeléseket listázzuk ki, amelyek egy és ugyanazon ügyfelünkre vonatkoznak (itt a *CustNo* mező az ügyfelünket azonosító számot tartalmazza).



A különböző SQL lekérdezések működésének bemutatására alakítsuk ki alkalmazásunk űrlapját a következő ábrán látható módon!



A **ComboBox** vezérlőelem segítségével megjeleníthető SQL-utasításokat a form betöltésékor egy fájlból olvassuk be, a vezérlőelem **Items** tulajdonságába:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    ComboBox1.Items.LoadFromFile('queries.sql');
    // a ComboBox beviteli mezőjében látható szöveg
    ComboBox1.Text := 'Select * From Program';
end;
```

Az utasításokat tartalmazó *queries.sql* egyszerű szöveges fájl, amelyet például a *Jegyzettömb* segítségével is létrehozhatunk. A **ComboBox** vezérlőelemből kiválasztott SQL-utasítást a **Query** vezérlőelem **SQL** tulajdonsága értékeként állíthatjuk be, illetve végre is hajthatjuk a *Végrehajtás* gombra kattintva:

```
procedure TForm1.btnVegrehajtasClick(Sender: TObject);
begin
    with Query1 do
    begin
        Close;
        SQL.Clear;
        SQL.Add(ComboBox1.Text);
        Open;
    end;
    // ha az utasítást kézzel írtuk be, és még nem szerepel a listában,
    // hozzáadjuk az utasítást a listához:
    with ComboBox1 do
    if Pos(UpperCase(Text), UpperCase(Items.Text))=0
    then Items.Add(Text);
end;
```

A kód működéséhez természetesen előbb fel kell építenünk az adatbázist elérő vezérlőelem-láncot, a következő beállítások megadásával:

Vezérlőelem neve	Tulajdonság	Érték
<i>Query1</i>	DatabaseName SQL	. \Data SELECT * FROM PROGRAM
<i>DataSource1</i>	DataSet	<i>Query1</i>
<i>DBGrid1</i>	DataSource	<i>DataSource1</i>

A **Query** vezérlőelem **DatabaseName** tulajdonságának értékeként a példaprogramot is tartalmazó könyvtár *Data* alkönyvtárának nevét adtuk meg, ahová bemásoltuk a példánkban használt *program.dbf* adattáblát.

A „SELECT * FROM program WHERE filesize” feliratú gomb működtetésével a nyomógomb után látható **RadioGroup** és **Edit** vezérlőelemekből nyert értékekkel összeállított SQL-utasítást hajtunk végre:

```
procedure TForm1.btnSelectClick(Sender: TObject);
begin
  with Query1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add(btnSelect.Caption + ' ' + RadioGroup1.Items[RadioGroup1.ItemIndex]);
      SQL.Add(' :filesize ORDER BY filename');
      ParamByName('filesize').AsInteger:=StrToInt(edtFileSize.Text);
      Open;
    end;
end;
```

A fentiekben látható kódban az SQL-utasítás utolsó elemét paraméterezéssel (:filesize) adjuk meg. Mivel a paramétert tartalmazó SQL-utasítás, illetve a paraméter kiértékelése egy helyen történik, a programokban hasonló helyzetben jobb az értéket közvetlenül hozzáadni az SQL sztringhez. Az SQL utasításban szereplő paraméterek elérését indexeléssel (0-tól kezdve) is megoldhatjuk, használva a **Query** vezérlőelem **Params** tömbjét (*Params[index]*). Arra is van lehetőség, hogy a példában is látható módon, a **ParamByName** tömbön keresztül adjuk meg a paramétereket (*ParamByName(paraméter_név)*).

Az „INSERT ...”, „DELETE ...” és „UPDATE ...” gombok segítségével új rekordot adhatunk hozzá az adattáblához, törölhetünk, illetve módosíthatunk rekordot. Mindhárom gomb esetén egy és ugyanazon bejegyzésről van szó, amelyet következő módon adhatunk a *program* táblához:

```
procedure TForm1.btnInsertClick(Sender: TObject);
begin
  with Query1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('INSERT INTO program (folder, filename, filesize, filedate, notes)'+
        'VALUES ("delph2", "sqlprog.exe", 635948, Cast("03.05.2000." AS
        Date), "Hello")');
      ExecSQL;
    end;
end;
```

Az **INSERT INTO** utasításban az adattábla megnevezése után, zárójelben fel kell sorolni azokat a mezőket, amelyeknek értéket adunk, utána pedig a **VALUES** kulcsszót követően a konkrét értékeket is meg kell adnunk.

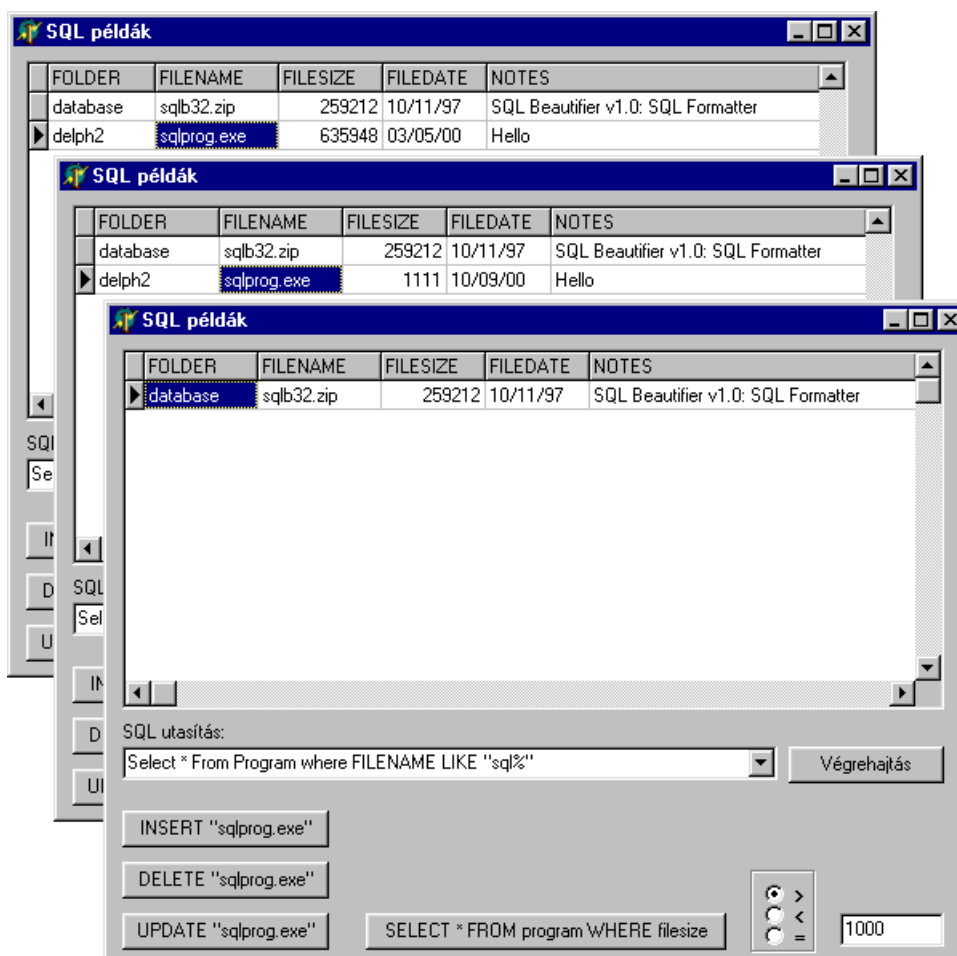
Már létező rekordok módosítását az **UPDATE** SQL-parancs segítségével végezhetjük el. A következő példában *1111*-re, illetve *10.09.2000*-re (2000. szeptember 8.) változtatjuk minden olyan bejegyzés *filesize* és *filedate* mezőjének értékét, amelynél a *filename* mezőérték az „*sqlprog.exe*”:

```
procedure TForm1.btnUpdateClick(Sender: TObject);
begin
  with Query1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('UPDATE program SET filesize=1111, filedate = Cast("10.09.2000." AS Date) ');
      SQL.Add('WHERE filename = "sqlprog.exe"');
      ExecSQL;
    end;
end;
```

A **DELETE** SQL parancsal bejegyzéseket törölhetünk az adattáblából:

```
procedure TForm1.btnDeleteClick(Sender: TObject);
begin
  with Query1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('DELETE FROM program WHERE filename="sqlprog.exe"');
      ExecSQL;
    end;
end;
```

A három utolsó eljárásban a **Query** vezérlőelem **Open** metódusa helyett az **ExecSQL** metódust hívtuk, mivel a **SELECT** utasítással szemben az **INSERT**, az **UPDATE** és a **DELETE** SQL-utasítások nem ad vissza semmilyen értéket.

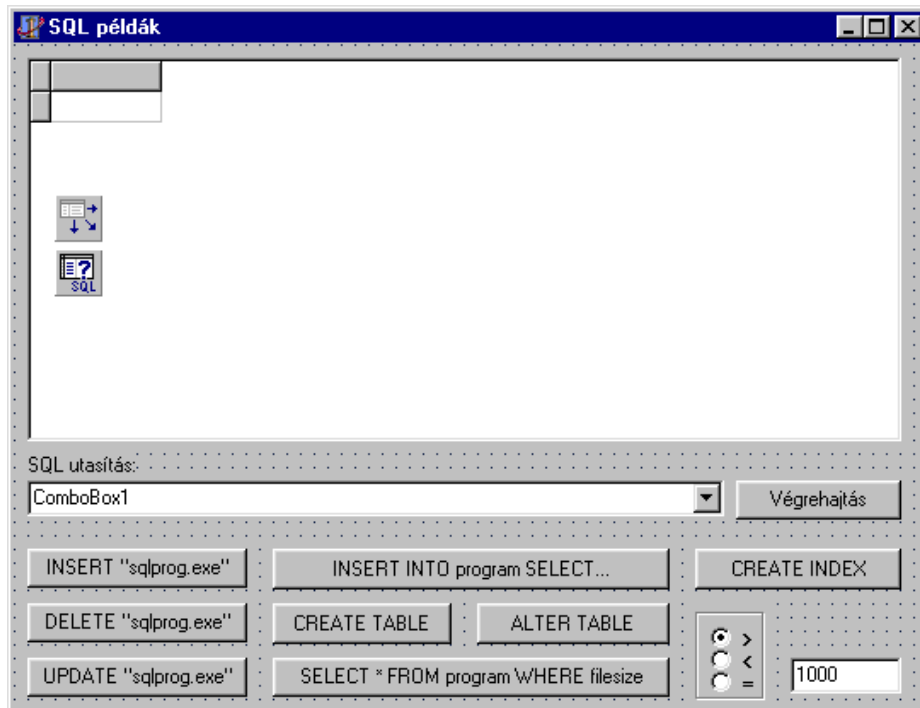


Az utasítások végrehajtásának eredménye úgy szemlélhető a legjobban, ha az „SQL utasítás:” listából a fenti ábrán is látható utasítást választjuk ki:

```
Select * From Program where FILENAME LIKE "sql%"
```

leszűkítve ezzel a megjelenő rekordok csoportját az „sql” karakterekkel kezdődő fájlnevekre, és ezek után rákattintunk a *Végrehajtás* gombra.

Az SQL-lekérdezések további lehetőségeinek bemutatására szükségünk van néhány újabb nyomógombra:



Az *“INSERT INTO program SELECT...”* feliratú nyomógombot a „Local SQL” azon utasításának bemutatására használjuk, melynek segítségével egy adattáblából a **SELECT** utasítással lekérdezett rekordokat egy másik táblába másolhatjuk:

```
INSERT INTO tábla1 SELECT * FROM tábla2 [WHERE tábla2.mező1 ...];
```

A nyomógomb megnyomásakor az azonos struktúrával rendelkező *program1.dbf* táblából a *program* táblába másoljuk az összes olyan rekordot, amely *filename* mezőjének értéke tartalmazza az „sql” karaktersorozatot:

```
procedure TForm1.btnInsertSelectClick(Sender: TObject);
begin
  with Query1 do
  begin
    Close;
    SQL.Clear;
    SQL.Add('Insert Into program '+
            'SELECT * FROM program1 WHERE program1.filename LIKE "sql%.exe"');
    ExecSQL;
  end;
end;
```

A „*Local SQL*” fentiekben bemutatott, DML (*Data Manipulation Language*) adatkezelési utasításain kívül használhatjuk a DDL (*Data Definition Language*) adatdefiníciós utasításokat is. A „*Local SQL*” a következő adatdefiníciós utasításokat támogatja:

Utasítás	Leírás
CREATE TABLE	Adattábla létrehozása
ALTER TABLE	Létező adattábla átstrukturálása
DROP TABLE	Adattábla törlése
CREATE INDEX	Index létrehozása
DROP INDEX	Index törlése

A **CREATE TABLE** utasítással egy új (üres) táblát hozhatunk létre, zárójelben felsorolva a tábla mezőneveit, illetve a mezők típusát:

```
CREATE TABLE tábla_név (mező1 típus1, [mező2 típus2 ...]);
```

A példánkban a „**CREATE TABLE**” nyomógombon kattintva létrehozunk egy újabb *dBASE* (*empl.dbf*) adattáblát:

```
procedure TForm1.btnCreateTableClick(Sender: TObject);
begin
  with Query1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('CREATE TABLE "empl.dbf" '+
        '(LAST_NAME CHAR(20), FIRST_NAME CHAR(15), SALARY NUMERIC(10,2),
        DEPT_NO SMALLINT)');
      ExecSQL;
    end;
end;
```

Ha egy *Paradox* (.DB) adattáblát hozunk létre, a *dBASE* táblákkal ellentétben megadhatunk elsődleges kulcsokat is. A következő példában a *LAST_NAME* és *FIRST_NAME* mezőt elsődleges kulcsként hozzuk létre (**PRIMARY KEY**):

```
CREATE TABLE "employee.db" (LAST_NAME CHAR(20), FIRST_NAME CHAR(15),
  SALARY NUMERIC(10,2), DEPT_NO SMALLINT, PRIMARY KEY(LAST_NAME, FIRST_NAME))
```

A következő táblázat az **CREATE TABLE** SQL utasításban megadható mezőtípusokat tartalmazza, illetve a *Paradox*, illetve a *dBASE* adattáblák megfelelő értéktípusait (a konverziót a BDE végzi). A táblában az *x* paraméter a pontosságot megadó értéket (alapértelmezés szerinti érték függ a használt adatbázis-meghajtótól), az *y* – tizedes jegyek számát (alapértelmezés szerinti érték 0), az *n* – a mező bájtban kifejezett méretét (alapértelmezés szerinti érték 0) jelenti. Az 1-től 5-ig terjedő értékek pedig a BLOB (*Binary large object*) altípusokat jelölik (bináris, feljegyzés, formázott feljegyzés, grafika és OLE; alapértelmezés szerinti érték 1):

SQL Syntax	BDE Logical	Paradox	dBASE
<i>SMALLINT</i>	<i>fldINT16</i>	<i>Short</i>	<i>Number (6,10)</i>
<i>INTEGER</i>	<i>fldINT32</i>	<i>Long Integer</i>	<i>Number (20,4)</i>
<i>DECIMAL(x,y)</i>	<i>fldBCD</i>	<i>BCD</i>	<i>N/A</i>
<i>NUMERIC(x,y)</i>	<i>fldFLOAT</i>	<i>Number</i>	<i>Number (x,y)</i>
<i>FLOAT(x,y)</i>	<i>fldFLOAT</i>	<i>Number</i>	<i>Float (x,y)</i>
<i>CHARACTER(n)</i>	<i>fldZSTRING</i>	<i>Alpha</i>	<i>Character</i>
<i>VARCHAR(n)</i>	<i>fldZSTRING</i>	<i>Alpha</i>	<i>Character</i>
<i>DATE</i>	<i>fldDATE</i>	<i>Date</i>	<i>Date</i>
<i>BOOLEAN</i>	<i>fldBOOL</i>	<i>Logical</i>	<i>Logical</i>
<i>BLOB(n,1)</i>	<i>fldstMEMO</i>	<i>Memo</i>	<i>Memo</i>
<i>BLOB(n,2)</i>	<i>fldstBINARY</i>	<i>Binary</i>	<i>Binary</i>

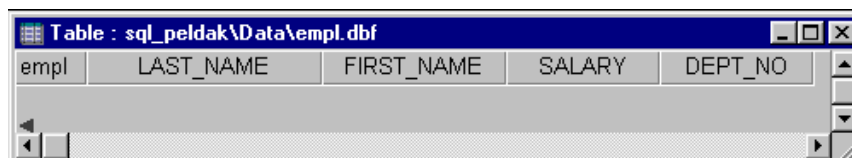
SQL Syntax	BDE Logical	Paradox	dBASE
BLOB(n,3)	fldstFMTMEMO	Formatted memo	N/A
BLOB(n,4)	fldstOLEOBJ	OLE	OLE
BLOB(n,5)	fldstGRAPHIC	Graphic	N/A
TIME	fldTIME	Time	N/A
TIMESTAMP	fldTIMESTAMP	Timestamp	N/A
MONEY	fldFLOAT, fldstMONEY	Money	Number (20,4)
AUTOINC	fldINT32, fldstAUTOINC	Autoincrement	N/A
BYTES(n)	fldBYTES(n)	Bytes	N/A

Az **ALTER TABLE** utasítás segítségével egy létező adattáblát strukturálhatunk át: az **ADD** változattal új oszlopokat (mezőket) adhatunk hozzá a táblához, a **DROP** változattal pedig létező oszlopokat törölhetünk a táblából.

A példánkban az **ALTER TABLE** nyomógombon kattintva elvégezhetjük az *empl.dbf* adattábla átstrukturálását úgy, hogy egyrészt töröljük a *LAST_NAME* és a *FIRST_NAME* mezőket, másrészt pedig hozzáadunk a táblához egy szöveges *FULL_NAME* mezőt:

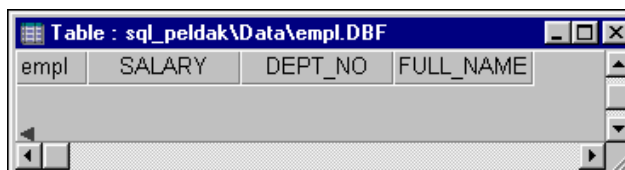
```
procedure TForm1.btnAlterTableClick(Sender: TObject);
begin
  with Query1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('ALTER TABLE "empl.dbf" DROP LAST_NAME, ');
      SQL.Add('DROP FIRST_NAME, ADD FULL_NAME CHAR[30] ');
      ExecSQL;
    end;
end;
```

Az eredeti adatstruktúra:



empl	LAST_NAME	FIRST_NAME	SALARY	DEPT_NO
------	-----------	------------	--------	---------

Az átstrukturálás után:



empl	SALARY	DEPT_NO	FULL_NAME
------	--------	---------	-----------

Létező adattáblák törlésére a **DROP TABLE** utasítás szolgál:

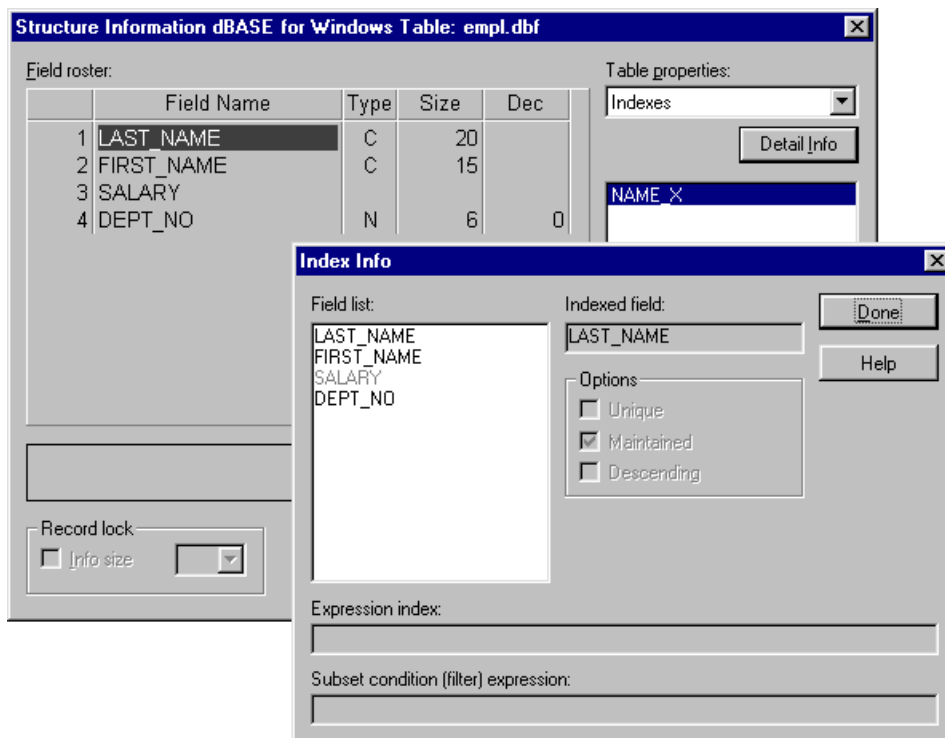
```
DROP TABLE "empl.dbf"
```

A **CREATE INDEX** utasítás segítségével indexeket hozhatunk létre egy adattáblához:

```
CREATE INDEX indexnév ON táblanévé(mező1 [, mező2 ...])
```

Példánkban a *CREATE INDEX* feliratú nyomógombon kattintva az *empl.dbf* (dBASE) táblában egy *NAME_X* indexet hozunk létre:

```
procedure TForm1.btnCreateIndexClick(Sender: TObject);
begin
  with Query1 do
    begin
      Close;
      SQL.Clear;
      SQL.Add('CREATE INDEX NAME_X ON "empl.dbf" (LAST_NAME)');
      ExecSQL;
    end;
end;
```



Paradox táblák esetén a *CREATE INDEX* utasítással csak akkor hozhatunk létre elsődleges indexet, ha a tábla létrehozásakor (*CREATE TABLE*) az adott mezőt elsődleges kulcsként (*PRIMARY KEY*) deklaráltunk. A többi mező esetén a *CREATE INDEX* utasítás eredménye másodlagos index lesz.

Az indexek törléséhez a *DROP INDEX* utasításnak a „Local SQL” által támogatott változatát kell alkalmazni:

```
DROP INDEX táblanév.indexnév
```

vagy

```
DROP INDEX táblanév.PRIMARY
```

A *PRIMARY* kulcsszót a *Paradox* adattálok elsődleges indexeinek törléséhez kell használni. Például:

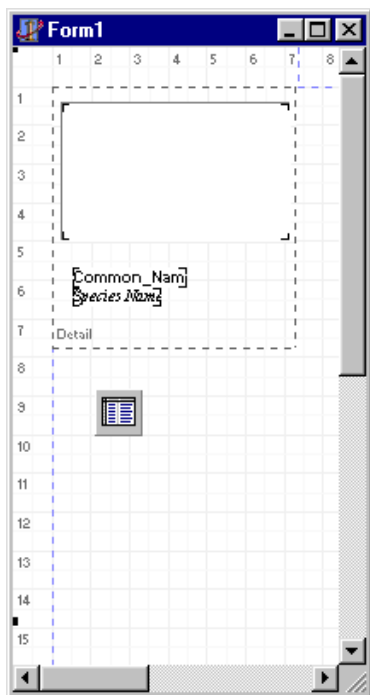
```
DROP INDEX "employee.db".PRIMARY
```

A *dBASE* indexek, illetve a *Paradox* táblák másodlagos indexeinek törlésekor az index nevét kell megadni táblanevet követően:

```
DROP INDEX "empl.dbf".NAME_X
```



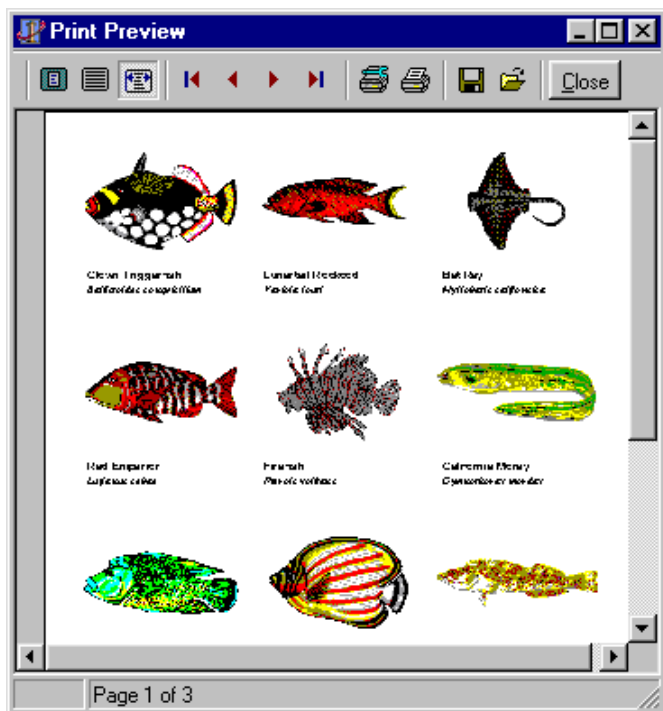

Az első (*beszamolo_1*) jelentésünk legyen egy halképeket tartalmazó ív! Ehhez helyezzünk el a formon egy **QuickRep** vezérlőelemet (**QReport** palettalap), és állítsuk át a **Bands** tulajdonságán keresztül elérhető **HasDetail** tulajdonság értékét **true**-ra! Az így létrejött **Detail** feliratú (**TQRBand**) sáv magasságát nagyobbítsuk meg az egér segítségével, vagy az objektum-szerkesztő ablakában átállítva a sáv **Height** tulajdonságának értékét! Ahhoz, hogy egy sorban egyszerre három halkép jelenjen meg, állítsuk át a **Page** tulajdonságon keresztül elérhető **Columns** tulajdonság értékét 3-ra!



A **QuickRep** vezérlőelemen jobb oldali egérbillentyűvel kattintva, nagyobbítsuk meg a képet a felbukkanó menü „**Zoom In**” menüpontjának többszöri kiválasztásával.

A halkatalógust tartalmazó adattáblát úgy kapcsoljuk hozzá a beszámolóunkhoz, hogy a formon elhelyezett **Table** vezérlőelem **DatabaseName** tulajdonságát beállítjuk **DBDEMOS**, a **TableName** tulajdonságát pedig **biolife.db** értékre. Ezek után megadjuk a **QuickRep** vezérlőelem **DataSet** tulajdonságának értékeként a **Table** vezérlőelem nevét (**Table1**), amelynek **Active** tulajdonságának értékét **true**-ra állítjuk.

A **Detail** sávban helyezzünk el egy **QRDBImage** és két **QRDBText** adatmegjelenítő vezérlőelemet! Mindegyik vezérlőt – a **DataSet** tulajdonságán keresztül - kapcsoljuk hozzá az adatkészletező **Table1** vezérlőelemhez, a **DataField** tulajdonságuk értékét pedig állítsuk be rendre a következő értékekre: **Graphic**, **Common_Name** és **Species Name**! Ahhoz, hogy a keretnél nagyobb, illetve kisebb képek teljes egészében lefedjék a **QRDBImage** vezérlőelem területét, állítsuk át a **Stretch** tulajdonság értékét **true**-ra!



A két feliratmező **Font.Name** tulajdonságát használva állítsuk be a betűtípust **Arial**-ra, illetve **Times New Roman**-ra, a **Font.Style** tulajdonság értékét pedig [**fsBold**]-ra, illetve [**fsItalic**]-ra!

Ezek után kattintsunk a **QuickRep** vezérlőelemen a jobb oldali egérgombbal, és válasszuk ki a megjelenő menüből a **Preview** menüpontot! A „**Print Preview**” ablakban az oldalak közötti léptetést, a beszámoló nyomtatását, fájlba való mentését stb az eszközsorban található gombok segítségével végezhetjük el. Az ablakot a **Close** gombra kattintva zárhatjuk be.

Ha a felbukkanó menüből a „**Report Settings**” menüpontot választjuk ki, akkor a megjelenő párbeszédablakban a beszámoló kialakítására, illetve nyomtatására vonatkozó beállításokat is megadhatunk:

Report Settings

Paper size: A4 210 x 297 mm | Width: 210.0 | Length: 297.0 | Portrait

Margins: Top: 10.00 | Left: 10.00 | Column space: 0.00 | Bottom: 10.00 | Right: 10.00 | Number of columns: 3

Other: Font: Arial | Size: 10 | Units: MM

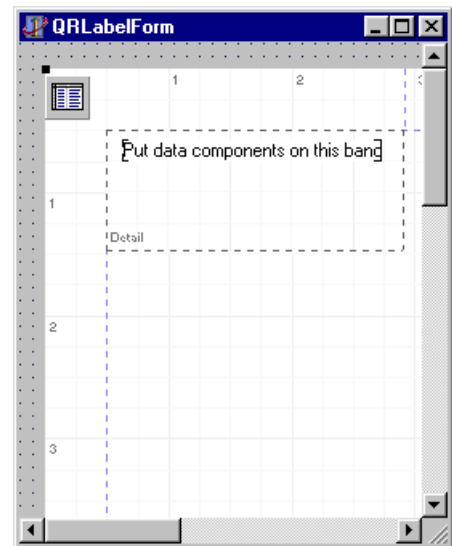
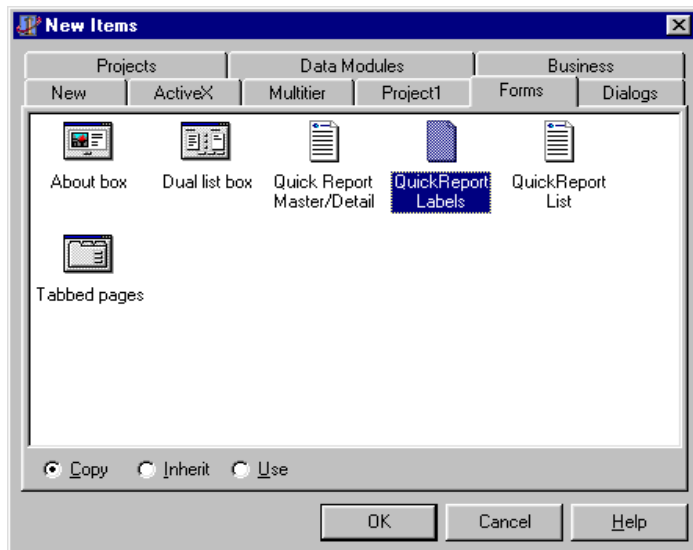
Page frame: ☐ Top ☐ Left ☐ Bottom ☐ Right | Color: ☐ Change | Frame width: 1

Bands:

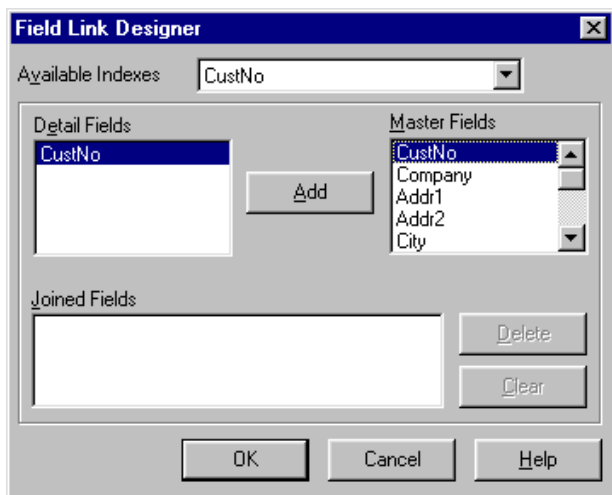
	Length		Length
<input type="checkbox"/> Page header		<input type="checkbox"/> Page footer	
<input type="checkbox"/> Title		<input type="checkbox"/> Summary	
<input type="checkbox"/> Column header		<input type="checkbox"/> Print first page header	
<input checked="" type="checkbox"/> Detail band	67.91	<input type="checkbox"/> Print last page footer	

Buttons: About QuickReport | Preview | Apply | OK | Cancel

A fenti lépés sorozatot felgyorsíthatjuk azzal, hogy a beszámoló készítését nem „üres lappal” kezdjük, hanem varázsló segítségével. Ehhez a **File | New...** választás után a megjelenő **New Items** párbeszédablak **Forms** lapján jelöljük be a „**QuickReport Labels**” elemet, az **OK** megnyomása után a Delphi felajánl egy olyan formot, amely már tartalmazza az összekapcsolt **Table** és egy **Detail** sávval rendelkező **QuickRep** vezérlőelemet:



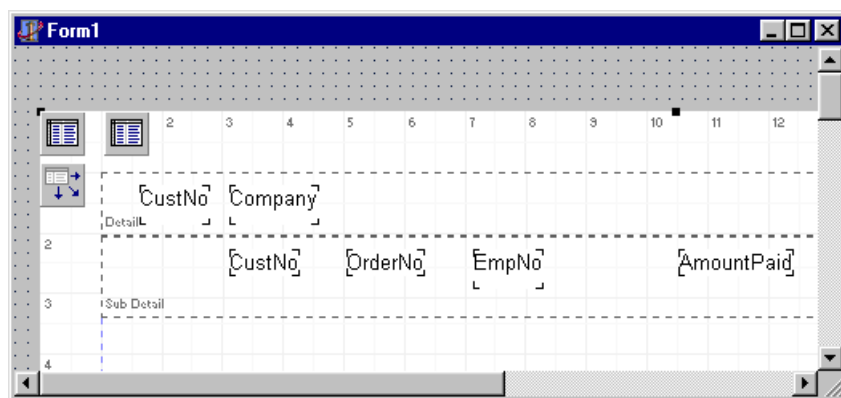
A másik (*beszamolo_2*) beszámolóunkban két adattábla adatait jelenítjük meg egyszerre úgy, hogy az első táblából (*Tabla1*, *DBDEMOS*, *customer.db*) csak a cégneveket, a másikkól pedig (*Tabla2*, *DBDEMOS*, *orders.db*) ezen cégek beszerzési adatait használjuk fel. A két tábla közötti hierarchikus alaptábla–részletező tábla kapcsolat megteremtéséhez szükségünk lesz még egy **DataSource** vezérlőelemre is, melynek a **DataSet** tulajdonság-értékét be kell állítani az alaptábla nevére (*Table1*).



A részletező tábla esetén meg kell adni az alaptáblát reprezentáló adatforrás nevét (*DataSource1*) - a **MasterSource** tulajdonságon keresztül. Ugyancsak megadjuk azt a mezőnevet, amely mind a két táblában megtalálható, így értékei felhasználhatók az adatkapcsolat megteremtéséhez.










Kattintsunk a *Table2* vezérlőelem tulajdonságait megjelenítő objektum-szerkesztőben a **MasterFields** mezőben látható gombon! Ezt követően a „**Field Link Designer**” párbeszédablakban válasszuk ki először az „**Available Indexes**” mezőből, utána pedig a „**Detail Fields**” és a „**Master Fields**” mezőkből a *CustNo* mezőnevet, és kattintsunk az **Add** gombon!

Állítsuk be a **QuickRep** vezérlőelem **Bands** tulajdonságán keresztül a **HasDetail** tulajdonság értékét **true**-ra, a vezérlőelem **DataSet** tulajdonságának értékét pedig az első **Table** vezérlőelem nevére (*Table1*)! Helyezzünk el a **QuickRep**-en egy **QRSubDetail** vezérlőt is, amelynek **DataSet** tulajdonságán keresztül megadhatjuk a másik **Table** vezérlőelem nevét (*Table2*)! A két **Table** vezérlőelem **Active** tulajdonságának értékét a végén ne felejtjük **true**-ra állítani!



Ezek után helyezzünk el a *Detail* és a „*Sub Detail*” feliratú sávokban több **QRDBText** vezérlőelemet, amelyek **DataField** tulajdonságának értékét állítsuk be megfelelő alap-, illetve részletező táblában definiált mezőnevekre! A **QuickRep** vezérlőelem felbukkanó menüjéből a **Preview** menüponttal az alábbihoz hasonló jelentést kapunk:

Print Preview

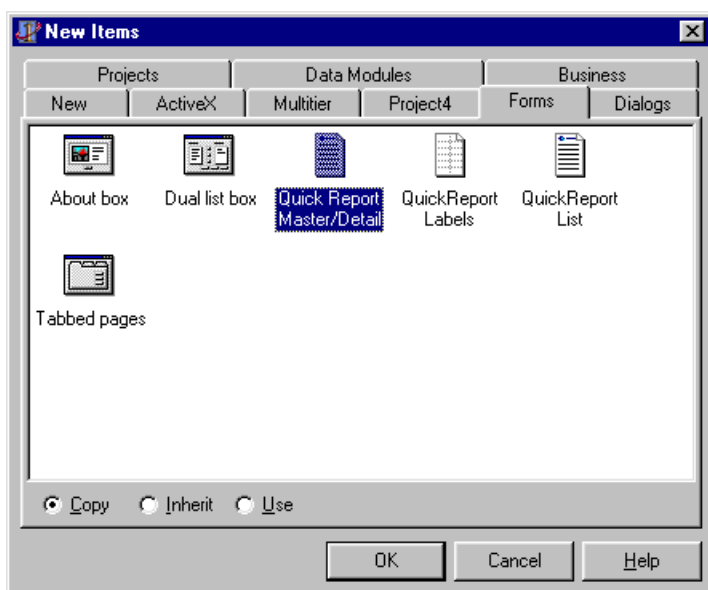
Close

1221	Kauai Dive Shoppe			
1221	1023	5	£4,674.00	
1221	1076	9	£17,781.00	
1221	1123	121	£13,945.00	
1221	1169	12	£9,471.95	
1221	1176	52	£4,178.85	
1221	1269	28	£1,400.00	
1231	Unisco			
1231	1060	94	£15,355.00	
1231	1073	2	£19,414.00	

1354	Cayman Divers World Unlimited			
1354	1104	83	£51,673.15	
1354	1292	136	£7,986.90	
1356	Tom Sawyer Diving Centre			
1356	1005	110	£4,807.00	
1356	1059	109	£2,150.00	
1356	1072	29	£3,596.00	
1356	1080	45	£9,634.00	
1356	1105	28	£31,219.95	
1356	1180	144	£3,640.00	

Page 1 of 5

A fenti ábrán látható kereteket a sávok **Frame** tulajdonságán keresztül elérhető kapcsolók segítségével (**DrawLeft**, **DrawRight** stb.) állíthatjuk be.

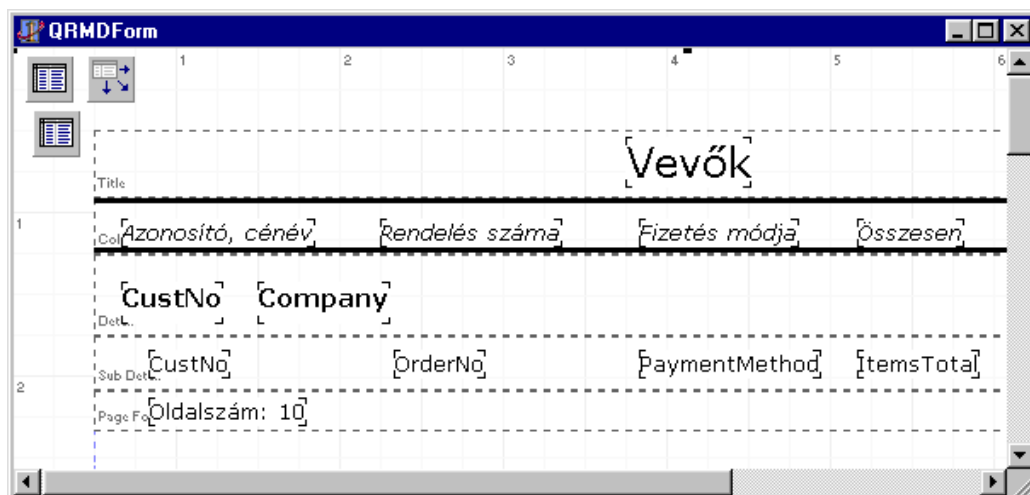


Ugyanúgy, mint az előző példában, az ilyen alaptábla-részletező tábla kapcsolatra épülő beszámolók esetén is használhatunk varázslót. Ehhez a **File|New...** választás után be kell jelölnünk a „**New Items**” párbeszédablak **Forms** lapján a „**Quick Report Master/Detail**” elemet. Az **OK** gomb megnyomása után a Delphi felajánlja a megfelelően kialakított formot:

Ez a kialakítás a már ismertett sávokon kívül három további sávot is tartalmaz, amelyek megjelenítését a **QuickRep** vezérlőelem **Bands** tulajdonságán keresztül elérhető **HasTitle**, **HasColumnHeader** és a **HasPageFooter** tulajdonság **true** értékre való állításával kapcsolhatjuk be.

A **Title** feliratú sáv a beszámoló címsorának megadására használható. Ebben a sávban általában csak egy **QRSysData** vezérlőelem helyezkedik el, ennek **Data** tulajdonságértékét **qrsReportTitle**-ra kell állítani, így a beszámoló ezen során az a szöveg jelenik meg, amelyet megadtunk a **QuickRep** vezérlőelem **ReportTitle** tulajdonságában.

A „**Column Header**”, illetve a „**Page footer**” feliratú sávok rendre a fejezetek statikusan megadott címeit (**QRLabel**), illetve a beszámoló oldalainak lábjegyzetét tartalmazzák. (Oldalszámok megjelenítéséhez a **QRSysData Data** tulajdonságának értékét állítsuk **qrsPageNumber**-ra, a **Text** tulajdonságát pedig **Oldalszám:-ra!**)



A kész jelentés:

Azonosító, cénév	Rendelés száma	Fizetés módja	Összesen
1221 Kauai Dive Shoppe			
1221	1023	Check	£4,674.00
1221	1076	Visa	£17,781.00
1221	1123	Check	£13,945.00
1221	1169	Credit	£9,471.95
1221	1176	Visa	£4,178.85
1221	1269	Credit	£1,400.00
1231 Unisco			
1231	1060	Check	£15,355.00
1231			
1231			
1231			

1351	1163	Credit	£342.00
Oldalszám: 1			

Különböző sávokat alkalmazva természetesen másféle beszámolókat is készíthetünk:

Ügyfelek						
Title						
1	2	3	4	5	6	7
Azonosító	Ényké	Név	Cím	Telefo		
Column Header						
2	ACCT_NBR		FIRST_NAME	CITY	TELEPHONE	
			LAST_NAME	ADDRESS_1		
			ZIP			
3			Érdeklődési kör	INTERESTS		
4	Detail					
	Oldalszám: 2					
	Page 1 of 10					

Itt a **Table** vezérlőelem **DatabaseName** tulajdonságának értékét *DBDEMOS*-ra állítottuk, a **TableName** tulajdonság értékét pedig *clients.dbf*-re. A feliratokat a **QRLabel** vezérlőelem segítségével jelenítettük meg, amely a **QRDBText** vezérlőelemmel szemben csak egy statikus szöveget reprezentál, nem pedig egy adat-táblából kiolvasott szövegmező tartalmát. A kész beszámoló:

Close

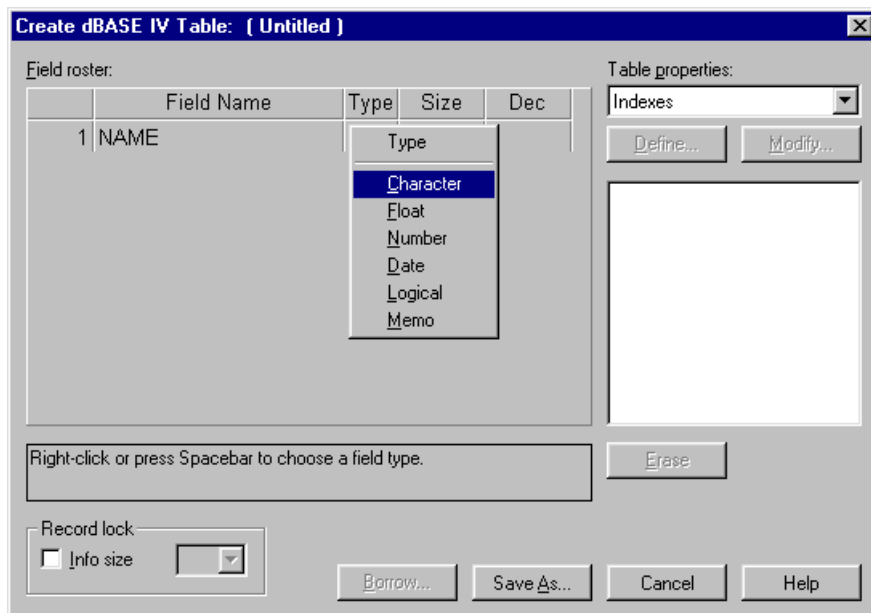
Ügyfelek

Azonosító	Fénykép	Név	Cím	Telefon
1023495		Jennifer Davis	Wellesley 100 Cranberry St. 02181	516-292-3946
Érdeklődési köre: Enjoys horseback riding and paints.				
2094056		Arthur Jones	Los Altos 10 Hunnewell St 94024	415-941-4321
Érdeklődési köre: Has five children. Loves to travel.				

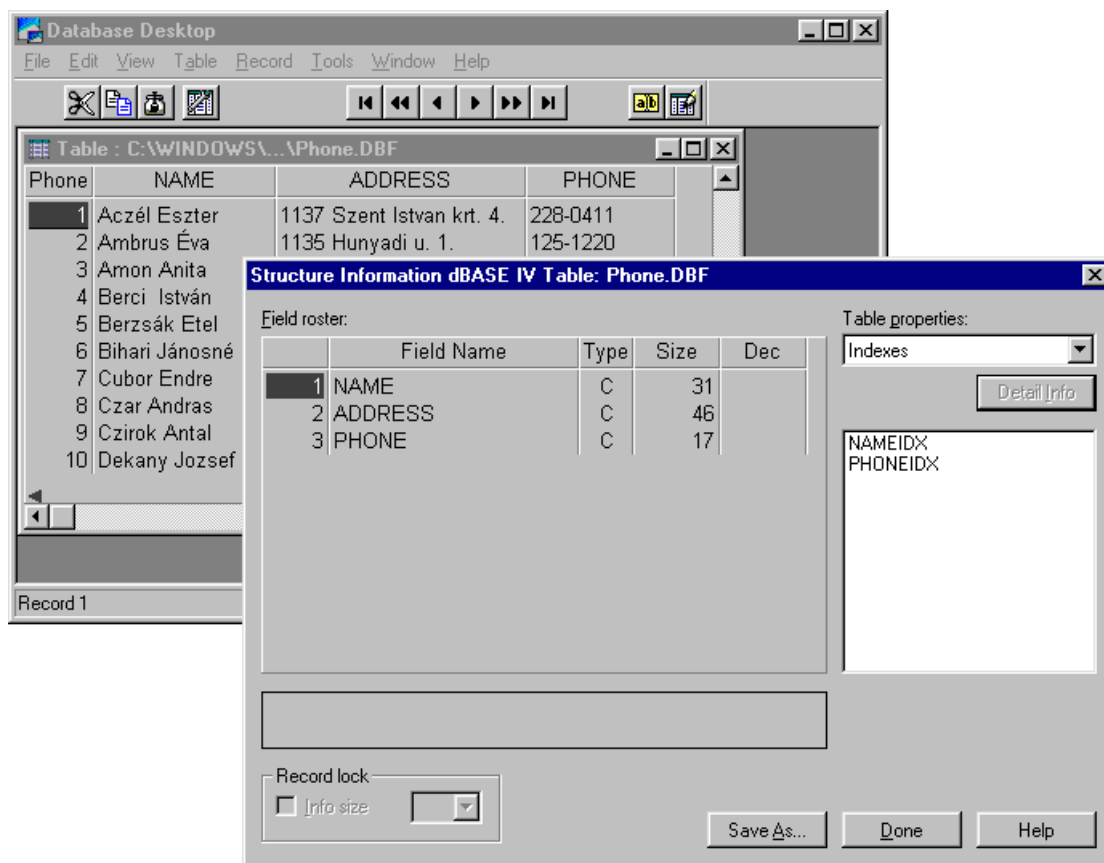
Page 1 of 2

▼ Készítsünk elektronikus telefonkönyvet! A megoldás foglalja magában az ismerőseink adatait tartalmazó adattáblát, az adatok megjelenítését végző alkalmazást, illetve az adatokat nyomtatásra kész formában megjelenítő jelentést! (*Telefon_Könyv*)

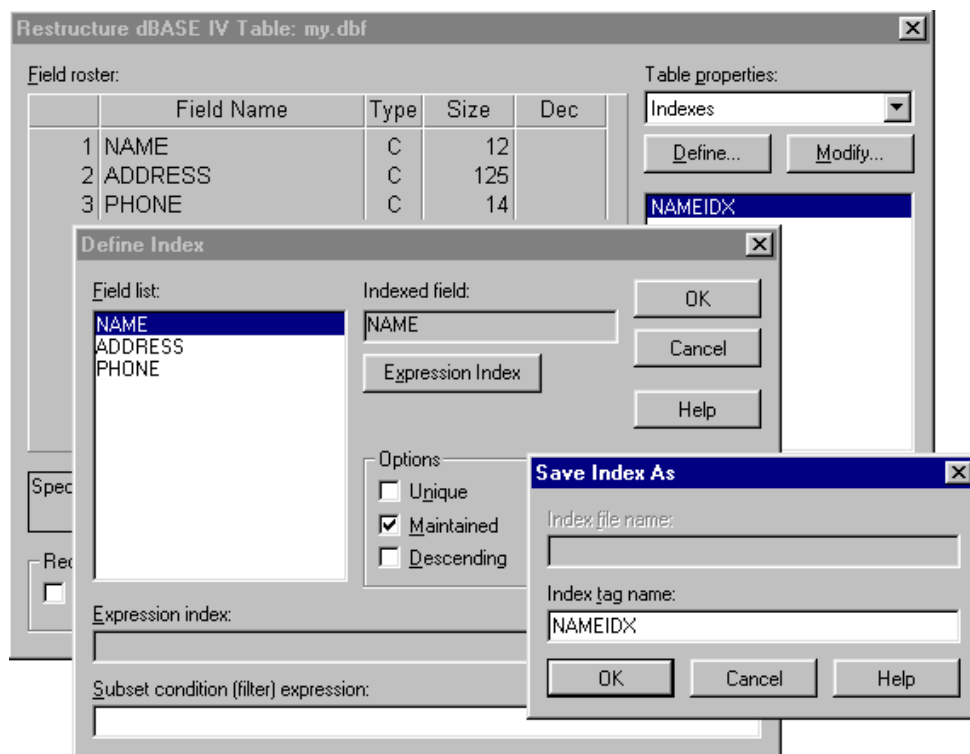
A munkát az adattábla - „*Database Desktop*” alkalmazás segítségével történő - létrehozásával kezdjük (*File | New | Table | dBASE IV*):



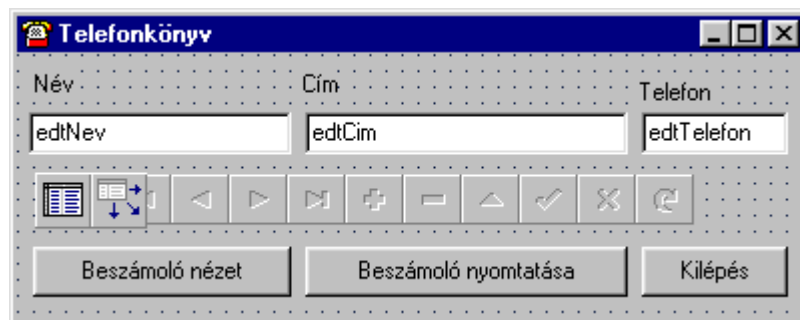
A struktúra létrehozása után mentjük el a táblát (*Save As...*)! A *File | Open | Table* és a *Table | Edit Data* menüpont bejelölése után adjuk meg a rekordokat:



Az indexek létrehozásához-, illetve módosításához válasszuk ki először a **Table | Restructure** menüpontot, utána pedig kattintsunk a **Define**, illetve a meglévő index változtatásához a **Modify** nyomógombon! A megjelenő párbeszédablakban válasszuk ki a megfelelő adatmezőt a (**Field Name**), és kattintsunk az **OK** gombon! A megjelenő „**Save Index As**” párbeszédablakban pedig adjuk meg a létrehozandó index nevét:



A Delphi alkalmazásunk létrehozását azzal kezdjük, hogy elhelyezzük a formon a következő ábrán is látható vezérlőket: egy **Table**, egy **DataSource**, egy **DBNavigator**, illetve három-három **Label**, **DBEdit** és **Button** vezérlőelemet:



A **Kilépés** feliratú nyomógombon való kattintásra az alkalmazásunk befejeződik:

```
procedure TForm1.btnKilepesClick(Sender: TObject);
begin
    Close;
end;
```

Az adattábla rekordjai közötti léptetéshez, illetve az aktuális bejegyzés adatainak megjelenítéséhez megfelelő módon össze kell kötni a **Table**, a **DataSource**, a **DBNavigator** és a három **DBEdit** vezérlőelemet.

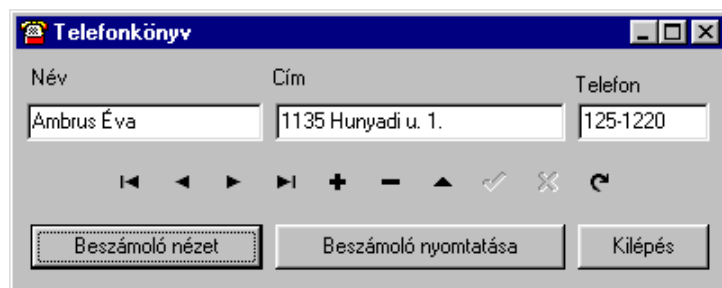
A **DBNavigator**, illetve a három **DBEdit** vezérlőelem **DataSource** tulajdonságának értékét állítsuk be az adatkészletező objektumunk nevére (**DataSource1**)! A **DBEdit** vezérlőelemek **DataField** tulajdonságának értékeként pedig rendre adjuk meg a **NAME**, az **ADDRESS**, illetve a **PHONE** mezőnevet!

A **DataSource** vezérlőelem **DataSet** tulajdonságának értékét állítsuk be az adatkészletező objektum névére (*Table1*)! A **Table** vezérlőelem **TableName** tulajdonságának értékeként adjuk meg a létrehozott adattáblánk nevét (a példában *phone.dbf*), a **DatabaseName** tulajdonságát pedig dinamikusan határozzuk meg, az alkalmazás elindításakor (a parancssor tartalmát visszaadó **ParamStr** tömb 0-ik eleme mindig a végrehajtott alkalmazás elérési útját, és nevét tartalmazza):

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    Table1.Active:=false;
    Table1.DatabaseName:=ParamStr(0)[1]+'..\TelefonKönyv';
    Table1.Open;
end;
```



A „*Beszámoló nézet*” feliratú nyomógombra való kattintással az adattáblánk tartalmát nyomtatásra kész formában jeleníthetjük meg úgy, hogy meghívjuk az alkalmazás második űrlapján elhelyezett **QuickRep** vezérlőelem **Preview** módszerét:



```
procedure TForm1.btnBeszamoloClick(Sender: TObject);
begin
    Form2.QuickRep1.Preview;
end;
```

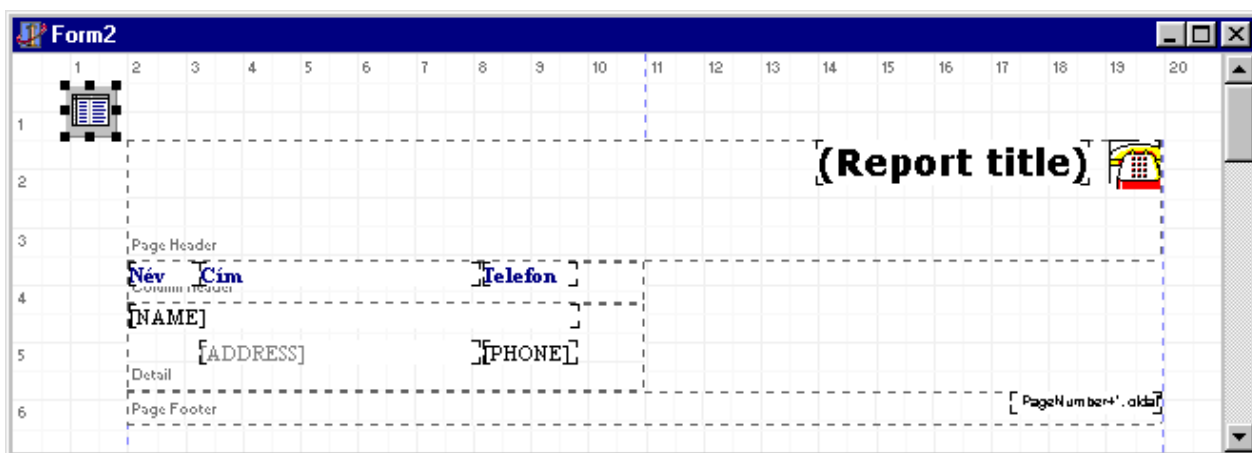


Ahhoz, hogy a fenti képhez hasonló beszámolót készíthessünk, először ki kell választani a **File** menüből a **New Form** menüpontot, és elhelyezni a megjelenő formon egy **QuickRep** (**QReport** palettalapon) és egy **Table** vezérlőt. A **QuickRep** vezérlő **ReportTitle** tulajdonságába írjuk be a *Telefonkönyv* szöveget!

A **QuickRep** vezérlőelem **DataSet** tulajdonságát állítsuk be a **Table** vezérlőelemre (*Table1*), a **Table** vezérlőelem beállításai pedig megegyeznek a *Form1*-en elhelyezett **Table** vezérlő beállításaiával. A **TableName** tulajdonságának értéke a *phone.dbf*, a **DatabaseName** beállítása pedig:

```
procedure TForm2.FormCreate(Sender: TObject);
begin
    Table1.Active:=false;
    Table1.DatabaseName:=ParamStr(0)[1]+'..\TelefonKonyv';
    Table1.Open;
end;
```

A **QuickRep** vezérlőelem **Band** tulajdonságán keresztül állítsuk **true**-ra a következő tulajdonságokat: **HasColumnHeader**, **HasDetail**, **HasPageFooter** és **HasPageHeader**! A **Page** tulajdonságon keresztül elért **Columns** mezőben pedig adjunk meg értékként 2-t (így az adatok két oszlopban lesznek megjelenítve)! A megjelenő **Page Header**, **Detail** stb. sávok magasságának igazítása után helyezzük el a sávokban a következő ábrán is látható vezérlőelemeket:



A „Page Header” sáv vezérlőelemei:

Vezérlőelem	Tulajdonság	Érték	Megjegyzés
QRSysData	Data	<i>qrsReportTitle</i>	
	Font.Size	18	
QRImage	Picture	(Icon)	Az alkalmazás ikonja

A „Column Header” sáv vezérlőelemei:

Vezérlőelem	Tulajdonság	Érték	Megjegyzés
QRLabel	Caption	<i>Név</i>	Mind a három vezérlőelem Font.Color , Font.Size , illetve Font.Style.Bold tulajdonságának értékét állítsuk be rendre <i>clNavy</i> -re, 10-re, illetve true -ra.
QRLabel	Caption	<i>Cím</i>	
QRLabel	Caption	<i>Telefon</i>	

A **Detail** sáv vezérlőelemei:

Vezérlőelem	Tulajdonság	Érték	Megjegyzés
QRExpr	Expression	<i>[NAME]</i>	Mind a három vezérlőelem Font.Size tulajdonságának értékét állítsuk be 10-re.
QRExpr	Expression	<i>[ADDRESS]</i>	
QRExpr	Expression	<i>[PHONE]</i>	

A „Page Footer” sáv vezérlőelemei:

Vezérlőelem	Tulajdonság	Érték	Megjegyzés
QRExpr	Expression	<i>PageNumber+ '. oldal'</i>	
	Font.Size	8	

Az oldalra vonatkozó többi beállítás elvégzéséhez kattintsunk a **QuickRep** vezérlőelemen az egér jobb oldali gombjával, és válasszuk ki a megjelenő menüből a „**Report Settings**” menüpontot! Az elvégzendő beállításokat a következő ábráról olvashatjuk le:

Report Settings

Paper size: A4 210 x 297 mm | Width: 210.0 | Length: 297.0 | Portrait

Margins: Top: 15.00 | Left: 20.00 | Column space: 0.00 | Bottom: 15.00 | Right: 10.00 | Number of columns: 2

Other: Font: Arial | Size: 8 | Units: MM

Page frame: ☐ Top ☐ Left ☐ Bottom ☐ Right | Color: [Black] | Change | Frame width: 1

Bands:

	Length		Length
<input checked="" type="checkbox"/> Page header	21.17	<input checked="" type="checkbox"/> Page footer	5.95
<input type="checkbox"/> Title		<input type="checkbox"/> Summary	
<input checked="" type="checkbox"/> Column header	7.28	<input checked="" type="checkbox"/> Print first page header	
<input checked="" type="checkbox"/> Detail band	15.21	<input checked="" type="checkbox"/> Print last page footer	

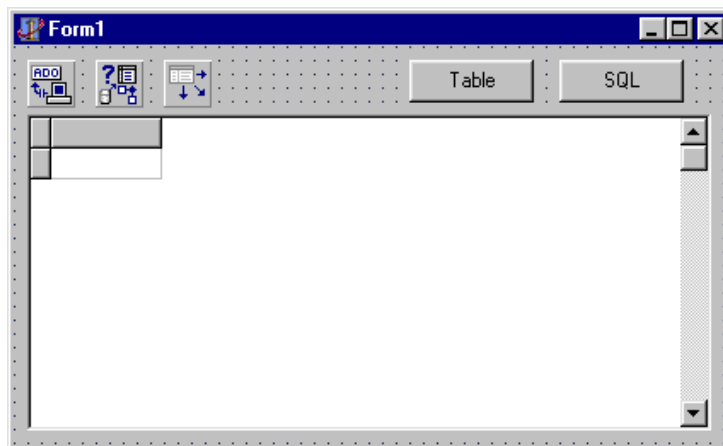
Buttons: About QuickReport | Preview | Apply | OK | Cancel

Ahhoz, hogy az alkalmazásunkban a „*Beszámoló nyomtatása*” feliratú nyomógombra kattintva elvégezhesük a nyomtatást, csupán a **QuickRep** vezérlőelem **Print** metódusának hívására van szükség:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
    Form2.QuickRep1.Print;
end;
```



Helyezzünk el a formon egy **ADOConnection**, egy **ADODataSet**, egy **DataSource**, egy **DBGrid** és két **Button** vezérlőelemet!




Állítsuk be a **DBGrid** vezérlőelem **DataSet** tulajdonságának értékét a **DataSource** vezérlőelem nevére (*DataSource1*), a **DataSource** vezérlőelem **DataSet** tulajdonságának értékét pedig az **ADODataSet** vezérlőelem nevére (*ADODataSet1*)!

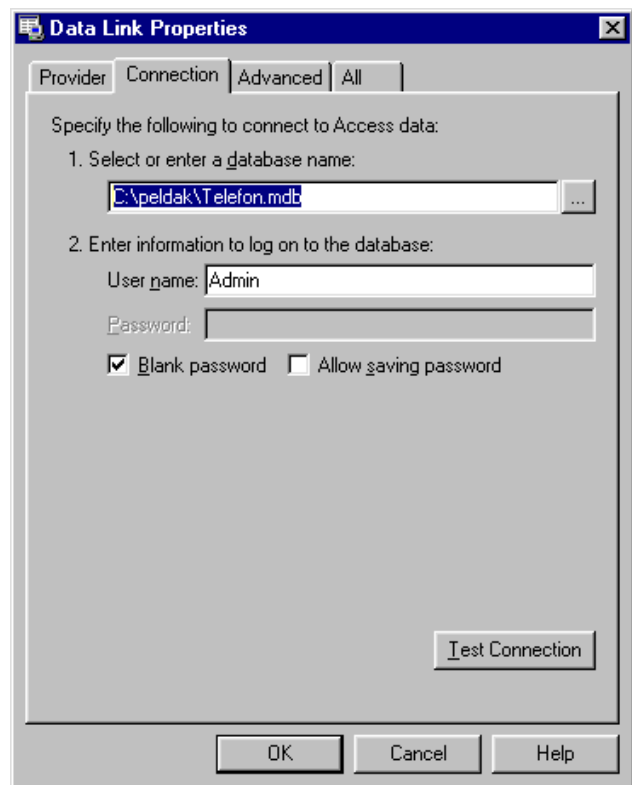
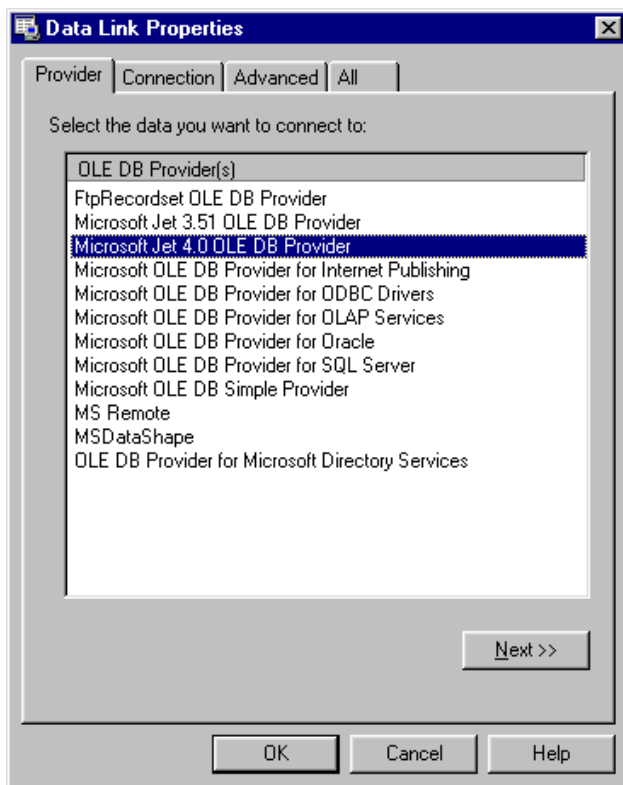
Az **ADODataSet** vezérlőelemet kapcsoljuk hozzá az **ADOConnection** vezérlőelemhez úgy, hogy beállítjuk a **Connection** tulajdonságának értékét az **ADOConnection** vezérlőelem nevére (*ADOConnection1*). Az alkalmazásunk **Table** feliratú nyomógombjára kattintva beállítjuk az **ADODataSet** vezérlőelem **CommandType** (a lekérdezési utasítás típusa) és **CommandText** (az utasítás szövege) tulajdonságait *cmdTable*-ra (adattáblázat-lekérdezés), illetve *telefon*-ra (az adattáblázat neve) értékre.

```
procedure TForm1.btnTableClick(Sender: TObject);
begin
  ADODataSet1.Close;
  ADODataSet1.CommandType:=cmdTable;
  ADODataSet1.CommandText:='telefon';
  ADODataSet1.Open;
end;
```

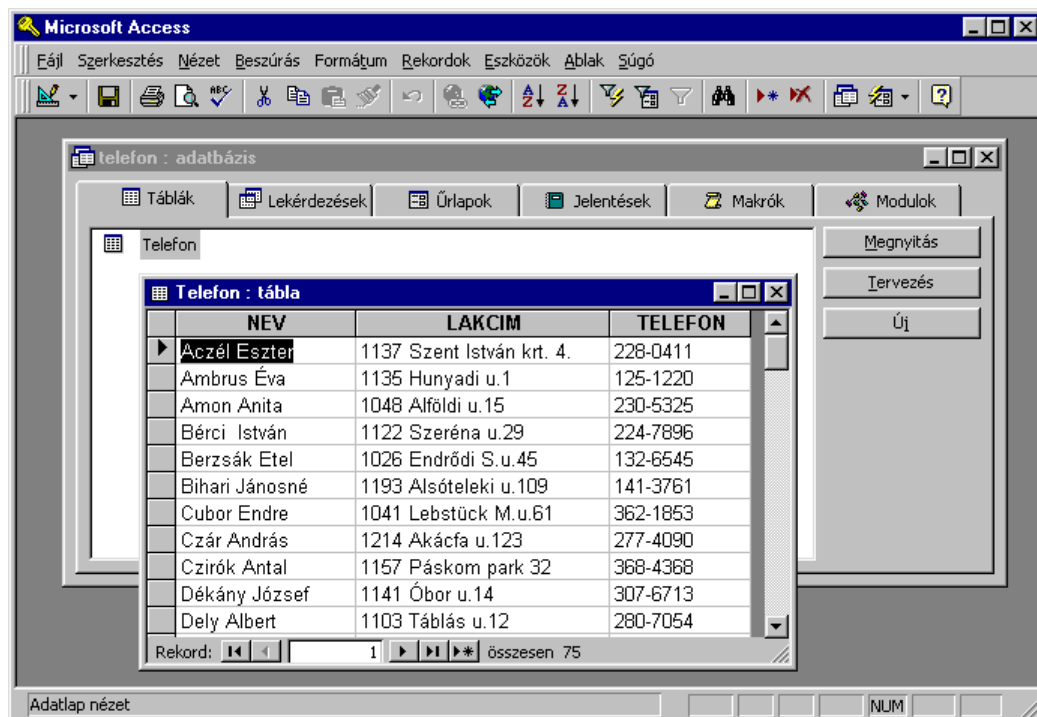
Ha az **SQL** feliratú nyomógombra kattintunk, az **ADODataSet** vezérlőelem **CommandType** tulajdonságának értékét *cmdText*-re (szöveges lekérdezésre), a **CommandText** értékét pedig a megfelelő SQL-utasítás szövegére állíthatjuk. Például a következő utasítással a *telefon* táblából csak azokat a rekordokat kérjük le, amelyeknél a *nev* mező értéke *S* betűvel kezdődik:

```
procedure TForm1.btnSQLClick(Sender: TObject);
begin
  ADODataSet1.Active:=false;
  ADODataSet1.CommandType:=cmdText;
  ADODataSet1.CommandText:='SELECT * FROM telefon WHERE nev LIKE "S%";'
  ADODataSet1.Active:=true;
end;
```

Ahhoz, hogy a programunk működjön, el kell végeznünk a legfontosabb beállítást, meg kell határoznunk az ún. kapcsolatteremtő karakterláncot. Ezt az **ADOConnection** vezérlőelem **ConnectionString** tulajdonságán keresztül tehetjük meg. A **ConnectionString** értékének összeállításához kattintsunk az értékmezőben látható  gombra, majd a megjelenő „Form1.ADOConnection1 ConnectionString” feliratú párbeszédablakban jelöljük be a „Use Connection String” választógombot, és kattintsunk a **Build** nyomógombra! A megjelenő „Data Link Properties” párbeszédablak **Provider** lapján válasszuk ki a „Microsoft Jet 4.0 OLE DB Provider” bejegyzést, ezután pedig válasszuk ki az adatbázis nevét a **Connection** lapon!



A példánkban a *Microsoft Access* program segítségével létrehozott adatbázist használunk, amely egyetlen - *Telefon* nevű – adattáblát tartalmaz:



A „*Test Connection*” nyomógombon kattintva tesztelhetjük a kapcsolat létrehozásának sikerességét. Ezek után az **OK** gombokkal zárjuk le a kinyitott párbeszédablakokat és indítsuk el az alkalmazást!

NEV	LAKCIM	TELEFON
Áczél Eszter	1137 Szent István krt. 4.	228-0411
Ambrus Éva	1135 Hunyadi u.1	125-1220
Amon Anita	1048 Alföldi u.15	230-5325
Bérci István	1122 Szeréna u.29	224-7896
Berzsák Etel	1026 Endrődi S.u.45	132-6545
Bihari Jánosné	1193 Alsóteleki u.109	141-3761
Cubor Endre	1041 Lebstűck M.u.61	362-1853
Czár András	1214 Akácfa u.123	277-4090
Czirók Antal	1157 Páskom park 32	368-4368

NEV	LAKCIM	TELEFON
Sánta Ágnes	1040 Liszt Ferenc tér 10	231-3300
Schulz Lajos	1038 Ráday u. 55	202-9999
Snekszer Béla	1025 Zöldmáli u.43/b	101-3200
Szabó Lajosné	1117 Pesti u.23	276-0203
Székely Árpád	1089 Sárkány u.67	204-8767
Szterló Géza	1028 Kertváros u.99	298-9999

Az *ADODataSet* „mindent tudó” vezérlőelemen kívül használhatjuk az *ADOTable*, az *ADOQuery* stb. vezérlőelemeket is:

A példánkban a formon három *DataSource* vezérlőelemen kívül egy *ADOConnection*, illetve egy *ADOTable*, egy *ADOQuery* és egy *ADODataSet* vezérlőelemet helyeztünk el (ezen vezérlőelemek *Connection* tulajdonságának értékét be kell állítani az *ADOConnection* vezérlőelem nevére, azaz az *ADOConnection1*-re).

Az *ADOConnection* vezérlőelem *ConnectionString* tulajdonságának beállítását végezzük el a fentiekben megismertetett módon! A *DataSource* vezérlőelemek *DataSet* tulajdonságának értékét állítsuk be rendre *ADOTable1*-re, *ADOQuery1*-re és *ADODataSet1*-re!

Az *ADOTable* vezérlőelem *TableName* tulajdonságának értékét állítsuk be *Telefon*-ra, az *ADOQuery* vezérlőelem *SQL* tulajdonságának értékét pedig a következő utasításra!

```
SELECT * FROM telefon
```

Az *ADODataSet* vezérlőelem esetén pedig a következő beállításokra lesz szükség:

Tulajdonság	Érték
<i>CommandType</i>	<i>cmdText</i>
<i>CommandText</i>	<i>SELECT * FROM telefon</i>

Az utolsó lépésben pedig aktiváljuk a három fenti vezérlőelemet azzal, hogy *Active* tulajdonságukat *true*-ra állítjuk!

Form1

NEV	LAKCIM	TELEFON
▶ Aczél Eszter	1137 Szent István krt. 4.	228-0411
Ambrus Éva	1135 Hunyadi u.1	125-1220
Amon Anita	1048 Alföldi u.15	230-5325

NEV	LAKCIM	TELEFON
▶ Aczél Eszter	1137 Szent István krt. 4.	228-0411
Ambrus Éva	1135 Hunyadi u.1	125-1220
Amon Anita	1048 Alföldi u.15	230-5325

NEV	LAKCIM	TELEFON
▶ Aczél Eszter	1137 Szent István krt. 4.	228-0411
Ambrus Éva	1135 Hunyadi u.1	125-1220
Amon Anita	1048 Alföldi u.15	230-5325