PostgreSQL 7.2.
Ćwiczenia praktyczne

Autor: Marcin Szeliga
ISBN: 83-7197-866-9
Format: B5, stron: 150

Jeśli kiedykolwiek chciałeś uruchomić bazę danych PostreSQL, a nie wiedziałeś jak lub jeśli chciałeś stworzyć bazę za pomocą tego narzędzia, a ciągle wydaje ci się, że technologia ta przypomina „czarną skrzynkę" – powinieneś przeczytać tę książkę.

Autor, Marcin Szeliga – wieloletni praktyk, twórca i administrator baz danych, certyfikowany inżynier Microsoft – podjął się zadania trudnego. Postanowił w sposób przystępny opisać instalację oraz projektowanie bazy postgresowej. Wielka pasja, jaką dla niego są język SQL oraz bazy danych i umiejętność przekazywania informacji, nawet najbardziej zawiłych – w prosty sposób, zaowocowały doskonałym podręcznikiem dla początkujących. Wykorzystaj szansę i naucz się:

- instalacji systemu PostgreSQL w środowiskach Linux i Windows;
- niezbędnych czynności składających się na codzienną pracę administratora baz danych;
- teorii relacyjnych baz danych. Zdobyte umiejętności zostaną wykorzystane przy tworzeniu przykładowej bazy danych;
- strukturalnego języka zapytań – języka zarządzania wszystkimi relacyjnymi bazami danych;
- tworzenia zaawansowanych instrukcji języka SQL;
- metod zdalnego dostępu do bazy danych poprzez dobrze znane aplikacje działające w środowisku Windows – takie jak Microsoft Access czy Microsoft Excel;
- zarządzania PostgreSQL-em z poziomu systemu Windows.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 2.

Administracja SZBD PostgreSQL

Konfiguracja środowiska SZBD PostgreSQL

Podczas uruchamiania *SZBD PostgreSQL* odczytuje zawartość plików konfiguracyjnych znajdujących się w katalogu */usr/local/pgsql/data/postgresql.conf* oraz *pg_hba.conf*. Najprostszą i najbardziej uniwersalną metodą konfiguracji środowiska systemu *Postgres* jest edycja tych plików.

Niektóre opcje mogą zostać ustawione poprzez połączenie się z działającą bazą danych i wykonanie instrukcji *SQL* `SET opcja wartość`.

Umożliwienie dostępu klientom zdalnym do baz danych

Ćwiczenie 2.1.

Domyślnie *PostgreSQL* nie zezwoli użytkownikom na zdalny dostęp do jakiejkolwiek bazy danych. Aby nadać im prawa do nawiązania połączenia, należy zmienić plik konfiguracyjny *pg_hba.conf*. Plik ten znajduje się w katalogu */usr/local/pgsql/data* i zawiera wpisy dotyczące udzielenia lub odebrania praw dla zdalnych użytkowników do połączenia się z bazą danych. Aby zezwolić na dostęp do wszystkich baz danych wszystkim komputerom z sieci lokalnej bez konieczności podawania hasła:

1. Za pomocą dowolnego edytora tekstu otwórz plik *pg_hba.conf*.

Wiersze rozpoczynające się od znaku # są komentarzami i nie wpływają na konfigurację *SZBD PostgreSQL*.

2. Dodaj dodatkowy wiersz na końcu pliku:

```
host all  192.168.0.0 255.255.0.0 trust
```

3. Zapisz plik.
4. Zatrzymaj i ponownie uruchom proces `postmaster`.
5. Od tej chwili, wszystkie komputery o adresach *IP* rozpoczynających się od *192.168.* będą mogły uzyskać dostęp, bez podawania hasła użytkownika, do wszystkich baz danych.

Uwierzytelnianie klientów za pomocą hasła

Ćwiczenie 2.2.

O ile połączenia klientów sieci lokalnej mogą być akceptowane bez konieczności uwierzytelniania użytkownika, o tyle klienci sieci zewnętrznej powinni przy próbie połączenia się z bazą danych podać prawidłową nazwę użytkownika i hasło. W tym celu:

1. Za pomocą dowolnego edytora tekstu otwórz plik *pg_hba.conf*.

Programy systemu *UNIX* takie jak *Postgres* niepoprawnie interpretują znak końca wiersza plików tekstowych utworzonych lub edytowanych za pomocą programów systemu *Windows*, takich jak np. *Notatnik*.

2. Dodaj dodatkowy wiersz na końcu pliku:

```
host all  0.0.0.0 0.0.0.0 md5
```

3. Zapisz plik.
4. Zatrzymaj i ponownie uruchom proces `postmaster`.
5. Od tej chwili, wszystkie komputery będą mogły uzyskać dostęp do wszystkich baz danych, o ile zostanie podane prawidłowa nazwa użytkownika i hasło. Hasła użytkowników są przechowywane w pliku *pg_pwd* znajdującym się w katalogu */pgsql/data/global* w postaci zaszyfrowanej algorytmem *MD5*.

Odrzucanie połączeń określonych klientów

Ćwiczenie 2.3.

Jeżeli chcemy uniemożliwić dostęp do wybranej bazy danych klientom o określonych adresach *IP*:

1. Za pomocą dowolnego edytora tekstu otwórz plik *pg_hba.conf*.
2. Dodaj dodatkowy wiersz przed wierszem zezwalającym na połączenie każdemu komputerowi:

```
host template1  217.96.0.0 255.255.0.0 reject
```

3. Zapisz plik.

4. Zatrzymaj i ponownie uruchom proces `postmaster`.
5. Od tej chwili próby nawiązania połączenia z systemową bazą danych przez komputery o adresach *IP 217.96.x.x* będą automatycznie odrzucane.

Listing 2.1. *Przykładowy plik pg_hba.conf*

```
# Put your actual configuration here
# ==================================
#
# TYPE     DATABASE    IP_ADDRESS    MASK              AUTH_TYPE  AUTH_ARGUMENT

local      all                                         trust
host       all         127.0.0.1       255.255.255.255   trust
host       all         192.168.0.0     255.255.0.0       trust
host       template1    217.96.0.0     255.255.0.0       reject
host       all         0.0.0.0         0.0.0.0           md5
```

Optymalizacja pracy SZBD PostgreSQL

Ćwiczenie 2.4.

Dane przechowywane w każdej bazie danych nie są niezmienne — codziennie dodawane są nowe rekordy, niektóre rekordy są zmieniane, a niektóre informacje usuwane.

W rezultacie, aby zapewnić wysoką wydajność *SZBD* należy okresowo „oczyścić" bazę danych, aby odzyskać wolne miejsca w bazie danych pozostałe po usuniętych rekordach i aby zaktualizować statystyki *optymalizatora poleceń SQL*[1]. W tym celu:

1. Zaloguj się na konto użytkownika *`postgres`*.
2. Wydaj polecenie:

   ```
   /usr/local/pgsql/bin/vacuumdb -d template1 --analyze --verbose
   ```

3. Jeżeli zostaną wyświetlone informacje podobne do pokazanych na rysunku 2.1, a wykonanie polecenia zostało potwierdzone poprzez wyświetlenie tekstu `VACUUM` — statystyki wybranej bazy danych zostały odświeżone.
4. Wyświetlenie komunikatu *vacuumdb: vacuum failed* oznacza błąd, prawdopodobnie spowodowany niemożliwością połączenia z wybraną bazą danych. Sprawdź, czy proces `postmaster` jest uruchomiony, czy w katalogu baz danych znajduje się baza o podanej nazwie i czy użytkownik *`postgres`* jest administratorem tej bazy. Za pomocą dodatkowych opcji polecenia `vacuumdb` możesz określić:
   - nazwę lub adres *IP* serwera: `-h` *`serwer`*,
   - port, na którym *SZBD* nasłuchuje połączeń klienckich: `-p` *`port`*.

[1] Optymalizator, na podstawie zgromadzonych statystyk bazy danych, decyduje o sposobie wykonania instrukcji *SQL*.

Rysunek 2.1. *Fragment informacji wyświetlanych podczas aktualizacji statystyk*

Monitorowanie pracy serwera

Ćwiczenie 2.5.

Z reguły warto zapisywać informacje o operacjach przeprowadzanych przez poszczególnych użytkowników oraz informacje diagnostyczne opisujące pracę *SZBD PostgreSQL*. Tak jak wiele innych opcji, tą również możemy ustawić modyfikując zawartość pliku *postgresql.conf*.

1. Za pomocą dowolnego edytora tekstu otwórz plik *postgresql.conf*.
2. Znajdź parametr `debug_level` (domyślną wartością jest `0` — w dzienniku zapisywane są tylko podstawowe dane opisujące pracę serwera).
3. Maksymalny poziom szczegółowości zapisywanych danych osiągniemy zmieniając wartość parametru `debug_level` na `4`:

```
debug_level=4
```

4. Ustawiając wartość parametru `log_connections` na `Yes` dodamy do dziennika informacje o każdej udanej próbie zalogowania klienta do serwera:

```
log_connections=yes
```

5. Zmieniając wartość parametru `log_timestamp` na `Yes` spowodujemy, że każda informacja zapisana w dzienniku zdarzeń będzie uzupełniona o *znacznik czasu* informujący o dokładnym czasie wystąpienia zdarzenia:

```
log_timestamp=yes
```

6. Zapisz plik.
7. Zatrzymaj i ponownie uruchom proces `postmaster`.

Listing 2.2. *Fragment pliku postgresql.conf*

```
#
#    Connection Parameters
#
tcpip_socket = false
ssl = false
max_connections = 16
```

```
port = 5432
hostname_lookup = false
show_source_port = false

#unix_socket_directory = ''
#unix_socket_group = ''
#unix_socket_permissions = 0777

#virtual_host = ''

#krb_server_keyfile = ''

#
#    Shared Memory Size
#
shared_buffers = 64        # 2*max_connections, min 16
max_fsm_relations = 100    # min 10, fsm is free space map
max_fsm_pages = 10000      # min 1000, fsm is free space map
max_locks_per_transaction = 64 # min 10
wal_buffers = 8            # min 4

#
#    Non-shared Memory Sizes
#
sort_mem = 512             # min 32
vacuum_mem = 8192          # min 1024
```

Przydzielanie zasobów systemowych

Ćwiczenie 2.6.

W przypadku serwera, który będzie obsługiwał wiele jednoczesnych połączeń lub który będzie nasłuchiwał na nietypowym porcie, należy zmienić domyślne ustawienia *SZBD PostgreSQL*. W tym celu:

1. Za pomocą dowolnego edytora tekstu otwórz plik *postgresql.conf*.
2. Znajdź parametr `port` i zmień domyślną wartość na numer portu, który będzie wykorzystywany do nawiązywania połączenia klientów z serwerem:

   ```
   port=20111
   ```

3. Ustaw wartość parametru `max_connections` — serwer będzie zezwalał na nawiązanie maksymalnie 50 jednoczesnych połączeń:

   ```
   max_connections=50
   ```

4. Ustawiając wartość parametru `shared_buffer` określamy, ile buforów pamięci operacyjnej będzie zarezerwowane dla *SZBD Postgres* (domyślna wielkość bufora to 8 192 bajty):

   ```
   shared_buffer=128
   ```

5. Przydziel ilość pamięci operacyjnej (w kilobajtach) wykorzystywanej m.in. przy sortowaniu danych przez system *Postgres*:

   ```
   sort_mem=2048
   ```

Niektóre skomplikowane zapytania lub operacje sortowania mogą być wykonywane równolegle. W takim przypadku każdemu procesowi zostanie przydzielona podana ilość pamięci operacyjnej. W praktyce całkowita liczba przydzielonej pamięci jest wielokrotnością wartości parametru `sort_mem`.

6. Zapisz plik.
7. Zatrzymaj i ponownie uruchom proces *postmaster*.

Zarządzanie bazami danych

Po uruchomieniu i skonfigurowaniu środowiska *Systemu zarządzania bazami danych* możemy utworzyć testową bazę danych. Chociaż ćwiczenia znajdujące się w kolejnym rozdziale przedstawiają proces samodzielnego zaprojektowania bazy danych *katalog*, to już teraz możemy utworzyć swoją pierwszą bazę danych. Wykonując kolejne ćwiczenia Czytelnik pozna nie tylko metody tworzenia i usuwania poszczególnych baz danych, ale również ograniczania dostępu do obiektów baz danych poszczególnym użytkownikom.

Tworzenie bazy danych

Ćwiczenie 2.7.

Jednym z etapów instalacji i konfiguracji *SZBD PostgreSQL* było utworzenie systemowej bazy danych *template1*. W tej bazie przechowywane są informacje niezbędne do zarządzania bazami danych użytkownika. *SZBD PostgreSQL*, tak jak zdecydowana większość innych systemów zarządzania bazami danych, umożliwia uruchomienie na pojedynczym serwerze dowolnej liczby baz danych.

Książka opisuje wyłącznie logiczną strukturę baz danych *SZBD PostgreSQL*. Sposób, w jaki poszczególne obiekty i dane przechowywane są na dysku twardym został opisany w książce *„Bazy danych i PostgreSQL. Od podstaw”*.

Aby utworzyć nową bazę danych z poziomu systemu operacyjnego:

1. Upewnij się, że *SZBD PostgreSQL* został pomyślnie uruchomiony.
2. Zaloguj się na konto użytkownika `postgres` (W przypadku zainstalowania programu w środowisku *Microsoft Windows* należy zalogować się do systemu jako użytkownik `postgres` lub zmodyfikować opcję `–login` w pliku uruchamiającym powłokę *Cygwin Bash c:\cygwin\cygwin.bat*).

   ```
   su - postgres
   ```

3. Utwórz bazę danych `test` (rysunek 2.2):

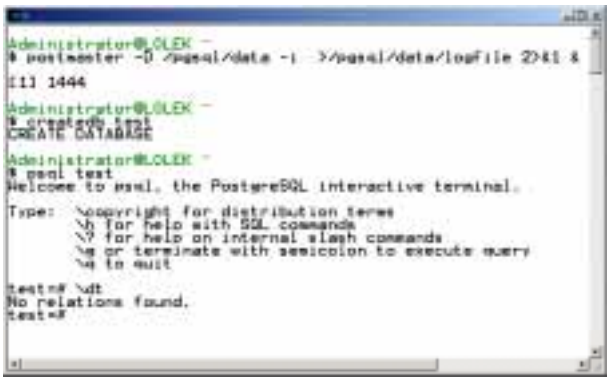
   ```
   /usr/local/pgsql/bin/createdb test
   ```

4. Po chwili zostanie wyświetlony komunikat informujący o pomyślnym utworzeniu bazy danych (rysunek 2.4):

   ```
   CREATE DATABASE
   ```

5. Sprawdź, czy możesz podłączyć się do nowo utworzonej bazy za pomocą narzędzia `psql` (rysunek 2.2):

```
/usr/local/pgsql/bin/psql test
```



Rysunek 2.2. *Utworzona baza danych nie zawiera żadnych obiektów*

Alternatywną metodą utworzenia bazy danych jest:

1. Zalogowanie do systemowej bazy danych *template1*:

```
psql template1
```

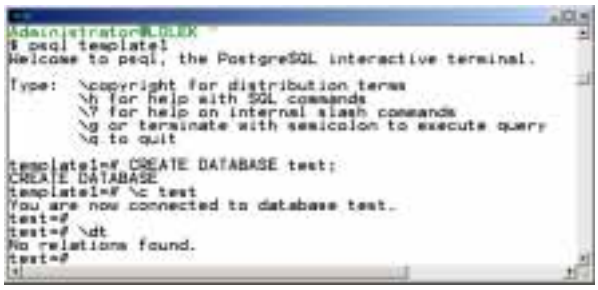
2. Wykonanie instrukcji *SQL* `CREATE DATABASE test;`

3. Po chwili zostanie wyświetlony komunikat informujący o pomyślnym utworzeniu bazy danych (rysunek 2.3):

```
CREATE DATABASE
```

4. Sprawdź, czy możesz podłączyć się do nowo utworzonej bazy:

```
\c test
```



Rysunek 2.3. *Do przełączania się pomiędzy bazami danych służy polecenie \c nazwa bazy danych*

Tworzenie kopii zapasowej

Ćwiczenie 2.8.

Tworzenie kopii zapasowych jest jednym z podstawowych obowiązków administratora bazy danych. Niezależnie od tego, ile wydano pieniędzy na niezawodny sprzęt, zawsze istnieje możliwość jego awarii i utraty danych. Ponadto, niedoświadczony użytkownik może usunąć lub zmodyfikować wszystkie dane przechowywane w pewnej tabeli niezgodnie ze

swoimi zamierzeniami. Wreszcie, każdy nawet najlepiej zabezpieczony system po podłączaniu do internetu może stać się przedmiotem udanego ataku pirata komputerowego. W każdej z tych sytuacji najszybszą (a czasami jedyną) metodą ponownego uruchomienia bazy danych jest jej odtworzenie z aktualnej kopii zapasowej.

SZBD PostreSQL umożliwia tworzenie logicznych kopii zapasowych baz danych bez konieczności wyłączania systemu. Służą do tego polecenia `pg_dump` oraz `pg_dumpall`.

Najprostszym sposobem wykonania kopii zapasowej bazy danych jest wydanie polecenia `pg_dump`:

1. Upewnij się, że *SZBD PostgreSQL* został pomyślnie uruchomiony.
2. Zaloguj się na konto użytkownika *postgres*:

```
su - postgres
```

3. Utwórz kopię zapasową wybranej bazy danych:

```
pg_dump test > test.bak
```

4. W katalogu domowym użytkownika, który wykonał kopię bazy danych utworzony został plik *test.bak*.

Działanie narzędzia `pg_dump` polega na utworzeniu skryptu złożonego z poleceń *SQL* oraz wewnętrznych poleceń programu `psql`, które po wykonaniu odtworzą całość bazy danych. Skrypt wynikowy jest zapisany w postaci tekstowej.

Oprócz wykonywania kopii bezpieczeństwa baz danych użytkownika należy zabezpieczyć się na wypadek uszkodzenia systemowej bazy danych *template1*. Za pomocą polecenia `pg_dumpall` można wykonać kopię zapasową wszystkich baz danych w całej instalacji (włącznie z bazą systemową i tabelami systemowymi). Operację tę może przeprowadzić jedynie administrator *SZBD PostgreSQL* (a nie administrator poszczególnych baz danych):

1. Upewnij się, że *SZBD PostgreSQL* został pomyślnie uruchomiony.
2. Zaloguj się na konto użytkownika *postgres*:

```
su - postgres
```

3. Utwórz kopię wszystkich baz danych utworzonych na serwerze:

```
pg_dumpall > serwer.bak
```

4. Zostaną wyświetlone informacje potwierdzające wykonanie kopii wszystkich baz danych, włącznie z bazą systemową:

```
connected to template1...
dumping database "template1"...
dumping database "test"...
dumping database "una"...
```

5. W katalogu domowym użytkownika, który wykonał kopię bazy danych utworzony został plik *serwer.bak*.

Tylko za pomocą narzędzia `pg_dumpall` wykonywana jest kopia metadanych *SZBD*, takich jak np. dane o użytkownikach.

Listing 2.3. *Skrypt kopii zapasowej serwera zawiera definicje użytkowników i tabel systemowych*

```
--
-- pg_dumpall (7.2)
--
\connect template1
DELETE FROM pg_shadow WHERE usesysid <> (SELECT datdba FROM pg_database WHERE datname
= 'template0');

CREATE USER "szel" WITH SYSID 101;
CREATE USER "postgres" WITH SYSID 100 PASSWORD 'postgres';

.
.
.

-- TOC Entry ID 83 (OID 16557)
--
-- Name: pga_queries Type: TABLE Owner: Administrator
--

CREATE TABLE "pga_queries" (
     "queryname" character varying(64),
     "querytype" character(1),
     "querycommand" text,
     "querytables" text,
     "querylinks" text,
     "queryresults" text,
     "querycomments" text
);
.
.
.
```

Odtwarzanie bazy danych z kopii zapasowej

Ćwiczenie 2.9.

Skoro skrypt kopii zapasowej zawiera polecenia tworzenia obiektów baz danych i wypełniania ich danymi, odtwarzanie bazy sprowadza się do jego wykonania.

Domyślnie, w skrypcie nie zostaje zapisana instrukcja tworząca bazę danych. Dlatego przed przywróceniem bazy danych na innym serwerze musimy ręcznie utworzyć pustą bazę danych. Nazwa nowo utworzonej bazy danych nie musi odpowiadać nazwie źródłowej bazy danych.

1. Upewnij się, że *SZBD PostgreSQL* został pomyślnie uruchomiony.
2. Zaloguj się na konto użytkownika *postgres*:

   ```
   su - postgres
   ```

3. Utwórz bazę danych, w której zostaną umieszczone dane z kopii zapasowej:

   ```
   createdb chwilowa
   ```

4. Uruchom skrypt kopii zapasowej:

   ```
   psql -f /home/postgres/test.bak chwilowa
   ```

Opcja `-f` powoduje, że program `psql` wykona instrukcje znajdujące się w pliku o podanej nazwie.

Odtwarzanie kopii zapasowej całego serwera wymaga połączenia z bazą danych *template1*. Ponieważ pełna kopia zapasowa zawiera instrukcje *SQL* tworzące wszystkie bazy danych użytkowników, w tym przypadku nie jest konieczne ręczne tworzenie pustych baz danych.

5. Upewnij się, że *SZBD PostgreSQL* został pomyślnie uruchomiony.
6. Zaloguj się na konto użytkownika *postgres*:

```
su - postgres
```

7. Uruchom skrypt kopii zapasowej serwera:

```
psql -f /home/postgres/serwer.bak template1
```

Usuwanie bazy danych

Ćwiczenie 2.10.

Czasami administrator musi usunąć całą bazę danych z serwera. Sytuacja taka może mieć miejsce w przypadku odtwarzania bazy danych z kopii zapasowej lub usuwania poprzedniej wersji modyfikowanej bazy danych. Aby usunąć wybrana bazę danych:

Usunięcie bazy danych powoduje nieodwracalne utracenie wszystkich przechowywanych w niej danych.

1. Upewnij się, że *SZBD PostgreSQL* został pomyślnie uruchomiony.
2. Zaloguj się na konto użytkownika *postgres*:

```
su - postgres
```

3. Usuń bazę danych *chwilowa*:

```
/usr/local/pgsql/bin/dropdb chwilowa
```

4. Na ekranie pojawi się komunikat informujący o pomyślnym usunięciu bazy danych:

```
DROP DATABASE
```

Tak jak w przypadku tworzenia bazy danych, możliwe jest jej usunięcie z poziomu języka *SQL*:

1. Zaloguj się do systemowej bazy danych *template1*:

```
psql template1
```

2. Wykonaj instrukcję *SQL* `DROP DATABASE chwilowa;`.
3. Na ekranie pojawi się komunikat informujący o pomyślnym usunięciu bazy danych:

```
DROP DATABASE
```

Tworzenie użytkowników

Ćwiczenie 2.11.

Po utworzeniu wzorcowej bazy danych *template1* jedynym użytkownikiem, który może nawiązać połączenie z jakąkolwiek bazą jest użytkownik *postgres*. Ponieważ użytkownik ten ma nieograniczone prawa modyfikowania zarówno danych, jak i obiektów baz danych, użytkownicy nie powinni mieć możliwość łączenia się z serwerem za pomocą tego konta.

Dane o użytkownikach są zapisane w tabeli systemowej `pg_user`. Wybierając wszystkie wiersze z tabeli `pg_user`, możemy wyświetlić informacje o wszystkich użytkownikach:

```
   SELECT * FROM PG_USER;
   usename    | usesysid | usecreatedb | usetrace | usesuper | usecatupd |  passwd  | valuntil
--------------+----------+-------------+----------+----------+-----------+----------+-------
 Administrator |        1 | t           | t        | t        | t         | ******** |
 szel          |      101 | f           | f        |          | f         | ******** |
 postgres      |      100 | f           | f        | f        | f         | ******** |
 (3 rows)
```

Nazwa użytkownika *SZBD PostgreSQL* nie musi odpowiadać nazwie logowania danego użytkownika do systemu operacyjnego. Jednak większość narzędzi (np. program `psql`) za domyślną nazwę użytkownika bazy danych przyjmują nazwę użytkownika, który uruchamia program.

Aby utworzyć nowego użytkownika:

1. Upewnij się, że *SZBD PostgreSQL* został pomyślnie uruchomiony.
2. Zaloguj się na konto administratora *SZBD* *postgres*:

   ```
   su - postgres
   ```

3. Dodaj nowego użytkownika:

   ```
   /usr/local/pgsql/bin/createuser szel
   ```

4. Określ, czy nowy użytkownik będzie miał prawo tworzenia nowych baz danych i nowych użytkowników (rysunek 2.4).

Rysunek 2.4. *Wykonanie instrukcji programu psql \du wyświetli listę wszystkich użytkowników*

5. Sprawdź, czy możesz zalogować się do jakiejkolwiek bazy danych na nowo utworzone konto użytkownika:

```
psql -U szel test
```

Możemy również tworzyć nowych użytkowników z poziomu języka *SQL*. W tym celu:

1. Zaloguj się do systemowej bazy danych *template1*:

```
psql template1
```

2. Wykonaj instrukcję *SQL*:

```
CREATE USER maliniak;
```

3. Po chwili zostanie wyświetlony komunikat informujący o pomyślnym utworzeniu nowego konta użytkownika:

```
CREATE USER
```

Modyfikowanie konta użytkowników

Ćwiczenie 2.12.

Konto każdego użytkownika powinno być chronione hasłem. Administrator może określić hasło użytkownika podczas tworzenia jego konta, może również nadać lub zmienić istniejące hasło w dowolnym momencie po utworzeniu konta użytkownika.

Aby zmienić hasło istniejącego użytkownika:

1. Zaloguj się do systemowej bazy danych *template1*:

```
psql template1
```

2. Wykonaj instrukcję *SQL*:

```
ALTER USER maliniak
WITH PASSWORD 'ka_p^tan';
```

3. Po chwili zostanie wyświetlony komunikat potwierdzający zmianę hasła użytkownika:

```
ALTER USER
```

Instrukcja ALTER USER pozwala również określić, czy dany użytkownik będzie miał nadane prawo do tworzenia własnych baz danych oraz do dodawania nowych użytkowników. Dodatkowo możemy określić, po jakim czasie konto zostanie automatycznie zablokowane. Pełna składnia instrukcji wygląda następująco:

```
ALTER USER nazwa_użytkownika
[WITH PASSWORD 'hasło']
[CREATEDB | NOCREATEDB] [CREATEUSER | NOCREATEUSER]
[VALID UNTIL 'czas_ważności']
```

Na przykład, aby uniemożliwić użytkownikowi *maliniak* łączenie się z serwerem po pierwszym maja 2002 roku, należy wykonać instrukcję:

```
ALTER USER maliniak
VALID UNTIL '2002-05-01';
```

Usuwanie użytkowników

Ćwiczenie 2.13.

Konta użytkowników, którym należy odebrać prawo do łączenia się z bazami serwera Postgres powinny zostać usunięte z systemu. Ta operacja również może zostać przeprowadzona zarówno z poziomu systemu operacyjnego jaki i języka *SQL*. Aby usunąć konto użytkownika:

1. Upewnij się, że *SZBD PostgreSQL* został pomyślnie uruchomiony.
2. Zaloguj się na konto administratora *SZBD* `postgres`:

```
su - postgres
```

3. Usuń konto użytkownika:

```
/usr/local/pgsql/bin/dropuser maliniak
```

4. Po chwili zostanie wyświetlony komunikat potwierdzający usunięcie konta użytkownika:

```
DROP USER
```

Natomiast jeżeli jesteśmy już zalogowani do systemu *Postgres* i chcemy usunąć konto użytkownika:

1. Wykonaj instrukcję:

```
DROP USER maliniak;
```

2. Jeżeli został wyświetlony komunikat `ERROR: DROP USER: permission denied`, oznacza to, że użytkownik aktualnie podłączony do bazy danych nie jest jej administratorem i nie może usuwać kont innych użytkowników. Wyloguj się z systemu *Postgres* i zaloguj na konto administratora *SZBD PostgreSQL*.
3. Pomyślne wykonanie instrukcji zostanie potwierdzone komunikatem:

```
DROP USER
```

Tworzenie grupy użytkowników

Ćwiczenie 2.14.

Grupy użytkowników tworzone są po to, aby ułatwić i uprościć administrowanie kontami wielu użytkowników. Zamiast wielokrotnie nadawać te same uprawnienie do poszczególnych obiektów każdemu z użytkowników, administrator może utworzyć grupę użytkowników, dopisać do listy jej członków wybrane konta i nadać odpowiednie uprawnienie grupie.

Uprawnienia nadane grupie zostają niejawnie nadane wszystkim jej członkom. Jeżeli użytkownik jest członkiem kilku grup, jego efektywne uprawnienia będą sumą uprawnień nadanych każdej grupie.

Aby utworzyć grupę `czytelnicy`:

1. Zaloguj się do systemowej bazy danych *template1*:

```
psql template1
```

2. Wykonaj instrukcję *SQL*:

```
CREATE GROUP czytelnicy
WITH USER szel;
```

3. Po chwili zostanie wyświetlony komunikat potwierdzający utworzenie grupy użytkowników:

```
CREATE GROUP
```

Do tworzenia grupy użytkowników służy instrukcja *SQL* `CREATE GROUP`. Pełna składnia instrukcji wygląda następująco:

```
CREATE GROUP nazwa
[WITH
[SYSID gid ]
[USER nazwa użytkownika [,...]]]
```

Dodawanie i usuwanie kont użytkowników z grupy

Ćwiczenie 2.15.

Uważni Czytelnicy zauważyli już, że instrukcje języka *SQL* dotyczące zarządzania kontami użytkowników i grupami użytkowników są bardzo podobne. Tak jak modyfikowaliśmy właściwości poszczególnych kont za pomocą instrukcji `ALTER USER`, tak do modyfikowania właściwości grup użytkowników wykorzystamy instrukcje `ALTER GROUP`.

Aby dodać do grupy *czytelnicy* konto użytkownika *maliniak*:

1. Zaloguj się do systemowej bazy danych *template1*:

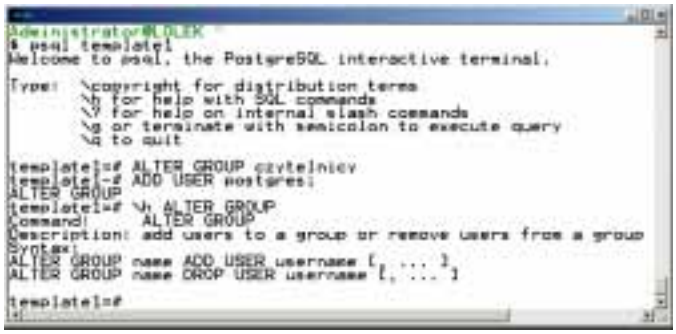
```
psql template1
```

2. Wykonaj instrukcję *SQL*:

```
ALTER GROUP czytelnicy
ADD USER postgres;
```

3. Po chwili zostanie wyświetlony komunikat potwierdzający zmodyfikowanie grupy użytkowników (rysunek 2.5):

```
ALTER GROUP
```



Rysunek 2.5. *Wykonując polecenie \h ALTER GROUP programu psql wyświetli kontekstowy plik pomocy*

Nadawanie uprawnień

Ćwiczenie 2.16.

SZBD PostgreSQL steruje dostępem do obiektów baz danych poprzez wykorzystanie systemu uprawnień, które mogą być udzielane lub odbierane za pomocą poleceń *SQL* `GRANT` i `REVOKE`. Administrator może nadać odpowiednie uprawnienia użytkownikowi (lub grupie użytkowników) do każdego z obiektów. Nadane uprawnienia określają, jakie operacje (np. odczytu danych) będą mogły zostać przeprowadzone przez wybranego użytkownika.

Polecenie `GRANT` ma następującą składnię:

```
GRANT uprawnienie [, ...] ON nazwa obiektu [, ...]
TO {PUBLIC | GROUP nazwa grupy | nazwa użytkownika }
```

Słowo kluczowe `PUBLIC` jest skrótem oznaczającym wszystkich użytkowników.

Dozwolone uprawnienia to:

- `SELECT` — umożliwia odczytywanie wierszy.
- `INSERT` — umożliwia tworzenie nowych wierszy.
- `DELETE` — umożliwia usuwanie wierszy.
- `UPDATE` — umożliwia aktualizację istniejących wierszy.
- `RULE` — umożliwia tworzenie reguł dla tabeli lub perspektywy.
- `ALL` — nadaje wszystkie wyżej wymienione uprawnienia.

Parametrem *nazwa obiektu* może być nazwa tabeli, perspektywy lub sekwencji.

Aby nadać uprawnienie do wstawiania nowych rekordów do systemowej tabeli[2] `pg_user` użytkownikowi *szel*:

1. Zaloguj się do systemowej bazy danych *template1*:

```
psql template1
```

2. Wykonaj instrukcję *SQL*:

```
GRANT insert
ON pg_user
TO szel;
```

3. Po chwili zostanie wyświetlony komunikat potwierdzający zmodyfikowanie uprawnień użytkownika:

```
GRANT
```

[2] W rzeczywistości tabela `pg_user` jest tabelą wirtualną — widokiem utworzonym na podstawie danych przechowywanych w innych tabelach. Wykonując ćwiczenie 2.19 Czytelnik pozna definicję tego widoku.

Odbieranie uprawnień

Ćwiczenie 2.17.

Prawie na pewno cześć informacji przechowywanych w bazie danych będzie miała charakter poufny. Dostęp do tych informacji powinni mieć jedynie wybrani użytkownicy. Ponadto, niektóre dane dostępne dla wszystkich użytkowników nie powinny być przez nich modyfikowane. Osiągnąć to możemy odbierając uprawnienia do wykonania określonych instrukcji pewnym użytkownikom lub ich grupom. Aby uniemożliwić modyfikowanie i dodawanie rekordów tabeli `pg_user` użytkownikom należącym do grupy *`czytelnicy`*:

1. Zaloguj się do systemowej bazy danych *template1*:

```
psql template1
```

2. Wykonaj instrukcję *SQL*:

```
REVOKE insert, update
ON pg_user
FROM GROUP czytelnicy;
```

3. Po chwili zostanie wyświetlony komunikat potwierdzający zmodyfikowanie uprawnień grupy użytkowników:

```
REVOKE
```

Program psql

Podstawowym narzędziem służącym do administrowania *SZBD PostgreSQL* dla *Linuksa* jest program `psql`. Program ten jest częścią systemu *PostgreSQL* i zostanie automatycznie zainstalowany podczas jego instalacji. Program `psql` był już wykorzystywany w poprzednich ćwiczeniach — teraz nadeszła pora, żeby poświęcić mu nieco więcej uwagi.

Uruchamianie psql

Ćwiczenie 2.18.

Pełna składnia polecenia `psql` jest następująca:

```
psql [opcje] [nazwa bazy danych [nazwa użytkownika]]
```

Niepodanie dowolnego argumentu spowoduje przyjęcie jego wartości domyślnej. Domyślna baza danych znajduje się na komputerze lokalnym, zaś do uwierzytelniania zostanie użyta nazwa tego użytkownika, który uruchomił program.

Aby połączyć się z bazą znajdującą się na serwerze wywołamy `psql` z nazwą bazy danych:

```
psql -d test
```

Domyślne wartości nazwy bazy danych, nazwy użytkownika, nazwy hosta serwera oraz portu można zastąpić poprzez ustawienie zmiennych środowiskowych: `PGDATABASE`, `PGUSER`, `PGHOST` oraz `PGPORT`.

Tabela 2.1 zawiera krótki opis najważniejszych opcji wywołania programu `psql`.

Tabela 2.1. *Lista opcji dostępnych przy uruchamianiu programu psql*

Opcja	Znaczenie
`-a`	Wyświetlanie wszystkich poleceń skryptu
`-A`	Wyświetlanie tabel bez wyrównywania (odpowiednik opcji `-P format = unaligned`)
`-c` <*zapytanie*>	Wykonanie pojedynczego zapytania i wyjście z programu
`-d` <*nazwa bazy danych*>	Określenie bazy danych, do której ma nastąpić podłączenie (domyślnie wartość zmiennej `$PGDATABASE` lub bieżąca nazwa zalogowanego użytkownika)
`-e`	Echo zapytań wysyłanych do wewnętrznego elementu przetwarzającego
`-E`	Wyświetla zapytania generowane przez polecenia wewnętrzne
`-f` <*nazwa pliku*>	Wykonuje zapytania z pliku, po czym kończy działanie
`-F` <*ciąg znaków*>	Ustawia separator pól (domyślnie: „|”) (odpowiednik opcji `-P fieldsep=<ciąg znaków>`)
`-h` <*host*>	Określa hosta serwera bazy danych (domyślnie wartość zmiennej `$PGHOST` lub komputer lokalny)
`-H`	Tabele wyświetlane w formacie HTML (odpowiednik opcji `-P format=html`).
`-l`	Wyświetlenie listy dostępnych baz danych i zakończenie działania.
`-n`	Wyłączenie mechanizmu *readline* (możliwości wywoływania poprzednich poleceń)
`-o` <*nazwa pliku*>	Wysłanie wyniku zapytania do pliku
`-p` <*port*>	Określenie portu serwera bazy danych (domyślnie wartość zmiennej `$PGPORT` lub wartość podana podczas kompilacji — domyślnie 5432)
`-P` *zmienna [=arg]*	Ustawienie zmiennej drukowania *zmienna* na wartość *arg*
`-q`	Uruchomienie w trybie `quiet` — bez wyświetlania komunikatów. Wyświetlane będą tylko wynik zapytań
`-R` <*ciąg znaków*>	Ustawienie separatora rekordów. Domyślnie jest nim znak końca wiersza (odpowiednik opcji `-P recordsep =` <*ciąg znaków*>)
`-s`	Tryb „krok po kroku” (potwierdzanie każdego zapytania)
`-S`	Tryb pojedynczego wiersza (znakiem zakończenia zapytania będzie znak końca wiersza, a nie średnik)
`-t`	Drukowanie samych wierszy (odpowiednik opcji `-P tuples_only`)
`-T` <*tekst*>	Ustawienie znaczników tabel HTML (szerokość, ramki) (odpowiednik opcji `-P tabbleattr=`<*tekst*>)
`-U` <*nazwa użytkownika*>	Określenie użytkownika bazy danych (domyślnie wartość zmiennej `$PGUSER` lub aktualne konto logowania)
`-v` *nazwa=wartość*	Ustawienie wartości zmiennej `psql` *nazwa* na wartość *wartość*
`-V`	Wyświetla informacje o wersji i kończy działanie
`-W`	Wyświetla pytanie o hasło (wywoływane automatycznie, gdy hasło jest wymagane)
`-x`	Włączenie rozszerzonego wyjścia tabel
`-X`	Pominięcie odczytu pliku startowego (pliku `~/.psqlrc`)

Wyświetlanie informacji o bazie danych za pomocą wewnętrznych poleceń programu psql

Ćwiczenie 2.19.

Program `psql` jest nie tylko terminalem, za pomocą którego możemy wykonywać instrukcje *strukturalnego języka zapytań*. Posiada on również kilkadziesiąt własnych poleceń, za pomocą których możemy:

- konfigurować środowisko programu,
- wyświetlać informacje o obiektach wybranej bazy danych,
- wywoływać polecenie *SQL*, takie jak `COPY` lub `SET`.

Aby podłączyć się do wybranej bazy danych i wyświetlić informacje o tabeli `pg_user`:

1. Zaloguj się do bazy danych *`test`*:

   ```
   psql test
   ```

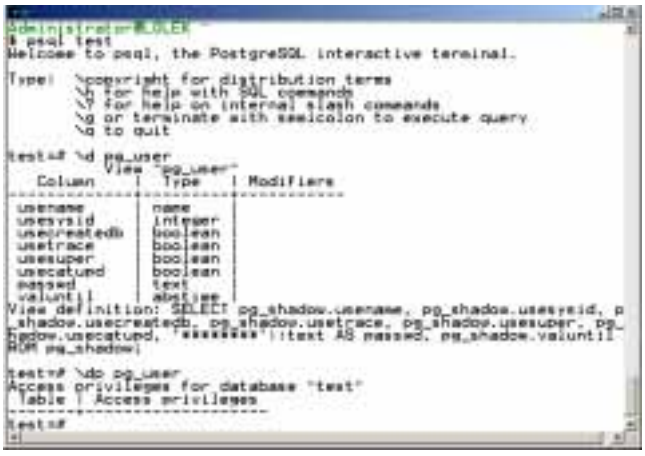
2. Wykonaj polecenie programu `psql`:

   ```
   \d pg_user
   ```

   Zostaną wyświetlone podstawowe informacje o widoku: lista kolumn, typ danych oraz składnia instrukcji `SELECT` będącej podstawą widoku.

3. Aby wyświetlić listę użytkowników, którzy mają nadane uprawnienia do tego obiektu wpisz (rysunek 2.6):

   ```
   \dp pg_user
   ```



Rysunek 2.6. *Za pomocą poleceń programu psql możemy uzyskać wiele cennych informacji o bazie danych*

4. Zakończ pracę programu:

   ```
   \q
   ```

Wywołanie polecenia `\dp` bez parametru będącego nazwą obiektu w bazie danych spowoduje wyświetlenie listy uprawnień nadanych do wszystkich obiektów bieżącej bazy danych.

Lista wewnętrznych poleceń programu `psql` znajduje się w tabeli 2.2.

Tabela 2.2. *Lista poleceń programu psql*

Polecenie	Znaczenie
`\a`	Przełączanie pomiędzy trybem z wyrównywaniem i bez wyrównywania wierszy wyników
`\c[onnect] [`*`nazwa bazy`*`| - [`*`użytkownik`*`]]`	Podłączenie do nowej bazy danych
`\C <`*`nagłówek`*`>`	Ustawienie nagłówka tabeli w wyniku
`\copy ...`	Wykonanie instrukcji *SQL* `COPY` — skopiowanie danych zewnętrznych do lub z bazy danych
`\copyright`	Wyświetlenie zasad użytkowania i dystrybucji *SZBD PostgreSQL*
`\d <`*`nazwa obiektu`*`>`	Opis tabeli (perspektywy, indeksu, sekwencji)
`\d{t|i|s|v}`	Wyświetlenie listy tabel, indeksów, sekwencji bądź perspektyw
`\d{p|S|l}`	Wyświetlenie uprawnień, tabel systemowych bądź obiektów typu *BLOB*
`\da`	Wyświetlenie listy funkcji agregacji
`\dd [`*`nazwa obiektu`*`]`	Wyświetlenie listy komentarzy dla tabeli, typu, funkcji lub operatora
`\df`	Wyświetlenie listy funkcji
`\do`	Wyświetlenie listy operatorów
`\dT`	Wyświetlenie listy typów
`\e [`*`nazwa pliku`*`]`	Edycja bieżącego bufora zapytań lub pliku *nazwa pliku* za pomocą zewnętrznego edytora
`\echo <`*`tekst`*`>`	Wyświetlenie tekstu na standardowym urządzeniu wyjściowym
`\encoding <`*`kodowanie`*`>`	Ustawienie kodowania na komputerze-kliencie
`\f <`*`separator`*`>`	Zmiana separatora pól
`\g <`*`nazwa pliku`*`>`	Wysłanie zapytania do wewnętrznego elementu przetwarzającego (a wyników do pliku lub potoku)
`\h [`*`polecenie`*`]`	Pomoc na temat składni poleceń *SQL*; w celu wyświetlenia szczegółowych informacji o wszystkich poleceniach należy wpisać *
`\H`	Włączenie trybu *HTML*
`\i <`*`nazwa pliku`*`>`	Wczytywanie i wykonywanie zapytań z pliku
`\l`	Wyświetlenie listy wszystkich baz danych
`\lo_export, \lo_import, \lo_list, \lo_unlink`	Operacje z obiektami *BLOB*
`\o [`*`nazwa pliku`*`]`	Wysłanie wszystkich wyników zapytania do pliku lub potoku
`\p`	Wyświetlenie zawartości bieżącego bufora zapytań
`\pset <`*`opcja`*`>`	Ustawienie opcji wyświetlania tabel. Dostępne są następujące opcje: `format`, `border`, `expanded`, `fieldsep`, `null`, `recordsep`, `tuples_only`, `title`, `tableattr`, `pager`

Tabela 2.2. *Lista poleceń programu psql — ciąg dalszy*

Polecenie	Znaczenie
\q	Zakończenie działania programu psql
\qecho <*tekst*>	Zapis tekstu *tekst* do strumienia wynikowego zapytań
\r	Kasowanie bufora zapytań
\s [*nazwa pliku*]	Wyświetlenie historii wykonywanych poleceń lub jej zapis do pliku
\set <*zmienna*> <*wartość*>	Ustawienie wartości zmiennej wewnętrznej
\t	Wyświetlanie tylko wierszy (bez nagłówków kolumn i liczby wierszy)
\T <*znaczniki*>	Znaczniki tabeli *HTML*
\unset <*zmienna*>	Wyzerowanie wartości zmiennej wewnętrznej
\w <*nazwa pliku*>	Zapis aktualnego bufora zapytań do pliku
\x	Włączenie trybu rozszerzonego dla wyniku
\z	Wyświetlenie listy uprawnień dostępu do tabel
\! [*polecenie*]	Wyjście do powłoki lub wykonanie polecenia powłoki
\i <*nazwa_pliku*>	Uruchomienie poleceń czytanych z pliku <*nazwa_pliku*>
\r	Wyzerowanie bufora
\?	Uzyskanie ekranu pomocy

Wydawanie poleceń języka SQL z programu psql

Ćwiczenie 2.20.

Domyślnie, znakiem końca pojedynczej instrukcji *SQL* dla programu psql jest średnik. Ponieważ instrukcja może zostać zapisana w dowolnej liczbie wierszy, program będzie czekał z wykonaniem instrukcji do momentu wykrycia znaku ;.

Dodatkowo, program ułatwia znalezienie błędu w wykonywanej właśnie instrukcji, wyświetlając oprócz komunikatu błędu miejsce jego wystąpienia.

Aby wyświetlić alfabetycznie uporządkowaną listę utworzonych użytkowników bazy danych:

1. Zaloguj się do bazy danych template1:

```
psql template1
```

2. Wykonaj instrukcję *SQL*:

```
SELECT usename, passwd
FROM pg_user
WHERE usesysid>100
ORDER BI usename;
```

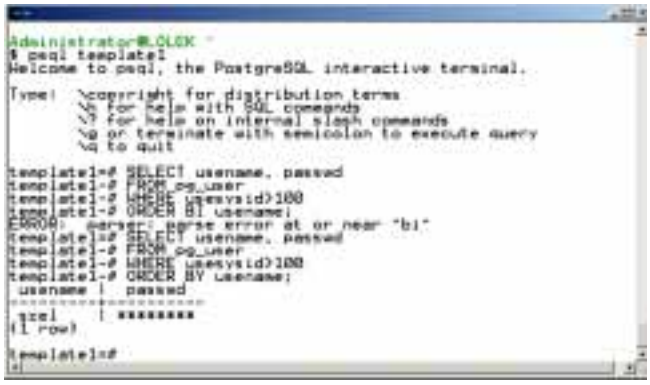
Zwróć uwagę, że program psql rozpoczął wykonywanie instrukcji dopiero po napotkaniu średnika — naciśnięcie klawisza *Enter* powodowało jedynie przejście do kolejnego wiersza.

3. Ponieważ ostatni wiersz instrukcji zawierał błąd składniowy, program `psql` zamiast wykonać polecenie, zwrócił komunikat o błędzie — nierozpoznanym słowie „bi”.

4. O ile tylko w Twoim systemie operacyjnym uruchomiona jest usługa bufora klawiatury, naciskając klawisz „↑” wywołasz ponownie wpisane ostatnio wiersze polecenia. Wiedząc, że błąd wystąpił w ostatnim wierszu powtórz trzy pierwsze wiersze instrukcji (po każdym naciskając klawisz *Enter*), przywołaj ostatni wiersz i popraw błąd. Poprawna składnia instrukcji przyjmie postać:

```
SELECT usename, passwd
FROM pg_user
WHERE usesysid>100
ORDER BY usename;
```

5. Tym razem program wykona instrukcję i wyświetli na ekranie wybrane dane o użytkownikach (rysunek 2.7).



Rysunek 2.7. *Program psql nie ma co prawda opcji sprawdzania poprawności instrukcji, ale wskazuje lokalizacje błędu*

Uruchamianie skryptów za pomocą programu psql

Ćwiczenie 2.21.

Kolejną możliwością programu `psql` jest wykonanie skryptu zawierającego zarówno instrukcje *SQL*, jak i polecenia wewnętrzne programu.

Wielu administratorów zapisuje grupy instrukcji do pliku i korzysta z niego jak z prostego skryptu. Odczytanie zestawu poleceń `psql` z pliku jest możliwe przy użyciu wewnętrznego polecenia `\i`.

1. Zaloguj się do bazy danych *`test`*:

```
psql test
```

2. Wyświetl dane o wszystkich użytkownikach:

```
SELECT *
FROM pg_user;
```

3. Zapisz ostatnio wykonywane polecenie (przechowywane w buforze) do pliku *\w selekt.sql*.

4. Za pomocą dowolnego edytora tekst wyświetl zawartość pliku */home/postgres/selekt.sql*.
5. Plik zawiera jedną instrukcję `SELECT`. Zamknij okno edytora i przywróć sesje programu `psql`.
6. Wykonaj instrukcje zapisane w pliku *selekt.sql*:

```
\i selekt.sql
```

7. Program wyświetli dane o wszystkich użytkownikach bazy danych.