A PostgreSQL adatbázis-kezelő használata (2. rész)

Sorozatunk előző részében az eljárások készítéséig jutottunk el, most az ügyféloldali programok bemutatásával folytatjuk.

A PG négyféle C/C++ API-felülettel rendelkezik. Ez a gazdag lehetőség azt az igényt tükrözi, hogy a különféle feladatokhoz és programozói elvárásokhoz másmás eszközkészlet vezethet a legkönnyebben megoldáshoz. A legfontosabb és legalapvetőbb API a „natív" C alapú programozói felület, amit gyakran „PQ" API-nak is hívnak, utalva arra, hogy ennek az eszközkészletnek a megvalósítása a ***libpq.a***c és ***libpq.so*** könyvtárakban található. A példaprogramban érdemes észrevenni azt, hogy bár a PQ API elemi szintű eszközkészlet, mégis könnyű programozni.

A ***PQ*** könyvtár mellett létezik egy egyszerűsített, PGEASY nevű API (ennek helye a ***libpgeasy.a*** és a ***lobpgeasy.so*** fájl), melynek megvalósítása során az volt a cél, hogy amit csak lehet, az API függvényei a háttérben önműködően kezeljenek. Ez azt a hátrányt is jelenti, hogy a programozó nem férhet hozzá könnyen olyan adatokhoz, mint például a munkamenetleíró, ugyanakkor az alkalmazói program sokat egyszerűsödhet.

A C++ adta absztrakciós lehetőségeket aknázza ki a harmadik, PQ++ nevű felület (ennek helye a ***libpq++.a*** és a ***libpq++.so*** fájl). Ez – kihasználva az objektumközpontú programozás fegyvertárát – egyesíti magában a PQ API finom részletezési lehetőségét és a PGEASY egyszerűbb programozhatóságát.

Egy SQL programozási környezetben alapkövetelménnyé vált, hogy egy gazdanyelvbe (esetünkben ez a C/C++ lesz) az SQL-parancsokat úgy ágyazhassuk be, mintha azokat egy SQL-konzolon adtuk volna ki. Ilyen például az `ORacle Pro` C vagy a PG negyedik típusú C/C++ API-készlete is (ennek helye a ***libecpg.a*** és a ***libecpg.so***). Itt az az alapelv, hogy egy ***program.pc*** nevű forrásprogramot készítsünk, amiben `EXEC SQL...` kezdetű makrókat helyezhetünk el, biztosítva az SQL-parancsok közvetlen beírását. A C/C++-változók és az SQL-parancsok közötti kapcsolattartást a változó neve elé elhelyezett kettős pont (:) karakter jelöli (ennek neve külső változó). Ebben a formában programunkat a C/C++-fordító nem tudja lefordítani, hiszen nem ismerheti az `EXEC SQL...` makrók feldolgozásának a módját. A PG rendelkezik egy `ecpg` nevű, beágyazott SQL-előfeldolgozóval, ami biztosítja a `*.pc` fájlok `*.c`-vØ alakítását, azaz egy olyan C/C++-program elkészítését, ami egy C/C++ fordítóprogrammal már lefordítható. A beágyazott SQL alapú program a natív API gyorsírásos változatának is tekinthető, hiszen az önműködően létrehozott C/C++-programot kézzel is megírhattuk volna.

Lényeges kiemelni, hogy az ismertetett négyféle API (sőt maga az adatbázis-kezelő kiszolgáló is) a Unix/Linux mellett Windows alatt is használható, így az operációs rendszerek különbsége PG-adatbázisunk elérésének nem emel gátat.

A továbbiakban nézzünk egy-egy példát az egyes API-k gyakorlati használatára! A programozási feladat minden esetben az lesz, hogy kilistázzuk a ***cs_adatok*** adatbázis-telefontábláját, majd kipróbálás gyanánt egy új sort (ez alól a `pgeasy` példa kivétel) szúrunk be a táblába.

4. lista PostreSQL-adatbázis natív módú elérése

```
#include <stdio.h>
#include <stdlib.h>

/* Ez az include fÆjl kell a nat v
   PostgreSQL elØrØshez */
#include "/usr/include/pgsql/libpq-fe.h"

int main(void)

char sqlStr[256]; // Itt lesz a mindenkori
                  // SQL-parancs.
PGconn *conn; // A kapcsolat le r ja.
PGresult *result; // Az eredmØnyhalmaz.
int i; // A for ciklushoz sz ksØges.

// Kapcsol dÆs az adatbÆzishoz, ami a
// gazdagØp 5432 kapujÆn vÆrakozik egy
// kapcsol dÆsi kØrØsre. Az adatbÆzisunk
// neve: cs_adatok
// user/password = postgres/inyiri
conn = PQsetdbLogin
➥("localhost","5432","","","cs_adatok",
➥"postgres","111111");

// MegvizsgÆljuk, hogy siker lt-e a
// kapcsol dÆs
if ( PQstatus(conn)==CONNECTION_BAD )

printf("!!! Hiba az adatbÆzishoz
➥kapcsol dÆskor...n");
exit(1);
...
```

A PQ (natív PG) API alapú mintapélda

A program teljes forráskódját az ***alap_pg.c*** fájl tartalmazza (4. lista, 44 CD Magazin/PostgreSQL könyvtár), amibe az egyes programsorok közé beírtam azokat a megjegyzéseket, amelyek elégségesek a működés megértéséhez.

Az ***alap_pg.c*** program fordítása és szerkesztése a következő paranccsal lehetséges:

```
# gcc alap_pg.c -o alap_pg.exe -lpq
```

A kapott ***alap_pg.exe*** program lefuttatása a következő eredményt jeleníti meg a képernyőn:

```
Nyiri Imre   --- 123456789
Kiss JÆnos   --- 123456789
PØntek TamÆs   --- 123456789
```

```
Janisovszky KÆroly  --- 11111
A postgreSQL pr baprogram V GE...
```

A PGEASY API alapú mintapélda

A program forráskódja az ***egyszeru_pg.c*** fájlban található (lásd *5. listánkon*). Vegyük észre, hogy ez a program nem kezeli explicit módon a munkamenetet, hiszen a kapcsolódás után az SQL-lekérdezésben a (`doQuery(...)`)-ben ez az adat nincs megadva.
Az ***egyszeru_pg.c*** program fordítása és szerkesztése a következő paranccsal lehetséges:

```
# gcc egyszeru_pg.c -o egyszeru_pg.exe -lpgeasy
```

Az ***egyszeru_pg.exe*** lefuttatása után a telefontábla tartalmát ez a program is kilistázza.

A beágyazott SQL alapú mintapélda

A program teljes forráskódját a ***beagyazott_pg.pc*** fájl (6. lista, 44. CD Magazin/PostgreSQL könyvtár) tartalmazza. Tekintettel arra, hogy a beágyazott SQL szabványos módszer, az ilyen programok készítését nagyon sok forrásból megismerhetjük. Nézzük tehát a „Listázás és egy beszúrás" mintaprogramot!
A futtatható program előállítása két lépésből áll. Az első lépésben a PG fejlesztői készletben található `ecpg` makrófeldolgozó segítségével a következő parancs révén előállítjuk a ***beagyazott_pg.c*** forrásfájlt:

```
# ecpg beagyazott_pg.pc
```

A kapott C-forrásprogramokat lefordíthatjuk

```
# gcc beagyazott_pg.c -o beagyazott_pg.exe
  ➥-lecpg -I/usr/include/pgsql
```

A ***beagyazott_pg.exe*** lefuttatása után az előző mintaprogramokhoz hasonló kimenetet kapjuk meg a képernyőn.

A C++ (PQ++) API alapú mintapélda

Ennek a mintapéldának a teljes forrása a ***cpp_pg.cpp*** fájlban (7. lista, 44. CD Magazin/PostgreSQL könyvtárában) található.
A program feladata ugyanaz, mint az eddigi C/C++ alapú példáknak. A `PGDatabase` osztály jelképezi a PG-adatbázis elérését és használatát. A forráskód egyszerűségét főleg az egységbezárás és az alapértelmezett értékek lehetőségének a kihasználása biztosítja.
A program fordítása és szerkesztése az alábbi paranccsal történhet, melynek eredményeképpen megkapjuk a ***cpp_pg.exe*** futtatható programot.

```
# gcc cpp_pg.cpp -o cpp_pg.exe -lpq++
  ➥-I/usr/include/pgsql
```

Ügyféloldali programozás Java-környezetben

A Java-környezet egyszerű, parancssoros használhatósága érdekében érdemes egy kis segítő héjprogramot használatba venni. Miért? A Java nyelvű forrásprogramok fordítása és futtatása is ugyanazt a dinamikus osztálymegkeresési (a fájlrendszerben az osztályok a ***.class*** fájlok) és -betöltési módszert követi. A dinamikusan betöltendő osztályok a `CLASSPATH` környezeti változóban megadott helyeken keresődnek. A héjprogram `SetCLASSPATH` függvénye önműködően beállítja a

5. lista PostreSQL-adatbázis easy módú elérése

```
#include <stdio.h>
#include <stdlib.h>

/* Ez az include fÆjl sz ksØges az easy
   PostgreSQL elØrØshez */
#include "/usr/include/pgsql/libpq-fe.h"
#include "/usr/include/pgsql/libpgeasy.h"

int main(void)

char sqlStr[256]; // Itt lesz a mindenkori
                  // SQL-parancs
char nev[50];
char szam[20];

// Kapcsol dÆs az adatbÆzishoz
connectdb("dbname=cs_adatok user=postgres
➥password=111111");

// VØgrehajtunk egy SQL select parancsot
// Figyelj k meg azt az eleganciÆt, ahogy
// a C/C++ nyelv seg tsØgØvel dinamikusan
// eliÆll tjuk az SQL-parancsot!
sprintf(sqlStr, "select %s, %s from
➥telefon", "nev", "szam");
doquery( sqlStr );

// Ki rjuk az E tÆblÆt a konzolra
// A fetch(...) f ggvØny vÆltoz  szÆmœ
// ØrtØket tartalmazhat. Most a nev Øs
// szam nevß meziket kØrj k le.
while ( fetch(nev,szam) != END_OF_TUPLES )

printf("N?v: %s Sz?m: %sn", nev, szam);


// Az adatbÆzis-kapcsolat bontÆsa
disconnectdb();

printf("A PostgreSQL pr baprogramnak
➥V GE...n");
return 0;
...
```

`CLASSPATH`-t. Ennek csak annyi az előfeltétele, hogy a használandó ****.jar*** és ****.zip*** fájlok hivatkozásai a parancsfájlban megjelölt ***ext*** könyvtárban létre legyenek hozva.
A SuSE 8.0 alatt a ***/usr/share/pgsql*** könyvtárban található meg a ***jdbc7.1-1.2.jar*** fájl, ami a PG JDBC-meghajtója (megjegyzés: ez a jar fájl az Internetről is letölthető, sőt egy 1.3-as változata is létezik már). Erre a fájlra hozzunk létre egy hivatkozást az ***ext*** könyvtárban. A hivatkozás neve bármi lehet, de ****.jar*** legyen a fájl kiterjesztése. A segítő parancsfájl a ***ffjava.sh*** fájlban (8. lista, 44. CD Magazin/PostgreSQL könyvárában) található.
Az ***ffjava.sh*** parancsfájl fordításra és futtatásra is használható. A ***pg_teszt.java*** program fordítása a ***fjava.sh -c pg_teszt.java***, míg a lefordított program `pg_teszt` osztályának elindítása az `ffjava.sh -r pg_teszt` paranccsal lehetséges.

6. lista PostreSQL-adatbázis beágyazott módú elérése

```
EXEC SQL WHENEVER SQLERROR sqlprint;

int main(void)

//
// Az itt megadott vÆltoz k hasznÆlhat k az
// EXEC SQL-kifejezØsekben k lsı
// vÆltoz kØnt. Erre val  hivatkozÆskor
// ":" kettıs pontot tesz nk elØ.
//
EXEC SQL BEGIN DECLARE SECTION;
char nev[50];
char szam[20];
char sqlStr[256];
EXEC SQL END DECLARE SECTION;

// Kapcsol dÆs az adatbÆzisunkhoz.
EXEC SQL CONNECT TO
cs_adatok@localhost:5432 ➥user postgres
using "111111";

// Ez lesz az SQL-parancs
sprintf(sqlStr, "select %s, %s from
➥telefon", "nev", "szam" );

// Az SQL-parancs elıfeldolgozÆsa
// (optimalizÆlÆsi lehetısØg)
EXEC SQL PREPARE s_telefon FROM :sqlStr;

...
```

Ennyi előkészület és környezetkialakítás után írjuk meg szokásos 1 select 1 insert típusú próbaprogramunkat Java nyelven is! Ennek a nyelvi környezetnek a nagy előnye az, hogy szinte minden módszerre, így az adatbázisok kezelésére is átgondolt objektumközpontú, szabványos eszközrendszert bocsát a rendelkezésünkre.
A program szellemisége egyszerű: létrehoz egy `Connection` objektumot (melynek `conn` a neve), ami egy adatbázis-kapcsolat objektumközpontú megfogalmazása. A `conn` objektum képes egy SQL-parancsot képviselő `Statement` osztálybeli objektumot szolgáltatni, ami már egy-egy valóságos SQL-parancs kiadását is biztosítja. A `select` parancsok eredménytábláját egy `ResultSet` osztálybeli `rs` objektum képviseli. Érdemes észrevenni, hogy itt sincs semmi új a nap alatt, hiszen szemléletében szinte az összes objektumközpontú adatbázis-kezelő ezt az osztály-együttműködési módszert követi.
A programban (fájl neve ***pg_teszt.java***) elhelyeztem azokat a megjegyzéseket, amik a kód könnyebb megértését szolgálják (9. lista, 44. CD Magazin/PostgreSQL).
A programnak létezik egy `getMagyarString()` tagfüggvénye, ami példát mutat arra, hogy az `rs.getString()` mellett más módon is hozzájuthatunk az oszlopok adataihoz. Erre a módszerre azért volt szükség, mert egy PG-adatbázis alapértelmezett kódolása az SQL_ASCII, ugyanakkor a karakterláncokat a Java belül Unicode-ban tárolja. Itt az volt a gond, hogy egy C++-programmal ISO-8859-2-ben dolgoztunk, aminek `insert`, `update` adatait a PG-adatbázis a saját

7. lista PostreSQL-adatbázis C++-elérése

```
...

int main(void)

char sqlStr[256]; // Itt lesz a mindenkori
                  // SQL-parancs
int i; // A for ciklusvÆltoz ja

// AB az adatbÆzis-objektum. A lØtrehoz
// f ggvØnyben az adatbÆzishoz kapcsol dÆs
// is megt rtØnik
PgDatabase AB("dbname=cs_adatok
➥user=postgres password=111111");

// Ha nem sikeres a connect, akkor vØge
if ( AB.ConnectionBad() )

cerr << "!!! Hiba az adatbÆzishoz
➥kapcsol dÆskor...n";
exit(1);

...
```

SQL_ASCII kódolásában bájthűen tárolta. Ezt azonban (az ő, ű betűket) a Java `rs.getString("nev")` tagfüggvény nem helyesen olvasta be. A végső megoldás az lett, hogy az adatbázis kódolása LATIN2-re lett meghatározva. Ekkor már az egyszerű `getString()`, `setString()` (lásd a `beszur()` forrását) tagfüggvények hibátlanul működnek.
A teljesség kedvéért nézzük meg azt a lehetőséget is, hogy miként történik egy tárolt eljárás meghívása a Java nyelvből. A jdbc leírás szerint ezt a feladatot a `CallableStatement interface`-t megvalósító osztály tudná végrehajtani, ez a felület azonban a PG-ben nincs megvalósítva. Helyette ezt a feladatot egy `select` eljárásba ágyazott függvényhívással oldhatjuk meg. A példaprogram a már létrehozott `MyFunc(integer)` meghívására mutat egy esetet (forrásprogram ***pg_storproc.java***). A PG-ben a `select f ggvØny()` forma is használható lenne, azaz a dualtáblára nincs is szükségünk, ennek kihasználása viszont a program hordozhatóságát rontaná (lásd a 10. listát, ami a 44. CD Magazin/PostgreSQL könyvtárában található).
A PG programozói felületek a `select`, `insert`, `update`, `delete` (DML=Data Manipulation Langauge műveletek) parancsokon kívül a többi SQL-utasítás (DDL=Data Definition Language műveletek) kiadását is támogatják. Erre rövid példa a következő három sor, ami az adatbázisból kitörölné a telefontáblát:

```
Statement st = conn.createStatement();
ResultSet rs = st.executeQuery("DROP TABLE
telefon");
st.close();
```

Grafikus segédprogramok

A `psql` nagyszerű eszköz adatbázisaink objektumainak kezelésére, de sokszor egy-egy grafikus eszköz használata is nagyon jól jön. A phpPgAdmin PHP-ben megírt webes ügyfél. A phpPgAdmin telepítése egyszerű. A webkiszolgálónk dokumentumterületére egyszerűen másoljuk be az összes fájlt, amit a `phpPgAdmin.tar.gz` tartalmaz. Én az Apache HTTPD-t

használom, dokumentumgyökerem: */var/www/htdocs*. A fenti tar.gz fájl-t e könyvtár alatt egy phpPgAdmin könyvtárba csomagoltam ki, így a böngészőből a ➲ http://gépnév/phpPgAdmin/index.php címmel el tudom indítani. Ezzel az eszközzel a leggyakoribb feladatokat el lehet végezni. Könnyen menedzselhetjük vele adatbázis-objektumainkat. A kioldók és tárolt eljárások készítése is eléggé kényelmesen elvégezhető vele. Nagy erénye, hogy bárhonnan grafikusan is elérhetjük az adatbázisunkat, hiszen csak egy böngésző szükséges hozzá az ügyféloldalon. A phpPgAdmin honlapja és letöltési helye a ➲ http://phppgadmin.sourceforge.net címen található.
A másik elterjedt GUI-ügyfél a `pgaccess`. Ezt az eszközt mostanában ismét feszített iramban elkezdték továbbfejleszteni, ezért javaslom, hogy folyamatosan kövessük az új változatokat, amelyek remélhetőleg elsősorban az űrlap- és riportkészítési lehetőségekben fognak újdonságokat hozni.
Egyszerű beviteli űrlapok és riportok készítésére már most is van lehetőség. A pgaccesshez egy HTML alapú leírás és egy oktatóanyag tartozik, így aki erről a programról többet szeretne tudni, olvassa el ezeket az írásokat. A pgaccesss a ➲ http://www.pgacces.org webcímen érhető el.
Befejezésül fontos kiemelni, hogy a Java-felületen sokféle grafikus adatbázis-kezelő GUI-eszköz létezik. Ezeknél általában két jellemzőt kell beállítani:

- Az adatbázis `JDBC driver` osztályának a nevét és a helyét (org.postgresql.Driver).
- A kapcsolódási karakterláncot (jdbc:postgresql://localhost:5432/cs_adatok).

A kapcsolók megadása után már használhatjuk is legújabb Java swinges adatbázis-kezelő felületünket.

Windows alapú ügyfélprogramok készítése

A PG natív elérését lehetővé tévő DLL-könyvtárak Windowson is rendelkezésünkre állnak, néha azonban egyszerűbb az ODBC-meghajtót használni. A PG ODBC-meghajtó Windows telepítőcsomag formájában letölthető az Internetről.
A PG ODBC-meghajtó telepítését követően PG adatbázisokra mutató ODBC adatforrásokat hozhatunk létre. Az adatforrásra az Excel, MS Access... programokból ***odbc_cs_adatok*** néven hivatkozhatunk. A lényeg az, hogy ezen alias név mögött a gazdagép-, a kapu- és az adatbázisnév-adatokat tároljuk.
A C/C++-, Java-ügyfélprogramok megértése után valószínűleg már nem is csodálkozunk azon, hogy miért pont ezek az adatok szükségesek egy TCP/IP-n keresztül történő PG-adatbázis ODBC-n keresztüli eléréséhez.
Érdekes lehetőséget kínál Delphi-, Borland C++ Builder-programozók számára az az út, hogy egy ODBC-kapcsolatot egyszerűen BDE (Borland Database Engine) alias névre lehet alkalmazni, azaz az ***odbc_cs_adatok*** névre hivatkozva a BDE Administrator program segítségével meg lehet határozni egy ***bde_cs_adatok*** adatbázis alias nevet, amit a Delphi-programokban már közvetlenül használhatunk. Nyitva áll tehát az út a különféle windowsos ügyfélprogramok készítése előtt.

Egyéb lehetőségek

A PG-adatbázisok természetesen még sok népszerű programozási eszközből kezelhetők. Ezek közül a legfontosabb talán a PHP, amiben olyan kiváló dolgok készültek, mint a már említett phpPgAdmin webes PG-adatbázis ügyfélkörnyezet. A PHP és a PG kapcsolatáról a Linuxvilág 2000. decemberi (I. évfolyam, 2. szám) és 2001. június–július havi (II. évfolyam, 6–7. szám) számai tartalmaznak egy-egy érdekes cikket. A PHP PG programozási lehetőségeit egyébként az Internetről letölthető és nagyrészt már magyarra is lefordított, több mint 1000 oldalas PHP-kézikönyv is részletesen tartalmazza.
A Perl nyelvhez is létezik egy PG-csomag, amit SuSE 8.0 alatt a postgresql-perl csomag telepítésével kapunk meg. A Perl az adatbázis-kezelő függvényeket egy DBI modulban (Database Interface Modul) tárolja, de minden adatbázis-kezelőhöz még a megfelelő adatbázis-kezelőhöz tartozó DBD (DataBase Dependent) modult is telepíteni kell. Nézzük meg szokásos mintalekérdezésünk Perlben megírt változatát!

```
#!/usr/bin/perl
use Pg; # HasznÆljuk a Pg csomagot
$conn = Pg::connectdb("dbname=cs_adatok");
# Kapcsol dÆs
die $conn->errorMessage unless
          ➥PGRES_CONNECTION_OK eq
          ➥$conn->status; #HibakezelØs
# A szokÆsos select parancsunk kiadÆsa j n:
$result = $conn->exec("select nev, szam from
                ➥telefon");
#HibakezelØs
die $conn->errorMessage unless PGRES_TUPLES_OK
          ➥eq $result->resultStatus;

// Ki rjuk a konzolra a lekØrdezØs eredmØnyØt:
while (@row = $result->fetchrow) {
  print @row, "\n";
}
```

A Linuxvilág 2002. augusztusi száma (III. évfolyam, 8. szám) egy érdekes lehetőséget, a clip (Clipper) fejlesztői környezetet mutatja be, ezen belül arra is választ ad, hogy hogyan lehet PG-adatbázisokat Clipper nyelvű programokból elérni.
A fentieken kívül a PG programozhatóságát TCL/Tk (a tárgyalt pgaccess GUI-ügyfél is ebben készült) és Python-környezetben is megvalósították.
Remélem, hogy a röviden bemutatott PG programozási eszközök gazdagsága mindenkinek felkeltette az érdeklődését a PostgreSQL adatbázis-kezelő iránt, amit nyugodtan tekinthetünk a világ egyik legkiforrottabb adatbázis-kezelő rendszerének.
Kellemes programozást kívánok mindenkinek!

Irodalomjegyzék

- A PostgreSQL alapleírása (a Linux-terjesztések is tartalmazzák): ➲ http://www.hu.postgresql.org (a PG honlapjának magyarországi tükörkiszolgálója)
- ***Bruce Momjian***: PostgreSQL Introduction and Concepts (Addison-Wesley, 2001). E könyv pdf formátumú alakja szabadon letölthető az Internetről.
- ***Jeffrey D. Ullman–Jennifer Widom***: Adatbázisrendszerek (Alapvetés) (Panem, 1998)
- ***Joe Celko***: SQL felsőfokon (Kiskapu Kft., 2002): ➲ http://www.kiskapu.hu, ISBN 963 9301 20 5

Nyíri Imre (inyiri@mol.hu)
Jelenleg a MOL Rt.-nél dolgozik. Informatikai vállalkozásában az Internet, a Linux, valamint a Java-programozás gyakorlati hasznosításával foglalkozik. Örök szerelme a C++ maradt.