Laborator 11 – Accesul la baze de date în C++ Builder

Tema 11.1

Să se realizeze 2 tabele în Paradox una cu numele „studenti.db” și una cu numele „note.db” care să conțină următoarele câmpuri:

Tabela	Nume câmp	Tip câmp	Dimensiune câmp	cheie
Studenti	ID	numeric	-	Da
	Nume	alfanumeric	50	
	sgr	alfanumeric	5	
Note	ID	numeric	-	Da
	Materia	alfanumeric	30	Da
	Nota	real ($)		

Considerații teoretice 11.1

Baza de date reprezintă o colecție de informații organizate în tabele în care fiecare coloană din tabelă memorează un anumit tip de informație. Se preferă utilizarea bazelor de date în locul unor fișiere simple deoarece bazele de date oferă instrumente pentru colectarea, căutarea și organizarea acestor informații. Astfel oferă funcții utile de stocare facilă a datelor, de căutare rapidă în acestea și de modificare ușoară a acestora. Există două tipuri mari de baze de date: un tip este acela în care baze de date se găsește într-un singur fișier (toate tabelele sunt într-un singur fișier pe disk - problema este că fișierul devin imense și greu de gestionat) și un alt tip de baze de date este acela în care se salvează fiecare tabelă într-un fișier separat pe disk.

Aplicația Database Desktop

Pentru lucru cu baze de date firma Borland oferă suport pentru baze de date de tip Paradox. În meniu „Borland C++ Builder 6” avem o aplicație simplă care știe să lucreze cu astfel de baze de date numită „Database Desktop”. Această aplicație permite crearea de tabele simple în Paradox, popularea și vizualizarea acestora.

Se pornește aplicația DatabaseDesktop de pe disk (se găsește în meniul de unde se pornește și aplicația C++ Builder). Pentru crearea unei tabele se poate folosi meniul ”File->New->Table”, unde se specifică tipul tabelei (în cazul nostru Paradox). În acest moment se creează antetul (hederul) tabelului cu numele fiecărei coloane din tabele și cu tipul datelor care vor fi memorate în acea tabelă. Descrierea câmpurilor:

- *FieldName* - se specifică numele coloanei din tabel care se dorește realizată;
- *Type* – se specifică tipul datelor care pot fi salvate în acea coloană (baza de date nu va permite salvarea altui tip de date decât cel specificat). Pentru a alege tipul de date dorit în acest câmp se poate utiliza tasta „Space” sau click dreapta de mouse. Dintre tipurile care sunt permise in Paradox detaliem:
  - *Alpha* – șir de caractere alfa numerice;
  - *Number* – numere întregi în domeniul -10^{307} - 10^{308};
  - *$(Money)* – numere reale (cu virgulă);
  - *Short* – număr întreg reprezentat pe 16 biți;
  - *Long Integer* – număr întreg reprezentat pe 32 de biți;

  - …..
- *Size* – unele tipuri de date necesită specificarea dimensiunii care va reprezenta numărul maxim de caractere care pot fi salvate în câmpul respectiv (lățimea colonei respective). Orice informație care depășește lungimea specificată va fi trunchiată automat;
- *Key* – permite specificarea faptului că vrem ca baza de date să verifice automat astfel încât pe acea coloană să nu apară valori duplicat. Astfel în momentul în care încercăm să salvăm o nouă înregistrare (linie) în tabela noastră baza de date va verifica automat câmpurile marcate cu key și nu va permite salvarea înregistrării dacă toate câmpurile marcate ca fiind câmpuri key sunt identice. Pentru a specifica câmpul key se poate folosi tasta „Space" sau click dreapta de mouse (va apare o steluță în dreptul câmpului marcat cu key) .

În momentul în care s-a terminat introducerea datelor despre tabelă se poate salva tabela într-un fișier folosind butonul „ Save As..". Atenție să salvați ambele tabele în același folder deoarece aplicația Paradox consideră salvează fiecare tabel în fișier separat. Pentru introducerea unor date în tabelă (înregistrări) se va deschide tabela creată folosind meniul „File->Open->Tabel" iar după deschiderea tabelei din meniu se selectează „Table->Edit Data".

Aplicația BDE Administrator

Borland Database Engine (BDE)

Pentru accesul facil la fișierele de baze de date firma Borland pune la dispoziție un motor de acces la baze de date numit BDE prin intermediul căreia aplicația creată în C++ Builder nu depinde de tipul bazei de date și de locul unde este salvată baza de date. Astfel pentru accesul la fișierele unei baze de date din aplicație este nevoie de crearea unui așa zis *Alias* care oferă două facilități mari:

- Face o legătură între un nume fix înregistrat la nivel de sistem de operare și o cale undeva pe hard disc. Astfel dacă se mută fișierele bazei de date în altă parte pe hard disk sau altundeva în rețea nu trebuie modificată aplicația ci se poate modifica ușor doar aliasul modificând calea către fișiere;
- Accesul la baze de date diferite. În momentul în care se creează aliasul la acesta trebuie specificat tipul bazei de date pe care vrem să o deschidem. În momentul acesta în alias se va atașa driver-ul pentru baza de date respectivă astfel că în aplicația care o vom dezvolta vom lucra cu tabelele din bazele de date la fel, indiferent în ce au fost construite acestea.

Pentru a crea un alias avem nevoie de o aplicație care să permită inserarea acestuia la nivel de sistem de operare. Pentru aceasta C++ Builder vine cu aplicația „BDE Administrator". Porniți aplicația BDE Administrator din meniul unde se găsește și aplicația Builder C++. În aplicația BDE Administrator pentru adăugarea unui nou alias de folosește meniul „Object->New". În momentul în care se solicită specificarea driverului pentru baza de date de tip Paradox (cum avem noi) se selectează cuvântul *Standard* din listă. În listă apar toate driverele de baze de date care se găsesc instalate pe calculator și pe care le recunoaște aplicația. Pentru alias trebuie specificate următoarele caracteristici:

- *Numele* – se va specifica numele dorit pentru alias (în loc de cuvântul Standard1);
- *Path* – se specifică calea către folderul în care se găsesc salvate tabelele din baza de date. În momentul în care se face click în fereastra albă din dreptul cuvântului *Path* va apărea un buton în partea dreaptă care ne permite alegerea căi dorite;
- *Default driver* – se specifică driverul pentru baza noastră de date (implicit Paradox);

Pentru salvarea alias-ului creat se folosește meniul „Object->Apply".

Indicații 11.1

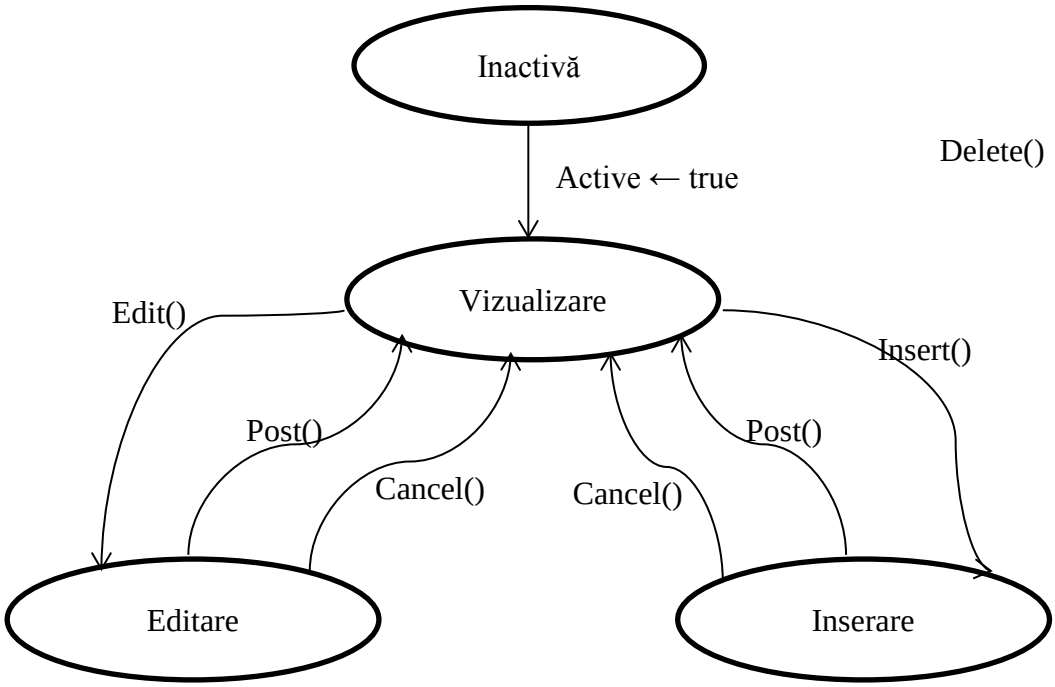
- ⇨ În timpul introducerii datelor în tabelă atenție la ceea ce scrie în partea de jos-stânga a ferestrei. Aplicația Database Desktop va afișa mesajele de eroare doar în acel loc.
- ⇨ Atenție să nu uitați să vă faceți un alias pentru accesul la baza de date creată. Aplicația C++ Builder încarcă lista de alias-uri la pornire, de aceea dacă nu vedeți aliasul tocmai creat reporniți aplicația C++ Builder.

Tema 11.2

Creați o aplicație în C++ Builder care permite afișarea informațiilor din tabela studenti.db pe o formă într-o componentă de tip DBGrid.

Considerații teoretice 11.2

În timpul rulării unei aplicații, o bază de date se poate găsi în mai multe stări (Inactivă, Vizualizare, Editare, Inserare). În funcție de starea în care se găsește baza de date se pot executa sau nu anumite comenzi asupra bazei de date (Comenzi: Edit, Insert, Delete, Post, Cancel) . Stările în care se poate găsi baza de date sunt prezentate în figura următoare:



Pentru a trece o tabelă din starea de inactivă (inaccesibilă) în starea în acare aceasta poate fi citită trebuie activat driverul pentru a avea acces la tabelă. În momentul în care dorim editarea (modificarea) unei linii din tabela respectivă se poate folosi comanda „Edit()”. Dacă dorim salvarea informațiilor modificate se poate folosi comanda Post() iar dacă dorim anularea modificărilor se poate folosi comanda Cancel(). Starea de Inserare înseamnă starea în care se introduce o linie nouă în tabelă în locul în care ne găsim. Pentru a trece în starea de introducere a unei noi linii în tabelă se folosește comanda Insert(). La fel pentru salvarea înregistrării noi introduse se folosește comanda Post(). Pentru ștergerea unei lini (înregistrări) din tabelă se folosește comanda Delete().

Componente care permit accesul la Baze de date

În C++ Builder pentru accesul din cadrul unei aplicații la o bază de date trebuiesc folosite 3 componente astfel (atenție la ordinea in care trebuie configurate componentele):

1. O componentă din tabul BDE care permite accesul la tabelă. Aceste componente sunt componente nevizuale care permit doar configurarea accesului la tabele. De exemplu dacă se alege componenta ***Tabele*** aceasta va permite utilizarea comenzile specificate mai sus pentru accesul la baza de date. Pentru această componentă trebuie realizate următoarele configurări (atenție la ordinea in care trebuie făcute configurărire):
    a. *DatabaseName* – se alege aliasul creat înainte;
    b. *TableName* – se alege numele tabelei care se dorește a fi deschisă;
    c. *Active* – se specifică dacă se activează sau nu accesul la acea tabelă. În momentul în care Active trece pe *true* tabela noastră trece din starea *Inactivă* în starea de *Vizualizare*.
2. O componentă din tabul *DataAcces.* Aceste componente sunt tot componente nevizuale care permit distribuirea datelor dintr-o tabelă la mai multe componente vizuale. De exemplu dacă se alege componenta *DataSource* pentru această componentă trebuie completat câmpul *DataSet* cu numele componentei de tip BDE introdusă la pasul anterior.
3. O componentă din tabul *Data Controls*. Acest tab conține componente vizuale care permit afișarea elementelor din baza de date. Pentru aceste componente trebuie să se specifice proprietatea *DataSource* unde se alege componenta specificată la pasul anterior.

Componente din tabul Data Controls

DBGrid – este o componentă care permite vizualizarea tuturor coloanelor dintr-o tabelă;

DBText – este o componentă care permite afișarea informație de pe linia curentă din tabelă și de pe coloana specificată. Pentru această componentă pe lângă proprietatea *DataSource* trebuie specificată și proprietatea *DataField* (este asemănătoare cu componenta de tip Label);

DBEdit – este o componentă asemănătoare cu DBText doar că permite și modificarea conținutului afișat. Atenție că dacă se dorește modificarea textului afișat în această componentă atunci tabela din baza de date trebuie trecută în prealabil în starea de editare folosind comanda Edit();

DbNavigator – este o componentă care ne pune la dispoziție butoane pentru a naviga în toate câmpurile din tabel. Implicit sunt afișate toate butoanele atât pentru adăugare cât și pentru ștergere și editare înregistrări din tabelă. Pentru a putea configura care butoane sunt vizibile și care nu sunt vizibile la un moment dat din această componentă se poate modifica proprietatea *VisibleButtons* specificându-se false în dreptul butonului care nu se dorește a fi afișat.

Orice componentă vizuală din C++ Builder permite configurarea astfel încât să afișeze un mic mesaj despre ce face butonul respectiv în dreptul cursorului de mouse în momentul în care utilizatorul stă cu mouse-ul deasupra componentei. Pentru a activa această caracteristică se modifică proprietatea *ShowHint*. În cazul acestei componente care conține mai multe butoane se poate specifica textul care va fi afișat pe fiecare buton în parte, în ordinea în care sunt butoanele afișate, modificând proprietatea *Hints.*

Componente din tabul BDE

Componenta *Table* – permite conectarea la o tabelă dintr-o bază de date. În momentul când proprietatea *Active* a acestei componente trece din *false* în *true* (și se poate face această trecere) baza de date este deschisă automat și trece în starea de vizualizare. Dacă driverul selectat nu reușește să deschidă baza de date specificată nu se va putea trece această proprietate din false în true. Pentru a trece tabela din baza de date și în celelalte stări în cod se pot folosi următoarele comenzi (considerăm că pentru componenta Table am modificat numele acesteia în TableStudenti).

TableStudenti->Edit() –permite trecerea tabelei în starea de editare. Se va putea edita înregistrarea curentă;

TableStudenti->Post() – permite salvarea în tabelă a modificărilor făcute pe înregistrarea curentă;

TableStudenti->Cancel() – permite anularea modificărilor făcute în tabelă;

TableStudenti->Delete() – permite ștergerea înregistrării (liniei) curente din tabelă;

TableStudenti->Append() – permite adăugării unei noi înregistrări la sfârșitul tabelei;

TableStudenti->Insert() – permite adăugarea unei noi înregistrări pe linia curenta. De exemplu dacă tabela noastră are 100 înregistrări iar noi la un moment dat ne aflăm pe înregistrarea 10 și apelăm Insert atunci toate înregistrările de după linia 10 (inclusiv linia 10) se vor incrementa cu unu iar linia 10 devine linie goală, linie pe care va fi scrisă noua înregistrare;

TableStudenti->FieldByName("ID")->AsInteger – permite citirea de pe înregistrarea curentă a unui anumit câmp și convertirea acestuia la tipul de date specificat.

Componenta *Query* permite lucru cu tabelele din baza de date folosind limbajul SQL (Structured Query Language) care este unul dintre cele mai puternice și folosite limbaje pentru interogarea bazelor de date. Acesta este un limbaj neprocedural și declarativ deoarece utilizatorul descrie ce date vrea să obțină fără să fie nevoie să stabilească modalități de a ajunge la datele respective. Nu poate fi considerat un limbaj de programare ci intră în categoria limbajelor de aplicații fiind orientat pe lucru cu date.

Comenzi SQL

Comanda de selecție este una dintre cele mai folosite. Sintaxa comenzii este:

```
SELECT listă_câmpuri FROM listă_tabele [WHERE câmpul_selecție [ORDER BY câmpuri_criteriu]]
```

Listă_câmpuri – specifică câmpurile care se doresc a fi selectate din tabelă cu virgulă între ele. Pentru a selecta toate câmpurile se poate folosi *;

Listă_tabele – specifică tabelul sau tabelele de unde se dorește selectarea datelor;

Câmpu_selecție – se poate specifica ce caracteristică trebuie să îndeplinească câmpul/câmpurile specificate pentru a putea fi selectate. Această parte din comandă este opțională.

Câmpuri_criteriu – se poate specifica criteriul / criteriile după care pot fi ordonate datele selectate.

Comanda de inserare într-o tabelă:

```
INSERT INTO nume_tabel (listă_câmpuri separate prin ,) VALUES (listă_valori data în ordinea lisei câmpurilor);
```

Comanda de ștergere dintr-o tabelă:

```
DELETE FROM nume_tabel WHERE condiție stergere;
```

ATENȚIE: această instrucțiune șterge întreaga înregistrare. Nu uitați să specificați condiția de ștergere care de obicei este o cheie primară

Comanda de modificare a unor câmpuri într-o tabelă:

```
UPDATE nume_tabel SET nume_camp1=noua_valoare1, nume_camp2=noua_valoare2 ...WHERE condiție modificare;
```

ATENȚIE: Nu uitați să specificați condiția de modificare care de obicei este o cheie primară. Altfel se va modifica întregul set de date.

Pentru mai multe date despre limbajul SQL recomand http://ro.wikipedia.org/wiki/SQL și http://www.sqlcourse.com/index.html.

Tema 11.3

Adăugați la aplicație și o componentă de tip DBNavigator care să permită parcurgerea tuturor înregistrărilor din tabelă precum și adăugarea și modificarea acestora.

Considerații teoretice 11.3

A se vedea secțiunea „**Componente din tabul Data Controls**” de la considerațiile teoretice 11.2.

Tema 11.4

Introduceți în aceeași aplicație, pe aceeași formă și un DBGrid care să afișeze în paralel cu cel existent datele din tabela note.db.

Considerații teoretice 11.3

Se vor repeta pașii executați pentru tabela studenții și care au fost prezentați în secțiunea „**Componente care permit accesul la Baze de date**” de la considerațiile teoretice 11.2.

Tema 11.5

Să se modifice aplicația astfel încât în momentul în care selectăm un student din ”DBGrid-ul Studenti” să vedem în „DBGrid-ul Note” doar notele acelui student.

Considerații teoretice 11.5

Componenta de tip Table permite realizarea de conexiuni între mai multe tabela. Astfel la componenta Table care accesează tabela Note se va completa și câmpul „MasterSource” (din fereastra ObjectInspector) prin specificarea numelui componentei Table care accesează tabela

Studenti. În câmpul „MasterFields” se va adaugă o legătură (un Join) între ID din prima tabelă și ID din ce-a de-a doua tabelă prin selectarea butonului ADD.

Tema 11.6

Adăugați în aplicație un buton pe care se afișează textul „Adăugare Student” și care la apăsare să deschidă o nouă formă în care, în câmpuri de editare specifice, să putem adăuga datele pentru un nou student.

Considerații teoretice 11.6

Pentru adăugarea unei noi înregistrări în baza de date trebui trecută aceasta în starea de Inserare. Comanda pentru a executa acest lucru este:

```
fMain->TableStudenti->Append();
```

Pe noua formă se recomandă introducerea de componente de tip DBEdit legate de fiecare coloană din baza de date astfel încât să putem scrie direct în tabelă datele noastre.

După ce am introdus datele, dacă dorim să memorăm în tabelă datele specificate trebuie să le salvăm utilizând comanda:

```
TableStudenti->Post();
```

Tema 11.7

Adăugați în aplicație următoarele caracteristici:

- un buton care afișează textul „Modificare Student” și care la apăsare să deschidă o nouă formă în care să putem modifica datele studentului curent selectat;
- un buton care afișează textul „Sterge Student” și care la apăsare să putem șterge studentul curent selectat;
- un buton care afișează textul „Inserare Student” și care la apăsare să deschidă o nouă formă în care să putem adăuga datele unui nou student care va fi adăugat in tabelă la sfârșitul acesteia;
- un buton care afișează textul „Adaugă Nouă” și care la apăsare să deschidă o nouă formă în care să putem adăuga notele pentru un student.
- un buton care afișează textul „Modificare Notă” și care la apăsare să deschidă o nouă formă în care să putem modifica nota curent selectată în DBGrid.
- un buton care afișează textul „Șterge Nota” și care la apăsare să permită ștergerea înregistrării curente din tabela Note.

Considerații teoretice 11.6

Pentru rezolvarea temei se recomandă citirea considerațiilor teoretice de la 1.2 secțiunea **Componente din tabul BDE, componenta Table.**