PostgreSQL 9.1: su uso con psql.

Y seguimos aprovechando el receso de Carnaval para continuar nuestro estudio del poderoso motor de base de datos relacional PostgreSQL 9.1.

Nuestra miniserie comienza acá y en la última entrada tocamos el tema de las consultas con uniones las cuales utilizaremos aquí también pero con ayuda del programa *psql* y tal como lo prometimos vamos a trabajar en profundidad por línea de comandos.

Como recordarán ya habíamos utilizado anteriormente a *psql* para garantizarle al usuario *adminsql* acceso íntegro a todas las bases de datos en el servidor *PostgreSQL* que instalamos.

Breve pausa para hablar sobre seguridad básica.

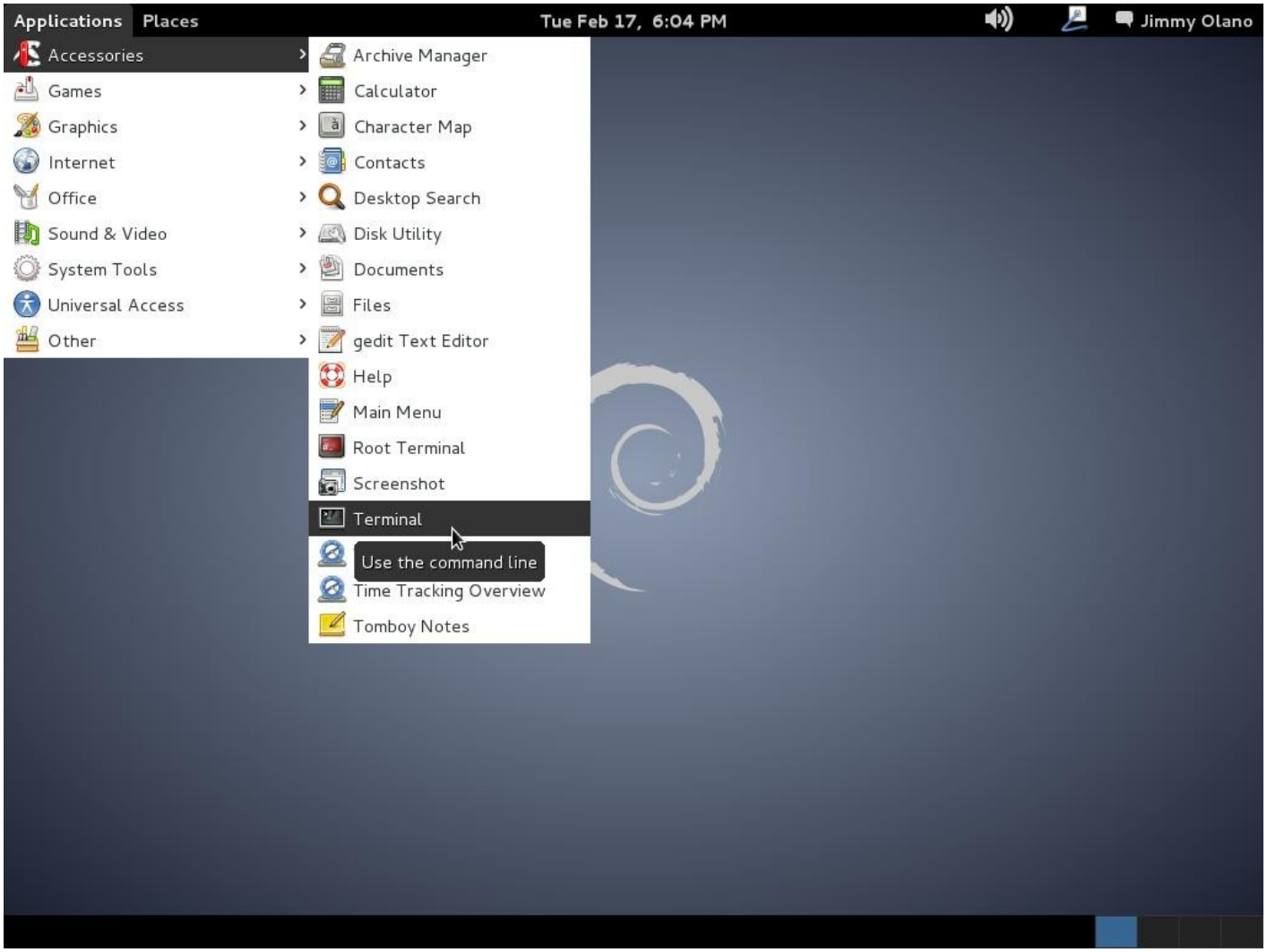
Normalmente lo que hicimos en el párrafo anterior sólamente se hace si hay un solo usuario y una sola base de datos, un sistema pequeño, tal como era a finales del siglo pasado. Pero ahora con computadoras con procesadores multinúcleo (e incluso hasta computadoras con multiprocesador) un servidor *PostgreSQL* puede, tranquilamente, atender muchos usuarios a la vez. Imaginemos que en nuestra *Academia de Software Libre* de ejemplo cada alumno debe acceder a este recurso único pero debemos de cuidar que cada quien trabaje en su propia base de datos y no toque las ajenas, ya sea por error o a propósito (que se ven casos así). Pues bien *PostgreSQL* soporta usuarios y grupos internamente, tal como si fuera una computadora con un sistema operativo moderno.

Un usuario pudiera ser cada alumno en clase con su base de datos propia para desarrollar su trabajo y aprobar el curso.

Un grupo, por ejemplo, pudiera ser un sección o curso en determinada aula de clase con derechos de sólo lectura sobre una base de datos perteneciente al instructor de clases y además **independiente** de otros instructores en otras seccionesde clase; eso nos facilita el trabajo al crear usuarios pues se le asigna al grupo y "hereda" la configuración.

Trabajando con *psql.*

Lo comentado en la sección anterior lo veremos ahora de forma práctica y de la manera más sencilla posible. En la máquina *Debian* que corre nuestro *PostgresSQL* seleccionamos *"Aplicaciones->Accesorios->Terminal":*



Abrimos así una ventana terminal donde procederemos a ingresar el comando *psql* pero en esta oportunidad lo haremos acompañado de opciones adicionales:

```
psql --username=adminsql --password --dbname=GNU_academy
```

Dichas opciones son:

- "*--username=**nombre_de_usuario***" permite colocar a continuación el nombre del usuario que se va a conectar; la forma abreviada es "*-U* **nombre_de_usuario**" con un espacio entre ambos (y tened en cuenta las mayúsculas y minúsculas, tal cual se escriben).
- "*--password*" para que nos pregunte la contraseña ("*--no-password*" será útil en trabajos *de procesos por lotes)*; la forma abreviada es "*-W*" (notad que las formas largas son con doble guión y las cortas con un solo guión).

- "--*dbname*=**nombre_de_la_base_de_datos**" nos conecta de una vez a donde vamos a trabajar; la forma abreviada es *"-d* nombre_de_la_base_de_datos" con un espacio entre ambos.

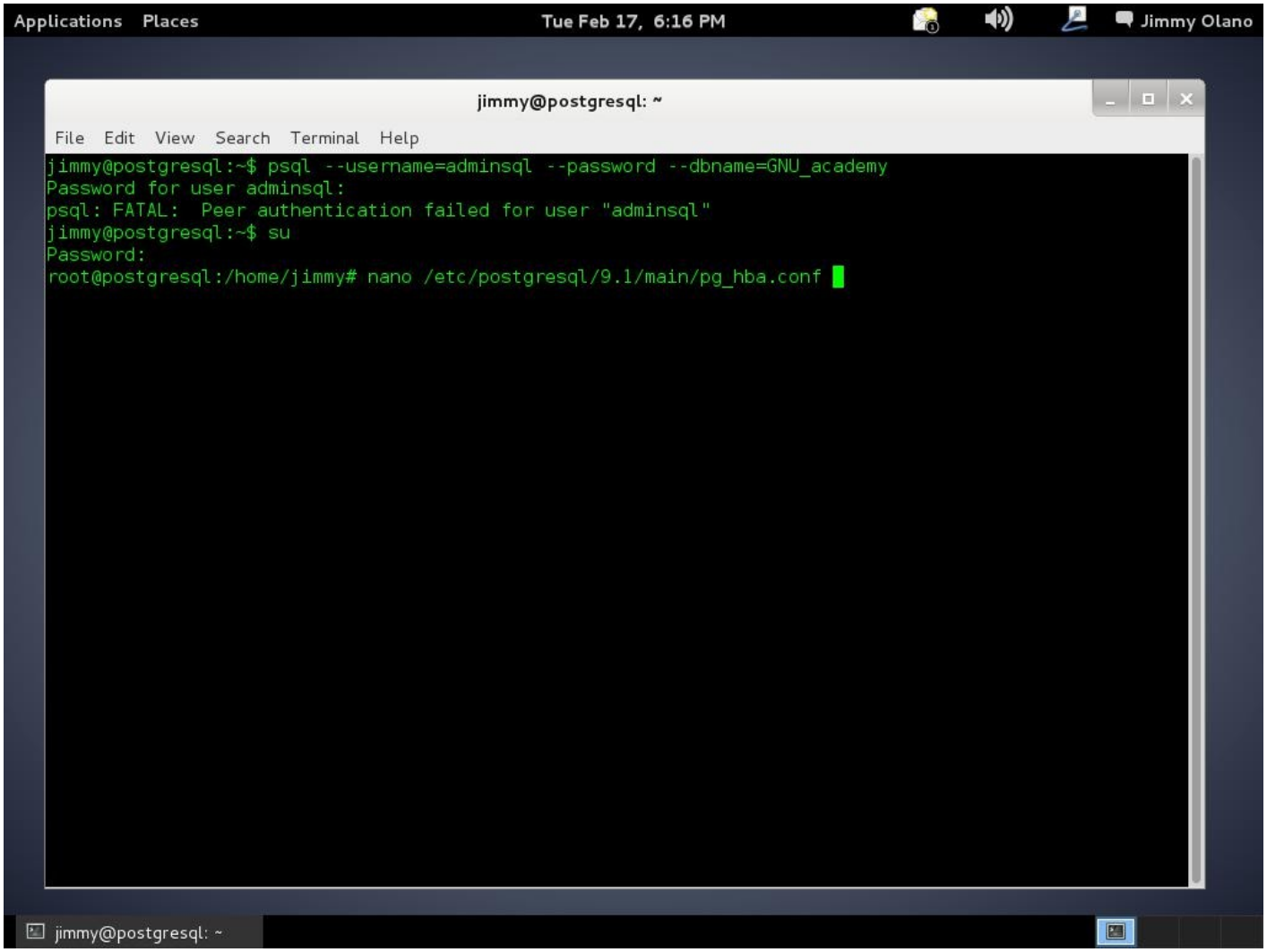
Por ahora ésas son las opciones que utilizaremos por la *terminal interactiva* pero en realidad *psql* tiene gran cantidad de agregados los cuales en otras entradas le daremos uso, pero no a todos. Si queréis echar un ojo a la lista completa, pincha aquí.

Una vez hayamos presionado *intro* y haber tecleado nuestra consabida contraseña al cabo de otro *intro* nos aparece el siguiente mensaje de error:

```
psql: FATAL: Peer authentication failed for user "adminsql"
```

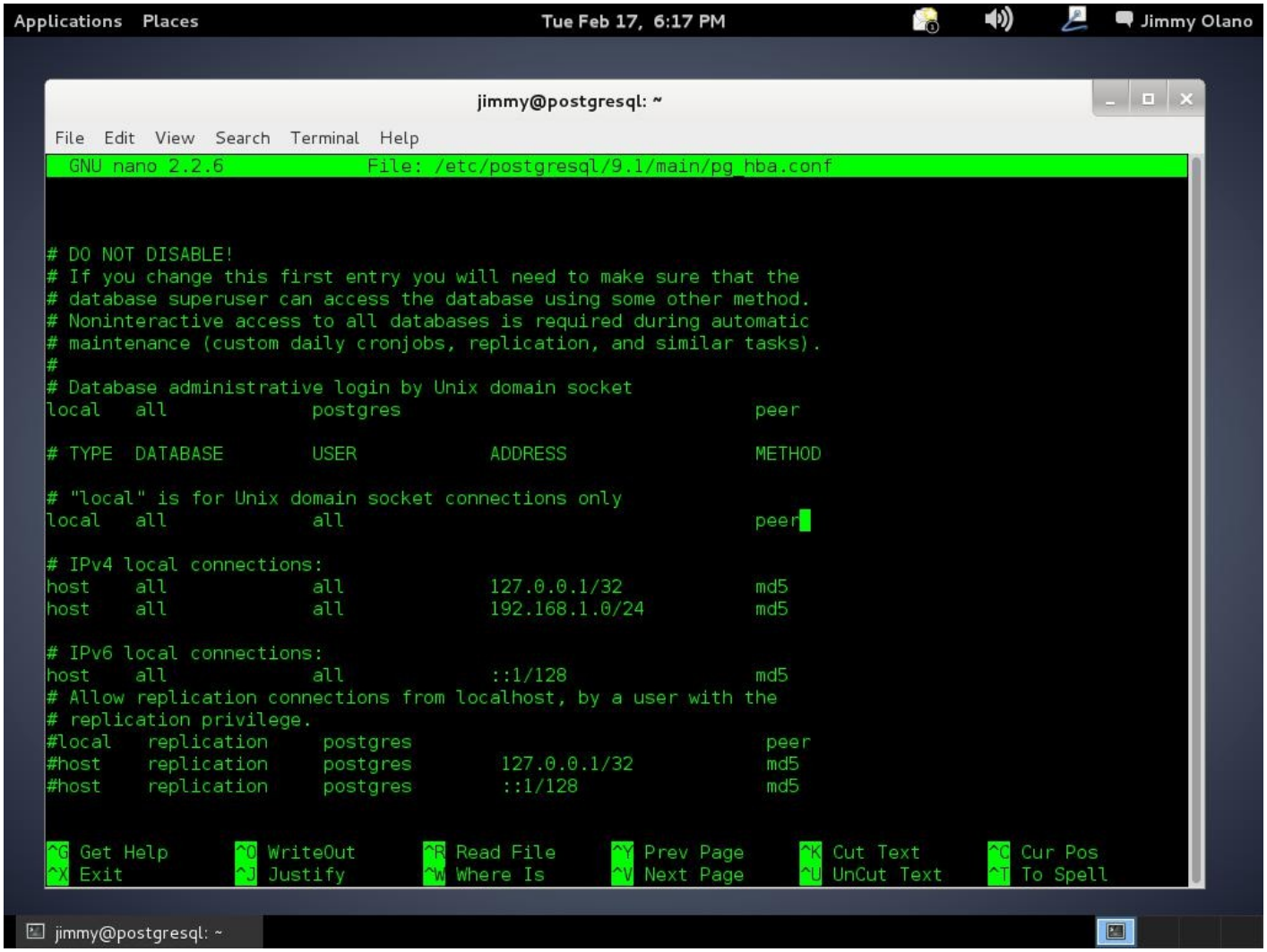
Y no, no es porque hayamos ingresado mal la contraseña, que es harta sencilla sino por el asunto de seguridad que describí hace poco. Para que podamos conectarnos **debemos** crear un usuario llamado *adminsql* con su correspondiente contraseña **en el sistema operativo Debian (o GNU/Linux que usemos).** Esto no sería deseable, ya que los usuarios tendrían acceso al sistema operativo en sí, amén que también si son muchos usuarios recarga inncesariamente el sistema de archivos (que aunque tengamos poderosas computadoras siempre la frugalidad es bienvenida), **y no, no necesitamos ése esquema de trabajo.**

Es por ello que contamos con la opción *peer* por defecto, apoyada por el sistema operativo **y contamos con la opción** *md5* que encripta la contraseña para mayor seguridad, así que sólo registraremos usuarios **dentro de PostgreSQL**. Para ello ganaremos acceso como *superusuario* y luego usaremos el programa editor de texto *nano* para editar la configuración. Por favor observen la imagen para que tengan el panorama completo:



nano /etc/postgresql/9.1/main/pg_hba.conf

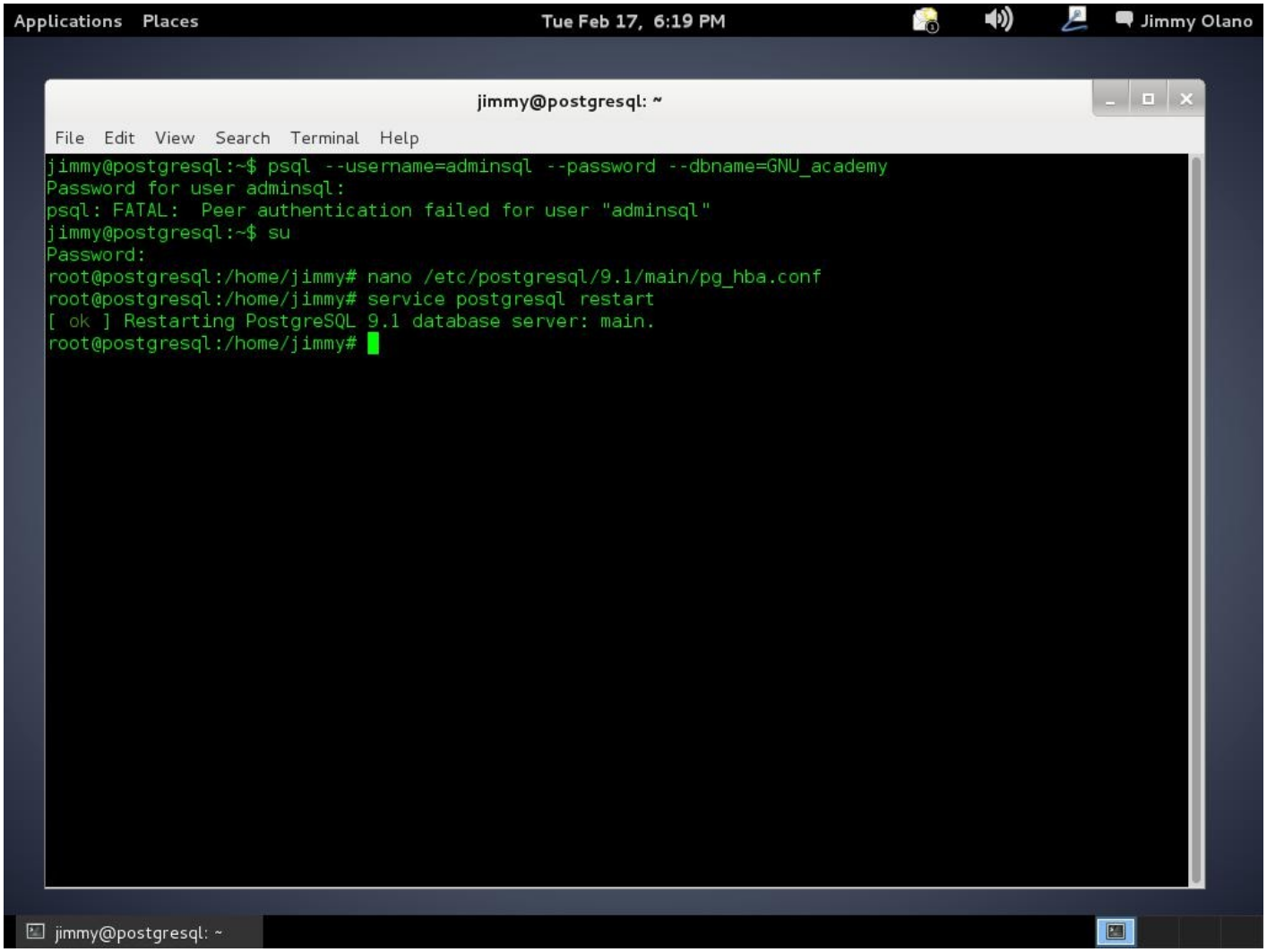
y buscaremos la sección *"is for Unix domain sockets connections only"* donde cambiaremos *"peer"* por *"md5"* tal como aparece en la siguiente figura:



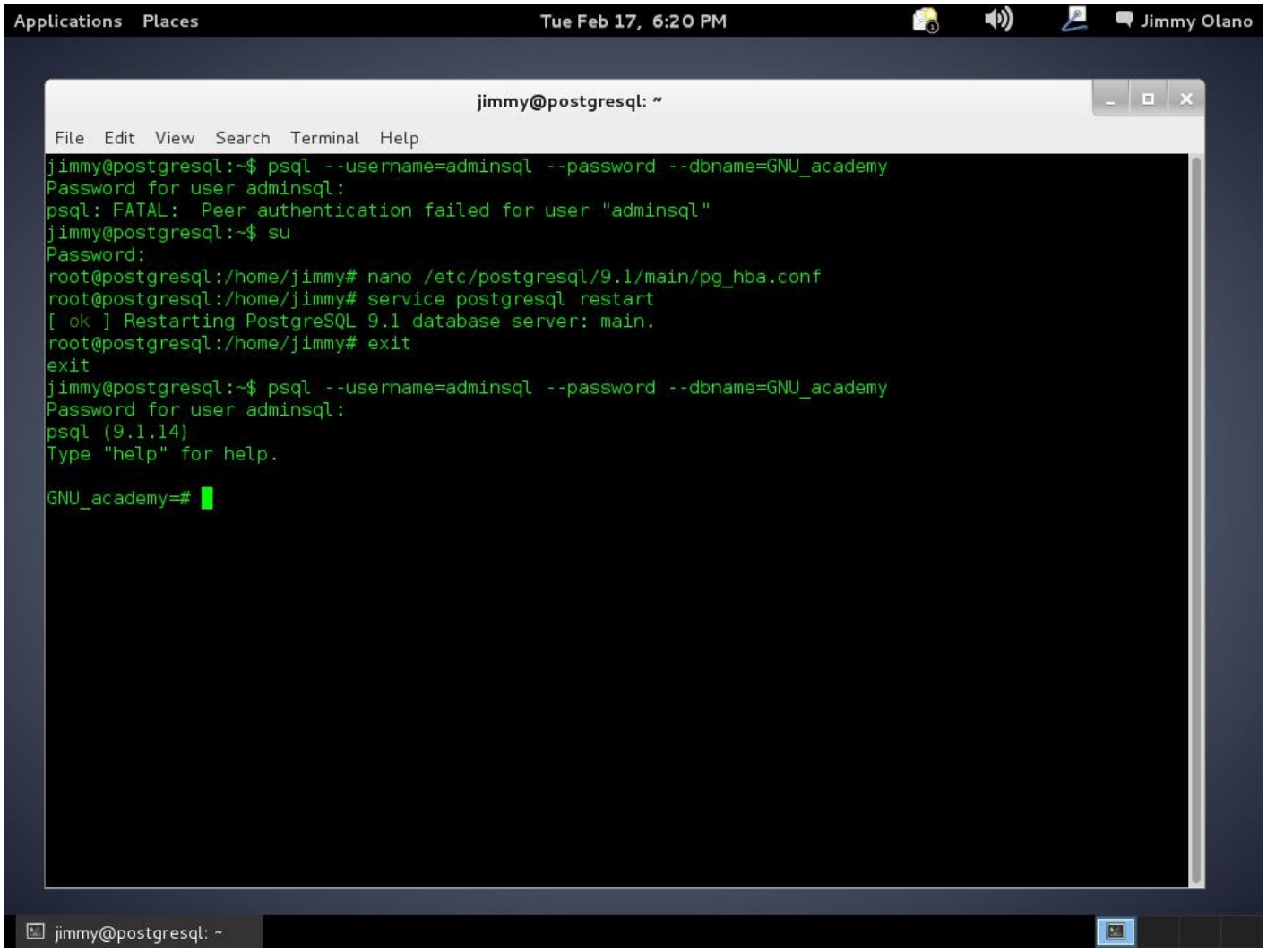
lo siguiente será (recuerden, estamos usando *nano*) CTRL+X -> "y" -> intro **osea guardar y salir en el editor de textos que hayan utilizado ustedes.** Una vez hecho esto, necesitamos que el servidor tome la nueva configuración, **en realidad el servicio o *demonio*** debemos reiniciarlo, no la máquina en si:

```
service postgresql restart
```

y si hicimos bien nuestro trabajo veremos más o menos lo siguiente:



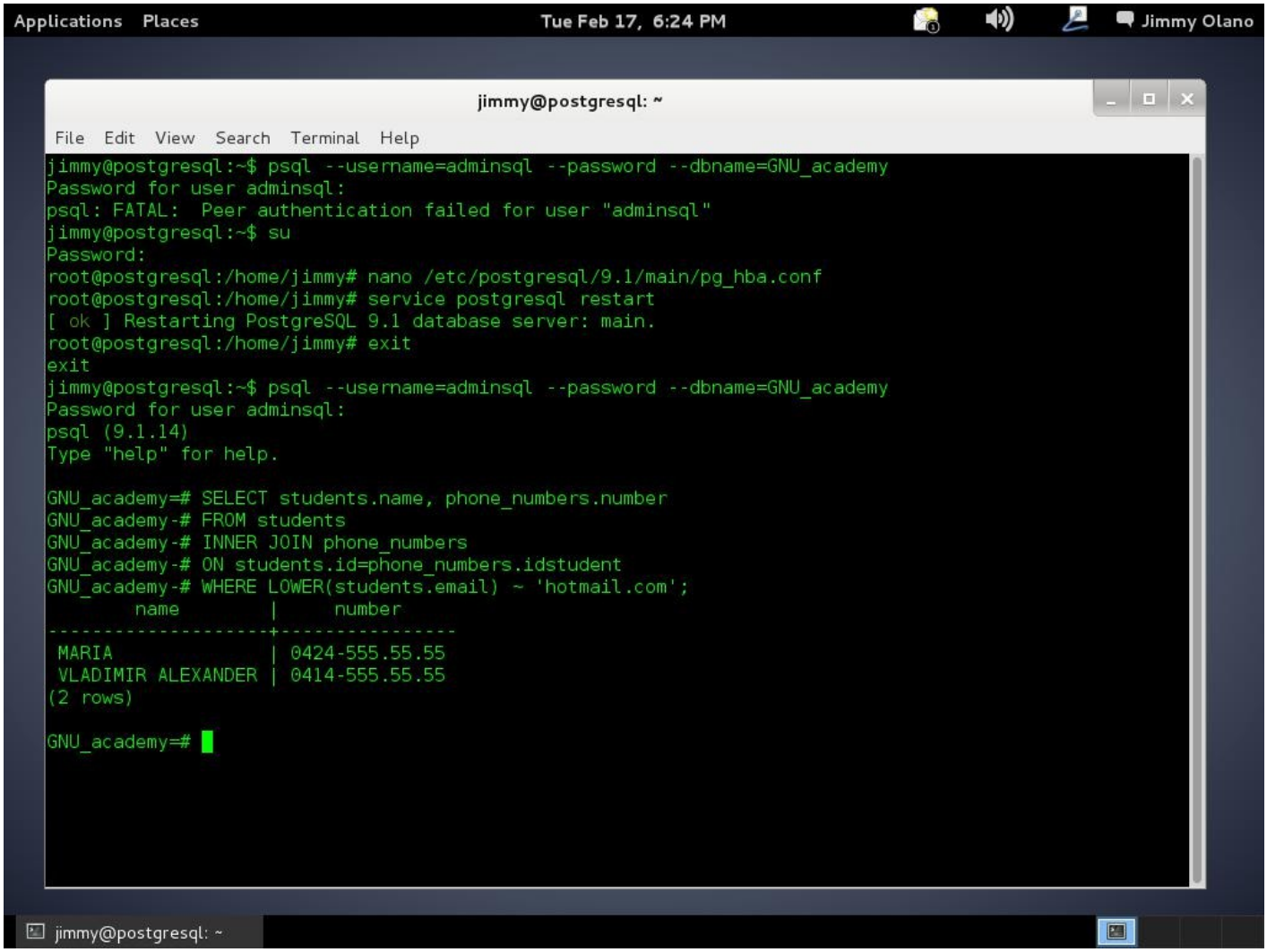
De nuevo introducimos el comando descrito para conectarnos a la base de datos pero esta vez si que tenemos éxito en nuestra tarea:



Y allí estamos, a la espera de la primera consulta, usaremos entonces la última de la entrada anterior:

```
SELECT students.name, phone_numbers.number
FROM students
INNER JOIN phone_numbers
ON students.id=phone_numbers.idstudent
WHERE LOWER(students.email) ~ ‘hotmail.com';
```

¿Recuerdan la utilidad del punto y coma para indicar el final de la consulta? Acá nos viene como anillo al dedo pues estas terminales son un tanto incómodas al estar generalmente limitadas a 80 columnas, así que presionamos *intro* y notad que el *prompt* cambia de **"=#"** a **"-="** para indicarnos que estamos en una sola sentencia a pesar que son varias lineas:



¡Y listo! hemos hecho nuestra primera consulta por *terminal interactiva*, ya sólo queda salir de *psql* y luego de nuestra terminal para ello bastará con teclear:

/quit

exit

En la próxima entrada seguiremos trabajando por línea de comandos para la creación de tablas e insertado de datos en las mismas y ampliaremos el estudio de consultas por uniones *JOIN*.

.