

Cómo trabajar con PostgreSQL

Para conectarse al servidor

Es necesario logearse a `anakena.dcc.uchile.cl` por SSH a través de una ventana terminal. En la ventana, ejecutar:

```
psql -U cc42aXX -d cc42aXXdb
```

En el caso de la conexión a dichato (postgrado), ejecutar:

```
psql -U cc55aXX -d cc55aXXdb
```

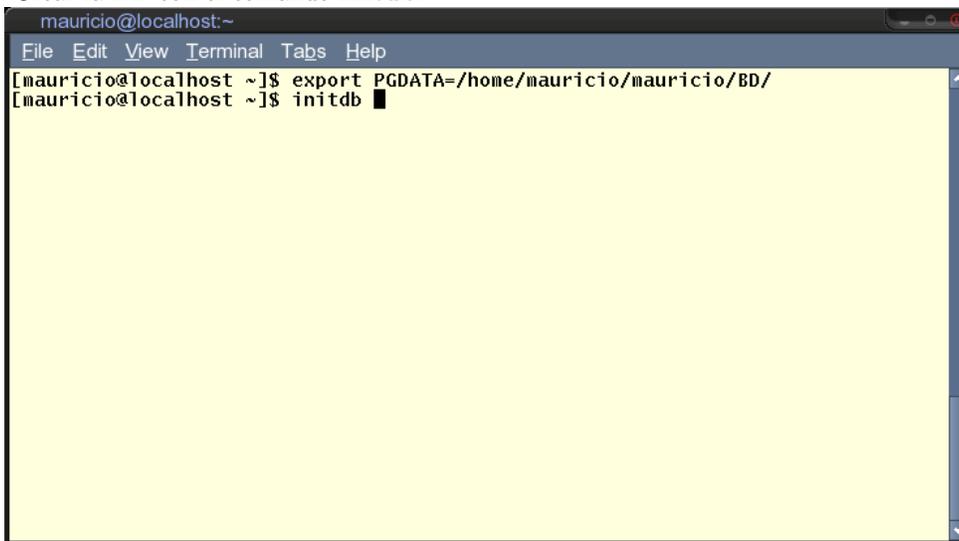
Para avanzar en una base de datos PostgreSQL personal

Si se trabaja en la casa, o el demonio Postmaster no está en ejecución, seguir los siguientes pasos:

Construir la base de datos:

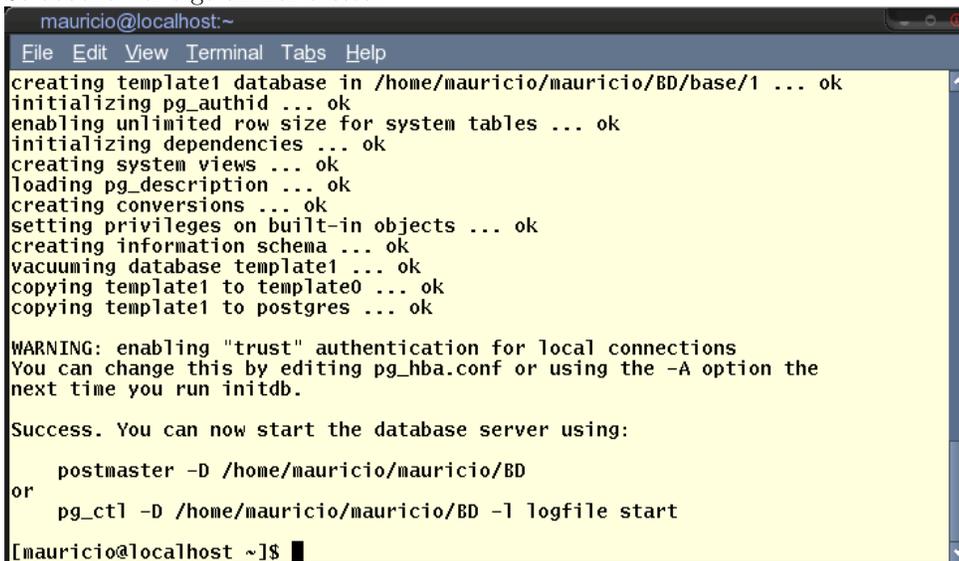
Setear la variable de ambiente **PGDATA** hacia la dirección donde la BD será creada.

Crear la BD con el comando **initdb**.



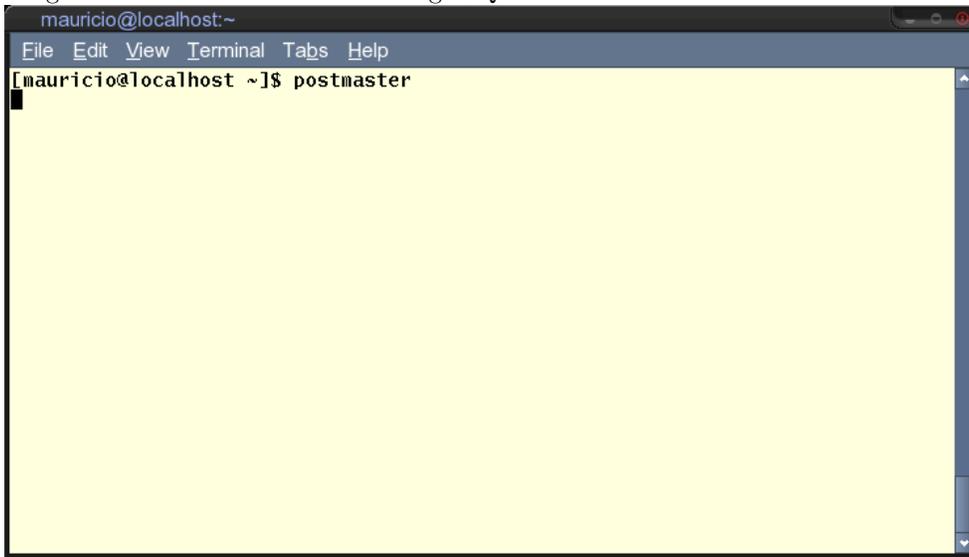
```
mauricio@localhost:~  
File Edit View Terminal Tabs Help  
[mauricio@localhost ~]$ export PGDATA=/home/mauricio/mauricio/BD/  
[mauricio@localhost ~]$ initdb
```

Se debería ver algo similar a esto:

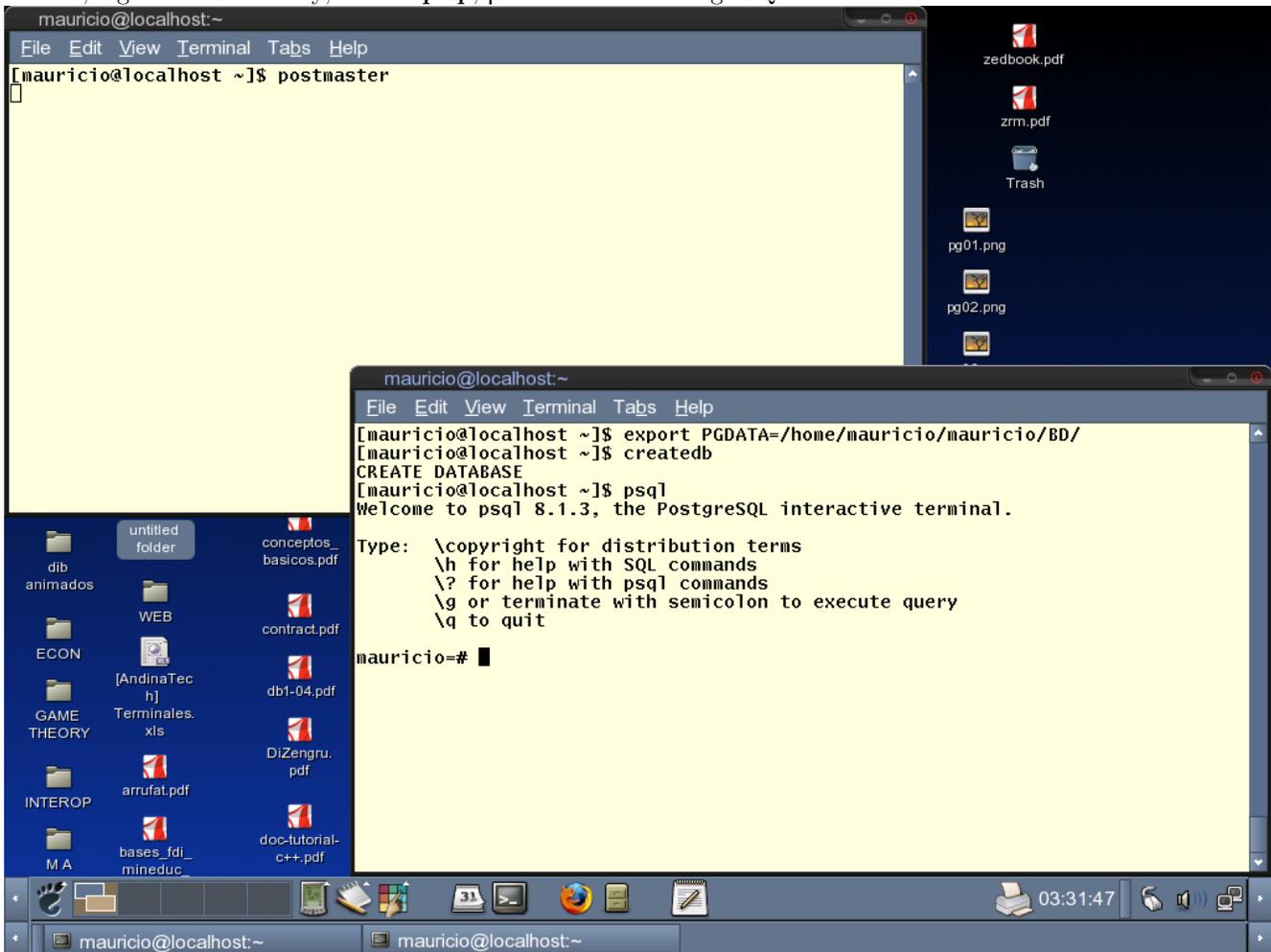


```
mauricio@localhost:~  
File Edit View Terminal Tabs Help  
creating template1 database in /home/mauricio/mauricio/BD/base/1 ... ok  
initializing pg_authid ... ok  
enabling unlimited row size for system tables ... ok  
initializing dependencies ... ok  
creating system views ... ok  
loading pg_description ... ok  
creating conversions ... ok  
setting privileges on built-in objects ... ok  
creating information schema ... ok  
vacuuming database template1 ... ok  
copying template1 to template0 ... ok  
copying template1 to postgres ... ok  
  
WARNING: enabling "trust" authentication for local connections  
You can change this by editing pg_hba.conf or using the -A option the  
next time you run initdb.  
  
Success. You can now start the database server using:  
  
    postmaster -D /home/mauricio/mauricio/BD  
or  
    pg_ctl -D /home/mauricio/mauricio/BD -l logfile start  
[mauricio@localhost ~]$
```

Luego iniciamos la base de datos PostgreSQL:

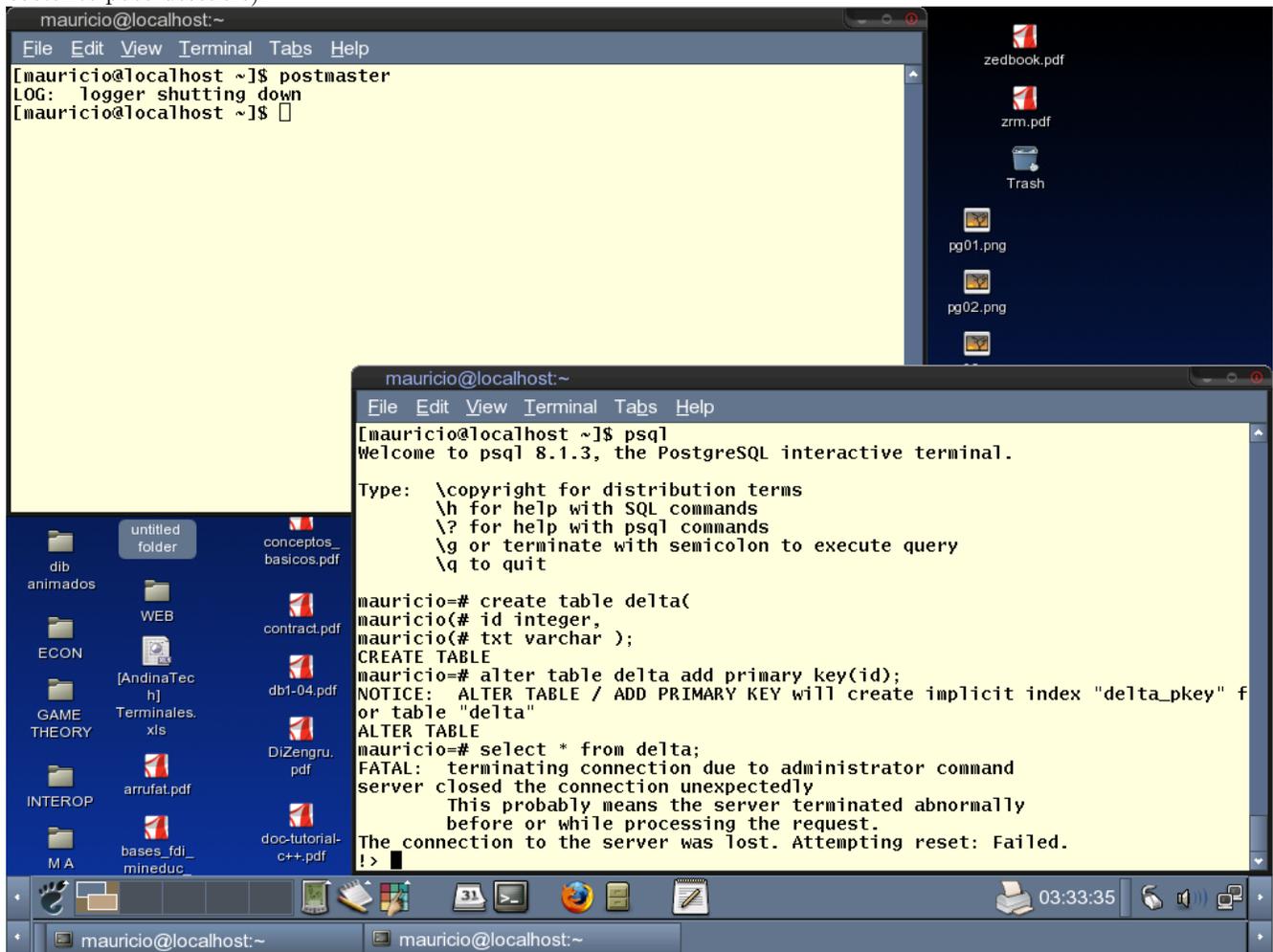


Ahora está en ejecución una BD. Localmente podemos acceder desde otro terminal. En este caso se puede abrir otra consola, registrar PGDATA y, usando **psql**, ¡voilà! Tenemos PostgreSQL funcionando localmente.

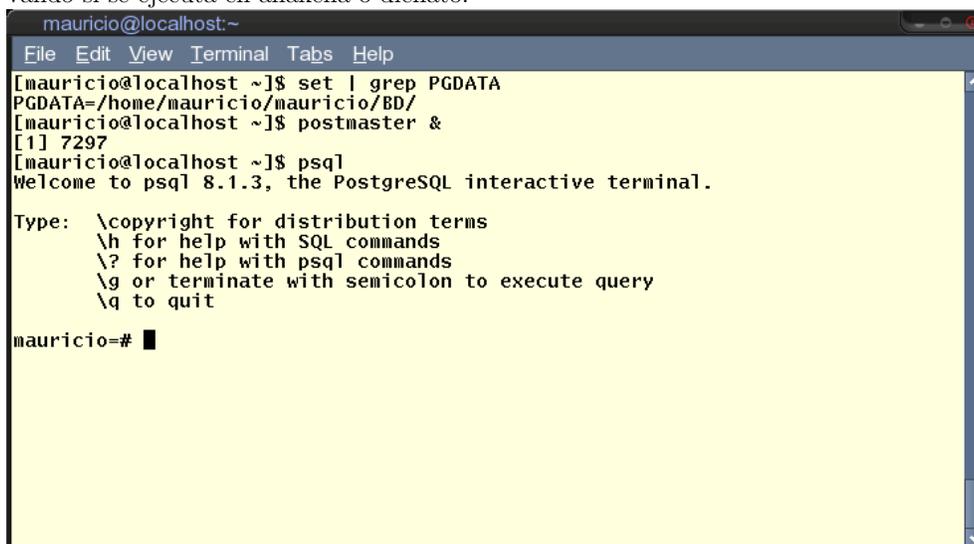


(No vea mi escritorio)

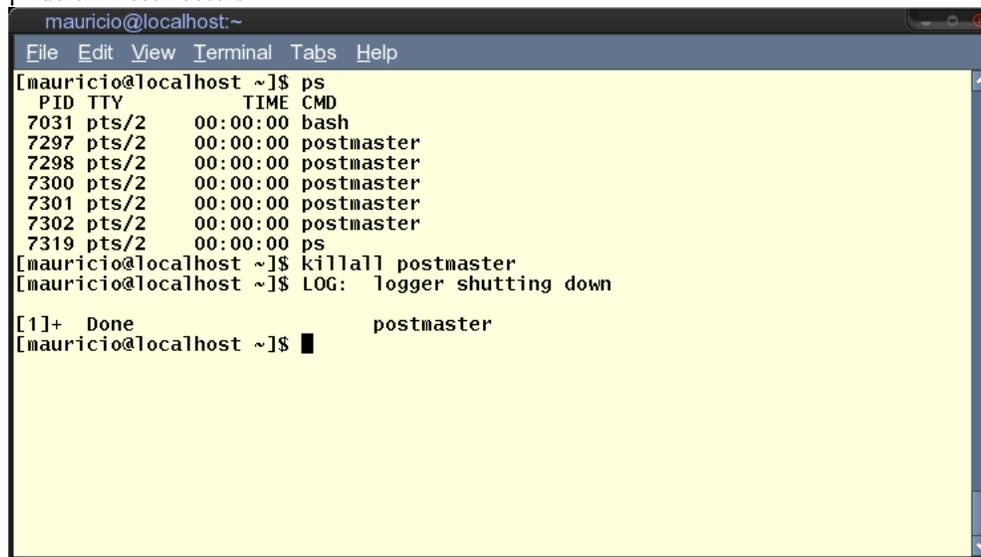
Pero es necesario que Postmaster siga en ejecución. Si el proceso es eliminado, psql dejará de funcionar (lo que es bastante poco deseable).



Otra alternativa, más cómoda -a mí parecer-, es ejecutar Postmaster en el mismo terminal y dejarlo en el fondo usando &. Cuando se desee destruir el proceso, aniquilarlo con kill. O bien, **killall postmaster**. Esto también es válido si se ejecuta en anakena o dichato.



¡Mueran Postmasters!

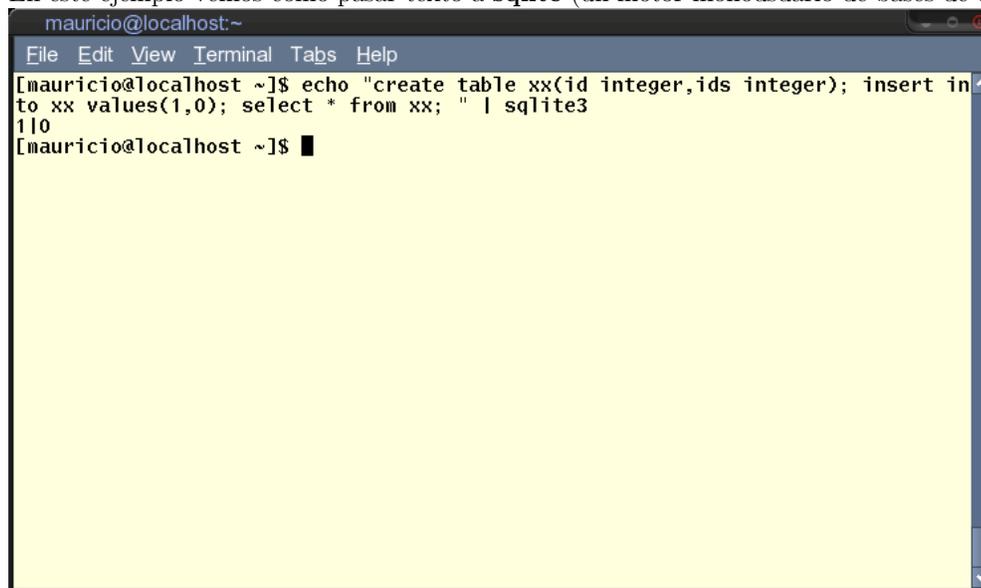


```
mauricio@localhost:~  
File Edit View Terminal Tabs Help  
[mauricio@localhost ~]$ ps  
  PID TTY          TIME CMD  
 7031 pts/2    00:00:00 bash  
 7297 pts/2    00:00:00 postmaster  
 7298 pts/2    00:00:00 postmaster  
 7300 pts/2    00:00:00 postmaster  
 7301 pts/2    00:00:00 postmaster  
 7302 pts/2    00:00:00 postmaster  
 7319 pts/2    00:00:00 ps  
[mauricio@localhost ~]$ killall postmaster  
[mauricio@localhost ~]$ LOG: logger shutting down  
  
[1]+  Done                  postmaster  
[mauricio@localhost ~]$ █
```

Una estrategia medianamente práctica para ejecutar comandos en bases de datos desde un terminal

Generalmente los sistemas administradores de bases de datos permiten el acceso desde terminal. Eso implica que se les pueden pasar comandos desde un **pipe** (`|`). El texto emitido por un proceso es pasado a otro proceso en el vuelo.¹

En este ejemplo vemos cómo pasar texto a **sqlite** (un motor monousuario de bases de datos):

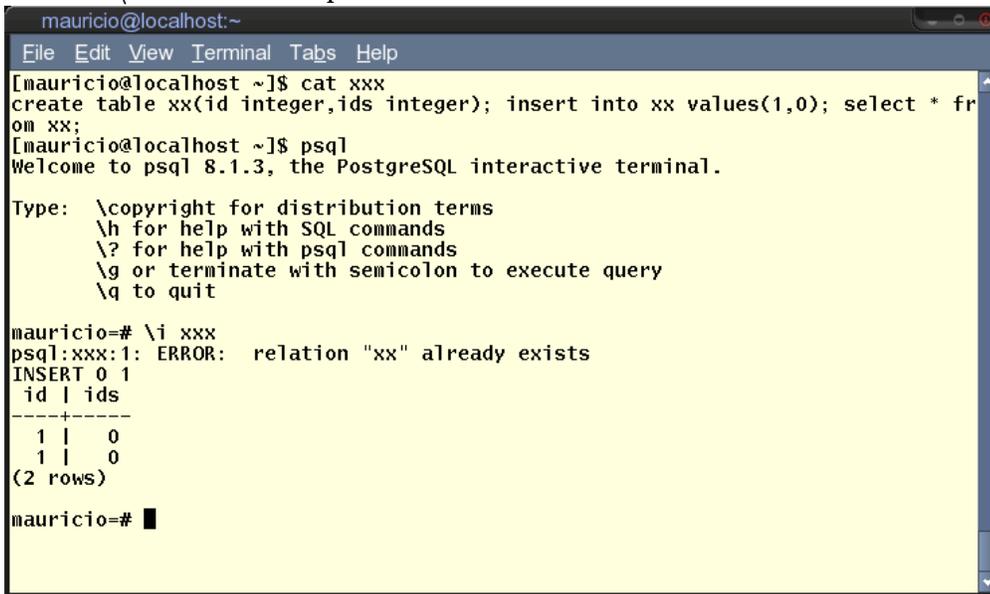


```
mauricio@localhost:~  
File Edit View Terminal Tabs Help  
[mauricio@localhost ~]$ echo "create table xx(id integer,ids integer); insert into xx values(1,0); select * from xx;" | sqlite3  
1|0  
[mauricio@localhost ~]$ █
```

Luego del pipe, la interfaz del sistema de base de datos cesará de funcionar (típicamente).

1 Esto se asegura en sistemas operativos que cumplen con POSIX. Probablemente esto no sea válido en Windows u otro sistema operativo no POSIX.

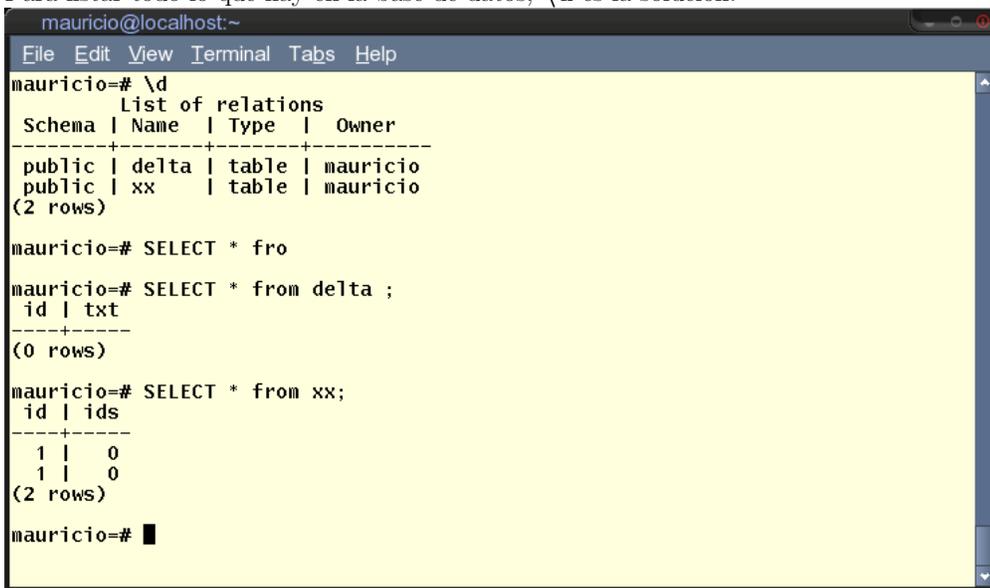
Esto mismo es válido con **psql**. Sin embargo, es más fácil dejar todo en un archivo separado y luego ejecutar el comando `\i archivo-con-sql-válido`.



```
mauricio@localhost:~  
File Edit View Terminal Tabs Help  
[mauricio@localhost ~]$ cat xxx  
create table xx(id integer,ids integer); insert into xx values(1,0); select * fr  
om xx;  
[mauricio@localhost ~]$ psql  
Welcome to psql 8.1.3, the PostgreSQL interactive terminal.  
  
Type: \copyright for distribution terms  
      \h for help with SQL commands  
      \? for help with psql commands  
      \g or terminate with semicolon to execute query  
      \q to quit  
  
mauricio=# \i xxx  
psql:xxx:1: ERROR: relation "xx" already exists  
INSERT 0 1  
 id | ids  
----+----  
  1 |   0  
  1 |   0  
(2 rows)  
  
mauricio=#
```

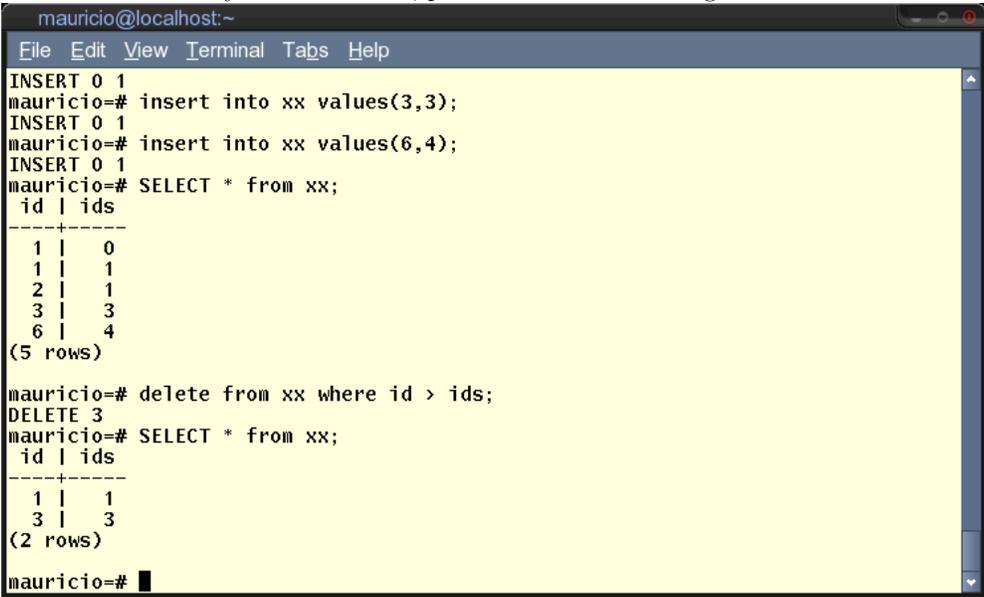
Otras ayudas prácticas para el proyecto

Para listar todo lo que hay en la base de datos, `\d` es la solución.



```
mauricio@localhost:~  
File Edit View Terminal Tabs Help  
mauricio=# \d  
List of relations  
Schema | Name | Type | Owner  
-----+-----+-----+-----  
public | delta | table | mauricio  
public | xx    | table | mauricio  
(2 rows)  
  
mauricio=# SELECT * fro  
mauricio=# SELECT * from delta ;  
 id | txt  
----+----  
(0 rows)  
  
mauricio=# SELECT * from xx;  
 id | ids  
----+----  
  1 |   0  
  1 |   0  
(2 rows)  
  
mauricio=#
```

Para insertar datos y eliminar tablas, podemos hacerlo como sigue:



```
mauricio@localhost:~  
File Edit View Terminal Tabs Help  
INSERT 0 1  
mauricio=# insert into xx values(3,3);  
INSERT 0 1  
mauricio=# insert into xx values(6,4);  
INSERT 0 1  
mauricio=# SELECT * from xx;  
id | ids  
---+---  
1 | 0  
1 | 1  
2 | 1  
3 | 3  
6 | 4  
(5 rows)  
  
mauricio=# delete from xx where id > ids;  
DELETE 3  
mauricio=# SELECT * from xx;  
id | ids  
---+---  
1 | 1  
3 | 3  
(2 rows)  
  
mauricio=#
```

Espero que esto les sea de ayuda, ¡saludos!

PD: Disculpen el enfoque de bajo nivel, pero yo soy 100% linuxero :P.
Mauricio