CSYSTEMS™

PROGRAMSKI PAKET ZA KNJIGOVODSTVO

COBA Systems

Alaska Xbase++
DBF to PostgreSQL database application

(PART 1)
Alaska Xbase++ PGDBE engine
and DBFUPSIZE technology

01.10.2023

SRB
UVOD

Alaska Xbase++ PGDBE engine je specijalizovana mašina koja aplikaciju urađenu u programskom jeziku Alaska Xbase++ pretvara u:

1. aplikaciju koja koristi **PostgreSQL bazu podataka i SQL table** putem komandi:

```
UNIVERZAL SQL komandi (ugrađene u PGDBE ili ODBCDBE)
PGDBE SQL komandi     (ugrađene PGDBE)
ODBCDBE SQL komandi   (ugrađene u ODBCDBE)
```

2. aplikaciju koja koristi PostgreSQL bazu podataka i SQL table putem putem prevođenja te PostgreSQL baze podataka na **UPSIZE PostgreSQL bazu podataka** posebnim programskim alatom DBFUPSIZE.EXE putem komandi:

```
ISAM DBF komandi (ugrađene u Xbase++ programski jezik i u PGDBE)
UNIVERZAL SQL komandi (ugrađene u PGDBE ili ODBCDBE)
PGDBE SQL komandi     (ugrađene PGDBE)
ODBCDBE SQL komandi   (ugrađene u ODBCDBE)
```

3. aplikaciju koja istovremeno koristi PostgreSQL bazu podataka i DBF bazu podataka, odnosno DBF fajlove i SQL table, putem prevođenja te PostgreSQL baze podataka na **UPSIZE PostgreSQL bazu podataka** posebnim programskim alatom DBFUPSIZE.EXE putem:

```
ISAM DBF komandi (ugrađene u Xbase++ programski jezik i u PGDBE)
UNIVERZAL SQL komandi (ugrađene u PGDBE ili ODBCDBE)
PGDBE SQL komandi     (ugrađene PGDBE)
ODBCDBE SQL komandi   (ugrađene u ODBCDBE)
```

Tačka 1. je budućnost velikih poslovnih SQL Xbase++ aplikacija.

Tačka 2. je radosna i zanimljiva budućnost za programere SQL Xbase++ aplikacija.

Tačka 3. je produžetak života i unapređenje postojećih velikih projekata i investicija u DBF Xbase++ aplikacije, odnosno prvenstveno zaštita postojećih investicija.

---------- **Moja ideja za dalji rad sa Xbase++ aplikacijama** ----------

Imam softver (poslovnu aplikaciju) **FISKALNA KASA ZA RESTORAN** urađenu u Alaska Xbase++ 2.0 i eXpress++ sa DBFNTX i ADSDBE database mašinama. Aplikacija spada u program koji se može koristiti bilo gde u svetu, jer se bavi operativnim evidentiranjem prodaje u restoranima i izdavanjem računa gostima restorana za prodate proiozvode. Iz tog razloga sam ovu aplikaciju izabrao za ovaj prikaz.

Aplikacija ima relativno proste poslovne procese i poslovnu logiku, ali ima složenu komunikaciju sa bazom podataka i posebno ima jako komplikovan korisnički interfejs.

Dobre strane:

Pod ADS serverom ova aplikacija radi odlično (brzo i tačno). Interfejs urađen u Xbase++ i u eXpress++ je jako dobar. Radi u mrežnom okruženju. Radi preko Windows RDP servisa, što znači da se može koristiti i preko interneta i Windows servera 2016 pa na dalje.

Loše strane:

ADS Advantage Database Server više nije podržan u daljem razvoju od strane SAP. Sadašnja aplikacija sa DBF/DBT/FPT i NTX/CDX fajlovima i bez ADS servera i sa ADS serverom nije dobro rešenje kada se na nju treba povezati preko mobilnih uređaja koji rade sa ANDROID OS (konobari sakupljaju porudžbine preko mobilnih uređaja). Takođe nije dobro rešenje kada iz aplikacije treba slati SMS poruke izabranim gostima u postojećem lojality sistemu rada. Upotreba DBF baze podataka za neko buduće prevođenje desktop aplikacije na web aplikaciju (što je trend) uvek je problematična i bolje je u tom slučaju imati PostgreSQL bazu podataka. DBF baza podataka još uvek ima feler kada je u pitanju UTF-8. Danas se većina stvari a posebno kod REST web servisa prenosi putem XML i JSON fajlova koji zahtevaju UTF-8. PostgreSQL server po defaultu radi sa UTF-8, a u Alaska Xbase++ PGDBE to je odlično rešeno – tako da programer ne treba da brine o tome (sa ovim nedostatkom DBF baze podataka imaju problem svi koji nisu sa engleskog govornog područja i imaju svoje posebne znakove u svom character set-u).

Odluka:

Zbog toga sam odlučio da ovu aplikaciju prebacim na UPSIZE PostgreSQL bazu podataka putem najnovije Alaska Xbase++ DBFUPSIZE i PGDBE tehnologije. Pri tome sam postavio sledeće ciljeve:

1. Koristiti PostgreSQL Server koji je dovoljno mali da može da se lako distribuira uz poslovnu aplikaciju.

2. Koristiti PostgreSQL server koji može da radi i sa 32 bitnim i sa 64 bitnim Windowsom i koji se od strane korisnika aplikacije može da jednostavno i brzo instalira.

3. Na osnovu tačke 1. i tačke 2. usvojio sam PostgreSQL server verzija 9.4.4 za razvoj ove aplikacije. To ne znači da ista aplikacija neće moći da radi sa novijim verzijama PostgreSQL servera. Dok ovo pišem aktuelna verzija je 16.

4. Odustati od istovremenog rada sa DBFNTX, FOXCDX i PGDBE, odnosno od istovremenog rada sa DBF fajlovima i njihovim upsize klonovima PostgreSQL tabelama. Raditi samo sa PGDBE i PostgreSQL tabelama i maksimalno koristiti postojeće ISAM komande i funkcije, koje postoje u aplikaciji, za komunikaciju sa PostgreSQL tabelama.

5. Odustati od emulacije NTX i CDX indeksnih fajlova u UPSIZE PostgreSQL tablama. Pokazalo se da se indeksi mogu dobro praviti na PostgreSQL tablama u toku rada PGDBE aplikacije a mogu se zameniti i drugim rešenjima iz moćnog pastrough SQL jezika ugrađenog u PGDBE. Isto važi i za ISAM komande SET FILTER TO, DELETE-PACK, APPEND FROM, COPY TO, TOTAL TO ...

6. Sve jako bitne komande i funkcije za rad sa PostgreSQL bazom podataka napisati u passtrough SQL koji je ugrađen u PGDBE. Odustati od pokušaja da se ove komande i funkcije simuliraju preko ISAM mašine (to još uvek nije dovoljno dugo testirano i mogući su raznorazni problemi). Ove komande i funkcije napisati kao posebne korisničke funkcije, ili kao snippet code, tako da se mogu primeniti u bilo kojoj sledećoj aplikaciji.

7. U toku rada sakupljati i evidentirati sve nastale probleme i sve moguće probleme koji mogu nastati u ekspoloataciji aplikacije.

8. Objaviti na eXpress++ forumu ovaj projekat kao zahvalnost majstoru Roger Dannay i mnogim drugim učesnicima ovog foruma za dobre stvari koje sam od njih do sada dobio.

ENG

INTRODUCTION

Alaska Xbase++ PGDBE engine is a specialized engine that converts an application made in the Alaska Xbase++ programming language into:

1. an application that uses a **PostgreSQL database and SQL tables** via commands:

    UNIVERSAL SQL commands (embedded in PGDBE or ODBCDBE)
    PGDBE SQL Commands (embedded PGDBE)
    ODBCDBE SQL commands (built into ODBCDBE)

2. an application that uses a PostgreSQL database and SQL tables by translating that PostgreSQL database to a **UPSIZE PostgreSQL database** with a special program tool DBFUPSIZE.EXE via commands:

    ISAM DBF commands (built into Xbase++ programming language and PGDBE)
    UNIVERSAL SQL commands (embedded in PGDBE or ODBCDBE)
    PGDBE SQL Commands (embedded PGDBE)
    ODBCDBE SQL commands (built into ODBCDBE)

3. an application that simultaneously uses a PostgreSQL database and a DBF database, i.e. DBF files and SQL tables, by translating that PostgreSQL database into a **UPSIZE PostgreSQL database** with a special program tool DBFUPSIZE.EXE via commands:

    ISAM DBF commands (built into Xbase++ programming language and PGDBE)
    UNIVERSAL SQL commands (embedded in PGDBE or ODBCDBE)
    PGDBE SQL Commands (embedded PGDBE)
    ODBCDBE SQL commands (built into ODBCDBE)

Point 1. is the future of large business SQL Xbase++ applications.

Point 2. is a joyful and interesting future for developers of SQL Xbase++ applications.

Point 3 is life extension and improvement of existing large projects and investments in DBF Xbase++ applications, i.e. primarily protection of existing investments.

---------- My idea for further work with Xbase++ applications ----------

I have software (business application) FISCAL CASH FOR RESTAURANT done in Alaska Xbase++ 2.0 and eXpress++ with DBFNTX and ADSDBE database engines. The application belongs to a program that can be used anywhere in the world, because it deals with operational recording of sales in restaurants and issuing invoices to restaurant guests for the products sold. For this reason, I chose this application for this display.
The application has relatively simple business processes and business logic, but it has complex communication with the database and, in particular, it has a very complicated user interface.

Good side:

Under ADS server this application works great (fast and accurate). The interface made in Xbase++ and eXpress++ is very good. It works in a network environment. It works through the Windows RDP service, which means that it can be used over the Internet and Windows Server 2016 and beyond.

Bad side:

ADS Advantage Database Server is no longer supported in further development by SAP. The current application with DBF/DBT/FPT and NTX/CDX files and without ADS server and with ADS server is not a good solution when you need to connect to it via mobile devices working with ANDROID OS (waiters collect orders via mobile devices). It is also not a good solution when you need to send SMS messages from the application to selected guests in the existing loyalty system. Using a DBF database for some future translation of a desktop application to a web application (which is a trend) is always problematic and it is better to have a PostgreSQL database in that case. The DBF database still has a faller when it comes to UTF-8. Today, most things, especially REST web services, are transmitted via XML and JSON files that require UTF-8. The PostgreSQL server works with UTF-8 by default, and in Alaska Xbase++ PGDBE this is perfectly solved - so the programmer does not need to worry about it (this lack of DBF database is a problem for everyone who is not from the English speaking area and has their own special characters in to your character set).

Decision:

Therefore, I decided to migrate this application to a UPSIZE PostgreSQL database using the latest Alaska Xbase++ DBFUPSIZE and PGDBE technology. In doing so, I set the following goals:

1. Use a PostgreSQL Server that is small enough to be easily distributed with a business application.

2. Use a PostgreSQL server that can work with both 32-bit and 64-bit Windows and which can be easily and quickly installed by the application user.

3. Based on point 1 and point 2, I adopted PostgreSQL server version 9.4.4 to develop this application. This does not mean that the same application will not be able to work with newer versions of the PostgreSQL server. As I write this, the current version is 16.

4. Give up simultaneous work with DBFNTX, FOXCDX and PGDBE, that is, simultaneous work with DBF files and their upsize clones of PostgreSQL tables. Work only with PGDBE and PostgreSQL tables and maximally use existing ISAM commands and functions, which exist in the application, to communicate with PostgreSQL tables.

5. Give up emulation of NTX and CDX index files in UPSIZE PostgreSQL tables. It has been shown that indexes can be well created on PostgreSQL tables during the operation of the PGDBE application, and they can be replaced by other solutions from the powerful pastrough SQL language built into PGDBE. The same applies to the ISAM commands SET FILTER TO, DELETE-PACK, APPEND FROM, COPY TO, TOTAL TO ...

6. Write all very important commands and functions for working with the PostgreSQL database in passthrough SQL, which is built into PGDBE. Give up trying to simulate these commands and functions through the ISAM machine (it has not been tested long enough and various problems are possible). Write these commands and functions as special user functions, or as snippet code, so that they can be applied in any subsequent application.

7. In the course of work, collect and record all the problems that have arisen and all possible problems that may arise in the operation of the application.

8. Publish this project on the eXpress++ forum as a thank you to master Roger Dannay and many other participants of this forum for the good things I have received from them so far.

20.10.2023, dipl.ing.Slobodan Stanojević Programmer at company COBA Systems, Serbia

SL.1

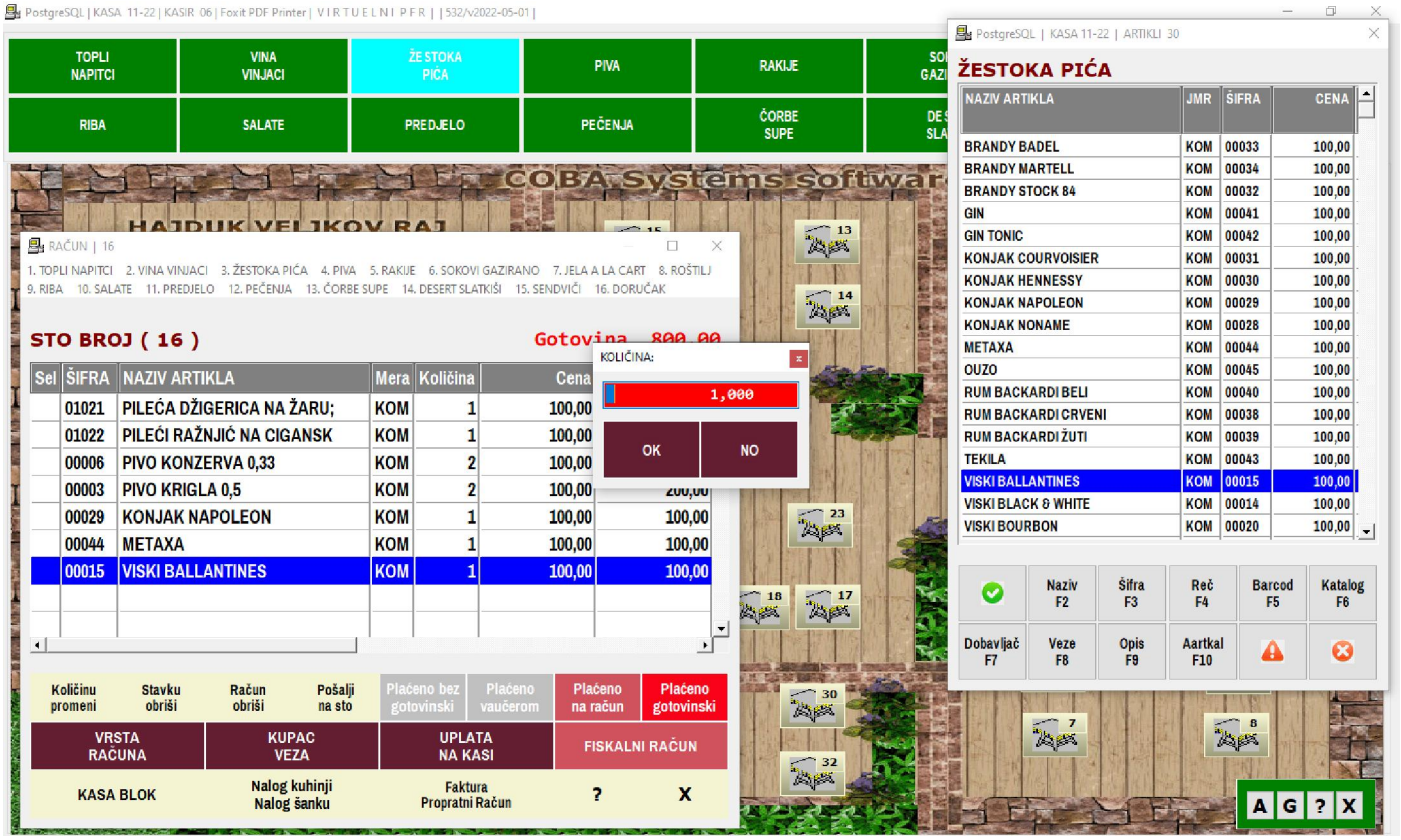
IZGLED I SADRŽAJ APLIKACIJE FISKALNA REGISTAR KASA ZA RESTORAN
APPEARANCE AND CONTENTS OF THE FISCAL CASHIER APPLICATION FOR THE RESTAURANT

SL.2

PRODAJNI OBJEKAT – OBJEKAT RESTORANA SA RASPOREDOM STOLOVA
SALES FACILITY – RESTAURANT FACILITY WITH TABLE ARRANGEMENT

SL.3

Preuzimanje artikla iz šifarnika artikala u račun za kupca
Downloading the item from the item codebook to the customer's account



DBF BAZA PODATAKA
DBF DATABASE

Ova apliakcija koristi sledeću bazu podataka:
This application uses the following database:

kup.dbf	registar kupaca customer register
kroba_11.dbf/dbt	registar proizvoda product registry
kasa_22.dbf	arhiva fiskalnih računa archive of fiscal accounts
kaso_22.dbf	arhiva nefiskalnih računa archive of non-fiscal accounts
kbon_22.dbf	trenutni aktivni račun u kasi

current active account in the cash register

kblok_22.dbf naslovni deo računa sa podacima o kupcu
the title part of the invoice with the customer's information

x1.dbf trenutni račun na restoranskom stolu broj 1
... the current account on the number 1 restaurant table
...
...
x100.dbf trenutni račun na restoranskom stolu broj 100
the current account on the number 100 restaurant table

POSTGRESQL BAZA PODATAKA
POSTGRESQL DATABASE

Ova apliakcija koristi sledeću bazu podataka:
This application uses the following database:

```
Server       = 'Localhost'
Database     = 'kasa'
Scheme       = 'public'
uid          = 'postgres'
pwd          = 'coba#1949'
```

Tables:

kup registar kupaca
customer register

kroba_11 registar proizvoda
product registry

kasa_22 arhiva fiskalnih računa
archive of fiscal accounts

kaso_22 arhiva nefiskalnih računa
archive of non-fiscal accounts

kbon_22 trenutni aktivni račun u kasi
current active account in the cash register

kblok_22 naslovni deo računa sa podacima o kupcu
the title part of the invoice with the customer's information

x1 trenutni račun na restoranskom stolu broj 1
... the current account on the number 1 restaurant table
...
...
x100 trenutni račun na restoranskom stolu broj 100
the current account on the number 100 restaurant table

Legenda:

Kod fajla kroba_11.dbf broj 11 je šifra restorana u firmi. Firma može imati više restorana (prodajni objekti) od 01 do 99.

Kod fajla kasa_22.dbf broj 22 je šifra kase u firmi. Firma može imati više kasa od 01 do 99.

kod fajla x100.dbf broj 100 je broj restoranskog stola u objektu restorana. Objekat restorana može imati najviše 100 stolova (u ovoj verziji programa).

Podrazumeva se da je isto i kod PostgreSQL tabli

Legend:

In the file kroba_11.dbf, number 11 is the code of the restaurant in the company. A company can have several restaurants (sales facilities) from 01 to 99.

In the file kasa_22.dbf, number 22 is the code of the cash register in the company. A company can have several cash registers from 01 to 99.

in the file x100.dbf the number 100 is the number of the restaurant table in the restaurant facility. The restaurant facility can have a maximum of 100 tables (in this version of the program).

It goes without saying that the same is true for PostgreSQL tables

---

STRUKTURA DBF FAJLOVA I UPSIZE SQL TABLI

STRUCTURE OF DBF FILES AND UPSIZE SQL TABLES

```
-- X B A S E  F I L E -- kup.dbf
--
--         1 KUDS_      C    4 0 --
--         2 KUDO_      C    1 0 --
--         3 VEZA_      C   13 0 --
--         4 BARC_      C   13 0 --
--         5 KUDN_      C   40 0 --
--         6 KUDU_      C   40 0 --
--         7 ADRE_      C   30 0 --
--         8 POST_      C    5 0 --
--         9 MEST_      C   24 0 --
--        10 REPU_      C   30 0 --
--        11 LICE_      C   30 0 --
--        12 TFON_      C   30 0 --
--        13 TFAX_      C   30 0 --
--        14 TLEX_      C   30 0 --
--        15 ZIRO_      C   30 0 --
--        16 MBR_       C   30 0 --
--        17 PIB_       C    9 0 --
--        18 SDEL_      C    5 0 --
--        19 IBAN_      C   30 0 --
--        20 ACOD_      C   30 0 --
```

```
--       21 MAIL_      C   40 0 --
--       22 WEBS_      C   40 0 --
--       23 DATO_      D    8 0 --
--       24 DATR_      D    8 0 --
--       25 ROK_       N    8 0 --
--       26 RABAT_     N    6 2 --
--       27 GRUPA_     C    4 0 --
--       28 STATU_     C    4 0 --
--       29 NOTES_     C   40 0 --
--       30 KONT_      C   10 0 --
--       31 FLAG_      C    1 0 --
--       32 ZNAK_      C    1 0 --


  -- P O S T G R E S Q L --> kup
  DROP TABLE IF EXISTS kup;

  CREATE TABLE kup
  (
  KUDS_               character(4)        ,
  KUDO_               character(1)        ,
  VEZA_               character(13)       ,
  BARC_               character(13)       ,
  KUDN_               character(40)       ,
  KUDU_               character(40)       ,
  ADRE_               character(30)       ,
  POST_               character(5)        ,
  MEST_               character(24)       ,
  REPU_               character(30)       ,
  LICE_               character(30)       ,
  TFON_               character(30)       ,
  TFAX_               character(30)       ,
  TLEX_               character(30)       ,
  ZIRO_               character(30)       ,
  MBR_                character(30)       ,
  PIB_                character(9)        ,
  SDEL_               character(5)        ,
  IBAN_               character(30)       ,
  ACOD_               character(30)       ,
  MAIL_               character(40)       ,
  WEBS_               character(40)       ,
  DATO_               date                ,
  DATR_               date                ,
  ROK_                integer             ,
  RABAT_              numeric(6,2)        ,
  GRUPA_              character(4)        ,
  STATU_              character(4)        ,
  NOTES_              character(40)       ,
  KONT_               character(10)       ,
  FLAG_               character(1)        ,
  ZNAK_               character(1)        ,    -- UPSIZE dodatak skraćeni - start
     __deleted boolean NOT NULL DEFAULT false, -- UPSIZE supplement shortened - start
     __record serial NOT NULL,
     __rowversion integer NOT NULL DEFAULT 1,
     __keyversion integer NOT NULL DEFAULT 0,
```

```
    __lock_owner integer NOT NULL DEFAULT 0,
    CONSTRAINT kup_pkey PRIMARY KEY (__record) -- UPSIZE supplement shortened - end
)                                              -- UPSIZE dodatak skraćeni - end
WITH (OIDS=FALSE,autovacuum_enabled=true);



-- X B A S E  F I L E -- kroba_11.dbf
--
--        1 GRUP_       C    2 0 --
--        2 ROBS_       C    5 0 --
--        3 ROBN_       C   25 0 --
--        4 ROBK_       C   25 0 --
--        5 MERA_       C    3 0 --
--        6 TARI_       C    1 0 --
--        7 PPUP_       N    5 1 --
--        8 PPAP_       N   10 2 --
--        9 CENA_FAK_   N   13 2 --
--       10 CENA_NAB_   N   13 2 --
--       11 CENA_VPR_   N   13 2 --
--       12 CENA_MPR_   N   13 2 --
--       13 CENA_DEV_   N   13 2 --
--       14 ZALI_       N   15 3 --
--       15 DAT0_       D    8 0 --
--       16 MAGS_       C    2 0 --
--       17 ZNAK_       C    1 0 --
--       18 FLAG_       C    1 0 --
--       19 STAT_       C    1 0 --
--       20 CENA_PRO_   N   13 3 --
--       21 OSNO_AKC_   N   13 3 --
--       22 ROK_        N    3 0 --
--       23 DOBS_       C    4 0 --
--       24 DOBK_       C   25 0 --
--       25 VEZA_       C   15 0 --
--       26 NAME_       C  200 0 --
--       27 OPIS_       M   10 0 --


  -- P O S T G R E S Q L --> kroba_11

  DROP TABLE IF EXISTS kroba_11;

  CREATE TABLE kroba_11
  (
  GRUP_               character(2)       ,
  ROBS_               character(5)       ,
  ROBN_               character(25)      ,
  ROBK_               character(25)      ,
  MERA_               character(3)       ,
  TARI_               character(1)       ,
  PPUP_               numeric(5,1)       ,
  PPAP_               numeric(10,2)      ,
  CENA_FAK_           numeric(13,2)      ,
  CENA_NAB_           numeric(13,2)      ,
```

```
  CENA_VPR_           numeric(13,2)       ,
  CENA_MPR_           numeric(13,2)       ,
  CENA_DEV_           numeric(13,2)       ,
  ZALI_               numeric(15,3)       ,
  DAT0_               date                ,
  MAGS_               character(2)        ,
  ZNAK_               character(1)        ,
  FLAG_               character(1)        ,
  STAT_               character(1)        ,
  CENA_PRO_           numeric(13,3)       ,
  OSNO_AKC_           numeric(13,3)       ,
  ROK_                integer             ,
  DOBS_               character(4)        ,
  DOBK_               character(25)       ,
  VEZA_               character(15)       ,
  NAME_               character(200)      ,
  OPIS_               text                ,
     __deleted boolean NOT NULL DEFAULT false,
     __record serial NOT NULL,
     __rowversion integer NOT NULL DEFAULT 1,
     __keyversion integer NOT NULL DEFAULT 0,
     __lock_owner integer NOT NULL DEFAULT 0,
     CONSTRAINT kroba_11_pkey PRIMARY KEY (__record)
  )
  WITH (OIDS=FALSE,autovacuum_enabled=true);


--
-- X B A S E  F I L E – kasa_22.dbf, kaso_22.dbf, kbon_22.dbf, x1.dbf... x100.dbf
--
--         1 BRJ0_      C   12 0 --
--         2 DAT0_      D    8 0 --
--         3 VRE0_      C   10 0 --
--         4 BRJ1_      C   30 0 --
--         5 DAT1_      C   30 0 --
--         6 BRJ2_      C   30 0 --
--         7 DAT2_      C   30 0 --
--         8 VAL1_      C   30 0 --
--         9 DOBS_      C    4 0 --
--        10 DOBO_      C    1 0 --
--        11 MAGS_      C    2 0 --
--        12 OPST_      C    2 0 --
--        13 KUP1_      C   30 0 --
--        14 KUP2_      C   30 0 --
--        15 RAC1_      C    2 0 --
--        16 RAC2_      C   30 0 --
--        17 KSIR_      C   13 0 --
--        18 ESIR_      C   30 0 --
--        19 ZNAK_      C    1 0 --
--        20 STAT_      C    1 0 --
--        21 GRUP_      C    2 0 --
--        22 ROBS_      C    5 0 --
--        23 ROBK_      C   30 0 --
--        24 DOBK_      C   30 0 --
--        25 ROBN_      C   50 0 --
```

```
--       26 MERA_       C    3 0 --
--       27 KOLI_       N   16 3 --
--       28 CENA_FAK_   N   16 2 --
--       29 CENA_MPR_   N   16 2 --
--       30 VRED_MPR_   N   16 2 --
--       31 TARI_       C    1 0 --
--       32 PPUP_       N    5 1 --
--       33 PPOD_       N   16 2 --
--       34 CENA_VPR_   N   16 2 --
--       35 VRED_VPR_   N   16 2 --
--       36 RABP_       N    6 2 --
--       37 RABD_       N   16 2 --
--       38 FLAG_       C    1 0 --
--       39 FLAG_IME_   C   30 0 --
--       40 UPLATA_     N   16 2 --
--       41 RACUN_      N   16 2 --
--       42 KUSUR_      N   16 2 --
--       43 GOT_        N   16 2 --
--       44 CEK_        N   16 2 --
--       45 KAR_        N   16 2 --
--       46 NAL_        N   16 2 --
--       47 INS_        N   16 2 --
--       48 VAU_        N   16 2 --
--       49 BOT_        N   16 2 --
--       50 AVA_        N   16 2 --
--       51 AVP_        N   16 2 --
--       52 COBA_       N    7 0 --
--       53 COK_        L    1 0 --
--       54 UUID_       C   36 0 --

  --
  -- P O S T G R E S Q L --> kasa_22, kaso_22, kbon_22, x1... x100

  DROP TABLE IF EXISTS kasa_22;  (kaso_22, kbon_22, x1... x100)

  CREATE TABLE kasa_22  (kaso_22, kbon_22, x1... x100)
  (
  BRJ0_                 character(12)       ,
  DAT0_                 date                ,
  VRE0_                 character(10)       ,
  BRJ1_                 character(30)       ,
  DAT1_                 character(30)       ,
  BRJ2_                 character(30)       ,
  DAT2_                 character(30)       ,
  VAL1_                 character(30)       ,
  DOBS_                 character(4)        ,
  DOBO_                 character(1)        ,
  MAGS_                 character(2)        ,
  OPST_                 character(2)        ,
  KUP1_                 character(30)       ,
  KUP2_                 character(30)       ,
  RAC1_                 character(2)        ,
  RAC2_                 character(30)       ,
  KSIR_                 character(13)       ,
  ESIR_                 character(30)       ,
```

```
  ZNAK_                character(1)        ,
  STAT_                character(1)        ,
  GRUP_                character(2)        ,
  ROBS_                character(5)        ,
  ROBK_                character(30)       ,
  DOBK_                character(30)       ,
  ROBN_                character(50)       ,
  MERA_                character(3)        ,
  KOLI_                numeric(16,3)       ,
  CENA_FAK_            numeric(16,2)       ,
  CENA_MPR_            numeric(16,2)       ,
  VRED_MPR_            numeric(16,2)       ,
  TARI_                character(1)        ,
  PPUP_                numeric(5,1)        ,
  PPOD_                numeric(16,2)       ,
  CENA_VPR_            numeric(16,2)       ,
  VRED_VPR_            numeric(16,2)       ,
  RABP_                numeric(6,2)        ,
  RABD_                numeric(16,2)       ,
  FLAG_                character(1)        ,
  FLAG_IME_            character(30)       ,
  UPLATA_              numeric(16,2)       ,
  RACUN_               numeric(16,2)       ,
  KUSUR_               numeric(16,2)       ,
  GOT_                 numeric(16,2)       ,
  CEK_                 numeric(16,2)       ,
  KAR_                 numeric(16,2)       ,
  NAL_                 numeric(16,2)       ,
  INS_                 numeric(16,2)       ,
  VAU_                 numeric(16,2)       ,
  BOT_                 numeric(16,2)       ,
  AVA_                 numeric(16,2)       ,
  AVP_                 numeric(16,2)       ,
  COBA_                integer             ,
  COK_                 boolean             ,
  UUID_                character(36)       ,
     __deleted boolean NOT NULL DEFAULT false,
     __record serial NOT NULL,
     __rowversion integer NOT NULL DEFAULT 1,
     __keyversion integer NOT NULL DEFAULT 0,
     __lock_owner integer NOT NULL DEFAULT 0,
     CONSTRAINT kasa_22_pkey PRIMARY KEY (__record)
)
  WITH (OIDS=FALSE,autovacuum_enabled=true);
```

(kaso_22_pkey, kbon_22_pkey, x1_pkey, x2_pkey... x100_pkey)

Fajl **kblok_22.dbf** odnosno tabela **kblok_22** ne učestvuje u ovoj verziji programa.
The kblok_22.dbf file, i.e. the kblok_22 table, does not participate in this version of the program.

KAKO IZ POSTOJEĆEG DBF/DBT FAJLA roba_11.dbf NAPRAVITI PostgreSQL TABLU roba_11
HOW TO CREATE A PostgreSQL TABLE roba_11 FROM AN EXISTING DBF/DBT FILE roba_11.dbf

Najjednostavniji način je upotreba Alaska Xbase++ programa DBFUPSIZE.EXE

UPSIZE je dodavanje specijalnih polja (kolona) u SQL tablu roba_11, a kojih nema u ISAM fajlu roba_11.dbf . Ove dodatne UPSIZE KOLONE u tabli roba_11, njihovi indeksi i trigeri, su deo **ISAM mašine za prevođenje** ISAM komandi i funkcija u SQL komande i funkcije.

Uz ove UPSIZE KOLONE, indekse i trigere u SQL tabli roba_11, koja se formira u bilo kojoj databazi, ali samo u šemi public, u šemi public se formiraju tri posebne sistemske ISAM TABLE i tri njma odgovarajuće sekvence kao i sekvenca za tablu roba_11, a formiraju se i dve triger funkcije plus triger funkcija za FullTextSearch triger ako u roba_11 tabli postoji kolona za Full Text Search. Ovo je takođe deo **ISAM mašine za prevođenje** ISAM komandi i funkcija u SQL komande i funkcije.

Uz sve ovo, u službenoj default databazi: postgres, koja se automatski formira instalacijom PostgreSQL servera, program DBFUPSIZE.EXE će u šemi: public formirati tri posebne sistemske ISAM TABLE i tri njima odgovarajuće sekvence identične kao i u databazi i njenoj šemi public u kojoj je kreirana tabla roba_11. Ovo je takođe deo **ISAM mašine za prevođenje** ISAM komandi i funkcija u SQL komande i funkcije.

Sve ovo je kompletna **ISAM MAŠINA za prevođenje** ISAM komandi i funkcija u SQL komande i funkcije, koja kao rezultat svog postojanja i rada daje mogućnost Xbase++ aplikaciji da preko ISAM komandi i funkcija komunicira sa PgSQL bazom podataka i PgSQL tablama koje su kreirane kao dograđeni klonovi DBF/DBT/NTX databaze sa ISAM fajlovima.

The simplest way is to use the Alaska Xbase++ program DBFUPSIZE.EXE

UPSIZE is the addition of special fields (columns) in the SQL table roba_11, which are not in the ISAM file roba_11.dbf. These additional UPSIZE COLUMNS in table roba_11, their indexes and triggers, are part of the **ISAM engine for translating** ISAM commands and functions into SQL commands and functions.

Along with these UPSIZE COLUMNS, indexes and triggers in the SQL table roba_11, which is formed in any database, but only in the schema public, three special system ISAM TABLES and three corresponding sequences are formed in the schema public, as well as the sequence for the table roba_11, and two trigger functions are also formed plus a trigger function for FullTextSearch trigger if there is a column for Full Text Search in the roba_11 table. This is also part of the **ISAM engine for translating** ISAM commands and functions into SQL commands and functions.

In addition to all this, in the official default database: postgres, which is automatically formed by installing the PostgreSQL server, the program DBFUPSIZE.EXE will form three special system ISAM TABLES in the scheme: public and three corresponding sequences identical to the database and its scheme in which public created board goods_11. This is also part of the **ISAM engine for translating** ISAM commands and functions into SQL commands and functions.

All this is a complete **ISAM MACHINE for translating ISAM commands and functions into SQL commands and functions**, which as a result of its existence and operation gives the Xbase++ application the ability to communicate via ISAM commands and functions with the PgSQL database and PgSQL tables that are created as built-in DBF clones /DBT/NTX databases with ISAM files.

Procedure:

1. Mora se instalirati PostgreSQL server kao:
1. PostgreSQL server must be installed as:

```
server      = 'Locahost'
database    = 'kasa' ('postgres' ili bilo koja druga - 'postgres' or any other)
scheme      = 'public' (mora ova šema - must this scheme)
uid         = 'postgres'
pwd         = 'coba#1949' (ili bilo koji drugi - or any other)
```

2. U isti folder stavite
2. Put in the same folder

fajl:
roba_11.dbf
i program:
DBFUPSIZE.EXE
i XML fajl:
roba_11.upsize

3. Fajl *.upsize
3. File *.upsize

Alaska Xbase++ je isporučila primer za XML fajl **northwind-customers.upsize**
Alaska Xbase++ has suplied an example XML file **northwind-customers.upsize**

\Documents\Xbase++\source\samples\sql\xbpbrowse

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<config>
<database_engines>
  <dbebuild name="FOXCDX">
     <storage name="FOXDBE"/>
     <order   name="CDXDBE"/>
  </dbebuild>
  <dbeload name="pgdbe"/>
</database_engines>
<connection name="test"
            dbe="pgdbe"
            srv="localhost"
            uid="postgres"
            pwd="postgres"
            database="xppsamples"/>
<table name = "customers"
       dbe  = "foxcdx"
       dbf  = "..\..\data\northwind\dbf\customers.dbf">
</table>
<upsize table="customers" connection="test" mode="isam"/>
</config>
```

Alaska Xbase++ je isporučila primer za XML fajl **mdidemo.upsize**
Alaska Xbase++ has supplied an example XML file **mdidemo.upsize**

\Documents\Xbase++\source\samples\apps\mdidemo

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<config>
<database_engines>
  <dbebuild name="DBFNTX">
     <storage name="DBFDBE"/>
     <order   name="NTXDBE"/>
  </dbebuild>
  <dbebuild name="DBFCDX">
     <storage name="DBFDBE"/>
     <order   name="CDXDBE"/>
  </dbebuild>
  <dbebuild name="FOXCDX">
     <storage name="FOXDBE"/>
     <order   name="CDXDBE"/>
  </dbebuild>
  <dbeload name="pgdbe"/>
</database_engines>

<connection name="test"
            dbe="pgdbe"
            srv="localhost"
            uid="postgres"
            pwd="postgres"
            database="mdidemo"/>

<!-- data\customer files -->
<table name = "customer"
       dbe  = "dbfntx"
       dbf  = "..\..\data\misc\customer.dbf">
       <order>..\..\data\misc\custa.ntx</order>
       <order>..\..\data\misc\custb.ntx</order>
</table>
<table name = "parts"
       dbe  = "foxcdx"
       dbf  = "..\..\data\misc\parts.dbf">
       <order>..\..\data\misc\parts.cdx</order>
</table>
<upsize table="customer" connection="test" mode="isam">
  <deferred>
    <!-- add a full text search column, based on given fields of the isam table -->
    <add column="fts_col" type="fts" binding="lastname,firstname,city,notes"
language="en"/>
  </deferred>
</upsize>
<upsize table="parts" connection="test" mode="isam"/>
</config>
```

Za potrebe testiranja rada programa DBFUPSIZE.EXE sa kroba_11.dbf napravio sam sledeći upsize fajl:kroba_11.upsize
za transformaciju dbf baze podataka:
For the purposes of testing the operation of the DBFUPSIZE.EXE program from kroba_11.dbf, I created the following upsize file: kroba_11.upsize
to transform the dbf database:

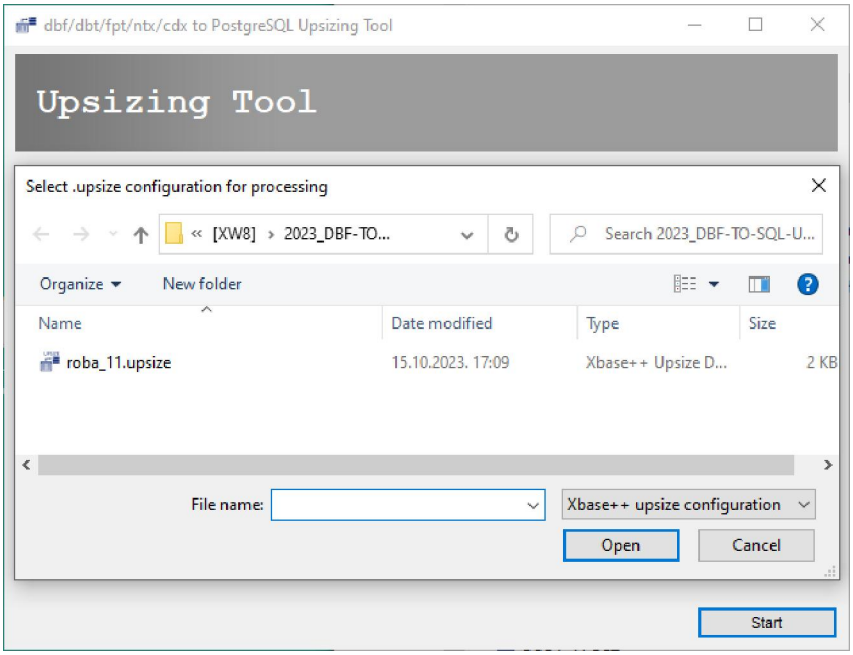
```
      roba_11.dbf    // DataBaseEngine DBFNTX
      roba_11.dbt    // content memo field OPIS_
      index_code.ntx // INDEX ON ROBS_ TO index_code.ntx
      index_name.ntx // INDEX ON ROBN_ TO index_name.ntx
      index_gtin.ntx // INDEX ON ROBK_ TO index_gtin.ntx
```

in PostgreSQL table

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<config>
<database_engines>
  <dbebuild name="DBFNTX">
     <storage name="DBFDBE"/>
     <order   name="NTXDBE"/>
  </dbebuild>
  <dbebuild name="DBFCDX">
     <storage name="DBFDBE"/>
     <order   name="CDXDBE"/>
  </dbebuild>
  <dbebuild name="FOXCDX">
     <storage name="FOXDBE"/>
     <order   name="CDXDBE"/>
  </dbebuild>
  <dbeload name="pgdbe"/>
</database_engines>
<connection name="test"
            dbe="pgdbe"
            srv="localhost"
            uid="postgres"
            pwd="coba#1949"
            database="test"/>
<!-- data\customer files -->
<table name = "roba_11"
       dbe  = "dbfntx"
       dbf  = "roba_11.dbf">
       <order>index_name.ntx</order>
       <order>index_code.ntx</order>
       <order>index_gtin.ntx</order>
</table>
<upsize table="roba_11" connection="test" mode="isam">
  <deferred>
    <!-- add a full text search column, based on given fields of the isam table -->
    <add column="fts_col" type="fts" binding="robn_" language="en"/>
  </deferred>
</upsize>
</config>
```

4. Formiranje SQL table iz programa DBFUPSIZE.EXE
4. Forming the SQL table from the program DBFUPSIZE.EXE

SL.4
PROGRAM DBFUPSIZE.EXE



Startuje se program DBFUPSIZE.EXE i učita se fajl roba_11.upsize u taj program. Program DBFUPSIZE.EXE izvrši automatsko formiranje PostgreSQL baze podataka 'test' i table roba_11 u bazi podataka 'test' u šemi 'public'

The program DBFUPSIZE.EXE is started and the file roba_11.upsize is loaded into that program. The program DBFUPSIZE.EXE automatically creates the PostgreSQL database 'test' and table roba_11 in the database 'test' in the schema 'public'

izmene koje DBFUPSIZE.EXE unosi u database = 'postgres' i u njenu scheme = 'public'

Vidi se da je program DBFUPSIZE.EXE u default-sistemsku bazu podataka PostgreSQL servera **database = 'postgres'** i u njenu default šemu **scheme = 'public'** upisao 3 table i 3 skvence, vidi sliku SL.5

Changes that DBFUPSIZE.EXE makes to database = 'postgres' and to its schema = 'public'

It can be seen that the program DBFUPSIZE.EXE entered 3 tables and 3 sequences into the default system database of the PostgreSQL server **database = 'postgres'** and into its default scheme **scheme = 'public'**, see Figure SL.5

```
-- Table: "alaska-software.isam.orders"
-- Table: "alaska-software.isam.tables"
-- Table: "alaska-software.system.connections"
-- Sequence: "alaska-software.isam.orders_order_id_seq"
-- Sequence: "alaska-software.isam.tables_table_id_seq"
-- Sequence: "alaska-software.system.connections_connection_id_seq"
```

Napomena:
Ove tri table i tri sekvence automatski se kreiraju u bazi podataka 'postgree' čim Alaska Xbase++ aplikacija uspostavi konekciju sa bazom podataka 'test' ili bilo kojom drugom, a to može da bude i default baza podataka 'postgres', preko PGDBE engine-a (PGDBE.DLL)

Note:
These three tables and three sequences are automatically created in the database 'postgree' as soon as the Alaska Xbase++ application establishes a connection with the database 'test' or any other, which can be the default database 'postgres', through the PGDBE engine (PGDBE.DLL)

izmene koje DBFUPSIZE.EXE unosi u database = 'test' i u njenu scheme = 'public'

Vidi se da je program DBFUPSIZE.EXE u zadatu bazu podataka **database = 'test'** i u njenu šemu **scheme = 'public'** upisao 4 table i 4 skvence, vidi sliku SL.5

Changes that DBFUPSIZE.EXE makes to database = 'test' and to its schema = 'public'

It can be seen that the program DBFUPSIZE.EXE entered 4 tables and 4 sequences into the given database **database = 'test'** and into its scheme **scheme = 'public'**, see Figure SL.5

```
-- Table: "alaska-software.isam.orders"
-- Table: "alaska-software.isam.tables"
-- Table: "alaska-software.system.connections"
-- Table: roba_11

-- Sequence: "alaska-software.isam.orders_order_id_seq"
-- Sequence: "alaska-software.isam.tables_table_id_seq"
-- Sequence: "alaska-software.system.connections_connection_id_seq"
-- Sequence: roba_11___record_seq
```

U tablu alaska-software.isam.orders upisano je je:
In the alaska-software.isam.orders table, the following was entered:

Edit Data - PostgreSQL 9.4 (x86) (localhost:5432) - test - alaska-software.isam.orders

File Edit View Tools Help

No limit

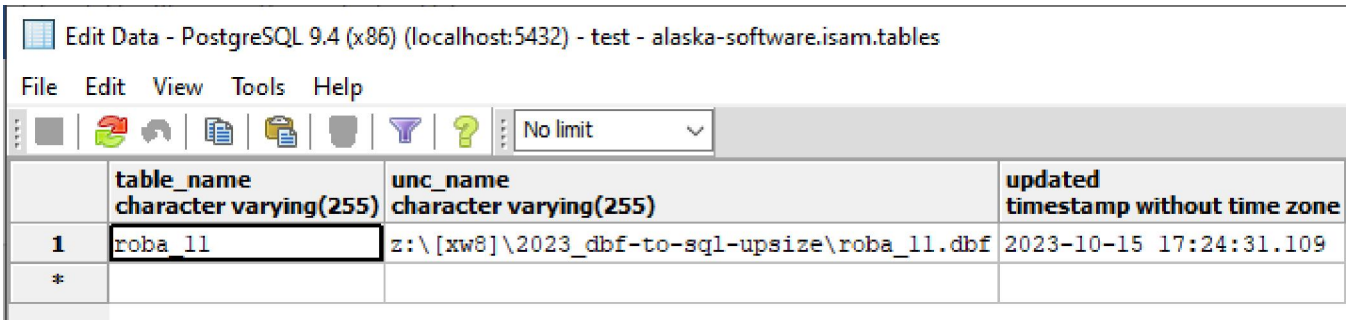
	tag character varying(255)	bag character varying(255)	for_expression character varying(255)	key_expression character varying(255)	key_column_name character varying(255)
1	INDEX_NAME	INDEX_NAME.NTX	''	ROBN_	__order_index_name_index_name
2	INDEX_CODE	INDEX_CODE.NTX	''	ROBS_	__order_index_code_index_code
3	INDEX_GTIN	INDEX_GTIN.NTX	''	ROBK_	__order_index_gtin_index_gtin
*					

levi deo table
left part of the table

key_column_type character varying(64)	key_type character(1)	key_len integer	key_dec integer	key_attributes integer	table_name character varying(255)	table_id integer	order_id [PK] integer
C	C	25	0	0	roba_11	1	1
C	C	5	0	0	roba_11	1	2
C	C	25	0	0	roba_11	1	3

desni deo table - right part of the table

U tablu alaska-software.isam.tables upisano je:
In the table alaska-software.isam.tables is written:



levi deo table - left part of the table

created timestamp without time zone	record_count bigint	update_count bigint	lock_owner integer	table_id [PK] integer
2023-10-15 17:24:29	196	588	0	1

desni deo table - right part of the table

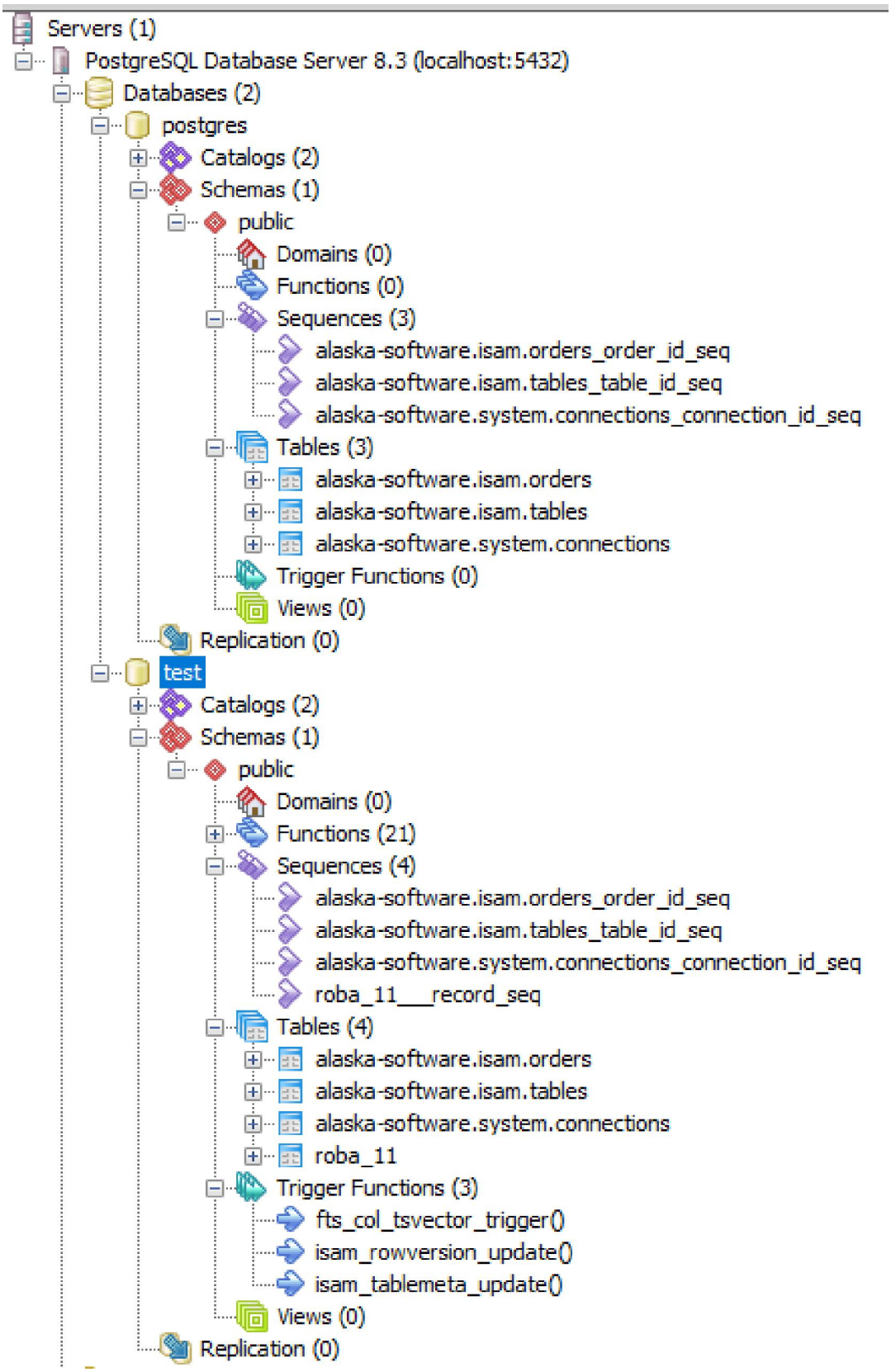
U tablu alaska-software.systems.connections nije ništa upisano.
There is nothing written in the table alaska-software.systems.connections.

U sve tri ove table koje se nalaze u databazi 'postgres' i šemi 'public' nije ništa upisano, one nemaju ni jedan red (record).
In all three of these tables that are in the 'Postgres' database and the 'public' schema, nothing is entered, they do not have a single row (record).

Zatim je program UPSIZE.EXE u **database='test' i scheme='public'** upisao tri trigger funkcije:
Then the UPSIZE.EXE program entered three trigger functions in database='test' and scheme='public':

- **-- Function: fts_col_tsvector_trigger()**
  samo kada postoji polje Full Text Search: fts_col tsvector,
  only when there is a field Full Text Search: fts_col tsvector,
- **-- Function: isam_rowversion_update()**
- **-- Function: isam_tablemeta_update()**

SL.5
Stanje na postgreSQL serveru

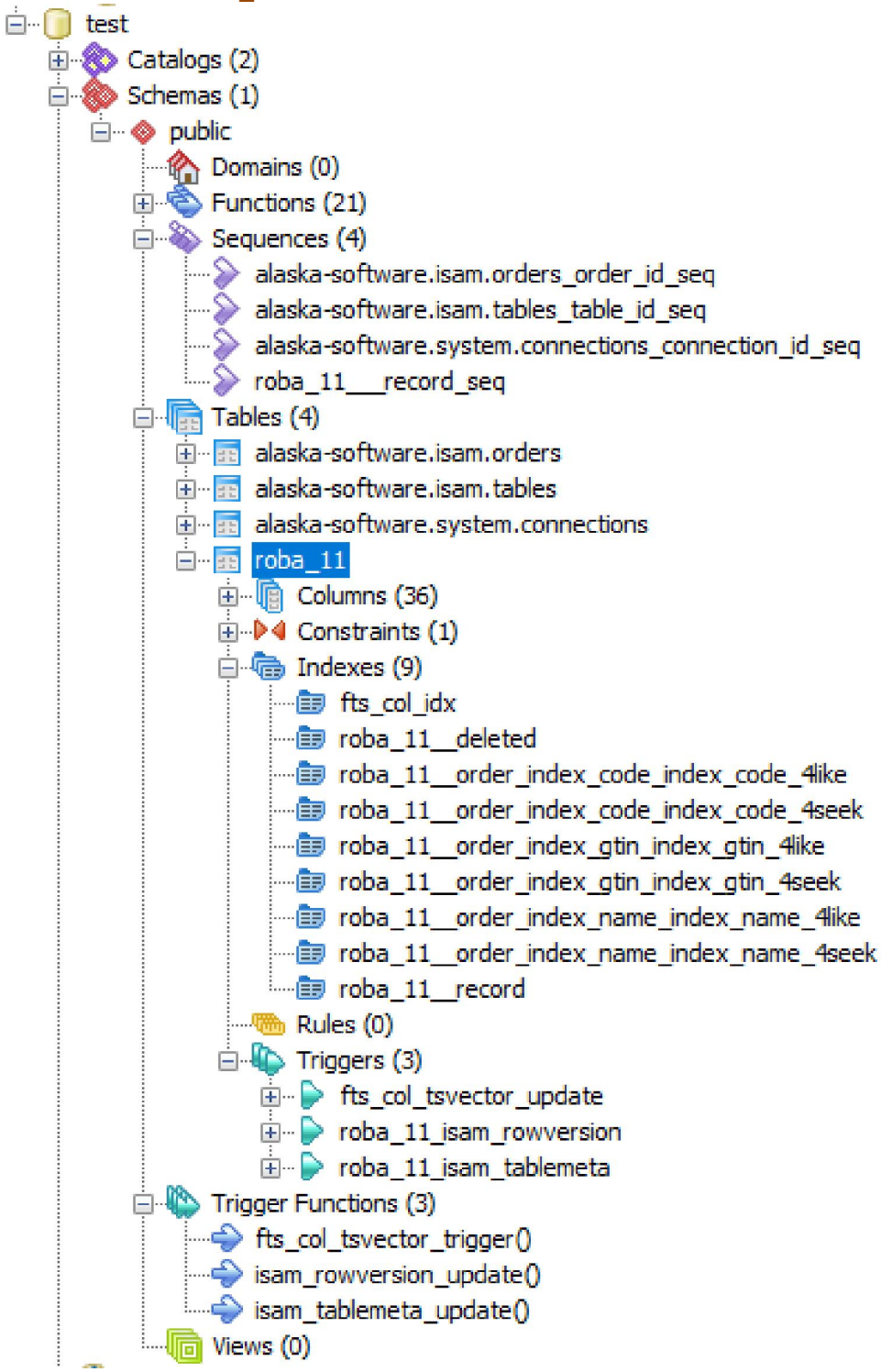


Izmene koje DBFUPSIZE.EXE unosi u novootvorenu tablu='roba_11' u database = 'test' i u scheme = 'public' (test.public.roba_11)

The changes that DBFUPSIZE.EXE makes in the newly opened table='roba_11' in database = 'test' and in scheme = 'public' (test.public.roba_11)

SL.6

UPSIZE TABLE roba_11



Ova tabla ima kolone, indekse i trigger-e kreirane putem SQL scripta:

```
-- Table: roba_11

-- DROP TABLE roba_11;

CREATE TABLE roba_11
(
  grup_ character(2),
  robs_ character(5),
  robn_ character(25),
  robk_ character(25),
  mera_ character(3),
  tari_ character(1),
  ppup_ numeric(4,1),
  ppap_ numeric(9,2),
  cena_fak_ numeric(12,2),
  cena_nab_ numeric(12,2),
  cena_vpr_ numeric(12,2),
  cena_mpr_ numeric(12,2),
  cena_dev_ numeric(12,2),
  zali_ numeric(14,3),
  dat0_ date,
  mags_ character(2),
  znak_ character(1),
  flag_ character(1),
  stat_ character(1),
  cena_pro_ numeric(12,3),
  osno_akc_ numeric(12,3),
  rok_ numeric(3),
  dobs_ character(4),
  dobk_ character(25),
  veza_ character(15),
  name_ character(200),
  opis_ text,
  __deleted boolean NOT NULL DEFAULT false,
  __record serial NOT NULL,
  __rowversion integer NOT NULL DEFAULT 1,
  __keyversion integer NOT NULL DEFAULT 0,
  __lock_owner integer NOT NULL DEFAULT 0,
  __order_index_name_index_name character(36),
  __order_index_code_index_code character(16),
  __order_index_gtin_index_gtin character(36),
  fts_col tsvector,
  CONSTRAINT roba_11_pkey PRIMARY KEY (__record)
)
WITH (OIDS=FALSE);
ALTER TABLE roba_11 OWNER TO postgres;

-- Index: fts_col_idx
-- DROP INDEX fts_col_idx;

CREATE INDEX fts_col_idx
  ON roba_11
  USING gist
```

```
  (fts_col);

-- Index: roba_11__deleted
-- DROP INDEX roba_11__deleted;

CREATE INDEX roba_11__deleted
  ON roba_11
  USING btree
  (__deleted);

-- Index: roba_11__order_index_code_index_code_4like
-- DROP INDEX roba_11__order_index_code_index_code_4like;

CREATE INDEX roba_11__order_index_code_index_code_4like
  ON roba_11
  USING btree
  (__order_index_code_index_code bpchar_pattern_ops);

-- Index: roba_11__order_index_code_index_code_4seek
-- DROP INDEX roba_11__order_index_code_index_code_4seek;

CREATE INDEX roba_11__order_index_code_index_code_4seek
  ON roba_11
  USING btree
  (__order_index_code_index_code);

-- Index: roba_11__order_index_gtin_index_gtin_4like
-- DROP INDEX roba_11__order_index_gtin_index_gtin_4like;

CREATE INDEX roba_11__order_index_gtin_index_gtin_4like
  ON roba_11
  USING btree
  (__order_index_gtin_index_gtin bpchar_pattern_ops);

-- Index: roba_11__order_index_gtin_index_gtin_4seek
-- DROP INDEX roba_11__order_index_gtin_index_gtin_4seek;

CREATE INDEX roba_11__order_index_gtin_index_gtin_4seek
  ON roba_11
  USING btree
  (__order_index_gtin_index_gtin);

-- Index: roba_11__order_index_name_index_name_4like
-- DROP INDEX roba_11__order_index_name_index_name_4like;

CREATE INDEX roba_11__order_index_name_index_name_4like
  ON roba_11
  USING btree
  (__order_index_name_index_name bpchar_pattern_ops);

-- Index: roba_11__order_index_name_index_name_4seek
-- DROP INDEX roba_11__order_index_name_index_name_4seek;

CREATE INDEX roba_11__order_index_name_index_name_4seek
  ON roba_11
```

```
  USING btree
  (__order_index_name_index_name);

-- Index: roba_11__record
-- DROP INDEX roba_11__record;

CREATE INDEX roba_11__record
  ON roba_11
  USING btree
  (__record);


-- Trigger: fts_col_tsvector_update on roba_11
-- DROP TRIGGER fts_col_tsvector_update ON roba_11;

CREATE TRIGGER fts_col_tsvector_update
  BEFORE INSERT OR UPDATE
  ON roba_11
  FOR EACH ROW
  EXECUTE PROCEDURE fts_col_tsvector_trigger();

-- Trigger: roba_11_isam_rowversion on roba_11
-- DROP TRIGGER roba_11_isam_rowversion ON roba_11;

CREATE TRIGGER roba_11_isam_rowversion
  BEFORE UPDATE
  ON roba_11
  FOR EACH ROW
  EXECUTE PROCEDURE isam_rowversion_update();

-- Trigger: roba_11_isam_tablemeta on roba_11
-- DROP TRIGGER roba_11_isam_tablemeta ON roba_11;

CREATE TRIGGER roba_11_isam_tablemeta
  AFTER INSERT OR UPDATE OR DELETE
  ON roba_11
  FOR EACH ROW
  EXECUTE PROCEDURE isam_tablemeta_update();
```

5. Formiranje SQL table iz sopstvenog programa
5. Forming a SQL table from your own program

U naslovu
In the title

4. Formiranje SQL table iz programa DBFUPSIZE.EXE
4. Forming the SQL table from the program DBFUPSIZE.EXE

prikazan je rad i rezultat rada programa DBFUPSIZE.EXE. Ljudi iz Alaska Xbase++ su to jako dobro napravili i to jako dobro radi.
the work and the result of the program DBFUPSIZE.EXE are shown. The folks at Alaska Xbase++ have done it very well and it works very well.

S pravom se postavlja pitanje: Zašto onda praviti svoj program za ovaj posao ?
The question is rightly raised: Why then create your own program for this business?

Moj odgovor:
My answer:

Program DBFUPSIZE.EXE generiše u PostgreSQL serveru kompletnu **ISAM MAŠINU** koja daje mogućnost **za istovremeni rad sa DBFNTX, DBFCDX, FOXCDX i PGDBE database engine, odnosno za istovremeni rad sa DBF fajlovima i PostgreSQL tablama.**

The program **DBFUPSIZE.EXE** generates in the PostgreSQL server a complete ISAM MACHINE that provides the possibility **for simultaneous work with DBFNTX, DBFCDX, FOXCDX and PGDBE database engine, that is, for simultaneous work with DBF files and PostgreSQL tables.**

Program PGDBE.DLL prevodi ISAM komande i funkcije na SQL jezik razumljiv PostgreSQL serveru, pa se u PGDBE aplikaciji može istovremeno komunicirati putem tih ISAM komandi i funkcija i sa DBF fajlovima i sa SQL tablama. A putem passtrough SQL-a ugrađenog u PGDBE može se na SQL jeziku komunicirati sa PostgreSQL serverom i zadatom bazom podataka.

The PGDBE.DLL program translates ISAM commands and functions into the SQL language understood by the PostgreSQL server, so the PGDBE application can simultaneously communicate with DBF files and SQL tables through these ISAM commands and functions. And through passthrough SQL built into PGDBE, it is possible to communicate with the PostgreSQL server and the given database in the SQL language.

Program DBFUPSIZE.EXE može se aktivirati iz bilo koje Alaska Xbase++ aplikacije komandom RunShell(), ali to je alat za programere i nezgodan je za isporuku korisniku i obuku korisnika za rad sa tim alatom. Takođe iz bilo koje Alaska Xbase++ aplikacije mogu se pozivati i funkcije iz biblioteke DBFUPSIZE.DLL/LIB, ali na žalost za tu biblioteku nisam našao dokumentaciju.

Moj cilj je bio:

- da napravim poslovnu aplikaciju koja će se isporučivati preko interneta i koja će posedovati sopstvenu autoinstalaciju,
- da prelazak korisnika sa DBF na SQL bude automatizovan i jednostavan tako da i korisnik, uz kratko i jasno uputstvo, može da rukuje sa tom operacijom.
- da odustanem od FTS i od INDEX-a.
- da odustanem od istovremenog rada sa DBF fajlovima i SQL tablama - Da koristim samo SQL table i ISAM komande,

Pa se nametnuo jedini izbor: treba napraviti svoj upsize alat. Treba napisati komande i funkcije iste ili slične onima koje izvršava program DBFUPSIZE.EXE a koje se mogu ugraditi u bilo koju korisničku aplikaciju i koje se mogu izvršavati iz te aplikacije bez potrebe za programom DBFUPSIZE.EXE/DLL/LIB. Posebno i zbog toga što se rad programa DBFUPSIZE.EXE može podešavati, čime se njegove komande i funkcije menjaju. Takođe i zbog najvažnijeg momenta, odnosno zbog mogućnosti kreiranja novih baza podataka i novih šema i tabli, dinamički iz korisničke aplikacije.

The DBFUPSIZE.EXE program can be run from any Alaska Xbase++ application with the RunShell() command, but it is a developer tool and is inconvenient to deliver to the user and train the user to work with the tool. Functions from the DBFUPSIZE.DLL/LIB library can also be called from any Alaska Xbase++ application, but unfortunately I have not found documentation for that library.

My goal was:

- to create a business application that will be delivered over the Internet and that will have its own application car installation,
- that the user's transition from DBF to SQL should be automated and simple so that the user, with short and clear instructions, can handle that operation.
- to give up FTS and INDEX.
- to give up simultaneous work with DBF files and SQL tables - to use only SQL tables and ISAM commands,

So the only choice was imposed: you should make your own upsize tool. You should write commands and functions the same or similar to those executed by the program DBFUPSIZE.EXE, which can be embedded in any user application and which can be executed from that application without the need for the program DBFUPSIZE.EXE/DLL/LIB. Especially because the operation of the DBFUPSIZE.EXE program can be adjusted, which changes its commands and functions. Also because of the most important moment, i.e. because of the possibility of creating new databases and new schemes and tables, dynamically from the user application.

Drugim rečima, ono što radi DBFUPSIZE.EXE moćiće da radi korisnička aplikacija uz mogućnost brojnih i brzih izmena koda odnosno načina rada sa PgSQL tablama.

In other words, what DBFUPSIZE.EXE does, the user application will be able to do with the possibility of numerous and quick changes to the code, i.e. the way of working with PgSQL tables.

Ova moja želja je nasleđe iz epohe Clipper/Xbase++ DBF database aplikacija, koje su bile sa relativno malom bazom podataka, koja se generisala i održavala iz same aplikacije, a što je kasnije dobro radilo i sa ADS serverom. Pokušavam da istu tu tehnologiju implementiram u Xbase++ aplikaciju sa PostgreSQL serverom. I gle čuda, to uspeva!

This wish of mine is a legacy from the era of Clipper/Xbase++ DBF database applications, which had a relatively small database, which was generated and maintained from the application itself, and which later worked well with the ADS server. I am trying to implement this same technology in an Xbase++ application with a PostgreSQL server. And lo and behold, it works!

Moje rešenje:
My solution:

Moje UPIZE procedure i funkcije spakovao sam u tri posebna programa:
I have packaged my UPIZE procedures and functions into three separate programs:

(1)
Program C-DBF2SQLFILE.EXE (servisni program za programera)

Iz zadatog DBF/DBT fajla formira poseban SQL script fajl sa SQL naredbom za kreiranje PostgreSQL table iz tog DBF/DBT fajla. Zatim formira poseban SQL script fajl sa podacima preuzetim iz DBF fajla i sa SQL naredbom za upis tih podataka u PostgreSQL tablu. Oba ova SQL script fajla koriste se iz programa C-POSTRGRESQL-DATABASE.EXE.

(1)
Program C-DBF2SQLFILE.EXE (developer service program)

From the given DBF/DBT file, it creates a separate SQL script file with SQL commands for creating a PostgreSQL table from that DBF/DBT file. Then it creates a separate SQL script file with data taken from the DBF file and with an SQL command to write that data into the PostgreSQL table. Both of these SQL script files are used from the program C-POSTRGRESQL-DATABASE.EXE.

(2)
Program C-POSTRGRESQL-DATABASE.EXE (dopunski program uz svaku EXE aplikaciju)

Program za upravljanje PostgreSQL bazom podataka od strane korisnika aplikacije koji se isporučuje korisniku uz aplikaciju. Sadrži popis naziva svih tabli sa kojima poslovna aplikacija radi (Lista tabli aplikacije). Sadrži SQL string sa naredbom za kreiranje svake od tabli sa popisa tabli. I nazivi table i SQL stringovi tabli primaju naziv databaze i naziv šeme kao parametre, pa se table mogu kreirati u bilo kojoj zadatoj databazi i šemi.
SQL string za kreiranje table preuzima se (copy-paste) iz programa **C-DBF2SQLFILE.EXE** i upisuje se u kod programa (hard coded) **C-POSTRGRESQL-DATABASE.EXE** gde trajno ostaje i ne može se menjati od strane korisnika aplikacije. Dakle, svaka aplikacija mora imati svoj lični dopunski program C-POSTRGRESQL-DATABASE.EXE.

Pored ovoga, program C-POSTRGRESQL-DATABASE.EXE poziva iz **biblioteke C-PGDB.DLL** funkcije za izvršenje sledećih operacija od strane korisnika aplikacije:

- Podešavanje rada sa aplikacijom: promena PostgreSQL servera, databaze, korisnika, pasworda, šeme, konekcije i ostalih parametara baze podataka i aplikacije;

- Kontrola postojanja zadate baze podataka i zadate šeme, koje su zadate u programu za Podešavanje rada sa aplikacijom;

- Kreiranje zadate baze podataka ako ista ne postoji i kreiranje zadate šeme ako ista ne postoji;

- kontrola postojanja tabli navedenih u listi (Lista tabli aplikacije).

- Kreiranje upsize tabli iz liste (Lista tabli aplikacije) ako ne postoje u bazi podataka

- Punjenje zadatih tabli podacima iz DBF fajla, koji su upisani u SQL script fajl formiran iz programa: **C-DBF2SQLFILE.EXE**

- Prikaz liste baza podataka na PostgreSQL serveru u AppBrowse browser-listi

- Prikaz liste šema u zadatoj bazi podataka u AppBrowse browser-listi

- Prikaz liste tabli u zadatoj šemi u AppBrowse browser-listi

Note:
Ako EXE aplikacija nosi naziv GENERAL-LEDGER.EXE
ovaj program nosi naziv GENERAL-LEDGER-DATABASE.EXE

(2)
Program C-POSTRGRESQL-DATABASE.EXE (supplementary program with each EXE application)

A program for managing the PostgreSQL database by the user of the application, which is supplied to the user with the application. It contains a list of names of all boards with which the business application works (List of application boards). Contains an SQL string with the command to create each of the tables from the list of tables. Both table names and table SQL strings take database name and schema name as parameters, so tables can be created in any given database and schema.
The SQL string for creating the table is downloaded (copy-paste) from the program C-DBF2SQLFILE.EXE and written into the code of the program (hard coded) C-POSTRGRESQL-DATABASE.EXE where it remains permanently and cannot be changed by the application user. Therefore, each application must have its own add-in program C-POSTRGRESQL-DATABASE.EXE.

In addition, the C-POSTRGRESQL-DATABASE.EXE program calls from the **C-PGDB.DLL library** functions to perform the following operations by the application user:

- Setting up work with the application: changing the PostgreSQL server, database, user, password, scheme, connection and other parameters of the database and application;

- Control of the existence of the given database and the given scheme, which are given in the program for setting up work with the application;

- Creation of a given database if it does not exist and creation of a given scheme if it does not exist;

- control of the existence of boards listed in the list (List of application boards).

- Creation of upsize tables from the list (List of application tables) if they do not exist in the database

- Filling the given tables with data from the DBF file, which are written into the SQL script file created by the program: C-DBF2SQLFILE.EXE

- Displaying the list of databases on the PostgreSQL server in the AppBrowse browser list

- Displaying the list of schemes in the given database in the AppBrowse browser-list

- Displaying the list of boards in the given scheme in the AppBrowse browser-list

Note:
If the EXE application is named GENERAL-LEDGER.EXE
This program is named GENERAL-LEDGER-DATABASE.EXE

(3)
Biblioteka C-PGDB.DLL/LIB (zajedničke procedure i funkcije za aplikacije)

Sadrži osnovne procedure i funkcije za upravljanje PostgreSQL bazom podataka iz bilo koje aplikacije.
Ove procedure i funkcije koriste se iz programa C-POSTRGRESQL-DATABASE.EXE koji je uvek deo neke poslovne apliakcije, a takođe i iz te poslovne aplikacije.
Tu su funkcije: Connect, Disconnect, create database, create scheme, create table, if database exists, if scheme exists, if table exists, delete from table, insert into table, vacuum table, SELECT i druge. Sve te funkcije kodirane su kao čist passtroughSQL.

Ove procedure i funkcije obezbeđuju sve navedene operacije koje se vrše iz programa C-POSTRGRESQL-DATABASE.EXE osim operacije formiranja liste (Lista tabli aplikacije) i SQL stringova svake table sa te liste.

Biblioteka C-PGDB.DLL/LIB linkuje se u svaku PGDBE Alaska Xbase++ aplikaciju koja se proizvodi u firmi COBA Systems.

(3)
Library C-PGDB.DLL/LIB (common procedures and functions for applications)

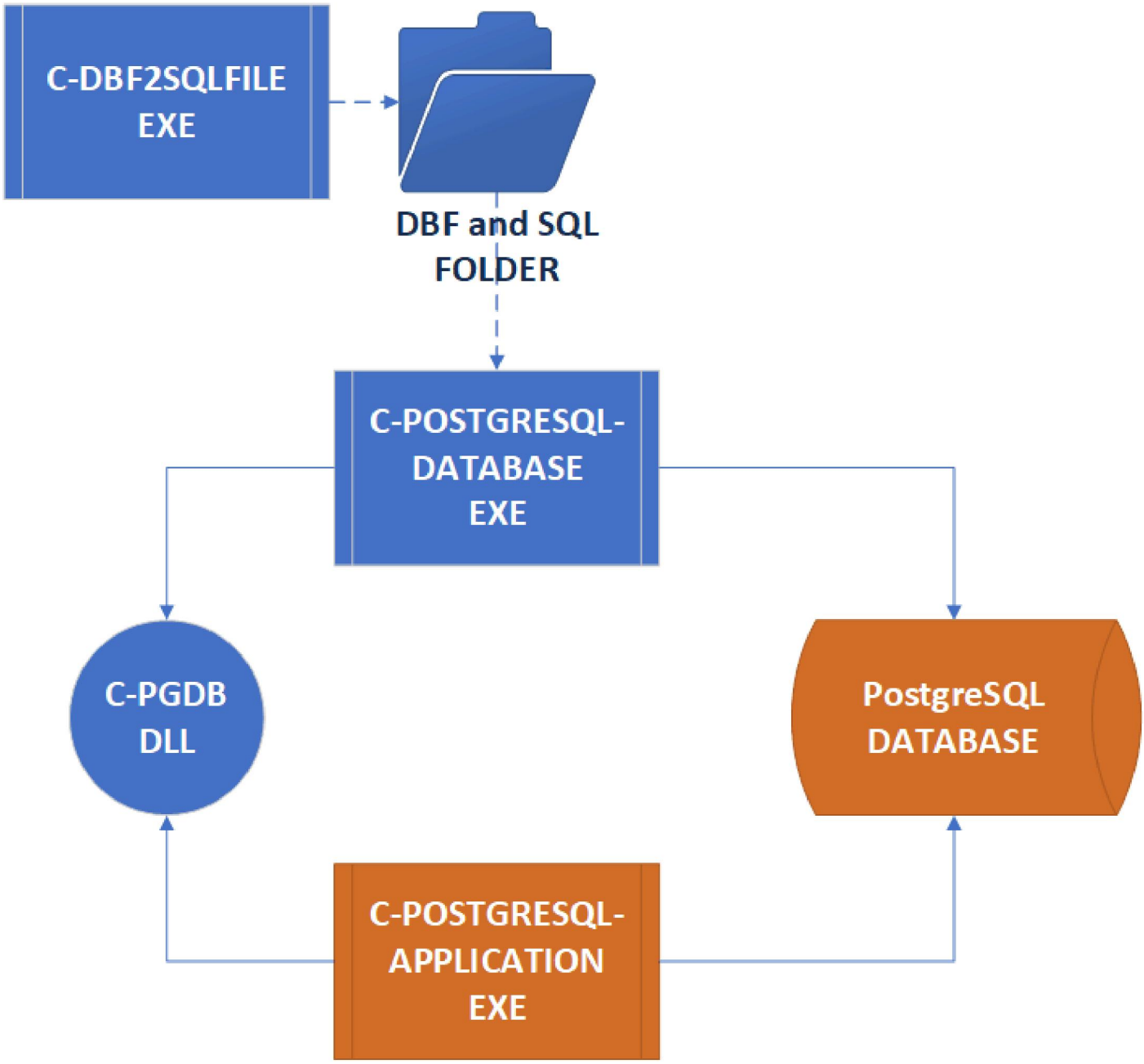
It contains the basic procedures and functions for managing a PostgreSQL database from any application.
These procedures and functions are used from the C-POSTRGRESQL-DATABASE.EXE program, which is always part of a business application, and also from that business application.
Here are the features: Connect, Disconnect, create database, create scheme, create table, if database exists, if scheme exists, if table exists, delete from table, insert into table, vacuum table, SELECT and others. All those functions are coded as pure passthroughSQL.

These procedures and functions provide all the listed operations that are performed from the program C-POSTRGRESQL-DATABASE.EXE except the operation of creating a list (List of application tables) and SQL strings of each table from that list.

The C-PGDB.DLL/LIB library is linked in every PGDBE Alaska Xbase++ application produced by COBA Systems.

PRILOG: ŠEMA ORGANIZACIJE PROGRAMA ZA RAD SA UPSIZE POSTGRESQL DATABAZOM

PROGRAM ORGANIZATION SCHEME FOR WORKING WITH THE UPSIZE POSTGRESQL DATABASE



PRILOG: konstrukcija table roba_11 za buduću Alaska Xbase++ PostgreSQL aplikaciju
Construction of table roba_11 for future Alaska Xbase++ PostgreSQL application

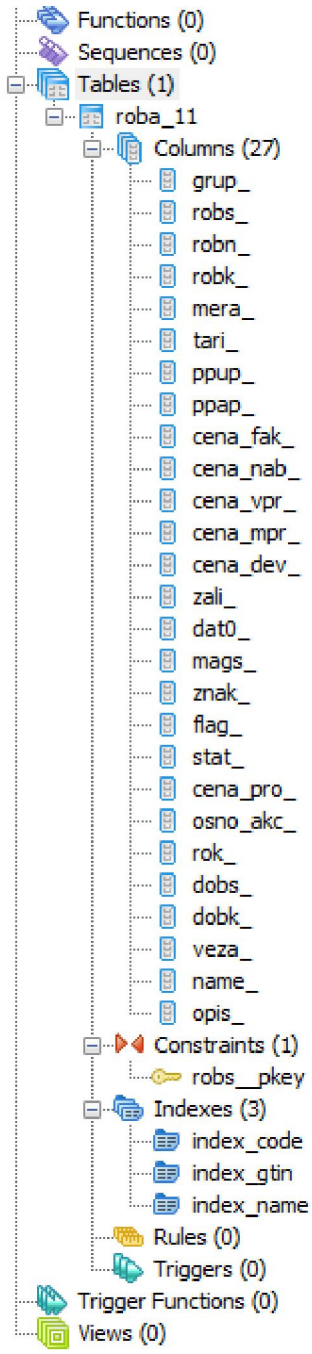
```
DROP TABLE roba_11
CREATE TABLE roba_11
(
  grup_ character(2),          -- product group
  robs_ character(5),          -- product code
  robn_ character(25),         -- trade name of product
  robk_ character(25),         -- barcode produces
  mera_ character(3),          -- product measurement unit
  tari_ character(1),          -- product tariff
  ppup_ numeric(4,1),          -- product tax rate - value-added tax
  ppap_ numeric(9,2),          -- product tax rate - excise tax
  cena_fak_ numeric(13,2),     -- invoice price
  cena_nab_ numeric(13,2),     -- purchase price
  cena_vpr_ numeric(13,2),     -- price without tax
  cena_mpr_ numeric(13,2),     -- price with tax
  cena_dev_ numeric(13,2),     -- foreign exchange price
  zali_ numeric(15,3),         -- stock
  dat0_ date,                  -- expiration date
  mags_ character(2),          -- warehouse code
  znak_ character(1),          -- free
  flag_ character(1),          -- free
  stat_ character(1),          -- type of product: R= rubber,U=service,M=material,P=product
  cena_pro_ numeric(13,3),     -- average price
  osno_akc_ numeric(13,3),     -- minimum stock
  rok_ numeric(3),             -- payment deadline in days
  dobs_ character(4),          -- manufacturer-supplier code
  dobk_ character(25),         -- manufacturer-supplier catalog number
  veza_ character(15),         -- product code that is a replacement for this product
  name_ character(200),        -- long product name
  opis_ text,                  -- textual detailed product description
  CONSTRAINT robs__pkey PRIMARY KEY (robs_)
)
WITH (OIDS=FALSE,autovacuum_enabled=true);
ALTER TABLE roba_11 OWNER TO postgres;

-- Index: index_code
DROP INDEX index_code;
CREATE INDEX index_code
  ON roba_11
  USING btree
  (robs_);

-- Index: index_name
DROP INDEX index_name;
CREATE INDEX index_name
  ON roba_11
  USING btree
  (robn_);

-- Index: index_gtin
DROP INDEX index_gtin;
CREATE INDEX index_gtin
  ON roba_11
  USING btree
  (robk_);
```

Rezultat izvršenja gornjeg SQL scripta u PostgreSQL databazi je
The result of executing above SQL script in PostgreSQL database is:



Podrazumeva se da se ovo koristi u aplikaciji koja koristi **PostgreSQL bazu podataka i SQL table** putem komandi:

UNIVERZAL SQL komandi (ugrađene u PGDBE ili ODBCDBE)
PGDBE SQL komandi (ugrađene u PGDBE)
ODBCDBE SQL komandi (ugrađene u ODBCDBE)

This is implied to be used in an application that uses a PostgreSQL database and SQL tables via commands:

UNIVERSAL SQL commands (embedded in PGDBE or ODBCDBE)
PGDBE SQL commands (built into PGDBE)
ODBCDBE SQL commands (built into ODBCDBE)

Alaska Xbase++ aplikacija koja radi preko PGDBE.DLL (PostgreSQL DataBaseEngine) radiće sa ovom PgSQL tablom samo preko SQL komandi i funkcija: PGDBE SQL i UNIVERZAL SQL ugrađenih u PGDBE.DLL. Ako aplikacija radi i preko ODBCDBE.DLL radiće i preko ODBCDBE SQL komandi i funkcija ugrađenih u ODBCDBE.DLL

Međutim, svaka dosadašnja Alaska Xbase++ aplikacija je napisana za DBF/DBT/NTX ISAM bazu podataka i koristi ISAM komande i funkcije ugrađene u DBFDBE i FOXDBE database engine i ne sadrži u svom kodu nikakve SQL komande i funkcije prethodno pobrojane. A cilj je da se te SQL komande i funkcije ne pišu i ne koriste – da se ne menja kod aplikacije, već da se preko nekog prevodioca (omotač ili wrapper) postojeće ISAM komande i funkcije prevedu na SQL komande i funkcije koje će raditi sa SQL tablom roba_11. Taj cilj pokušavamo da ostvarimo u ovom primeru aplikacije.

Alaska Xbase++ application running through PGDBE.DLL (PostgreSQL DataBaseEngine) will work with this PgSQL table only through SQL commands and functions: PGDBE SQL and UNIVERSAL SQL embedded in PGDBE.DLL. If the application also works through ODBCDBE.DLL, it will also work through ODBCDBE SQL commands and functions embedded in ODBCDBE.DLL

However, every Alaska Xbase++ application to date has been written for the DBF/DBT/NTX ISAM database and uses the ISAM commands and functions built into the DBFDBE and FOXDBE database engines and does not contain in its code any of the SQL commands and functions listed above. And the goal is that these SQL commands and functions are not written and not used - that the application code is not changed, but that the existing ISAM commands and functions are translated into SQL commands and functions that will work with SQL through a translator (wrapper) table roba_11. We are trying to achieve that goal in this application example.

Note:

U delu 2 ove knjige biće opisan i dokumentovan program
Part 2 of this book will describe and document the program
C-DBF2SQLFILE.EXE (developer service program)

U delu 3 ove knjige biće opisan i dokumentovan program
Part 3 of this book will describe and document the program
C-POSTRGRESQL-DATABASE.EXE (supplementary program with each EXE application)

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