Debreceni Egyetem
Informatikai Kar
Esti Programozó Matematikus Szak

Szakdolgozat

Tárgyi eszköz nyilvántartó program fejlesztése Borland Delphiben

Témavezető: Dr. Bajalinov Erik

Készítette: Kovács Ferenc
IV. évf. programozó matematikus- hallgató

Debrecen
2007.

Tartalomjegyzék

Bevezetés

1. A téma megközelítése

A tárgyi eszköz a számvitelben azoknak az eszközöknek a gyűjtőneve, amelyek

- anyagi formában léteznek (szemben például az immateriális javakkal, amelyek „nem megfoghatóak”) és
- több mint egy éven keresztül maradnak a vállalkozás vagyonában.

A tárgyi eszközök a vállalati vagyon kevéssé mobil, dologi formában megtestesülő részei, melyek általában hosszabb idő alatt, fokozatosan használódnak el, miközben működőképességüket megőrzik.

A tárgyi eszközök a vállalat olyan erőforrásai, amelyek alapvetően meghatározzák a termelési folyamatok, technológia műszaki színvonalát, a termék minőségét, a munkaerő termelékenységét. Éppen ezért a tárgyi eszköz-nyilvántartás minden szervezet esetében döntő szerepű folyamat. A tárgyi eszköz-nyilvántartó rendszer feladata a vállalat különböző eszközeinek, jogcímek nyilvántartása. Ennek a kezelését a vállalatok könyvelői csoportjai végzik általában, de magát a leltárt nem ők készítik.

Jelenlegi munkahelyemen, a Debreceni Campus Kht. különböző telephelyein én magam is számtalan leltári nyilvántartást készítettem manuálisan, hiszen a tárgyi eszköz-nyilvántartó program használata itt is a könyvelői csoport tagjaira korlátozódik.

Dolgozatomban egy olyan tárgyi eszköz-nyilvántartó programot készítettem és mutatok be, amely ugyan nem felel meg a törvényben előírtaknak, de megkönnyíti mind a könyvelői csoportok, mind a leltárt végző dolgozók munkáját. Lehetőséget biztosít arra, hogy a leltári adatok áttekinthetőbbek legyenek, és egyszerűbben használható legyen maga az adatbázis.

A program létrehozásához, megírásához Delphi fejlesztői környezetet használtam. A programot elkészítettem Delphi 5 rendszerben BDE felhasználásával, valamint Delphi 6 rendszerben ADO segítségével.

2. A programozási nyelvről

A Delphi a Borland Software Corporation cég Windows grafikus felületen futó Object Pascal alapú negyedik generációs (4GL) programozási nyelve.

A Delphi nemcsak az objektumközpontú és a vizuális programozás előnyeit egyesíti, de a hatékonyság, és a nagyfokú ellenőrizhetőség terén is ideális választásnak bizonyul. A munkát jelentősen megkönnyítő, nagyszerűen alkalmazható eszköz, ugyanakkor a háttérben egy összetett programozási fejlesztőkörnyezet rejlik. Egy modern programozási nyelv. Egy integrált fejlesztőkörnyezetet, és egy vizuális komponenskönyvtárat tartalmaz.

Központi eleme az Object Pascal programnyelv, mely az objektumorientált nyelvek egyike. Rendelkezik mindazokkal a tulajdonságokkal, melyek egy modern OO nyelv jellemzői (interfészek, unikódos karakterek, tulajdonságok, kivételkezelés). Teljes mértékben támogatja a COM és ActiveX szabványokat, az objektumorientált grafikus könyvtárakat (VCL), és a kiterjeszthető és testre szabható gyors alkalmazásfejlesztő környezetet. Ezek a tulajdonságai tették a Delphi –t a legjobb Windows alatti programozási nyelvvé, és ezért választottam magam is a programom elkészítéséhez.

A téma kifejtése

I. A Delphi adatbázis-kezelő eszközei

1. Adatbázis-kezelés Delphiben

A Delphi egy relációs adatmodellen alapuló, objektumorientált adatbázis-felületet biztosít a programozónak. Nem relációs módon oldja meg az adatbázisokhoz való hozzáférést, hanem objektumos megoldást ad. Ezek az objektumok valójában procedurális kapcsolatot teremtenek a relációs táblákkal, de a program felé igazi objektumorientált-felületet nyújtanak. Ez a felület nagyon sokoldalú és komplex szolgáltatásokkal rendelkezik, így adatbázis-kezelő alkalmazások fejlesztésére az egyik legalkalmasabb fejlesztő eszköz.

A Delphiben történő adatbázis-kezelés lehetőséget biztosít az adatok formátumfüggetlen elérésére. Ez azt jelenti, hogy az alkalmazás szempontjából teljesen mindegy, hogy milyen szerkezetben tároltuk az adatokat, az alkalmazáson belül valamennyi formátumot ugyanúgy lehet kezelni. Ennek megvalósításához olyan kommunikációs felületet kellett készíteni, amely a különböző formátumok eszközmeghajtó programjainak felhasználásával egységes elérési módszert szolgáltat a különböző típusú adatbázisokhoz, az adatbázist alkotó táblákhoz.

Az így felépített alkalmazások három, jól elkülönített részre tagolhatók:

1. Az **adatbázis** ahol az adatok fizikailag tárolódnak. Az alkalmazott adatbázis típusának kiválasztása számos tényezőn múlik. Például: adatmennyiség, táblák mérete, felhasználók száma, válaszidő, adatmanipulációs műveletek gyakorisága.

2. Az **adatbázismotor** biztosítja a különböző formátumok elérését. Ennek érdekében különböző eszközmeghajtókkal rendelkezik. Ez tulajdonképpen nem más. mint egy egységes kommunikációs felület az adatbázis-kezelő alkalmazások számára.

3. Az **adatbázis-kezelő alkalmazás** (Delphi alkalmazás), amelyben különböző komponensek segítségével érhetjük el az adatokat.

1. 1. A Delphi adatbázis-kezelésének legfontosabb strukturális elemei – tárolási formátumoktól független adatkezelést biztosító programozási platformok

- BDE (Borland Database Engine): egységes adatbázis-programozói felületet biztosít, amely lehetővé teszi, hogy az alkalmazásunkban formátumtól függetlenül kezelhessük az adatbázisokat. Ezen felület segítségével teremthetjük meg a kapcsolatot a különböző formátumú adatbázisokkal. Azokat az adatbázis- eszközmeghajtókat, amelyeket a BDE tartalmaz, natív drivereknek nevezzük. A BDE alapú adatbázis-kezelő alkalmazások csak akkor működnek megfelelően, ha a számítógépen telepítve van a Borland Database Engine megfelelő verziója is.

- ODBC (Open Database Connectivity): egy API-n keresztül tudjuk használni. Ez az API interfész tartalmazza az adatkezeléshez szükséges függvényeket. Mi ezeket a függvényeket hívogatjuk, az ODBC Drive Manager pedig továbbadja a megfelelő meghajtóhoz: ahhoz, amelyik az éppen használt adatforráshoz használandó. Az ODBC már elavult. Az oka, hogy mégis tovább él az, hogy sok adatbázis-rendszerhez vagy formátumhoz nincs ADO meghajtó, de az ODBC támogatja.

- **ADO (Microsoft ActiveX Data Objects)**: Segítségével közvetlenül – BDE – nélkül elérhetünk különböző formátumokat.

1. 2. Aliasok

Egy adatbázis-kezelő alkalmazásban az adatok tárolási helyét és formátumát valamilyen módon specifikálni kell. Ha egy fixen meghatározott elérési utat írunk az alkalmazásunkba, akkor előfordulhat, hogy egy másik számítógépen a programunk nem fog helyesen működni.

A Delphi adatbázis-kezelő alkalmazásaiban lehetőségünk van egy un. alias segítségével hivatkozni az adatok tárolási helyére, illetve az adatbázis formátumára. Így az alkalmazásunkat teljesen függetleníthetjük az adatok tárolási formátumától. Ezzel a módszerrel az adatbázis-kezelő alkalmazásunkat bármilyen adatbázis-kezelő rendszer esetén egyszerűen használhatjuk.

1. 3. Az alkalmazáson belüli adatelérés legfontosabb komponensei

Az alkalmazáson belül az egyes adathalmazokat és adatbáziselemeket az adatelérési komponensekkel érhetjük el. Ezek a komponensek reprezentálják az alkalmazásban az adatbázis különböző szerkezeti elemeit, adattáblákat, lekérdezéseket, tárolt eljárásokat. Az alkalmazásban az adatok megjelenítését a felhasználói felületen az adatmegjelenítő komponensek végzik. Ezt a két komponenst az adatforrás komponensek kapcsolják össze. Az adatelérési és az adatforrás komponensek képezik az alkalmazásunk adatmodulját, melyeket az alkalmazás többi részétől elkülönítve tárolunk.

Az adatmegjelenítő komponensekhez kapcsolhatjuk hozzá az adatforrást. Ez a megfelelő komponenseik beállításával történik. Egy adatmegjelenítő komponens egyidejűleg csak egy adatforrás komponenshez kapcsolódhat, de egy adatforráshoz több adatmegjelenítő komponens is kapcsolódhat.

Az adatelérési komponensek csoportjába tartoznak mindazon objektumok, amelyek segítségével az adatbázisokban tárolt adatokhoz férhetünk hozzá. Ide sorolhatjuk az alkalmazás és adatbázis-kezelő kommunikációs felületének kezelésére szolgáló objektumokat, a fizikai adatbázist, az adatbázisban tárolt adattáblákat, az SQL alapú lekérdezéseket, a tárolt eljárásokat reprezentáló komponenst.

1. 4. Adatelérési komponensek

Az adatelérési komponensek csoportjába tartoznak mindazon objektumok, melyek segítségével az adatbázisokban tárolt adatokhoz férhetünk hozzá. Ide sorolhatjuk az alkalmazás, és az adatbázis-kezelő kommunikációs felületének kezelésére szolgáló objektumokat, a fizikai adatbázist, az adatbázisban tárolt adattáblákat, az SQL alapú lekérdezéseket, tárolt eljárásokat reprezentáló, valamint néhány speciális komponenst is.

Az adatelérési komponensek az alábbi palettákon találhatóak meg:

- Data Access: azok a komponensek helyezkednek el ezen a palettán melyek az adathozzáférés módjától függetlek, vagyis minden adatelérési mód esetében használhatóak, valamint itt találhatóak az XML adatformátumot támogató komponensek is.

- **BDE**: a komponenspalettán található komponensekkel a BDE-n keresztül férhetünk hozzá az adatbázisokhoz, és azok elemeihez.

- **ADO**: BDE nélküli adatelérést biztosítják az itt található komponensek. Az ADO-alapú komponensek ActiveX-es adatobjektumokat használnak az adatbázis információk eléréséhez. Mindezt az OLEDB kommunikációs felületen keresztül valósítják meg.

- **MIDAS**: itt találhatóak az ügyfél-adathalmazok, amelyek kezelése a memóriában történik.

- **Interbase**: az Interbase adatbázis-szerverekhez közvetlenül kapcsolódó komponensek helyezkednek el ezen a palettán.

- **dbExpress**: itt szerepelnek a keresztpaltformos fejlesztést támogató dbExpress alapú adatelérési komponensek.

1. 5. Az adathalmazok tartalmának megjelenítése

Az adathalmazok megjelenítésére az ún. adatmegjelenítő komponenseket használhatjuk. Az adatmegjelenítő komponenseket az alkalmazás különböző űrlapjaira helyezhetjük fel. Az adatelérési és adatmegjelenítési komponenseket közvetlenül nem kapcsolhatjuk egymáshoz, ezért az adathalmazok tartalmának megjelenítéséhez szükséges egy adatforrás objektum.

Az adatforrásokat a Delphiben a TDataSource komponensosztály objektumai reprezentálják, és ezekhez az objektumokhoz kapcsolhatjuk hozzá az adatmegjelenítő komponenseket.

A TDataSource típusú komponensek szolgáltatják a kapcsolatot az adatelérési és adatmegjelenítő komponensek között. Az adatforrás komponens segítségével teremthető meg a kapcsolat az adatelérési és adatmegjelenítő komponensek között. Az adatmegjelenítő komponensek mindig az adatforrásra hivatkoznak, ezáltal biztosítva a programunk függetlenségét az adatbázis típusától, formátumától. A Delphiben ez a komponensosztály teszi lehetővé többrétegű alkalmazások készítését is.

Az adatforráshoz egyidejűleg csak egy adathalmazt lehet kapcsolni, de egy

adatforráshoz tetszőleges számú adatmegjelenítési komponens kapcsolódhat. Az adatforrás komponensekkel lehetőségünk van kapcsolatot teremteni az adatelérési és adatmegjelenítő komponensek között, az adathalmaz és adatelérési komponensek közötti kapcsolatot ideiglenesen felfüggeszteni, az adatforráshoz kapcsolt adathalmaz aktuális állapotát lekérdezni, az adathalmazok szerkesztő módba kapcsolását automatikussá tenni.

Az adathalmazok és adatforrások összekapcsolására a TDataSource komponensosztály DataSet tulajdonsága szolgál. A DataSet tulajdonsággal határozhatjuk meg, hogy az adatforráshoz az alkalmazás melyik adathalmaz komponensét szeretnénk hozzákapcsolni.

Az adathalmaz tartalmát az adatmegjelenítő komponensekkel tehetjük láthatóvá. Az adatmegjelenítő komponenseket a DataControls komponenspalettán találhatjuk. Az adatmegjelenítő komponensek tetszőleges típusú, elérésű és formátumú adathalmazok tartalmának megjelenítésére, szerkesztésére szolgálnak. Gyakorlatilag olyan kezelőszervek, melyek az adatokat nem a felhasználói felületről, vagy az alkalmazás kódjából kapják, hanem egy adathalmazból olvassák ki. Mivel a komponenseknek nincs közvetlen kapcsolatuk a fizikai adathalmazzal, ezért a megjelenítés szempontjából teljesen mindegy, hogy milyen adatszolgáltató komponenstől kapják az adatokat. Ezzel a megoldással az adatbázis-kezelő programjainkban teljesen különválaszthatjuk az adatfeldolgozási, illetve adatmegjelenítő modulokat.

Adatmegjelenítő komponensek:

- **TDBText –** az adathalmaz egy mezőjének feliratként történő megjelenítése

- **TDBEdit** – az adathalmazok mezőinek megjelenítésére és tartalmának szerkesztésére használható komponens

- **TDBMemo** – feljegyzés típusú, többsoros karakterláncokat tartalmazó adatmezők megjelenítése és szerkesztése

- **TDBImage** – képeket tartalmazó adatmezők megjelenítése

- **TDBListBox** – nem adatmegjelenítésre, hanem a hozzá kapcsolt adatmező lista alapján történő szerkesztésére használható; a lista elemeivel lehet módosítani az adatmező értékét

- **TDBComboBox** – nem adatmegjelenítésre, hanem a hozzá kapcsolt adatmező legördülő lista alapján történő szerkesztésre használható; a lista elemeivel, valamint a felhasználó által beírt értékekkel is módosíthatjuk az adatmező tartalmát
- **TDBCheckBox** – olyan adatmezők tartalmát lehet jelölő négyzettel megjeleníteni és módosítani, melyek kétféle értéket vehetnek fel; a logikai adatmezők értékének tipikus eszköze.

- **TDBRadioGroup** – az adatmezők tartalmának módosítását és megjelenítését választógombokkal megvalósító komponens. Általában olyan mezőtípusoknál használjuk, melyek előre meghatározott értékeket vehetnek fel.

- **TDBRichEdit** – formázott szövegek megjelenítésére és szerkesztésére szolgáló komponens

- **TDBGrid –** az adathalmaz táblázatos megjelenítését és szerkesztését megvalósító komponens

- **TDBNavigator** – az adathalmazok kezelésére szolgáló komponens, mely a leggyakoribb navigációs műveletek elérését teszi lehetővé nyomógombok formájában

- **TDBLookupListBox** – a TDBListBox komponenshez hasonlóan ez a komponens is az adathalmaz egy mezőjének lista alapján történő módosítását teszi lehetővé, de itt a lista elemeit egy másik adathalmaz mezőjéhez tartozó értékek határozzák meg. Tipikus felhasználási területe a hivatkozási integritásokat tartalmazó adattáblák mezőinek szerkesztése.

- **TDBLookupComboBox** – funkciója megegyezik a TDBLookupListBox komponenssel, de itt legördíthető listában jelenítjük meg az adatokat.

Az általam készített programban használt adatmegjelenítő komponenseket a következő nagy egységben, **Az alkalmazás fejlesztése** című fejezetben fogom részletezni.

2. Az adatforrás és az adatmegjelenítő komponensek kapcsolata

Az adatmegjelenítő komponenseket mindig egy adatforráson keresztül csatlakoztathatjuk az adathalmazokhoz, ezért teljesen mindegy, hogy milyen formátumú adathalmazt szeretnénk megjeleníteni, vagy szerkeszteni.

Minden adatmegjelenítő komponens rendelkezik DataSource tulajdonsággal, amiben azt az adatforrást kell beállítanunk, amelyhez tartozó adattábla tartalmát meg akarjuk jeleníteni.

Az adatelérési komponensek nagy részének fontos jellemzője még a **DataField**, mellyel azt állítjuk be, hogy az adott adathalmaz melyik mezőjének értékét jelenítse meg. Az adatmegjelenítő komponens az aktuális rekord itt meghatározott mezőjének tartalmát jeleníti meg, illetve teszi szerkeszthetővé. A **DataField** tulajdonság értékének az adathalmaz tetszőleges mezőjének nevét beállíthatjuk. Az adatforrás komponensek **DataSet** tulajdonságát futási időben tetszőleges helyen megváltoztathatjuk.

3. Az adathalmazok tartalmának szerkesztése

Az adatmegjelenítő komponensek nem csupán az adatok megjelenítését, hanem azok szerkesztését is lehetővé teszik. Az adatmegjelenítő komponenseken keresztül végrehajtott szerkesztési műveletek azonban csak akkor hajthatók végre, ha a megjelenítést végző komponensekhez tartozó adatforrás **AutoEdit** tulajdonsága igaz értékre van állítva. A tulajdonság igazra állításával engedélyezhetjük, hogy az adatforráshoz kapcsolt adatmegjelenítő komponensekben a módosítási műveleteket a program automatikusan engedélyezze-e. Ha a tulajdonság értéke *false*, akkor is szerkeszthető az adat, csak előtte az **Edit** metódust szükséges meghívni.

4. Az adathalmazok kezelése

Ahhoz, hogy az adatbázis-kezelő programunkban használni tudjuk az adatbázisban tárolt adatokat, valamilyen komponenssel reprezentálnunk kell azokat. Az adathalmazok közvetlen kapcsolatban állnak az adatbázismotorral, és leképezik az adatbázisokban található különböző objektumokat.

A Delphiben az adatok kezelésének alapvető egysége az adathalmaz-objektumok családja, melyeket a különböző típusú adatbázisokban tárolt relációk, metaadatok és egyéb adatbázis-elemek leképezésére használhatunk.

Az adatbázis-kezelő alkalmazásokban használt összes adathalmaz-objektum a **TDataSet Objektumosztályból** származik, és örökli az ott definiált, implementált, jellemzőket, eseményeket és metódusokat.

Mivel a **TDataSet** osztály nagyon sok absztrakt metódust tartalmaz, nem lehet közvetlenül használni az alkalmazásunkban, helyette az objektumosztály valamely leszármazottját kell példányosítani. Természetesen lehetőség van saját adatelérési osztály definiálására is az összes absztrakt metódus implementálásával.

A TDataSet objektum osztály írja le alapvetően az adathalmazok működését, függetlenül attól, hogy a későbbiek során milyen származtatott adathalmaz objektumot fogunk használni.

Az objektumosztályban az alábbi fontosabb tulajdonságok, metódusok találhatóak:

- Adathalmazok megnyitására és bezárására szolgáló Active tulajdonság, valamint az **Open** és **Close** metódus

- Navigációs műveletekhez szükséges **Bof, Eof** tulajdonságok és **First, Last, Next, Prior, MoveBy** metódusok

- Az adathalmazba tetszőlegesen beszúrható könyvjelzők kezelését megvalósító **Bookmark** tulajdonság, és az ezek használatához elengedhetetlen **CompareBookmarks, FreeBookmark, GetBookmark, GotoBookmark, BookmarkValid** metódusok

- A szűrési műveletek végrehajtására szolgáló **Filter, Filtered, FilterOptions** jellemzők, valamint a szűrt adattáblákban való navigálást megvalósító **FindFirst, FindLast, FindNext, FindPrior** metódusok

- A keresés funkcióját megvalósító **Locate** és **Lookup** metódusok

- Az adatmegjelenítő és adathalmaz komponensek kapcsolatát szabályozó **DisableControls** és **EnableControls** metódusok

- Az adattábla állapotainak, módosíthatóságának meghatározására szolgáló **State, CanModify, Modified** tulajdonságok, valamint a **CheckBrowseMode**, **IsEmpty** metódusok

- Az adathalmazok adatmanipulációs műveleteit megvalósító **Append, AppendRecord, Insert, InsertRecord, Delete, Edit, Post, Cancel** metódusok

- Az adathalmaz frissítését, újraolvasását végrehajtó **Refresh** metódus

- A mezőszintű ellenőrzésekhez szükséges **Constraints** jellemző

- Az adathalmaz mezőinek kezelésére szolgáló **Fields, FieldCount**, **FieldDefs, FieldList, FieldValues** tulajdonságok, és **ClearFields, FieldByName, FindField** műveletek

- A számított mezők kezelésére szolgáló **AutoCalcFields** és **InternalCalcFields** tulajdonságok

A fenti adatmezők, tulajdonságok és metódusok bármilyen adathalmaz komponens esetében felhasználhatók.

II. Az ADO

A Microsoft ActiveX Data Object (ADO) technológiája egy magas szintű adatelérési réteg. Nem használja a Borland Database Engine – t (BDE) és azt nem is szükséges az ügyfélgépekre telepítenünk, ehelyett a Microsoft ADO és OLE DB adatbázis-kezelő programjainak megléte szükséges. Mivel a Windows 2000 és a Windows 98 eleve tartalmazza ezeket a programokat, telepítési gondjaink a jövőben valószínűleg csökkenni fognak. Az ADO magát az API-t jelenti, amelyet a programozóknak használniuk kell az alkalmazások felépítéséhez, ha a Microsoft módszerét követik. Az ADO tervezésének az volt a célja, hogy tetszőleges programozói feladat esetén is csak egy adatfelületre legyen szükség. Olyan programozási objektumokat jelent, amelyek az adatbázist és a benne tárolt adatok felépítését képviselik. Azért készült, hogy a Microsoft legújabb és legnagyobb teljesítményű adathozzáférési modelljének, az OLE DB-nek az alkalmazásszintű felülete legyen. Az OLE DB minden adatforráshoz hatékony hozzáférést biztosít. Az ADO és az OLE DB együtt alkotják a Universal Data Access (univerzális adathozzáférés) stratégia alapjait. Az OLE DB univerzális hozzáférést biztosít mindenféle adathoz. Az ADO a fejlesztők számára egyszerűbbé teszi a programozást. Mivel az OLE DB-re épül, kihasználja az OLE DB gazdag, univerzális adathozzáférési eszköztárát.

Az ADO népszerűségének okai:

- egyszerűen használható
- különböző típusú adatbázisok elérhetőségét támogatja
- igen egyszerűen kezelhető felület
- több eszközt és nyelvet támogat.

Az ADO használatával táblák és lekérdezések hozhatók létre és módosíthatók, adatbázisok védelme állítható be, vagy külső adatforrások adatai érhetők el. A kódban használt ADO-objektumokkal az adatbázisban tárolt adatok is kezelhetők. Segítségével minden olyan adatforrás elérhető, amely ODBC illesztőprogrammal, vagy az adatforráshoz létrehozott OLE DB szolgáltatóval rendelkezik. Tulajdonképpen egy COM felületre épülő szolgáltatásszerver, amely egy dinamikus csatolású könyvtárban (DLL) található, így mindig a kliensgép memóriájába tölthető és ott is futtatható.

1. Az ADO objektumszerkezete

Az ADO egyik célja, hogy az OLE DB leggyakrabban használt tulajdonságait egyesítse, ezért objektummodellje olyan programozható objektumokból épül fel, melyek olyan platformokon alkalmazhatók, ami támogatja a mind COM-ot, mind az OLE Automationt.

Hét objektumot és négy objektumgyűjteményt tartalmaz:

Objektumok:

- Connection
- Command
- Recordset
- Error
- Field
- Parameter
- Property

Objektum gyűjtemények:

- Fields
- Parameters
- Properties
- Errors

Connection objektum:

Az adatforrás és az ADO objektum közötti kapcsolatot hozza létre, valamint a kapcsolatok kiépítéséhez szükséges adatok, tulajdonságok itt állíthatók be.

Command objektum:

Egy lekérdezés vagy utasítást képvisel, amely az adatforráshoz kerül feldolgozásra. Segítségével különböző parancsokat adhatunk ki a kiszolgálónak. Ezek lehetnek különböző lekérdezések, felparaméterezett lekérdezések. Az hogy milyen parancsokat használunk az adatbázis kiszolgálótól függ.

Recordset objektum:

Azt a rekordhalmazt tartalmazza, amelyet az adatforrás küld vissza egy lekérdezés után. Metódusokat biztosít az eredményhalmazok kezeléséhez. Segítségével az eredményhalmazok átvizsgálhatók, új rekordok vihetők fel, módosíthatók, törölhetők.

Tartalmaz egy Fields gyűjteményt, melynek segítségével az egyes rekordok értékei visszakereshetők, módosíthatók.

Field objektum:

A Recordset-ben lévő adat egy oszlopát jelenti. A Field objektummal új rekord állítható össze vagy módosíthatók a meglévő adatok. Frissítése az adat módosításakor automatikusan megtörténik, nincs szükség külön frissítési metódusra.

Error objektum:

Információt tartalmaz az adatforrás által küldött hibákról.

Parameter objektum:

Az információ egy darabkáját tartalmazza, vagy paramétert a Command objektumhoz kapcsolódóan. Információkat és adatokat szolgáltat a Command objektum számára. Akkor van rá szükség, ha a Command objektum lekérdezés-karakterlánca paramétereket igényel. Ezek a Parameter objektum segítségével olvashatók illetve írhatók.

Property objektum:

Az ADO objektum egy tulajdonságát adja meg. Információt szolgáltat a Connection, a Command, a Recordset, valamint a Field objektumok jellemzőiről. Nemcsak a tulajdonságok értékét és típusát, hanem azok attribútumait is visszaadja. Az attributumok megmondják, hogy egy objektum megadott tulajdonsága használható-e, szükséges-e, illetve olvasható vagy írható.

2. Delphi ADO komponensei:

- **ADOConnection**: adatbázis kapcsolat létrehozása ADO adatbázisokkal

- **ADOCommand**: nem adathalmazt eredményező SQL-utasítások végrehajtása ADO adathalmazokon

- **ADODataset**: általános adatelérési komponensosztály

- **ADOTable**: ADO-relációk komponensosztálya

- **ADOQuery**: lekérdezés végrehajtása ADO adatbázisokon; a lekérdezés tetszőleges táblára vonatkozhat

- **ADOStoredProc**: tárolt eljárás hívása ADO alapú adatbázisokban

- **RDSConnection**: számítógépek közötti adathalmazok mozgatására szolgáló komponens

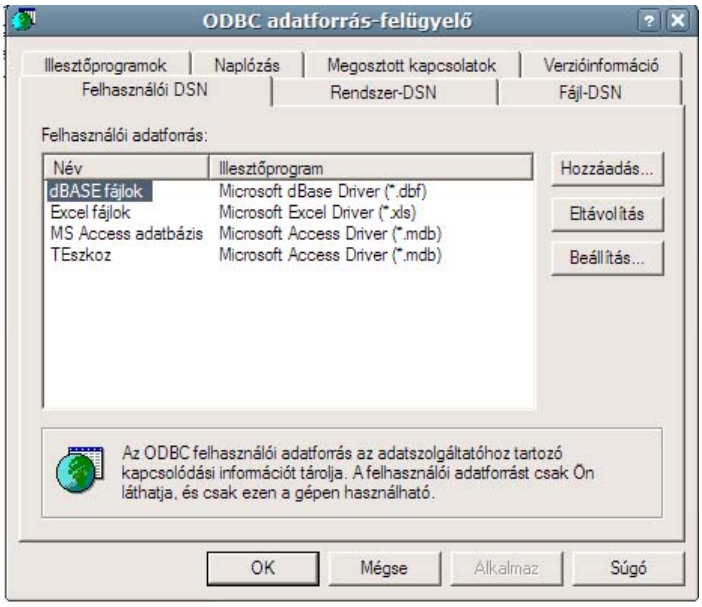
III. Az alkalmazás fejlesztése

1. Általános bevezetés

Az alkalmazás az adatokat egy Microsoft Access adatbázisban tárolja. Az adatbázis két táblát tartalmaz. Az egyik egy Telephely nevű, mely rekordjai négy mezőből állnak. ID a telephely azonosítója, NEV a telephely neve, CIM a telephely címe és TELEFON a telephely telefonszáma. A másik egy ESZKOZ nevű tábla, mely rekordjai hét mezőből állnak. Első egy ID azonosító nevű, második LELTARISZAM mely az eszköz leltári számát, egy harmadik ESZKOZ az eszköz megnevezése, a negyedik MENNYISEG az eszközök darabszámát, az ötödik BESZERAR az eszköz beszerzési árát, a hatodik BESZERDATUM a beszerzés dátumát, a hetedik a MEGJEGYZES nevű az eszközre vonatkozó megjegyzéseket jelöli.

Ahhoz, hogy az alkalmazás használni tudja az adatbázist, létre kellett hoznom egy natív vagy ODBC adatbázis-meghajtót. Ezt a következő elérési úton lehet megtenni.

Megnyitjuk a Vezérlőpult > Felügyeleti eszközök > ODBC adatforrások (Control Panel > Administrative Tools > ODBC Datasources) konzolt. Rákattintunk a Hozzáadás gombra. Az új adatforrás létrehozásához kiválasztjuk az adatforráshoz tartozó illesztőprogramot, mely jelen esetben a Microsoft Access Driver (*.mdb). A Befejezés gombra kattintva a Beállításokban meg kell adnunk az adatbázis nevét és az elérési útját. A speciális beállításokban meg lehet adni a felhasználói nevet és a jelszót. (Én ezeket a mezőket üresen hagytam, ezért az adatbázis eléréséhez semmit nem kell beírni az alkalmazás elindításakor.)



1. ábra

Mivel dolgozatomat két különböző Delphi verzió (Borland Delphi 5 és Borland Delphi 6) és két különböző adatkészletező vezérlőelem (BDE és ADO) felhasználásával készítettem, a két program adatforrás-vezérlő elemei teljesen különböznek, de egyéb elemeik azonosak, éppen ezért a programok forráskódjai ezektől eltekintve teljesen megegyeznek.

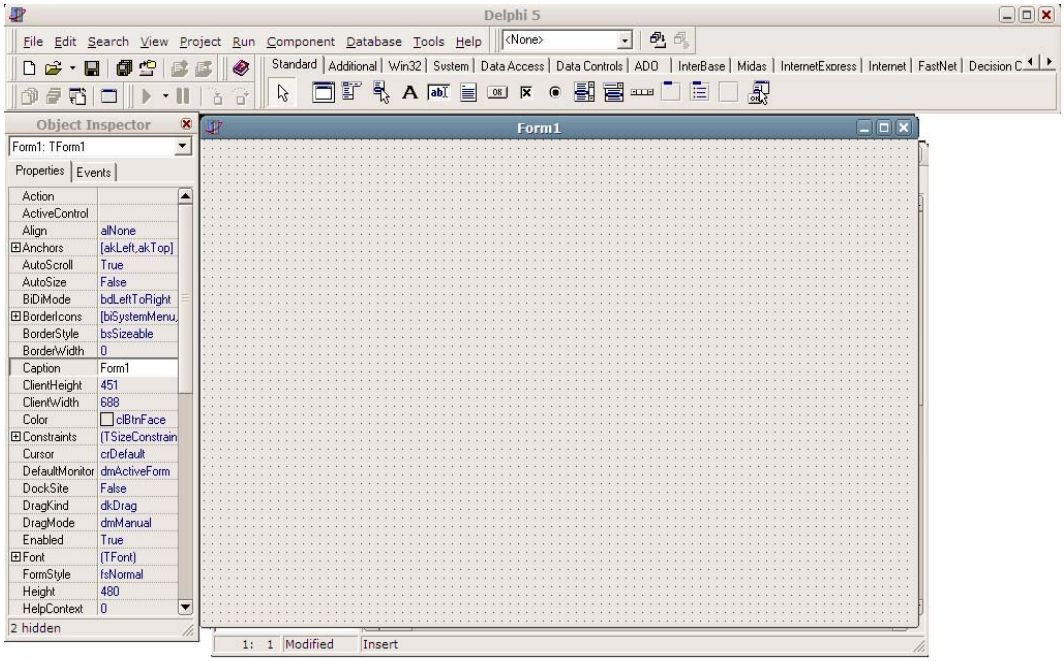
A következőkben ismertetem az általam készített adatbázis struktúráját, felépítését. Ezzel párhuzamosan bemutatom a munkafolyamatot, a használt komponenseket és azok indokait. A leírást a főformokkal kezdem, majd további modulokra bontva haladok az alkalmazás bemutatásával.

Az alkalmazás két formot használ. Egy Form1-et amely a fő form és egy Hozzaadasfrm-ot, mely a táblák bővítésénél van jelen.

1. 1. Formok

A Windows alkalmazásokhoz általában tarozik egy ablak, amelyet az program főablakának nevezünk. A Delphiben az ablakok tulajdonságait megtestesítő osztály a TForm, melynek deklarációját a Forms modul tatalmazza. A Windows alkalmazás főprogramjában történik a program ablakának létrehozása is. A főablakot az **Application** objektum **CreateForm** metódusának aktiválásával hozhatjuk létre. A létrehozott objektum kezeli az ablakon – mint megjelenítő eszközön – bekövetkező eseményeket.

Amikor új projektet nyitunk Delphiben, akkor az ablak modelljét (formját) látjuk a képernyő jobb oldalán és a Form1 objektum-felügyelőjét (Object Inspector) a bal oldalon.



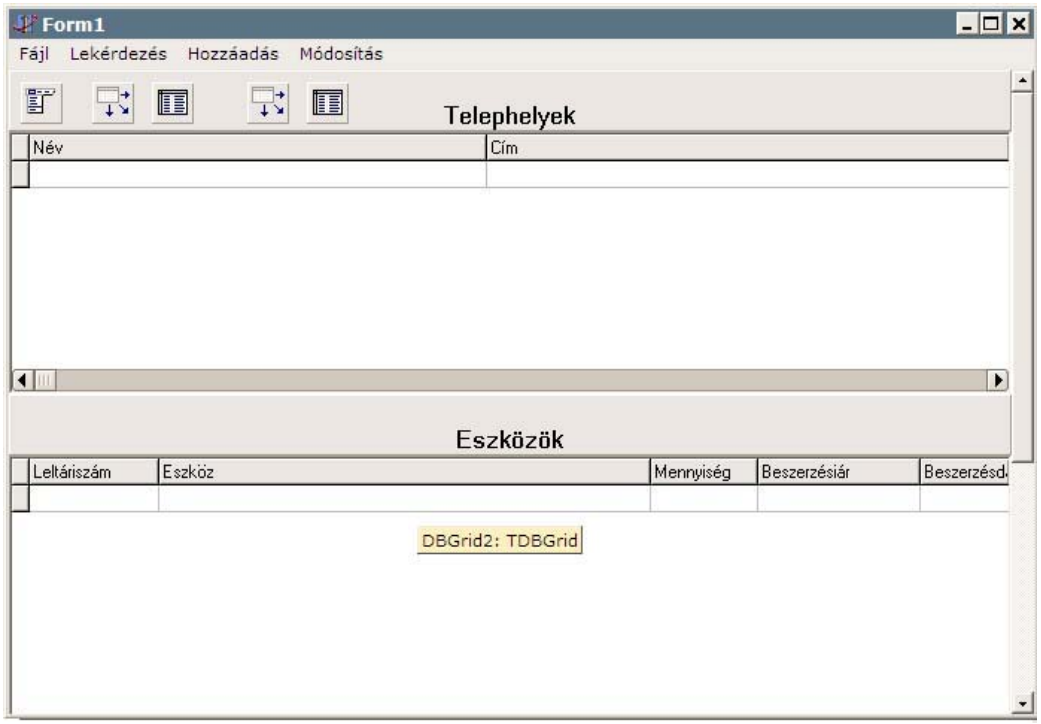
2. ábra

Az objektum-felügyelőben beállíthatjuk a tulajdonságok értékét és az eseménykezelő eljárásokat, melyek futási időben is megváltoztathatóak.

1. 2. Az alkalmazás egyéb komponensei, elemei

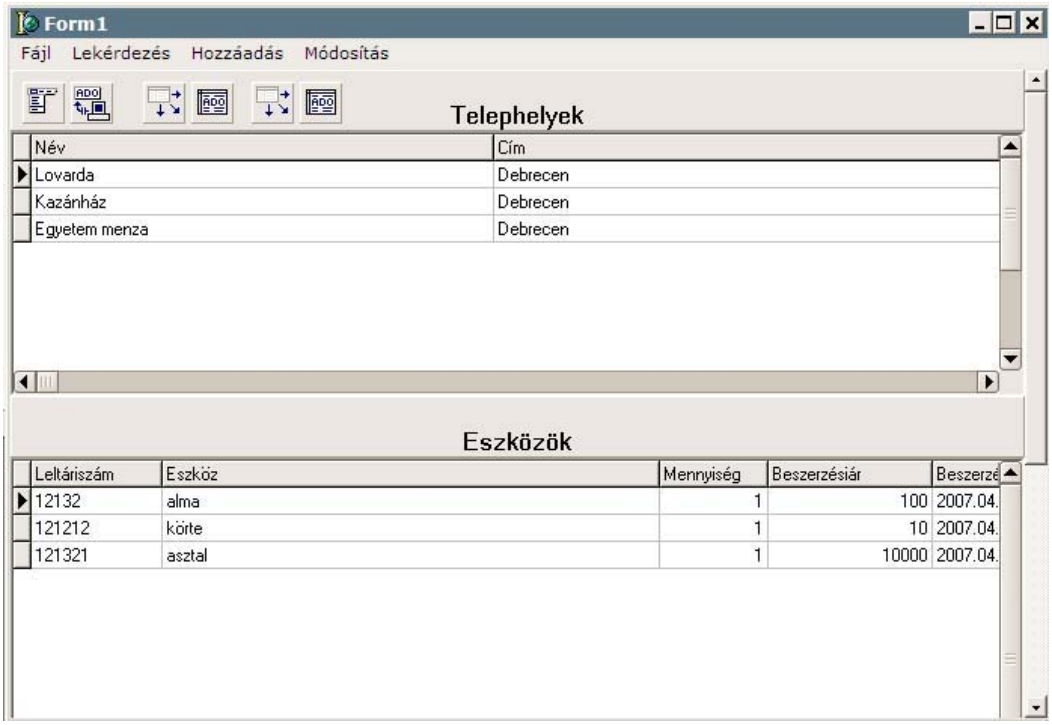
- Form1

A BDE komponensekkel elkészített Form1 tartalmaz egy **MainMenu** komponenst: **MainMenu1**, két **DataSource** adatforrás vezérlőelemet: **DataSource1** és **DataSource2**, két hozzátartozó **Table** vezérlőelemet: **Table1** és **Table2**. A főform tartalmaz továbbá két **BDGrid** adatmegjelenítő komponenst: **DBGrid1** és **DBGrid2**, melyek a **Label1** és a **Label2** címkével jelölt táblák adatait jelenítik meg.



3. **ábra**

Az ADO komponensekkel elkészített Form1 csak az adatbázis eléréséhez szükséges komponensekben tér el. Ezek: egy darab **ADOConnection** vezérlőelem és két darab **ADOTable** vezérlőelem az **ADOTable1** és **ADOTable2** szerepel.



4. ábra

- ADOConnection

Az **ADOConnection** vezérlőelem és az általa elérhető adatbázis közötti kapcsolatot a **ConnectionString** tulajdonság értéke írja le. Ezt a több paramétert tartalmazó karakterláncot eléggé körülményes dolog kézzel összeállítani. A Delphi-ben az összeállítást a **Microsoft Connection String** szerkesztője segíti.

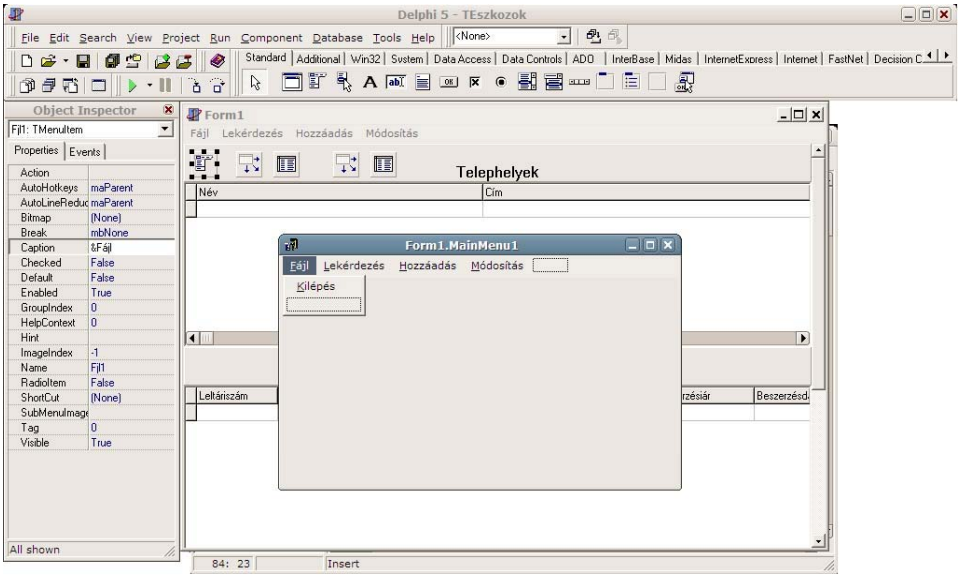
- Menü

A legtöbb alkalmazás menürendszert használ a különféle részfeladatok végrehajtásához. Az ablakmenü a program ablakának címsora alatt helyezkedik el, mely egy főmenüből és az abból nyíló legördülő menüből áll. A menü olyan speciális, csoportosított vezérlő halmaz, mely halmaz **TMenuItem** típusú elemeinek ablakban való megjelenítése kötött (fő vagy felbukkanó menü).

A menüelemek eseményei kezelik a felhasználó menüválasztását. Az **Item** tulajdonság tartalmazza a menüelemeket. A programban elhelyezett menüelemek legfontosabb tulajdonságai a **Name**, amely a komponenst azonosítja s a **Caption**, mely a megjelenő menücímet határozza meg. Ha a program futása során a felhasználó kiválaszt egy menüpontot, akkor annak **OnClick**

eseménye aktiválódik.

A főmenüt és a legördülő menüket a Delphi rendszer menütervezőjével szerkesztettem meg.



5. ábra

A főmenü négy részből áll: a Fájl menü, melynek egy Kilépés legördülő menüpontja van, a Lekérdezés, a Hozzáadás, melynek egy Telephely és egy Eszköz legördülő menüpontja van, és a Módosítás, melynek szintén Telephely és egy Eszköz legördülő menüpontjai vannak.

- DataSource

Ahhoz, hogy egy **Table** adatkészletező vezérlőelemek által képviselt adatokat elérhetővé tegyük, egy adatközvetítő objektumra – a **DataSource**-re – van szükség. A kapcsolat úgy épül fel, hogy a **DataSource** objektumot a **DataSet** tulajdonságán keresztül hozzákapcsoljuk egy adattároló komponenshez.

- Table

A **Table** vezérlőelem segítségével minden olyan adattáblára hivatkozhatunk, amelyet a BDE eszközeivel elérhetünk. Az elérni kívánt adatbázis logikai nevét a vezérlőelem

DatabaseName tulajdonságában kell megadni. A **TableName** tulajdonság értékét pedig az adatbázison belüli táblák egyikének a nevére kell beállítanunk. Ezeket a beállításokat a **Table** vezérlőelem Objektum-felügyelőjében állítottam be. Az adatokat a **Table** vezérlőelem **Open** metódusának hívása után érhetjük el, vagy a vezérlőelem **Active** tulajdonságának **true**-ra való beállításával. Az adatbázissal való kapcsolat megszüntetéséhez – amit mindig el kell végeznünk – meg kell hívnunk a **Close** metódust, vagy az **Active** tulajdonság értékét **false**-ra kell állítanunk.

Ahhoz, hogy a felhasználó módosítani tudja az adatbázisban tárolt adatokat, a **Table** vezérlőelem **ReadOnly** tulajdonságát **false**-ra kell állítani.

A **Table** vezérlőelem az adattábla egy csoportját is képviselheti, amely csoportot különböző szűrők segítségével definiálhatunk. A szűrő karakterláncot a **Filter** tulajdonság értékeként kell megadni, majd pedig engedélyezni kell a szűrést, a **Filtered** tulajdonság **true**-ra való állításával.

Ha a megjelenítendő adatok sorrendjén szeretnénk változtatni, akkor a **Table IndexName** tulajdonság értékeként a megfelelő index nevét kell beállítanunk.

- DBGrid

Az adatokat a formon elhelyezett vizuális adatvezérlők segítségével táblázatos (**DBGrid**), szövegmezős (**DBEdit**), feljegyzési (**DBMemo**), grafikonos (**DBChart**) formában jeleníthetjük meg.

A **DBGrid** az adatokat táblázatos formában jeleníti meg. Mint minden adatmegjelenítő vezérlőelemet a **DBGrid**-et is hozzá kell kapcsolni a **DataSource** tulajdonságán keresztül a megfelelő adatforrás komponenshez, amelyet előzőleg egy adatkészletező objektummal kapcsoltunk össze. Az adatkészletező elem aktiválása után az adatrácsban megjelennek a rekordok. A **DBGrid** típusú objektum **Options** halmaz típusú tulajdonságának használatán keresztül nem csak az adatok módosíthatóságát, hanem azoknak a megjelenítését is szabályozhatjuk. Például a *dgRowSelect* opció **true**-ra állításával nem csak az aktuális cella, hanem a teljes sor kiválasztható lesz.

A **DBGrid Columns** tulajdonságának szerkesztőjében beállíthatjuk az oszlopok tulajdonságait. Az oszlop létrehozása után először a **FieldName** tulajdonság értékét kell a megfelelő adatmezőre állítani. Majd a **Title** tulajdonság **Caption** tulajdonságának értékével Az

oszlopoknak saját nevet adhatunk.

- Label

A címke vagy felirat (**TLabel** típusú) vezérlő szöveg megjelenítésére szolgál. Ezért alapvető tulajdonságai a feliratok elhelyezkedésével és a szöveg megjelenítési módjának beállításával kapcsolatosak.

Fontosabb tulajdonságai közül megemlíteném, hogy

- ha a címke befogadó téglalapját a kívánt méretre széthúzzuk, a szöveget vízszintesen pozícionálhatjuk az ***Alignment*** tulajdonság értékével

- ha az ***AutoSize*** tulajdonságot *true*-ra állítjuk, a címke mérete automatikusan a megjelenítendő szöveg hosszához és magasságához igazodik

- a ***Caption*** tulajdonság a címke szövegét tartalmazza

- ***WordWarp*** tulajdonság *true* értéke lehetővé teszi, hogy a címke szövege több soros legyen, ha az érték *false* akkor a hosszú szöveg végét levágja.

2. Modulokra bontás

Az alkalmazás öt modulra lett felbontva. A fő program, mely a *Form1*-et is tartalmazza, a Fo nevet kapta. A *Hozzaadas* modul, mely az adattáblák bővítésével kapcsolatos műveleteket tartalmazza, egy külön formon helyezkedik el. A harmadik a *Lekerdezesek* modul, mely a lekérdezés műveletét valósítja meg. A negyedik a *Modositasok* modul, mely a Hozzaadas formjának felhasználásával végzi el a kijelölt rekordok módosítását. Utolsó a *Törlések* modul, mely az adattáblák rekordjainak törlését végzi.

Az alkalmazás Fo formjának **OnCreate** eseményének bekövetkezésekor állítódik be a Form1 **Caption** tulajdonsága, és a Table1 és Table2 vezérlőelemek **Open** metódusának hívása. Ezek után, a DBGrid1 megjeleníti a Table1 (TELEPHELY) adattábla adatait és a DBGrid2 a Table2 (ESZKOZOK) adattábla adatait. A Table komponensek **Close** metódusának hívása a Form1 OnClose eseményének bekövetkezésekor kerül végrehajtásra.

A Fo modul feladata még az összes – korábban leírt - menü esemény kezelése, melyek implementálása a műveleteknek megfelelő modulokban vannak.

Néhány esemény figyelése aktív még ezeken kívül.

Az egyik a DBGrid1-en történő dupla kattintás, mely meghívja a Lekerdezes modul Filter eljárását. Ekkor beállítódik a Table2 Filter tulajdonság értéke a Table1 ID mezőjének értékére.
Így a DBGrid2–ben csak azok a rekordok jelennek meg, amelyek Table2-beli ID értékükkel megegyeznek. A Lekérdezés menüpont választásakor a Filtered tulajdonság értéke false-ra állítódik, így az összes telephely összes eszközét megjeleníti.

Ha a DBGrid1-en van a kurzor és a Delete billentyű lett lenyomva, akkor a Torlesek modul TelephelyTorles eljárása lesz meghívva. Ekkor a TELEPHELY tábla aktuális rekordjának ID mezőjének értéke tárolódik, mert ellenőrzés történik arra vonatkozóan, hogy az ESZKOZOK táblában vannak-e olyan rekordok, melyek ID értéke megegyezik ezzel, vagy is létezik-e olyan eszköz az ESZKOZOK táblában, amely az aktuális telephelyhez tartozik. Ha igen akkor ezek törölve lesznek az adatbázisból, majd az aktuális telephely is törölve lesz a TELEPHELY táblából.

Ha a DBGrid2-ön van a kurzor és a Delete billentyű lett lenyomva, akkor a Torlesek modul EszkozTorles lesz meghívva. Ekkor az aktuális rekord törlődik az ESZKOZOK táblából.

A főmenü Hozzáadás menüpontjának elemei a Hozzaadasfrm **Visible** tulajdonságát **true**-ra állítva megjeleníti a formot.

6. ábra

A formon telephely és eszköz hozzáadásának komponensei egyaránt elhelyezkednek. A komponensek szétválasztását a komponensek **Tag** tulajdonság értékeivel oldottam meg. A telephely hozzáadásához tartozó komponensek Tag értékét fejlesztési időben 1-re állítottam, míg az eszközök hozzáadásához tartozó komponensek Tag értékét 2-re állítottam.
Ezután a form komponenseit egy ciklussal bejárva, ha a formot a Hozzáadás Telephely menüelem hívja meg, akkor azon komponenseket jeleníti meg a formon, melyek Tag értéke 1.
Ha a formot a Hozzáadás Eszköz menüelem hívja, akkor azok a komponensek jelennek meg, melyek Tag értéke 2. Ezzel egy időben a **HozzaCimke Caption** a Telephely hozzáadása, ellenkező esetben pedig az Eszköz hozzáadása értéket kapja.

Telephely hozzáadása

Telephely hozzáadásánál három **TEdit** típusú szövegszerkesztő szövegmezőt használtam.

Az **Edit** vezérlő, egysoros szövegszerkesztési feladatok ellátására, adatok beolvasására, információ kijelzésére használható. A (string típusú) **Text** a szövegszerkesztő alapvető tulajdonsága, amely a vezérlő által megjelenített szöveget tartalmazza.

A TELEPHELY tábla rekordjainak mezői: az ID egy számolt érték és a NEV, a CIM, a

TELEFON string típusúak, így könnyű egymáshoz rendelni az adatokat. A Hozzáadás gombra kattintás után ellenőrzés történik, hogy a NEV-nek és CIM-nek megfelelő mezők tartalmaznak e adatot és arra vonatkozólag, hogy a NEV-vel szerepel-e már telephely a táblában.

Továbbá a *LehetségesID* függvény kerül meghívásra, mely visszatérési értéke egy lehetséges ID, mely még nem szerepelt eddig a táblában.

Ha ezek teljesülnek, akkor szerkesztési módba állítja a **Table1** vezérlőelemet, majd hozzáad egy üres rekordot és beállítja mezőit az **Edit** vezérlők **Text** tulajdonságában lévő adatokra és meghívja a Table **Post** metódusát, mely kimenti a változtatásokat az adattáblába.

Eszköz hozzáadása

Amennyiben a menü Hozzáadás Eszköz menüpont **OnClick** eseménye lett kiválasztva, akkor a formon egy **ComboBox**, öt **Edit** és egy **DateTimePicker** vezérlő komponens kerül megjelenítésre.

- ComboBox

A **ComboBox** egy kombinált listavezérlő. A kombinált lista két vezérlő összeépítése, tulajdonságai és metódusai a **TEdit** és a **TListaBox** osztályoknak felelnek meg.

- **DateTimePicker**

A Win32 komponens paletta eleme a dátum és időpont beállítására szolgáló **DateTimePicker** komponens. A vezérlő egy legördülő listában a dátum, illetve az időpont mezőket tartalmazza.

A Hozzaadasfrm eszköz hozzáadásánál, a megfelelő komponensek megjelenítése után, bejárja TELEPHELY tábla összes elemét és a NEV mező értékeivel feltölti **ComboBox1**-et.
Ezek után, a legördülő listából kiválasztható a kívánt telephely, melyhez az aktuális eszközt szeretnénk hozzáadni. A **ComboBox1Change** esemény bekövetkeztekor egy *id* nevű egész típusú globális változó értéke beállítódik, az éppen aktuális telephely ID értékére. Így az eszköz az ESZKOZ táblázatba ennek megfelelő ID értékkel kerül be és rendelődik hozzá a telephelyhez.

A Hozzáadás gomb **OnClick** eseményének bekövetkezésekor, ellenőrzés történik, hogy a **ComboBox1**, az **Edit4 Text** és **Edit5 Text** tulajdonsága tartalmaz e bejegyzést.
Ha igen, akkor egy logikai értékkel visszatérő *LehetsegesLeltariszam* függvény kerül meghívásra, mely az **Edit7 Text** tulajdonságát kapja paraméterül. Ha a függvény true-val tér vissza, akkor az aktuális leltári szám még nem szerepel az ESZKOZ táblában.

Ezek után, **EConvertError** kivétel figyelésével ellenőrizzük, hogy az Edit5 és Edit6 Text tulajdonsága ténylegesen érvényes, a tábla aktuális mezőjének megfelelő típusú adatot tartalmaz-e.

Ha igen, akkor a Table2 szerkesztési módba állításával és új rekord hozzáadásával a megfelelő komponensek értékei hozzárendelődnek az aktuális rekord megfelelő mezőihez, majd az új rekord kimentése történik az adattáblába.

Módosítás

A Módosítás menüpont legördülő menüjének két eleme van. Módosítás Telephely és Eszköz. A módosításokat a Hozzaadasfrm használatával valósítottam meg. A form **HozzaCimke Caption** tulajdonságát Módosítás-ra és a **HozzaadBtn Caption** tulajdonságát Módosít-re állítottam.

Ha a Módosítás menü Telephely legördülő menüpontjának **OnClick** eseménye következett be, akkor a hozzáadásnál leírt módon a telephely hozzáadásánál használt komponensek jelennek meg.

Ezt követően, a komponensekben megjelennek az aktuális rekord pozíció által tartalmazott adatok.

A Módosít gombra történő kattintás után a TELEPHELY tábla szerkesztési módra állításával, a módosított adatokkal felülíródnak az adattábla mezői.

Ha a Módosítás menü Eszköz legördülő menüpontjának **OnClick** eseménye következett be, akkor az ESZKOZ tábla adatai kerülnek a megfelelő komponensekbe és kerülnek módosításra.

Azért ezt a módszerét választottam a módosításnak, mert az adatok típusának összeegyeztetését jobban kézben tarthatónak találtam ezzel a megoldással.

IV. Az alkalmazás felhasználói szempontból

1. Lekérdezés, törlés

Amikor a felhasználó elindítja az alkalmazást, a belépéshez, az adatbázis eléréséhez felhasználói név és jelszó megadása szükséges. (Melyek megadásáról az előző fejezetben már írtam.)

Ezek után láthatóvá válik a Telephelyek és az Eszközök tábla, valamint a főmenü. A Telephelyek adatmegjelenítő komponensben találhatók a TELEPHELY tábla rekordjai, melyek mezői a következők: név, cím, telefonszám. Az Eszközök adatmegjelenítő komponensben találhatók a ESZKOZ tábla rekordjai, melyek mezői: leltári szám, eszköz, mennyiség, beszerzési ár, beszerzés dátuma és megjegyzés.

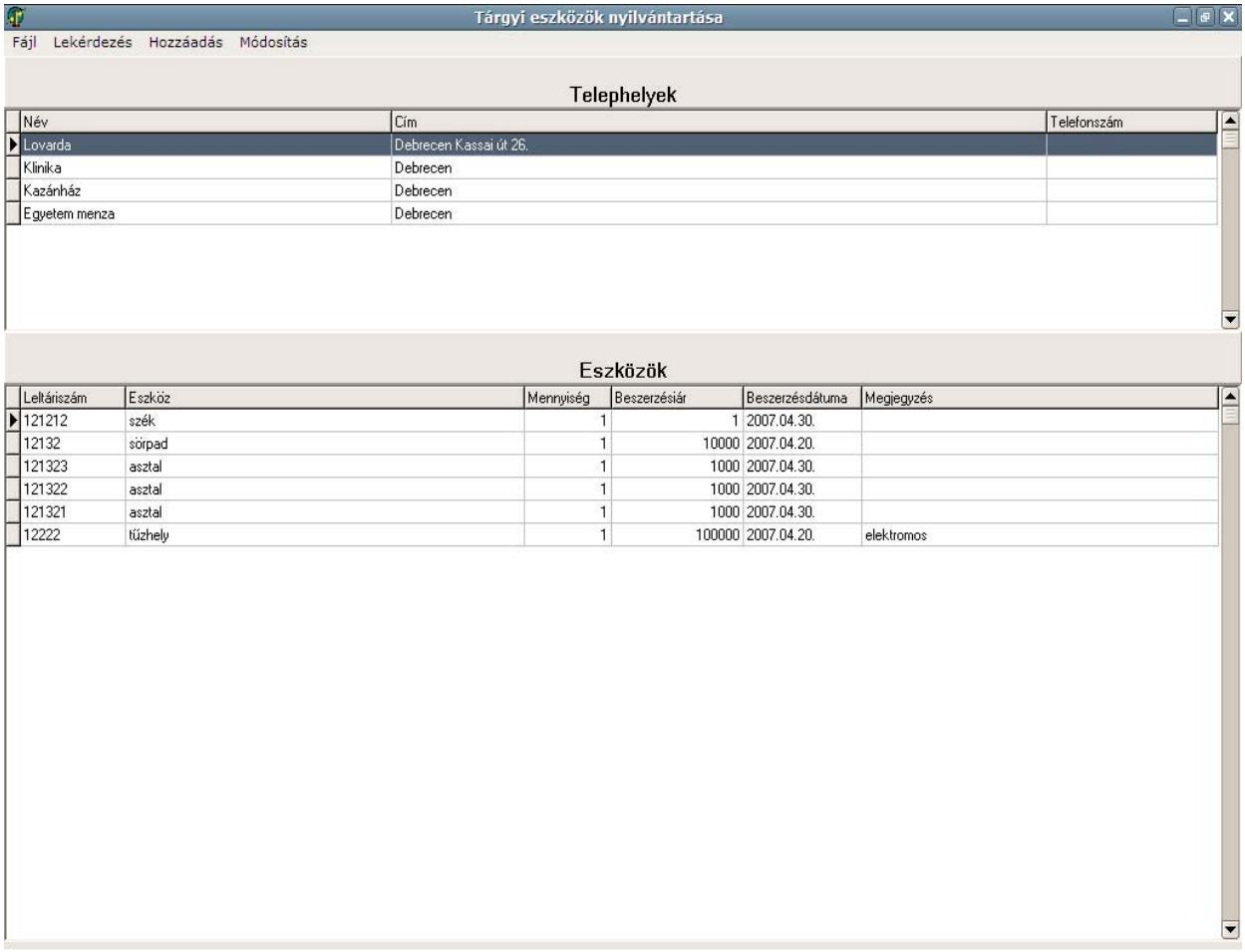
A TELEPHELY tábla aktuális rekordjára duplán kattintva, az alatta lévő ESZKOZ táblában megjelenítődnek a hozzá tartozó tárgyi eszközök és a hozzájuk tartozó elemek.

A Lekérdezés menüpont kiválasztásakor az összes telephely összes eszköze megjeleníthető.

Felhasználói szempontból ez előnyös megoldásnak bizonyul, mert ha csak egyetlen telephely eszközeire vagyunk kíváncsiak, akkor lehetőségünk van azt a fent leírt módon elérni. Ugyanakkor az összes telephely összes eszközeinek megjelenítésére is lehetőségünk van.

Ha a TELEPHELY táblán ki van jelölve valamelyik telephely, az aktuális rekordot egy Delete gombbal egyszerűen törölhetem. Ilyenkor az adott telephelyhez tartozó ESZKOZ táblában lévő összes hozzá tartozó tárgyi eszköz is törlésre kerül. (Ez azért fontos, mert ha esetleg megszűnik egy telephely, akkor az adatbázisban nem maradnak olyan tárgyi eszközök, amelyek már nincsenek egyik telephelyhez hozzárendelve.)

Ha az ESZKOZ táblán szintén az eszköz kijelölésekor Delete gombot nyomunk, az adott eszközt és a hozzá tartozó adatokat törli ki, tehát az egész rekordot.


Tárgyi eszközök nyilvántartása
Fájl Lekérdezés Hozzáadás Módosítás
Telephelyek
Név Cím Telefonszám
Lovarda Debrecen Kassai út 26.
Klinika Debrecen
Kazánház Debrecen
Egyetem menza Debrecen
Eszközök
Leltáriszám Eszköz Mennyiség Beszerzésiár Beszerzésdátuma Megjegyzés
121212 szék 1 1 2007.04.30.
12132 sörpad 1 10000 2007.04.20.
121323 asztal 1 1000 2007.04.30.
121322 asztal 1 1000 2007.04.30.
121321 asztal 1 1000 2007.04.30.
12222 tűzhely 1 100000 2007.04.20. elektromos

7. ábra

2. Hozzáadás

A főmenüben kiválasztható másik művelet a Hozzáadás. A felhasználónak itt lehetősége van aszerint választani, hogy a telephelyet vagy az eszközöket kívánja bővíteni. A legördülő menüben ennek megfelelően arra a menüpontra kell kattintania, amelyiket használni szeretné.

A hozzáadás a következőképpen történik. Ha telephelyet akar hozzáadni, a megjelenő ablakban értelemszerűen be kell írnia a megfelelő adatokat a megfelelő mezőkbe. Eszerint meg kell adnia a Telephely nevét, Címét, telefonszámát, és a Hozzáad gombbal már bővítette is az adattáblát.

8. ábra

Hasonlóképpen kell eljárni az eszköz hozzáadás esetén is. A megjelenő ablakban a Telephely nevét egy legördülő listából lehet kiválasztani. Az adott helyekre be kell írnia az eszköz leltári számát – mely korábban még nem szerepelt, magát az eszközt, a mennyiséget, beszerzési árát forintban megadva, a beszerzés dátumát – melyet szintén legördülő menüből választhat a felhasználó, valamint az eszközhöz tartozó esetleges megjegyzéseket. Az ablakot az alábbi ábra szemlélteti.

9. ábra

Mindkét esetben a Hozzáadás gombra kattintva van lehetőség a beírt adatok rögzítésére az adattáblában.

3. Módosítás

Az adattábla módosítása is egyszerűen történhet. A hozzáadáshoz hasonlóan itt is a főmenü Módosítás menüpontban választhatja ki a felhasználó a kívánt módosítandó eszközt vagy telephelyet.

A menüpont kiválasztásakor az aktív rekord mezői jelennek meg az ablak megfelelő beviteli mezőjében.

A felhasználónak lehetősége van a Telephely nevét, Címét és Telefonszámát módosítani úgy, hogy egyszerűen csak át kell írnia a benne szereplő vagy hiányzó adatokat.

Majd a Módosít gombra kattintva az adatok módosítása megtörténik.

10. ábra

Ugyanígy járhat el a felhasználó, ha az eszközöket szeretné módosítani.

Itt a legördülő Telephely listából kiválaszthatja a telephely nevét, ha esetleg oda szeretné áthelyezni az eszközt. Így a hozzátartozó adatok a másik telephelyhez kapcsolva változatlanul áthelyezésre kerülnek.

Ugyanitt lehetősége van a felhasználónak, hogy módosítsa az eszköz adatainak bármelyikét. Mindkét esetben a Módosít gomb aktiválásával véglegesíthetők a módosított adatok.

11. ábra

4. Az alkalmazásról alkotott vélemény felhasználói szempontból

Az alkalmazás fejlesztése során kitűzött célt, a könnyen használhatóságot, ugyanakkor a problémamentes működést az általam készített alkalmazás elérte, hiszen nagyon egyszerűen, bárki számára könnyen használható. Lehetőséget ad a gyors és pontos munkára, hiszen az adatok bevitelében minimális a hibalehetőség: a nem megfelelő adattípusok bevitele esetén a program hibaüzenettel jelzi az esetleges hibát, a nem megfelelő adatot. (Például, ha a darabszám mezőkbe nem számot ír a felhasználó, akkor a következő hibaüzenetet küldi: 'A mennyiségnél megadott érték nem szám!') Éppen ezért megkönnyíti a leltárt végző emberek munkáját.

Összefoglalás

Dolgozatomban az általam fejlesztett tárgyi eszköz-nyilvántartó programot mutattam be. A program fejlesztéséhez kétféle Delphi verzió két különböző adatbázis vezérlőelemét használtam. Az egyiket Borland Delphi 5 BDE, a másikat Borland Delphi 6 ADO felhasználásával. A fejlesztés során magam is tapasztaltam a két módszer közötti különbséget. Mindkét módszernek vannak előnyei és hátrányai a fejlesztés szempontjából, használata mellett szólnak érvek és ellenérvek.

Egyik fontos különbség a hordozhatóságban van. Az ADO Microsoft fejlesztés. A MS OLE DB adatelérési technológián alapul, amely elérést biztosít relációs és nem-relációs adatbázisokhoz, levelezési és fájlrendszerekhez, és egyedi üzleti objektumokhoz. Nem igényli a BDE könyvtárait, de szükségesek hozzá az ADO/OLE DB futás idejű csomagjai (Windows 2000-től felfelé).

Ha a BDE megközelítést választjuk az adatbázis eléréséhez, akkor mindenféleképpen telepítenünk kell a már megírt adatbázis-kezelő alkalmazásunk mellé a BDE-t is. A Delphi különféle licenszei foglalkoznak a BDE terjeszthetőségével - "elvileg" szabadon terjeszthető.

Az ADO ehhez képest nem igényel semmilyen új telepítést.

Mivel COM szerverként DLL-ből töltődik be, így a kliens alkalmazás címterébe futhat, ez teljesítmény szempontjából kifejezetten előnyös. Segítségével többrétegű alkalmazásoknál jelentősen csökkenthetjük az adatforgalmat a BDE-hez képest.

Az ADO beszerzése több helyről is lehetséges. Egyrészt a Windows 2000 óta az operációs rendszer része, másrészt az Office programcsalád telepítőjének is része, harmadrészt a Microsoft oldaláról is ingyenesen letölthető a legújabb verziója.

Az alkalmazás fejlesztése közben számomra egyszerűbbnek bizonyult a BDE adatvezérlő elemek adatbázishoz való kapcsolása, mint az ADO vezérlőelemeké. Míg a BDE vezérlőelemek esetében elegendőek voltak az adatbázis logikai nevének megadása a **DatabaseName** tulajdonságban és **TableName** tulajdonság értékeként megadni a adatbázison belüli tábla nevét, addig az ADO **ADOConnection** vezérlőelemnél, egy jóval bonyolultabb feladat volt a **ConnectionString** előállítása.

Úgy gondolom, hogy az általam fejlesztett alkalmazás elérte célját, hiszen a felhasználók számára könnyen megérthető és egyszerűen használható tárgyi eszköz-nyilvántartó program készült.

Irodalomjegyzék

1. **Programozzunk Delphi 5 rendszerben! / Tamás Péter et al. – ComputerBooks: Budapest, 2001.**
2. **http://www.softwareonline.hu/art905/tcustomadodataset+objektum.html**
3. **http://www.prog.hu/cikkek/895/Access+adatbazis+hasznalata.html**