CSYSTEMS™

PROGRAMSKI PAKET ZA KNJIGOVODSTVO

COBA Systems

Alaska Xbase++
DBF to PostgreSQL database application

(PART 3)
PostgreSQL Database with DBF Upsize tables

20.10.2023

UVOD
INTODUCTION

U delu 1 ove knjige opisan je PGDBE (PostGreDatabaseEngine) za rad Alaska Xbase++ aplikacija sa PostgerSQL databazom. Opisana je i UPSIZE tehnologija koja se može primeniti uz PGDBE, tako da se sa PostgreSQL databazom može komunicirati putem ISAM komandi i funkcija i putem SQL komandi i funkcija. Navedena su 3 programa koji se koriste u COBA Systems poslovnim aplikacijama za rad sa PostgreSQL bazom podataka.

Part 1 of this book describes PGDBE (PostGreDatabaseEngine) for the operation of Alaska Xbase++ applications with the PostgerSQL database. UPSIZE technology, which can be applied with PGDBE, is also described, so that it is possible to communicate with the PostgreSQL database through ISAM commands and functions and through SQL commands and functions. There are 3 programs used in COBA Systems business applications for working with the PostgreSQL database.

U delu 2 ove knjige je opisan i dokumentovan prvi od ta tri programa:
In part 2 of this book, the first of those three programs is described and documented:

C-DBF2SQLFILE.EXE (developer service program)

U delu 3 ove knjige opisan je i dokumentovan je program
Part 3 of this book describes and documents the program

C-POSTRGRESQL-DATABASE.EXE (supplementary program with each EXE application)

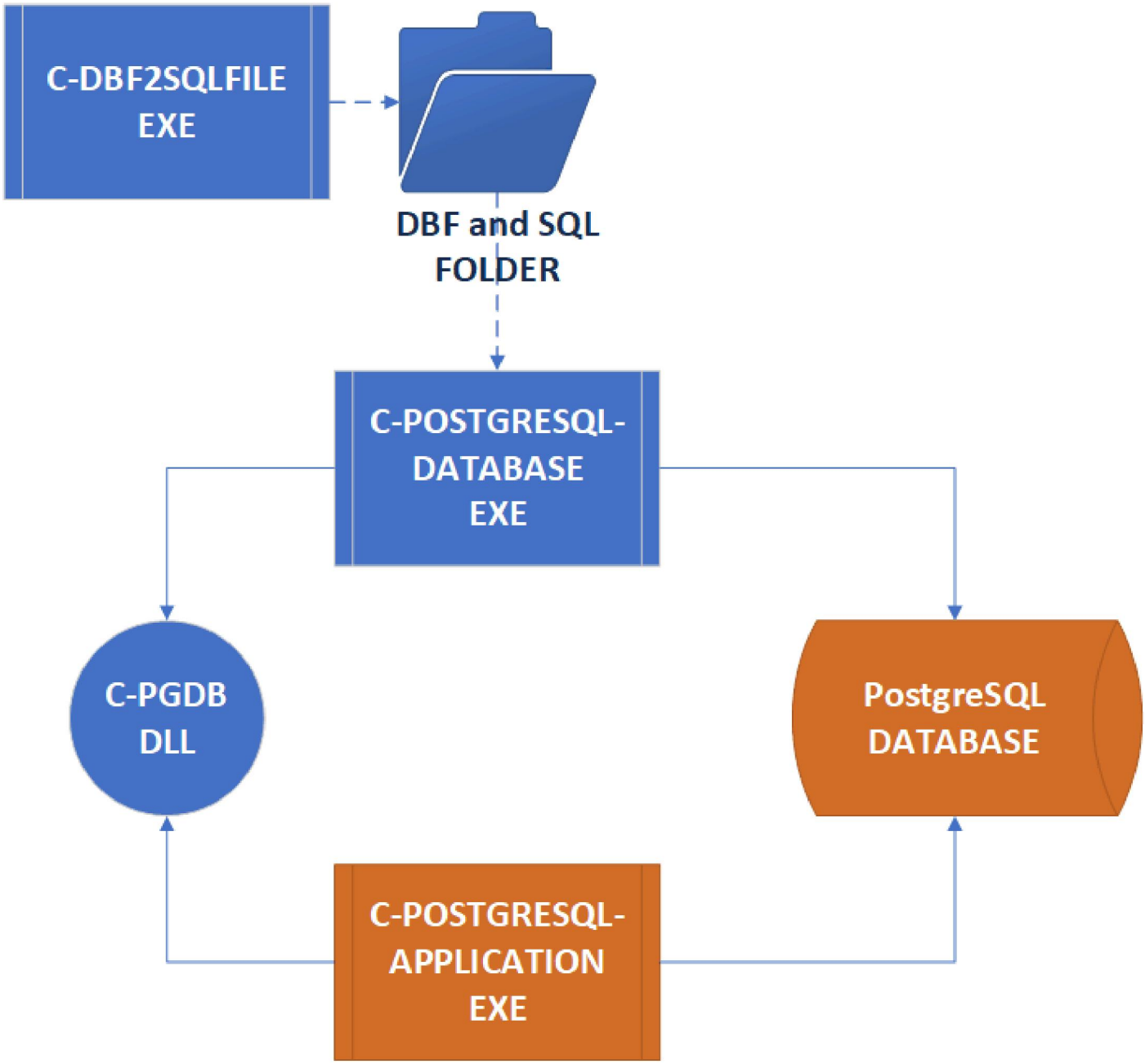
Ovaj program se isporučuje uz postgreSQL bazu podataka svake poslovne EXE aplikacije i služi za formiranje i održavanje te konkretne baze podataka.
Zbog posebnih zahteva u radu sa PostgreSQL bazom podataka i DBF Upsize - SQL tablama, bolje je rešenje da ovaj program bude posebna externa EXE aplikacija, od rešenja u kome je on sasatavni deo EXE aplikacije.
Osim (samo jedne) svoje osnovne funkcije, kojom se formira lista svih tabli baze podataka konkretne EXE aplikacije i SQL stringovi za kreiranje svake od tih tabli, ovaj program koristi sve ostale funkcije iz biblioteke C-PGDB.DLL.

This program is supplied with the postgreSQL database of every business EXE application and is used to create and maintain that particular database.
Due to special requirements in working with the PostgreSQL database and DBF Upsize - SQL tables, it is a better solution for this program to be a separate external EXE application, than a solution in which it is an integral part of the EXE application.
Apart from (just one) its basic function, which forms a list of all database tables of a specific EXE application and SQL strings for creating each of those tables, this program uses all other functions from the C-PGDB.DLL library.

U delu 4 ove knjige biće opisan i dokumentovan program
Part 4 of this book will describe and document the program
C-PGDB.DLL (common procedures and functions for applications)

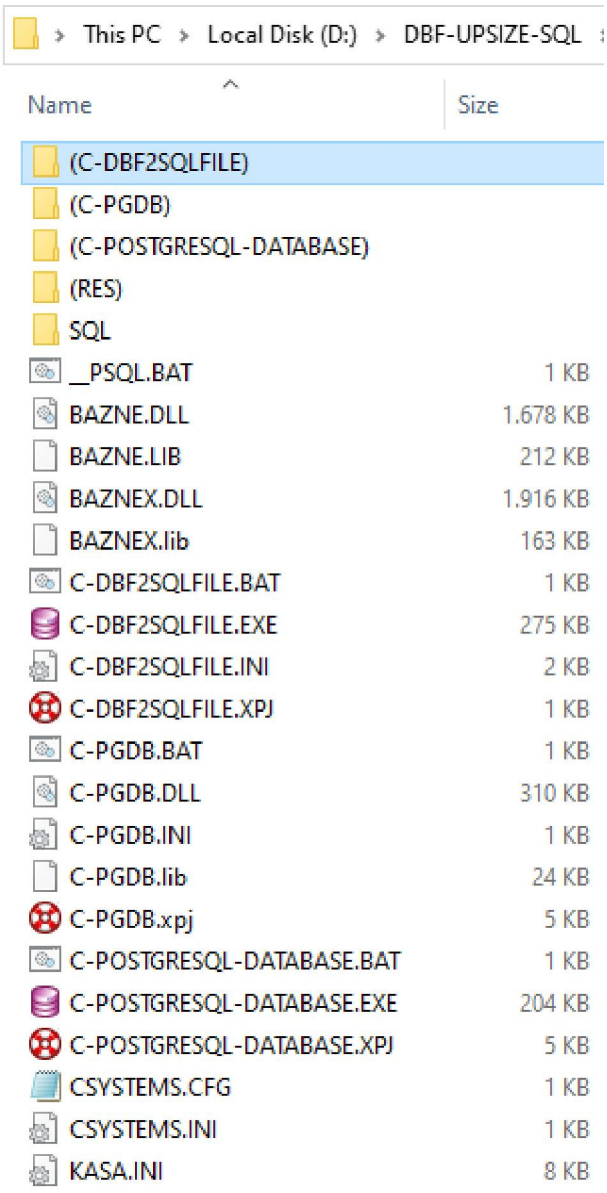
PRILOG: ŠEMA ORGANIZACIJE PROGRAMA ZA RAD SA UPSIZE POSTGRESQL DATABAZOM

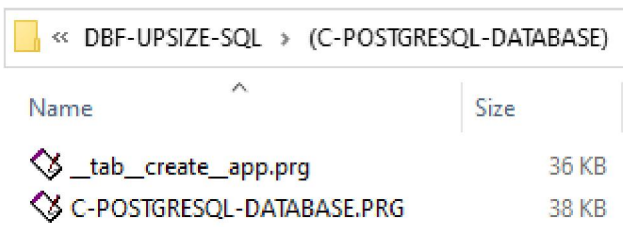
PROGRAM ORGANIZATION SCHEME FOR WORKING WITH THE UPSIZE POSTGRESQL DATABASE



ORGANIZACIJA PROJEKTA: Alaska Xbase++ To PostgreSQL

C-DBF2SQLFILE.EXE (developer service program)
C-POSTRGRESQL-DATABASE.EXE (supplementary program with each EXE application)
C-PGDB.DLL (common procedures and functions for applications)





```
// C-POSTGRESQL-DATABASE.XPJ --- START

[PROJECT]
    DEBUG         = yes
    VERSION       = 2.3
    C-POSTGRESQL-DATABASE.XPJ

[C-POSTGRESQL-DATABASE.XPJ]
    C-POSTGRESQL-DATABASE.EXE

[C-POSTGRESQL-DATABASE.EXE]
    COMPILE       = xpp
//    COMPILE_FLAGS = /wi /wl /wu /q /w
    COMPILE_FLAGS = /q /dUSE_POSTGRES
    DEBUG_SAVE    = no
    GUI           = yes
    LINKER        = alink
    LINK_FLAGS    =
    RC_COMPILE    = arc
    RC_FLAGS      =
    // INTERMEDIATE_DEBUG   = .debug
    // INTERMEDIATE_RELEASE = .release
    OBJ_DIR       = __________C-POSTGRESQL-DATABASE.EXE

// $START-AUTODEPEND
    collat.ch
    gra.ch
    common.ch
    font.ch
    dac.ch
    memvar.ch
    std.ch
    sql.ch
    set.ch
    xbp.ch
    natmsg.ch
    appevent.ch
    get.ch
    prompt.ch

    C-POSTGRESQL-DATABASE.OBJ
    // uses C-PGDB.DLL/C_PGDB.INI

    __tab__create__app.obj

// $STOP-AUTODEPEND

    (RES)\C-POSTGRESQL-DATABASE.RES

    dclipx.lib
    bazne.lib
    baznex.lib
 //-------------
    C-PGDB.LIB
 //-------------
 // C-PGDB.INI
 // KASA.INI
 //-------------
```

```
    (C-POSTGRESQL-DATABASE)\C-POSTGRESQL-DATABASE.PRG
    // uses C-PGDB.DLL/C_PGDB.INI

    (C-POSTGRESQL-DATABASE)\__tab__create__app.prg
    // __tab__create__app.prg
    // generisanje liste sa nazivima i sql stringovima tabli
    // za __tab__create() -> C_PGDB.DLL koja kreira te table,
    // vrši se:
    // u svakoj EXE aplikaciji koja se linkuje sa C_PGDB.DLL

    // __tab__create__app.prg
    // generate list with table names and sql strings
    // for __tab__create() -> C_PGDB.DLL which creates those tabs,
    // is done:
    // in every EXE application that links to C_PGDB.DLL

// C-POSTGRESQL-DATABASE.XPJ --- END


:: C-POSTGRESQL-DATABASE.BAT --- START

PBUILD C-POSTGRESQL-DATABASE.XPJ > __________.TXT
NOTEPAD __________.TXT

:: C-POSTGRESQL-DATABASE.BAT --- END


;; C-PGDB.INI --- START
[COBA Systems]
[DATABASE]
_server_=localhost
_uid_=postgres
_pwd_=coba#1949
_db_=coba
_she_=public
_tab_=coba_systems
_lAlert_=0
_lMsg_=0
_lDBcontrol_=0
;; C-PGDB.INI --- END

;; KASA.INI --- START
[COBA Systems]
[KASA_CFG]
cObjekat_Kase=11
cBroj_Kase=22
;; KASA.INI --- END
```

C-POSTGRESQL-DATABASE.PRG

```
////////////////////////////////////////////////////////////////////////
//                                                                    //
//                                                                    //
//   C-POSTGRESQL-DATABASE.PRG
//                                                                    //
//   24-10-2023                                                       //
//                                                                    //
//   www.cobasystems.com --- COBA Systems --- Slobodan Stanojević Coba //
//   Open Source Project BAST Business Account Software Technology    //
//   www.Alaska-Software.com --- Alaska Xbase++ version 2.0.1503       //
//   www.Donnay-software.com --- eXpress++     version 2.0.268         //
//   Sergej Spirin --- FastReport for Xbase++  version 27.03.2015     //
//                                                                    //
//   Database Server PostgreSQL                version 9.4.4.        //
//                                                                    //
//                                                                    //
////////////////////////////////////////////////////////////////////////


* COMMON FUNCTIONS
*----------------------
* BAZNE.DLL, BAZNEX.DLL
*  xRUN_STOP()
*  gde_exe()
*  gde_ver()
*  ec_p()
*  ec_k()
*  c__greska()
*  c__poruka(
*  c__neda()
*  c__sifra()
*  c__procitaj()
*  cTxt()
*  xc_autor1()
*  Font_codepage()
*  C_EDIT()
*
* POSTGRESQL COMMON FUNCTIONS
*----------------------------
* C-PGDB.DLL
*  __pg__ini()
*  __pg__see()
*  __pg__ini__edit()
*  __pg__lists()
*  __pg__dbinserver()
*  __pg__connect_app()
*  __tab__control__ISAM__()
*  fill__table__all()
*
*  APP FUNCTIONS:
*----------------------------
*  PROCEDURE APPSYS()
*  PROCEDURE DBESYS()
*  PROCEDURE MAIN()
```

```
*  FUNCTION ButtonXP_oConfig1()
*  FUNCTION fu1()
*  FUNCTION fu2()
*  FUNCTION fu3()
*  FUNCTION fu4()
*  STATIC FUNCTION help_programa()
*  FUNCTION function__tab__control__ISAM__()
*  FUNCTION rezerva() //  not in use



/*

Project:
C-POSTGRESQL-DATABASE.BAT
C-POSTGRESQL-DATABASE.XPJ
C-POSTGRESQL-DATABASE.EXE
C-PGDB.INI
KASA.INI

----------

je produkcioni EXE program koji isključivo koristi funkcije
iz C_PGDB.DLL/LIB biblioteke
kao i poseban program sa podacima za kreiranje tabli koji
se ugrađuje i u konkretnu EXE aplikaciju:

     __tab__create__all.prg -> __tab__create__all()

Program služi kao produkcioni korisnički samostalni konfiguracioni
alat za generisanje postgreSQL UPSIZE baze podataka, šema i tabli
za konkretnu aplikaciju *.EXE čije table se učitavaju iz programa
     __tab__create__all.prg -> __tab__create__all()
Program se linkuje sa __tab__create__all.prg koji je poseban za
svaku EXE aplikaciju, pa je i program poseban za svaku aplikaciju.
Naziv programa je naziv EXE aplikacije na čijem kraju je _ LOWLINE

Program kreira i održava ini fajl: C-PGDB.INI
u kome su svi potrebni parametri za kreiranje i start postgreSQL
UPSIZE database, šema i tabli te UPSIZE database.

Program proverava postojanje UPSIZE database pa ako ne postoji on
kreira: databazu, šeme i table koje koristi EXE aplikacija.
A ako postoji on kontroliše postojanje svih potrebnih tabli te
database i prijavljuje grešku ako neka od tabli ne postoji.

Program ima funkcije za generisanje i prikaz listi svih databaza
u postgreSQL serveru, svih šema u konektovanoj databazi i svih
tabli u konektovanoj databazi i zadatoj šemi.

Program puni table sa podacima preuzetim iz SQL script fajlova
iz posebne opcije rada (iz menija programa) na zahtev korisnika.
SQL script fajlovi se nalaze u SQL subfolderu EXE aplikacije:
           primer: C:\EXE\SQL\customers.sql
SQL fajlovi naose naziv table koja se putem njih puni podacima
           primer: table name = 'customers'
                   sql file   = 'customers.sql'
Procedure za punjenje tabli podacima iz SQL script fajlova nalaze
se u __tab__create__all.prg -> __tab__create__all()
```

```
----------

is a production EXE program that only uses functions
from the C_PGDB.DLL/LIB library
as well as a special data program for creating tables that
is also embedded in a specific EXE application:

      __tab__create__all.prg -> __tab__create__all()

The program serves as a production user independent configuration
tool for generating postgreSQL UPSIZE databases, schemas and tables
for a specific application *.EXE whose panels are loaded from the program
      __tab__create__all.prg -> __tab__create__all()
The program links to __tab__create__all.prg which is special for
each EXE application, so the program is special for each application.
The program name is the name of the EXE application ending with _ LOWLINE

The program creates and maintains an ini file: C-PGDB.INI
which contains all the necessary parameters to create and start postgreSQL
UPSIZE database, scheme and tables and UPSIZE database.

The program checks the existence of the UPSIZE database and if it does not exist
creates: the database, schemas and tables used by the EXE application.
And if it exists, it controls the existence of all necessary boards and
database and reports an error if one of the tables does not exist.

The program has functions for generating and displaying a list of all databases
in the postgreSQL server, of all schemas in the connected database and all
tables in the connected database and the given schema.

The program fills tables with data taken from SQL script files
from a special work option (from the program menu) at the user's request.
The SQL script files are located in the SQL subfolder of the EXE application:
            example: C:\EXE\SQL\customers.sql
SQL files contain the name of the table that is filled with data through them
            example: table name = 'customers'
                     sql file = 'customers.sql'
There are procedures for filling tables with data from SQL script files
se in __tab__create__all.prg -> __tab__create__all()

11.10.2023

*/



 #include "dac.ch"
 #pragma library("dbfupsize.lib") // C:\Program Files (x86)\Alaska Software\xpp20\lib
                                  // required for: oInfo := DacSchema():New(oS)
                                  // ================================================
*-----------------------------------
* Xbase++
#include "Xbp.ch"
#include "AppEvent.ch"
#include "Common.ch"
*-----------------------------------
* XbToolIII++
#include "xbtsys.ch"
*-----------------------------------
```

```
* eXpress++
#include "dcdialog.ch"
*-----------------------------------
* Xbase++ PGDBE
#include "pgdbe.ch"
#include "std.ch"
#include "sql.ch"
#pragma library("adac20b.lib")
*-----------------------------------
* Xbase++ APPBROWSE, APPDISPLAY
* for testing the program:
* #include "Appbrow.ch"
* MEMVAR appObject
*-----------------------------------


PROCEDURE APPSYS()

     nOldIcon := DC_ICONDEFAULT(1) // IKONA BROJ 1
     SET CHARSET TO ANSI           // ANSI SLOVA
     SET DATE FORMAT TO "dd.mm.yyyy"
     DC_DotHotKey(180)             // Prebaci DotPrompt sa Alt+D na ...
     SetCancel(.F.)                // turn off Alt+C
 *************
  xRUN_STOP() // BAZNE.DLL
 *************


RETURN
PROCEDURE DBESYS()
  DbeLoad("PGDBE")
RETURN


PROCEDURE MAIN()

LOCAL GetList := {},  GetOptions, oDlg
LOCAL oLogo, oFokus, lButton := .T., oGroup, oMsgBox
LOCAL aSize := Appdesktop():currentsize(), aPos := SetAppWindow():currentpos()
LOCAL oParent := SetAppWindow()
// oParent ne sme da bude PRIVATE ili PUBLIC varijabla !
// ako nije zadat (nil) formira ga DCREADGUI sa PARENT @oParent
// oParent must not be a PRIVATE or PUBLIC variable !
// if not specified (nil) it is created by DCREADGUI with PARENT @oParent

LOCAL Bduz := 34, Bsir := 3.5     // Command Buttons Duzina-Sirina
LOCAL x := 4, y := 1, z := Bsir

PRIVATE aCUR := {"user32.dll",114,XBPWINDOW_POINTERTYPE_POINTER}

SET CHARSET TO ANSI

*******************************************
__pg__ini()  // C_PGDB.DLL

IF __pg__dbinserver() = .F. // C_PGDB.DLL // lRet =.T.,.F.

    ec_k() // BAZNE.DLL
    c__greska("NO DATABASE EXISTS" +chr(59)+;  // BAZNE.DLL
              PADC(_cdbc_,19)      +chr(59)+;
```

```
              "","STOP",,"14.Consolas Bold")

    cTitle:="NO database exists  |  "+_db_
ELSE
    cTitle:="PostgreSQL Database  |  "+_db_
ENDIF
// lRet =.T.,.F.
*********************************************

*****************************
oConfig1 := ButtonXP_oConfig1()  // here
*****************************

PUBLIC RESTART:=.F.

* @ 40,0 DCSTATIC TYPE XBPSTATIC_TYPE_BITMAP PIXEL CAPTION 2001 PARENT oGroup

@ 0,0  DCPUSHBUTTONXP BITMAP 2001 ;
       SIZE Bduz+2,Bsir-0.5 WHEN {|| lButton }  ;
       ACTION {|a,b,o| xc_autor1() } ;  // bazne.dll
       CURSOR aCUR CONFIG oConfig1 ;
       TOOLTIP "MANUFACTURER"


@ x,y  DCPUSHBUTTONXP CAPTION ;
       "Parametri baze podataka;"+_cdbc_+";Database Parameters" OBJECT o1;
       SIZE Bduz,Bsir WHEN {|| lButton }  ;
       ACTION {|a,b,o| IIF(fu2()=.T., ; // here
       DC_ReadGuiEvent(DCGUI_EXIT_OK,GetList),NIL) } ;
        CURSOR aCUR CONFIG oConfig1 ;
       TOOLTIP "PG BAZA PODATAKA ZA PROGRAM: [ KASA STOLOVI ZA UGOSTITELJA ]";
       MESSAGE "PG DATABASE FOR THE PROGRAM"+chr(13)+"[ KASA STOLOVI ZA UGOSTITELJA ]"


x := x+z
@ x,y  DCPUSHBUTTONXP CAPTION "Sadržaj baze podataka;"+_cdbc_+";Database Contents" ;
       SIZE Bduz,Bsir WHEN {|| lButton }   ;
       ACTION {|a,b,o|  fu1(), ;  // here
       SetAppFocus(o) } ;
       CURSOR aCUR CONFIG oConfig1 ;
       TOOLTIP "SERVER LISTE: DATABAZE, ŠEME I TABLE";
       MESSAGE "SERVER LISTS: DATABASES, SCHEMES AND TABLES"


x := x+z
@ x,y  DCPUSHBUTTONXP CAPTION "Kontrola baze podataka;"+_cdbc_+";Database Control" ;
       SIZE Bduz,Bsir WHEN {|| lButton } ;
       ACTION {|a,b,o| IIF(fu3()=.T., ;  // here
       DC_ReadGuiEvent(DCGUI_EXIT_OK,GetList),NIL) } ;
        CURSOR aCUR CONFIG oConfig1 ;
       TOOLTIP "KONTROLA IF EXISTS I KREIRANJE DATABAZE";
       MESSAGE "CONTROL IF EXISTS AND CREATE DATABASE"

x := x+z
@ x,y  DCPUSHBUTTONXP CAPTION "Punjenje baze podataka;"+_cdbc_+";Database Filling" ;
       SIZE Bduz,Bsir WHEN {|| lButton } ;
       ACTION {|a,b,o| fu4(), ;   // here
       SetAppFocus(o) } ;
        CURSOR aCUR CONFIG oConfig1 ;
       TOOLTIP "PRENOS PODATAKA IZ SQL FAJLOVA U TABLE";
```

```
       MESSAGE "DATA TRANSFER FROM SQL FILES TO TABLES"

x := x+z
@ x,y  DCPUSHBUTTONXP CAPTION "Help" ;
       SIZE Bduz/2,Bsir WHEN {|| lButton } ;
       ACTION {|a,b,o| help_programa(),;  // here
        SetAppFocus(o) } ;
        CURSOR aCUR CONFIG oConfig1 ;
       TOOLTIP " USER GUIDE ";
       MESSAGE " USER GUIDE "


@ x,y+Bduz/2;
       DCPUSHBUTTONXP CAPTION "Exit" ;
       SIZE Bduz/2,Bsir WHEN {|| lButton } ;
       ACTION {|a,b,o| DC_ReadGuiEvent(DCGUI_EXIT_OK,GetList) } ;
        CURSOR aCUR CONFIG oConfig1 ;
       TOOLTIP " EXIT FROM PROGRAM " ;
       MESSAGE " EXIT FROM PROGRAM "

 DCHOTKEY 46 ;  // umesto 46 ide asc(".") ili neki od: xbeK_ALT_S ili slično
          ACTION {|a,b,o| SetAppFocus(o1) }  // KEYBOARD


x:=x+z
   @ x,y  DCMESSAGEBOX OBJECT oMsgBox SIZE Bduz,2 ;
          ; // MOTION ;  // WINDOWS 10 DOES NOT SUPPORT MOTION
          ; // TYPE XBPSTATIC_TYPE_RECESSEDBOX  ;
          TEXTOBJECT MESSAGE OPTIONS XBPSTATIC_TEXT_WORDBREAK;
          COLOR GRA_CLR_DARKGREEN, GRA_CLR_WHITE // FONT "9.Helv Bold"



* "ALT+DEL"
* "brisanje svih redova iz table: alaska-software.isam.tables
* "i ponovni upis naziva svih tabli aplikacije u tu tablu
* "Postoji određena opasnost u ovome - vidi u: __tab__create.prg
*----------
* "ALT+DEL"
* "delete all rows from table: alaska-software.isam.tables
* "and rewriting the names of all application panels in that panel
* "There is some danger in this - see in: __tab__create.prg

DCHOTKEY xbeK_ALT_DEL ACTION {|| function__tab__control__ISAM__() }
// here and __tab__create() // C-PGDB.DLL



DCGETOPTIONS ICON 1 TABSTOP NOMINBUTTON NOMAXBUTTON NORESIZE COLOR GRA_CLR_WHITE ;
FONT "11.Archivo Narrow Bold"

DCREAD GUI TITLE {||ctitle} ;
   FIT ;
   OPTIONS GetOptions ;
   MODAL ;
   SETFOCUS @o1   ;
   OWNER oParent  ;
   PARENT @oDlg   ;
   EVAL {|o| SetAppWindow(o)}  // zbog modalnog sledeceg prozora
```

```
IF RESTART=.T.
   MAIN()
ENDIF


RETURN


*********************************************************************
FUNCTION fu1()
*********************************************************************
 IF C__NEDA("PREGLED STANJA BAZE PODATAKA" +chr(59)+;
            "OVERVIEW OF DATABASE STATUS"  +chr(59)+;
            "","OVERVIEW POSTGRESQL DATABASE",,"G3")==.F.
    RETURN NIL
 ENDIF


    __pg__lists(2,.T.)
    //  2  = formira PUBLIC array: _databases_, _schemas_, _tables_
    // .T. = prikaz appbrowse listview: databases, schemas, tables


RETURN NIL
*********************************************************************
FUNCTION fu2()
*********************************************************************
    __pg__ini()   // no connect  // __pg__ini__.prg // C_PGDB.DLL
    __pg__see()   // no connect  // __pg__ini__.prg // C_PGDB.DLL

 IF C__NEDA("POSTAVLJANJE PARAMETARA BAZE PODATAKA" +chr(59)+;
            "SETTING PARAMETERS OF THE DATA BASE"   +chr(59)+;
            "","SETTING DATABASE PARAMETERS",,"G3")==.F.
    RETURN .F.
 ENDIF


    __pg__ini__edit()   // no connect // __pg__ini__edit.prg // C_PGDB.DLL
    __pg__ini()   // no connect  // __pg__ini__.prg // C_PGDB.DLL
    __pg__see()   // no connect  // __pg__ini__.prg // C_PGDB.DLL

    RESTART := .T.

RETURN .T.
*********************************************************************
FUNCTION fu3()
*********************************************************************
LOCAL qq

 IF C__NEDA("KONTROLA ISPRAVNOSTI BAZE PODATAKA"     +chr(59)+;
            "CHECKING THE CORRECTNESS OF DATABASE"   +chr(59)+;
            "","DATABASE CONTROL",,"G3")==.F.
    RETURN NIL
 ENDIF

    ************
    __pg__ini()   // no connect  // __pg__ini__.prg // C_PGDB.DLL
    __pg__see()   // no connect  // __pg__ini__.prg // C_PGDB.DLL
    ************
```

```
      IF C__NEDA("D A T A B A S E  C O N T R O L"          +chr(59)+;
                 " (PostgreSQL database server) "          +chr(59)+;
                 "                               "         +chr(59)+;
                 "Ako zadata baza podataka ne postoji"     +chr(59)+;
                 "biće startom ove operacije kreirana."    +chr(59)+;
                 "Ako zadata baza podataka postoji"        +chr(59)+;
                 "vrši se kontrola njene ispravnosti."     +chr(59)+;
                 "                                  "      +chr(59)+;
                 "If the given database does not exist"    +chr(59)+;
                 "will be created by this operation."      +chr(59)+;
                 "If the given database exists"            +chr(59)+;
                 "its correctness is being checked."       +chr(59)+;
                 "","CHECKING THE CORRECTNESS OF THE DATABASE",3333,"G3")==.F.
         RETURN .F.
      ENDIF

    * __pg__ini()     // no connect  // __pg__ini__.prg // C_PGDB.DLL
    * __pg__lists(1)  // pročitaj samo verziju postgresql servera radi prikaza u __pg__see()
    * __pg__see()     // no connect  // __pg__ini__.prg // C_PGDB.DLL



      *************************
      qq := __pg__connect_app()   // connect-work-disconnect // __pg__ini__.prg // C_PGDB.DLL
      *************************



      IF qq==.T.
         c__poruka("BAZA PODATAKA JE ISPRAVNA"               +chr(59)+;
                   "SVE TABLE U BAZI PODATAKA SU NA BROJU"   +chr(59)+;
                   " "                                       +chr(59)+;
                   "DATABASE IS CORRECT"                     +chr(59)+;
                   "ALL TABLES IN DATABASE ARE ON NUMBER"    +chr(59)+;
                   "","OK",,"G3")
      ELSE
         c__greska("BAZA PODATAKA NIJE ISPRAVNA"             +chr(59)+;
                   "NISU NAĐENE SVE TABLE U BAZI PODATAKA"   +chr(59)+;
                   " "                                       +chr(59)+;
                   "DATABASE IS NOT CORRECT"                 +chr(59)+;
                   "NOT ALL TABLES FOUND IN THE DATABASE"    +chr(59)+;
                   "","STOP",,"G3")
      ENDIF

      RESTART := .T.

RETURN .T.

*****************************************************************************
FUNCTION fu4()
*****************************************************************************
******************************************
__pg__ini()  // C_PGDB.DLL
IF __pg__dbinserver() = .F. // C_PGDB.DLL

    ec_k()
    c__greska("NO DATABASE EXISTS"+chr(59)+;
              PADC(_cdbc_,19)      +chr(59)+;
              "","STOP",,"14.Consolas Bold")
```

```
   * cTitle:="NO database exists  |  "+_db_
   RETURN NIL
ELSE
   * cTitle:="PostgreSQL Database  |  "+_db_
ENDIF
// lRet =.T.,.F.
*******************************************




__pg__ini()     // no connect  // __pg__ini__.prg // C_PGDB.DLL
ctab:="kroba_"+_MAG_

IF C__NEDA("PRENOS PODATAKA IZ SQL FAJLOVA U TABLE" +chr(59)+;
           "DATA TRANSFER FROM SQL FILES TO TABLES" +chr(59)+;
           " "                                      +chr(59)+;
           "Tabla kup (kupci) i tabla "+ctab+" (proizvodi)"         +chr(59)+;
           "biće ispražnjene i napunjene podacima iz sql fajla"     +chr(59)+;
           "\SQL\kup.sql i \SQL\"+ctab+".sql"                       +chr(59)+;
           " "                                                      +chr(59)+;
           "Table kup (customers) and table "+ctab+" (products)"    +chr(59)+;
           "will be emptied and filled with data from the sql file" +chr(59)+;
           "\SQL\kup.sql and \SQL\"+ctab+".sql"                     +chr(59)+;
           "","FILL POSTGRESQL DATABASE TABLES",,"G3")==.F.
   RETURN NIL
ENDIF

IF c__sifra(1)==.F.
 c__greska("NEMATE OVLAŠĆENJE ZA OVO;YOU ARE NOT AUTHORIZED","STOP",,"G3")
 RETURN NIL
ENDIF



* ------------------ punjenje table podacima iz SQL fajla --- start

  * PUBLIC sqlfile_tabledata := gde_exe()+"\SQL\"+ctablename+".SQL"
  * primer: sqlfile_tabledata := "C:\app\SQL\kupci.SQL"

  c__poruka("PUNJENJE TABLE | FILL TABLE --- START")

  * pre poziva ove funkcije vrši se diskonekcija sa database
  * konekcija i diskonekcija obavljaju se u samoj funkciji.
  * Funkcija koristi PUBLIC _db_ i _she_ iz __pg__ini()
  * Funkcija koristi LOCAL _table_ koja dolazi kao parametar

   * before calling this function, the database is disconnected
   * connection and disconnection are performed in the function itself.
   * Function uses PUBLIC _db_ and _she_ from __pg__ini()
   * The function uses LOCAL _table_ which comes as a parameter

  _table_ := ctab
  ************************ // FILL TABLE IN DATABASE
  fill__table__all(_table_) // __pg__tab.prg // C-PGDB.DLL
  ************************
  c__poruka("Tabla kroba_11 je ispražnjena pa napunjena"+chr(59)+;
            "Kroba_11 table was emptied and then filled")
```

```
  _table_ := "kup"
  ************************ // FILL TABLE IN DATABASE
  fill__table__all(_table_) // __pg__tab.prg // C-PGDB.DLL
  ************************
  c__poruka("Tabla kup je ispražnjena pa napunjena"+chr(59)+;
            "kup table was emptied and then filled")

 c__poruka("PUNJENJE TABLE | FILL TABLE --- END")

* ------------------ punjenje table podacima iz SQL fajla --- end


RETURN NIL



//
//*********************************************************************
//FUNCTION rezerva()
//*********************************************************************
//
//    __pg__ini__edit()   // no connect // __pg__ini__edit.prg // C_PGDB.DLL
//
//
//
//      IF C__NEDA("D A T A B A S E  C O N T R O L"          +chr(59)+;
//                 " (PostgreSQL database server) "          +chr(59)+;
//                 "                                "        +chr(59)+;
//                 "Ako zadata baza podataka ne postoji"     +chr(59)+;
//                 "biće startom ove operacije kreirana."    +chr(59)+;
//                 "Ako zadata baza podataka postoji"        +chr(59)+;
//                 "vrši se kontrola njene ispravnosti."     +chr(59)+;
//                 "                                    "    +chr(59)+;
//                 "If the given database does not exist"    +chr(59)+;
//                 "will be created by this operation."      +chr(59)+;
//                 "If the given database exists"            +chr(59)+;
//                 "its correctness is being checked."       +chr(59)+;
//                 "","CHECKING THE CORRECTNESS OF THE DATABASE",,"G3")==.F.
//
//
//               //--------------------------------------------------------------
//                 IF C__NEDA("PREGLED STANJA BAZE PODATAKA" +chr(59)+;
//                            "OVERVIEW OF DATABASE STATUS"  +chr(59)+;
//                            "","OVERVIEW POSTGRESQL DATABASE",,"G3")==.T.
//
//                    __pg__lists(2,.T.)
//                    // 2 = formira PUBLIC array: _databases_, _schemas_, _tables_
//                    // .T. = prikaz appbrowse listview: databases, schemas, tables
//
//                 ENDIF
//
//
//                 IF C__NEDA("PRENOS PODATAKA IZ SQL FAJLOVA U TABLE" +chr(59)+;
//                            "DATA TRANSFER FROM SQL FILES TO TABLES" +chr(59)+;
//                            "","FILL POSTGRESQL DATABASE TABLES",,"G3")==.T.
//
//                 ENDIF
//               //--------------------------------------------------------------
```

```
//                 QUIT
//       ENDIF
//
//
//
//    __pg__ini()     // no connect  // __pg__ini__.prg // C_PGDB.DLL
//    __pg__lists(1)  // pročitaj samo verziju postgresql servera radi prikaza u __pg__see()
//    __pg__see()     // no connect  // __pg__ini__.prg // C_PGDB.DLL
//
//      qq := __pg__connect_app()   // connect-work-disconnect // __pg__ini__.prg // C_PGDB.DLL
//
//      IF qq==.T.
//         c__poruka("BAZA PODATAKA JE ISPRAVNA"               +chr(59)+;
//                   "SVE TABLE U BAZI PODATAKA SU NA BROJU"   +chr(59)+;
//                   " "                                       +chr(59)+;
//                   "DATABASE IS CORRECT"                     +chr(59)+;
//                   "ALL TABLES IN DATABASE ARE ON NUMBER"    +chr(59)+;
//                   "","OK",,"G3")
//      ELSE
//         c__greska("BAZA PODATAKA NIJE ISPRAVNA"             +chr(59)+;
//                   "NISU NAĐENE SVE TABLE U BAZI PODATAKA"   +chr(59)+;
//                   " "                                       +chr(59)+;
//                   "DATABASE IS NOT CORRECT"                 +chr(59)+;
//                   "NOT ALL TABLES FOUND IN THE DATABASE"    +chr(59)+;
//                   "","STOP",,"G3")
//      ENDIF
//
//
//
//
//               //-------------------------------------------------------------------
//                  IF C__NEDA("PREGLED STANJA BAZE PODATAKA" +chr(59)+;
//                             "OVERVIEW OF DATABASE STATUS"  +chr(59)+;
//                             "","OVERVIEW POSTGRESQL DATABASE",,"G3")==.T.
//
//                     __pg__lists(2,.T.)
//                     // 2 = formira PUBLIC array: _databases_, _schemas_, _tables_
//                     // .T. = prikaz appbrowse listview: databases, schemas, tables
//
//                  ENDIF
//
//
//                  IF C__NEDA("PRENOS PODATAKA IZ SQL FAJLOVA U TABLE" +chr(59)+;
//                             "DATA TRANSFER FROM SQL FILES TO TABLES" +chr(59)+;
//                             "","FILL POSTGRESQL DATABASE TABLES",,"G3")==.T.
//
//                  ENDIF
//               //-------------------------------------------------------------------
//
//RETURN  NIL
//
```

```
************************************************************************ 04-12-2020
FUNCTION ButtonXP_oConfig1() // -> oConfig1
************************************************************************
* PRIVATE oConfig1 := ButtonXP_oConfig1()
```

```
* @ 10,10 DCPUSHBUTTONXP CAPTION 'Test' SIZE 10,2 ;
*         CONFIG oConfig1

oConfig1 := DC_XbpPushButtonXPConfig():new()

// default
*oConfig1:bitmapOffset := 5
oConfig1:focusRectStyle  := 1
*oConfig1:focusRectColor  := GraMakeRGBColor({ 200, 30, 70 }) // GRA_CLR_BLACK // GRA_CLR_BLACK
// Color of focus rectangle


oConfig1:outline         := .T. // will cause the button to be outlined.
                                // BORDERCOLOR <nBorderColor> defines the color of the outline.
oConfig1:borderColor     := GraMakeRGBColor({ 256, 256, 256 }) // Color of the button border
when the :outline is .t.


// dugme
oConfig1:bgColor          := GRA_CLR_DARKGREEN   //  Background color
oConfig1:fgColor          := GRA_CLR_WHITE       //  Foreground color of text
// default za CLICK
oConfig1:fgColorClick     := GRA_CLR_BLACK   //  Foreground color of text when button is
clicked.
oConfig1:bgColorClick     := GRA_CLR_GREEN   //  Background color when button is clicked.
// default za SELECT
*oConfig1:enableSelect    := .T.            // RadioButtons - kliknuto je odabrano, a ostala su
blokirana u istom PARENT
*oConfig1:fgColorSelected := GRA_CLR_BLACK  // Foreground color of text when button is selected
(see :enableSelect)
*oConfig1:bgColorSelected := GRA_CLR_GREEN  // Background color when button is selected (see
:enableSelect)
// default za MOUSE MOVE
oConfig1:fgColorMouse     := GRA_CLR_BLACK  // Foreground color of text when mouse is moved
over button.
oConfig1:bgColorMouse     := GRA_CLR_CYAN   // Background color when mouse is moved over
button.
// GRADIJENTI
//oConfig1:gradientStep := 10
//oConfig1:gradientReverse := .t.
//oConfig1:radius := 10


 oConfig1:disabledBGColor  :=  GraMakeRGBColor({ 92, 170, 92 })    // { 244, 244, 210 } //
Background Color when the button is disabled.
// ovo je blokirano i ne može se menjati
* oConfig1:disabledFGColor  :=  GraMakeRGBColor({NIL, NIL, NIL })    // { 150, 150, 150 } //
Foreground Color of text when the button is disabled.

* oConfig1:isTransparent := .t.
RETURN oConfig1
```

```
*********************************************************************
STATIC FUNCTION help_programa()
*********************************************************************
LOCAL txt := "", cr := chr(13)+chr(10)
exe:=STRTRAN(appname(),"_DATABASE")
prg:=appname()+"  ver."+gde_ver()
TEXT INTO txt WRAP cr //TRIMMED

  &prg


  SERVISNI PROGRAM ZA PostgreSQL DATABAZU PROGRAMA &exe
  SERVICE PROGRAM FOR PostgreSQL DATABASE PROGRAM  &exe



  ------------------------------------------------------------------

  OPERATION:
  Parametri baze podataka
  Database Parameters
  PG BAZA PODATAKA ZA PROGRAM: [ KASA STOLOVI ZA UGOSTITELJA ]
  PG DATABASE FOR THE PROGRAM: [  KASA STOLOVI ZA UGOSTITELJA ]

  ------------------------------------------------------------------

   O B A V E Z N I  P A R A M E T R I  P R O G R A M A

   Formiranje parametara aplikacije za rad sa PostgreSQL serverom i
   databazom. Bez ispravnih parametara databazi se ne može pristupiti.
   Parametre upisuje korisnik programa.

   - Za pristup PostgreSQL serveru na lokalu HostName ili ServerName je
     'Localhost'
     ----------

   - Za pristup PostgreSQL serveru na udaljenom serverskom računaru
     HostName kao i podešavanje parametara servera dati su u posebnom
     uputstvu:
     'Pristup_remote_postgreSQL_serveru.pdf'
     ---------------------------------------

   Za testiranje rada programa
   postgreSQL databaze server se instalira kao:

   - host name    : 'Localhost'(zadat u instalaciji postgreSQL servera)
   - user name    : 'postgres' (zadat u instalaciji postgreSQL servera)
   - password     : 'postgres' (zadat u instalaciji postgreSQL servera)

   - databasename : 'kasa'     (obavezno ime)
   - schemename   : 'public'   (obavezno ime)

   - Code Shop-magazine  - šifra prodajnog objekta  = '11' (obavezno)
   - Code Casch-register - šifra fiskalne kase ESIR = '22' (obavezno)


     Moraju se upisati svi zadati parametri aplikacije.
     Server može biti naziv servera ili IP adresa servera.
```

```
   U zagradama su dati default parametri koji će raditi
   sa default instaliranim postgreSQL serverom sa njima.

   Sve se upisuje isključivo malim slovima.

 - User mod messages - prikaz poruka greške korisniku

 - Test mod messages - prikaz poruka administratoru

 - Database control - na startu programa uvek kontroliše
   ispravnost baze podataka programa. Zbog ubrzanja rada
   programa može se isključiti.


 ++++++++++++++++++++++++++++
 MANDATORY PROGRAM PARAMETERS

 Formation of application parameters for working with PostgreSQL server i
 database. Without the correct parameters, the database cannot be accessed.
 Parameters are entered by the user of the program.

 - To access the PostgreSQL server on the local HostName or ServerName is
   'localhost'
   ----------

 - To access the PostgreSQL server on a remote server computer
   HostName as well as setting server parameters are given in a separate
   instructions:
   'Access_remote_postgreSQL_server.pdf'
   ------------------------------------

 To test the operation of the program
 The postgreSQL database server is installed as:

 - host name : 'Localhost' (given in the postgreSQL server installation)
 - user name : 'postgres' (given in the postgreSQL server installation)
 - password : 'postgres' (given in the postgreSQL server installation)

 - databasename : 'kasa' (required name)
 - schemename : 'public' (mandatory name)

 - Code Shop-magazine - store code = '11' (required)
 - Code Casch-register - fiscal cash register code ESIR = '22' (mandatory)


   All the given parameters of the application must be entered.
   Server can be a server name or server IP address.
   The default parameters that will work are given in parentheses
   with postgreSQL server installed by default with them.

   Everything is written exclusively in lowercase letters.

 - User mode messages - displaying error messages to the user

 - Test mode messages - display of messages to the administrator

 - Database control - always controls at the start of the program
   correctness of the program database. Due to acceleration of work
   program can be turned off.
 ++++++++++++++++++++++++++++
```

```
------------------------------------------------------------------------

OPERATION:
Sadržaj baze podataka
Database Contents
SERVER LISTE: DATABAZE, ŠEME I TABLE
SERVER LISTS: DATABASES, SCHEMES AND TABLES

------------------------------------------------------------------------

 S A D R Ž A J  S E R V E R A  I  B A Z E  P O D A T A K A

 DAJE-PRIKAZUJE LISTU SVIH DATABAZA NA POSTGRESQL SERVERU,
 DAJE-PRIKAZUJE LISTU SVIH ŠEMA U AKTIVNOJ DATABAZI,
 DAJE-PRIKAZUJE LISTU SVIH TABLI U ZADATOJ ŠEMI U AKTIVNOJ DATABAZI

  ++++++++++++++++++++++++++++
  SERVER AND DATABASE CONTENT

  GIVES-DISPLAYS THE LIST OF ALL DATABASES ON THE POSTGRESQL SERVER,
  GIVES-DISPLAYS A LIST OF ALL SCHEMAS IN THE ACTIVE DATABASE,
  GIVES-DISPLAYS A LIST OF ALL PANELS IN THE GIVEN SCHEMA IN THE ACTIVE DATABASE

  ++++++++++++++++++++++++++++

------------------------------------------------------------------------

OPERATION:
Kontrola baze podataka
Database Control
KONTROLA IF EXISTS I KREIRANJE DATABAZE
CONTROL IF EXISTS AND CREATE DATABASE

------------------------------------------------------------------------

 K O N T R O L A  I S P R A V N O S T I  B A Z E  P O D A T A K A
 K R E I R A N J E  Z A D A T E  N O V E  B A Z E  P O D A T A K A

 Ova operacija proverava postojanje i ispravnost postgreSQL databaze
 odnosno baze podataka aplikacije.
 Ako zadata databaza - koja je zadata iz operacije:
     Parametri baze podataka
     Database Parameters
 ne postoji na postgreSQL serveru, ova operacija je kreira, ali samo
 ako za kreiranje databaze dozvolu izda korisnik programa.

 Ako zadata databaza postoji, proverava se postojanje zadate šeme,
 koja je takođe zadata iz operacije:
     Parametri baze podataka
     Database Parameters

 Ako zadata šema ne postoji kreira se automatski.

 Ako zadata šema postoji, a ako potrebne table u šemi ne postoje,
 table se automatski kreiraju.

 Kada sve postoji, program se konektuje na databazu i startuje se
```

glavni meni aplikacije bez izdavanja poruka korisniku o tome
da je databaza kontrolisana i ispravna.

Na kraju programa, odnosno aplikacije, program se diskonektuje sa
databaze, ako je korektno završen - ako nije nasilno prekinut.

Podrazumeva se da u računaru mora da prethodno bude instaliran
postgreSQL server.

Ovaj softver je pravljen i testiran sa PostgreSQL serverom
verzija 9.4.4

++
CHECKING THE CORRECTNESS OF THE DATABASE
CREATING THE DEFAULT NEW DATABASE

This operation checks the existence and correctness of the postgreSQL database
that is, application databases.
If the default database - which is the default from the operation:
    Database parameters
    Database Parameters
does not exist on the postgreSQL server, this operation creates it, but only
if the permission to create the database is issued by the user of the program.

If the given database exists, the existence of the given schema is checked,
which is also given from the operation:
    Database parameters
    Database Parameters

If the default scheme does not exist, it is created automatically.

If the given scheme exists, and if the required boards in the scheme do not exist,
boards are created automatically.

When everything exists, the program connects to the database and starts
the main menu of the application without issuing a message to the user about it
that the database is controlled and correct.

At the end of the program, that is, the application, the program disconnects from
of the database, if it was completed correctly - if it was not violently terminated.

It goes without saying that it must be pre-installed on the computer
postgreSQL server.

This software is built and tested with PostgreSQL server
version 9.4.4

++

OPERATION:
Punjenje baze podataka
Database Filling
PRENOS PODATAKA IZ SQL FAJLOVA U TABLE
DATA TRANSFER FROM SQL FILES TO TABLES

P U N J E N J E B A Z E P O D A T A K A

P R E N O S P O D A T A K A I Z S Q L F A J L A U T A B L U

Tabla se puni podacima iz SQL script fajla koji nosi naziv table.

SQL script fajl se formira iz posebnog servisnog programa za
transfer podataka iz DBF fajla u SQL fajl.
Ili za transfer podataka iz PgSQL table u SQL script fajl, ili
za transfer podataka iz EXCEL tabele u SQL fajl, ili za transfer
podataka iz CSV tekst fajla u SQL fajl, ili za transfer podataka
iz XML i JSON fajla u SQL fajl.

NEOPHODAN USLOV

Mora da postoji SQL script fajl.
Naziv SQL script fajla mora biti = nazivtable.SQL,
a sadržaj mora odgovarati strukturi table.
SQL fajl mora da ima komandu INSERT INTO nazivtable.

na primer:
fajl: KUPCI.SQL ili kupci.sql
čiji je sadržaj:

```
-- START
-- upis redova u tablu kupci
INSERT INTO kupci
(
CODE_,
NAME_,
STATUS_
)
  VALUES
(
12345,
'COBA Systems',
TRUE
),
(
12346
'COBA Export'
FALSE
),
(
12347
'COBA Free Software'
FALSE
);
-- ukupno 3 reda
-- END
```

Ako se EXE aplikacija nalazi u folderu C:\EXE tada se sql fajl
mora nalaziti u folderu C:\EXE\SQL

primer: C:\EXE\SQL\kupci.SQL

Note:

Kada se zada komanda za punjenje table 'kupci' ona će biti
izvršena samo ako postoji fajl: C:\EXE\SQL\kupci.SQL

U primeru prikazana tabla 'kup' je takozvana UPSIZE TABLA
sa kojom radi Alaska Xbase++ aplikacija koristeći ISAM komande
i funkcije, uz korišćenje SQL komandi i funkcija, za navigaciju
sa postgreSQL tablama.

Ova tabla 'kup', iz datog primera, mora da ima sledeću strukturu:

```
CREATE TABLE kup
(
  CODE_   integer,
  NAME_   character(50),
  STATUS_ boolean,
  __deleted boolean NOT NULL DEFAULT false,
  __record serial NOT NULL,
  __rowversion integer NOT NULL DEFAULT 1,
  __keyversion integer NOT NULL DEFAULT 0,
  __lock_owner integer NOT NULL DEFAULT 0,
  CONSTRAINT kup_pkey PRIMARY KEY (__record)
 )
 WITH (
   OIDS=FALSE,
   autovacuum_enabled=true
 );
 ALTER TABLE kup
   OWNER TO postgres;
```

Posebna napomena:

(A)

kada se radi sa aplikacijom koja koristi sve tri database engine:
DBFNTX, FOXCDX, PDDBE tada je potrebno imati punu podršku za ISAM
mašinu koju je u postgreSQL server implementirala Alaska Xbase++.
U tom slučaju u database tablu:
"alaska-software.isam.tables"
treba upisati sve table koje se koriste od strane aplikacije po
nazivu table, sa id brojem table i ostalim podacima. To će se kod
kreiranja nove databaze automatski uraditi od strane C-PGDB.DLL
programa. Međutim, to se može uraditi bilo kada na poseban zahtev
korisnika klikom na ALT+DELETE tastere.
Tada će tabla "alaska-software.isam.tables" biti ispražnjena i
ponovo napunjena nazivima i id brojevima tabli koje se koriste od
strane aplikacije.

Ovaj softver ne koristi DBFNTX i FOXCDX database engine već samo
PGDBE engine. Međutim, zadržano je formiranje i ažuriranje ove
liste tabli, jer je aplikacija koristi za druge svoje potrebe.

(B)

kada se radi sa aplikacijom koja koristi sve tri database engine:
DBFNTX, FOXCDX, PDDBE tada je potrebno imati punu podršku za ISAM
mašinu koju je u postgreSQL server implementirala Alaska Xbase++.
U tom slučaju u database tablu:
"alaska-software.isam.orders"
treba upisati sve indeksne NTX i CDX fajlove po njihovom nazivu i
po extenziji, po DBF polju ili izrazu po kome je indeks formiran,
po njihovom nazivu u sql tabli i po ostalim parametrima.

```
Ovo se kod kreiranje nove baze podataka iz C-PGDB.DLL ne radi,
a ne radi se ni na poseban zahtev korisnika, jer aplikacija koristi
samo PGDBE engine. Dakle, aplikacija ne radi istovremeno sa DBF,
NTX, CDX, DBT, FPT fajlovima ISAM database i sa postgreSQL database
tablama.

(C)

Kako se vrši puna implementacija Alaska Xbase++ ISAM mašine u
postgreSQL bazu podataka i šta se kao rezultat dobija u postgreSQL
bazi podataka detaljno je opisano i može se testirati iz programa
DBFUPSIZE.EXE koji se isporučuje u Alaska Xbase++ 2.0 paketu i u
        \Documents\Xbase++\source\samples\apps\mdidemo
        \Documents\Xbase++\source\samples\sql\xbpbrowse
primeru koji kreira database: 'postgres' šemu 'public' i navedene
table za ISAM mašinu kao i table koje koristi aplikacija mdidemo.


--------------------------------------------------------------------------


  ++++++++++++++++++++++++++++++++++++++++

  FILLING THE DATABASE
  DATA TRANSFER FROM SQL FILE TO TABLE

  The table is filled with data from the SQL script file that bears
  the name of the table.

  The SQL script file is created from a special service program for
  data transfer from DBF file to SQL file.
  Or to transfer data from PgSQL table to SQL script file, or
  for data transfer from EXCEL table to SQL file, or for transfer
  data from CSV text file to SQL file, or for data transfer
  from XML and JSON file to SQL file.

   A NECESSARY CONDITION
   ---------------
   There must be a SQL script file.
   The name of the SQL script file must be = nametable.SQL,
   and the content must match the structure of the board.
   The SQL file must have an INSERT INTO nametable command.

   for example:
   file: CUSTOMERS.SQL or customers.sql
   the contents of which are:

   -- START
   -- entry of rows in the customer table
   INSERT INTO customers
   (
   CODE_,
   NAME_,
   STATUS_
   )
     VALUES
   (
   12345,
   'COBA Systems',
   TRUE
```

```
),
(
12346
'COBA Export'
FALSE
),
(
12347
'COBA Free Software'
FALSE
);
-- 3 rows in total
-- END
```

If the EXE application is located in the C:\EXE folder, then the sql file must be in the C:\EXE\SQL folder

example: C:\EXE\SQL\customers.SQL

Notes:

When the command is given to fill the 'customers' panel it will be executed only if the file exists: C:\EXE\SQL\kupci.SQL

In the example, the 'cup' board shown is the so-called UPSIZE BOARD with which the Alaska Xbase++ application works using ISAM commands and functions, using SQL commands and functions, for navigation with postgreSQL tables.

This 'cup' board, from the given example, must have the following structure:

```
CREATE TABLE purchase
(
  CODE_ integer,
  NAME_ character(50),
  STATUS_ boolean,
  __deleted boolean NOT NULL DEFAULT false,
  __record serial NOT NULL,
  __rowversion integer NOT NULL DEFAULT 1,
  __keyversion integer NOT NULL DEFAULT 0,
  __lock_owner integer NOT NULL DEFAULT 0,
  CONSTRAINT kup_pkey PRIMARY KEY (__record)
 )
 WITH (
   OIDS=FALSE,
   autovacuum_enabled=true
 );
 ALTER TABLE purchase
   OWNER TO postgres;
```

Special note:

(A)

when working with an application that uses all three database engines:

```
   DBFNTX, FOXCDX, PDDBE then it is necessary to have full ISAM support
   engine implemented in postgreSQL server by Alaska Xbase++.
   In that case, in the database table:
                   "alaska-software.isam.tables"
   all boards used by the application should be entered
   name of the board, with the id number of the board and other data.
   It will code creation of a new database will be done automatically
   by C-PGDB.DLL program. However, this can be done at any time upon
   special request user by clicking the ALT+DELETE keys.
   Then the table "alaska-software.isam.tables" will be emptied and
   reloaded with the names and id numbers of the boards used by
   foreign applications.

   This software does not use DBFNTX and FOXCDX database engine but only
   PGDBE engine. However, the formation and updating of this was retained
   list boards, because the application uses it for other purposes.

   (B)

   when working with an application that uses all three database engines:
   DBFNTX, FOXCDX, PDDBE then it is necessary to have full ISAM support
   engine implemented in postgreSQL server by Alaska Xbase++.
   In that case, in the database table:
                  "alaska-software.isam.orders"
   all index NTX and CDX files should be entered by their name and
   by extension, by DBF field or expression by which the index was formed,
   by their name in the sql table and by other parameters.
   This does not work when creating a new database from C-PGDB.DLL,
   and it is not even done at the special request of the user, because
   the application uses PGDBE engine only. So the app doesn't run concurrently
   with DBF, NTX, CDX, DBT, FPT files ISAM database and postgreSQL database
   boards.

   (C)

   How to do a full implementation of the Alaska Xbase++ ISAM engine in the
   postgreSQL database and what is output in postgreSQL
   database is described in detail and can be tested from within the program
   DBFUPSIZE.EXE supplied in the Alaska Xbase++ 2.0 package and in the
           \Documents\Xbase++\source\samples\apps\mdidemo
           \Documents\Xbase++\source\samples\sql\xbpbrowse
   example that creates a database: 'postgres' schema 'public' and listed
   boards for the ISAM machine as well as the boards used by the mdidemo
   application.

   ++++++++++++++++++++++++++++++++++++++++
   zadnja revizija: 13-10-2023
   last revision: 13-10-2023
ENDTEXT
  TxtFile:=cTxt()
  MEMOWRIT(TxtFile,txt)
  ****************************************
  off := Font_codepage(10,"Consolas",238,.f.)
  C_EDIT(TxtFile,"PostgreSQL Database Tools",;
                 "PostgreSQL Database Tools",,,,,,off) // baznex.dll
  ****************************************
   CLEAR TYPEAHEAD
RETURN NIL
```

```
*************************************************************************
FUNCTION function__tab__control__ISAM__()
*************************************************************************


IF c__neda(;   // BAZNE.DLL
"Proverava se postojanje svake table u postgreSQL databazi:"       + chr(59)+;
"         "+_db_+"."+_she_                                         + chr(59)+;
"prema listi tabli upisanoj u aplikaciju. Ako tabla postoji,"      + chr(59)+;
"naziv i id broj table upisuje se u sistemsku Alaska Xbase++"      + chr(59)+;
"postgreSQL listu, u tablu:  < alaska-software.isam.tables >"      + chr(59)+;
"koja se prethodno isprazni. Ovim se formira lista database"       + chr(59)+;
"tabli aplikacije, koja se koristi od strane ISAM mašine kad"      + chr(59)+;
"aplikacija radi sa ISAM DBFNTX DBE, FOXCDX DBE i PGDBE"           + chr(59)+;
" "                                                                + chr(59)+;
"Ova lista u tabli < alaska-software.isam.tables > formirana"      + chr(59)+;
"je kod prvog kreiranja database aplikacije i ako je imala"        + chr(59)+;
"90 tabli one su dobile svoje nazive i id brojeve od 1 do 90."     + chr(59)+;
"Svako izvršenje ove operacije uvećaće id broj svake table za"     + chr(59)+;
"broj 90 i resetovaće datume i vremena kreiranja i izmene kod"     + chr(59)+;
"svake table. O ovome treba brinuti ako aplikacija koristi id"     + chr(59)+;
"brojeve tabli i ostale podatke iz ovih tabli."                    + chr(59)+;
"---------- ENG ----------"                                        + chr(59)+;
 "Checking the existence of each table in the postgreSQL database:"+ chr(59)+;
 "         "+_db_+"."+_she_                                        + chr(59)+;
 "according to the list of tabless written in the application."    + chr(59)+;
 "If the table exists, the name and id number of the table are"    + chr(59)+;
 "entered into the system Alaska Xbase++ postgreSQL list, "        + chr(59)+;
 "in table: < alaska-software.isam.tables > which is previously"   + chr(59)+;
 "emptied. This creates a database list application table, "       + chr(59)+;
 "used by the ISAM engine when application works with "            + chr(59)+;
 "ISAM DBFNTX DBE, FOXCDX DBE and PGDBE"                           + chr(59)+;
 " "                                                               + chr(59)+;
 "This list in table < alaska-software.isam.tables > formed"       + chr(59)+;
 "is when the database application was first created and if it had"+ chr(59)+;
 "90 tables they got their names and id numbers from 1 to 90."     + chr(59)+;
 "Each execution of this operation will increment the id number of"+ chr(59)+;
 "each table by number 90 and will reset the dates and times of"   + chr(59)+;
 "creation and modification of the code every table. This should"  + chr(59)+;
 "be taken care of if the application uses id table numbers and"   + chr(59)+;
 "other data from these tables."                                   + chr(59)+;
 "","KONTROLA DATABASE TABLI | alaska-software.isam.tables | ") == .F.

  RETURN NIL

ENDIF


__tab__control__ISAM__()   // __tab__create.prg // C-PGDB.DLL


RETURN NIL
```

__TAB__CREATE__APP.PRG

```
//////////////////////////////////////////////////////////////////////////
//                                                                      //
//                                                                      //
//   __tab__create__app.prg                                             //
//                                                                      //
//   24-10-2023                                                         //
//                                                                      //
//   www.cobasystems.com --- COBA Systems --- Slobodan Stanojević Coba  //
//   Open Source Project BAST Business Account Software Technology      //
//   www.Alaska-Software.com --- Alaska Xbase++ version 2.0.1503         //
//   www.Donnay-software.com --- eXpress++      version 2.0.268          //
//   Sergej Spirin --- FastReport for Xbase++   version 27.03.2015      //
//                                                                      //
//   Database Server PostgreSQL                 version 9.4.4.          //
//                                                                      //
//                                                                      //
//////////////////////////////////////////////////////////////////////////


* COMMON FUNCTION
* INI_WRITE() // BAZNE.DLL
* INI_READ()  // BAZNE.DLL

* APP FUNCTION
* FUNCTION __tab__create__app()
* FUNCTION __tab__create__app__1()
* FUNCTION SQLstring()
* FUNCTION SQLstring__()



* ------------------------------------------------------------------------
* OVAJ PROGRAM NALAZI SE U SVAKOJ XBASE++ APLIKACIJI KOJA RADI SA PGDBE
* POSEBAN JE ZA SVAKU XBASE++ APLIKACIJU - JER FORMIRA SAMO NJENE TABLE
* ------------------------------------------------------------------------
* Formira array _aListAppTable_ koji sadrži n članova, gde je svaki član:
*       {tablename, schemename.tablename, sqlstring}
* Nazivi tabli za database _db_ (xdb) i schemas _she_ (xse) u COBA
* Systems aplikaciji i SQL string za kreiranje svake od tabli iz liste.
* Lista naziva tabli koristi se iz biblioteke: C-PGDB.DLL/LIB
* za kreiranje postgreSQL database, baze podataka koju koristi aplikacija.
*
* Funkcija:
* _aListAppTable_ := __tab__cerate__app() -> __tab__create__app.prg
* vrši generisanje liste sa nazivima tabli i sql stringovima tabli
* za funkciju: __tab__create() -> C-PGDB.DLL koja kreira te table,
* Ova funkcija je u svakoj EXE aplikaciji koja se linkuje sa C-PGDB.DLL
*------------------------------------------------------------------------

* ------------------------------------------------------------------------
* THIS PROGRAM IS IN EVERY XBASE++ APPLICATION THAT WORKS WITH PGDBE
* IT IS SPECIAL FOR EACH XBASE++ APPLICATION - BECAUSE IT FORM ONLY ITS
```

```
* TABLES
* ------------------------------------------------------------------------
* Forms an array _aListAppTable_ containing n members, where each member
* is: {tablename, schemename.tablename, sqlstring}
* Table names for database _db_ (xdb) and schemas _she_ (xse) in the
* COBA Systems application and the SQL string to create each of the
* tables from the list.
* The table name list is used from the library: C-PGDB.DLL/LIB
* to create the postgreSQL database, the database used by the application.
*
* Function:
* _aListAppTable_ := __tab__cerate__app() -> __tab__create__app.prg
* generates a list with table names and table sql strings
* for the function: __tab__create() -> C-PGDB.DLL which creates those
* tables.
* This function is in every EXE application that links with C-PGDB.DLL
*-------------------------------------------------------------------------




*-----------------------------------
#include "Xbp.ch"
#include "AppEvent.ch"
#include "Common.ch"
*-----------------------------------
* XbTools++
#include "xbtsys.ch"
* eXpress++
#include "dcdialog.ch"
*-----------------------------------
* for test
*  #include "Appbrow.ch"
*  MEMVAR appObject



*************************************************************************
FUNCTION __tab__create__app()
*************************************************************************
* Formira array _aListAppTable_ koji sadrži n članova, gde je svaki član:
*       {tablename, schemename.tablename, sqlstring}
* Nazivi tabli za database _db_ (xdb)i schemas _she_ (xse)
* u COBA Systems aplikaciji
* i SQL string za kreiranje svake od tabli iz liste.
* Lista naziva tabli koristi se iz funkcije:
*
*          __tab__create() -> __tab__create.prg -> C-PGDB.DLL
*
* za kreiranje tabli u bazi podataka _db_
* ----------
* Forms an array _aListAppTable_ containing n members, where each member is:
* {tablename, schemename.tablename, sqlstring}
* Table names for database _db_ (xdb)and schemas _she_ (xse)
* in the COBA Systems application
* and the SQL string to create each of the tables from the list.
* The list of board names is used from the function:
*
*          __tab__create() -> __tab__create.prg -> C-PGDB.DLL
```

```
*
* to create tables in the _db_ (xdb)database
* ----------




* ----------------------------------------
* Ovde je procedura za aplikaciju:
*                 KASA_STOLOVI.EXE
* Here is procedure for aplication:
*                 KASA_STOLOVI.EXE
* ----------------------------------------


LOCAL xdb, xse, _MAG_, _KAS_, cObjekat_Kase, cBroj_Kase, i:=0

  xdb := "kasa"     // default database name
  xse := "public"   // default scheme name

 _MAG_ := INI_READ("C","KASA_CFG","cObjekat_Kase",gde_exe()+"\KASA.INI") // broj prodavnice
 _KAS_ := INI_READ("C","KASA_CFG","cBroj_Kase"   ,gde_exe()+"\KASA.INI") // broj kase


  IF EMPTY(_MAG_)
     INI_WRITE("C","KASA_CFG","cObjekat_Kase","11",gde_exe()+"\KASA.INI")  // D E F A U L T
     _MAG_ := INI_READ("C","KASA_CFG","cObjekat_Kase",gde_exe()+"\KASA.INI")
  ENDIF
  IF EMPTY(_KAS_)
     INI_WRITE("C","KASA_CFG","cBroj_Kase","11",gde_exe()+"\KASA.INI")  // D E F A U L T
     _KAS_ := INI_READ("C","KASA_CFG","cBroj_Kase",gde_exe()+"\KASA.INI")
  ENDIF

  _MAG_ := STRZERO( VAL(_MAG_), 2 )   // od '00' do '99'
  _KAS_ := STRZERO( VAL(_KAS_), 2 )   // od '00' do '99'

  cObjekat_Kase := _MAG_   // šifra restorana     | restaurant code             --- '11'
  cBroj_Kase    := _KAS_   // šifra fiskalne kase | code of the fiscal register --- '22'


PUBLIC _aListAppTable_ := {}

* ----------------------------------------
  tablename   := "kup"
  __tab__create__app__1(xdb,xse,tablename)
* ----------------------------------------
  tablename   := "kroba_"+_MAG_
  __tab__create__app__1(xdb,xse,tablename)
* ----------------------------------------
  tablename   := "kasa_"+_KAS_
  __tab__create__app__1(xdb,xse,tablename)
* ----------------------------------------
  tablename   := "kaso_"+_KAS_
  __tab__create__app__1(xdb,xse,tablename)
* ----------------------------------------
  tablename   := "kbon_"+_KAS_
  __tab__create__app__1(xdb,xse,tablename)
* ----------------------------------------
  tablename   := "kblok_"+_KAS_
  __tab__create__app__1(xdb,xse,tablename)
```

```
* ----------------------------------------
  FOR i=1 TO 100
    tablename := "x"+var2char(i)
  __tab__create__app__1(xdb,xse,tablename)
  NEXT i
* ----------------------------------------

RETURN _aListAppTable_



*************************************************************************
FUNCTION __tab__create__app__1(xdb,xse,tablename)
*************************************************************************
tableschema := xse+"."+tablename
SQLstring   := SQLstring(xdb,xse,tablename,tableschema)
AADD( _aListAppTable_, { tablename, tableschema, SQLstring } )
RETURN NIL



*************************************************************************
FUNCTION SQLstring(xdb,xse,tablename,tableschema)
*************************************************************************
LOCAL txt := "", cr := chr(13)+chr(10)
LOCAL sqltxt1 := "", sqltxt2 := "", sqltxt := ""

/*
-- ****************************************
-- CREATE TABLE:
-- ****************************************
--  Table kreiraj sa dodatnim kolonama
--  ZA UPSIZE - ZA ISAM MAŠINU:
--  Create tables with additional columns
--  FOR UPSIZE - FOR THE ISAM MACHINE:
--
--  __deleted boolean NOT NULL DEFAULT false,
--  __record serial NOT NULL,
--  __rowversion integer NOT NULL DEFAULT 1,
--  __keyversion integer NOT NULL DEFAULT 0,
--  __lock_owner integer NOT NULL DEFAULT 0,
--  CONSTRAINT tablename_pkey PRIMARY KEY (__record)
*/


*" (KUPCI_SPISAK) variable for DBF name in application KUP.DBF
*########################################################## START
IF tablename=="kup"
*##########################################################
// --- poseban lični kod za svaku od tabli  --- start
//     moraji se formirati PRIVATE zbog upisa varijabli sa & u text into
//     special personal code for each of the tables
//     you have to form PRIVATE because of writing variables with & in text into

   PRIVATE ctable := tableschema
   PRIVATE ckey   := tablename + "_pkey"
   PRIVATE caddress  := xdb+"."+xse+"."+tablename
```

```
TEXT INTO sqltxt1 WRAP cr // TRIMMED
--
-- Table: &caddress
--
                                                                                -- START
CREATE TABLE &ctable   -- THIS PART OF THE CODE IS DOWNLOADED FROM THE FILE KUP.TXT
  (                    -- WHICH IS FORMED IN \SQL FOLDER FROM PROGRAM C-DBF2SQLFILE.EXE
  KUDS_                character(4)        ,
  KUDO_                character(1)        ,
  VEZA_                character(13)       ,
  BARC_                character(13)       ,
  KUDN_                character(40)       ,
  KUDU_                character(40)       ,
  ADRE_                character(30)       ,
  POST_                character(5)        ,
  MEST_                character(24)       ,
  REPU_                character(30)       ,
  LICE_                character(30)       ,
  TFON_                character(30)       ,
  TFAX_                character(30)       ,
  TLEX_                character(30)       ,
  ZIRO_                character(30)       ,
  MBR_                 character(30)       ,
  PIB_                 character(9)        ,
  SDEL_                character(5)        ,
  IBAN_                character(30)       ,
  ACOD_                character(30)       ,
  MAIL_                character(40)       ,
  WEBS_                character(40)       ,
  DATO_                date                ,
  DATR_                date                ,
  ROK_                 integer             ,
  RABAT_               numeric(6,2)        ,
  GRUPA_               character(4)        ,
  STATU_               character(4)        ,
  NOTES_               character(40)       ,
  KONT_                character(10)       ,
  FLAG_                character(1)        ,
  ZNAK_                character(1)        ,
  __deleted boolean NOT NULL DEFAULT false,
  __record serial NOT NULL,
  __rowversion integer NOT NULL DEFAULT 1,
  __keyversion integer NOT NULL DEFAULT 0,
  __lock_owner integer NOT NULL DEFAULT 0,
  CONSTRAINT &ckey PRIMARY KEY (__record)
)
WITH (OIDS=FALSE,autovacuum_enabled=true);                                      -- END

ENDTEXT

* "formiran je SQL string 'sqltxt1'
* "formed SQL string 'sqltxt1'
// --- poseban lični kod za svaku od tabli
// --- special personal code for each of the tables --- end

// --- zajednički common kod za svaku od tabli  --- start
//     dobija se dodatak za string 'sqltxt'
//     shared common code for each of the tables
//     get append for string 'sqltxt'
```

```
  ************************************************
  sqltxt2 := SQLstring__(xdb,xse,tablename,tableschema)
  ************************************************
// --- shared common code for each of the tables  --- end

// spajanje dva stringa:
// concatenate two strings:
  sqltxt := sqltxt1 + sqltxt2
RETURN sqltxt
*#########################################################
ENDIF // IF tablename=="kup"
*#########################################################  END



*" (ARTIKLI_SPISAK) variable for DBF name in application KROBA_11.DBF
*######################################################### START
IF LEFT(tablename,6)=="kroba_"
*#########################################################
// --- poseban lični kod za svaku od tabli  --- start
// moraji se formirati PRIVATE zbog upisa varijabli sa & u text into
   PRIVATE ctable := tableschema
   PRIVATE ckey   := tablename + "_pkey"
   PRIVATE caddress  := xdb+"."+xse+"."+tablename

TEXT INTO sqltxt1 WRAP cr // TRIMMED
--
-- Table: &caddress
--
CREATE TABLE &ctable
(
  GRUP_               character(2)        ,
  ROBS_               character(5)        ,
  ROBN_               character(25)       ,
  ROBK_               character(25)       ,
  MERA_               character(3)        ,
  TARI_               character(1)        ,
  PPUP_               numeric(5,1)        ,
  PPAP_               numeric(10,2)       ,
  CENA_FAK_           numeric(13,2)       ,
  CENA_NAB_           numeric(13,2)       ,
  CENA_VPR_           numeric(13,2)       ,
  CENA_MPR_           numeric(13,2)       ,
  CENA_DEV_           numeric(13,2)       ,
  ZALI_               numeric(15,3)       ,
  DAT0_               date                ,
  MAGS_               character(2)        ,
  ZNAK_               character(1)        ,
  FLAG_               character(1)        ,
  STAT_               character(1)        ,
  CENA_PRO_           numeric(13,3)       ,
  OSNO_AKC_           numeric(13,3)       ,
  ROK_                integer             ,
  DOBS_               character(4)        ,
  DOBK_               character(25)       ,
  VEZA_               character(15)       ,
  NAME_               character(200)      ,
  OPIS_               text                ,
  __deleted boolean NOT NULL DEFAULT false,
  __record serial NOT NULL,
```

```
  __rowversion integer NOT NULL DEFAULT 1,
  __keyversion integer NOT NULL DEFAULT 0,
  __lock_owner integer NOT NULL DEFAULT 0,
  CONSTRAINT &ckey PRIMARY KEY (__record)
)
WITH (OIDS=FALSE,autovacuum_enabled=true);

ENDTEXT
* "formiran je SQL string 'sqltxt1'
// --- poseban lični kod za svaku od tabli  --- end


// --- zajednički common kod za svaku od tabli  --- start
//     dobija se dodatak za string 'sqltxt'
  ************************************************
  sqltxt2 := SQLstring__(xdb,xse,tablename,tableschema)
  ************************************************
// --- zajednički common kod za svaku od tabli  --- end
// spajanje dva stringa:
  sqltxt := sqltxt1 + sqltxt2
RETURN sqltxt
*#########################################################
ENDIF // IF LEFT(tablename,6)=="kroba_"
*######################################################### END



*" (KASA_STAV) variable for DBF name in application KASA_22.DBF
*######################################################### START
IF LEFT(tablename,5)=="kasa_"
*#########################################################
// --- poseban lični kod za svaku od tabli  --- start
// moraji se formirati PRIVATE zbog upisa varijabli sa & u text into
   PRIVATE ctable := tableschema
   PRIVATE ckey   := tablename + "_pkey"
   PRIVATE caddress  := xdb+"."+xse+"."+tablename

TEXT INTO sqltxt1 WRAP cr // TRIMMED
--
-- Table: &caddress
--
CREATE TABLE &ctable
  (
  BRJ0_               character(12)       ,
  DAT0_               date                ,
  VRE0_               character(10)       ,
  BRJ1_               character(30)       ,
  DAT1_               character(30)       ,
  BRJ2_               character(30)       ,
  DAT2_               character(30)       ,
  VAL1_               character(30)       ,
  DOBS_               character(4)        ,
  DOBO_               character(1)        ,
  MAGS_               character(2)        ,
  OPST_               character(2)        ,
  KUP1_               character(30)       ,
  KUP2_               character(30)       ,
  RAC1_               character(2)        ,
  RAC2_               character(30)       ,
  KSIR_               character(13)       ,
```

```
  ESIR_               character(30)       ,
  ZNAK_               character(1)        ,
  STAT_               character(1)        ,
  GRUP_               character(2)        ,
  ROBS_               character(5)        ,
  ROBK_               character(30)       ,
  DOBK_               character(30)       ,
  ROBN_               character(50)       ,
  MERA_               character(3)        ,
  KOLI_               numeric(16,3)       ,
  CENA_FAK_           numeric(16,2)       ,
  CENA_MPR_           numeric(16,2)       ,
  VRED_MPR_           numeric(16,2)       ,
  TARI_               character(1)        ,
  PPUP_               numeric(5,1)        ,
  PPOD_               numeric(16,2)       ,
  CENA_VPR_           numeric(16,2)       ,
  VRED_VPR_           numeric(16,2)       ,
  RABP_               numeric(6,2)        ,
  RABD_               numeric(16,2)       ,
  FLAG_               character(1)        ,
  FLAG_IME_           character(30)       ,
  UPLATA_             numeric(16,2)       ,
  RACUN_              numeric(16,2)       ,
  KUSUR_              numeric(16,2)       ,
  GOT_                numeric(16,2)       ,
  CEK_                numeric(16,2)       ,
  KAR_                numeric(16,2)       ,
  NAL_                numeric(16,2)       ,
  INS_                numeric(16,2)       ,
  VAU_                numeric(16,2)       ,
  BOT_                numeric(16,2)       ,
  AVA_                numeric(16,2)       ,
  AVP_                numeric(16,2)       ,
  COBA_               integer             ,
  COK_                boolean             ,
  UUID_               character(36)       ,
  __deleted boolean NOT NULL DEFAULT false,
  __record serial NOT NULL,
  __rowversion integer NOT NULL DEFAULT 1,
  __keyversion integer NOT NULL DEFAULT 0,
  __lock_owner integer NOT NULL DEFAULT 0,
  CONSTRAINT &ckey PRIMARY KEY (__record)
)
WITH (OIDS=FALSE,autovacuum_enabled=true);

ENDTEXT
* "formiran je SQL string 'sqltxt1'
// --- poseban lični kod za svaku od tabli  --- end


// --- zajednički common kod za svaku od tabli  --- start
//     dobija se dodatak za string 'sqltxt'
  ************************************************
  sqltxt2 := SQLstring__(xdb,xse,tablename,tableschema)
  ************************************************
// --- zajednički common kod za svaku od tabli  --- end
// spajanje dva stringa:
  sqltxt := sqltxt1 + sqltxt2
RETURN sqltxt
```

```
*#########################################################
ENDIF // IF LEFT(tablename,5)=="kasa_"
*######################################################### END



*" (KASA_DEL) variable for DBF name in application KASO_22.DBF
*######################################################### START
IF LEFT(tablename,5)=="kaso_"
*#########################################################
// --- poseban lični kod za svaku od tabli  --- start
// moraji se formirati PRIVATE zbog upisa varijabli sa & u text into
   PRIVATE ctable := tableschema
   PRIVATE ckey   := tablename + "_pkey"
   PRIVATE caddress  := xdb+"."+xse+"."+tablename

TEXT INTO sqltxt1 WRAP cr // TRIMMED
--
-- Table: &caddress
--
CREATE TABLE &ctable
  (
  BRJ0_               character(12)       ,
  DAT0_               date                ,
  VRE0_               character(10)       ,
  BRJ1_               character(30)       ,
  DAT1_               character(30)       ,
  BRJ2_               character(30)       ,
  DAT2_               character(30)       ,
  VAL1_               character(30)       ,
  DOBS_               character(4)        ,
  DOBO_               character(1)        ,
  MAGS_               character(2)        ,
  OPST_               character(2)        ,
  KUP1_               character(30)       ,
  KUP2_               character(30)       ,
  RAC1_               character(2)        ,
  RAC2_               character(30)       ,
  KSIR_               character(13)       ,
  ESIR_               character(30)       ,
  ZNAK_               character(1)        ,
  STAT_               character(1)        ,
  GRUP_               character(2)        ,
  ROBS_               character(5)        ,
  ROBK_               character(30)       ,
  DOBK_               character(30)       ,
  ROBN_               character(50)       ,
  MERA_               character(3)        ,
  KOLI_               numeric(16,3)       ,
  CENA_FAK_           numeric(16,2)       ,
  CENA_MPR_           numeric(16,2)       ,
  VRED_MPR_           numeric(16,2)       ,
  TARI_               character(1)        ,
  PPUP_               numeric(5,1)        ,
  PPOD_               numeric(16,2)       ,
  CENA_VPR_           numeric(16,2)       ,
  VRED_VPR_           numeric(16,2)       ,
  RABP_               numeric(6,2)        ,
  RABD_               numeric(16,2)       ,
  FLAG_               character(1)        ,
```

```
  FLAG_IME_           character(30)       ,
  UPLATA_             numeric(16,2)       ,
  RACUN_              numeric(16,2)       ,
  KUSUR_              numeric(16,2)       ,
  GOT_                numeric(16,2)       ,
  CEK_                numeric(16,2)       ,
  KAR_                numeric(16,2)       ,
  NAL_                numeric(16,2)       ,
  INS_                numeric(16,2)       ,
  VAU_                numeric(16,2)       ,
  BOT_                numeric(16,2)       ,
  AVA_                numeric(16,2)       ,
  AVP_                numeric(16,2)       ,
  COBA_               integer             ,
  COK_                boolean             ,
  UUID_               character(36)       ,
  __deleted boolean NOT NULL DEFAULT false,
  __record serial NOT NULL,
  __rowversion integer NOT NULL DEFAULT 1,
  __keyversion integer NOT NULL DEFAULT 0,
  __lock_owner integer NOT NULL DEFAULT 0,
  CONSTRAINT &ckey PRIMARY KEY (__record)
)
WITH (OIDS=FALSE,autovacuum_enabled=true);

ENDTEXT
* "formiran je SQL string 'sqltxt1'
// --- poseban lični kod za svaku od tabli  --- end


// --- zajednički common kod za svaku od tabli  --- start
//     dobija se dodatak za string 'sqltxt'
  ***************************************************
  sqltxt2 := SQLstring__(xdb,xse,tablename,tableschema)
  ***************************************************
// --- zajednički common kod za svaku od tabli  --- end
// spajanje dva stringa:
  sqltxt := sqltxt1 + sqltxt2
RETURN sqltxt

*#########################################################
ENDIF // IF LEFT(tablename,5)=="kaso_"
*######################################################### END



*" (KASA_BON) variable for DBF name in application KBON_22.DBF
*######################################################### START
IF LEFT(tablename,5)=="kbon_"
*#########################################################
// --- poseban lični kod za svaku od tabli  --- start
// moraji se formirati PRIVATE zbog upisa varijabli sa & u text into
   PRIVATE ctable := tableschema
   PRIVATE ckey   := tablename + "_pkey"
   PRIVATE caddress  := xdb+"."+xse+"."+tablename

TEXT INTO sqltxt1 WRAP cr // TRIMMED
--
-- Table: &caddress
--
```

```
CREATE TABLE &ctable
  (
  BRJ0_                 character(12)       ,
  DAT0_                 date                ,
  VRE0_                 character(10)       ,
  BRJ1_                 character(30)       ,
  DAT1_                 character(30)       ,
  BRJ2_                 character(30)       ,
  DAT2_                 character(30)       ,
  VAL1_                 character(30)       ,
  DOBS_                 character(4)        ,
  DOBO_                 character(1)        ,
  MAGS_                 character(2)        ,
  OPST_                 character(2)        ,
  KUP1_                 character(30)       ,
  KUP2_                 character(30)       ,
  RAC1_                 character(2)        ,
  RAC2_                 character(30)       ,
  KSIR_                 character(13)       ,
  ESIR_                 character(30)       ,
  ZNAK_                 character(1)        ,
  STAT_                 character(1)        ,
  GRUP_                 character(2)        ,
  ROBS_                 character(5)        ,
  ROBK_                 character(30)       ,
  DOBK_                 character(30)       ,
  ROBN_                 character(50)       ,
  MERA_                 character(3)        ,
  KOLI_                 numeric(16,3)       ,
  CENA_FAK_             numeric(16,2)       ,
  CENA_MPR_             numeric(16,2)       ,
  VRED_MPR_             numeric(16,2)       ,
  TARI_                 character(1)        ,
  PPUP_                 numeric(5,1)        ,
  PPOD_                 numeric(16,2)       ,
  CENA_VPR_             numeric(16,2)       ,
  VRED_VPR_             numeric(16,2)       ,
  RABP_                 numeric(6,2)        ,
  RABD_                 numeric(16,2)       ,
  FLAG_                 character(1)        ,
  FLAG_IME_             character(30)       ,
  UPLATA_               numeric(16,2)       ,
  RACUN_                numeric(16,2)       ,
  KUSUR_                numeric(16,2)       ,
  GOT_                  numeric(16,2)       ,
  CEK_                  numeric(16,2)       ,
  KAR_                  numeric(16,2)       ,
  NAL_                  numeric(16,2)       ,
  INS_                  numeric(16,2)       ,
  VAU_                  numeric(16,2)       ,
  BOT_                  numeric(16,2)       ,
  AVA_                  numeric(16,2)       ,
  AVP_                  numeric(16,2)       ,
  COBA_                 integer             ,
  COK_                  boolean             ,
  UUID_                 character(36)       ,
  __deleted boolean NOT NULL DEFAULT false,
  __record serial NOT NULL,
  __rowversion integer NOT NULL DEFAULT 1,
  __keyversion integer NOT NULL DEFAULT 0,
```

```
  __lock_owner integer NOT NULL DEFAULT 0,
  CONSTRAINT &ckey PRIMARY KEY (__record)
)
WITH (OIDS=FALSE,autovacuum_enabled=true);

ENDTEXT
* "formiran je SQL string 'sqltxt1'
// --- poseban lični kod za svaku od tabli  --- end


// --- zajednički common kod za svaku od tabli  --- start
//     dobija se dodatak za string 'sqltxt'
  ************************************************
  sqltxt2 := SQLstring__(xdb,xse,tablename,tableschema)
  ************************************************
// --- zajednički common kod za svaku od tabli  --- end
// spajanje dva stringa:
  sqltxt := sqltxt1 + sqltxt2
RETURN sqltxt
*#########################################################
ENDIF // IF LEFT(tablename,5)=="kbon_"
*######################################################### END



*" (KASA_BLOK) variable for DBF name in application KBLOK_22.DBF
*######################################################### START
IF LEFT(tablename,6)=="kblok_"
*#########################################################
// --- poseban lični kod za svaku od tabli  --- start
// moraji se formirati PRIVATE zbog upisa varijabli sa & u text into
   PRIVATE ctable := tableschema
   PRIVATE ckey   := tablename + "_pkey"
   PRIVATE caddress  := xdb+"."+xse+"."+tablename

TEXT INTO sqltxt1 WRAP cr // TRIMMED
--
-- Table: &caddress
--
CREATE TABLE &ctable
  (
  BRJ0_               character(12)       ,
  DAT0_               date                ,
  BRJ1_               character(12)       ,
  DAT1_               date                ,
  VAL1_               date                ,
  DOBS_               character(4)        ,
  DOBO_               character(1)        ,
  DOBN_               character(40)       ,
  DOBE_               character(40)       ,
  MBR_                character(30)       ,
  MAGS_               character(2)        ,
  MAGN_               character(25)       ,
  OPST_               character(2)        ,
  ZNAK_               character(1)        ,
  FLAG_               character(1)        ,
  STAT_               character(1)        ,
  DKUR_               numeric(13,2)       ,
  ZBIR_               numeric(15,2)       ,
  ADRESA_             character(30)       ,
```

```
  MESTO_              character(30)       ,
  PIB_                character(30)       ,
  OBNAZIV_            character(30)       ,
  OBADRESA_           character(30)       ,
  OBMESTO_            character(30)       ,
  OBPROD1_            character(30)       ,
  OBPROD2_            character(30)       ,
  OBKUP1_             character(30)       ,
  OBKUP2_             character(30)       ,
  DOBI_               character(4)        ,
  DOBM_               character(40)       ,
  DOBU_               character(40)       ,
  DOBMES_             character(40)       ,
  DOBADR_             character(40)       ,
  DOBTEL_             character(40)       ,
  DOBPIB_             character(40)       ,
  DOBMBR_             character(40)       ,
  DOBU__              character(40)       ,
  __deleted boolean NOT NULL DEFAULT false,
  __record serial NOT NULL,
  __rowversion integer NOT NULL DEFAULT 1,
  __keyversion integer NOT NULL DEFAULT 0,
  __lock_owner integer NOT NULL DEFAULT 0,
  CONSTRAINT &ckey PRIMARY KEY (__record)
)
WITH (OIDS=FALSE,autovacuum_enabled=true);

ENDTEXT
* "formiran je SQL string 'sqltxt1'
// --- poseban lični kod za svaku od tabli  --- end


// --- zajednički common kod za svaku od tabli  --- start
//     dobija se dodatak za string 'sqltxt'
  ****************************************************
  sqltxt2 := SQLstring__(xdb,xse,tablename,tableschema)
  ****************************************************
// --- zajednički common kod za svaku od tabli  --- end
// spajanje dva stringa:
  sqltxt := sqltxt1 + sqltxt2
RETURN sqltxt
*#########################################################
ENDIF // IF LEFT(tablename,6)=="kblok_"
*######################################################### END



*" (STO) variable for DBF name in application X1.DBF,X2.DBF... X100.DBF
*######################################################### START
IF LEFT(tablename,1)=="x"
*#########################################################
// --- poseban lični kod za svaku od tabli  --- start
// moraji se formirati PRIVATE zbog upisa varijabli sa & u text into
   PRIVATE ctable := tableschema
   PRIVATE ckey   := tablename + "_pkey"
   PRIVATE caddress  := xdb+"."+xse+"."+tablename

TEXT INTO sqltxt1 WRAP cr // TRIMMED
--
-- Table: &caddress
```

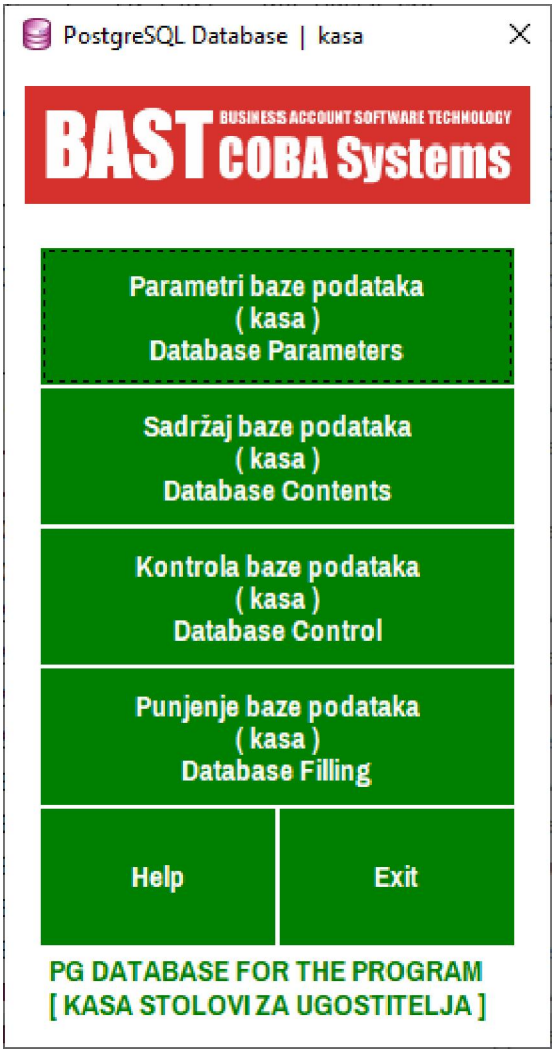
```
--
CREATE TABLE &ctable
  (
  BRJ0_                 character(12)       ,
  DAT0_                 date                ,
  VRE0_                 character(10)       ,
  BRJ1_                 character(30)       ,
  DAT1_                 character(30)       ,
  BRJ2_                 character(30)       ,
  DAT2_                 character(30)       ,
  VAL1_                 character(30)       ,
  DOBS_                 character(4)        ,
  DOBO_                 character(1)        ,
  MAGS_                 character(2)        ,
  OPST_                 character(2)        ,
  KUP1_                 character(30)       ,
  KUP2_                 character(30)       ,
  RAC1_                 character(2)        ,
  RAC2_                 character(30)       ,
  KSIR_                 character(13)       ,
  ESIR_                 character(30)       ,
  ZNAK_                 character(1)        ,
  STAT_                 character(1)        ,
  GRUP_                 character(2)        ,
  ROBS_                 character(5)        ,
  ROBK_                 character(30)       ,
  DOBK_                 character(30)       ,
  ROBN_                 character(50)       ,
  MERA_                 character(3)        ,
  KOLI_                 numeric(16,3)       ,
  CENA_FAK_             numeric(16,2)       ,
  CENA_MPR_             numeric(16,2)       ,
  VRED_MPR_             numeric(16,2)       ,
  TARI_                 character(1)        ,
  PPUP_                 numeric(5,1)        ,
  PPOD_                 numeric(16,2)       ,
  CENA_VPR_             numeric(16,2)       ,
  VRED_VPR_             numeric(16,2)       ,
  RABP_                 numeric(6,2)        ,
  RABD_                 numeric(16,2)       ,
  FLAG_                 character(1)        ,
  FLAG_IME_             character(30)       ,
  UPLATA_               numeric(16,2)       ,
  RACUN_                numeric(16,2)       ,
  KUSUR_                numeric(16,2)       ,
  GOT_                  numeric(16,2)       ,
  CEK_                  numeric(16,2)       ,
  KAR_                  numeric(16,2)       ,
  NAL_                  numeric(16,2)       ,
  INS_                  numeric(16,2)       ,
  VAU_                  numeric(16,2)       ,
  BOT_                  numeric(16,2)       ,
  AVA_                  numeric(16,2)       ,
  AVP_                  numeric(16,2)       ,
  COBA_                 integer             ,
  COK_                  boolean             ,
  UUID_                 character(36)       ,
  __deleted boolean NOT NULL DEFAULT false,
  __record serial NOT NULL,
  __rowversion integer NOT NULL DEFAULT 1,
```

```
  __keyversion integer NOT NULL DEFAULT 0,
  __lock_owner integer NOT NULL DEFAULT 0,
  CONSTRAINT &ckey PRIMARY KEY (__record)
)
WITH (OIDS=FALSE,autovacuum_enabled=true);

ENDTEXT
* "formiran je SQL string 'sqltxt1'
// --- poseban lični kod za svaku od tabli  --- end


// --- zajednički common kod za svaku od tabli  --- start
//     dobija se dodatak za string 'sqltxt'
  ************************************************
  sqltxt2 := SQLstring__(xdb,xse,tablename,tableschema)
  ************************************************
// --- zajednički common kod za svaku od tabli  --- end
// spajanje dva stringa:
  sqltxt := sqltxt1 + sqltxt2
RETURN sqltxt
*#########################################################
ENDIF // IF LEFT(tablename,1)=="x"
*######################################################### END


RETURN NIL


*************************************************************************
FUNCTION SQLstring__(xdb,xse,tablename,tableschema)
*************************************************************************
LOCAL txt := "", cr := chr(13)+chr(10), sqltxt2 := ""

// --- zajednički common kod za svaku od tabli  --- start
// --- shared common code for each of the tables --- start


// moraji se formirati PRIVATE zbog upisa varijabli sa & u text into
// you have to form PRIVATE because of writing variables with & in text into
   PRIVATE ctable := tableschema
   PRIVATE ckey   := tablename + "_pkey"

   PRIVATE ;
   index_deleted          := tablename+"__deleted"        ,;
   index_record           := tablename+"__record"         ,;
   triger_isam_rowversion := tablename+"_isam_rowversion" ,;
   triger_isam_tablemeta  := tablename+"_isam_tablemeta"




TEXT INTO sqltxt2 WRAP cr // TRIMMED
-- ******************************************
-- O W N E R:
-- ******************************************
-- mora se postaviti user na 'postgres' (superuser)
-- morate biti logovani na database kao superuser
-- user must be set to 'postgres' (superuser)
-- you must be logged into the database as superuser
```

```
   ALTER TABLE &ctable OWNER TO postgres;



-- ********************************************
-- I N D E K S I
-- ********************************************
-- * index_deleted
-- Index: &index_deleted
-- DROP INDEX &index_deleted;

   CREATE INDEX &index_deleted
   ON &ctable
   USING btree
   (__deleted);

-- * index_record
-- Index: &index_record
-- DROP INDEX &index_record;

   CREATE INDEX &index_record
   ON &ctable
   USING btree
   (__record);


-- ********************************************
-- TRIGGERS AND TRIGER FUNCTIONS:
-- ********************************************

-- * triger_isam_rowversion
-- (uvek obavezno kod upsize tehnologije)
-- Trigger: &triger_isam_rowversion on &ctable
-- DROP TRIGGER &triger_isam_rowversion ON &ctable;

   CREATE TRIGGER &triger_isam_rowversion
   BEFORE UPDATE
   ON &ctable
   FOR EACH ROW
   EXECUTE PROCEDURE isam_rowversion_update();

-- * triger_isam_tablemeta
-- (uvek obavezno kod upsize tehnologije)
-- Trigger: &triger_isam_tablemeta on &ctable
-- DROP TRIGGER &triger_isam_tablemeta ON &ctable;

   CREATE TRIGGER &triger_isam_tablemeta
   AFTER INSERT OR UPDATE OR DELETE
   ON &ctable
   FOR EACH ROW
   EXECUTE PROCEDURE isam_tablemeta_update();

ENDTEXT

// --- shared common code for each of the tables --- end

RETURN sqltxt2
```

IZGLED I SADRŽAJ APLIKACIJE
C-POSTGRESQL-DATABASE.EXE -> KASA-STOLOVI-DATABASE.EXE

APPLICATION APPEARANCE AND CONTENT
C-POSTGRESQL-DATABASE.EXE -> KASA-STOLOVI-DATABASE.EXE



Sadržaj baze podataka
(kasa)
Database Contents

List all Database in Server [Cluster]

Redni broj	Database name
1	coba
2	kasa
3	postgres
4	template0
5	template1
6	test

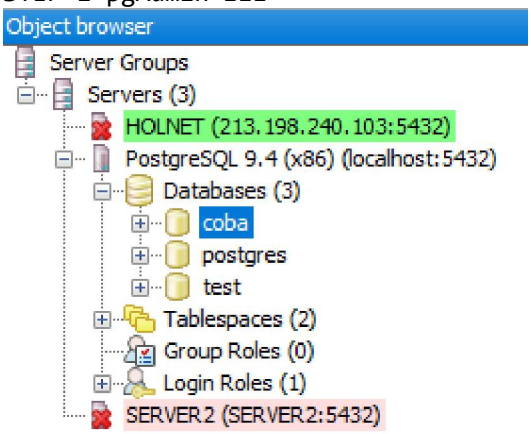
List all Schemas in database [kasa]

Redni broj	Schema name
1	information_schema
2	pg_catalog
3	pg_temp_1
4	pg_toast
5	pg_toast_temp_1
6	public

List all Tables in Scheme [kasa.public]

Redni broj	Table name
1	alaska-software.isam.orders
2	alaska-software.isam.tables
3	alaska-software.system.connectio
4	coba_systems
5	kasa_22
6	kaso_22
7	kblok_22
8	kbon_22
9	kroba_11
10	kup
11	x1
12	x10
13	x100
14	x11
15	x12
16	x13
17	x14

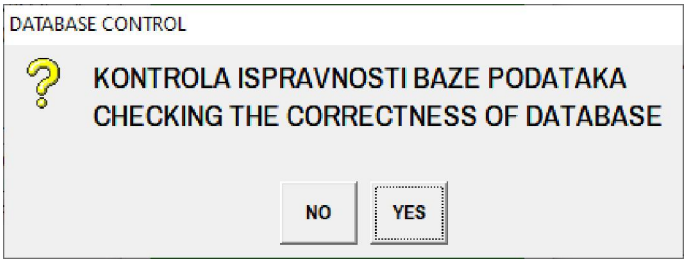
STEP 1 pgAdmin III



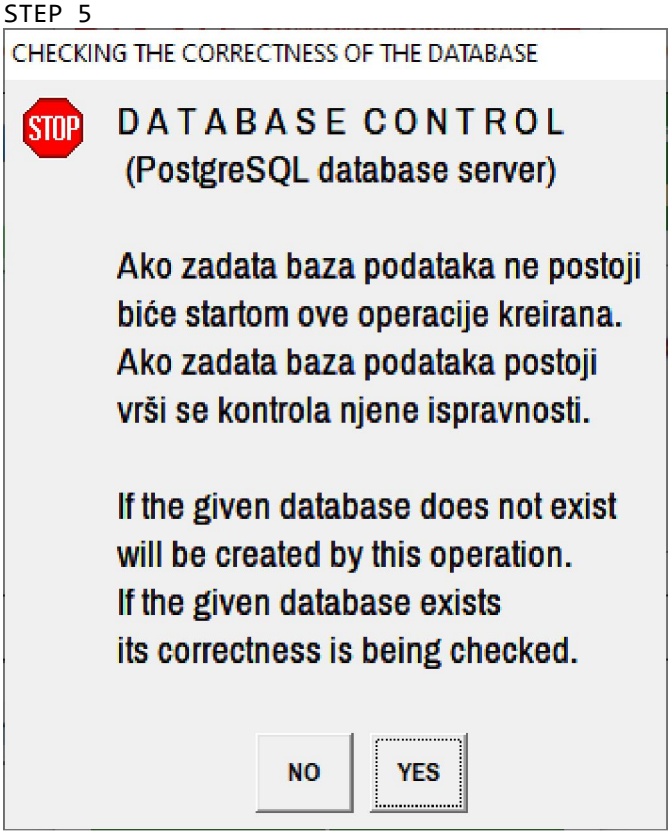
STEP 2 START:
KASA-STOLOVI-DATABASE.EXE
(C-POSTGRESQL-DATABASE.EXE)

STEP 3

STEP 4

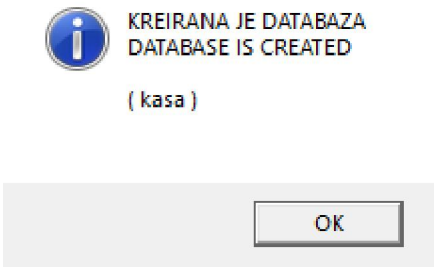


STEP 5

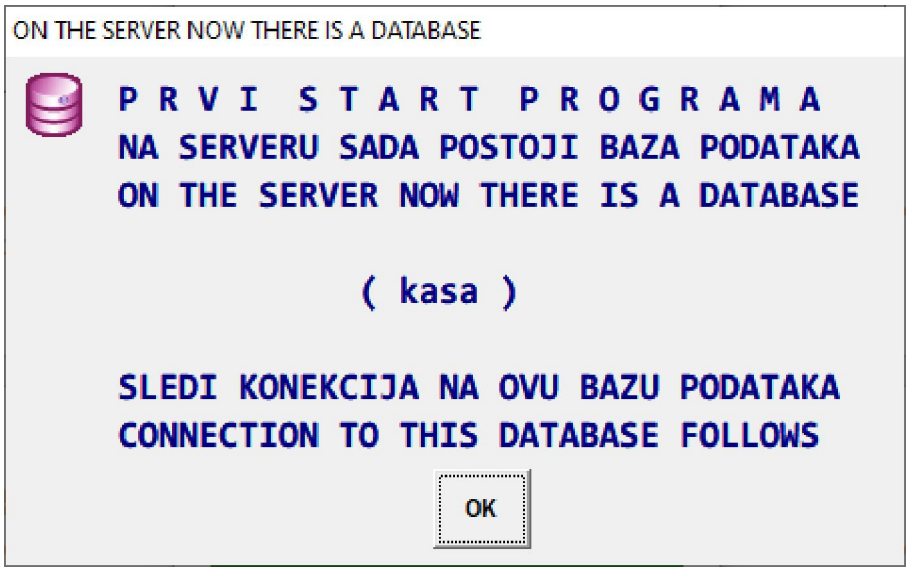


STEP 6

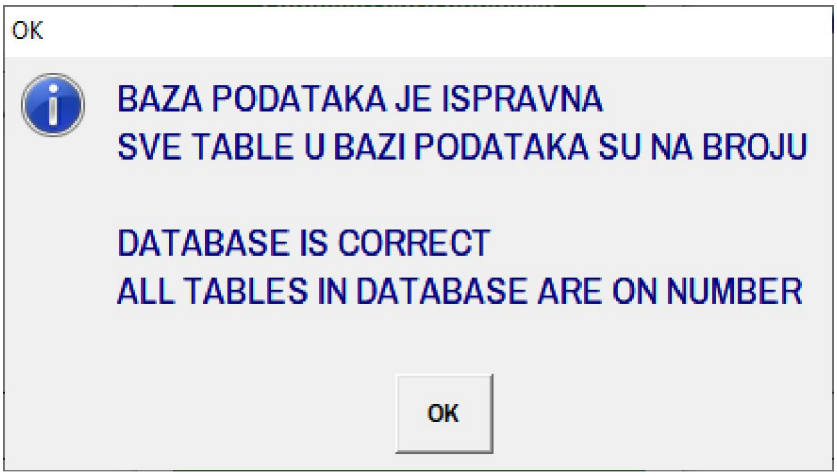
STEP 7



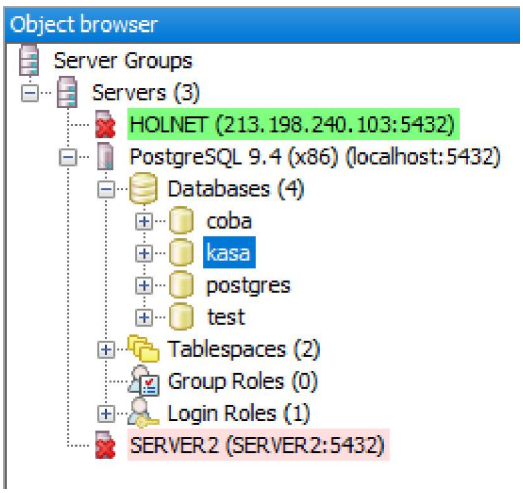
STEP 8



STEP 9

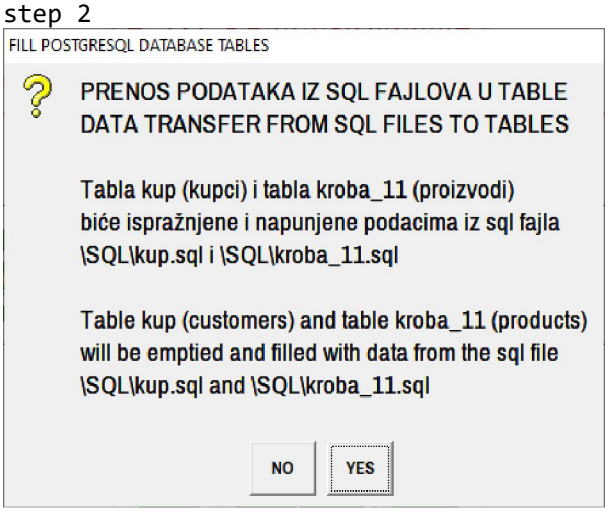


STEP 10 pgAdmin III

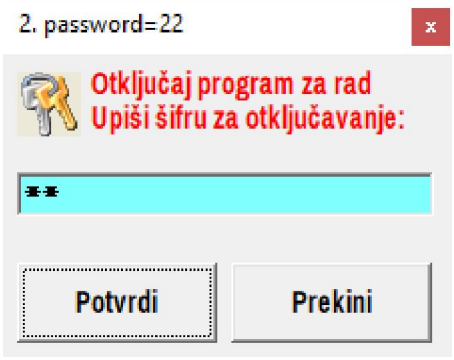


step 1

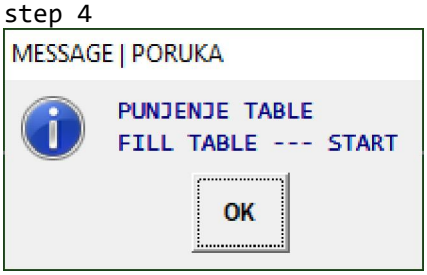
step 2



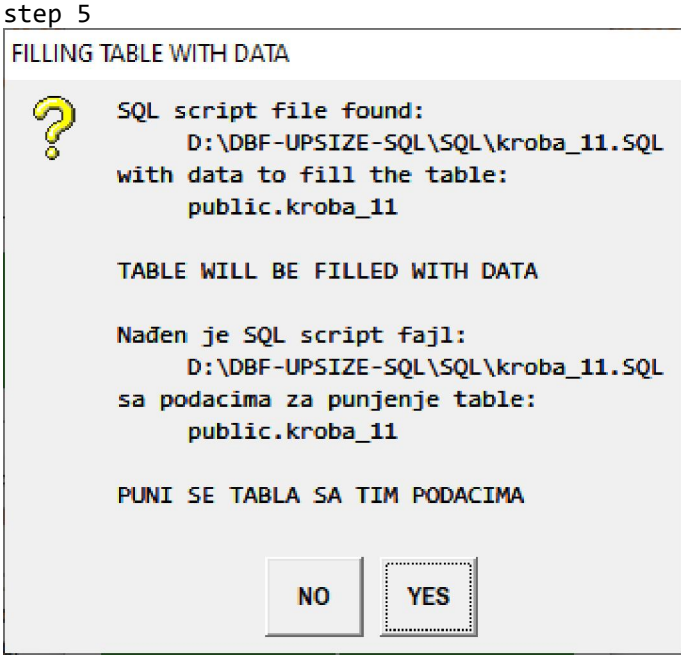
step 3



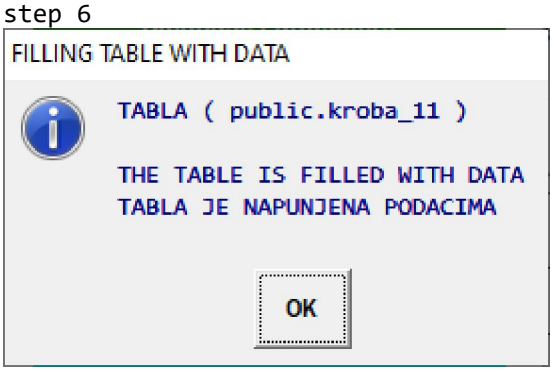
step 4



step 5

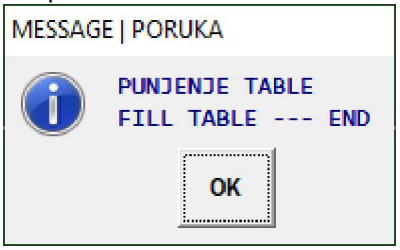


step 6



step 7

step 8



Note:

U delu 4 ove knjige biće opisan i dokumentovan program
Part 4 of this book will describe and document the program
C-PGDB.DLL (common procedures and functions for applications)

U delu 5 ove knjige biće opisana i dokumentovana poslovna aplikacija
FISKALNA REGISTAR KASA ZA RESTORAN
KASA-STOLOVI.EXE
koja kao Alaska Xbase++ (and eXpress++) PGDBE aplikacija koristi postgreSQL bazu podataka sa UPSIZE tablama

In part 5 of this book, the business application will be described and documented
FISCAL CASH REGISTER FOR THE RESTAURANT
KASA-STOLOVI.EXE
which as an Alaska Xbase++ (and eXpress++) PGDBE application uses a postgreSQL database with UPSIZE tables