Instalujeme databázový server PostgreSQL

Pavel Janík ml.
http://www.janik.cz

Úvod

Operační systém Linux je velmi často používán jako router, firewall nebo webový či aplikační server. Záměrně jsem zde nezmínil možnost nasazení Linuxu jako databázového serveru, protože právě tomuto nasazení se budeme věnovat v sérii článků na tomto serveru.

Pro Linux existuje velké množství databázových serverů od těch nejmenších, které slouží spíše k seznámení se s databázemi a jejich principy (miniSQL společnosti Hughes), přes ty nejčastěji používané (MySQL) až po robustní Open Source databázové servery jako je PostgreSQL. Samozřejmě nesmíme zapomenout ani na komerční databázové servery jako je Oracle, Sybase, Informix nebo Interbase, která je dostupná (resp. bude) pod licencí IPL. Další alternativou může být např. česká databáze WinBase602.

Příprava instalace

V tomto článku se budeme zabývat instalací databázového serveru PostgreSQL, který byl před nedávnem uvolněn ve verzi 7.0 a později doznal ještě dvou kosmetických úprav až do nynější verze 7.0.2, kterou si můžete stáhnout ze serveru ftp://ftp.postgresql.org/. Verze 7.1, která by již měla mít odstraněn limit na velikost jednoho řádku tabulky, by měla být k dispozici na konci srpna nebo v září.

PostgreSQL je šířen ve zdrojových textech ve dvou podobách. První z nich (soubor `postgresql-7.0.2.tar.gz`) obsahuje kompletní distribuci. Druhá obsahuje stejný balík, který je ale rozdělen do několika souborů a je na uživateli, aby si vybral, které z nich potřebuje. Kompletní distribuce má přibližně 7MB, což může být také jeden z faktorů, které mluví spíše pro druhou formu distribuce.

Samozřejmě není nutné překládat PostgreSQL ze zdrojových textů, protože většina linuxových distribucí již obsahuje PostgreSQL ve svých instalačních balíčcích. Ne vždy však v poslední verzi, a proto si ukážeme, jak právě poslední verzi databázového serveru PostgreSQL nainstalovat. RPM balíky poslední verze PostgreSQL se nachází např. na FTP serveru Českého sdružení uživatelů operačního systému Linux.

Ještě před tím, než přistoupíme k vlastnímu rozbalení a překladu databázového serveru, si musíme říci něco o tom, jaké prostředí server požaduje pro svůj běh apod. Databázový server PostgreSQL je možné pro testovací účely spouštět přímo z vašeho uživatelského účtu, ale v reálném nasazení je většinou použito speciálního účtu, např. s názvem postgres. Stejného postupu se budeme držet i my v našem článku. Pro zvláštního uživatele hovoří jak bezpečnostní, tak i administrativní důvody. První fází našeho postupu tedy bude vytvoření uživatele, pod kterým náš server bude nainstalován a poběží:

```
[root@SnowWhite /root]# /usr/sbin/adduser postgres
```

Uživatel postgres je tedy vytvořen a musíme mu ještě změnit přístupové heslo, abychom pod jeho právy mohli celý server nainstalovat:

```
[root@SnowWhite /root]# passwd postgres
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
[root@SnowWhite /root]#
```

Výborně, podařilo se nám i podruhé zadat stejné heslo a máme vytvořen účet postgres, který bude určen pouze pro administraci databázového serveru. Veškeré další kroky provedeme pod identitou nového uživatele a musíme se tedy přihlásit s jeho právy. Pokud máme k dispozici administrátorská práva, můžeme použít příkaz `su`:

```
[root@SnowWhite /root]# su - postgres
[postgres@SnowWhite postgres]$
```

Nebo se můžeme přihlásit z linuxové konzoly přímo jako uživatel postgres a můžeme přistoupit k vlastní instalaci.

Věřím, že kompletní zdrojové texty již máte stažené (pokud ne, asi byste měli změnit poskytovatele připojení :-) a můžeme tedy přikročit k jejich rozbalení. PostgreSQL je standardně šířen ve formátu `.tar.gz` a jde tedy o zkomprimovaný archivní soubor. Jeho rozbalení provedeme následujícím příkazem:

```
[postgres@SnowWhite postgres]$ tar xfz postgresql-7.0.2.tar.gz
```

Tento příkaz dekomprimuje a rozbalí celý archív zdrojových textů a připraví je tím ke konfiguraci a překladu. Pokud jste si stáhli verzi rozdělenou na jednotlivé části, jednoduše je rozbalte pomocí následujících příkazů:

```
for i in base docs support test; do tar xfz postgresql-7.0.2.$i.tar.gz; done
```

Struktura zdrojových textů serveru PostgreSQL

V adresáři, do kterého se zdrojové texty PostgreSQL rozbalí (`postgresql-7.0.2`), je několik důležitých souborů a adresářů:

```
[postgres@SnowWhite postgresql-7.0.2]$ ls -l
total 133
-rw-r--r--   1 postgres postgres     1169 Jan 29 09:53 COPYRIGHT
-rw-r--r--   1 postgres postgres   110178 Jun  5 20:15 HISTORY
-rw-r--r--   1 postgres postgres    14737 Jun  5 20:15 INSTALL
-rw-r--r--   1 postgres postgres     1927 Jun  5 20:15 README
drwxr-xr-x  26 postgres postgres     1024 Jun  1 14:04 contrib
drwxr-xr-x   4 postgres postgres     1024 Jun  5 20:16 doc
-rw-r--r--   1 postgres postgres      740 Jun  5 20:15 register.txt
drwxr-xr-x  16 postgres postgres     1024 Jun  5 20:16 src
```

Soubor `COPYRIGHT` obsahuje licenci databázového serveru, která je poměrně benevolentní (není to však GNU GPL, ale BSD-like licence) a umožňuje nám používat, kopírovat, modifikovat a distribuovat PostgreSQL a jeho dokumentaci pouze za předpokladu, že obsah tohoto souboru bude součástí všech distribuovaných verzí. Soubor `HISTORY` obsahuje seznam všech změn, které byly ve zdrojových textech a dokumentaci provedeny a obsahuje také změny mezi jednotlivými verzemi databázového serveru PostgreSQL. Pokud bude uvolněna nová verze serveru, vždy si přečtěte tento soubor (alespoň jeho začátek), abyste nebyli nepříjemně překvapeni novinkami. V souboru `README` se můžeme dočíst, kde si stáhnout novou verzi a kde najít další informace. V souboru `register.txt` najdeme informaci o tom, jak se zaregistrovat (což samozřejmě není povinné). Soubor `INSTALL` obsahuje nejpodstatnější informace, které budeme potřebovat při instalaci, a proto si jej někdy v budoucnu (nejlépe před instalací) přečtěte.

V adresáři se zdrojovými texty dále najdeme tři adresáře: `contrib`, `doc` a `src`. Obsahově nejzajímavější pro nového uživatele PostgreSQL je adresář `doc`, který obsahuje až neuvěřitelně podrobnou dokumentaci, manuálové stránky, často kladené dotazy apod. Dokumentace je rozdělena do jednoduchého tutoriálu (`doc/tutorial.tar.gz`), příručky uživatele (`user.tar.gz`), příručky správce serveru (`admin.tar.gz`) a také je k dispozici manuál pro programátory (`programmer.tar.gz`). Dokumentace k databázovému serveru PostgreSQL je tedy dostatek.

Adresář `contrib` obsahuje některé zajímavé nástroje, které dokáží velmi zjednodušit práci s databázovým serverem PostgreSQL. Jeho součástí je například i popis nastavení webového serveru Apache tak, aby veškeré přístupy ke svým dokumentům zapisoval do databáze apod. Pokud vám bude něco u PostgreSQL chybět, zkuste se nejprve podívat do tohoto adresáře.

Konfigurace PostgreSQL

A konečně se dostáváme k tomu nejlepšímu, k vlastním zdrojovým textům, které jsou uloženy v adresáři `src/`. Tento adresář obsahuje známý soubor `configure`, pomocí něhož si můžeme před překladem nastavit, jaké vlastnosti budeme chtít do výsledného binárního souboru přeložit. V další verzi serveru bude již tento soubor umístěn přímo v hlavním adresáři se zdrojovými texty. Pokud spustíme příkaz `configure --help`, dozvíme se, jaké vlastnosti můžeme povolit, a které naopak zakázat. Podle toho můžeme ovlivnit jak velikost, tak i rychlost a schopnosti PostgreSQL:

```
[postgres@SnowWhite src]$ ./configure --help
Usage: configure [options] [host]
Options: [defaults in brackets after descriptions]
Configuration:
  --cache-file=FILE       cache test results in FILE
  --help                  print this message
  --no-create             do not create output files
  --quiet, --silent       do not print 'checking...' messages
  --version               print the version of autoconf that created configure
Directory and file names:
  --prefix=PREFIX         install architecture-independent files in PREFIX
                          [/usr/local/pgsql]
  --exec-prefix=EPREFIX   install architecture-dependent files in EPREFIX
```

```
                          [same as prefix]
  --bindir=DIR            user executables in DIR [EPREFIX/bin]
  --sbindir=DIR           system admin executables in DIR [EPREFIX/sbin]
  --libexecdir=DIR        program executables in DIR [EPREFIX/libexec]
  --datadir=DIR           read-only architecture-independent data in DIR
                          [PREFIX/share]
  --sysconfdir=DIR        read-only single-machine data in DIR [PREFIX/etc]
  --sharedstatedir=DIR    modifiable architecture-independent data in DIR
                          [PREFIX/com]
  --localstatedir=DIR     modifiable single-machine data in DIR [PREFIX/var]
  --libdir=DIR            object code libraries in DIR [EPREFIX/lib]
  --includedir=DIR        C header files in DIR [PREFIX/include]
  --oldincludedir=DIR     C header files for non-gcc in DIR [/usr/include]
  --infodir=DIR           info documentation in DIR [PREFIX/info]
  --mandir=DIR            man documentation in DIR [PREFIX/man]
  --srcdir=DIR            find the sources in DIR [configure dir or ..]
  --program-prefix=PREFIX prepend PREFIX to installed program names
  --program-suffix=SUFFIX append SUFFIX to installed program names
  --program-transform-name=PROGRAM
                          run sed PROGRAM on installed program names
Host type:
  --build=BUILD           configure for building on BUILD [BUILD=HOST]
  --host=HOST             configure for HOST [guessed]
  --target=TARGET         configure for TARGET [TARGET=HOST]
Features and packages:
  --disable-FEATURE       do not include FEATURE (same as --enable-FEATURE=no)
  --enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
  --with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
  --without-PACKAGE       do not use PACKAGE (same as --with-PACKAGE=no)
  --x-includes=DIR        X include files are in DIR
  --x-libraries=DIR       X library files are in DIR
--enable and --with options recognized:
  --with-template=TEMPLATE
                          use operating system template file
                              see template directory
  --with-includes=DIRS    look for header files for tcl/tk, etc in DIRS
  --with-libraries=DIRS   look for additional libraries in DIRS
  --with-libs=DIRS        alternate spelling of --with-libraries
  --enable-locale         enable locale support
  --enable-recode         enable cyrillic recode support
  --enable-multibyte      enable multibyte character support
  --with-pgport=PORTNUM   change default postmaster port
  --with-maxbackends=N    set default maximum number of server processes
  --with-tcl              build Tcl interfaces and pgtclsh
  --with-tclconfig=DIR    tclConfig.sh and tkConfig.sh are in DIR
  --with-tkconfig=DIR     tkConfig.sh is in DIR
  --with-perl             build Perl interface and plperl
  --with-odbc             build ODBC driver package
  --with-setproctitle     use setproctitle() (EXPERIMENTAL)
  --with-odbcinst=DIR     change default directory for odbcinst.ini
  --enable-cassert        enable assertion checks (for debugging)
  --with-CC=compiler      use specific C compiler
  --enable-debug          build with debugging symbols (-g)
  --with-CXX=compiler     use specific C++ compiler
  --without-CXX           prevent building C++ code
  --with-x                use the X Window System
```

Kromě standardních voleb `configure` (tedy `--prefix` apod.) máme k dispozici i volby ovlivňující schopnosti PostgreSQL lépe pracovat s národními znakovými sadami, překódovávat řetězce mezi kódováním klienta a kódováním databáze nebo podporu vícebytových znakových sad. Podrobnější dokumentaci k těmto volbám (`--enable-locale`, `--enable-recode` a `--enable-multibyte`) naleznete v souborech `README.local`, `README.mb` a `README.Charsets` v adresáři s dokumentací. Pokud přeložíme PostgreSQL s podporou locales (`--enable-locale`), bude se databázový server chovat podle toho, jaké hodnoty budou mít proměnné `LC_*` (přesněji `LC_CTYPE`, `LC_COLLATE` a `LC_MONETARY` a nikoli např. `LC_NUMERIC`) v prostředí před startem vlastního serveru. Tato volba má samozřejmě nepříznivý vliv na rychlost vlastního serveru.

Další důležitou volbou, kterou můžeme v příkazu `configure` použít je `--with-pgport`, která nám umožní změnit standardní komunikační port (5432). S databázovým serverem PostgreSQL můžeme komunikovat dvěma odlišnými způsoby. První z nich pravděpodobně využijeme při pokusné instalaci – budeme komunikovat se serverem na stejném počítači pomocí tzv. unixového socketu. Druhý způsob je vhodný pro komunikaci s databázovým serverem na jiném stroji. Tato komunikace potom probíhá přes protokol TCP/IP na portu 5432 (standardně), nebo na portu námi definovaném právě touto volbou.

Pokud budeme provozovat server pod větším zatížením, je možné, že nám v jednom okamžiku bude na server přistupovat opravdu velké množství klientů. PostgreSQL je standardně nakonfigurován tak, že spustí maximálně 32 procesů, které by tyto požadavky obsluhovaly. Pomocí volby `--with-maxbackends` můžeme toto číslo zvětšit na námi požadovanou hodnotu.

Ostatní volby jsou pro nás zatím nezajímavé a většinou definují další rozhraní, která jsou přeložena. Např. perlové rozhraní nebo ODBC ovladač.

Databázový server si tedy nakonfigurujeme tak, aby se přeložil a nainstaloval do adresáře `~/PostgreSQL` s podporou locales:

```
[postgres@SnowWhite src]$ ./configure --prefix=~/PostgreSQL --enable-locale
```

Konfigurační program si nyní ověří, že máme vše potřebné pro překlad a instalaci a vytvoří soubory pro program `make`. Pokud se tato fáze nezdaří a program `configure` skončí s chybou, doporučuji konzultovat soubor `doc/FAQ_Linux` a další soubory v adresáři `doc/`. Pokud zde nenaleznete odpověď, jistě se ji dozvíte v poštovní konferenci `databases@linux.cz`, kde je přihlášeno mnoho databázových odborníků ochotných pomoci.

Překlad

Fázi konfigurace tedy máme úspěšně za sebou a můžeme přistoupit k vlastnímu překladu, který je pravděpodobně nejjednodušší fází přípravy instalace. Překlad nastartujeme příkazem:

```
[postgres@SnowWhite src]$ make
```

A nyní již máme několik chvil např. na kávu či čaj. Překlad serveru může trvat od 2 hodin na velmi pomalých procesorech (např. 486) nebo také tři minuty, pokud je váš stroj výkonnější. Na mém testovacím stroji trvá překlad celého serveru asi čtyři minuty, ale na vašem počítači to bude pravděpodobně trvat déle. Pro kompilaci serveru je nutné mít na disku alespoň 35MB místa. Pokud je proces překladu ukončen úspěšně, uvidíme na terminálu následující text:

```
make[3]: Leaving directory ‘/home/postgres/postgresql-7.0.2/src/pl/plpgsql/src’
make[2]: Leaving directory ‘/home/postgres/postgresql-7.0.2/src/pl/plpgsql’
make[1]: Leaving directory ‘/home/postgres/postgresql-7.0.2/src/pl’
All of PostgreSQL is successfully made. Ready to install.
```

Budeme-li si chtít ověřit, zda je PostgreSQL správně přeložen a funguje tedy přesně podle předpokladu vývojářů, můžeme provést tzv. „regresní testy“. Více viz dokumentace k těmto testům v souboru `README` v adresáři `src/test/regress` ve zdrojových textech.

Instalace

A můžeme přistoupit k vlastní instalaci, která je stejně jednoduchá jako překlad databázového serveru PostgreSQL, stačí pouze zadat příkaz `make install`:

```
[postgres@SnowWhite src]$ make install
...
```

```
Thank you for choosing PostgreSQL, the most advanced open source database
engine.
```

V této chvíli máme databázový server nainstalován v adresáři `~/PostgreSQL` a můžeme s ním začít pracovat. Kompletní instalace serveru včetně klienta a vývojových knihoven zabírá přibližně 2.5MB diskového prostoru a je tedy velmi nenáročná.

Budeme-li chtít nainstalovat i dokumentaci, musíme provést ještě následující příkazy:

```
[postgres@SnowWhite src]$ cd ../doc
[postgres@SnowWhite doc]$ make install
```

Kompletní dokumentace je připravena na instalaci příkazem `make install` v adresáři `doc` v kořeni zdrojových textů. PostgreSQL umožňuje nainstalovat pouze dokumentaci v podobě manuálových stránek pomocí příkazu `make man`. Manuálové stránky jsou potom nainstalovány do adresáře `~/PostgreSQL/man`.

Inicializace

Databázový server PostgreSQL potřebuje své databáze ukládat ve formě souborů na disku, musíme mu tudíž připravit prostor, kam bude tyto soubory ukládat. Pro zjednodušení další práce umožňuje PostgreSQL pojmenování tohoto „úložného prostoru“ pomocí proměnné prostředí `PGDATA`, a proto si tuto proměnnou nadefinujeme v souboru `~/.bashrc` (za předpokladu, že používáme příkazový interpret Bourne Again Shell), abychom ji měli vždy automaticky k dispozici. Taktéž si do tohoto souboru musíme uložit definici cest k binárním programům databázového serveru PostgreSQL a cestu k jeho knihovnám. V souboru `~/.bashrc` tedy budeme mít následující záznamy:

```
#
# Konfigurace DB serveru PostgreSQL
#
export PGDATA=~/PostgreSQL/data
export PATH=~/PostgreSQL/bin/:$PATH
export LD_LIBRARY_PATH=~/PostgreSQL/lib
```

Po ukončení a novém přihlášení tyto změny nabudou platnosti a my můžeme inicializovat úložný prostor serveru pomocí příkazu `initdb`:

```
[postgres@SnowWhite postgres]$ initdb
This database system will be initialized with username "postgres".
This user will own all the data files and must also own the server process.

Creating database system directory /home/postgres/PostgreSQL/data
Creating database system directory /home/postgres/PostgreSQL/data/base
Creating database XLOG directory /home/postgres/PostgreSQL/data/pg_xlog
Creating template database in /home/postgres/PostgreSQL/data/base/template1
Creating global relations in /home/postgres/PostgreSQL/data/base
Adding template1 database to pg_database

Creating view pg_user.
Creating view pg_rules.
Creating view pg_views.
Creating view pg_tables.
Creating view pg_indexes.
Loading pg_description.
Vacuuming database.

Success. You can now start the database server using:

        /home/postgres/PostgreSQL/bin//postmaster -D \
                       /home/postgres/PostgreSQL/data
or
        /home/postgres/PostgreSQL/bin//pg_ctl -D \
```

```
/home/postgres/PostgreSQL/data start
```

Pokud dojde k nějaké chybě, je možné použít u příkazu `initdb` parametr `--debug`, který vypisuje ladící informace, nebo nezdokumentovanou volbu `--show`, která zobrazí aktuální nastavení proměnných ovlivňujících funkci programu `initdb`. Další parametry, jež je možné příkazu `initdb` zadat, je možné získat pomocí přepínače `--help`.

Skript `initdb` nám připravil kompletně prostředí pro databáze a na konci své činnosti nám dokonce ukázal, jak spustit databázový server. Ovšem tak jednoduché to zase není. Kdybychom nyní jeden z těchto příkazů poslechli, tak by běžící databázový server nepodporoval české třídění a to by nás příliš neuspokojilo. Řešení tohoto problému tkví v nastavení locales na vhodnou hodnotu:

```
[postgres@SnowWhite postgres]$ export LC_ALL=cs_CZ
```

Spuštění serveru

Nyní již můžeme PostgreSQL spustit i s podporou českého třídění:

```
[postgres@SnowWhite postgres]$ postmaster
DEBUG:  Data Base System is starting up at Thu Jun 15 22:14:38 2000
DEBUG:  Data Base System was shut down at Thu Jun 15 22:10:19 2000
DEBUG:  Data Base System is in production state at Thu Jun 15 22:14:39 2000
```

A server běží. Jak jednoduché. Skript `initdb` nám ukázal složitější, ale naprosto ekvivalentní způsob jak spustit vlastní server `postmaster`. Parametr `-d` a následující řetězec udávají místo, kde je vlastní databáze uložena. My máme cestu k databázi uloženu v proměnné prostředí `PGDATA`, a proto tento parametr nemusíme zadávat.

Databázový server PostgreSQL, pokud je spuštěn příkazem `postmaster` bez parametrů, očekává komunikaci přes unixový socket `/tmp/.s.PGSQL.5432`, kde číslo 5432 je možné změnit volbou `--with-pgport` při konfiguraci zdrojových textů (viz výše). Pokud bychom chtěli, aby byl celý databázový server přístupný i ze sítě, musíme při startu programu `postmaster` použít navíc volbu `-i`. Popis síťového provozu PostgreSQL je již však nad rámec tohoto článku a jistě se mu budeme později věnovat.

Příkaz `postmaster` akceptuje i další parametry, které mohou být velmi důležité pro optimalizaci výkonu serveru. Např. již výše zmíněný maximální počet procesů, který je standardně nastaven na 32, je možné zvýšit pomocí volby `-N číslo`, kde číslo je omezeno pouze schopnostmi vašeho hardwaru a konstantou 1024. Množství ladících informací, které `postmaster` produkuje, je možné ovlivnit volbou `-d`. Volba `-B` má zase vliv na počet alokovaných sdílených bloků paměti.

Program `postmaster` jako takový je pouze jakýsi správce starající se o vlastní databázové dříče (v anglické terminologii se používá pojmu backend), tedy o procesy, které vykonávají vlastní dotazy. Pokud chceme, aby `postmaster` spouštěl jiný backend, můžeme použít parametru `-b`. `postmaster` předává každému novému backendu volby, které obdrží při startu (viz volba `-o` u programu `postmaster`). Tak tedy můžeme ještě dále ovlivnit chování jednotlivých backendů. Budeme-li tedy chtít spustit každý backend tak, aby po každém zápisu nečekal na dokončení fyzického zápisu na disk (`fsync()` – viz soubor `doc/README.fsync`), ale aby se spokojil s uložením do bufferů operačního systému (pozor na výpadky elektrického proudu, v takovém případě je vhodné použít UPS, ale výkon je nesrovnatelně lepší), přidáme do argumentu volby `-o` parametr `-F`:

```
[postgres@SnowWhite postgres]$ postmaster -o '-F'
```

Budeme-li navíc chtít zvětšit velikost paměti určené pro třídění, použijeme volbu `-S`:

```
[postgres@SnowWhite postgres]$ postmaster -o '-F -S4096'
```

Velikost je uvedena v kB, a proto tento příkaz umožní každému novému backendu použít až 4MB paměti pro třídění. Samozřejmě čím více paměti je k dispozici, tím je třídění rychlejší (zvláště u velkých objemů dat).

Pro další optimalizaci výkonu je vhodné si povšimnout dalších voleb, které nám ukazují statistiky a časovou náročnost jednotlivých dotazů (viz volby `-s` a `-t`).

Pro start databázového serveru je možné použít i skript `pg_ctl`, který je velmi podobný startovacímu skriptu webového serveru Apache (`apachectl`). Umožňuje nastartovat, zastavit, restartovat i zjistit stav databázového serveru.

Vytvoření pokusné databáze

Nyní nám běží samotný databázový server, který má inicializován „úložný prostor“ pro databáze. Musíme tedy ještě tento prostor něčím zaplnit. PostgreSQL do tohoto úložného prostoru ukládá databáze. Vytvoříme si pokusnou databázi s názvem ‘test’, na které budeme demonstrovat základní schopnosti PostgreSQL. Databázi je možné vytvořit příkazem `createdb`:

```
[postgres@SnowWhite postgres]$ createdb test
CREATE DATABASE
```

V tomto okamžiku PostgreSQL vytvořil novou databázi podle šablony s názvem ‘template1’. Pokud dostaneme po zadání tohoto příkazu odpověď `CREATE DATABASE`, je vše v pořádku a naše pokusná databáze je vytvořena. Dalším obvyklým krokem je vytvoření databázových uživatelů, ale tomuto kroku se budeme věnovat v budoucnu.

Nyní se k naší databázi již konečně můžeme připojit pomocí řádkového klienta `psql`:

```
[postgres@SnowWhite postgres]$ psql test
Welcome to psql, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

test=#
```

Poté můžeme vkládat SQL příkazy, vytvářet tabulky apod. Vyzkoušíme si, zda je náš server připraven na české třídění na velmi jednoduchém příkladu. Vytvoříme tabulku se jménem ‘czech’ v databázi ‘test’ s jedním polem (‘jmeno’), které bude obsahovat řetězec znaků:

```
test=# CREATE TABLE czech (jmeno TEXT);
CREATE
```

Naplnění pokusné databáze daty

Tuto tabulku naplníme znaky české abecedy (pokud možno s háčky a čárkami). Chceme-li ovšem mít možnost zadat tyto znaky pomocí řádkového klienta, který je přeložen s knihovnou readline, musíme ji správně nastavit. Podrobnější informace jsou v dokumentu Czech-HOWTO, pokud však jste netrpělivý, vytvořte si soubor `~/.inputrc` s následujícím obsahem:

```
set meta-flag on
set convert-meta off
set output-meta on
```

a spusťte znovu řádkového klienta. Nyní již můžeme zadávat nové záznamy do tabulky:

```
test=# INSERT INTO czech VALUES('a');
INSERT 18825 1
test=# INSERT INTO czech VALUES('á');
INSERT 18826 1
test=# INSERT INTO czech VALUES('b');
INSERT 18827 1
test=# INSERT INTO czech VALUES('c');
INSERT 18828 1
test=# INSERT INTO czech VALUES('č');
INSERT 18829 1
test=# INSERT INTO czech VALUES('d');
INSERT 18830 1
test=# INSERT INTO czech VALUES('e');
INSERT 18831 1
```

```
test=# INSERT INTO czech VALUES('f');
INSERT 18832 1
test=# INSERT INTO czech VALUES('g');
INSERT 18833 1
test=# INSERT INTO czech VALUES('h');
INSERT 18834 1
test=# INSERT INTO czech VALUES('i');
INSERT 18835 1
test=# INSERT INTO czech VALUES('í');
INSERT 18836 1
test=# INSERT INTO czech VALUES('z');
INSERT 18837 1
test=# INSERT INTO czech VALUES('š');
INSERT 18838 1
test=# INSERT INTO czech VALUES('s');
INSERT 18839 1
test=# INSERT INTO czech VALUES('ž');
INSERT 18840 1
```

Výborně, databázový server přijal všechny naše požadavky a můžeme začít s našimi dotazy. Věřím, že základy jazyka SQL nejsou našemu čtenáři cizí, a proto položíme databázovému serveru pouze jediný dotaz: „Ukaž nám seznam jmen v tabulce 'czech' seřazený abecedně podle jména (položky 'jmeno')". V jazyce SQL se tento dotaz zapíše následujícím způsobem:

```
test=# SELECT jmeno FROM czech ORDER BY jmeno;
 jmeno
-------
 a
 á
 b
 c
 č
 d
 e
 f
 g
 h
 i
 í
 s
 š
 z
 ž
(16 rows)
```

A opravdu, PostgreSQL nám seznam dokázal seřadit podle abecedního pořádku a je tedy připraven pro nasazení v databázových aplikacích kladoucích velké nároky např. na podporu českého jazyka. Samozřejmostí také je, že SQL funkce jako UPPER fungují také korektně, a proto např. můžeme tuto oblast funkcí přenechat přímo na databázovém serveru a nestarat se o ni až ve vlastní aplikaci.

To ale není vše, co nám databázový server PostgreSQL nabízí pro podporu českého jazyka. Další vlastností, kterou oceníme zejména v multiplatformním prostředí je možnost překódování mezi klientem a serverem, a to pro každého klienta zvlášť. Pokud např. první klient pracuje se stejným kódováním, jako naše databáze, může pracovat bez problémů. Pokud však druhý klient pracuje v jiném kódování, může si pomocí příkazu `SET CLIENT_ENCODING` požádat o překódování veškeré komunikace (podrobnější dokumentace je v souboru `doc/README.mb`).

Další odkazy a dokumentace

Dokumentace k databázovému serveru PostgreSQL je velmi podrobná a je součástí zdrojových textů.

Většinu odpovědí na své otázky tedy naleznete v adresáři `doc` zdrojových textů. Pokud i přesto odpověď nenaleznete, můžete využít buď české konference zaměřené na databáze (databases@linux.cz) nebo konferencí projektu PostgreSQL (viz webová stránka projektu PostgreSQL).

Shrnutí

V článku je podrobně popsána instalace databázového serveru PostgreSQL verze 7.0.2 s podporou českého třídění. Článek je podán velmi srozumitelně a jeho hlavním cílem je rozšířit obec českých uživatelů systému PostgreSQL. Snad se to autorovi povedlo.

Poděkování

Autor by chtěl poděkovat Karlu Žákovi za pomoc při tvorbě tohoto článku.

O autorovi

Autor je nezávislým konzultantem v oboru informačních technologií, specializuje se na Linux, unixové operační systémy a programování Open Source projektů.