CSYSTEMS™

PROGRAMSKI PAKET ZA KNJIGOVODSTVO

COBA Systems

Alaska Xbase++
DBF to PostgreSQL database application

(PART 4)
PostgreSQL Database with DBF Upsize tables
ISAM and SQL comunications

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)

U delu 4 ove knjige opisana je i dokumentovana je bublioteka funkcija za komunikaciju sa Alaska Xbase++ PostgreSQL upsize bazom podataka.
Part 4 of this book describes and documents the library of functions for communicating with the Alaska Xbase++ PostgreSQL upsize database.

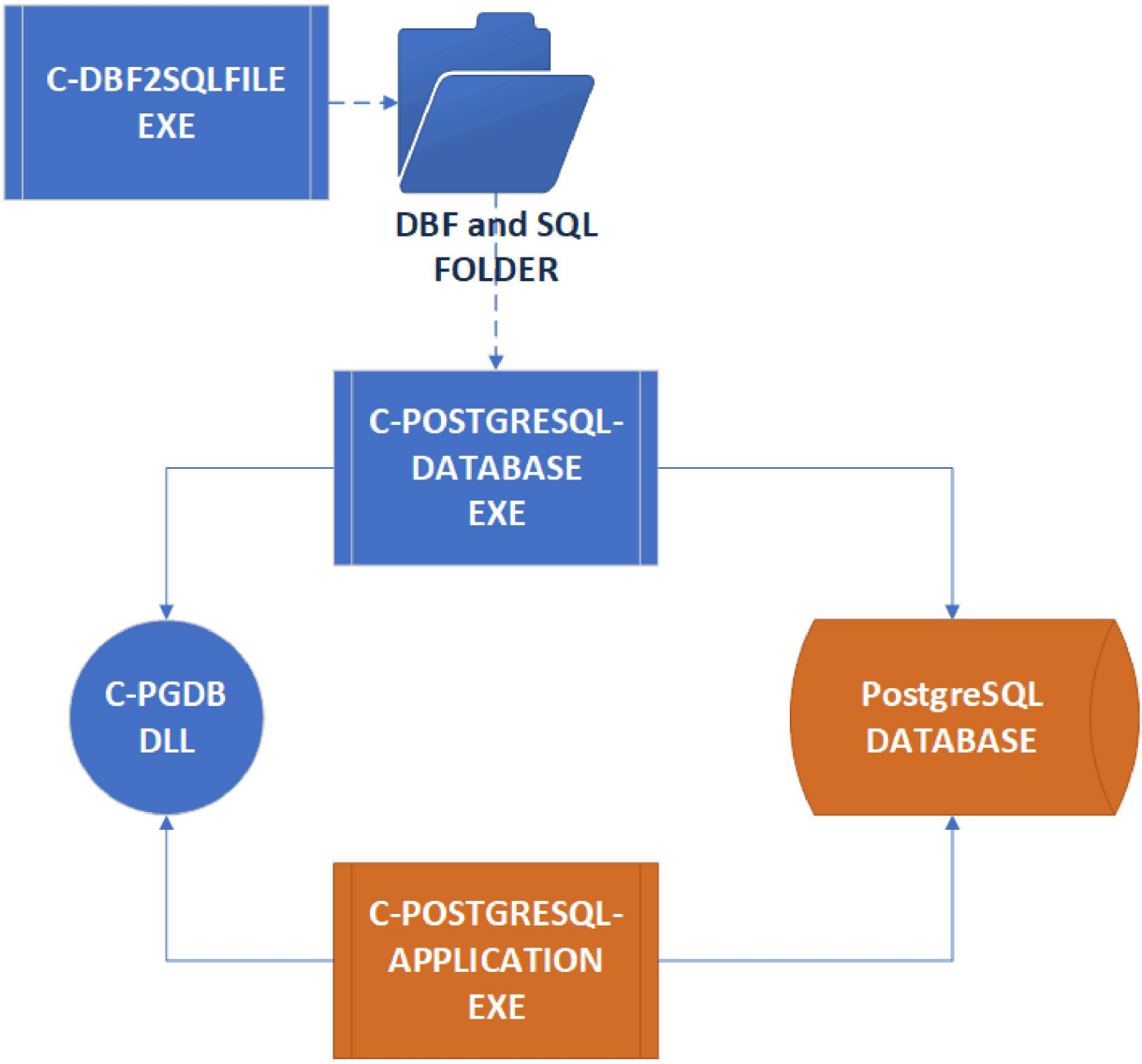
C-PGDB.DLL (common procedures and functions for applications)

Ovaj program se sastoji od skupa common funkcija i isporučuje se uz postgreSQL bazu podataka svake poslovne EXE aplikacije. Služi za komunikaciju sa bazom podataka (sa bilo kojom PostgreSQL bazom podataka).
This program consists of a set of common functions and is supplied with the PostgreSQL database of every enterprise EXE application. It serves for communication with the database (with any PostgreSQL database).

U delu 5 ove knjige biće opisana i dokumentovana poslovna aplikacija koja je sa Alaska Xbase++ DBF ISAM baze podataka prneta na Alaska Xbase++ PostgreSQL upsize bazu podataka.
In part 5 of this book, a business application that was transferred from the Alaska Xbase++ DBF ISAM database to the Alaska Xbase++ PostgreSQL upsize database will be described and documented.
KASA-STOLOVI.EXE (Fiscal register cash register for a restaurant)

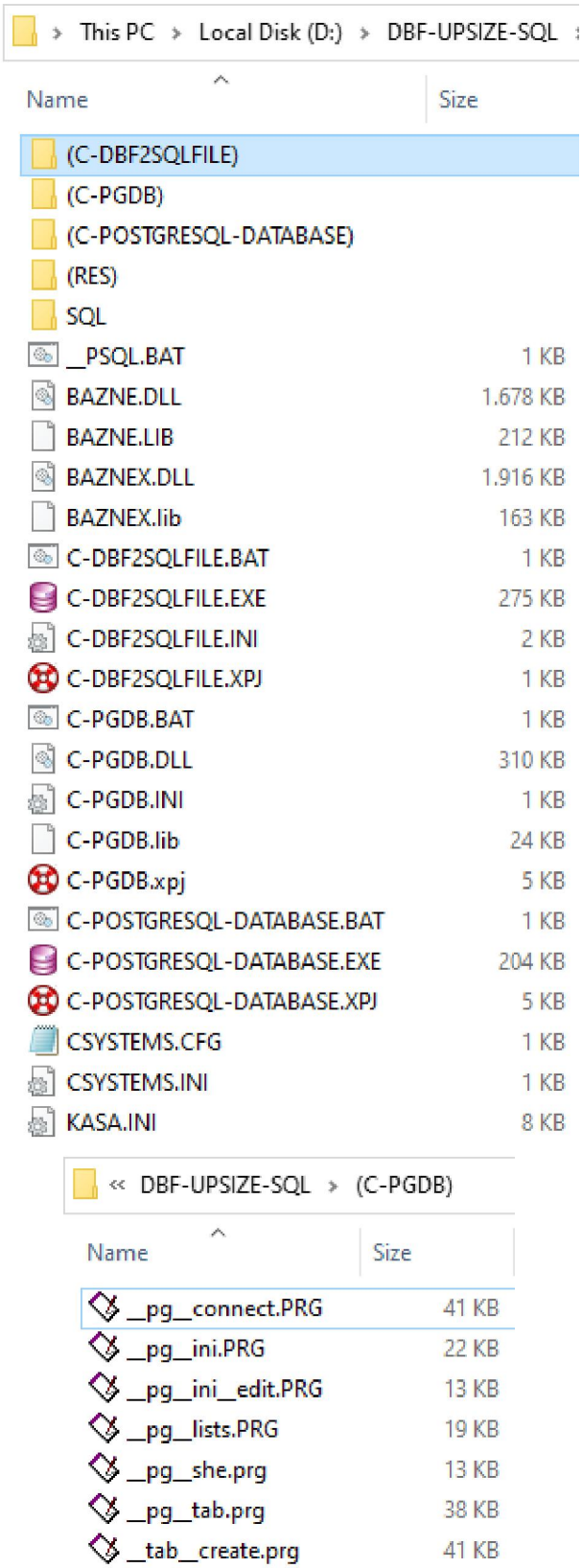
PRILOG: ŠEMA ORGANIZACIJE PROGRAMA ZA RAD SA UPSIZE POSTGRESQL DATABAZOM

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-PGDB.XPJ --- START

[PROJECT]
    DEBUG          = no
    VERSION        = 2.3
    C-PGDB.XPJ

[C-PGDB.XPJ]
    C-PGDB.DLL

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

// $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

    __pg__ini.obj
    __pg__ini__edit.obj
    __pg__connect.obj
    __pg__lists.obj

    __pg__she.obj
    __pg__tab.obj

    __tab__create.obj

    C-PGDB.DEF
    C-PGDB.LIB

// $STOP-AUTODEPEND

    (RES)\C-PGDB.RES

    xbtbase1.lib  // XbToolsIII++
```

```
    dclipx.lib    // eXpress++
    bazne.lib     // COBA Systems
   //------------------------------
   // C-PGDB.INI
   // KASA.INI
   //------------------------------
    (C-PGDB)\__pg__ini.prg
    (C-PGDB)\__pg__ini__edit.prg
    (C-PGDB)\__pg__connect.prg
    (C-PGDB)\__pg__lists.prg

    (C-PGDB)\__pg__she.prg
    (C-PGDB)\__pg__tab.prg
//-------------------------------------------------------------------------
// program to create all application tables --- production version

    (C-PGDB)\__tab__create.prg

// Each of tabless is created, which is by its name and with its own
// sql string for creation written in the PRG code (hardcoded) of the
// module __tab__create__app.prg which is an integral part of every *.EXE
// application which links to C_PGDB.DLL

// program to create all application tables --- production version
//-------------------------------------------------------------------------
// C-PGDB.XPJ --- END

:: C-PGDB.BAT --- START
PBUILD C-PGDB.XPJ > __________.TXT
NOTEPAD __________.TXT
:: C-PGDB.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
```

__PG__INI.PRG

```
////////////////////////////////////////////////////////////////////////
//                                                                    //
//                                                                    //
//   __pg__ini.PRG
//                                                                    //
//   25-10-2023                                                       //
//                                                                    //
//   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.        //
//                                                                    //
//                                                                    //
////////////////////////////////////////////////////////////////////////



* APP FUNCTION in __pg__ini.PRG
*
*  FUNCTION __pg__ini(akcija)
*    --- koristi|uses C-PGDB.INI
*    --- koristi|uses KASA.INI
*  STATIC FUNCTION __pg__fix()
*  FUNCTION __pg__see()
*  FUNCTION __pg__connection_string()
*  FUNCTION __pg__kasa_MAG_KAS() // --> FORMIRA PUBLIC KASE: _MAG_, _KAS_
*
*  Svaka od ovih funkcija može se pozvati kao samostalna funkcija:
*  Each of these functions can be called as a standalone function:
*



*-----------------------------------
#include "Xbp.ch"
#include "AppEvent.ch"
#include "Common.ch"
*-----------------------------------



/// <descrioption>
/// O B A V E Z N E  P U B L I C  A P L I K A C I J E
/// Formiraju se samo u ovoj funkciji i nigde više u ovoj aplikaciji.
/// Ili su hardkodovane FIXED u kodu, ili se učitavaju iz C-PGSQL.INI
/// Funkcija se obavezno poziva na startu svake aplikacije iz main()
/// procedure i formira PUBLIC varijable aplikacije za rad sa PostgreSQL
/// database serverom. Bez poziva ove funkcije aplikacija neće raditi.
/// </description>
```

```
/// <description>
/// MANDATORY PUBLIC APPLICATIONS
/// They are only formed in this function and nowhere else in this application.
/// They are either hardcoded FIXED in the code, or they are loaded from C-PGSQL.INI
/// The function must be called at the start of each application from main()
/// procedures and forms PUBLIC application variables for working with PostgreSQL
/// database server. Without calling this function, the application will not work.
/// </description>
*************************************************************************
FUNCTION __pg__ini(akcija)
*************************************************************************
* akcija = "DEFAULT"
* akcija = "FIXED"
* akcija = NIL
*
*
* PUBLIC APLIKACIJE:
* _server_
* _uid_
* _pwd_
* _db_
* _she_
* _tab_
* _oSession_
* _version_
* _dblist_
* _shelist_
* _tablist_
*
* _cdbc_
* _lAlert_
* _lMsg_

* ---

LOCAL _xserver_,_xuid_,_xpwd_,_xdb_,_xshe_,_xtab_
LOCAL _xlAlert_, _xlMsg_

 ****************************************************
 PUBLIC cobapgsqlini := gde_exe()+"\C-PGDB.INI"
 ****************************************************

* ---

  * D E F A U L T
  * work database --- preuzima se iz INI fajla
   PUBLIC _server_ := "localhost"    // host name      INI
   PUBLIC _uid_    := "postgres"     // user name      INI
   PUBLIC _pwd_    := "coba#1949"    // password       INI
   PUBLIC _db_     := "coba"         // database name  INI

  * Schemes and Tables --- preuzima se iz INI fajla
   PUBLIC _she_    := "public"       // scheme name    INI
   PUBLIC _tab_    := "coba_systems" // table name     INI licenca

  //  dobija se tek posle DbeSys()
  //  i posle konekcije na posgreSQL server ili neku od baza podataka

  * connection --- generiše se iz aplikacije posle konekcije
   PUBLIC _oSession_  := NIL
```

```
   PUBLIC _version_   := ""

  * liste --- generiše se iz aplikacije posle konekcije
   PUBLIC _dblist_    := {} // lista svih baza podataka na serveru  _server_
   PUBLIC _shelist_   := {} // lista svih shema u bazi podataka _db_
   PUBLIC _tablist_   := {} // lista svih tabli u bazi podataka _db_ u schema _she_

   PUBLIC _cdbc_   := "( "+_db_+" )" // za prikaz u dijalozima
                                     // koriguj posle preuzimanja _db_
   PUBLIC _lAlert_     := .F.        // za prikaz user info poruka (neobavezne poruke)
   PUBLIC _lMsg_       := .F.        // za prikaz test info poruka (u razvoju aplikacije)
   PUBLIC _lDBcontrol_ := .F.        // izvršava se T/F kontrola postojanja baze podataka
                                     // pozivom funkcije: __pg__connect_app()/prg

   * dobija se:
   * PUBLIC _cConnStr_
   * PUBLIC _cConnStrTest_
   * iz funkcije:
   *************************
   __pg__connection_string()   // __pg__ini.prg // i ovde i na kraju funkcije !
   *************************                   // both here and at the end of the function !



   * EXTRA PARAMETRI ZA PROGRAM KASA:
   * PUBLIC _MAG_ // broj prodavnice '01' do '99' default '11'
   * PUBLIC _KAS_ // broj kase u firmi '01' do '99' default '22'
   *********************
   __pg__kasa_MAG_KAS() // --> PUBLIC _MAG_, PUBLIC _KAS_
   *********************


   IF akcija=="DEFAULT"
      RETURN NIL
   ENDIF
   IF akcija=="FIXED"
      *************
      __pg__fix() // ovde
      *************
      RETURN NIL
   ENDIF

* ---



* --- SERVER NAME
   _xserver_ := INI_READ("C","DATABASE","_server_",cobapgsqlini)
   IF EMPTY(_xserver_)
      INI_WRITE("C","DATABASE","_server_",_server_,cobapgsqlini)  // D E F A U L T
      _server_ := INI_READ("C","DATABASE","_server_",cobapgsqlini)
   ELSE
      _server_ := _xserver_
   ENDIF
* ---
* --- USER NAME
   _xuid_ := INI_READ("C","DATABASE","_uid_",cobapgsqlini)
   IF EMPTY(_xuid_)
      INI_WRITE("C","DATABASE","_uid_",_uid_,cobapgsqlini)  // D E F A U L T
      _uid_ := INI_READ("C","DATABASE","_uid_",cobapgsqlini)
```

```
   ELSE
      _uid_ := _xuid_
   ENDIF
* ---
* --- PASSWORD NAME
   _xpwd_ := INI_READ("C","DATABASE","_pwd_",cobapgsqlini)
   IF EMPTY(_xpwd_)
      INI_WRITE("C","DATABASE","_pwd_",_pwd_,cobapgsqlini)  // D E F A U L T
      _pwd_ := INI_READ("C","DATABASE","_pwd_",cobapgsqlini)
   ELSE
      _pwd_ := _xpwd_
   ENDIF
* ---
* --- DATABASE NAME
   _xdb_ := INI_READ("C","DATABASE","_db_",cobapgsqlini)
   IF EMPTY(_xdb_)
      INI_WRITE("C","DATABASE","_db_",_db_,cobapgsqlini)  // D E F A U L T
      _db_ := INI_READ("C","DATABASE","_db_",cobapgsqlini)
   ELSE
      _db_ := _xdb_
   ENDIF
* ---

   PUBLIC _cdbc_   := "( "+_db_+" )" // za prikaz u dijalozima


* ---  SCHEME NAME
   _xshe_ := INI_READ("C","DATABASE","_she_",cobapgsqlini)
   IF EMPTY(_xshe_)
      INI_WRITE("C","DATABASE","_she_",_she_,cobapgsqlini)  // D E F A U L T
      _she_ := INI_READ("C","DATABASE","_she_",cobapgsqlini)
   ELSE
      _she_ := _xshe_
   ENDIF
* ---
* --- TABLE NAME
   _xtab_ := INI_READ("C","DATABASE","_tab_",cobapgsqlini)
   IF EMPTY(_xtab_)
      INI_WRITE("C","DATABASE","_tab_",_tab_,cobapgsqlini)  // D E F A U L T
      _tab_ := INI_READ("C","DATABASE","_tab_",cobapgsqlini)
   ELSE
      _tab_ := _xtab_
   ENDIF
* ---

* --- ALERT
   _lAlert_ := INI_READ("L","DATABASE","_lAlert_",cobapgsqlini)
   // default = .F. ili .T.
* ---

* --- MSG
   _lMsg_ := INI_READ("L","DATABASE","_lMsg_",cobapgsqlini)
  // default = .F. ili .T.
* ---

* --- DBCONTROL
   _lDBcontrol_ := INI_READ("L","DATABASE","_lDBcontrol_",cobapgsqlini)
  // default = .F. ili .T.
* ---
```

```
// upiši ove parametre u ini ako ih tamo nema upisanih
IF AT("_lAlert_",memoread(cobapgsqlini))=0
   INI_WRITE("L","DATABASE","_lAlert_",.F.,cobapgsqlini)
ENDIF
IF AT("_lMsg_",memoread(cobapgsqlini))=0
   INI_WRITE("L","DATABASE","_lMsg_",.F.,cobapgsqlini)
ENDIF
IF AT("_lDBcontrol_",memoread(cobapgsqlini))=0
   INI_WRITE("L","DATABASE","_lDBcontrol_",.F.,cobapgsqlini)
ENDIF

* dobija se:
* PUBLIC _cConnStr_
* PUBLIC _cConnStrTest_
* iz funkcije:
*************************
__pg__connection_string()   // __pg__ini.prg
*************************



RETURN NIL


/// <description>
/// Kada se u aplikaciji fiksiraju parametri za rad sa posgreSQL
/// serverom, tako da ih korisnik ne može menjati, tada se upis
/// parametara vrši u kod aplikacije (hardcoded) u ovu funkciju.
/// </description>
/// <description>
/// When parameters for working with posgreSQL are fixed in the application
/// by the server, so that the user cannot change them, then the entry
/// parameters are hardcoded into this function.
/// </description>
*****************************************************************************
STATIC FUNCTION __pg__fix()
*****************************************************************************

  * F I X E D

   PUBLIC _server_ := "localhost"    // host name
   PUBLIC _uid_    := "postgres"     // user name
   PUBLIC _pwd_    := "coba#1949"    // password
   PUBLIC _db_     := "test"         // database name

  * Schemes and Tables --- preuzima se iz INI fajla
   PUBLIC _she_    := "2023"         // scheme name
   PUBLIC _tab_    := "coba_systems" // table name - licenca

  //  dobija se tek posle DbeSys()
  //  i posle konekcije na posgreSQL server ili neku od baza podataka

  * connection --- generiše se iz aplikacije posle konekcije
   PUBLIC _oSession_  := NIL
   PUBLIC _version_   := ""

  * liste --- generiše se iz aplikacije posle konekcije
   PUBLIC _dblist_    := {} // lista svih baza podataka na serveru  _server_
   PUBLIC _shelist_   := {} // lista svih shema u bazi podataka _db_
   PUBLIC _tablist_   := {} // lista svih tabli u bazi podataka _db_ u schema _she_
```

```
  * generisano i ne menjano u __pg__ini(akcija)
   PUBLIC _cdbc_   := "( "+_db_+" )" // za prikaz u dijalozima
   PUBLIC _lAlert_ := .F.        // za prikaz user info poruka (neobavezne poruke)
   PUBLIC _lMsg_   := .F.        // za prikaz test info poruka (u razvoju aplikacije)
   PUBLIC _lDBcontrol_ := .F.    // izvršava se T/F kontrola postojanja baze podataka
                                 // pozivom funkcije: __pg__connect_app()/prg


   * EXTRA PARAMETRI ZA PROGRAM KASA:
   * PUBLIC _MAG_ // broj prodavnice '01' do '99' default '11'
   * PUBLIC _KAS_ // broj kase u firmi '01' do '99' default '11'
   *********************
   __pg__kasa_MAG_KAS() // --> PUBLIC _MAG_, PUBLIC _KAS_
   *********************


* dobija se:
* PUBLIC _cConnStr_
* PUBLIC _cConnStrTest_
* iz funkcije:
**************************
__pg__connection_string()   // __pg__ini.prg
**************************


RETURN NIL


/// <descrioption>
/// Pregled PostgreSQL Database Server parameters odnosno
/// pregled aktuelnih PUBLIC VARIJABLI APLIKACIJE
/// </description>
/// <description>
/// Overview of PostgreSQL Database Server parameters ie
/// overview of the current APPLICATION PUBLIC VARIABLES
/// </description>
*****************************************************************************
FUNCTION __pg__see(nn)
*****************************************************************************
LOCAL cr:=chr(59), txt:=""
LOCAL xcConnStr, xcConnStrTest


*************************
*** pre ove funkcije mora se iz main() pozvati:
***__pg__ini()                  // __pg__ini.prg
*** u __pg__ini)= poziva se:
***__pg__connection_string()   // __pg__ini.prg
*** i dobija se:
*** PUBLIC _cConnStr_
*** PUBLIC _cConnStrTest_

*** before this function, main() must be called:
***__pg__ini() // __pg__ini.prg
*** in __pg__ini)= is called:
***__pg__connection_string() // __pg__ini.prg
*** and we get:
*** PUBLIC _cConnStr_
*** PUBLIC _cConnStrTest_
```

```
*************************


IF nn==1 // razvojna test-varijanta  (prikaz za programera)

      xcConnStr     := strtran( _cConnStr_     ,";","|")   // zbog c__procitaj()
      xcConnStrTest := strtran( _cConnStrTest_ ,";","|")   // zbog c__procitaj()

      txt:=;
      "-------------------------- "                          +cr+;
      "C-PGDB.INI"                                    +cr+;
      "PgSQL Version     _version_ " + var2char(_version_)    +cr+;
      "Host name          _server_ " + _server_               +cr+;
      "User name             _uid_ " + _uid_                  +cr+;
      "Password              _pwd_ " + _pwd_                  +cr+;
      "Database name         _db_  " + _db_                   +cr+;
      "Scheme name           _she_ " + _she_                  +cr+;
      "Table name            _tab_ " + _tab_                  +cr+;
      "-------------------------- "                          +cr+;
      "StrKonekcije _cConnStrTest_ " + xcConnStrTest          +cr+;
      "StrKonekcije     _cConnStr_ " + xcConnStr              +cr+;
      "ObjSesija        _oSession_ " + var2char(_oSession_)   +cr+;
      "-------------------------- "                          +cr+;
      "List Databases     _dblist_ " + IIF( len(_dblist_) =0, var2char(_dblist_),  _dblist[1] )
+cr+;
      "List Schemes      _shelist_ " + IIF( len(_shelist_)=0, var2char(_shelist_), _shelist[1]
)  +cr+;
      "List Tables       _tablist_ " + IIF( len(_tablist_)=0, var2char(_tablist_), _tablist[1]
)  +cr+;
      "-------------------------- "                          +cr+;
      "Database (name)      _cdbc_ " + _cdbc_                +cr+;
      "lAlert (user)      _lAlert_ " + var2char(_lAlert_)    +cr+;
      "lMsg   (test)        _lMsg_ " + var2char(_lMsg_)      +cr+;
      "lDBcontrol     _lDBcontrol_ " + var2char(_lDBcontrol_) +cr+;
      "-------------------------- "                          +cr+;
      "FISKALNA KASA               "                          +cr+;
      "Broj Prodavnice       _MAG_ " + var2char(_MAG_)        +cr+;
      "Broj kase             _KAS_ " + var2char(_KAS_)        +cr+;
      "-------------------------- "                          +cr+;
      ""
else // produkciona varijanta (prikaz za korisnika)

      txt:=;
      "C-PGDB.INI "                          +cr+;
      "------------------ "                          +cr+;
      "PgSQL Version      " + var2char(_version_)    +cr+;
      "Host name          " + _server_               +cr+;
      "User name          " + _uid_                  +cr+;
      "Password           " + _pwd_                  +cr+;
      "Database name      " + _db_                   +cr+;
      "Scheme name        " + _she_                  +cr+;
      "------------------ "                          +cr+;
      "Alert   (user)     " + var2char(_lAlert_)     +cr+;
      "Message (test)     " + var2char(_lMsg_)       +cr+;
      "Start DBcontrol    " + var2char(_lDBcontrol_) +cr+;
      "------------------ "                          +cr+;
      "FISKALNA KASA      "                          +cr+;
      "Broj Prodavnice    " + var2char(_MAG_)        +cr+;  // cObjekat_Kase := _MAG_
      "Broj kase          " + var2char(_KAS_)        +cr+;  // cBroj_Kase    := _KAS_
      ""
```

```
    *  "----------------- "                          +cr+;
    *  "Connection String:"                          +cr+;
    *  _cConnStr_                                    +cr+;

endif

ec_k()
c__procitaj(txt,"Current PostgreSQL Database Server parameters")

RETURN NIL


/// <descrioption>
/// KONEKCIONI STRING NA PostgreSQL DATABASE SERVER
/// cConnStr
/// konekcioni string na PostgreSQL server na Database name DB=_db_
/// cConnStrTest
/// konekcioni string na PostgreSQL server bez Database
/// radi test provere da li na serveru postoji Database name DB=_db_ !
/// </description>
/// <description>
/// CONNECTION STRING TO PostgreSQL DATABASE SERVER
/// cConnStr
/// connection string to PostgreSQL server on Database name DB=_db_
/// cConnStrTest
/// connection string to PostgreSQL server without Database
/// to test whether there is a Database name DB=_db_ on the server!
/// </description>
*********************************************************************
FUNCTION __pg__connection_string()
*********************************************************************

  // Assemble connection string.
  // Sastavljanje connection string-a (string za vezu sa serverom)
  * _cConnStr_ := "DBE=pgdbe;"        // Data Base Engine = PGDBE (PGDBE.DLL)
  * _cConnStr_ += "UID=postgres;"     // user name (super user) baze podataka
  * _cConnStr_ += "PWD=coba#1949;"    // password baze podataka
  * _cConnStr_ += "SERVER=localhost;" // serverski računar u kome je baza podataka
  * _cConnStr_ += "DB=coba;"          // naziv baze podataka na serveru

  PUBLIC _cConnStr_ // konekcija na PostgreSQL server YES Database DB=_db_

  _cConnStr_ := "DBE="    +"pgdbe" +";"
  _cConnStr_ += "UID="    +_uid_   +";"
  _cConnStr_ += "PWD="    +_pwd_   +";"
  _cConnStr_ += "SERVER=" +_server_+";"
  _cConnStr_ += "DB="     +_db_    +";"

  PUBLIC _cConnStrTest_  // konekcija na PostgreSQL server NO Database
                         // radi test provere da li postoji Database _db_ !

  _cConnStrTest_ := "DBE="    +"pgdbe" +";"
  _cConnStrTest_ += "UID="    +_uid_   +";"
  _cConnStrTest_ += "PWD="    +_pwd_   +";"
  _cConnStrTest_ += "SERVER=" +_server_+";"

* memowrit("t.txt",_cConnStr_+chr(13)+_cConnStrTest_)
* dobija se:
* PUBLIC _cConnStrTest_ // konekcija na PostgreSQL server bez database name _db_
```

```
* _cConnStrTest_ = "DBE=pgdbe;UID=postgres;PWD=coba#1949;SERVER=localhost;"
* PUBLIC _cConStr_       // konekcija na PostgreSQL server sa database name _db_
* _cConnStr_     = "DBE=pgdbe;UID=postgres;PWD=coba#1949;SERVER=localhost;DB=kasa;"

* memowrit("t.txt",_cConnStr_+chr(13)+_cConnStrTest_)
* yields:
* PUBLIC _cConnStrTest_ // connection to PostgreSQL server without database name _db_
* _cConnStrTest_ = "DBE=pgdbe;UID=postgres;PWD=coba#1949;SERVER=localhost;"
* PUBLIC _cConStr_ // connection to PostgreSQL server with database name _db_
* _cConnStr_ = "DBE=pgdbe;UID=postgres;PWD=coba#1949;SERVER=localhost;DB=kasa;"

RETURN NIL



/// <descrioption>
/// BROJ PRODAVNICE MAG - BROJ KASE KAS
/// PUBLIC VARIJABLE ZA PROGRAM: KASA
///
/// Varijable se učitavaju iz gde_exe()+\"KASA.INI" fajla
///
///   [KASA_CFG]
///   cObjekat_Kase=11
///   cBroj_Kase=01
///
/// </description>
/// <description>
/// STORE NUMBER MAG - CASH REGISTER NUMBER CASH
/// PUBLIC VARIABLES FOR APPLICATIONS
///
/// Variables are loaded from gde_exe()+\"KASA.INI" file
///
/// [KASA_CFG]
/// cObjekat_Kase=11
/// cBroj_Kase=01
///
/// </description>
*************************************************************************
FUNCTION __pg__kasa_MAG_KAS() // --> FORMIRA PUBLIC KASE: _MAG_, _KAS_
*************************************************************************
*"Posebno: koristi se samo ta program KASA

PUBLIC _MAG_ := "11", _KAS_ := "11"
  // zbog kompatibilnosti sa starim verzijama
PUBLIC cObjekat_Kase := "11", cBroj_Kase    := "11"


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


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_
cBroj_Kase    := _KAS_

* PUBLIC varijable _MAG_ i _KAS_ koriste se za formiranje naziva tabli:
*
* za _MAG_ = "11" formira se tabla "kroba_11"
* za _MAG_ = "22" formira se tabla "kroba_22"
*
* za _KAS_ = "01" formira se tabla "kasa_01","kaso_01','kbon_01,'kblok_01'
* za _KAS_ = "02" formira se tabla "kasa_02","kaso_02','kbon_02,'kblok_02'

RETURN NIL
```

__PG__INI__EDIT.PRG

```
////////////////////////////////////////////////////////////////////////
//                                                                    //
//                                                                    //
//   __pg__ini__edit.PRG                                              //
//                                                                    //
//   09-10-2023                                                       //
//                                                                    //
//   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.        //
//                                                                    //
//                                                                    //
////////////////////////////////////////////////////////////////////////



* APP FUNCTION in __pg__ini__edit.prg
* FUNCTION __pg__ini__edit()
* STATIC FUNCTION xpassword() // password je službena reč u PGDBE
* FUNCTION ini___read()       // password is the official word in PGDBE
* FUNCTION ini___default(nn)
* FUNCTION ini___write()


  #include "Appevent.ch"
  #include "Xbp.ch"
  #include "common.ch"
  #INCLUDE "dcdialog.ch"


*************************************************************************

 FUNCTION __pg__ini__edit()

*************************************************************************

LOCAL GetList := {}, oDlg
LOCAL xx
PRIVATE xpassword := .F.

SET CHARSET TO ANSI

co0 := GraMakeRGBColor({100,30,50}) // CRNO VINO
co1 := GRA_CLR_WHITE
co2 := GRA_CLR_DARKGREEN
co3 := GRA_CLR_BLACK
co4 := GRA_CLR_YELLOW
of1  := Font_codepage(14,"Archivo Narrow",238,.t.)
of2  := Font_codepage(12,"Archivo Narrow",238,.f.)
of22  := Font_codepage(12,"Archivo Narrow",238,.t.)
```

```
of3  := Font_codepage(11,"Archivo Narrow",238,.t.)
of4  := Font_codepage(11,"Consolas",238,.t.)
aCUR := {"user32.dll",114}
oCUR := {"user32.dll",112}


*****************
 ini___read()
*****************
* formirane su:
* PUBLIC ;
* _server_     ,;
* _uid_        ,;
* _pwd_        ,;
* _db_         ,;
* _she_        ,;
* _tab_        ,;
* _lAlert_     ,;
* _lMsg_       ,;
* _lDBcontrol_ ,;
* _MAG_        ,;
*  cObjekat_Kase ,;
*  cBroj_Kase



xx:=0
//-------------------------------------------------------------------------
  @ xx,1  DCSAY "WORKING MODE PostgreSQL DATABASE" FONT of22 SAYSIZE 0


xx:=xx+1+1
  _server_ := PADR(_server_,36)
  @ xx,1  DCSAY "Server name (Localhost or 127.0.0.1)" FONT of2 SAYSIZE 0
xx:=xx+1
  @ xx,1  DCGET _server_ FONT of4 WHEN {|| xpassword }

xx:=xx+1
  _uid_ := PADR(_uid_,36)
  @ xx,1  DCSAY "User name (default=postgres)" FONT of2 SAYSIZE 0
xx:=xx+1
  @ xx,1  DCGET _uid_ FONT of4 WHEN {|| xpassword }

xx:=xx+1
  _pwd_ := PADR(_pwd_,36)
  @ xx,1  DCSAY "Password (default=postgres)" FONT of2 SAYSIZE 0
xx:=xx+1
  @ xx,1  DCGET _pwd_ FONT of4 WHEN {|| xpassword }

xx:=xx+1
  _db_ := PADR(_db_,36)
  @ xx,1  DCSAY "Database (default=postgres)" FONT of2 SAYSIZE 0
xx:=xx+1
  @ xx,1  DCGET _db_ FONT of4 WHEN {|| xpassword }

xx:=xx+1
  _she_ := PADR(_she_,36)
  @ xx,1  DCSAY "Scheme (default=public)" FONT of2 SAYSIZE 0
xx:=xx+1
```

```
  @ xx,1  DCGET _she_ FONT of4 WHEN {|| xpassword }

xx:=xx+1
  _tab_ := PADR(_tab_,36)
  @ xx,1  DCSAY "Table (mandatory=coba_systems)" FONT of2 SAYSIZE 0
xx:=xx+1
  @ xx,1  DCGET _tab_ FONT of4 WHEN {|| .F. }


xx:=xx+1
  _MAG_ := PADR(_MAG_,36)
  @ xx,1  DCSAY "Code Shop-magazine (code=01,02,03...99)" FONT of2 SAYSIZE 0
xx:=xx+1
  @ xx,1  DCGET _MAG_ FONT of4 WHEN {|| xpassword }

xx:=xx+1
  _KAS_ := PADR(_KAS_,36)
  @ xx,1  DCSAY "Code Casch-register (code=01,02,03...99)" FONT of2 SAYSIZE 0
xx:=xx+1
  @ xx,1  DCGET _KAS_ FONT of4 WHEN {|| xpassword }

xx:=xx+1


xx:=xx+1
  @ xx,1  DCCHECKBOX  _lAlert_ PROMPT "User mod msg - display error messages to user" FONT of2
WHEN {|| xpassword }
xx:=xx+1
  @ xx,1  DCCHECKBOX  _lMsg_   PROMPT "Test mod msg - display error messages to admin" FONT of2
WHEN {|| xpassword }
xx:=xx+1
  @ xx,1  DCCHECKBOX  _lDBcontrol_ PROMPT "Database control - database correctness control"
FONT of2 WHEN {|| xpassword }



//-------------------------------------------------------------------------

xx:=xx+2
  @ xx,01 ;
  DCPUSHBUTTONXP CAPTION "Unlock;for changes" ;
         SIZE 14,3 ACTION {||xpassword(),DC_getrefresh(GetList)} ;
         CURSOR aCUR OUTLINE COLOR co1,co2 CLICKCOLOR co3,co4 FONT of22
         // WHEN {||.F.} // B L O K I R A N O ---> u ovoj verziji

  @ xx,14 ;
  DCPUSHBUTTONXP CAPTION "OK" ;
         SIZE 10,3 ACTION {||DC_ReadGuiEvent(DCGUI_EXIT_OK,GetList)} ;
         CURSOR aCUR OUTLINE COLOR co1,co2 CLICKCOLOR co3,co4 FONT of22

  @ xx,14+10 ;
  DCPUSHBUTTONXP CAPTION "Help" ACCELKEY xbeK_F1 ;
         SIZE 10,3 ACTION {||;
         c__procitaj(;
         "Moraju se upisati svi zadati parametri aplikacije."     +chr(59)+;
         "Server može biti naziv servera ili IP adresa servera."  +chr(59)+;
         "U zagradama su dati default parametri koji će raditi "  +chr(59)+;
         "sa default instaliranim postgreSQL serverom sa njima."  +chr(59)+;
         "Sve se upisuje isključivo malim slovima."               +chr(59)+;
         "User mod messages - prikaz poruka greške korisniku"     +chr(59)+;
```

```
         "Test mod messages - prikaz poruka administratoru"       +chr(59)+;
         "Database control - na startu programa uvek kontroliše"  +chr(59)+;
         "ispravnost baze podataka programa. Zbog ubrzanja rada"  +chr(59)+;
         "programa može se isključiti."                           +chr(59)+;
         "----------"                                             +chr(59)+;
          "All default application parameters must be entered."   +chr(59)+;
          "Server can be a server name or a server IP address."   +chr(59)+;
          "The default parameters that will work are given in "   +chr(59)+;
          "parentheses with postgreSQL server installed by "      +chr(59)+;
          "default with them."                                    +chr(59)+;
          "Everything is written in lowercase only."              +chr(59)+;
          "User mode messages - display of error messages to user"    +chr(59)+;
          "Test mode messages - display of messages to administrator" +chr(59)+;
          "Database control - always controls at start of program"    +chr(59)+;
          "correctness of program database. Due to acceleration"      +chr(59)+;
          "of work the program can be turned off."                    +chr(59)+;
         " ","WORKING MODE PostgreSQL DATABASE")} ;
         CURSOR aCUR OUTLINE COLOR co1,co2 CLICKCOLOR co3,co4 FONT of22

  @ xx,14+10+10 ;
  DCPUSHBUTTONXP CAPTION "Exit" ACCELKEY xbeK_ESC;
         SIZE 10,3 ACTION {||DC_ReadGuiEvent(DCGUI_EXIT_OK,GetList)};
         CURSOR aCUR OUTLINE COLOR co1,co2 CLICKCOLOR co3,co4 FONT of22
//-------------------------------------------------------------------------

 DCGETOPTIONS NOMAXBUTTON NOMINBUTTON noescapekey ;
 COLORGETS { { GRA_CLR_WHITE, GraMakeRGBColor({208,82,92}) },{ GRA_CLR_DARKRED, GRA_CLR_WHITE }
}
// This will paint selected Gets White on Red, unselected Gets
// Black on White (default).

 DCREAD GUI OPTIONS GetOptions ENTEREXIT PARENT @oDlg MODAL ;
 FIT TITLE "WORKING MODE PostgreSQL DATABASE"

     *****************
       ini___write()
     *****************
     c__poruka("UPISANO-REGISTERED","OK",,"G3")

RETURN(.T.)
***************************
 STATIC FUNCTION xpassword()
***************************
// xpassword := .F.
IF C__sifra(2,"Password=22")=.F.
   RETURN NIL
ENDIF
xpassword := .T.
RETURN NIL



*************************************************************************
FUNCTION ini___read()
*************************************************************************


PUBLIC cobapgsqlini := gde_exe()+"\C-PGDB.INI"

PUBLIC ;
```

```
 _server_     ,;
 _uid_        ,;
 _pwd_        ,;
 _db_         ,;
 _she_        ,;
 _tab_        ,;
 _lAlert_     ,;
 _lMsg_       ,;
 _lDBcontrol_ ,;
 _MAG_        ,;
 _KAS_        ,;
  cObjekat_Kase ,;
  cBroj_Kase


 //----------------------------- učitaj podatak iz INI fajla

 _server_ := INI_READ("C","DATABASE","_server_" ,cobapgsqlini)
 _MAG_    := INI_READ("C","KASA_CFG","cObjekat_Kase",gde_exe()+"\KASA.INI")

 IF EMPTY(ALLTRIM(_server_)) // ako nema podatka
    **********************  // upiši u INI sve oznake u default vrednostima
    //stop("default _server_",1)
    ini___default(1)
    **********************
 ENDIF

 IF EMPTY(ALLTRIM(_MAG_))    // ako nema podatka
    **********************  // upiši u INI sve oznake u default vrednostima
    //stop("default _MAG_",1)
    ini___default(2)
    **********************
 ENDIF

 //----------------------------- ponovo učitaj sve podatke iz INI fajla

 _server_     := INI_READ("C","DATABASE","_server_"   ,cobapgsqlini)
 _uid_        := INI_READ("C","DATABASE","_uid_"      ,cobapgsqlini)
 _pwd_        := INI_READ("C","DATABASE","_pwd_"      ,cobapgsqlini)
 _db_         := INI_READ("C","DATABASE","_db_"       ,cobapgsqlini)
 _she_        := INI_READ("C","DATABASE","_she_"      ,cobapgsqlini)
 _tab_        := INI_READ("C","DATABASE","_tab_"      ,cobapgsqlini)
 _lAlert_     := INI_READ("L","DATABASE","_lAlert_"   ,cobapgsqlini)
 _lMsg_       := INI_READ("L","DATABASE","_lMsg_"     ,cobapgsqlini)
 _lDBcontrol_ := INI_READ("L","DATABASE","_lDBcontrol_",cobapgsqlini)

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

  cObjekat_Kase := _MAG_
  cBroj_Kase    := _KAS_

 //----------------------------- ponovo učitaj sve podatke iz INI fajla

RETURN NIL


*****************************************************************************
FUNCTION ini___default(nn)
*****************************************************************************
```

```
* ako ne postoji INI fajl, biće napravljen i u njega će biti upisane ove
* default vrednosti. Ako pak postoji INI fajl u njega će biti upiane ove
* default vrednosti

PUBLIC cobapgsqlini := gde_exe()+"\C-PGDB.INI"

IF nn==1
 INI_WRITE("C","DATABASE","_server_"   , "Localhost"     , cobapgsqlini)
 INI_WRITE("C","DATABASE","_uid_"      , "postgres"      , cobapgsqlini)
 INI_WRITE("C","DATABASE","_pwd_"      , "postgres"      , cobapgsqlini)
 INI_WRITE("C","DATABASE","_db_"       , "postgres"      , cobapgsqlini)
 INI_WRITE("C","DATABASE","_she_"      , "public"        , cobapgsqlini)
 INI_WRITE("C","DATABASE","_tab_"      , "coba_systems"  , cobapgsqlini)
 INI_WRITE("L","DATABASE","_lAlert_"   , .F.             , cobapgsqlini)
 INI_WRITE("L","DATABASE","_lMsg_"     , .F.             , cobapgsqlini)
 INI_WRITE("L","DATABASE","_lDBcontrol_",.F.             , cobapgsqlini)
ENDIF

IF nn==2
 INI_WRITE("C","KASA_CFG","cObjekat_Kase", "11", gde_exe()+"\KASA.INI")
 INI_WRITE("C","KASA_CFG","cBroj_Kase"   , "11", gde_exe()+"\KASA.INI")
ENDIF

RETURN NIL


*****************************************************************************
FUNCTION ini___write()
*****************************************************************************
PUBLIC cobapgsqlini := gde_exe()+"\C-PGDB.INI"

 INI_WRITE("C","DATABASE","_server_"   , _server_ , cobapgsqlini)
 INI_WRITE("C","DATABASE","_uid_"      , _uid_    , cobapgsqlini)
 INI_WRITE("C","DATABASE","_pwd_"      , _pwd_    , cobapgsqlini)
 INI_WRITE("C","DATABASE","_db_"       , _db_     , cobapgsqlini)
 INI_WRITE("C","DATABASE","_she_"      , _she_    , cobapgsqlini)
 INI_WRITE("C","DATABASE","_tab_"      , _tab_    , cobapgsqlini)
 INI_WRITE("L","DATABASE","_lAlert_"   , _lAlert_ , cobapgsqlini)
 INI_WRITE("L","DATABASE","_lMsg_"     , _lMsg_   , cobapgsqlini)
 INI_WRITE("L","DATABASE","_lDBcontrol_",_lDBcontrol_, cobapgsqlini)

 INI_WRITE("C","KASA_CFG","cObjekat_Kase", _MAG_, gde_exe()+"\KASA.INI")
 INI_WRITE("C","KASA_CFG","cBroj_Kase"   , _KAS_, gde_exe()+"\KASA.INI")

RETURN NIL
```

__PG__CONNECT.PRG

```
////////////////////////////////////////////////////////////////////////
//                                                                    //
//                                                                    //
//   __pg__connect.PRG                                                //
//                                                                    //
//   25-10-2023                                                       //
//                                                                    //
//   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
* ALERTBOX()   // BAZNE.DLL
* C__GRESKA()  // BAZNE.DLL
* C__PORUKA()  // BAZNE.DLL
* EC_P()       // BAZNE.DLL
* EC_K()       // BAZNE.DLL
* STOP()       // BAZNE.DLL


* FUNCTIONS in __pg__connect.PRG
*
*  FUNCTION __pg__connect_app()
*  FUNCTION __pg__dbinserver()
*  FUNCTION __pg__dbcreate()
*  FUNCTION __pg__connect()
*  FUNCTION __pg__disconnect()
*  FUNCTION __pg__version()
*  FUNCTION __pg__appquit()
*
*  Svaka od ovih funkcija može se pozvati kao samostalna funkcija:
*  Each of these functions can be called as a standalone function:



/*


=======================
Projekat C_PGDB.DLL/LIB
=======================
Svaka Xbase++ aplikacija koja radi sa PostgreSQL serverom
ima svoju postgreSQL databazu.

Ovde u C-PGDB.DLL se koristi UPSIZE DATABASE.
                             ---------------
```

```
UPSIZE database je baza podataka kreirana iz Alaska Xbase++
PGDBE.DLL engin-e, koja ima UPSIZE table u šemi 'public', a koje
mogu da se kontrolišu i upravljaju putem SQL komandi i funkcija
i putem ISAM komandi i funkcija koje se koriste u DBF-NTX ISAM
bazi podataka.
tako da treba vrlo malo menjati kod aplikacije koja radi sa DBF-NTX
da bi ta aplikacija radila sa postgreSQL UPSIZE tablama umesto sa DBF tablama.

Funkcija
       __pg__connect_app()

koja se nalazi u C-PGDB.DLL poziva se na startu svake EXE aplikacije
radi provere postojanja baze podataka i radi kontrole postojanja
svih tabli baze podataka i radi kreiranja baze podataka i tabli ako
iste ne postoje

===========
C-PGDB.DLL
===========
Koristi PGDBE.DLL/LIB/CH
Koristi DBFUPSIZE.DLL/LIB
Koristi BAZNE.LIB

sadrži sve potrebne funkcije za :
čekiranje postojanja zadate databaze na postgreSQL serveru,
auto kontrolu baze podataka, šeme i tabli,
auto kreiranje baze podataka, šema i tabli,
konekciju na bazu podataka,
diskonekciju sa baze podataka,
punjenje tabli podacima preuzetim iz SQL script fajla,
generisanje liste svih databaza na serveru
generisanje liste svih šema u konektovanoj databazi
generisanje listi svih tabli u konektovanoj databazi i šemi


=======================
Project C-PGDB.DLL/LIB
=======================
Any Xbase++ application that works with a PostgreSQL server
has its own postgreSQL database.

UPSIZE DATABASE is used here in C-PGDB.DLL.
                                ---------------
UPSIZE database is a database created from Alaska Xbase++
PGDBE.DLL engin-e, which has UPSIZE tables in the scheme 'public', and which
they can be controlled and managed through SQL commands and functions
and through ISAM commands and functions used in DBF-NTX ISAM
database.
so there is very little need to change the code of the application that works with DBF-NTX
to make that application work with postgreSQL UPSIZE tables instead of DBF tables.

Function
        __pg__connect_app()

located in C-PGDB.DLL is called at the start of each EXE application
to check the existence of the database and to control its existence
of all database tables and to create the database and tables if
they do not exist

===========
```

```
C_PGDB.DLL
==========
Uses PGDBE.DLL/LIB/CH
Uses DBFUPSIZE.DLL/LIB
Uses BAZNE.LIB

contains all the necessary functions for:
checking the existence of the given database on the postgreSQL server,
auto control of database, schema and table,
auto creation of database, schemas and tables,
database connection,
database disconnection,
filling tables with data downloaded from the SQL script file,
generating a list of all databases on the server
generating a list of all schemas in the connected database
generating a list of all tables in the connected database and schema


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



 ŠTA RADI OVAJ PROGRAM

 PROGRAM PRIPADA SETU PROGRAMA KOJIM SE STARTUJE Alaska Xbase++ aplikaciju
 koja radi sa PGDBE.DLL i PostgreSQL bazom podataka sa UPSIZE TABLAMA koje
 dozvoljavaju upotrebu ISAM navigacije na SQL tablama.

 Ovaj set programa na startu aplikacije proverava postojanje i ispravnost
 postgreSQL baze podataka aplikacije. Ako databaza ne postoji kreira je.
 Ako šema ne postoji kreira je. Ako table u šemi ne postije kreira ih.
 Kada sve postoji, konektuje se na databazu i startuje gl.meni aplikacije.
 Na kraju aplikacije diskonektuje se sa databaze.
 Podrazumeva se da u računaru mora da bude instaliran postgreSQL server.
 Ovaj softver je pravljen i testiran sa PostgreSQL serverom verzija 9.4.4

  WHAT THIS PROGRAM DOES

  THE PROGRAM BELONGS TO THE SET OF PROGRAMS THAT START THE Alaska Xbase++ application
  which works with PGDBE.DLL and a PostgreSQL database with UPSIZE TABLES which
  allow the use of ISAM navigation on SQL tables.

  This set of programs checks the existence and correctness of the application at the start
  postgreSQL application database. If the database does not exist, create it.
  If the schema does not exist, create it. If the tables in the scheme do not exist, create
them.
  When everything is there, it connects to the database and starts the main menu of the
application.
  At the end of the application, it disconnects from the database.
  It goes without saying that the computer must have a postgreSQL server installed.
  This software was built and tested with PostgreSQL server version 9.4.4


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

 OVAJ SET PROGRAMA SASTOJI SE IZ PRG MODULA:
 (svaka aplikacija ima ove programe - isti su za svaku aplikaciju
  svaka aplikacija ima različite parametre u C-PGDB.INI )
```

```
     __pg__connect.PRG   ---> __pg__connect_app() --> main function
                                  control-create database,
                                  control-create schemes,
                                  control-create tables

     __pg__ini.prg       ---> database parameters C-PGDB.INI
     __pg__ini__edit.prg ---> database parameters C-PGDB.INI
     __pg__she.prg       ---> Schemes              --- control-create
     __pg__lists.prg     ---> Lista databases, schemes, tables
     __pg__tab.prg       ---> table 'coba_systems' --- control-create
     __tab__create.prg   ---> tables control, create

     __tab__create__all.prg  ---> List with table names and sql strings
                                  Not in C-PGDB.DLL
                                  It is located in the EXE application
                                  C-POSTGRESQL-DATABASE.EXE

----------
   Public varijable databaze aplikacije:
   _server_, _uuid_, _pwd_, _db_, _she_, _tab_ ...

   __pg__ini.prg
----------
   Redosled operacija - manager kontrole, generisanja i startovanja databaze
   Kontrola postojanja databaze aplikacije _db_
   Kreiranje databaze aplikacije _db_
   Connect and Disconnect na databazu aplikacije

   __pg__connect.PRG
----------
   Kontrola postojanja šeme aplikacije
   Kreiranje šeme aplikacije

   __pg__she.prg
----------
   Kontrola postojanja obavezne 'coba_systems' table aplikacije
   Kreiranje obavezne 'coba_systems' table aplikacije

   __pg__tab.prg
----------
  kontrola postojanja tabli za databazu ive aplikacije
  Kreiranje nepostojećih tabli

   __tab__create.prg
   __tab__create__all.prg
----------

  Ako neki od ovih programa-operacija ne uspeju, aplikacija se prekida.
  Posle uspešnog izvršenja ovih programa-operacija, aplikacija se nastavlja
  konekcijom na izabranu bazu podataka _db_
  Na kraju aplikacije izvršava se diskonekcija sa databaze _db_

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

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

 THIS PROGRAM SET CONSISTS OF PRG MODULES:
 (every application has these programs - they are the same for every application
  each application has different parameters in C-PGDB.INI )
```

```
      __pg__connect.PRG ---> __pg__connect_app() --> main function
                                  control-create database,
                                  control-create schemes,
                                  control-create tables

      __pg__ini.prg       ---> database parameters C-PGDB.INI
      __pg__ini__edit.prg ---> database parameters C-PGDB.INI
      __pg__she.prg       ---> Schemes --- control-create
      __pg__lists.prg     ---> List of databases, schemes, tables
      __pg__tab.prg       ---> tab 'coba_systems' --- control-create
      __tab__create.prg   ---> tables control, create

      __tab__create__all.prg ---> List with table names and sql strings
                                   Not in C-PGDB.DLL
                                   It is located in the EXE application
  ----------
     Public variables of the application database:
     _server_, _uuid_, _pwd_, _db_, _she_, _tab_ ...

     __pg__ini.prg
  ----------
     Order of operations - manager of database control, generation and startup
     Checking the existence of application database _db_
     Creating application database _db_
     Connect and Disconnect to the application database

     __pg__connect.PRG
  ----------
     Checking the existence of the application schema
     Creating an application schema

     __pg__she.prg
  ----------
     Checking the presence of the mandatory 'coba_systems' application table
     Creating the required 'coba_systems' application table

     __pg__tab.prg
  ----------
    control of the existence of tables for the database of the live application
    Creating non-existing boards

     __tab__create.prg
     __tab__create__all.prg
  ----------

    If any of these program-operations fail, the application terminates.
    After the successful execution of these program-operations, the application continues
    by connecting to the selected database _db_
    At the end of the application, a disconnection from the _db_ database is performed

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

*/



*-----------------------------------
* Xbase++
#include "dac.ch"
```

```
#pragma library("dbfupsize.lib")
// dbfupsize.dll/lib required for: oInfo:=DacSchema():New(_oSession_)
#include "Xbp.ch"
#include "AppEvent.ch"
#include "Common.ch"
* Xbase++ PGDBE
#include "pgdbe.ch"
#include "std.ch"
#include "sql.ch"
#pragma library("adac20b.lib")
*-----------------------------------
* eXpress++
#include "dcdialog.ch"
*-----------------------------------




/// <descrioption>
/// START APLIKACIJE I KONEKCIJA NA POSTGRESQL SERVER DATABASE _db_
/// ================================================================
/// __pg__dbinsever()     // connect-work-disconnect
/// __pg__dbcreate()      // connect-work-disconnect
/// __pg__dbconnect()     // connect
///
/// PUBLIC _lAlert_, _lMsg_, lDBcontrol su iz funkcije __pg__ini.prg
///
///
/// </description>

/// <description>
/// APPLICATION START AND CONNECTION TO POSTGRESQL SERVER DATABASE _db_
/// ================================================== ==================
/// __pg__dbinsever() // connect-work-disconnect
/// __pg__dbcreate() // connect-work-disconnect
/// __pg__dbconnect() // connect
///
/// PUBLIC _lAlert_, _lMsg_, lDBcontrol are from the __pg__ini.prg function
///
///
/// </description>


* Postupak za upotrebu ove funkcije u aplikaciji:
* Procedure to use this function in the application:
*
* PROCEDURE APPSYS
*       SET CHARSET TO ANSI
*       SET DATE GERMAN
* RETURN
* PROCEDURE DBSYS
*       DBELOAD('PGDBE')
* RETURN
* PROCEDURE MAIN
*  __pg__ini() ---> _lDBcontrol_
*  IF _lDBcontrol_=.T.
*     IF __pg__connect_app() = .F. // kontrola i kreiranje database
*        __pg__appquit()
```

```
*     ENDIF
*  ENDIF
* __pg__connect()
*  APP_MENU() ... APLIKACIJA - procedure i funkcije aplikacije
* __pg__disconnect() ili __pg__appquit()
* RETURN

*************************************************************************
FUNCTION __pg__connect_app()
*************************************************************************
LOCAL okDBexist, okDBcreate, okDBconnect, qq, lRet := .T.


* PUBLIC VARIJABLE PostgreSQL servera iz C-PGDB.INI


* --- 1 ---
*"-----------------------------------------------------------------
*" VAŽNO:
*" Uvek pozovi __pg__ini() pred svaku konekciju koja se ne vrši sa
*" __pg__connect() --- jer __pg__connect() poziva __pg__ini() ---
*" Uvek pozovi __pg__ini() za dobijanje ispravnih PUBLIC database
*---
*" IMPORTANT:
*" Always call __pg__ini() before any connection not made with
*" __pg__connect() --- because __pg__connect() calls __pg__ini() ---
*" Always call __pg__ini() to get the correct PUBLIC databases
*"-----------------------------------------------------------------
*****************************
 __pg__ini()                     // __pg__ini.prg
*****************************
* iz __pg__ini() se poziva
* __pg__connection_string()    // __pg__ini.dbf
* i dobija se:
* PUBLIC _cConnStr_
* PUBLIC _cConnStrTest_
* iz funkcije:
* PUBLIC cConnStrTest // konekcija na PostgreSQL server bez database name _db_
* cConnStrTest = "DBE=pgdbe;UID=postgres;PWD=coba#1949;SERVER=localhost;"
* PUBLIC cConStr      // konekcija na PostgreSQL server sa database name _db_
* cConnStr     = "DBE=pgdbe;UID=postgres;PWD=coba#1949;SERVER=localhost;DB=kasa;"

* --- 1 ---

/*

* --- 13-10-2023
* važno:
* Ovde se blokira mogućnost da se kao databaza aplikacije koristi databaza
* postgres i njena šema public. Dtabaza postgres se ne može obrisati ali njena
* šema public se može obrisati.
* Ne bi trebalo ovo blokirati. Zato što se šema public iz databaze postgres
* može obrisati a tada program pukne.
* Ako se ovde dalji program prekine kada se pokuša rad sa databazom progres,
* tada korisnik ne može ponovo iz aplikacije napraviti šemu public i ne može
* nastaviti rad bez intervenice servisera i administratora baze podataka.
* Ništa ne smeta ako se i u databazi postgres formira aplikaciona šema i u
* njoj aplikacione table. Ipak testirati ovo na dalje
*---
* important:
```

```
* Here, the possibility to use a database as the database of the application is blocked
* postgres and its public scheme. The postgres database cannot be deleted, but its own
* schema public can be deleted.
* Shouldn't block this. Because the schema is public from the postgres database
* can delete and then the program breaks.
* If the further program here is interrupted when an attempt is made to work with the progress
database,
* then the user cannot make the schema public again from the application and cannot
* continue work without the intervention of service personnel and database administrators.
* Nothing gets in the way if the application scheme is formed in the postgres database as well
* her application board. Still testing this further
* --- 13-10-2023

IF _db_ == "postgres"

   C__GRESKA("Nije dozvoljen rad sa databazom ( postgres )" +chr(59)+;
             "Illegal work with the database ( postgres )"  +chr(59)+;
             " "                                            +chr(59)+;
             "Podesi mod rada postgreSQL baze podataka"     +chr(59)+;
             "iz servisnog programa CSYSTEMS_PgSQL.EXE."    +chr(59)+;
             " "                                            +chr(59)+;
             "Set postgreSQL database operation mode"       +chr(59)+;
             "from service program CSYSTEMS_PgSQL.EXE."     +chr(59)+;
             "","STOP",,"G3")

   __pg__appquit()

ENDIF
*/


*" D A T A B A S E  C O N T R O L
*" Provera postojanja databaze _db_
*" na postgreSQL serveru
****************************** //-----------------------
okDBexist := __pg__dbinserver()  // connect-work-disconnect
****************************** //-----------------------

*=====================================================================
IF okDBexist == .F.
*=====================================================================

   qq:=alertbox(,;
   "P R V I  S T A R T  P R O G R A M A   "+chr(59)+;
   "NA SERVERU NE POSTOJI BAZA PODATAKA"   +chr(59)+;
   "THERE IS NO DATABASE ON THE SERVER"    +chr(59)+;
   "                                  "    +chr(59)+;
    PADC(_cdbc_,33)                        +chr(59)+;   // za G3=75
   "                                  "    +chr(59)+;
   "FORMIRAJ OVU BAZU PODATAKA PROGRAMA"   +chr(59)+;
   "FORM THIS PROGRAM DATABASE"            +chr(59)+;
   " ",;
   {"   YES   ", "   NO   "},,;
   "THERE IS NO DATABASE OF THIS PROGRAM","14.Consolas Bold","R",,,11044)

   IF qq=2 .or. qq=0

      *" PREKINI PROGRAM
      C__GRESKA("PROGRAM STOP","PROGRAM SE PREKIDA",,"G3")
```

```
      ***************
      __pg__appquit()
      ***************

   ENDIF


*" ako na serveru ne postoji zadata APP database _db_
*" napravi je automatski - bez postavljanja pitanja korisniku

   ec_p()
   ****************************  //-----------------------
   okDBcreate := __pg__dbcreate()  // connect-work-disconnect
   ****************************  //-----------------------
   ec_k()

   IF okDBcreate == .F.

      *" PREKINI PROGRAM
      *" ako kreiranje _db_ nije uspelo prekini program sa QUIT
      ***************
      __pg__appquit()
      ***************

   ENDIF
   *" NASTAVI PROGRAM

      // ako je kreiranje _db_ uspelo nastavi program
   *  c__poruka("*** DATABASE "+ _db_ +" CREATED ***","FORMIRANA JE BAZA PODATAKA")


*" REZULTAT
*" Ako databaza _db_ ne postoji i ako create databaze nije uspelo
*" bezuslovni prekid programa. Ako databaza _db_ postoji nastavak
*" programa...
*" RESULT
*" If database _db_ does not exist and create database failed
*" unconditional termination of the program. If database _db_ exists continue
*" program...

*====================================================================
ENDIF // IF okDBexist == .F.
*====================================================================




*" S C H E M E  C O N T R O L
*" proveri postojanje šeme _she_ u databazi _db_
**************************** //-----------------------
okSheexist := __pg__sheexist()  // connect-work-disconnect
**************************** //-----------------------



*====================================================================
IF okSheexist == .F.
```

```
*=====================================================================

*" ako na serveru ne postoji zadata APP šema _she_
*" napravi _she_ automatski - bez postavljanja pitanja korisniku
   ********************************  //-----------------------
   okShecreate := __pg__shecreate()   // connect-work-disconnect
   ********************************  //-----------------------

   IF okShecreate == .F.

      *" PREKINI PROGRAM
      *" ako kreiranje _she_ nije uspelo prekini program sa QUIT
      ***************
      __pg__appquit()
      ***************

   ENDIF
   *" NASTAVI PROGRAM

      // ako je kreiranje _she_ uspelo nastavi program
   *  c__poruka("*** SCHEME "+ _she_ +" CREATED ***","FORMIRANA JE ŠEMA BAZE PODATAKA")


*" REZULTAT
*" Ako šema _she_ ne postoji i ako create schema nije uspelo
*" bezuslovni prekid programa. Ako šema _she_ postoji nastavak
*" programa...
*" RESULT
*" If schema _she_ does not exist and create schema failed
*" unconditional program termination. If the schema _she_ exists continue
*" program...
*=====================================================================
ENDIF // IF okSheexist == .F.
*=====================================================================




*" T A B L E  C O N T R O L
*" proveri postojanje obavezne aplikacione table 'coba_systems'
*" u šemi _she_ -  proveravaj varijablu: 'schemename.coba_systems'
*" Ako tabla ne postoji vrati .F.
*" Ako je F. podrazumeva se da ne postoje I ostale table aplikacije
*" Ako tabla postoji vrati .T.
*" Ako je .T. podrazumeva se da postoje sve ostale table aplikacije
*---
*" check the existence of the required application board 'coba_systems'
*" in scheme _she_ - check variable: 'schemename.coba_systems'
*" If the board does not exist return .F.
*" If it is F, it is understood that there are no other application boards
*" If the board exists return .T.
*" If it is .T., it is assumed that all other application panels exist

****************************************** //-----------------------
okTabexist := __pg__Tabexist__coba_systems() // connect-work-disconnect
****************************************** //-----------------------

* stop("tabexist",okTabexist,1);__pg__appquit()
```

```
*=================================================================
IF okTabexist == .F.
*=================================================================

*" ako na serveru ne postoji obavezna APP tabla coba_systems
*" napravi coba_systems automatski - bez postavljanja pitanja
*" a zatim napravi sve table aplikacije kao nove prazne table.
*" Ako table postoje - brišu se - i prave se nove
*" Bez postavljanja pitanja korisniku
*---
*" if there is no mandatory APP board coba_systems on the server
*" create coba_systems automatically - no questions asked
*" and then make all application boards as new blank boards.
*" If boards exist - they are deleted - and new ones are created
*" Without asking the user
   ******************************************  //------------------------
   okTabcreate := __pg__tabcreate__coba_systems()  // connect-work-disconnect
   ******************************************  //------------------------

   IF okTabcreate == .F.

      *" PREKINI PROGRAM
      *" ako kreiranje 'coba_systems' nije uspelo prekini program sa QUIT
      ***************
      __pg__appquit()
      ***************

   ENDIF
   *" NASTAVI PROGRAM
   *" CONTINUE PROGRAM


     ****************************************** production verzija
     *" KREIRAJ SVE OSTALE TABLE APLIKACIJE
     *" CREATE ALL OTHER APPLICATION TABLES
     ec_p()
      __tab__create()   // __tab__create.prg
     ec_k()

     * "UPIŠI TABLE APLIKACIJE U LISTU public.alaska-software.isam.tables"
     * "WRITE APPLICATION TABLES TO LIST public.alaska-software.isam.tables"
     ec_p()
      __tab__control__ISAM()   // __tab__create.prg
     ec_k()
     ******************************************


   qq:=alertbox(,;
   "P R V I  S T A R T  P R O G R A M A   "+chr(59)+;
   "NA SERVERU SADA POSTOJI BAZA PODATAKA" +chr(59)+;
   "ON THE SERVER NOW THERE IS A DATABASE" +chr(59)+;
   "                                  "    +chr(59)+;
    PADC(_cdbc_,33)                       +chr(59)+;   // za G3=75
   "                                  "    +chr(59)+;
   "SLEDI KONEKCIJA NA OVU BAZU PODATAKA"  +chr(59)+;
   "CONNECTION TO THIS DATABASE FOLLOWS"   +chr(59)+;
   " ",;
   {"   OK   "},,;
   "ON THE SERVER NOW THERE IS A DATABASE","14.Consolas Bold","B",,,3357)
```

```
*" REZULTAT
*" Ako tabla 'coba_systems' ne postoji i ako create table nije uspelo
*" bezuslovni prekid programa. Ako tabla 'coba_systems' postoji nastavak
*" programa...
*" RESULT
*" If table 'coba_systems' does not exist and create table failed
*" unconditional program termination. If board 'coba_systems' exists continue
*" program...

*=====================================================================
ENDIF // IF okTabexist == .F.
*=====================================================================

     *" PROVERI DA LI POSTOJE SVE TABLE U DATABAZI
     *" CHECK IF ALL THE TABLES EXIST IN THE DATABASE
     ****************************************** konekcija
      ec_p()
      lRet := __tab__control()   // __tab__create.prg
      ec_k()
     ****************************************** diskonekcija




* OVDE SE ZAVRŠAVA OVA FUNKCIJA
* Funkcija u glavni program vraća:
* - ili __pg__appquit()             // PREKID APLIKACIJE - DATABASE NEISPRAVNA
*                                   // DATABASE ISPRAVNA ALI TABLE NISU-JESU
* - ili lRet = .F. ili lRet = .T.   // .F. SVE TABLE NISU NA BROJU
                                    // .T. SVE TABLE SU NA BROJU

* THIS FUNCTION ENDS HERE
* The function returns to the main program:
* - or __pg__appquit()               // APPLICATION TERMINATION - DATABASE FAULTY
*                                    // DATABASE IS CORRECT BUT THE TABLES ARE NOT
* - or lRet = .F. or lRet = .T.      // .F. ALL BOARDS ARE NOT NUMBERED
                                    // .T. ALL BOARDS ARE NUMBERED

RETURN lRet




*#####################################################################
*
*                         database funkcije
*
*#####################################################################
```

```
/// <descrioption>
/// KONEKCIJA NA POSTGRESQL SERVER  BEZ ZADATE DATABAZE
/// ===================================================================
/// I PROVERA POSTOJANJA ZADATE DATABAZE NAME _db_
/// Connect
/// is _db_ exist YES = TRUE
/// is _db_ exist NO  = FALSE
/// Disconnect
/// return TRUE or FALSE
/// </description>

/// <description>
/// CONNECTION TO POSTGRESQL SERVER WITHOUT DEFAULT DATABASE
/// ===================================================================
/// AND CHECKING THE EXISTENCE OF THE DEFAULT DATABASE NAME _db_
/// Connect
/// is _db_ exist YES = TRUE
/// is _db_ exist NO = FALSE
/// Disconnect
/// return TRUE or FALSE
/// </description>

*************************************************************************
FUNCTION __pg__dbinserver() // lRet =.T.,.F.
*************************************************************************
* _db_ = "coba" je database name koje se proverava u database serveru

LOCAL lRet, cDatab, cOwner, cErrMsg, oInfo

* --- 1 ---
*"-----------------------------------------------------------------
*" VAŽNO: Uvek pozovi __pg__ini()
*" IMPORTANT: Always call __pg__ini()
*"-----------------------------------------------------------------

*****************************
 __pg__ini()                    // __pg__ini.prg
*****************************
* iz __pg__ini() se poziva
* __pg__connection_string()    // __pg__ini.dbf
* i dobija se:
* PUBLIC _cConnStr_
* PUBLIC _cConnStrTest_
* iz funkcije:
* PUBLIC cConnStrTest // konekcija na PostgreSQL server bez database name _db_
* cConnStrTest = "DBE=pgdbe;UID=postgres;PWD=coba#1949;SERVER=localhost;"
* PUBLIC cConStr      // konekcija na PostgreSQL server sa database name _db_
* cConnStr     = "DBE=pgdbe;UID=postgres;PWD=coba#1949;SERVER=localhost;DB=kasa;"

* --- 1 ---
```

```
* --- 2 ---

* iz konekcionog stringa '_cConnStr_' preuzima se user name i database name
* što u cobasystems ne mora jer su to public varijable aplikacije, _db_ i _uid_
* ali je ovde ostavljeno zbog evidentiranja komandi za to:
  * cDatab := _GetValueFromConnectionTuple(_cConnStr_,"DB")
  * cOwner := _GetValueFromConnectionTuple(_cConnStr_,"UID")
  * stop(cDatab,cOwner,1)
  * "kasa", "postgres"
  *
  * Primer upotrebe je:
  * _oSession_  := DacSession():New(_cConnStrTest_)
  * oInfo := DacSchema():New(_oSession_) // DBUPSIZE.DLL
  * IF oInfo:isDatabase(cDatab) == .T.
  *   c_poruka("Postoji Databaza "+cDatab)
  *   lRet := .T.
  * ELSE
  *   c_greska("Ne Postoji Databaza "+cDatab)
  *   lRet := .F.
  * ENDIF

* --- 3 ---

  * Konekcija se vrši na PostgreSQL database server sa konekcionim stringom
  * '_cConnStrTest_' koji ne sadrži database name

  ***********************************************
  _oSession_  := DacSession():New(_cConnStrTest_)
  ***********************************************

  IF _oSession_:IsConnected() == .F.


     cErrMsg := _oSession_:GetLastMessage()
     cErrMsg := "Error ("+cErrMsg+") when connecting"+chr(13)+;
                "Connection-String: "+chr(13)+_cConnStrTest_

    _oSession_ := NIL        // ako konekcija ne uspe
                             // ponovo se postavi _oSession_ := NIL


     ec_k()
     confirmbox(,;
                 "STOP - NEMOGUĆA KONEKCIJA na PostgreSQL server"+chr(13)+;
                 "STOP - UNABLE TO CONNECT to PostgreSQL server" +chr(13)+;
                 " "                                             +chr(13)+;
                 cErrMsg                                         +chr(13)+;
                 "","CONNECTION  |  STOP",XBPMB_OK,XBPMB_CRITICAL)


     RETURN .F.

  ELSE

     ***************
     __pg__version()  // __pg__connect.prg -> _version_, PgVersion
     ***************

     IF _lMsg_==.T.
     ec_k()
```

```
     confirmbox(,;
                 "OK - KONEKCIJA na PostgreSQL server je uspostavljena"+chr(13)+;
                 "OK - CONNECTION to PostgreSQL server is established" +chr(13)+;
                 " "                                                   +chr(13)+;
                 PgVersion                                             +chr(13)+;
                 _version_                                             +chr(13)+;
                 "","CONNECTION  |  OK",XBPMB_OK,XBPMB_INFORMATION)
     ENDIF

  ENDIF


* --- 4 ---

* Check if database exists
* Sada se vrši provera postojanja Database sa zadatim nazivpm _db_
* (u database klasteru) na PosggreSQL serveru:

  ***********************************
  oInfo := DacSchema():New(_oSession_)
  ***********************************
  * Rezultat je objekat:  oInfo = "DacSchema"

  * Klasa DacSchema() se nalazi u DBUPSIZE.DLL/LIB biblioteci
  * Postojanje database se može proveriti i bez DBUPSIZE.DLL
  * The DacSchema() class is located in the DBUPSIZE.DLL/LIB library
  * Database existence can be checked even without DBUPSIZE.DLL


  IF oInfo:isDatabase(_db_) == .T.
    /*
    c_poruka("Postoji Databaza "+_db_)
    */
    lRet := .T.
  ELSE

    IF _lMsg_==.T.
    ec_k()
    c__greska("NO DATABASE EXISTS"+chr(59)+;
              ;//"                  "+chr(59)+;
              PADC(_cdbc_,19)      +chr(59)+;
              "","STOP",,"14.Consolas Bold")
    ENDIF

    lRet := .F.
  ENDIF

  ////////////////////////

  *  __pg__disconnect()
    _oSession_:disconnect()  // bez poruke
    _oSession_ := NIL        // ako konekcija ne uspe
                             // ponovo se postavi _oSession_ := NIL

  ////////////////////////

RETURN lRet
```

```
/// <descrioption>
/// KONEKCIJA NA POSTGRESQL SERVER I KREIRANJE Z A D A T E DATABASE _db_
/// ================================================================
/// Connect
/// create database YES = TRUE
/// create database NO  = FALSE
/// Disconnect
/// return TRUE or FALSE
/// </description>

/// <description>
/// CONNECTION TO POSTGRESQL SERVER AND CREATION OF DATABASE _db_
/// ============================================== ====================
/// Connect
/// create database YES = TRUE
/// create database NO = FALSE
/// Disconnect
/// return TRUE or FALSE
/// </description>
*************************************************************************
FUNCTION __pg__dbcreate() // lRet=.T.,.F.
*************************************************************************
* _db_ = "coba" je database name na koje se konektuje

LOCAL lRet, cDatab, cOwner, cErrMsg, oInfo

* --- 1 ---
*"-----------------------------------------------------------------
*" VAŽNO: Uvek pozovi __pg__ini()
*"-----------------------------------------------------------------

*****************************
 __pg__ini()                    // __pg__ini.prg
*****************************
* iz __pg__ini() se poziva
* __pg__connection_string()    // __pg__ini.dbf
* i dobija se:
* PUBLIC _cConnStr_
* PUBLIC _cConnStrTest_
* iz funkcije:
* PUBLIC cConnStrTest // konekcija na PostgreSQL server bez database name _db_
* cConnStrTest = "DBE=pgdbe;UID=postgres;PWD=coba#1949;SERVER=localhost;"
* PUBLIC cConStr      // konekcija na PostgreSQL server sa database name _db_
* cConnStr     = "DBE=pgdbe;UID=postgres;PWD=coba#1949;SERVER=localhost;DB=kasa;"

* --- 1 ---

* --- 2 ---

  * Konekcija se vrši na PostgreSQL database server sa konekcionim stringom
  * '_cConnStrTest_' koji ne sadrži database name

  *********************************************
  _oSession_  := DacSession():New(_cConnStrTest_)
  *********************************************

  IF _oSession_:IsConnected() == .F.
```

```
    cErrMsg := _oSession_:GetLastMessage()
    cErrMsg := "Error ("+cErrMsg+") when connecting"+chr(13)+;
               "Connection-String: "+chr(13)+_cConnStr_

   _oSession_ := NIL        // ako konekcija ne uspe
                            // ponovo se postavi _oSession_ := NIL

    ec_k()
    confirmbox(,;
                "STOP - NEMOGUĆA KONEKCIJA na PostgreSQL server"+chr(13)+;
                "STOP - UNABLE TO CONNECT to PostgreSQL server" +chr(13)+;
                "       "                                       +chr(13)+;
                "Nije moguća konekcija - no connection possible"+chr(13)+;
                "na bazu podataka - to database:"               +chr(13)+;
                "       "                                       +chr(13)+;
                "                 "+_cdbc_                        +chr(13)+;
                "       "                                       +chr(13)+;
                cErrMsg                                         +chr(13)+;
                "","CONNECTION  |  STOP",XBPMB_OK,XBPMB_CRITICAL)

    RETURN .F.

  ELSE

    ***************
    __pg__version()  // __pg__connect.prg -> _version_, PgVersion
    ***************

    IF _lMsg_==.T.
    ec_k()
    confirmbox(,;
                "OK - KONEKCIJA na PostgreSQL server je uspostavljena"+chr(13)+;
                "OK - CONNECTION to PostgreSQL server is established" +chr(13)+;
                "       "                                               +chr(13)+;
                PgVersion                                               +chr(13)+;
                _version_                                               +chr(13)+;
                "","CONNECTION  |  OK",XBPMB_OK,XBPMB_INFORMATION)
    ENDIF

  ENDIF

    *-----------------
    * OVO FUNKCIONIŠE:
    * ----------------
    * cSQL := "CREATE DATABASE "+_db_+" OWNER "+_uid_+" ENCODING 'WIN1250' TEMPLATE template0"
    * oStmt := DacSqlStatement():fromChar(cSQL)
    * oStmt:build():execute()
    * ENCODING 'UTF8' ili ENCODING 'WIN1250' za postojeće aplikacije
    * ako ne postoji ENCODING po defaultu se kodira na 'UTF8'
    * ako postoji ENCODING 'WIN1250' mora da sledi i TEMPLATE template0
    *-----------------


********************************************************************** START

    *---------------------------------------------- start
    * KREIRAJ BAZU PODATAKA PROGRAMA
    *----------------------------------------------
```

```
    * -- DROP DATABASE IF EXIST coba;
    TEXT INTO cSQL WRAP
    CREATE DATABASE &_db_
      WITH OWNER = &_uid_
           ENCODING = 'UTF8'
           TABLESPACE = pg_default
           LC_COLLATE = 'Serbian (Latin)_Serbia.1250'
           LC_CTYPE = 'Serbian (Latin)_Serbia.1250'
           CONNECTION LIMIT = -1;
    ENDTEXT
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()
        * ako se kodira WIN1250 tada ide:
        * ENCODING = 'WIN1250'
        * TEMPLATE template0
    *-------------------------------------------
    * KREIRAJ BAZU PODATAKA PROGRAMA
    *------------------------------------------- end

************************************************************************ END

    ec_k()
    c_poruka("KREIRANA JE DATABAZA " +chr(13)+;
             "DATABASE IS CREATED  " +chr(13)+;
             "                     " +chr(13)+;
             _cdbc_                  +chr(13)+;
             " ","New Database is created")


  //////////////////////

  * __pg__disconnect()
    _oSession_:disconnect() // bez poruke
    _oSession_ := NIL        // ako konekcija ne uspe
                             // ponovo se postavi _oSession_ := NIL
  //////////////////////

RETURN .T.
```

```
/// <descrioption>
/// KONEKCIJA NA POSTGRESQL SERVER NA  ZADATU DATABAZU _db_
/// ======================================================================
/// Connect
/// is Connect YES = TRUE
/// is Connect NO  = FALSE
/// return TRUE or FALSE
```

```
/// </description>

/// <description>
/// CONNECTION TO POSTGRESQL SERVER ON DEFAULT DATABASE _db_
/// ======================================================================
/// Connect
/// is Connect YES = TRUE
/// is Connect NO = FALSE
/// return TRUE or FALSE
/// </description>
**************************************************************************
FUNCTION __pg__connect() // lRet=.T.,.F.
**************************************************************************
* _db_ = "coba" je database name na koje se konektuje

LOCAL lRet, cDatab, cOwner, cErrMsg, oInfo


* --- 1 ---
*"----------------------------------------------------------------
*" VAŽNO: Uvek pozovi __pg__ini()
*"----------------------------------------------------------------

*****************************
 __pg__ini()                    // __pg__ini.prg
*****************************
* iz __pg__ini() se poziva
* __pg__connection_string()    // __pg__ini.dbf
* i dobija se:
* PUBLIC _cConnStr_
* PUBLIC _cConnStrTest_
* iz funkcije:
* PUBLIC cConnStrTest // konekcija na PostgreSQL server bez database name _db_
* cConnStrTest = "DBE=pgdbe;UID=postgres;PWD=coba#1949;SERVER=localhost;"
* PUBLIC cConStr      // konekcija na PostgreSQL server sa database name _db_
* cConnStr     = "DBE=pgdbe;UID=postgres;PWD=coba#1949;SERVER=localhost;DB=kasa;"

* --- 1 ---


* --- 2 ---

  * Konekcija se vrši na PostgreSQL database server sa konekcionim stringom
  * '_cConnStr_' koji sadrži database name

  *********************************************
  _oSession_  := DacSession():New(_cConnStr_)
  *********************************************

  IF _oSession_:IsConnected() == .F.



     cErrMsg := _oSession_:GetLastMessage()
     cErrMsg := "Error ("+cErrMsg+") when connecting"+chr(13)+;
                "Connection-String: "+chr(13)+_cConnStr_

     _oSession_ := NIL        // ako konekcija ne uspe
                              // ponovo se postavi _oSession_ := NIL
```

```
     ec_k()
     confirmbox(,;
                 "STOP - NEMOGUĆA KONEKCIJA na PostgreSQL server"+chr(13)+;
                 "STOP - UNABLE TO CONNECT to PostgreSQL server" +chr(13)+;
                 "       "                                       +chr(13)+;
                 "Nije moguća konekcija - no connection possible"+chr(13)+;
                 "na bazu podataka - to database:"               +chr(13)+;
                 "       "                                       +chr(13)+;
                 "                 "+_cdbc_                        +chr(13)+;
                 "       "                                       +chr(13)+;
                 cErrMsg                                         +chr(13)+;
                 "","CONNECTION  |  STOP",XBPMB_OK,XBPMB_CRITICAL)


     __pg__appquit() // BEZUSLOVNI PREKID APLIKACIJE

  ELSE

     ***************
     __pg__version()  // __pg__connect.prg -> _version_, PgVersion
     ***************

     IF _lMsg_==.T.
     ec_k()
     confirmbox(,;
                 "OK - KONEKCIJA na PostgreSQL server je uspostavljena"+chr(13)+;
                 "OK - CONNECTION to PostgreSQL server is established" +chr(13)+;
                 " "                                                   +chr(13)+;
                 "Izvršena je konekcija - connection has been made"+chr(13)+;
                 "na bazu podataka - to database:"                 +chr(13)+;
                 "       "                                         +chr(13)+;
                 "                 "+_cdbc_                          +chr(13)+;
                 "       "                                         +chr(13)+;
                 PgVersion                                         +chr(13)+;
                 _version_                                         +chr(13)+;
                 "","CONNECTION  ||  OK",XBPMB_OK,XBPMB_INFORMATION)
     ENDIF

  ENDIF

  ////////////////////////     // STOP
                              //
  *** __pg__appquit()         // NE KORISTI SE OVDE
  *** __pg__disconnect()      // poziva se:
  *** _oSession_:disconnect() // na izlazu iz aplikacije
                              // on exiting the application
  *** _oSession_ := NIL       // ako konekcija ne uspe
                              // ponovo se postavi _oSession_ := NIL

  ////////////////////////

RETURN .T.


/// <descrioption>
/// PREKID KONEKCIJE NA POSTGRESQL SERVER I NA ZADATU DATABASE _db_
/// ==============================================================
/// </description>
/// <description>
/// TERMINATION OF THE CONNECTION TO THE POSTGRESQL SERVER AND THE
```

```
/// DEFAULT DATABASE _db_
/// ============================================================
/// </description>
*************************************************************************
FUNCTION __pg__disconnect()
*************************************************************************
                            // poziva se:
  _oSession_:disconnect()   // na izlazu iz aplikacije
                            // on exiting the application

        _oSession_ := NIL   // ako konekcija ne uspe
                            // ponovo se postavi _oSession_ := NIL
        IF _lAlert_==.T.
          ec_k()
          confirmbox(,;
          "DISCONNECTED (KONEKCIJA PREKINUTA)" +chr(13)+;
          "PostgreSQL Server"           +chr(13)+;
          "Disconnected from Database: "+chr(13)+chr(13)+;
          _cdbc_                        +chr(13)+;
          "","DISCONNECTED",XBPMB_OK,XBPMB_CRITICAL,XBPMB_SYSMODAL)
        ENDIF

RETURN NIL


/*
 * Routine to terminate the programm
 */
*************************************************************************
FUNCTION __pg__appquit()
*************************************************************************
   LOCAL nButton

/* isključeno dok je testiranje
   nButton := ConfirmBox( , ;
                "Do you really want to quit ?", ;
                "Quit", ;
                 XBPMB_YESNO , ;
                 XBPMB_QUESTION+XBPMB_APPMODAL+XBPMB_MOVEABLE )

*/

  nButton := XBPMB_RET_YES

   IF nButton == XBPMB_RET_YES
      COMMIT
      CLOSE ALL

      // Uvek treba prekinuti vezu.
      // We always should disconnect.
      ****************************************************

      // Disconnect from server - close all sessions
         AEval( DacSession():sessionList(), {|o|o:disconnect()})

      // ili prostije:
      // Disconnect from server  - the simplest way
      *  _oSession_:disconnect() // naj jednostavniji način

      ****************************************************
```

```
      QUIT

   ENDIF

RETURN NIL


*************************************************************************
FUNCTION __pg__version()
*************************************************************************
LOCAL QUERY, cSQL, oStmt
PUBLIC _version_, PgVersion

 // dobija se tek posle DbeSys()
 // i posle konekcije na postgreSQL server i to ili
 // bez konekcije na database, ili sa konekcijom na database
 //---
 // obtained only after DbeSys()
 // and after connecting to the postgreSQL server and that or
 // without database connection, or with database connection
    cSQL := "SELECT VERSION();"
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():query(,@xresult)
    SELECT QUERY
    go top
    _version_ := Version
    CLOSE QUERY // mora se zatvoriti inače ne rade sledeći: SELECT QUERY

 // samo broj verzije bez prikaza kompajlera
    PgVersion := left( _version_, at(",",_version_)-1 )
    * stop(_version_,cVersion,1)

RETURN NIL
```

__PG__LISTS.PRG

```
////////////////////////////////////////////////////////////////////////
//                                                                    //
//                                                                    //
//   __pg__lists.PRG
//                                                                    //
//   25-10-2023                                                       //
//                                                                    //
//   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
* ALERTBOX()   // BAZNE.DLL
* C__GRESKA()  // BAZNE.DLL
* C__PORUKA()  // BAZNE.DLL
* EC_P()       // BAZNE.DLL
* EC_K()       // BAZNE.DLL
* STOP()       // BAZNE.DLL


* FUNCTIONS in __pg__lists.prg
*
* FUNCTION __pg__lists(nn, lListView) // _version_, _dblist_, _shelist_, _tablist_
* FUNCTION __list__databases(lList)   // LISTA DATABAZA u KLASTERU _dblist_
* FUNCTION __list__schemas(lList,cdatabasename)            // LISTA ŠEMA u DATABAZI _shelist_
* FUNCTION __list__tables(lList,cdatabasename,cschemaname) // LISTA TABLI u ŠEMi u DATABAZI
_tablist_


*-----------------------------------
* Xbase++
#include "dac.ch"
#pragma library("dbfupsize.lib")
// dbfupsize.dll/lib potrebno za: oInfo:=DacSchema():New(_oSession_)
#include "Xbp.ch"
#include "AppEvent.ch"
#include "Common.ch"
* Xbase++ PGDBE
#include "pgdbe.ch"
#include "std.ch"
#include "sql.ch"
#pragma library("adac20b.lib")
*-----------------------------------
* eXpress++
#include "dcdialog.ch"
```

```
*----------------------------------
  #include "Appbrow.ch"
  MEMVAR appObject




/// <descrioption>
/// PRIKAZ VERZIJE POSTGRESQL SERVERA,
///    vraća: PUBLIC _version_ (full string), PgVersion (lite string)
/// PRIKAZ LISTE DATABAZA NA POSTGRESQL SERVERU,
///    vraća: PUBLIC _dblist_ = Array, AppBrowse-ListView
/// PRIKAZ LISTE ŠEMA U AKTIVNOJ DATABAZI,
///    vraća: PUBLIC _shelist_ = Array, AppBrowse-ListView
/// PRIKAZ LISTE TABLI U ZADATOJ ŠEMI U AKTIVNOJ DATABAZI,
///    vraća: PUBLIC _tablist_ = Array, AppBrowse-ListView
/// </description>

/// <description>
/// POSTGRESQL SERVER VERSION DISPLAY,
/// returns: PUBLIC _version_ (full string), PgVersion (lite string)
/// DISPLAY LIST OF DATABASES ON POSTGRESQL SERVER,
/// returns: PUBLIC _dblist_ = Array, AppBrowse-ListView
/// DISPLAY OF THE LIST OF SCHEMES IN THE ACTIVE DATABASE,
/// returns: PUBLIC _shelist_ = Array, AppBrowse-ListView
/// DISPLAY THE LIST OF PANELS IN THE DEFAULT SCHEME IN THE ACTIVE DATABASE,
/// returns: PUBLIC _tablist_ = Array, AppBrowse-ListView
/// </description>
*************************************************************************
FUNCTION __pg__lists(nn, lListView)
*************************************************************************
* nn=1 vraća _version_
* nn=2 vraća _version_, _dblist_ _shelist_, _tablist_
*
* lListView = false - NO  appbrowse listview
* lListView = true  - YES appbrowse listview



LOCAL okDBexist, okDBcreate, okDBconnect, qq, lRet := .T.


* PUBLIC VARIJABLE PostgreSQL servera iz C-PGDB.INI
****************** // NO CONNECT
  __pg__ini()       // __pg__ini.prg
****************** // NO DISCONNECT
* dobijen je: PUBLIC '_cConnStrTest_' koji ne sadrži database name
****************** // ne konektuje se preko ove funkcije:
*** __pg__connect() // već se vrši default konekcija na postgreSQL
****************** // server bez zadate baze podataka - na database 'postgres'




* --- konektuj se na database 'postgres' da proveriš postojanje database _db_ --- start
* --- connect to database 'postgres' to check existence of database _db_      --- start
```

```
  * Konekcija se vrši na PostgreSQL database server sa konekcionim stringom
  * '_cConnStrTest_' koji ne sadrži database name
  * KONEKCIJA SE VRŠI NA DEFAULT DATABASE 'postgres'
  * The connection is made to the PostgreSQL database server with a connection string
  * '_cConnStrTest_' which does not contain a database name
  * THE CONNECTION IS MADE TO THE DEFAULT DATABASE 'postgres'
  *********************************************
  _oSession_  := DacSession():New(_cConnStrTest_)
  *********************************************

  IF _oSession_:IsConnected() == .F.


     cErrMsg := _oSession_:GetLastMessage()
     cErrMsg := "Error ("+cErrMsg+") when connecting"+chr(13)+;
                "Connection-String: "+chr(13)+_cConnStrTest_


    _oSession_ := NIL        // ako konekcija ne uspe
                             // ponovo se postavi _oSession_ := NIL
     ec_k()
     confirmbox(,;
                 "STOP - NEMOGUĆA KONEKCIJA na PostgreSQL server"+chr(13)+;
                 "STOP - UNABLE TO CONNECT to PostgreSQL server" +chr(13)+;
                 " "                                             +chr(13)+;
                 cErrMsg                                         +chr(13)+;
                 "","CONNECTION  |  STOP",XBPMB_OK,XBPMB_CRITICAL)

     __pg__appquit()

   ENDIF


* --- Check if database _db_ exists  start



* Check if database exists
* Sada se vrši provera postojanja Database sa zadatim nazivpm _db_
* (u database klasteru) na PosggreSQL serveru:

  ***********************************
  oInfo := DacSchema():New(_oSession_)  // #pragma library("dbfupsize.lib")
  ***********************************
  * Rezultat je objekat:  oInfo = "DacSchema"

  * Klasa DacSchema() se nalazi u DBUPSIZE.DLL/LIB biblioteci
  * Postojanje database se može proveriti bez DBUPSIZE.DLL
  * na osnovu pretrage liste svih databaza na serveru:
  * funkcija: __pg__db_in_list(_db_)    -> Array {dbname1,dbname2...}
  * funkcija: __pg__is_db_in_list(_db_) -> lRet=.T.,.F.


IF oInfo:isDatabase(_db_) == .T.
*" -------------------------------------
*" AKO POSTOJI ZADATA DATABASE = _db_
*" IF EXISTS DEFAULT DATABASE = _db_
*" -------------------------------------

    * _db_ := _db_    // u klasteru postoji databse _db_
```

```
                      // pa sledi konekcija na _db_
    STOP_PROGRAMA := .F.

ELSE
*" -------------------------------------
*" AKO NE POSTOJI ZADATA DATABASE = _db_
*" IF NOT EXISTS DEFAULT DATABASE = _db_
*" -------------------------------------

    STOP_PROGRAMA := .T.

ENDIF



  // ne postoji zadata _db_ ali je konekcija napravljena na 'postgres'
     ********************* // dobija se PUBLIC _version_, PgVersion
     __pg__version()  // __pg__connect.prg
     ********************* // uvek traži verziju kad si u konekciji



  *  __pg__disconnect()
     _oSession_:disconnect()  // bez poruke
     _oSession_ := NIL        // ako konekcija ne uspe
                              // ponovo se postavi _oSession_ := NIL







*" ------------------------------------- start
*" AKO NE POSTOJI ZADATA DATABASE = _db_
*" IF NOT EXISTS DEFAULT DATABASE = _db_
*" -------------------------------------
  IF STOP_PROGRAMA==.T.

     // ne postoji zadata _db_ ali je konekcija napravljena na 'postgres'
     // ako se traži samo verzija ovde treba prekinuti program bez poruke
     // jer je __pg__version() izvršena
     IF nn==1
        RETURN .F.
     ELSE

        // ne postoji zadata _db_ ali je konekcija napravljena na 'postgres'
        // ako se traže _db_ liste: database, schemas, tables,
        // nn=2 ili nn=3 tada ovde treba prekinuti program sa porukom
        // jer je samo __pg__version() izvršena, a liste ne mogu biti izvršene

           ec_k()
           c__greska(;
           "NA SERVERU NE POSTOJI BAZA PODATAKA"   +chr(59)+;
           "THERE IS NO DATABASE ON THE SERVER"    +chr(59)+;
           "                                  "    +chr(59)+;
            PADC(_cdbc_,33)                       +chr(59)+;   // za G3=75
           "","STOP",,"14.Consolas Bold")
```

```
           RETURN .F.

     ENDIF // IF nn==1

  ENDIF  // IF STOP_PROGRAMA==.T.
*" -------------------------------------
*" AKO NE POSTOJI ZADATA DATABASE = _db_
*" IF NOT EXISTS DEFAULT DATABASE = _db_
*" ------------------------------------- end


* --- Check if database _db_ exists  end

* --- konektuj se na database 'postgres' da proveriš postojanje database _db_ --- end







*" ---------------------------------- start
*" AKO POSTOJI ZADATA DATABASE = _db_
*" IF EXISTS DEFAULT DATABASE = _db_
*" ----------------------------------


* --- konektuj se na database _db_ da dobiješ liste: databaza, šema i tabli --- start

* PUBLIC VARIJABLE PostgreSQL servera iz C-PGDB.INI
*******************
  __pg__ini()       // __pg__ini.prg
*******************
  __pg__connect()   // __pg__connect.prg
*******************



IF nn==2

   *------------------------------------------------------ Databases list
 // dobija se tek posle DbeSys()
 // i posle konekcije na neku od baza podataka
    PUBLIC _dblist_    := {}
    ************************************  // PG_DATABASE_CREATE.PRG
    _dblist_ := __list__databases(lListView)  // DOBIJANJE LISTE POSTOJEĆIH DATABASE KLASTERA -
-- U RADU
    ************************************  // .T. prikaz liste u appbrowse, .F. nema prikaza
    * stop(_dblist_,1)

   *------------------------------------------------------ Databases list


   *------------------------------------------------------ Schemes list
    PUBLIC _shelist_   := {}
    ****************************************   // PG_SHEMA.PRG
    _shelist_ := __list__schemas(lListView,_db_)   // DOBIJANJE LISTE POSTOJEĆIH SHEMA U
DATABAZI _db_
    ****************************************
    * stop(_shelist_,1)
```

```
   *------------------------------------------------------ Schemes list




   *------------------------------------------------------ Tables list
    PUBLIC _tablist_   := {}
    ********************************************  // PG_SHEMA.PRG
    _tablist_ := __list__tables(lListView,_db_,_she_)   // DOBIJANJE LISTE POSTOJEĆIH TABLI U
SHEMI _she_ U DATABAZI _db_
    ********************************************
    * stop(_tablist_,1)
   *------------------------------------------------------ Tables list

ENDIF


__pg__disconnect()

* --- konektuj se na database _db_ da dobiješ liste: databaza, šema i tabli --- end
* --- connect to database _db_ to get lists: database, schema and boards    --- end

*" ----------------------------------
*" AKO POSTOJI ZADATA DATABASE = _db_
*" IF EXISTS DEFAULT DATABASE = _db_
*" ---------------------------------- end

RETURN .T.




* lista sve databaze u server klasteru iz konekcije na bilo koju od njih
* list all databases in the server cluster from a connection to any of them
************************************************************************** 19-08-2023
FUNCTION __list__databases(lList) // DOBIJANJE LISTE POSTOJEĆIH DATABASES
**************************************************************************
* lList=.T. prikaz liste u appbrowse
* lList=.F. nema prikaza liste u appbrowse

*Server side
*Copy (Select * From foo) To '/tmp/test.csv' With CSV DELIMITER ',';

*Client side
*\copy (Select * From foo) To '/tmp/test.csv' With CSV




// DOBIJANJE LISTE POSTOJEĆIH BAZA PODATAKA
//----------------------------------------
// JAKO VAŽNO (videti da li se ipak nešto može uraditi) 14-09-2018
// lista postojećih baza podataka u klasteru učitava se iz
// systemske table: pg_database, koja se nalazi u sekciji: Catalogs, pa
// u sekciji:  PostgreSQL (pg_catalog) i ima strukturu:
// GET LIST OF EXISTING DATABASES
```

```
//-------------------------------------------------
// VERY IMPORTANT (see if something can be done) 09-14-2018
// the list of existing databases in the cluster is loaded from
// system tables: pg_database, located in the section: Catalogs, pa
// in the section: PostgreSQL (pg_catalog) and has the structure:

// CREATE TABLE pg_database
// (
//  datname       name    NOT NULL,// database name: postgre, template1...
//  datdba        oid     NOT NULL,
//  encoding      integer NOT NULL,  *
//  datistemplate boolean NOT NULL,  *
//  datallowconn  boolean NOT NULL,  *
//  datconnlimit  integer NOT NULL,  *
//  datlastsysoid oid     NOT NULL,
//  datfrozenxid  xid     NOT NULL,
//  dattablespace oid     NOT NULL,
//  datconfig     text[],
//  datacl        aclitem[]
// )
// Iz PgAdmin se lista dobija sa: SELECT datname FROM pg_database;
// Iz Xbase++ aplikacije se dobija sa:
// From PgAdmin, the list is obtained with: SELECT datname FROM pg_database;
// From the Xbase++ application it is obtained with:

LOCAL cSQL:="", oStmt:=nil, aa:={}


    cSQL := "SELECT datname FROM pg_database ORDER BY datname;"
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():query(,@xresult)

    SELECT QUERY
  //---------------------
    aa:={}
    go top
    do while .not. eof()
      aadd(aa,datname)
      skip
    enddo
    * stop(aa,1)
  //---------------------

     IF lList==.T.
       ec_k()
       go top
       appbrowse position center title "List all Database in Server [ Cluster ] " FONT
"11.Consolas"
       APPFIELD nVar := Recno()  CAPTION  "Redni broj"
       APPFIELD datname COLOR GRA_CLR_RED CAPTION  "Database name"
       appdisplay
     ENDIF
  //---------------------
    CLOSE QUERY  // mora se zatvoriti !



* Testovi:
* vidi u appbrowse-ru
* SELECT datname daje samo jedno polje: datname  --------- OK
```

```
/*
    cSQL := "SELECT datname FROM pg_database;"
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():query(,@xresult)

    SELECT QUERY
    go top
    appbrowse position center
    appdisplay
    CLOSE QUERY
*/

* vidi u appbrowse-ru
* SELECT datconfig daje samo jedno polje: datname ---------- ???
/*
    cSQL := "SELECT datconfig FROM pg_database;"
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():query(,@xresult)

    SELECT QUERY
    go top
    appbrowse position center
    appdisplay
    CLOSE QUERY
*/

* vidi u appbrowse-ru
* SELECT * daje samo jedno polje: datname ---------- ???
/*
    cSQL := "SELECT * FROM pg_database;" // WHERE false;"
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():query(,@xresult)

    SELECT QUERY
    go top
    appbrowse position center
    appdisplay
    CLOSE QUERY

*/

RETURN aa



* lista šema samo u onoj databazi na koju je izvršena konekcija
* list of schemes only in the database to which the connection was made
**************************************************************************
FUNCTION __list__schemas(lList,cdatabasename) // lista šema
**************************************************************************
* isto kao ovo gore ali sa pregledom u Xbase++ appbrowse
/*
test=# \dn

        List of schemas
        Name         |  Owner
--------------------+----------
 godina_2019        | postgres
 godina_2020        | postgres
 godina_2021        | postgres
```

```
 information_schema | postgres
 pg_catalog         | postgres
 pg_toast           | postgres
 pg_toast_temp_1    | postgres
 public             | postgres
 test2022           | postgres
 test2023           | postgres
(10 rows)
*/


LOCAL cSQL:="", oStmt:=nil, aa:={}

* We can list all PostgreSQL schemas using the (ANSI) standard INFORMATION_SCHEMA:

    cSQL := "SELECT schema_name FROM information_schema.schemata ORDER BY schema_name;"
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():query(,@xresult)


    SELECT QUERY
  //---------------------
    aa:={}
    go top
    do while .not. eof()
      aadd(aa,schema_name)
      skip
    enddo
    * stop(aa,1)
  //---------------------


    IF lList==.T.

       ec_k()
       go top
       appbrowse position center title "List all Schemas in database [ " +cdatabasename +" ]"
FONT "11.Consolas"
       APPFIELD nVar := Recno()  CAPTION  "Redni broj"
       APPFIELD schema_name COLOR GRA_CLR_RED CAPTION  "Schema name"
       appdisplay

    ENDIF
  //---------------------
    CLOSE QUERY  // mora se zatvoriti !

RETURN aa



* lista table samo na onoj databazi na koju je izvršena konekcija i samo
* na zadatoj šemi koja postoji u toj databazi
* table list only on the database to which the connection was made and only
* on the given schema that exists in that database
********************************************************************** 10.10.2023
FUNCTION __list__tables(lList,cdatabasename,cschemaname) // LISTA TABLI u ŠEMi
**********************************************************************
* lList=.T. prikaži appbrowse listview
*
* cdatabasename = baza podataka na koju je prethodno izvršena konekcija:
```

```
*       - ili je to bez parametra DB u konekcionom stringu
*         koja daje default konekciju na databazu DB='postgres'
*                                                  -------------
*       - ili je to sa zadatum parametrom DB=_db_ u konekcionom stringu
*         koja daje konekciju na zadatu databazu DB=_db_
*                                                  -------
*         Ovde se koristi samo za naslovni deo AppBrowse liste
*         ----------------------------------------------------
*
* cschemaname = 'public' (default) ili _she_
*               i uvek se traži u cdatabasename
*
*---
* lList=.T. show appbrowse listview
*
* cdatabasename = the database to which the connection was previously made:
* - or it is without the DB parameter in the connection string
* which gives the default connection to the database DB='postgres'
* -------------
* - or it is with the default parameter DB=_db_ in the connection string
* which provides a connection to the given database DB=_db_
* -------
* Used here only for the title part of the AppBrowse list
* ------------------------------------------------ ---
*
* cschemaname = 'public' (default) or _she_
* and is always searched in cdatabasename
*




* "LISTA TABLI ZADATE ŠEME --- U AKTIVNOJ  DATABASE
* "DATABASE NAME SE NE ZADAJE !
*
* iz PUBLIC APLIKACIJE koristi PUBLIC: _server_,_db_,_uid_,_pwd_,;
* _osession_,_she_,_tab_,_dblist_,_shelist_,_tablist_
*
* "LIST OF TABLES DEFAULT SCHEME --- IN ACTIVE DATABASE
* "DATABASE NAME IS NOT DEFAULT !
*
* from PUBLIC APPLICATION uses PUBLIC: _server_,_db_,_uid_,_pwd_,;
* _osession_,_she_,_tab_,_dblist_,_shelist_,_tablist_
*

/*
*" IZLISTAJ SVE TABLE U ZADATOJ DATABAZI I ZADATOJ ŠEMI
*-------------------------------------------------------

* test=# SELECT schemaname, tablename FROM pg_catalog.pg_tables where schemaname='public';
*     schemaname     |             tablename
*--------------------+-----------------------------------
* public             | alaska-software.isam.tables
* public             | alaska-software.isam.orders
* public             | alaska-software.system.connections
*/

LOCAL aa:={}, cSQL, oStmt, _cschemaname_

DEFAULT cschemaname TO _she_
```

```
    * --- test
    * prikaži sve nazive tabli u zadatoj šemi: schemaname=public
    * cSQL := "SELECT schemaname, tablename FROM pg_catalog.pg_tables where
schemaname='public';"
    * --- test
    _cschemaname_:="'"+cschemaname+"'"

*****************************************************************************************
***
    cSQL := "SELECT schemaname, tablename FROM pg_catalog.pg_tables where
schemaname="+_cschemaname_+" ORDER BY tablename;"

*****************************************************************************************
***
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():query(,@xresult)
    SELECT QUERY
  //-------------------
    go top
    do while .not. eof()
         * tabla ctablename se nalazi u šemi cschemaname
         aadd(aa,{schemaname,tablename})
      skip
    enddo
    * stop(aa,1)

    IF lList==.T.

       ec_k()
       go top
       appbrowse position center title "List all Tables in Scheme  [
"+cdatabasename+"."+cschemaname+" ]" FONT "11.Consolas"
       *                                                          -------------     ------
----
       APPFIELD nVar := Recno()  CAPTION  "Redni broj"
       APPFIELD tablename  COLOR GRA_CLR_RED CAPTION  "Table name"
       APPFIELD schemaname COLOR GRA_CLR_DARKBLUE CAPTION  "Schema name"
       appdisplay

     ENDIF

    CLOSE QUERY


RETURN aa
```

__PG__SHE.PRG

```
////////////////////////////////////////////////////////////////////////
//                                                                    //
//                                                                    //
//   __pg__she.prg
//                                                                    //
//   25-10-2023                                                       //
//                                                                    //
//   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
* ALERTBOX()   // BAZNE.DLL
* C__GRESKA()  // BAZNE.DLL
* C__PORUKA()  // BAZNE.DLL
* EC_P()       // BAZNE.DLL
* EC_K()       // BAZNE.DLL
* STOP()       // BAZNE.DLL

*
* FUCTIONS in __pg__she.prg
*
* FUNCTION __pg__sheexist()
* FUNCTION __pg__shecreate()
* FUNCTION __pg__shedrop()
*
*
*  Svaka od ovih funkcija može se pozvati kao samostalna funkcija:
*  Each of these functions can be called as a standalone function:

*-----------------------------------
* Xbase++
#include "dac.ch"
* #pragma library("dbfupsize.lib")
// dbfupsize.dll/lib potrebno za: oInfo:=DacSchema():New(_oSession_)
#include "Xbp.ch"
#include "AppEvent.ch"
#include "Common.ch"
* Xbase++ PGDBE
#include "pgdbe.ch"
#include "std.ch"
#include "sql.ch"
#pragma library("adac20b.lib")
*-----------------------------------
* XbTools++
```

```
#include "xbtsys.ch"
* eXpress++
#include "dcdialog.ch"
*-----------------------------------

* koristi se za testiranje programa
*  #include "Appbrow.ch"
*  MEMVAR appObject




/// <descrioption>
/// PROVERA POSTOJANJA ŠEME _she_ u DATABAZI _db_
/// U DBFUPSIZE.DLL postoje funkcije za proveru postojanja databaze i
/// table:
///   oInfo := DacSchema():New(_oSession_)
///   oInfo:isDatabase(_db_)
///   oInfo:isTable(_tab_)
/// ali nema funkcije:
///   oInfo:isScheme(_she_)
/// pa ovde koristim direktnu SQL komandu, koju sam ja iskonstruisao,
/// za proveru postojanja šeme, a ako šema ne postoji kreira se.
/// ======================================================================
/// </description>

/// <description>
/// CHECKING THE EXISTENCE OF SCHEMA _she_ in DATABASE _db_
/// In DBFUPSIZE.DLL there are functions for checking the existence of the database i
/// boards:
/// oInfo := DacSchema():New(_oSession_)
/// oInfo:isDatabase(_db_)
/// oInfo:isTable(_tab_)
/// but no function:
/// oInfo:isScheme(_she_)
/// so here I use a direct SQL command, which I constructed,
/// to check the existence of the scheme, and if the scheme does not exist, it is created.
/// ======================================================== ======================
/// </description>
**************************************************************************
FUNCTION __pg__sheexist()
**************************************************************************
* _db_ = "coba" je database name koje se proverava u database serveru

LOCAL QUERY, cSQL, oStmt, aa := {}, lOk := .F.

IF _she_="public" // ako je scheme name='public' (sistemska šema PgSQL-a)
   RETURN .T.     // ne pokušavaj proveru postojanja i kreiranje šeme.
ENDIF             // Prekini program

*************************
***__pg__ini()               // __pg__ini.prg
*************************
  IF __pg__connect()=.F.     // __pg__connect.prg
     RETURN .F.
  ENDIF
```

```
*************************


   * ---
   * --- Proveri da li postoji šema _she_  | Check if there is a schema _she_
   * ---
   * We can list all PostgreSQL schemas using the (ANSI) standard
   * INFORMATION_SCHEMA:
   * ec_p()
    cSQL := "SELECT schema_name FROM information_schema.schemata;"
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():query(,@xresult)
   * ec_k()
    SELECT QUERY
    aa:={}
    go top
    do while .not. eof()
      aadd(aa,schema_name)
      if schema_name = _she_
         lOk := .T. // na listi šema postoji šema _she_
         exit
      endif
      skip
    enddo
    CLOSE QUERY  // mora se zatvoriti !
   * ---


  IF lOk = .T.

     *" ako šema postoji vraća se .T.
     IF _lMsg_=.T.
         ec_k()
         c_poruka("There is a scheme: "+_she_)
     ENDIF

  ELSE

   *" ako šema ne postoji vraća se .F.
     IF _lMsg_=.T.
         ec_k()
         c_poruka("There is no scheme: "+_she_)
     ENDIF

  ENDIF

*************************
  * __pg__disconnect()  // __pg__connect.prg
   _oSession_:disconnect()
   _oSession_ := NIL  // ako konekcija ne uspe
                      // ponovo se postavi _oSession_ := NIL
*************************

RETURN lOk


/// <descrioption>
/// KREIRANJE ŠEME _she_ u DATABAZI _db_
/// ====================================
```

```
/// Bez provere da li šema postoji
/// </description>

/// <description>
/// CREATE SCHEMA _she_ in DATABASE _db_
/// ==================================
/// Without checking if the schema exists
/// </description>
*********************************************************************
FUNCTION __pg__shecreate()
*********************************************************************
* _db_ = "coba" je database name koje se proverava u database serveru

LOCAL QUERY, cSQL, oStmt, aa := {}, lOk := .F. , lRet := .F.
cComment := "ESIR FISKALNA KASA COBA Systems, www.cobasystems.com"

IF _she_="public"
   RETURN .T.
ENDIF

*************************
***_pg__ini()                // __pg__ini.prg
*************************
  IF __pg__connect()=.F.     // __pg__connect.prg
     RETURN .F.
  ENDIF
*************************


*------------------------------------------ // definiši error codeblock:
bErrorHandler :=ErrorBlock( {|e| Break(e)} )  // prekini kod dođe do greške
BEGIN SEQUENCE                                // Break() radi sa BEGIN SEQUENCE
*------------------------------------------ // definiši error codeblock:
   ec_p()
    cSQL := "CREATE SCHEMA "+_she_+" AUTHORIZATION postgres;"
    cSQL += "GRANT ALL ON SCHEMA "+_she_+" TO postgres;"
    cSQL += "GRANT ALL ON SCHEMA "+_she_+" TO public;"
    cSQL += "COMMENT ON SCHEMA   "+_she_+" IS "+"'"+cComment+"'"+";"

    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()
   ec_k()

lRet := .T.
*-----------------------------------------
         // Break() radi sa RECOVER
RECOVER  // izvrši kod dođe do greške
         // vezano za definisani Break(e)
*-----------------------------------------
lRet := .F.
   ec_k()
   confirmbox(,;
   "ŠEMU NIJE MOGUĆE KREIRATI"         +ch+;
   "Schema ["+_she_+"] already exists" +ch+;
   "in database ["+_db_+"]"            +ch+;
   "This Schema can not be created "   +ch+;
   "","STOP",;
   XBPMB_OK,XBPMB_CRITICAL,;
   XBPMB_SYSMODAL+XBPMB_MOVEABLE)
```

```
END SEQUENCE

ErrorBlock( bErrorHandler ) // reset old codeblock


*************************
  * __pg__disconnect()  // __pg__connect.prg
   _oSession_:disconnect()
   _oSession_ := NIL  // ako konekcija ne uspe
                      // ponovo se postavi _oSession_ := NIL
*************************

RETURN lRet


/// <descrioption>
/// BRISANJE ŠEME _she_ u DATABAZI _db_
/// ==================================
/// uz proveru da li šema postoji IF EXIST
/// </description>

/// <description>
/// DELETE SCHEMA _she_ in DATABASE _db_
/// ==================================
/// checking if the schema exists IF EXIST
/// </description>
*****************************************************************************
FUNCTION __pg__shedrop()
*****************************************************************************
* _db_ = "coba" je database name koje se proverava u database serveru

LOCAL QUERY, cSQL, oStmt, aa := {}, lOk := .F. , lRet := .F.

IF _she_="public"
   RETURN .T.
ENDIF

*************************
***  __pg__ini()             // __pg__ini.prg
*************************
  IF __pg__connect()=.F.     // __pg__connect.prg
     RETURN .F.
  ENDIF
*************************


*------------------------------------------ // definiši error codeblock:
bErrorHandler :=ErrorBlock( {|e| Break(e)} )  // prekini kod dođe do greške
BEGIN SEQUENCE                                // Break() radi sa BEGIN SEQUENCE
*------------------------------------------ // definiši error codeblock:

    ec_p("Drop Schema: "+_she_)
    cSQL := "DROP SCHEMA IF EXISTS "+_she_+" CASCADE;"
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()
    ec_k()
    IF lAlert==.T.
       MsgBox("It's Dropped Schema ["+_she_+"]","OK")
    ENDIF
lRet := .T.
```

```
*-----------------------------------------
         // Break() radi sa RECOVER
RECOVER  // izvrši kod dođe do greške
         // vezano za definisani Break(e)
*-----------------------------------------
lRet := .F.
   ec_k()
   confirmbox(,;
   "ŠEMU NIJE MOGUĆE OBRISATI"              +ch+;
   "Schema ["+_she_+"] already exists"      +ch+;
   "in database ["+_db_+"]"                 +ch+;
   "this Schema can not be dropped"         +ch+;
   "is being accessed by other users "      +ch+;
   "","STOP",;
   XBPMB_OK,XBPMB_CRITICAL,;
   XBPMB_SYSMODAL+XBPMB_MOVEABLE)
*-----------------------------------------
END SEQUENCE

ErrorBlock( bErrorHandler ) // reset old codeblock

*************************
  * __pg__disconnect()  // __pg__connect.prg
   _oSession_:disconnect()
   _oSession_ := NIL  // ako konekcija ne uspe
                      // ponovo se postavi _oSession_ := NIL
*************************

RETURN lRet
```

__PG__TAB.PRG

```
//////////////////////////////////////////////////////////////////////
//                                                                  //
//                                                                  //
//   __pg__tab.prg                                                  //
//                                                                  //
//   01-10-2023                                                     //
//                                                                  //
//   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
* ALERTBOX()   // BAZNE.DLL
* C__GRESKA()  // BAZNE.DLL
* C__PORUKA()  // BAZNE.DLL
* EC_P()       // BAZNE.DLL
* EC_K()       // BAZNE.DLL
* STOP()       // BAZNE.DLL


*
*  FUNCTION in __pg__tab.prg
*
*  FUNCTION __pg__tabexist__coba_systems()   // IF EXISTS 'coba_systems'
*  FUNCTION __pg__tabcreate__coba_systems()  // CREATE TABLE 'coba_systems'
*
*  STATIC FUNCTION create__table__coba_systems() // CREATE TABLE 'coba_systems'
*  STATIC FUNCTION sql__table__coba_systems()    // formira se sql script za 'coba_systems'

*  FUNCTION shema_triggerfunction_isam_rowversion_update()
*  STATIC FUNCTION shematriggerfunction_isam_rowversion_update()
*  FUNCTION shema_triggerfunction__isam_tablemeta_update()
*  STATIC FUNCTION shematriggerfunction_isam_tablemeta_update()
*  ----------
*  FUNCTION drop__table__all(_table_) // DROP ONE TABLE IN DATABASE
*  FUNCTION fill__table__all(_she_,_table_) // FILL ONE TABLE IN DATABASE
*  ----------
*  FUNCTION TABLE_pg(_tablename_) // scheme.table -- nema connect-disconnect
                                  // zamena za funkciju TABLE(table,oSession_)
                                  // koja radi samo sa table 'public'

* FUNCTION TABLE_pg(_tablename_)  // scheme.table -- no connect-disconnect
                                  // replacement for function TABLE(table,oSession_)
                                  // which only works from the 'public' board
```

```
*-----------------------------------
* Xbase++
#include "dac.ch"
* #pragma library("dbfupsize.lib")
// dbfupsize.dll/lib potrebno za: oInfo:=DacSchema():New(_oSession_)
#include "Xbp.ch"
#include "AppEvent.ch"
#include "Common.ch"
* Xbase++ PGDBE
#include "pgdbe.ch"
#include "std.ch"
#include "sql.ch"
#pragma library("adac20b.lib")
*-----------------------------------
* XbTools++
#include "xbtsys.ch"
* eXpress++
#include "dcdialog.ch"
*-----------------------------------

* koristi se za testiranje programa
*  #include "Appbrow.ch"
*  MEMVAR appObject
```

```
/// <descrioption>
///
/// PROVERA POSTOJANJA TABLE coba_systems U DATABAZI _db_ ŠEMI _she_
/// ---------------------------------------------------------------------
/// Ova funkcija se poziva na startu aplikacije posle provere postojanja
/// šeme _she_. Ako šema _she_ postoji tada se proverava postojanje
/// obavezne aplikacione table 'coba_systems' u šemi _she_.
///
///   Ako tabla ne postoji vrati .F.
///   Ako je F. podrazumeva se da ne postoje I ostale table aplikacije
///   Ako tabla postoji vrati .T.
///   Ako je .T. podrazumeva se da postoje sve ostale table aplikacije
///
///   Posle ovoga treba preduzeti sledeće:
///   Ako tabla postoji:
///   - nastavi program
///   Ako tabla ne postoji:
///   - Formiraj tablu 'coba_systems' u šemi _she_
///   - Formiraj sve table aplikacije u šemi _she_
```

```
///
/// </description>

/// <description>
///
/// CHECKING THE EXISTENCE OF TABLE coba_systems IN DATABASE _db_ SCHEMA _she_
/// ------------------------------------------------- --------------------
/// This function is called at the start of the application after checking for existence
/// schemas _she_. If the scheme _she_ exists then the existence is checked
/// required application boards 'coba_systems' in schema _she_.
///
/// If the board does not exist return .F.
/// If it is F, it is understood that there are no other application boards
/// If the board exists return .T.
/// If .T. it is assumed that all other application panels exist
///
/// After this, the following should be done:
/// If the board exists:
/// - continue the program
/// If the board does not exist:
/// - Create board 'coba_systems' in schema _she_
/// - Form all application boards in _she_ scheme
///
/// </description>
*************************************************************************
FUNCTION __pg__tabexist__coba_systems()
*************************************************************************
* iz funkcije __pg__ini()
* _db_  = "coba" je database name koje se proverava u database serveru
* _she_ = "public" je scheme name
* _tab_ = "coba_systems" fiksirano u aplikaciji kao obavezna app tabla
* _shetab_ := _she_+"."+_tab_
* iz funkcije __pg__connect()
* _oSession_


LOCAL cschemaname := _she_
LOCAL ctablename  := "coba_systems"

LOCAL QUERY, cSQL, oStmt, aa := {}, lOk := .F. , lRet := .F.

*************************
***__pg__ini()               // __pg__ini.prg
*************************
  IF __pg__connect()=.F.     // __pg__connect.prg
     RETURN .F.
  ENDIF
*************************


IF _she_ == lower("public")

   // funkcija TABLE() traži samo u šemi 'public'
   lRet := TABLE(ctablename,_oSession_)

ELSE


   // ako šema nije 'public' tada se koristi ovaj metod
    * ec_p()
```

```
    * prikaži sve nazive tabli i sve nazive šema kojima table pripadaju:
    ************************************************************
    cSQL := "SELECT schemaname, tablename FROM pg_catalog.pg_tables;"
    ************************************************************
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():query(,@xresult)
    * ec_k()
    SELECT QUERY
    go top
    do while .not. eof()

      IF alltrim(tablename) ==alltrim(ctablename) .and. ;
         alltrim(schemaname)==alltrim(cschemaname)

         * tabla ctablename se nalazi u šemi cschemaname
         lRet:=.T.
         EXIT

      ENDIF

      skip
    enddo
    CLOSE QUERY

ENDIF // IF _she_ == lower("public")


  IF lRet = .T.

     *" ako tabla postoji vraća se .T.
     IF _lMsg_=.T.
         ec_k()
         c_poruka("There is a table: "+ctablename)
     ENDIF

  ELSE

   *" ako tabla ne postoji vraća se .F.
     IF _lMsg_=.T.
         ec_k()
         c_poruka("there is no table: "+ctablename)
     ENDIF

  ENDIF

*************************
  * __pg__disconnect()  // __pg__connect.prg
   _oSession_:disconnect()
   _oSession_ := NIL  // ako konekcija ne uspe
                      // ponovo se postavi _oSession_ := NIL
*************************

RETURN lRet
```

```
/// <descrioption>
///
/// KREIRANJE TABLE coba_systems U DATABAZI _db_ ŠEMI _she_
/// ---------------------------------------------------------------------
/// Ova funkcija se poziva na startu aplikacije posle provere postojanja
/// šeme _she_. Ako šema _she_ postoji tada se proverava postojanje
/// obavezne aplikacione table 'coba_systems' u šemi _she_.
///
///   Ako tabla ne postoji vrati .F.
///   Ako je F. podrazumeva se da ne postoje I ostale table aplikacije
///   Ako tabla postoji vrati .T.
///   Ako je .T. podrazumeva se da postoje sve ostale table aplikacije
///
///   Posle ovoga treba preduzeti sledeće:
///   Ako tabla postoji:
///   - nastavi program
///   Ako tabla ne postoji:
///   - Formiraj tablu 'coba_systems' u šemi _she_
///   - Formiraj sve table aplikacije u šemi _she_
///
/// Delimični COBA Upsize - za ISAM komande za rad sa upsize tablom
/// Ovde se koriste PGDEBE SQL komande za kreiranje upsize table
///
/// </description>

/// <description>
///
/// CREATE TABLE coba_systems IN DATABASE _db_ SCHEMA _she_
/// ------------------------------------------------- --------------------
/// This function is called at the start of the application after checking for existence
/// schemas _she_. If the scheme _she_ exists then the existence is checked
/// required application boards 'coba_systems' in schema _she_.
///
/// If the board does not exist return .F.
/// If it is F, it is understood that there are no other application boards
/// If the board exists return .T.
/// If .T. it is assumed that all other application panels exist
///
/// After this, the following should be done:
/// If the board exists:
/// - continue the program
/// If the board does not exist:
/// - Create board 'coba_systems' in schema _she_
/// - Form all application boards in _she_ scheme
///
/// Partial COBA Upsize - for ISAM commands to work with the upsize table
/// PGDEBE SQL commands are used here to create the upsize table
///
/// </description>
**************************************************************************
FUNCTION __pg__tabcreate__coba_systems()
**************************************************************************
* _db_ = "coba" je database name koje se proverava u database serveru

LOCAL lRet, cDatab, cOwner, cErrMsg, oInfo
LOCAL QUERY, cSQL, oStmt, aa := {}, lOk := .F.
cComment := "ESIR FISKALNA KASA COBA Systems, www.cobasystems.com"
```

```
************************
***__pg__ini()               // __pg__ini.prg
************************
  IF __pg__connect()=.F.     // __pg__connect.prg
     RETURN .F.
  ENDIF
************************
```

```
  *" ------------------------------------------------------
  *"                   Table coba_systems
  *" ------------------------------------------------------
  *-- samo u ovom PRG test programu koriste se ove
  *-- fiksne varijable za naziv scheme i naziv table:

  PRIVATE;
  cschemename := _she_               ,;
  ctablename  := "coba_systems"      ,;
  ctable:=cschemename+"."+ctablename ,;
  ckey:=ctablename+"_pkey"

  *" ------------------------------------------------------
  *"                   Table coba_systems
  *" ------------------------------------------------------
```

```
// bez poruka
    cSQL := "DROP TABLE IF EXISTS "+ctable+" CASCADE;"
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()

// Prvo se kreiraju u okviru shema_public dve Xbase++ neophodne funkcije
// za dodavanje novih tabli sa CREATE TABLE iz bilo koje Alaska Xbase++
// aplikacije koja radi sa (PGDBE)
// inače se ne mogu izvesti komande: CREATE TABLE :
//---
// First, two necessary Xbase++ functions are created in the schema_public frame
// to add new tables with CREATE TABLE from any Alaska Xbase++
// applications that work with (PGDBE)
// otherwise the commands cannot be executed: CREATE TABLE :
    ********************************************
     shema_triggerfunction_isam_rowversion_update()    // ovde U P S I Z E
    ********************************************
     shema_triggerfunction__isam_tablemeta_update()    // ovde U P S I Z E
    ********************************************

    ****************************
    create__table__coba_systems()  // CREATE TABLE IN DATABASE
    ****************************
```

```
************************
  * __pg__disconnect()  // __pg__connect.prg
   _oSession_:disconnect()
   _oSession_ := NIL  // ako konekcija ne uspe
                      // ponovo se postavi _oSession_ := NIL
************************


RETURN NIL







*********************************************************************
STATIC FUNCTION create__table__coba_systems() // CREATE TABLE IN DATABASE
*********************************************************************
// bez poruka



    ec_p()
    // formira se SQL script i/ili SQL string za upotrebu u SQL komandi
    ************************************
     cSQL := sql__table__coba_systems()
    ************************************
    // obrada i izvršenje PGDBE SQL komande
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()


    // * prikaz sadržaja SQL stringa cSQL koji se šalje kroz PGDBE SQL komandu
    //    memowrit("cSQL.txt",cSQL)
    //    runshell("cSQL.txt","notepad.exe")
    //---
    // * display the contents of the cSQL SQL string sent through the PGDBE SQL command
    // memowrite("cSQL.txt",cSQL)
    // runshell("cSQL.txt","notepad.exe")


    // rekonstrukcija table --- VACUUM FULL zaključava tablu
    // FULL može povratiti više prostora, ali traje mnogo duže
    // i isključivo zaključava tablu.
    // Ovo zahtijeva dodatni prostor na disku, jer piše novu
    // kopiju table i ne oslobađa staru dok se operacija ne završi.
    // Obično ovo treba koristiti samo kada je potrebno povratiti
    // značajnu količinu prostora unutar tabele.
    //
    // Inače treba umesto opcije: FULL koristiti opciju: ANALYZE
    // koja samo ažurira statistiku koju koristi planer da odredi
    // najefikasniji način za izvršavanje upita.
    // Opcija FREEZE vrši agresivno "zamrzavanje" torki.
    //
    // Opcijae FREEZE je ekvivalentna izvođenju VACUUM
```

```
    // s parametrima vacuum_freeze_min_age i vacuum_freeze_table_age
    // postavljenim na nulu. Agresivno zamrzavanje se uvijek izvodi
    // kada se tabela ponovo piše, tako da je ova opcija suvišna
    // kada je specificirana opcija FULL.
    //---
    // rebuild the board --- VACUUM FULL locks the board
    // FULL can reclaim more space, but takes much longer
    // and exclusively locks the board.
    // This requires additional disk space, as it is writing a new one
    // a copy of the table and does not release the old one until the operation is complete.
    // Normally this should only be used when a rollback is required
    // significant amount of space inside the table.
    //
    // Otherwise, instead of the option: FULL, you should use the option: ANALYZE
    // which just updates the statistics used by the scheduler to determine
    // most efficient way to execute the query.
    // FREEZE option aggressively "freezes" tuples.
    //
    // The FREEZE option is equivalent to executing VACUUM
    // with parameters vacuum_freeze_min_age and vacuum_freeze_table_age
    // set to zero. Aggressive freezing is always performed
    // when the table is being rewritten, so this option is redundant
    // when the FULL option is specified.


        cSQL := "VACUUM FULL "+ctable+";"
        oStmt := DacSqlStatement():fromChar(cSQL)
        oStmt:build():execute()
        ec_k()

RETURN NIL









*************************************************************************
STATIC FUNCTION sql__table__coba_systems() // formira se sql script
*************************************************************************
LOCAL txt := "", cr := chr(13)+chr(10)




*#########################################################################  START
TEXT INTO sqltxt WRAP cr // TRIMMED
-- ******************************************
-- CREATE TABLE:
-- ******************************************
--  Table kreiraj sa dodatnim kolonama za
--  U P S I Z E  Z A  I S A M  M A Š I N U:
--  __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 kroba_11_pkey PRIMARY KEY (__record)

-- Table kreiraj
CREATE TABLE &ctable
  (
  MANUFACTURER_       varchar(100)      ,
  COMPANY_            varchar(100)      ,
  ADDRESS_            varchar(50)       ,
  ADDRESSCODE_        varchar(30)       ,
  CITY_               varchar(50)       ,
  ZIPCODE_            varchar(10)       ,
  COUNTRY_            varchar(50)       ,
  PERSON_             varchar(100)      ,
  COUNTRYCODE_        varchar(10)       ,
  CURRENCY_           varchar(10)       ,
  TELEPHONE_          varchar(50)       ,
  EMAIL_              varchar(50)       ,
  WEBSITE_            varchar(50)       ,
  ACCOUNT_            varchar(30)       ,
  IBAN_               varchar(30)       ,
  MBR_                varchar(30)       ,
  PIB_                varchar(30)       ,
  DELATNOST_          varchar(30)       ,
  DAT1_               date              ,
  DAT2_               date              ,
  ROK_                integer           ,
  PRICE_              numeric(12,2)     ,
  GRUPA_              varchar(50)       ,
  STATUS_             varchar(50)       ,
  FLAG_               boolean           ,
  UUID_               varchar(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);

-- ********************************************
-- O W N E R:
-- ********************************************
-- mora se postaviti user na 'postgres' (superuser)
-- morate biti logovani na database kao superuser

ALTER TABLE &ctable OWNER TO postgres;

-- ********************************************
-- I N D E K S I
-- ********************************************

-- Index: coba_systems__deleted      --- "uvek obavezno kod upsize tehnologije
-- DROP INDEX coba_systems__deleted; --- "always mandatory with upsize technology

CREATE INDEX coba_systems__deleted
  ON &ctable
  USING btree
```

```
  (__deleted);

-- Index: coba_systems__record --- "uvek obavezno kod upsize tehnologije
-- DROP INDEX coba_systems__record;

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



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

-- Trigger: coba_systems_isam_rowversion on kup   --- "uvek obavezno kod upsize tehnologije
-- DROP TRIGGER coba_systems_isam_rowversion ON kup;

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


---- Function: isam_rowversion_update()            --- "uvek obavezno kod upsize tehnologije
---- DROP FUNCTION isam_rowversion_update();
--
--CREATE OR REPLACE FUNCTION isam_rowversion_update()
--  RETURNS trigger AS
--$BODY$
----- version:1.1
--begin
--  if( TG_OP = 'UPDATE' ) then
--    NEW.__rowversion := OLD.__rowversion + 1;
--  end if;
--  return NEW;
--end;
--$BODY$
--  LANGUAGE 'plpgsql' VOLATILE
--  COST 100;
--ALTER FUNCTION isam_rowversion_update() OWNER TO postgres;


-- ******************************************

-- Trigger: coba_systems_isam_tablemeta on kup    --- "uvek obavezno kod upsize tehnologije
-- DROP TRIGGER coba_systems_isam_tablemeta ON kup;

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


---- Function: isam_tablemeta_update() --- "uvek obavezno kod upsize tehnologije
---- DROP FUNCTION isam_tablemeta_update();
--
```

```
--CREATE OR REPLACE FUNCTION isam_tablemeta_update()
--  RETURNS trigger AS
--$BODY$
--begin
--  if( TG_OP = 'INSERT' ) then
--    update "alaska-software.isam.tables" SET record_count = record_count+1, update_count =
update_count +1,updated=current_timestamp WHERE table_name=TG_RELNAME;
--  end if;
--  if( TG_OP = 'DELETE' ) then
--    update "alaska-software.isam.tables" SET record_count = record_count-1, update_count =
update_count +1,updated=current_timestamp WHERE table_name=TG_RELNAME;
--  end if;
--  if( TG_OP = 'UPDATE' ) then
--    update "alaska-software.isam.tables" SET update_count = update_count
+1,updated=current_timestamp WHERE table_name=TG_RELNAME;
--  end if;
--  return NULL;
--end;
--$BODY$
--  LANGUAGE 'plpgsql' VOLATILE
--  COST 100;
--ALTER FUNCTION isam_tablemeta_update() OWNER TO postgres;


-- ********************************************
-- O B A V E Z N E  S E Q U E N C E
-- ********************************************

-- Sequence: "alaska-software.isam.orders_order_id_seq"  --- "uvek obavezno kod upsize
tehnologije
-- DROP SEQUENCE "alaska-software.isam.orders_order_id_seq";

--CREATE SEQUENCE "alaska-software.isam.orders_order_id_seq"
--  INCREMENT 1
--  MINVALUE 1
--  MAXVALUE 9223372036854775807
--  START 979
--  CACHE 1;
--ALTER TABLE "alaska-software.isam.orders_order_id_seq" OWNER TO postgres;



---- Sequence: "alaska-software.isam.tables_table_id_seq"  --- "uvek obavezno kod upsize
tehnologije
---- DROP SEQUENCE "alaska-software.isam.tables_table_id_seq";
--
--CREATE SEQUENCE "alaska-software.isam.tables_table_id_seq"
--  INCREMENT 1
--  MINVALUE 1
--  MAXVALUE 9223372036854775807
--  START 50
--  CACHE 1;
--ALTER TABLE "alaska-software.isam.tables_table_id_seq" OWNER TO postgres;



---- Sequence: "alaska-software.system.connections_connection_id_seq" --- "uvek obavezno kod
upsize tehnologije
---- DROP SEQUENCE "alaska-software.system.connections_connection_id_seq";
--
```

```
--CREATE SEQUENCE "alaska-software.system.connections_connection_id_seq"
--  INCREMENT 1
--  MINVALUE 1
--  MAXVALUE 9223372036854775807
--  START 606
--  CACHE 1;
--ALTER TABLE "alaska-software.system.connections_connection_id_seq" OWNER TO postgres;



---- Sequence: roba_11___record_seq  --- "uvek obavezno kod upsize tehnologije
---- DROP SEQUENCE roba_11___record_seq;
--
--CREATE SEQUENCE roba_11___record_seq
--  INCREMENT 1
--  MINVALUE 1
--  MAXVALUE 9223372036854775807
--  START 197
--  CACHE 1;
--ALTER TABLE roba_11___record_seq OWNER TO postgres;

-- ******************************************


ENDTEXT
*#########################################################################  END

* "u ovom momentu postoji formiran string: sqltext
* "at this moment there is a formed string: sqltext

RETURN sqltxt









/// <descrioption>
///
/// --- COMMON ---
/// KREIRANJE TRIGGER FUNKCIJA U ŠEMI 'public'
/// ------------------------------------------------------------------
/// OVE ISTE FUNKCIJE KORISTE SE KOD KREIRANJA SVAKE TABLE U DATABAZI !
///
/// Ove funkcije kreiraju dve obavezne trigger funkcije za dva obavezna
/// trigger-a koji se ugrađuju u svaku UPSIZE TABLU databaze _db_

/// Ove trigere formira program DBFUPSIZE.EXE kada DBF fajl transformiše
/// u postgreSQL UPSIZE TABLU na koju se moggu primenjivazi ISAM KOMANDE
/// I FUNKCIJE - ISAM NAVIGACIJA korišćena za DBF ISAM bazu podataka.
///
/// Umesto upotrebe programa DBFUPSIZE.EXE/DLL/LIB ja ovde iz koda
/// kreiram UPSIZE TABLU i njen public_key, indx-e, trigger-e i trigger
/// funkcije. Sekvence i ostalo automatski kreira PGDBE. Primer za ovo
```

```
/// dat je u prethodnim funkcijama.
///
/// </description>
///
/// --- COMMON ---

/// <description>
///
/// --- COMMON ---
/// CREATING TRIGGER FUNCTIONS IN 'public' SCHEMA
/// ------------------------------------------------- --------------------
/// THESE SAME FUNCTIONS ARE USED WHEN CREATING EACH TABLE IN THE DATABASE!
///
/// These functions create two mandatory trigger functions for two mandatory
/// triggers that are embedded in each UPSIZE TABLE of the _db_ database

/// These triggers are formed by the program DBFUPSIZE.EXE when the DBF file is transformed
/// in a postgreSQL UPSIZE TABLE to which ISAM COMMANDS can be applied
/// AND FUNCTIONS - ISAM NAVIGATION used for DBF ISAM database.
///
/// Instead of using the program DBFUPSIZE.EXE/DLL/LIB I here from the code
/// I create the UPSIZE TABLE and its public_key, indx-e, trigger-e and trigger
/// functions. Sequences and other things are created automatically by PGDBE. An example of
this
/// is given in previous functions.
///
/// </description>
///
/// --- COMMON ---

* obavezne UPSIZE funkcije: nalaze se u __pg__tab.prg
* mandatory UPSIZE functions: located in __pg__tab.prg
*************************************************************************
FUNCTION shema_triggerfunction_isam_rowversion_update()
*************************************************************************
    // formira se sql script kao string varijabla: cSQL
    cSQL := shematriggerfunction_isam_rowversion_update()
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()

   * MsgBox("I S A M  M A S C H I N E"  +chr(13)+;
   *        "Create trigger function for shema: public"+chr(13)+;
   *        "function _isam_rowversion_update() for ISAM navigate")
RETURN NIL
*************************************************************************
STATIC FUNCTION shematriggerfunction_isam_rowversion_update()
*************************************************************************
LOCAL txt := "", cr := chr(13)+chr(10)

TEXT INTO sqltxt WRAP cr // TRIMMED
-- Function: isam_rowversion_update()
-- DROP FUNCTION isam_rowversion_update();
CREATE OR REPLACE FUNCTION isam_rowversion_update()
  RETURNS trigger AS
$BODY$
begin
  if( TG_OP = 'UPDATE' ) then
    NEW.__rowversion := OLD.__rowversion + 1;
  end if;
  return NULL; -- no result as it is a after trigger
```

```
end;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE
  COST 100;
ALTER FUNCTION isam_rowversion_update() OWNER TO postgres;
ENDTEXT
RETURN sqltxt




/// --- COMMON ---
* obavezne UPSIZE funkcije: nalaze se u __pg__tab.prg
* mandatory UPSIZE functions: located in __pg__tab.prg
*************************************************************************
FUNCTION shema_triggerfunction__isam_tablemeta_update()
*************************************************************************
    // formira se sql script kao string varijabla: cSQL
    cSQL := shematriggerfunction_isam_tablemeta_update()
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()

   * MsgBox("I S A M  M A S C H I N E"  +chr(13)+;
   *        "It's Created trigger function for shema: public"+chr(13)+;
   *        "function _isam_tablemeta_update() for ISAM navigate")
RETURN NIL
*************************************************************************
STATIC FUNCTION shematriggerfunction_isam_tablemeta_update()
*************************************************************************
LOCAL txt := "", cr := chr(13)+chr(10)

TEXT INTO sqltxt WRAP cr // TRIMMED
-- Function: isam_tablemeta_update()
-- DROP FUNCTION isam_tablemeta_update();

CREATE OR REPLACE FUNCTION isam_tablemeta_update()
  RETURNS trigger AS
$BODY$
begin
  if( TG_OP = 'INSERT' ) then
    update "alaska-software.isam.tables" SET record_count = record_count+1, update_count =
update_count +1,updated=current_timestamp WHERE table_name=TG_RELNAME;
  end if;
  if( TG_OP = 'DELETE' ) then
    update "alaska-software.isam.tables" SET record_count = record_count-1, update_count =
update_count +1,updated=current_timestamp WHERE table_name=TG_RELNAME;
  end if;
  if( TG_OP = 'UPDATE' ) then
    update "alaska-software.isam.tables" SET update_count = update_count
+1,updated=current_timestamp WHERE table_name=TG_RELNAME;
  end if;
  return NULL;
end;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE
  COST 100;
```

```
ALTER FUNCTION isam_tablemeta_update() OWNER TO postgres;
ENDTEXT
RETURN sqltxt
```

```
/// <descrioption>
///
/// --- COMMON ---
/// BRISANJE-UKLANJANJE-DROP TABLE _table_ iz database _db_ iz šeme _she_
/// ---------------------------------------------------------------------
/// Ova funkcija se poziva bilo gde iz aplikacije
/// - pre toga se mora diskonektovati sa database
/// - ova funkcija se konektuje-briše-diskonektuje
/// - posle toga se mora konektovati na databazu
///
/// vraća NIL
/// Operacija Počinje sa upitom, završava sa porukom
///
/// FUNKCIJA SE KORISTI ZA SVAKU TABLU, MENJA SE SAMO PARAMETAR _table_
///
/// </description>

/// <description>
///
/// --- COMMON ---
/// DELETE-REMOVE-DROP TABLE _table_ from database _db_ from schema _she_
/// ------------------------------------------------- --------------------
/// This function is called anywhere from the application
/// - before that it must be disconnected from the database
/// - this function connects-deletes-disconnects
/// - after that it must be connected to the database
///
/// returns NIL
/// Operation Starts with a query, ends with a message
///
/// THE FUNCTION IS USED FOR EACH TABLE, ONLY THE _table_ PARAMETER IS CHANGED
///
/// </description>
**********************************************************************
FUNCTION drop__table__all(_table_) // DROP TABLE IN DATABASE
**********************************************************************
* _table_ = naziv table koja se puni podacima
*" FUNKCIJA SE KORISTI ZA SVAKU TABLU MENJA SE SAMO PARAMETAR _table_
* _table_ = the name of the table to be filled with data
*" THE FUNCTION IS USED FOR EACH TABLE ONLY THE _table_ PARAMETER IS CHANGED
LOCAL cSQL, cStmt

************************
***__pg__ini()               // __pg__ini.prg
************************
```

```
  *" ------------------------------------------------------
  *"                   Table = _table_
  *" ------------------------------------------------------
  LOCAL;
  cschemename := _she_   ,; // "public"
  ctablename  := _table_ ,; // "kupci"
  ctable:=cschemename+"."+ctablename ,;
  ckey:=ctablename+"_pkey"
  *" ------------------------------------------------------
  *"                   Table = _table_
  *" ------------------------------------------------------
        ec_k()
        IF c__neda(;
                "TABLA ( "+ctable+" )"                    + chr(59)+;
                " "                                       + chr(59)+;
                "TABLA ĆE BITI OBRISANA IZ BAZE PODATAKA" + chr(59)+;
                "TABLE WILL BE DELETED FROM DATABASE"     + chr(59)+;
                " ","DELETING TABLE FROM DATABASE",,"V") = .F.

                RETURN NIL // P R E K I D  P R O G R A M A

        ENDIF

*************************
  IF __pg__connect()=.F.     // __pg__connect.prg
     RETURN .F.
  ENDIF
*************************
    ec_p()
    cSQL := "DROP TABLE IF EXISTS "+ctable+" CASCADE;"
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()
    ec_k()
*************************
  * __pg__disconnect()  // __pg__connect.prg
   _oSession_:disconnect()
   _oSession_ := NIL  // ako konekcija ne uspe
                      // ponovo se postavi _oSession_ := NIL
*************************

        ec_k()
        c__poruka(;
                "TABLA ( "+ctable+" )"                 + chr(59)+;
                " "                                    + chr(59)+;
                "THE TABLE IS DELETED FROM DATABASE" + chr(59)+;
                "TABLA JE OBRISANA IZ BAZE PODATAKA" + chr(59)+;
                " ","OK",,"V")

RETURN NIL




/// <descrioption>
///
```

```
/// --- COMMON ---
/// PUNJENJE TABLE _table_ iz database _db_ iz šeme _she_ PODACIMA
/// ----------------------------------------------------------------
/// Ova funkcija se poziva bilo gde iz aplikacije
/// - pre toga se mora diskonektovati sa database
/// - ova funkcija se konektuje-brišetablu-punitablu-diskonektuje
///
///  * pre poziva ove funkcije vrši se diskonekcija sa database
///  * konekcija i diskonekcija obavljaju se u samoj funkciji.
///  * Funkcija poziva __pg__ini() i uzima PUBLIC _db_ i _she_
///  * Funkcija koristi LOCAL _table_ koja dolazi kao parametar
///  * Funkcija koristi SQL script fajl
///  * PUBLIC sqlfile_tabledata := gde_exe()+"\SQL\"+ctablename+".SQL"
///
/// 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.
///
/// vraća NIL
/// Operacija Počinje sa upitom, završava sa porukom
///
/// FUNKCIJA SE KORISTI ZA SVAKU TABLU, MENJA SE SAMO PARAMETAR _table_
///
/// </description>


/// <description>
///
/// --- COMMON ---
/// FILLING TABLE _table_ from database _db_ from schema _she_ WITH DATA
/// ---------------------------------------------------- ---------------------
/// This function is called anywhere from the application
/// - before that it must be disconnected from the database
/// - this function connects-clears the table-fills the table-disconnects
///
/// * before calling this function, the database is disconnected
/// * connection and disconnection are performed in the function itself.
/// * Function calls __pg__ini() and takes PUBLIC _db_ and _she_
/// * The function uses the LOCAL _table_ which comes as a parameter
/// * The function uses a SQL script file
/// * PUBLIC sqlfile_tabledata := gde_exe()+"\SQL\"+ctablename+".SQL"
///
/// The table is filled with data from the SQL script file named table.
/// The SQL script file is created from a special service program for
/// data transfer from DBF file to SQL file, or for data transfer
/// from PgSQL table to SQL script file, or for data transfer from EXCEL
/// tables to SQL file, or to transfer data from CSV text file to
/// SQL file, or to transfer data from XML and JSON files to SQL file.
///
/// returns NIL
/// Operation Starts with a query, ends with a message
///
/// THE FUNCTION IS USED FOR EACH TABLE, ONLY THE _table_ PARAMETER IS CHANGED
///
/// </description>

**********************************************************************
```

```
FUNCTION fill__table__all(_table_) // FILL TABLE IN DATABASE
*********************************************************************
* _table_ = naziv table koja se puni podacima
*" FUNKCIJA SE KORISTI ZA SVAKU TABLU MENJA SE SAMO PARAMETAR _table_
* _table_ = the name of the table to be filled with data
*" THE FUNCTION IS USED FOR EACH TABLE ONLY THE _table_ PARAMETER IS CHANGED
LOCAL cSQL, cStmt

*************************
__pg__ini()               // __pg__ini.prg
*************************


  *" ------------------------------------------------------
  *"                   Table = _table_
  *" ------------------------------------------------------
  *-- samo u ovom PRG test programu koriste se ove
  *-- fiksne varijable za naziv scheme i naziv table:

  PRIVATE ;
  cschemename := _she_               ,;  // "public"
  ctablename  := _table_             ,;  // "kupci"
  ctable:=cschemename+"."+ctablename ,;
  ckey:=ctablename+"_pkey"

  * ako postoji fajl:

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

  * gde_exe()   // bazne.dll // gde_exe()=strtran(appname(.T.),"\"+appname()) = c:\app"
  * user_name() // bazne.dll // user_name="coba"

  IF FILE(gde_exe()+"\SQL","D")=.F.
     DirMake( gde_exe()+"\SQL" )  // XbtSys.ch
  ENDIF

  * iz njega se preuzimju svi slogovi-records u tablu,
  * ali samo iz onih kolona-polja koja su decidirano
  * navedena u INSTERT INTO ctable (polje1, polje2,...)
  *" ------------------------------------------------------
  *"                   Table = _table_
  *" ------------------------------------------------------


*---------- PREUZIMANJE PODATAKA IZ SQL SCRIPTA --------------- start


    IF FILE(sqlfile_tabledata)==.T.

        ec_k()
        IF c__neda(;
                "SQL script file found:"         + chr(59)+;
                "     "+sqlfile_tabledata        + chr(59)+;
                "with data to fill the table:"   + chr(59)+;
                "     "+ctable                   + chr(59)+;
                " "                              + chr(59)+;
                "TABLE WILL BE FILLED WITH DATA" + chr(59)+;
                " "                              + chr(59)+;
                "Nađen je SQL script fajl:"      + chr(59)+;
```

```
                "     "+sqlfile_tabledata        + chr(59)+;
                "sa podacima za punjenje table:" + chr(59)+;
                "     "+ctable                   + chr(59)+;
                " "                              + chr(59)+;
                "PUNI SE TABLA SA TIM PODACIMA"  + chr(59)+;
                "","FILLING TABLE WITH DATA",,"V") = .F.

                RETURN NIL // P R E K I D  P R O G R A M A

        ELSE
                // NASTAVI PROGRAM:
        ENDIF

    ELSE

        ec_k()
        c__greska(;
                "SQL script file not found:"     + chr(59)+;
                "     "+sqlfile_tabledata        + chr(59)+;
                "with data to fill the table:"   + chr(59)+;
                "     "+ctable                   + chr(59)+;
                " "                              + chr(59)+;
                "DATA IN TABLE DOES NOT CHANGE"  + chr(59)+;
                " "                              + chr(59)+;
                "Nije nađen SQL script fajl:"    + chr(59)+;
                "     "+sqlfile_tabledata        + chr(59)+;
                "sa podacima za punjenje table:" + chr(59)+;
                "     "+ctable                   + chr(59)+;
                " "                              + chr(59)+;
                "NE MENJAJU SE PODACI U TABLI"   + chr(59)+;
                "","FILLING TABLE WITH DATA",,"V")

        RETURN NIL  // P R E K I D  P R O G R A M A

    ENDIF // IF FILE(sqlfile_tabledata)==.T.

*"  NEOPHODAN USLOV
*"  ---------------
*"  Mora da postoji SQL script fajl: sqlfile_tabledata
*"  Naziv SQL script fajla mora biti naziv table,
*"  a sadržaj mora odgovarati strukturi table. SQL fajl
*"  mora da ima komandu INSER INTO na 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

*" Učita se ovaj fajl u cSQL string:
************************************************************
cSQL := alltrim( memoread( sqlfile_tabledata ) )
************************************************************


    // * prikaz sadržaja SQL stringa cSQL koji se šalje kroz PGDBE SQL komandu
    //   runshell(sqlfile_tabledata,"notepad.exe")




*" Ovaj cSQL string se izvrši iz PGDBE SQL komande:
*************************
  IF __pg__connect()=.F.     // __pg__connect.prg
     RETURN .F.
  ENDIF
*************************

    ec_p()

 *---
    // brisanje svih redova iz table:

    xcSQL:="DELETE FROM "+ctable+";"
    oStmt := DacSqlStatement():fromChar(xcSQL)
    oStmt:build():execute()

    xcSQL := "VACUUM FULL "+ctable+";"
    oStmt := DacSqlStatement():fromChar(xcSQL)
    oStmt:build():execute()

 *---

    // obrada i izvršenje PGDBE SQL komande
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()

    cSQL := "VACUUM FULL "+ctable+";"
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()

 * ---

    ec_k()
```

```
************************
  * __pg__disconnect()  // __pg__connect.prg
   _oSession_:disconnect()
   _oSession_ := NIL  // ako konekcija ne uspe
                      // ponovo se postavi _oSession_ := NIL
************************

        ec_k()
        c__poruka(;
                "TABLA ( "+ctable+" )"           + chr(59)+;
                " "                              + chr(59)+;
                "THE TABLE IS FILLED WITH DATA"  + chr(59)+;
                "TABLA JE NAPUNJENA PODACIMA "   + chr(59)+;
                "","FILLING TABLE WITH DATA",,"V")


*---------- PREUZIMANJE PODATAKA IZ SQL SCRIPTA -------------- end

RETURN NIL







/// <descrioption>
///
/// PROVERA POSTOJANJA TABLE _tablename_ U DATABAZI _db_ ŠEMI _she_
/// ---------------------------------------------------------------------
///   Ako tabla ne postoji vrati .F.
///   Ako tabla postoji vrati .T.
///   Koristi se passtrough SQL
///
///   Zahteva prethodnu konekciju na databazu
///
/// </description>

/// <description>
///
/// CHECKING THE EXISTENCE OF TABLE _tablename_ IN DATABASE _db_ SCHEMA _she_
/// ------------------------------------------------- --------------------
/// If the board does not exist return .F.
/// If the board exists return .T.
/// Passthrough SQL is used
///
/// Requires a previous connection to the database
///
/// </description>
*************************************************************************
FUNCTION TABLE_pg(_she_,_table_) // scheme.table -- nema connect-disconnect
*************************************************************************
* iz funkcije __pg__ini()  // C-PGDB.DLL
*  _db_  = "coba"   je database name koje se proverava u database serveru
*  _she_ = "public" je scheme name
* iz funkcije __pg__connect() // C-PGDB.DLL
*  _oSession_       je sesija
*---
* from function __pg__ini() // C-PGDB.DLL
```

```
* _db_ = "coba" is the database name that is checked in the database server
* _she_ = "public" is the scheme name
* from function __pg__connect() // C-PGDB.DLL
* _oSession_ is a session



LOCAL c_schemaname := _she_
LOCAL c_tablename  := _table_

LOCAL QUERY, cSQL, oStmt, aa := {}, lRet := .F.

    * u konektovanoj databazi u šema.tabla: pg_catalog.pg_tables
    * popisane su sve šeme i sve njihove table:
    *   u koloni schemaname = naziv šeme
    *   u koloni tablename  = naziv table
    * U ovom spisku (query) traži zadatu šema.tabla
    * i ako je nađeš vrati TRUE, u protivnom vrati FALSE

    * in the connected database in schema.table: pg_catalog.pg_tables
    * all schemes and all their tables are listed:
    * in column schemaname = schema name
    * in column tablename = table name
    * In this list (query) searches for the given scheme.table
    * and if you find it, return TRUE, otherwise return FALSE

    ****************************************************************
    cSQL := "SELECT schemaname, tablename FROM pg_catalog.pg_tables;"
    ****************************************************************
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():query(,@xresult)
    * ec_k()
    SELECT QUERY
    go top
    do while .not. eof()

      IF alltrim(tablename) ==alltrim(c_tablename) .and. ;
         alltrim(schemaname)==alltrim(c_schemaname)

         * tabla c_tablename se nalazi u šemi c_schemaname
         lRet:=.T.
         EXIT

      ENDIF

      skip
    enddo
    CLOSE QUERY

RETURN lRet
```

__TAB__CREATE.PRG

```
//////////////////////////////////////////////////////////////////////
//                                                                  //
//                                                                  //
//   __tab__create.prg                                              //
//                                                                  //
//   25-10-2023                                                     //
//                                                                  //
//   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
* ALERTBOX()   // BAZNE.DLL
* C__GRESKA()  // BAZNE.DLL
* C__PORUKA()  // BAZNE.DLL
* EC_P()       // BAZNE.DLL
* EC_K()       // BAZNE.DLL
* STOP()       // BAZNE.DLL



*-------------------------------------------------------------------------
* Program:
* __tab__create.prg koristi podatke iz array liste '_aListAppTable_'
* koja se formira u EXE aplikaciji iz funkcije:
*      _aListAppTable_ := __tab__create__app() // __tab__create__app.prg
* Program:
* __tab__create.prg uses data from the array list '_aListAppTable_'
* which is formed in the EXE application from the function:
* _aListAppTable_ := __tab__create__app()      // __tab__create__app.prg
*-------------------------------------------------------------------------
* FUNCTION in __tab__create.prg
*
* FUNCTION __tab__create()
*   STATIC FUNCTION drop__tables()    // DROP TABLES IN DATABASE
*   STATIC FUNCTION create__tables()  // CREATE TABLES IN DATABASE
*
* FUNCTION __tab__control()           // KONTROLA IF EXISTS TABLE IN DATABASE IN SCHEMES
*
* FUNCTION __tab__control__ISAM__()   // IF EXISTS TABLE IN LIST "alaska-software.isam.tables"
* FUNCTION __tab__control__ISAM()     // IF EXISTS TABLE IN LIST "alaska-software.isam.tables"
*
*  obavezne UPSIZE funkcije: nalaze se u __pg__tab.prg
*  mandatory UPSIZE functions: located in __pg__tab.prg
*  FUNCTION shema_triggerfunction_isam_rowversion_update()
```

```
*  FUNCTION shematriggerfunction_isam_rowversion_update()
*  FUNCTION shema_triggerfunction__isam_tablemeta_update()
*  FUNCTION shematriggerfunction_isam_tablemeta_update()
*------------------------------------------------------------------------

/*
   Note:

   UPSIZE TABLE - ISAM MAŠINA
   nema polja za emulaciju DBFNTX indeksa
   nema polje za Full Text Search (FTS)

   KREIRANJE VEĆEG BROJA TABLI U ŠEMI public
   ================================================
   BEZ PUNJENJA TABLI PODACIMA KOJI SE
   PREUZIMAJU IZ SQL SCRIPT FAJLA.
   POSLE KREIRANJA SVAKE TABLE IDE NAREDBA:
       VACUUM FULL tablename;


    UPSIZE TABLE - ISAM MACHINE
    no DBFNTX index emulation field
    no Full Text Search (FTS) field

    CREATION OF MORE BOARDS IN SCHEME public
    ================================================
    WITHOUT FILLING THE PANELS WITH THE DATA THAT IS
    DOWNLOAD FROM SQL SCRIPT FILE.
    AFTER CREATING EACH PANEL, THE COMMAND GOES:
        VACUUM FULL tablename;
*/



*-----------------------------------
* Xbase++
#include "dac.ch"
* #pragma library("dbfupsize.lib")
// dbfupsize.dll/lib potrebno za: oInfo:=DacSchema():New(_oSession_)
#include "Xbp.ch"
#include "AppEvent.ch"
#include "Common.ch"
* Xbase++ PGDBE
#include "pgdbe.ch"
#include "std.ch"
#include "sql.ch"
#pragma library("adac20b.lib")
*-----------------------------------
* XbTools++
#include "xbtsys.ch"
* eXpress++
#include "dcdialog.ch"
*-----------------------------------

* koristi se za testiranje programa
*  #include "Appbrow.ch"
*  MEMVAR appObject
```

```
/// <descrioption>
///
/// KREIRANJE VIŠE TABLI DATABAZE APLIKACIJE U DATABAZI _db_ ŠEMI _she_
/// ---------------------------------------------------------------------
///
/// Sa array liste tabli '_aListAppTable_' program učitava nazive tabli
/// i sql stringove za kreiranje tabli, za sve table koje se kreiraju.
/// Ova array lista kreira se iz svake EXE aplikacije sa tablama te
/// aplikacije iz funkcije:            --------------
///
/// _aListAppTable_ := __tab__create__app() // __tab__create__app.prg
///
/// Funkcija __tab__create__app() ne nalazi se u C_PGDB.DLL library
/// ---------------------------------------------------------------------
///  __tab__create() se poziva na startu aplikacije po provere postojanja
/// šeme _she_ iz __pg__connect__app() -> __pg__connect.prg
/// Ako šema _she_ postoji tada se proverava postojanje obavezne
/// aplikacione table 'coba_systems' u šemi _she_.
///
///   Ako tabla ne postoji vrati .F.
///   Ako je F. podrazumeva se da ne postoje i ostale table aplikacije
///   Ako tabla postoji vrati .T.
///   Ako je .T. podrazumeva se da postoje sve ostale table aplikacije
///
///   Posle ovoga treba preduzeti sledeće:
///   Ako tabla postoji:
///   - nastavi program
///   Ako tabla ne postoji:
///   - Formiraj tablu 'coba_systems' u šemi _she_
///   - FORMIRAJ SVE TABLE APLIKACIJE U ŠEMI _she_
///
/// FUNKCIJA FORMIRA SVE TABLE APLIKACIJE, ALI SAMO ONE KOJE NE KORISTI
/// NI JEDNA DRUGA APLIKACIJA U PROGRAMSKOM PAKETU ZA POSLOVANJE FIRME
/// TABLE KOJE SE KORISTE I OD STRANE DRUGIH APLIKACIJA KREIRAJU SE IZ
/// POSEBNIH PROGRAMA - IZ POSEBNIH PRG MODULA.
///
/// Delimični COBA Upsize - za ISAM komande za rad sa upsize tablom
/// Ovde se koriste PGDEBE SQL komande za kreiranje upsize table
///
/// Važno:
/// Funkcija se može pozvati iz bilo koje tačke aplikacije i ona će
/// obrisati sve postojeće table aplikacije a zatim će ih ponovo
/// kreirati kao prazne table.
///
/// </description>


/// <description>
///
/// CREATE MULTIPLE APPLICATION DATABASE TABLES IN DATABASE _db_ SCHEMA _she_
/// ------------------------------------------------- --------------------
///
/// From the array list of tables '_aListAppTable_' the program loads the names of the tables
/// and sql strings for creating tables, for all tables that are created.
/// This array list is created from each EXE application with tables and
```

```
/// applications from function: --------------
///
/// _aListAppTable_ := __tab__create__app() // __tab__create__app.prg
///
/// The __tab__create__app() function is not in the C_PGDB.DLL library
/// ------------------------------------------------- --------------------
/// __tab__create() is called at the start of the application to check for existence
/// schemas _she_ from __pg__connect__app() -> __pg__connect.prg
/// If the schema _she_ exists then the existence of the mandatory is checked
/// application boards 'coba_systems' in schema _she_.
///
/// If the board does not exist return .F.
/// If it is F, it is understood that there are no other application boards
/// If the board exists return .T.
/// If .T. it is assumed that all other application panels exist
///
/// After this, the following should be done:
/// If the board exists:
/// - continue the program
/// If the board does not exist:
/// - Create board 'coba_systems' in schema _she_
/// - FORM ALL APPLICATION TABLES IN SCHEMA _she_
///
/// THE FUNCTION FORM ALL TABLES OF THE APPLICATION, BUT ONLY THE ONES IT IS NOT USING
/// NO OTHER APPLICATION IN THE COMPANY'S BUSINESS SOFTWARE PACKAGE
/// BOARDS THAT ARE ALSO USED BY OTHER APPLICATIONS ARE CREATED FROM
/// SPECIAL PROGRAMS - FROM SPECIAL PRG MODULES.
///
/// Partial COBA Upsize - for ISAM commands to work with the upsize table
/// PGDEBE SQL commands are used here to create the upsize table
///
/// Important:
/// The function can be called from any point in the application and it will
/// clear all existing application boards and then recreate them
/// create as empty boards.
///
/// </description>

*" Delimični COBA Upsize za ISAM komande za rad sa upsize tablom
*" PGDEBE SQL komande za kreiranje upsize table
*" Partial COBA Upsize for ISAM commands to work with the upsize board
*" PGDEBE SQL command to create an upsize table
*************************************************************************
FUNCTION __tab__create()
*************************************************************************
* iz funkcije __pg__ini()
* _db_  = "coba" je database name koje se proverava u database serveru
* _she_ = "public" je scheme name
* _tab_ = "coba_systems" fiksirano u aplikaciji kao obavezna app tabla
*
* iz funkcije __pg__connect()
* _oSession_

LOCAL QUERY, cSQL, oStmt, aa := {}, lOk := .F., i := 0
cComment := "ESIR FISKALNA KASA COBA Systems, www.cobasystems.com"



//----------------------------------------
// UPOTREBA FUNKCIJE KOJA NIJE U C-PGDB.DLL
```

```
//----------------------------------------
  *
  * u COBA Systems EXE aplikaciji, iz funkcije:
  *
  * _aListAppTable_ := __tab__create__app() // __tab__create__app.prg
  *
  * Formira se array '_aListAppTable_' koji sadrži n članova,
  * gde je svaki član grupa podataka za jednu SQL tablu:
  *
  *       {tablename, schemenametablename, sqlstring}
  *
  * tablename            = naziv SQL table
  * schemenametablename  = naziv šeme.naziv SQL table
  * sqlstring            = SQL string za kreiranje svake od tabli iz liste
  *
  * Ova lista '_aListAppTable_' koristi se iz funkcije:
  *      __tab__create()       // __tab__create.prg
  *      __tab__all__control() // __tab__create.prg
  * za preuzimanje strigova: tablename, schemenametablename i sqlstring
  * za CREATE TABLE i druge komande koje se koriste iz C-PGDB.DLL
  *
//---------------------------------------
// USING A FUNCTION NOT IN C-PGDB.DLL
//---------------------------------------
   *
   * in the COBA Systems EXE application, from the function:
   *
   * _aListAppTable_ := __tab__create__app() // __tab__create__app.prg
   *
   * The array '_aListAppTable_' is formed, which contains n members,
   * where each member is a data group for one SQL table:
   *
   * {tablename, schemenametablename, sqlstring}
   *
   * tablename = SQL table name
   * schemenametablename = name of the scheme. name of the SQL table
   * sqlstring = SQL string to create each table from the list
   *
   * This list '_aListAppTable_' is used from the function:
   * __tab__create() // __tab__create.prg
   * __tab__all__control() // __tab__create.prg
   * to retrieve strings: tablename, schemenametablename and sqlstring
   * for CREATE TABLE and other commands used from C-PGDB.DLL
   *

  **********************************
  MFUNCTION := "__tab__create__app()"
  PUBLIC _aListAppTable_ := &MFUNCTION
  **********************************
  IF LEN(_aListAppTable_)=0
     __pg__ini()  // __pg__ini.prg
     ec_k()
     c__greska("Aplikacija nema Zadatu Listu Tabli" +chr(59)+;
               "Application does not have a Default Table List" +chr(59)+;
               "     za databazu "+_db_              +chr(59)+;
               "     za šemu "+_she_                 +chr(59)+;
               " ","STOP |  CREATE TABLE",,"G3")
     RETURN .F.
  ENDIF
  /*
```

```
  * --- test start
  for i=1 to len(  _aListAppTable_  )
  dc_memoedit( _aListAppTable_[i,1]+chr(13)+chr(10)+;
               _aListAppTable_[i,2]+chr(13)+chr(10)+;
               _aListAppTable_[i,3]+chr(13)+chr(10)+;
               "")
     if i=11
        exit
     endif
  next i
  QUIT
  * --- test end
  */
//----------------------------------------
// UPOTREBA FUNKCIJE KOJA NIJE U C_PGDB.DLL
// USING A FUNCTION NOT IN C-PGDB.DLL
//--------------------------------------



*************************
***__pg__ini()               // __pg__ini.prg
*************************
  IF __pg__connect()=.F.     // __pg__connect.prg
     RETURN .F.
  ENDIF
*************************




    **************
    drop__tables()      // DROP MORE TABLE IF EXIST IN DATABASE
    **************


//
// Prvo se kreiraju u okviru shema_public dve Xbase++ neophodne funkcije
// za dodavanje novih tabli sa CREATE TABLE iz bilo koje Alaska Xbase++
// aplikacije koja radi sa (PGDBE)
// inače se ne mogu izvesti komande: CREATE TABLE :
//
// First, two necessary Xbase++ functions are created in the schema_public frame
// to add new tables with CREATE TABLE from any Alaska Xbase++
// applications that work with (PGDBE)
// otherwise the commands cannot be executed: CREATE TABLE :
//
    ***********************************************
     shema_triggerfunction_isam_rowversion_update()    // ovde U P S I Z E
    ***********************************************
     shema_triggerfunction__isam_tablemeta_update()    // ovde U P S I Z E
    ***********************************************

    ****************
    create__tables()  // CREATE MORE TABLE IN DATABASE
    ****************

*************************
  * __pg__disconnect()  // __pg__connect.prg
   _oSession_:disconnect()
```

```
   _oSession_ := NIL  // ako konekcija ne uspe
                      // ponovo se postavi _oSession_ := NIL
*************************



RETURN NIL



*" BRISANJE SVIH TABLI IZ SCHEME public
*" DELETING ALL TABLES FROM THE SCHEME _she_
*****************************************************************************
STATIC FUNCTION drop__tables() // DROP TABLES IN DATABASE
*****************************************************************************
LOCAL i:=0, tablename, schemenametablename, sqlstring
LOCAL cSQL, cStmt

FOR i=1 TO LEN( _aListAppTable_ )

   tablename           := _aListAppTable_[i,1]
   schemenametablename := _aListAppTable_[i,2]
   sqlstring           := _aListAppTable_[i,3]
   ****************************************
    cSQL := "DROP TABLE IF EXISTS "+schemenametablename+" CASCADE;"
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()

NEXT i


RETURN NIL



*" KREIRANJE SVIH TABLI U SCHEME _she_
*" CREATING ALL TABLES IN SCHEMA _she_
*****************************************************************************
STATIC FUNCTION create__tables()  // CREATE TABLES IN DATABASE
*****************************************************************************
LOCAL i:=0, tablename, schemenametablename, sqlstring
LOCAL cSQL, cStmt

FOR i=1 TO LEN( _aListAppTable_ )

   tablename           := _aListAppTable_[i,1]
   schemenametablename := _aListAppTable_[i,2]
   sqlstring           := _aListAppTable_[i,3]
   * dc_memoedit( sqlstring )

   ****************************************
    // formira se SQL script i/ili SQL string
    // za upotrebu u PGDBE SQL komandi
    cSQL := sqlstring
    // obrada i izvršenje PGDBE SQL komande
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()
   ****************************************
    // rekonstrukcija formirane prazne i/ili
    // napunjene table:
```

```
    cSQL := "VACUUM FULL "+schemenametablename+";"
    // obrada i izvršenje PGDBE SQL komande
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()
   ****************************************
    // rekonstrukcija table --- VACUUM FULL zaključava tablu
    // FULL može povratiti više prostora, ali traje mnogo duže
    // i isključivo zaključava tablu.
    // Ovo zahtijeva dodatni prostor na disku, jer piše novu
    // kopiju table i ne oslobađa staru dok se operacija ne završi.
    // Obično ovo treba koristiti samo kada je potrebno povratiti
    // značajnu količinu prostora unutar tabele.
    //
    // Inače treba umesto opcije: FULL koristiti opciju: ANALYZE
    // koja samo ažurira statistiku koju koristi planer da odredi
    // najefikasniji način za izvršavanje upita.
    // Opcija FREEZE vrši agresivno "zamrzavanje" torki.
    //
    // Opcijae FREEZE je ekvivalentna izvođenju VACUUM
    // s parametrima vacuum_freeze_min_age i vacuum_freeze_table_age
    // postavljenim na nulu. Agresivno zamrzavanje se uvijek izvodi
    // kada se tabela ponovo piše, tako da je ova opcija suvišna
    // kada je specificirana opcija FULL.
    //---
     // rebuild the board --- VACUUM FULL locks the board
     // FULL can reclaim more space, but takes much longer
     // and exclusively locks the board.
     // This requires additional disk space, as it is writing a new one
     // a copy of the table and does not release the old one until the operation is complete.
     // Normally this should only be used when a rollback is required
     // significant amount of space inside the table.
     //
     // Otherwise, instead of the option: FULL, you should use the option: ANALYZE
     // which just updates the statistics used by the scheduler to determine
     // most efficient way to execute the query.
     // FREEZE option aggressively "freezes" tuples.
     //
     // The FREEZE option is equivalent to executing VACUUM
     // with parameters vacuum_freeze_min_age and vacuum_freeze_table_age
     // set to zero. Aggressive freezing is always performed
     // when the table is being rewritten, so this option is redundant
     // when the FULL option is specified.
   ****************************************

NEXT i


RETURN NIL
```

```
#include "dmlb.ch"
*************************************************************************
FUNCTION __tab__control() // KONTROLA IF EXISTS TABLE
*************************************************************************
LOCAL i:=0, lRet:=.T., tablename:=""



//-----------------------------------------
// UPOTREBA FUNKCIJE KOJA NIJE U C-PGDB.DLL
//-----------------------------------------
  *
  * u COBA Systems EXE aplikaciji, iz funkcije:
  *
  * _aListAppTable_ := __tab__create__app() // __tab__create__app.prg
  *
  * Formira se array '_aListAppTable_' koji sadrži n članova,
  * gde je svaki član grupa podataka za jednu SQL tablu:
  *
  *       {tablename, schemenametablename, sqlstring}
  *
  * tablename            = naziv SQL table
  * schemenametablename  = naziv šeme.naziv SQL table
  * sqlstring            = SQL string za kreiranje svake od tabli iz liste
  *
  * Ova lista '_aListAppTable_' koristi se iz funkcije:
  *      __tab__create()       // __tab__create.prg
  *      __tab__all__control() // __tab__create.prg
  * za preuzimanje strigova: tablename, schemenametablename i sqlstring
  * za CREATE TABLE i druge komande koje se koriste iz C-PGDB.DLL
  *---

//---------------------------------------
// USING A FUNCTION NOT IN C-PGDB.DLL
//---------------------------------------
   *
   * in the COBA Systems EXE application, from the function:
   *
   * _aListAppTable_ := __tab__create__app() // __tab__create__app.prg
   *
   * The array '_aListAppTable_' is formed, which contains n members,
   * where each member is a data group for one SQL table:
   *
   * {tablename, schemenametablename, sqlstring}
   *
   * tablename = SQL table name
   * schemenametablename = name of the scheme. name of the SQL table
   * sqlstring = SQL string to create each table from the list
   *
   * This list '_aListAppTable_' is used from the function:
   * __tab__create() // __tab__create.prg
   * __tab__all__control() // __tab__create.prg
   * to retrieve strings: tablename, schemenametablename and sqlstring
   * for CREATE TABLE and other commands used from C-PGDB.DLL
   *---
```

```
  **********************************
  MFUNCTION := "__tab__create__app()"
  PUBLIC _aListAppTable_ := &MFUNCTION
  **********************************
  IF LEN(_aListAppTable_)=0
     __pg__ini()  // __pg__ini.prg
     ec_k()
     c__greska("Aplikacija nema Zadatu Listu Tabli" +chr(59)+;
               "Application does not have a Default Table List" +chr(59)+;
               "     za databazu "+_db_             +chr(59)+;
               "     za šemu "+_she_                +chr(59)+;
               " ","STOP |  CONTROL IF EXISTS TABLE",,"G3")
     RETURN .F.
  ENDIF
  /*
  * --- test start
  for i=1 to len(  _aListAppTable_  )
  dc_memoedit( _aListAppTable_[i,1]+chr(13)+chr(10)+;
               _aListAppTable_[i,2]+chr(13)+chr(10)+;
               _aListAppTable_[i,3]+chr(13)+chr(10)+;
               "")
     if i=11
        exit
     endif
  next i
  QUIT
  * --- test end
  */
//-----------------------------------------
// UPOTREBA FUNKCIJE KOJA NIJE U C_PGDB.DLL
// USING A FUNCTION NOT IN C-PGDB.DLL
//-----------------------------------------


//------------------------------------------------------------------ START
// SISTEM KONTROLE KOJI RADI SAMO ZA scheme='public' sa 'tablename'
// CONTROL SYSTEM ONLY WORKS FOR scheme='public' with 'tablename'
//------------------------------------------------------------------

* DbeLoad("PGDBE")
* DBESETDEFAULT("PGDBE")
* aa:=dbeinfo()

* __pg__ini()    // _lMsg_
__pg__connect()  // mora postojati konekcija i _oSession_ objekat
                 // za funkciju TABLE()
lRet := .T.

FOR i=1 TO LEN( _aListAppTable_ )

   tablename           := _aListAppTable_[i,1]
   schemenametablename := _aListAppTable_[i,2]
   sqlstring           := _aListAppTable_[i,3]
   *** IF TABLE(tablename, _oSession_)=.F. // radi samo sa šemom 'public'
   IF TABLE_pg(_she_,tablename)==.F.       // radi sa stringom 'scheme.table' // __pg__tab.prg
      ec_k()
      c__greska(tablename+";ne postoji - not exists","TABLA: ")
      lRet:=.F.
```

```
   ENDIF

NEXT i

__pg__disconnect()

//----------------------------------------------------------------
// SISTEM KONTROLE KOJI RADI SAMO ZA scheme='public' sa 'tablename'
// CONTROL SYSTEM ONLY WORKS FOR scheme='public' with 'tablename'
//---------------------------------------------------------------- END


/*
//---------------------------------------------------------------- START
// SISTEM KONTROLE KOJI RADI ZA SVE scheme sa 'schemenametablename'
// CONTROL SYSTEM THAT WORKS FOR ALL schemes with 'schemenametablename'
//----------------------------------------------------------------


FOR i=1 TO LEN( _aListAppTable_ )

   tablename           := _aListAppTable_[i,1]
   schemenametablename := _aListAppTable_[i,2]
   sqlstring           := _aListAppTable_[i,3]

   she := strtran(schemenametablename,"."+tablename)
   IF TABLE_PG(she,tablename)=.F.    // __pg__tab.prg
      c__greska(schemenametablename+";ne postoji - not exists","TABLA: ")
      lRet:=.F.
   ENDIF

NEXT i

//----------------------------------------------------------------
// SISTEM KONTROLE KOJI RADI ZA SVE scheme sa 'schemenametablename'
// CONTROL SYSTEM THAT WORKS FOR ALL schemes with 'schemenametablename'
//---------------------------------------------------------------- END
*/


IF lRet=.T.
   if _lMsg_=.T.
   ec_k()
   c__poruka("SVE TABLE U BAZI PODATAKA SU NA BROJU;THERE ARE ALL TABLES IN THE DATABASE","OK")
   endif
ELSE
   if _lMsg_=.T.
   ec_k()
   c__greska("NEDOSTAJU TABLE U BAZI PODATAKA;TABLES IN THE DATABASE ARE MISSING","STOP")
   endif
ENDIF


RETURN lREt
```

```
* FUNCTION __tab__control__ISAM__()
* FUNCTION __tab__control__ISAM() // IF EXISTS TABLE IN LIST "alaska-software.isam.tables"

*************************************************************************
FUNCTION __tab__control__ISAM__()
*************************************************************************

IF c__sifra(5)=.F.
   RETURN NIL
ENDIF

    PRIVATE number_rows_in_table := 0
    PRIVATE number_table_in_app  := 0

    ******************************************************
    lRet:=__tab__control__ISAM() // KONTROLA IF EXISTS TABLE
    ******************************************************

IF lRet=.T.


   ec_k()
   c_poruka("SVE TABLE U BAZI PODATAKA SU NA BROJU I"  + chr(13)+;
            "UPISANE SU U DATABAZU U LISTU ISAM TABLI"  + chr(13)+;
            "","OK | BROJ UPISANIH TABLI = "+var2char( number_rows_in_table ))


ELSE


   ec_k()
   c_greska("NE POSTOJE SVE TABLE APLIKACIJE U BAZI PODATAKA : "+var2char(number_table_in_app)
+ chr(13)+;
            "POSTOJEĆE SU UPISANE DATABAZU U LISTU ISAM TABLI: "+var2char( number_rows_in_table
)  + chr(13)+;
            "","STOP | BROJ UPISANIH TABLI = "+var2char( number_rows_in_table ))

ENDIF

RETURN NIL



*************************************************************************
FUNCTION __tab__control__ISAM() // KONTROLA IF EXISTS TABLE
*************************************************************************
* OVU FUNKCIJU ZA UPIS NAZIVA TABLI U SISTEMSKI SPISAK ISAM TABLI:
*             tabla = "alaska-software.isam.tables"
* TREBA POZVATI SAMO POSLE KREIRANJE BAZE PODATAKA I SVIH TABLI
* U ŠEMI "public".
*
*
* Kada se kreira baza podataka i kada se posle toga u tablu isam.tables
* upišu nazivi tabli te baze podataka - te aplikacije iz ove funkcije
* uz svaku upisanu tablu upiše se naziv i id broj te table
* počevši od broja 1 pa na dalje, u polje:
*                 "table_id" Primary Key Integer,
* Svako ponovno pozivanje ove funkcije, uvećava id broj table na sledeći
* način: ako je prva tabla imala broj 1 a poslednja tabla broj 90, tada
```

```
* su novi brojevi uvećani za 90, odnosno od 91 do 180
* pa ako se taj broj koristi u aplikaciji, biće njegova vrednost uvek
* posle ponovnog starta ove funkcije na ovaj način promenjena !
*
* Ova funkcija se može pozvati komandnim tasterom ALT+DEL iz menija
* aplikacije



* u sistemsku Alaska Xbase++ UPSIZE tablu: "alaska-software.isam.tables"
* treba upisati nazive svih tabli apliakcije koje se koriste u databazi
* aplikacije.
* Program prvo izbroji sve redove table (može i bez ovoga)
* Program prvo obriše sve redove iz table: "alaska-software.isam.tables"
* Program proverava postojanje svake table iz array liste _aListAppTable_
* plus proverava postojanje table: "coba_systems"
* u databazi, pa za one table koje postoje u databazi, upisuje naziv table
* u tablu: "alaska-software.isam.tables"

      * Note:
      * aplikacija sa Coba (skraćenom) ISAM navigacijom:
      * bez indeksa, bez FTS i bez DBFNTX i FOXCDX DBE
      * već samo sa PGDBE, funkcionisaće i sa praznom
      * tablom: alaska-software.isam.tables

*---


* THIS FUNCTION FOR REGISTRATION OF TABLE NAMES IN THE SYSTEM LIST OF ISAM TABLES:
* table = "alaska-software.isam.tables"
* SHOULD BE CALLED ONLY AFTER CREATION OF DATABASE AND ALL TABLES
* In SCHEME "public".
*
*
* When the database is created and after that in the table isam.tables
* enter the names of the tables and databases - and applications from this function
* the name and id number of that table are written next to each entered table
* starting from number 1 and onward, in the field:
* "table_id" Primary Key Integer,
* Every time this function is called again, it increases the id number of the table to the next
one
* method: if the first board had number 1 and the last board had number 90, then
* are new numbers increased by 90, i.e. from 91 to 180
* so if that number is used in the application, it will always be its value
* after restarting this function changed in this way!
*
* This function can be called with the command key ALT+DEL from the menu
* applications

* to the system Alaska Xbase++ UPSIZE table: "alaska-software.isam.tables"
* the names of all the application tables used in the database should be entered
* applications.
* The program first counts all the rows of the table (can be done without this)
* The program first deletes all rows from the table: "alaska-software.isam.tables"
* The program checks the existence of each table from the array list _aListAppTable_
* plus checks for table existence: "coba_systems"
* in the database, so for those tables that exist in the database, enter the name of the table
* in table: "alaska-software.isam.tables"

       * Notes:
```

```
       * application with Coba (abbreviated) ISAM navigation:
       * no index, no FTS and no DBFNTX and FOXCDX DBE
       * but only with PGDBE, it will also work with blank
       * table: alaska-software.isam.tables


LOCAL i:=0, lRet:=.T., tablename:=""



//----------------------------------------
// UPOTREBA FUNKCIJE KOJA NIJE U C_PGDB.DLL
//----------------------------------------
  *
  * u COBA Systems EXE aplikaciji, iz funkcije:
  *
  * _aListAppTable_ := __tab__create__app() // __tab__create__app.prg
  *
  * Formira se array '_aListAppTable_' koji sadrži n članova,
  * gde je svaki član grupa podataka za jednu SQL tablu:
  *
  *       {tablename, schemenametablename, sqlstring}
  *
  * tablename            = naziv SQL table
  * schemenametablename  = naziv šeme.naziv SQL table
  * sqlstring            = SQL string za kreiranje svake od tabli iz liste
  *
  * Ova lista '_aListAppTable_' koristi se iz funkcije:
  *      __tab__create()       // __tab__create.prg
  *      __tab__all__control() // __tab__create.prg
  * za preuzimanje strigova: tablename, schemenametablename i sqlstring
  * za CREATE TABLE i druge komande koje se koriste iz C_PGDB.DLL
  *
  ************************************
  MFUNCTION := "__tab__create__app()"
  PUBLIC _aListAppTable_ := &MFUNCTION
  ************************************

  number_table_in_app := LEN(_aListAppTable_)

  IF LEN(_aListAppTable_)=0
     __pg__ini()  // __pg__ini.prg
     ec_k()
     c__greska("Aplikacija nema Zadatu Listu Tabli" +chr(59)+;
               "     za databazu "+_db_             +chr(59)+;
               "     za šemu "+_she_                +chr(59)+;
               " ","STOP |  CONTROL IF EXISTS TABLE",,"G3")
     RETURN .F.
  ENDIF
  /*
  * --- test start
  for i=1 to len(  _aListAppTable_  )
  dc_memoedit( _aListAppTable_[i,1]+chr(13)+chr(10)+;
               _aListAppTable_[i,2]+chr(13)+chr(10)+;
               _aListAppTable_[i,3]+chr(13)+chr(10)+;
               "")
     if i=11
        exit
     endif
  next i
```

```
  QUIT
  * --- test end
  */
//-----------------------------------------
// UPOTREBA FUNKCIJE KOJA NIJE U C_PGDB.DLL
//-----------------------------------------



* __pg__ini()    // _lMsg_
__pg__connect()  // mora postojati konekcija i _oSession_ objekat
                 // za funkciju TABLE()
lRet := .T.


ec_p()


   *------------- SCHEME NAME ----------------------------------------------------
   * schemaname := strtran(schemenametablename,tablename)
     schemaname := strtran(_aListAppTable_[1,2],_aListAppTable_[1,1])
     schemaname := strtran(schemaname,".")
   * "public.kup" -> "public"

   * ------------ SINTAKSA JE OVDE VAŽNA: ----------------------------------------
   * tabla := '"'+'alaska-software.isam.tables'+'"'  // OK
   * tabla := '"'+'public'+'"' +'.'+ '"'+'alaska-software.isam.tables'+'"'  // OK
     tabla := '"'+schemaname+'"' +'.'+ '"'+'alaska-software.isam.tables'+'"'  // OK

   *-------------------------------------------------------------------------------

   ****************************************** UKUPAN BROJ SVIH REDOVA TABLE
   ** SELECT COUNT(*) FROM "public"."alaska-software.isam.tables" ;
   *cSQL := "SELECT COUNT(*) FROM " + tabla + ";" // izbroji sve redove table
   *oStmt := DacSqlStatement():fromChar(cSQL)
   *oStmt:build():query(,@xresult)
   *select query
   *number_rows_in_table:=count // stop(count,1)
   *close query
   *// stop( "Broj tabli ove aplikacije upisan u alaska-software.isam.tables = "+var2char(
number_rows_in_table ),1)
   ******************************************

   ****************************************** BRISANJE SVIH REDOVA IZ TABLE
   * DELETE FROM "public"."alaska-software.isam.tables" ;
   cSQL := "DELETE FROM " + tabla + ";" // obriši sve redove iz table
     // obrada i izvršenje PGDBE SQL komande
   oStmt := DacSqlStatement():fromChar(cSQL)
   oStmt:build():execute()
   ******************************************


*-----------------------------------------------------------------------
* koristi samo 'public' šemu
* IF TABLE("coba_systems", _oSession_)=.F.
*    ec_k()
*    c__greska("coba_systems"+";ne postoji","TABLA: ")
*    lRet:=.F.
* ELSE
```

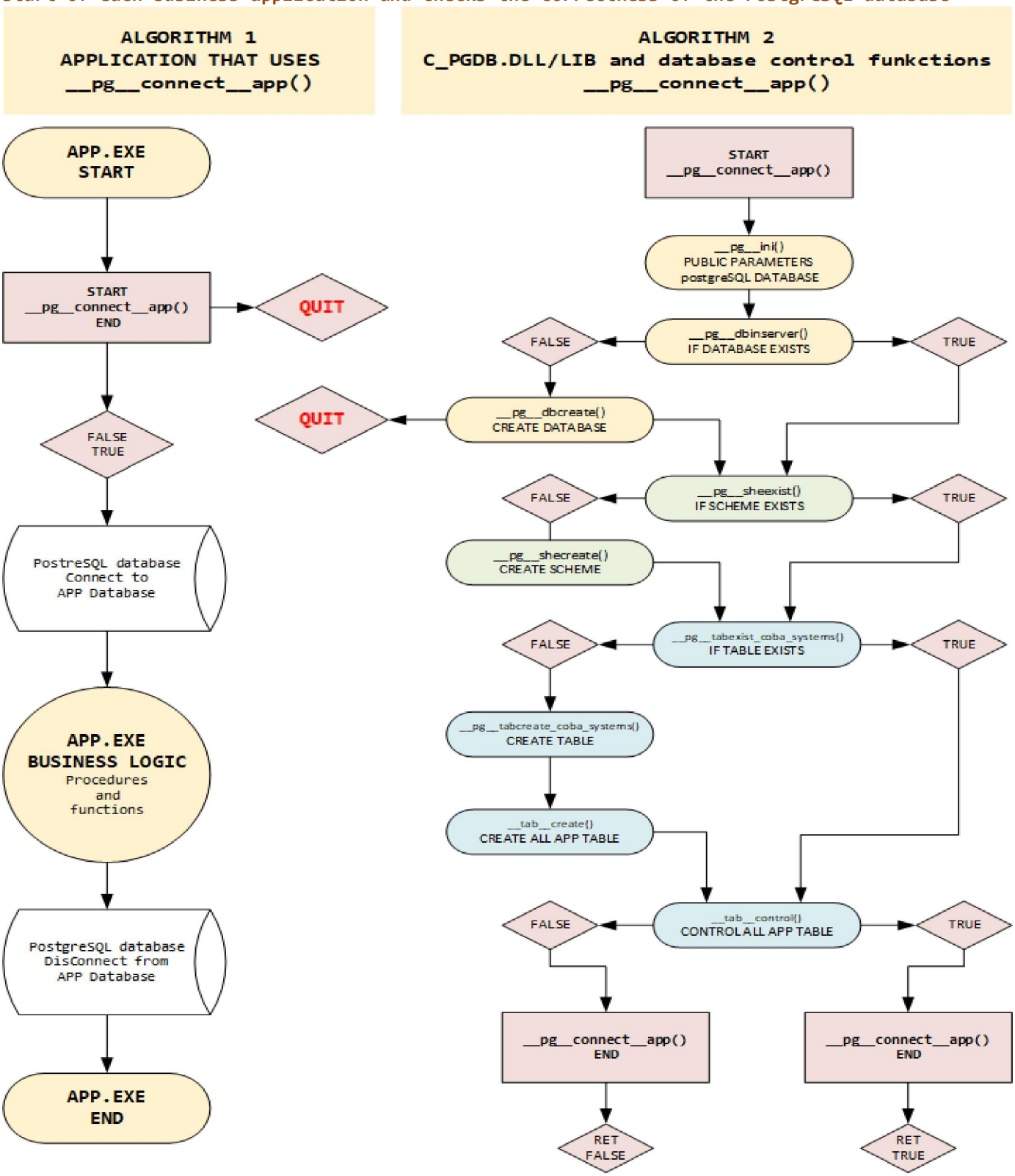
```
* koristi schema.table string
IF TABLE_pg(_she_,"coba_systems")=.F.  // __pg__tab.prg
   ec_k()
   c__greska("coba_systems"+";ne postoji","TABLA: ")
   lRet:=.F.
ELSE



// formira se SQL script i/ili SQL string
// za upotrebu u PGDBE SQL komandi
// Upis podataka u sve kolone table: &tabla="public"."alaska-software.isam.tables"
TEXT INTO cSQL WRAP
INSERT INTO &tabla
(
  table_name
 ,unc_name
 ,updated
 ,created
 ,record_count
 ,update_count
 ,lock_owner
)
VALUES
(
 'coba_systems'
 ,null               -- unc_name - character varying(255)
 ,CURRENT_TIMESTAMP -- updated - timestamp without time zone
 ,CURRENT_TIMESTAMP -- created - timestamp without time zone
 ,0                  -- record_count - bigint
 ,0                  -- update_count - bigint
 ,0                  -- lock_owner - integer
);
ENDTEXT

       oStmt := DacSqlStatement():fromChar(cSQL)
       oStmt:build():execute()

ENDIF // IF TABLE("coba_systems", _oSession_)=.F.
*------------------------------------------------------------------------



FOR i=1 TO LEN( _aListAppTable_ )

   tablename           := _aListAppTable_[i,1]
   schemenametablename := _aListAppTable_[i,2]
   sqlstring           := _aListAppTable_[i,3]

   *** IF TABLE(tablename, _oSession_)=.F. // koristi samo šemu 'public'
   IF TABLE_pg(_she_,tablename) == .F.     // koristi scheme.table string // __pg__tab.prg

      ec_k()
      c__greska(tablename+";ne postoji","TABLA: ")
      lRet:=.F.

   ELSE

      ****************************************
      * // brisanje aktuelnog reda
```

```
      * apptable := "'" + tablename + "'"
      * cSQL := "DELETE FROM ONLY " +tabla +" WHERE table_name = "+apptable+";"
      *  // obrada i izvršenje PGDBE SQL komande
      * oStmt := DacSqlStatement():fromChar(cSQL)
      * oStmt:build():execute()
      ****************************************


// formira se SQL script i/ili SQL string
// za upotrebu u PGDBE SQL komandi
// Upis podataka u sve kolone table: &tabla="public"."alaska-software.isam.tables"
apptable := "'" + tablename + "'"
TEXT INTO cSQL WRAP
INSERT INTO &tabla
(
  table_name
 ,unc_name
 ,updated
 ,created
 ,record_count
 ,update_count
 ,lock_owner
)
VALUES
(
 &apptable
 ,null               -- unc_name - character varying(255)
 ,CURRENT_TIMESTAMP -- updated - timestamp without time zone
 ,CURRENT_TIMESTAMP -- created - timestamp without time zone
 ,0                  -- record_count - bigint
 ,0                  -- update_count - bigint
 ,0                  -- lock_owner - integer
);
ENDTEXT
* ec_k()
* dc_memoedit(cSQL);quit

       // formira se SQL script i/ili SQL string
       // za upotrebu u PGDBE SQL komandi
       // upis podataka u jednu kolonu table
       * SINTAKSA:
       * ovde su važni znaci navoda !!! kod table dvostruki a kod values jednostruki:
       * INSERT INTO "public"."alaska-software.isam.tables" (table_name) VALUES ('kup');
       * apptable := "'" + tablename + "'"                                    // string za
upotrebu
       * cSQL := "INSERT INTO "+ tabla +" (table_name) VALUES ("+apptable+");" // string za
upotrebu
        // obrada i izvršenje PGDBE SQL komande
       oStmt := DacSqlStatement():fromChar(cSQL)
       oStmt:build():execute()
      ****************************************
      * Note:
      * aplikacija sa Coba (skraćenom) ISAM navigacijom:
      * bez indeksa, bez FTS i bez DBFNTX i FOXCDX DBE
      * već samo sa PGDBE, funkcionisaće i sa praznom
      * tablom: alaska-software.isam.tables
       * Notes:
       * application with Coba (abbreviated) ISAM navigation:
       * no index, no FTS and no DBFNTX and FOXCDX DBE
       * but only with PGDBE, it will also work with blank
```

```
       * table: alaska-software.isam.tables


   ENDIF // IF TABLE(

NEXT i

   ****************************************
    // rekonstrukcija formirane prazne i/ili
    // napunjene table:
    cSQL := "VACUUM FULL " + tabla + ";"
    // obrada i izvršenje PGDBE SQL komande
    oStmt := DacSqlStatement():fromChar(cSQL)
    oStmt:build():execute()
   ****************************************

   **************************************** UKUPAN BROJ SVIH REDOVA TABLE
   * SELECT COUNT(*) FROM "public"."alaska-software.isam.tables" ;
   cSQL := "SELECT COUNT(*) FROM " + tabla + ";" // izbroji sve redove table
   oStmt := DacSqlStatement():fromChar(cSQL)
   oStmt:build():query(,@xresult)
   select query
   number_rows_in_table:=count // stop(count,1)
   close query
   // stop( "Broj tabli ove aplikacije upisan u alaska-software.isam.tables = "+var2char(
number_rows_in_table ),1)
   ****************************************


ec_k()


__pg__disconnect()



IF lRet=.T.

 if _lMsg_=.T.
   ec_k()
   c_poruka("SVE TABLE U BAZI PODATAKA SU NA BROJU I"  + chr(13)+;
            "UPISANE SU U DATABAZU U LISTU ISAM TABLI"  + chr(13)+;
            "","OK | BROJ UPISANIH TABLI = "+var2char( number_rows_in_table ))
 endif

ELSE

 if _lMsg_=.T.
   ec_k()
   c_greska("NE POSTOJE SVE TABLE APLIKACIJE U BAZI PODATAKA : "+var2char(number_table_in_app)
+ chr(13)+;
            "POSTOJEĆE SU UPISANE DATABAZU U LISTU ISAM TABLI: "+var2char( number_rows_in_table
)  + chr(13)+;
            "","STOP | BROJ UPISANIH TABLI = "+var2char( number_rows_in_table ))
 endif

ENDIF


RETURN lREt
```

PRILOG: šema rada funkcije __pg__connect_app() koja se poziva na startu svake poslovne aplikacije i vrši kontrolu ispravnosti postgreSQL baze podataka

ATTACHMENT: the working scheme of the __pg__connect_app() function, which is called at the start of each business application and checks the correctness of the PostgreSQL database



14.10.2023

COBA Systems BAST | BUSINESS ACCOUNT SOFTWARE TECHNOLOGY | www.cobasystems.com,
Open Source Project | Alaska Xbase++, eXpress++ | PostgreSQL database server + ISAM technology

Note:

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