



UNIVERSIDAD NACIONAL EXPERIMENTAL DE GUAYANA
VICE-RECTORADO ACADÉMICO
COORDINACIÓN GENERAL DE PREGRADO
PROYECTO DE CARRERA: INGENIERÍA EN INFORMÁTICA
ASIGNATURA: BASE DE DATOS II.
SECCION 1

PROYECTO 1
(Replicación)

PRESENTADO POR:
JONATHAN CUOTTO
DEISYURIS GUZMAN
STALIN SANCHEZ
XAVIER TALAVERA

CIUDAD GUAYANA, JUNIO 2015

PLANTEAMIENTO DEL PROBLEMA

La empresa “XYZ” desea implementar un sistema de capacitación para sus empleados Tomando en cuenta las siguientes consideraciones.

- Cada curso pertenece a un área (Computación, Gerencia, RRHH, Producción) y tiene un nivel entre 1 y 3. Adicional, un curso puede ser prelado por otro curso de un nivel inferior.
- Los cursos tienen diferentes ediciones por lapso, con un empleado como instructor, el cual tiene que haber aprobado todos los cursos del área para poder dictar dicha edición. Las ediciones manejan un status (O: Ofertada; E: En Proceso, C: Cerrada, S: Suspendida).
- Los participantes de cada curso deben haber cursado las prelaaciones de ese curso. Un participante no puede cursar nuevamente un curso ya aprobado.
- Un empleado no puede participar al mismo tiempo en una edición como instructor y participante.
- Las notas de un participante pueden ser A: Aprobado; R: Reprobado.
- La empresa cuenta con una sede principal y dos sedes foráneas ubicadas en diferentes ciudades.
- Los empleados están adscritos a diferentes departamentos. Un Departamento puede tener personal en diferentes sedes.
- Un empleado tiene un status con los posibles valores (A: Activo; R: Retirado; S: Suspendido; J: Jubilado)
- Un mismo curso puede dictarse en diferentes sedes en el mismo lapso.
- Para una edición de un curso dictada en una sede sólo pueden participar empleados de esa misma sede. El instructor de la edición puede ser de una sede diferente.

Se requieren que cada grupo realice las siguientes actividades:

1. Crear la BD “cursos” utilizando PostgreSQL 9.0 o superior.

- Construcción de tablas, especificación de tipo de datos, null o not null, valores por omisión.
- Definir todas las reglas de validación que estén presentes.
- Garantizar la integridad de entidad y referencial para todas las tablas.
- Poblar la BD (Introducir datos)

2. Crear los grupos de usuario grupo_sede1, grupo_sede2, grupo_sede3 y grupo_admin.

- Los grupos de cada sede pueden leer información de todas las sedes pero sólo pueden escribir en los datos de su misma sede.
- El grupo_admin puede modificar la estructura de la bd pero no puede leer/escribir sobre la bd.
- Deben crearse los perfiles en postgres para cada grupo y restringir la conexión de la bd a los grupos a la dirección ip de su propia sede.

3. Replicación. Implemente algún mecanismo de fragmentación/replicación que permita distribuir la bd de la siguiente forma:

- En la sede principal estará el servidor principal con una copia de toda la bd.
- En cada sede foránea sólo existirá una copia de los datos de dicha sede (empleados, ediciones y participantes).

Las tablas comunes (cursos, prelaaciones, sedes, departamentos) estarán replicadas en cada bd.

SOLUCION DE REPLICACION

Se realiza la replicación a través del uso de Slony-I, el cual es un sistema de replicación de tipo Maestro – Esclavo, solución de replicación para PostgreSQL.

En el uso de la replicación se utilizan ciertos conceptos los cuales son:

- *Clúster*: Es un conjunto de instancias de base de datos PostgreSQL que están envueltos en la replicación.
- *Nodo*: Es cada una de las base de datos envueltas en la replicación.
- *Replicación set*: Es el conjunto de tablas y/o secuencias a ser replicadas.
- *Origin*: Es el nodo principal o maestro, el cual es el único en el que se puede escribir.
- *Subscribers*: Es cada uno de los nodos que no son Maestro que pertenecen al cluster, es decir los esclavos, y son aquellos que reciben los datos en la réplica.
- *Providers*: Es un nodo esclavo que sirve como proveedor para un subconjunto de nodos en el clúster, el cual actúa como un nodo origin pero no se permite a ninguna aplicación escribir en el.

Los requisitos para la replicación son que PostgreSQL acepte conexiones a través del listen_addresses y el archivo pg_hba.conf.

De igual manera hay que definir la estructura del Clúster, ya que a cada nodo del clúster se le debe asignar un número como identificador.

Lo primero que hay que hacer es crear la base de datos junto con sus tablas y restricciones en cada nodo, tanto como en el nodo Maestro como en los nodos Esclavos, y de este modo cada nodo tenga las mismas tablas en común.

Una vez esto, se procede a configurar el archivo pg_hba en la maquina Maestro, pero antes, para que todos los nodos se puedan conectar, es necesario abrir los puertos en el firewall de cada, permitiendo así la comunicación entre ellos.

Para recalcar este punto, si los nodos están conectados mediante un punto de acceso o mal llamado router, también hay que abrir el puerto 5432 al punto de acceso.

El archivo `pg_hba` se encuentra dentro de la carpeta `Data`, la cual a su vez se encuentra dentro de la carpeta original de PostgreSQL. Este archivo se abre mediante un bloc de notas y se pasa a la sección `IPv4 local connections`, que es donde se ingresan las direcciones ip de cada una de las maquinas o nodo, Maestro y esclavos. En este archivo también se indica que se usa el método `md5`. La misma configuración se debe realizar en cada una de las maquinas esclavas.

Ahora se pasa a realizar un script en el nodo Maestro mediante el cual permitirá que se realice la replicación en PostgreSQL. Este script se debe guardar dentro de la carpeta `Bin` de la carpeta original de PostgreSQL. Este paso es diferente para el Maestro de los Esclavos.

En el nodo Maestro, creamos un archivo de texto llamado `Maestro`, pasado a ser `Maestro.txt`. Primero lo que debemos hacer es definir un nombre para el clúster y de igual manera definimos los nodos, siendo el nodo Maestro siempre el nodo 1, mientras que los Esclavos serán los números consecutivos.

Pasamos a definir las propiedades de cada nodo. Cada nodo contendrá el nombre de la base de datos, la dirección IP de la maquina donde se encuentra, el usuario de la base de datos y la contraseña. Quedando la definición de los nodos de la siguiente manera:

```
node 1 admin conninfo = 'dbname= cursos host= 192.168.0.101 user =postgres
password=2222222';
```

```
node 2 admin conninfo = 'dbname= cursos host= 192.168.0.100 user =postgres
password=2222222';
```

```
#node 3 admin conninfo = 'dbname= cursos host= 192.168.0.103 user =postgres
password=1234';
```

Posteriormente hacemos uso de una línea de comando en el cual inicializamos el clúster, el cual es:

```
init cluster (id=1, comment='Sede Principal/sede1');
```

Ese comentario de nodo Maestro es como una etiqueta para identificar al nodo que pertenece al cluster.

Luego hay que crear un set de replicación, y para replicar todas las tablas que posee la base de datos, hacemos uso de la propiedad `fully qualified`, de esta manera decimos que se repliquen todas las tablas con sus atributos que contengan.

Debemos especificar la ruta, es decir el esquema en que se encuentran las tablas y el nombre de la tabla, quedando de esta manera:

```
create set (id=1, origin=1, comment='tablas de cursos');
```

```
set add table(set id=1, origin=1, id=1, fully qualified name= 'public.empleados',  
comment='tabla empleados');
```

```
set add table(set id=1, origin=1, id=2, fully qualified name= 'public.empleados_sede1',  
comment='tabla empleados sede 1');
```

```
set add table(set id=1, origin=1, id=3, fully qualified name= 'public.empleados_sede2',  
comment='tabla empleados sede 2');
```

```
set add table(set id=1, origin=1, id=4, fully qualified name= 'public.empleados_sede3',  
comment='tabla empleados sede 3');
```

```
set add table(set id=1, origin=1, id=5, fully qualified name= 'public.ediciones',  
comment='tabla ediciones');
```

```
set add table(set id=1, origin=1, id=6, fully qualified name= 'public.ediciones_sede1',  
comment='tabla ediciones sede 1');
```

```
set add table(set id=1, origin=1, id=7, fully qualified name= 'public.ediciones_sede2',  
comment='tabla ediciones sede 2');
```

```
set add table(set id=1, origin=1, id=8, fully qualified name= 'public.ediciones_sede3',  
comment='tabla ediciones sede 3');
```

```
set add table(set id=1, origin=1, id=9, fully qualified name= 'public.participantes',  
comment='tabla participantes');
```

```
set add table(set id=1, origin=1, id=10, fully qualified name= 'public.participantes_sede1',  
comment='tabla participantes sede 1');
```

```
set add table(set id=1, origin=1, id=11, fully qualified name= 'public.participantes_sede2',  
comment='tabla participantes sede 2');
```

```
set add table(set id=1, origin=1, id=12, fully qualified name= 'public.participantes_sede3',
comment='tabla participantes sede 3');
```

```
set add table(set id=1, origin=1, id=13, fully qualified name= 'public.cursos',
comment='tabla cursos');
```

```
set add table(set id=1, origin=1, id=14, fully qualified name= 'public.sedes',
comment='tabla sedes');
```

```
set add table(set id=1, origin=1, id=15, fully qualified name= 'public.departamentos',
comment='tabla departamentos');
```

```
set add table(set id=1, origin=1, id=16, fully qualified name= 'public.prelaciones',
comment='tabla sedes');
```

Por lo tanto debemos ingresar todos los parámetros necesarios, el nombre de la base de datos, el host del usuario y la contraseña, de esta manera estamos especificando el lugar donde se almacenan los cambios en la base de datos.

```
store node (id=2, comment = 'nodo esclavo', EVENT NODE =1);
#store node (id=3, comment = 'nodo esclavo', EVENT NODE =1);
```

```
store node (id=2, comment = 'Sede foranea 2/sede2', EVENT NODE =1);
#store node (id=3, comment = 'Sede foranea 3/sede3', EVENT NODE =1);
```

```
store path (server=1, client=2, conninfo='dbname= cursos host = 192.168.0.101
user=postgres password= 2222222');
store path (server=2, client=1, conninfo='dbname= cursos host = 192.168.0.100
user=postgres password= 2222222');
#store path (server=1, client=3, conninfo='dbname= cursos host = 192.168.0.101
user=postgres password= 2222222');
#store path (server=3, client=1, conninfo='dbname= cursos host = 192.168.0.103
user=postgres password= 1234');
#store path (server=2, client=3, conninfo='dbname= cursos host = 192.168.0.100
user=postgres password= 2222222');
#store path (server=3, client=2, conninfo='dbname= cursos host = 192.168.0.103
user=postgres password= 1234');
```

Ahora para indicar de que maquina a que maquina se realizara la replicación, hacemos uso del comando Store Listen, de la siguiente manera:

```
store listen (origin=1, provider=1, receiver=2);
store listen (origin=2, provider=2, receiver=1);
#store listen (origin=1, provider=1, receiver=3);
#store listen (origin=3, provider=3, receiver=1);
#store listen (origin=2, provider=2, receiver=3);
#store listen (origin=3, provider=3, receiver=2);
```

Se tiene el origen, cual es el proveedor de la replicación y cuál será el nodo que recibirá la replicación, de este modo se puede definir que el nodo 1 proveerá tanto al nodo 2 y al nodo 3 y los nodos esclavos proveerán al Maestro.

De manera parecida, se debe crear un script en un archivo txt para los esclavos, en un archivo llamado esclavo.txt. Este script tendrá de igual manera la definición tanto del cluster como de los diferentes nodos, quedando de esta manera:

```
cluster name= slony_cursos;
```

```
node 1 admin conninfo = 'dbname= cursos host= 192.168.0.101 user =postgres
password=2222222';
```

```
node 2 admin conninfo = 'dbname= cursos host= 192.168.0.100 user =postgres
password=2222222';
```

En el caso de los Esclavos, la diferencia es que debemos indicar cuál es nuestro proveedor y el id del cual se va a recibir, definiendo a los subscriptores, quedando de esta manera:

```
subscribe set (id=1, provider=1, receiver=2, forward=yes);
```

Ahora, a través de la consola de comandos de Windows, debemos ingresar a la ruta donde se encuentran nuestros archivos Maestro.txt y Esclavo.txt, ejecutándolos a través del comando SLONIK. De esta manera los nodos se sincronizan permitiendo así el inicio de la replicación.

Una vez hecho esto, en nuestro PgAdmin, en nuestra base de datos, aparece un Slony Replication, con los nodos definidos.

Luego, de nuevo en la consola de comandos de Windows, procedemos a ubicarnos en la misma carpeta BIN, hacemos uso del comando SLON, seguido del nombre del clúster

que creamos anteriormente, seguido entre comillas del nombre de la base de datos, del usuario y de la contraseña para acceder.

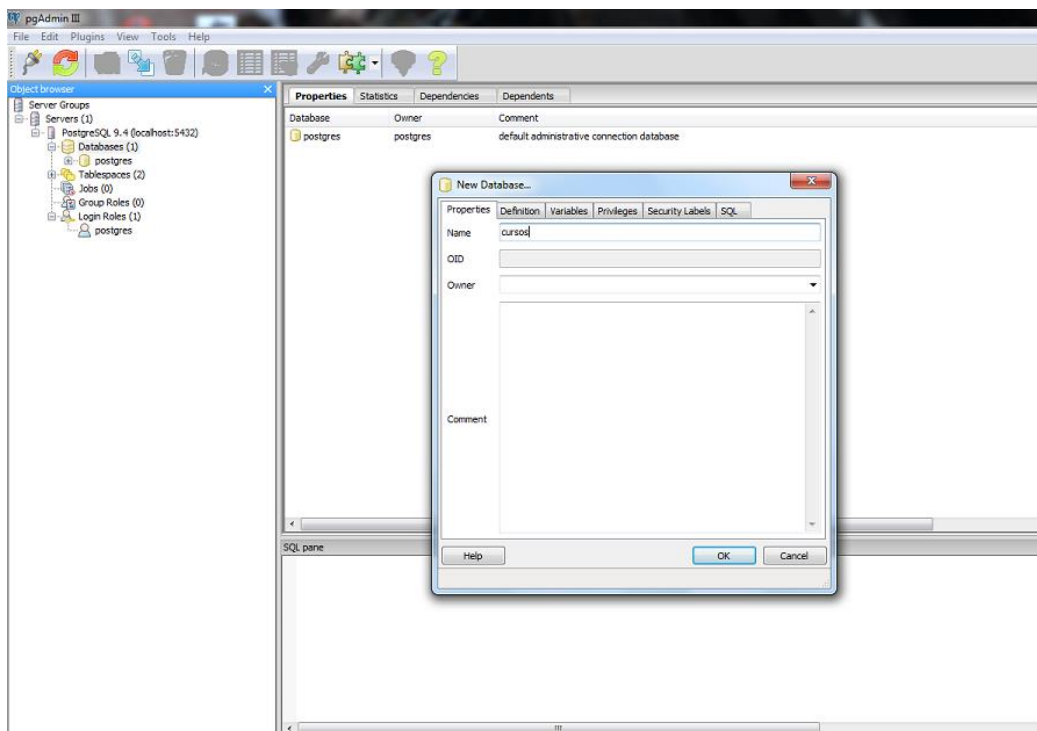
De igual manera se realiza la ejecución del clúster con el comando SLON en los nodos Esclavos con los mismos parámetros.

De esta manera procederá a replicar todos los datos desde el nodo Maestro al nodo Esclavo. Cabe destacar que esta ventana de la consola, no debe de cerrarse ya que es la que permite que los datos se repliquen.

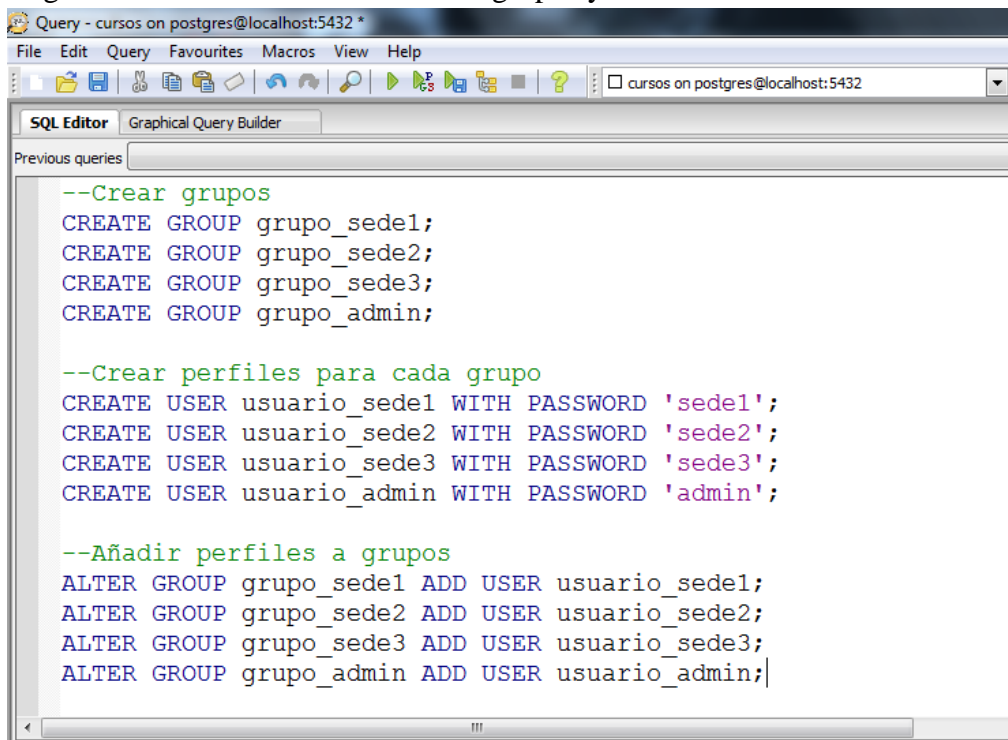
Una vez realizado todo esto, ya se puede realizar la replicación dentro de la base de datos, ingresando datos o borrándolos para su comprobación y uso adecuado según las necesidades del sistema.

PASOS DE INSTALACION

- Lo primero que hay que realizar es crear la base de datos Cursos en cada uno de los nodos:

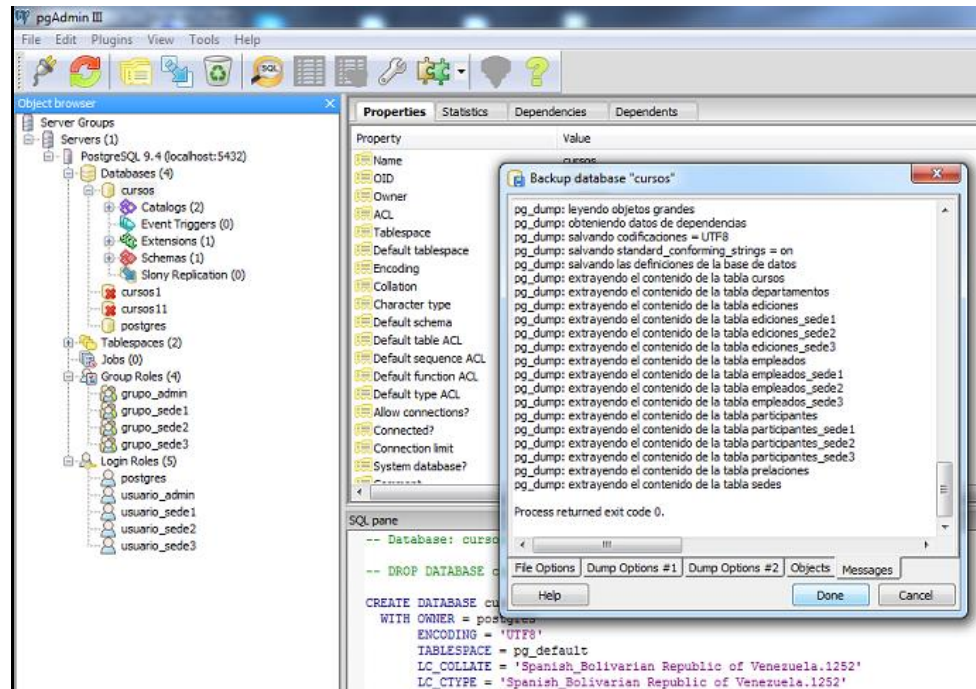


Seguidamente se crean los diferentes grupos y los usuarios

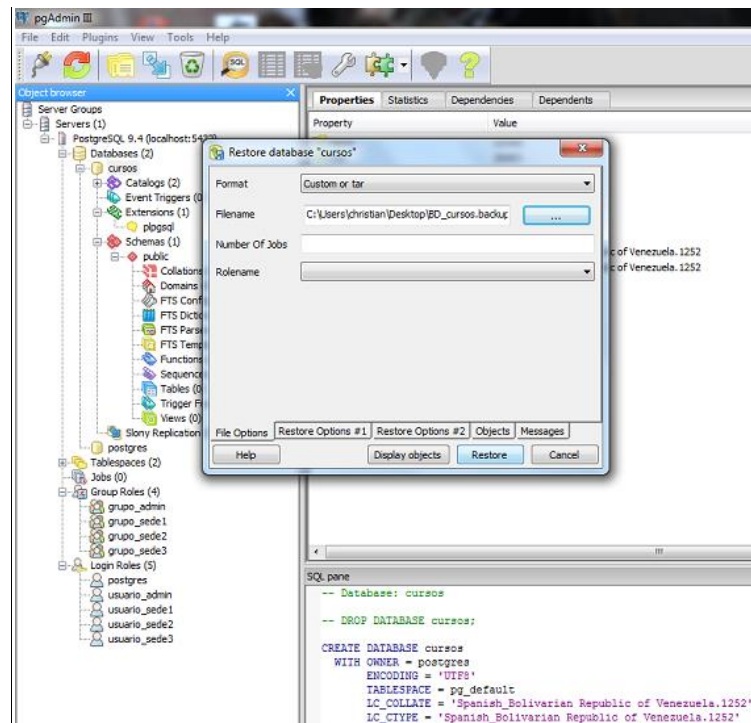


- Seguidamente procedemos a realizar la configuración de respaldo y la de restauración:

Respaldo:



Restauración:



- El siguiente paso sería configurar el archivo PG_HBA el cual se encuentra en la carpeta Data dentro de la carpeta original de PostgreSQL.

```

73 # configuration parameter, or via the -i or -h command line switches.
74
75
76
77 # TYPE      DATABASE      USER      ADDRESS      METHOD
78
79 # IPv4 local connections:
80 host        all            all        127.0.0.1/32      md5
81
82 #Sede principal 1
83 host        all            all        192.168.0.101/32   md5
84 host        all            all        192.168.0.101/24   md5
85
86 #Sede foranea 2
87 host        all            all        192.168.0.100/32   md5
88 host        all            all        192.168.0.100/24   md5
89
90 #Sede foranea 3
91 host        all            all        192.168.0.103/32   md5
92 host        all            all        192.168.0.103/24   md5
93
94 # IPv6 local connections:
95 host        all            all        ::1/128         md5
96 # Allow replication connections from localhost, by a user with the
97 # replication privilege.
98 #host        replication    postgres   127.0.0.1/32      md5
99 #host        replication    postgres   ::1/128          md5
100

```

- Luego a continuación en la carpeta BIN, se procede a crear tanto el Script Maestro como el Script Esclavo.

Maestro:

```

1 cluster name= slony_cursos;
2
3 node 1 admin conninfo = 'dbname= cursos host= 192.168.0.101 user =postgres password=2222222';
4 node 2 admin conninfo = 'dbname= cursos host= 192.168.0.100 user =postgres password=2222222';
5 #node 3 admin conninfo = 'dbname= cursos host= 192.168.0.103 user =postgres password=1234';
6
7 init cluster (id=1, comment='Sede Principal/sede1');
8
9 create set (id=1, origin=1, comment='tablas de cursos');
10
11 set add table(set id=1, origin=1, id=1, fully qualified name= 'public.empleados', comment='tabla empleados');
12 set add table(set id=1, origin=1, id=2, fully qualified name= 'public.empleados_sede1', comment='tabla empleados sede 1');
13 set add table(set id=1, origin=1, id=3, fully qualified name= 'public.empleados_sede2', comment='tabla empleados sede 2');
14 set add table(set id=1, origin=1, id=4, fully qualified name= 'public.empleados_sede3', comment='tabla empleados sede 3');
15 set add table(set id=1, origin=1, id=5, fully qualified name= 'public.ediciones', comment='tabla ediciones');
16 set add table(set id=1, origin=1, id=6, fully qualified name= 'public.ediciones_sede1', comment='tabla ediciones sede 1');
17 set add table(set id=1, origin=1, id=7, fully qualified name= 'public.ediciones_sede2', comment='tabla ediciones sede 2');
18 set add table(set id=1, origin=1, id=8, fully qualified name= 'public.ediciones_sede3', comment='tabla ediciones sede 3');
19 set add table(set id=1, origin=1, id=9, fully qualified name= 'public.participantes', comment='tabla participantes');
20 set add table(set id=1, origin=1, id=10, fully qualified name= 'public.participantes_sede1', comment='tabla participantes sede 1');
21 set add table(set id=1, origin=1, id=11, fully qualified name= 'public.participantes_sede2', comment='tabla participantes sede 2');
22 set add table(set id=1, origin=1, id=12, fully qualified name= 'public.participantes_sede3', comment='tabla participantes sede 3');
23 set add table(set id=1, origin=1, id=13, fully qualified name= 'public.cursos', comment='tabla cursos');
24 set add table(set id=1, origin=1, id=14, fully qualified name= 'public.sedes', comment='tabla sedes');
25 set add table(set id=1, origin=1, id=15, fully qualified name= 'public.departamentos', comment='tabla departamentos');
26 set add table(set id=1, origin=1, id=16, fully qualified name= 'public.prelaciones', comment='tabla sedes');
27
28 store node (id=2, comment = 'Sede foranea 2/sede2', EVENT NODE =1);
29 #store node (id=3, comment = 'Sede foranea 3/sede3', EVENT NODE =1);
30
31 store path (server=1, client=2, conninfo='dbname= cursos host = 192.168.0.101 user=postgres password= 2222222');
32 store path (server=2, client=1, conninfo='dbname= cursos host = 192.168.0.100 user=postgres password= 2222222');
33 #store path (server=1, client=3, conninfo='dbname= cursos host = 192.168.0.101 user=postgres password= 2222222');
34 #store path (server=3, client=1, conninfo='dbname= cursos host = 192.168.0.103 user=postgres password= 1234');
35
36 store listen (origin=1, provider=1, receiver=2);
37 store listen (origin=2, provider=2, receiver=1);
38 #store listen (origin=1, provider=1, receiver=3);
39 #store listen (origin=3, provider=3, receiver=1);

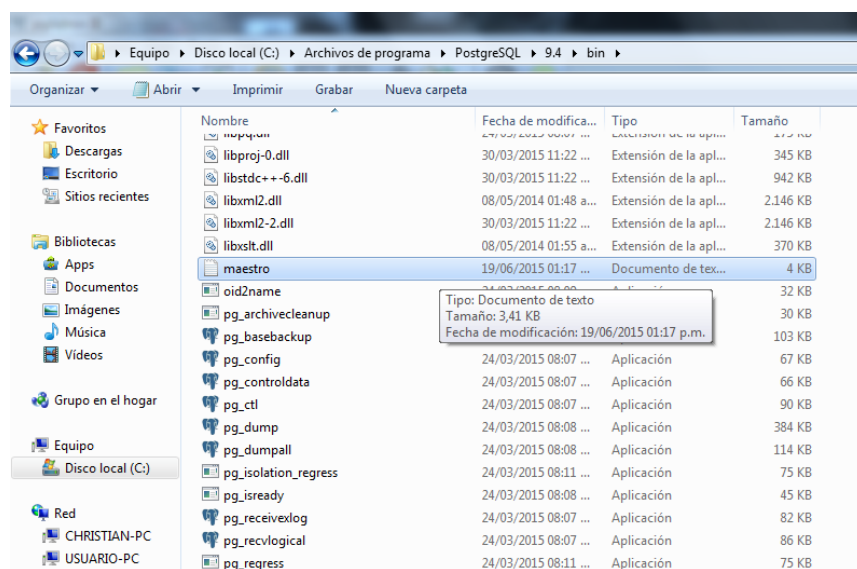
```

Esclavo:

```
1 cluster name= slony_cursos;
2
3 node 1 admin conninfo = 'dbname= cursos host= 192.168.0.101 user =postgres password=22222222';
4 node 2 admin conninfo = 'dbname= cursos host= 192.168.0.100 user =postgres password=22222222';
5
6 subscribe set (id=1, provider=1, receiver=2, forward=yes);
```

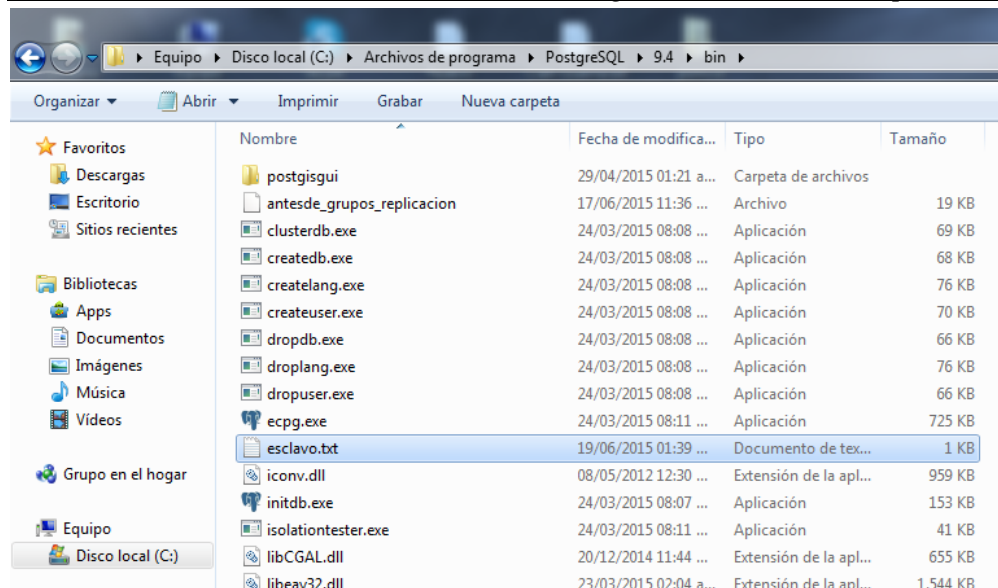
- Ahora cada archivo es colocado en su nodo correspondiente:

Archivo Maestro.txt en el nodo Maestro, en la carpeta BIN



Nombre	Fecha de modifica...	Tipo	Tamaño
libproj-0.dll	30/03/2015 11:22 ...	Extensión de la apl...	345 KB
libstdc++-6.dll	30/03/2015 11:22 ...	Extensión de la apl...	942 KB
libxml2.dll	08/05/2014 01:48 a...	Extensión de la apl...	2.146 KB
libxml2-2.dll	30/03/2015 11:22 ...	Extensión de la apl...	2.146 KB
libxslt.dll	08/05/2014 01:55 a...	Extensión de la apl...	370 KB
maestro	19/06/2015 01:17 ...	Documento de tex...	4 KB
oid2name			32 KB
pg_archivecleanup			30 KB
pg_basebackup			103 KB
pg_config	24/03/2015 08:07 ...	Aplicación	67 KB
pg_controldata	24/03/2015 08:07 ...	Aplicación	66 KB
pg_ctl	24/03/2015 08:07 ...	Aplicación	90 KB
pg_dump	24/03/2015 08:08 ...	Aplicación	384 KB
pg_dumpall	24/03/2015 08:08 ...	Aplicación	114 KB
pg_isolation_regress	24/03/2015 08:11 ...	Aplicación	75 KB
pg_isready	24/03/2015 08:08 ...	Aplicación	45 KB
pg_receivexlog	24/03/2015 08:07 ...	Aplicación	82 KB
pg_recvlogical	24/03/2015 08:07 ...	Aplicación	86 KB
pg_regress	24/03/2015 08:11 ...	Aplicación	75 KB

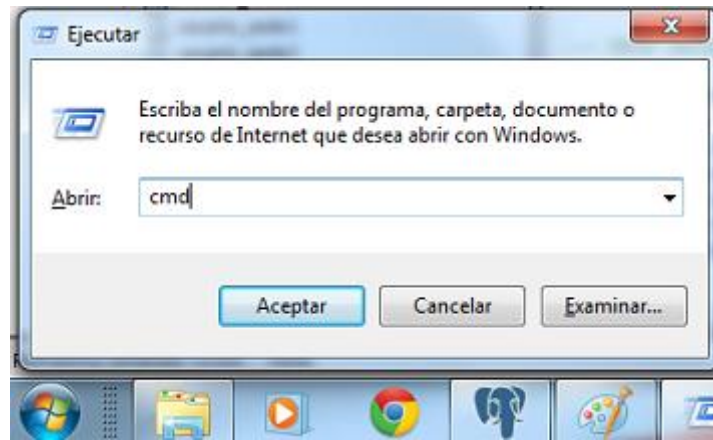
Archivo Esclavo.txt en los nodos Esclavos, de igual manera en la carpeta BIN



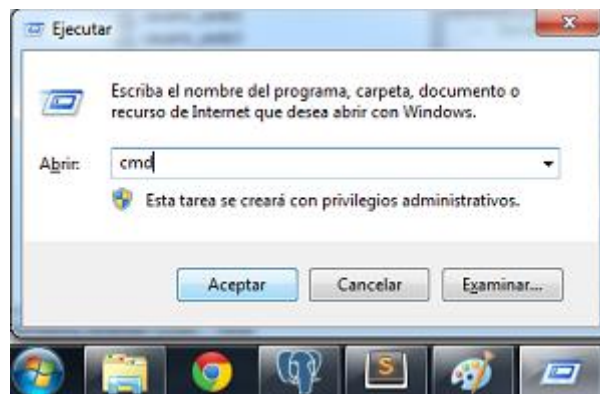
Nombre	Fecha de modifica...	Tipo	Tamaño
postgisgui	29/04/2015 01:21 a...	Carpeta de archivos	
antesde_grupos_replicacion	17/06/2015 11:36 ...	Archivo	19 KB
clusterdb.exe	24/03/2015 08:08 ...	Aplicación	69 KB
createdb.exe	24/03/2015 08:08 ...	Aplicación	68 KB
createlang.exe	24/03/2015 08:08 ...	Aplicación	76 KB
createuser.exe	24/03/2015 08:08 ...	Aplicación	70 KB
dropdb.exe	24/03/2015 08:08 ...	Aplicación	66 KB
droplang.exe	24/03/2015 08:08 ...	Aplicación	76 KB
dropuser.exe	24/03/2015 08:08 ...	Aplicación	66 KB
ecpg.exe	24/03/2015 08:11 ...	Aplicación	725 KB
esclavo.txt	19/06/2015 01:39 ...	Documento de tex...	1 KB
iconv.dll	08/05/2012 12:30 ...	Extensión de la apl...	959 KB
initdb.exe	24/03/2015 08:07 ...	Aplicación	153 KB
isolationtester.exe	24/03/2015 08:11 ...	Aplicación	41 KB
libCGAL.dll	20/12/2014 11:44 ...	Extensión de la apl...	655 KB
libeay32.dll	23/03/2015 02:04 a...	Extensión de la apl...	1.544 KB

- Se procede a abrir la consola de Windows para ejecutar mediante el comando slonik tanto el archivo Maestro como el archivo Esclavo.

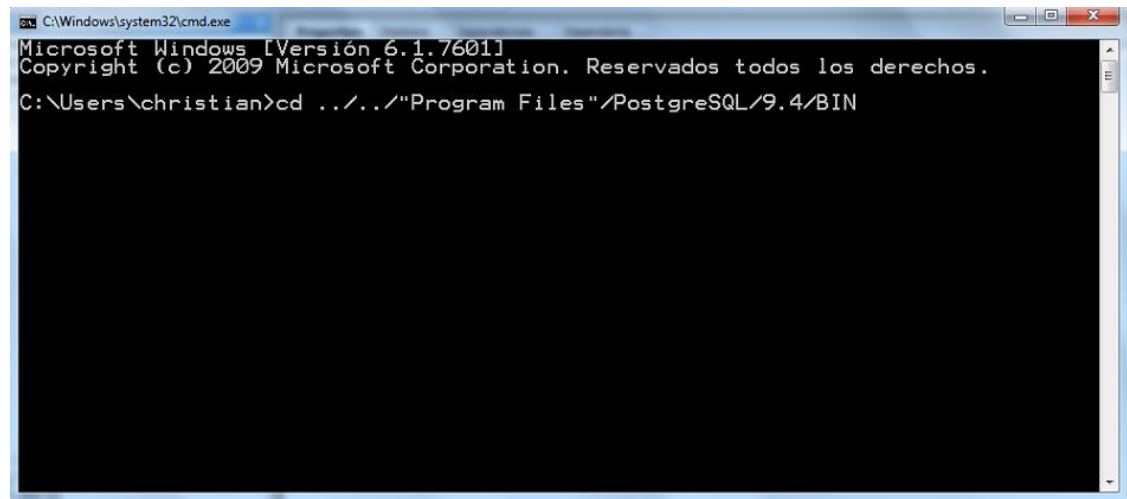
Nodo Maestro:



Nodo Esclavo:

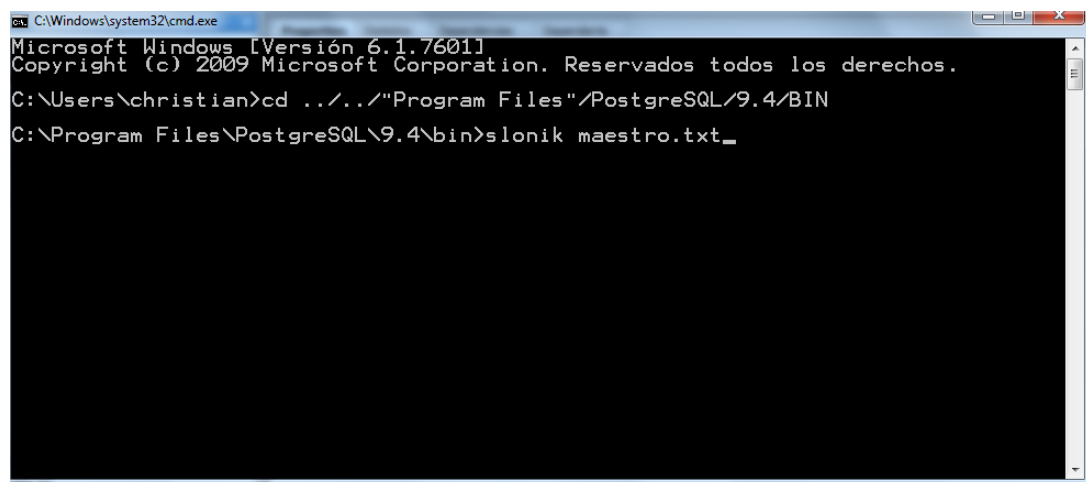


- Ahora buscamos mediante la ruta la carpeta BIN que es donde se encuentra nuestro archivo Maestro.txt en el nodo Maestro:



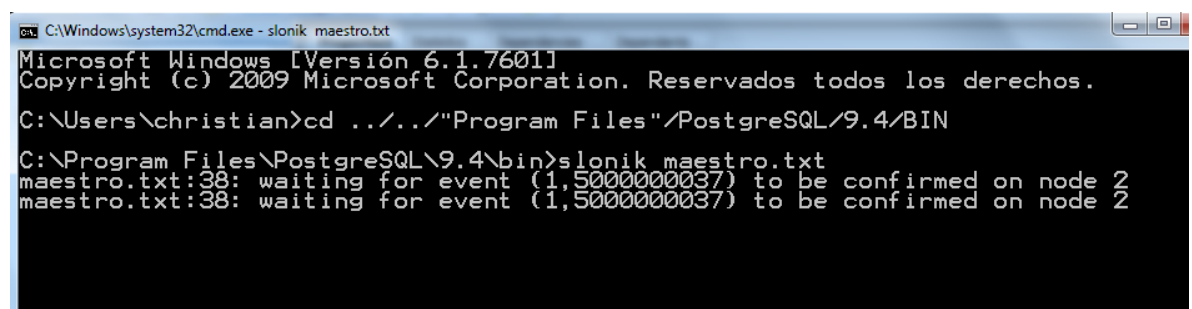
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Users\christian>cd ../../"Program Files"/PostgreSQL/9.4/BIN
```

Una vez colocado en la ruta correcta, pasamos a ejecutar el archivo Maestro.txt a través del comando SLONIK:



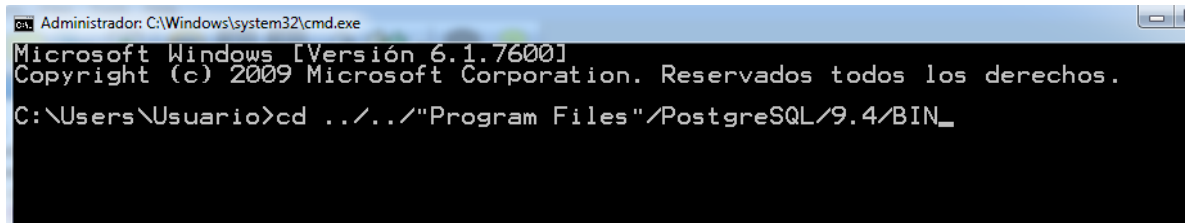
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Users\christian>cd ../../"Program Files"/PostgreSQL/9.4/BIN
C:\Program Files\PostgreSQL\9.4\bin>slonik maestro.txt_
```

Una vez ejecutado el archivo, este nodo pasara a esperar al nodo Esclavo.



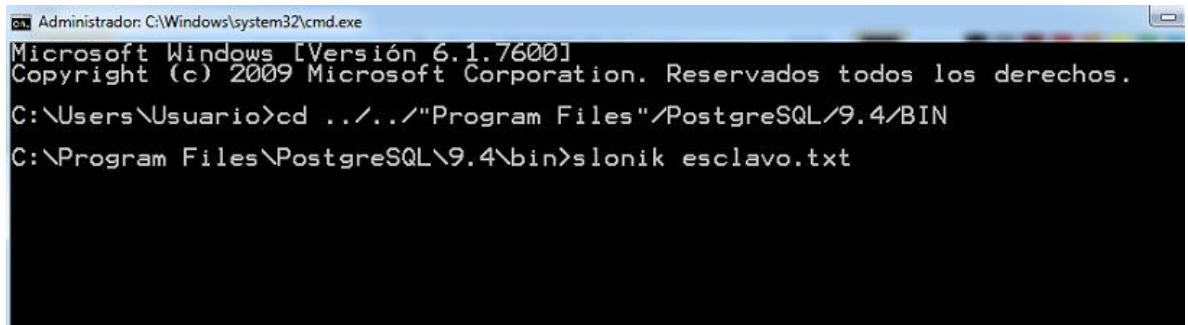
```
C:\Windows\system32\cmd.exe - slonik maestro.txt
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Users\christian>cd ../../"Program Files"/PostgreSQL/9.4/BIN
C:\Program Files\PostgreSQL\9.4\bin>slonik maestro.txt
maestro.txt:38: waiting for event (1,5000000037) to be confirmed on node 2
maestro.txt:38: waiting for event (1,5000000037) to be confirmed on node 2
```

- De igual manera en los nodos Esclavos, se procede a ejecutar mediante el comando SLONIK el archivo correspondiente, en este caso debe ser el archivo Esclavo.txt



```
Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Users\Usuario>cd ../../"Program Files"/PostgreSQL/9.4/BIN_
```

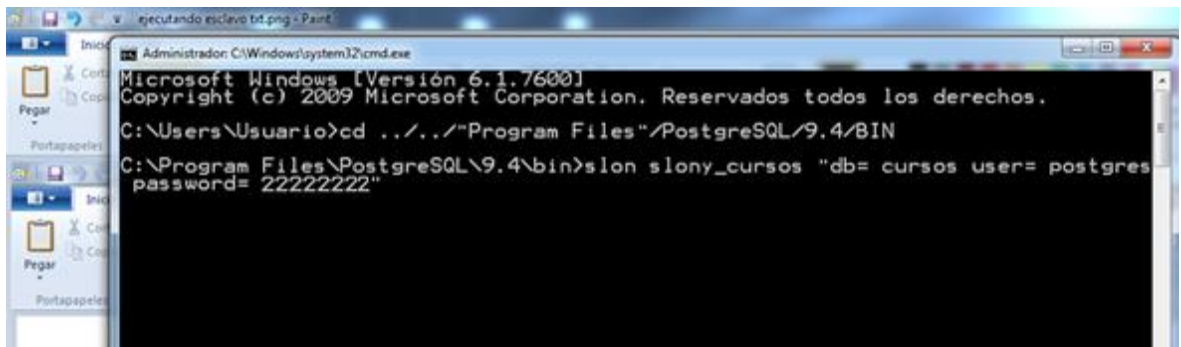
Una vez ubicada la dirección del archivo, pasamos a ejecutarlo.



```
Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Users\Usuario>cd ../../"Program Files"/PostgreSQL/9.4/BIN
C:\Program Files\PostgreSQL\9.4\bin>slonik esclavo.txt
```

Como el nodo maestro quedo en espera de los nodos Esclavos, al ejecutar el archivo Esclavo.txt, estos quedaran conectados

- Para poder iniciar la replicación, solo hacer alta ejecutar el clúster definido en los archivos Maestro.txt y Esclavo.txt



```
Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Users\Usuario>cd ../../"Program Files"/PostgreSQL/9.4/BIN
C:\Program Files\PostgreSQL\9.4\bin>slon slony_cursos "db= cursos user= postgres
password= 22222222"
```

La ejecución del Cluster, ya explicada anteriormente, se realiza mediante el comando SLON, junto a un string en el cual se indica el nombre del clúster junto al nombre de la base de datos, el usuario y la contraseña.

Cabe destacar que este la ventana que se mantiene abierta durante la ejecución del Clúster, no debe cerrarse ya que es la que permite la replicación de todos los datos.

COMPROBACION DE LA REPLICACION

Ya teniendo el Cluster ejecutandose, podemos notar que en nuestras base de datos, ahora parece un Slony Replication.

pgAdmin III

File Edit Plugins View Tools Help

Object browser

Server Groups

Servers (1)

PostgreSQL 9.4 (localhost:5432)

Databases (2)

cursos

Catalogs (3)

Event Triggers (0)

Extensions (1)

Schemas (1)

Slony Replication (1)

slony_cursos

Nodes (2)

Sede Principal/sede1

Paths (1)

Sede foranea 2/sede2

listens (1)

Sede foranea 2/sede2 (Sede foranea 2/sede2)

Replication Sets (1)

tablas de cursos

Sequences (0)

Tables (16)

public.empleados

public.empleados_sede1

public.empleados_sede2

public.empleados_sede3

public.ediciones

public.ediciones_sede1

public.ediciones_sede2

public.ediciones_sede3

public.participantes

public.participantes_sede1

public.participantes_sede2

public.participantes_sede3

public.cursos

public.sedes

public.departamentos

public.prelaciones

Subscriptions (0)

postgres

Properties

Statistics

Dependencies

Dependents

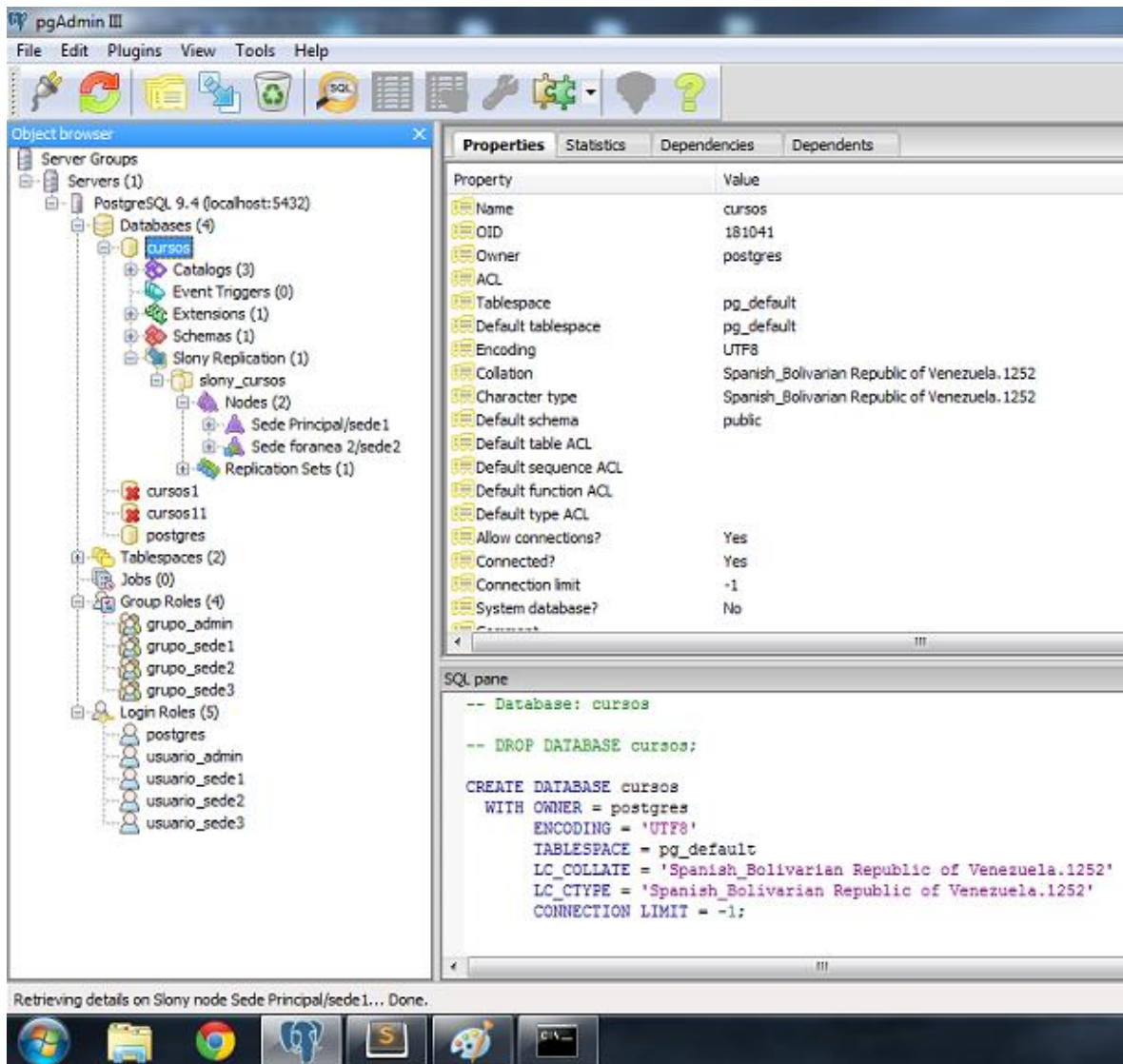
Property	Value
Name	cursos
OID	26983
Owner	postgres
ACL	
Tablespace	pg_default
Default tablespace	pg_default
Encoding	UTF8
Collation	Spanish_Bolivarian Republic of Venezuela.1252
Character type	Spanish_Bolivarian Republic of Venezuela.1252
Default schema	public
Default table ACL	
Default sequence ACL	
Default function ACL	
Default type ACL	
Allow connections?	Yes
Connected?	Yes
Connection limit	-1
System database?	No
Comment	

SQL pane

```
-- Database: cursos
-- DROP DATABASE cursos;

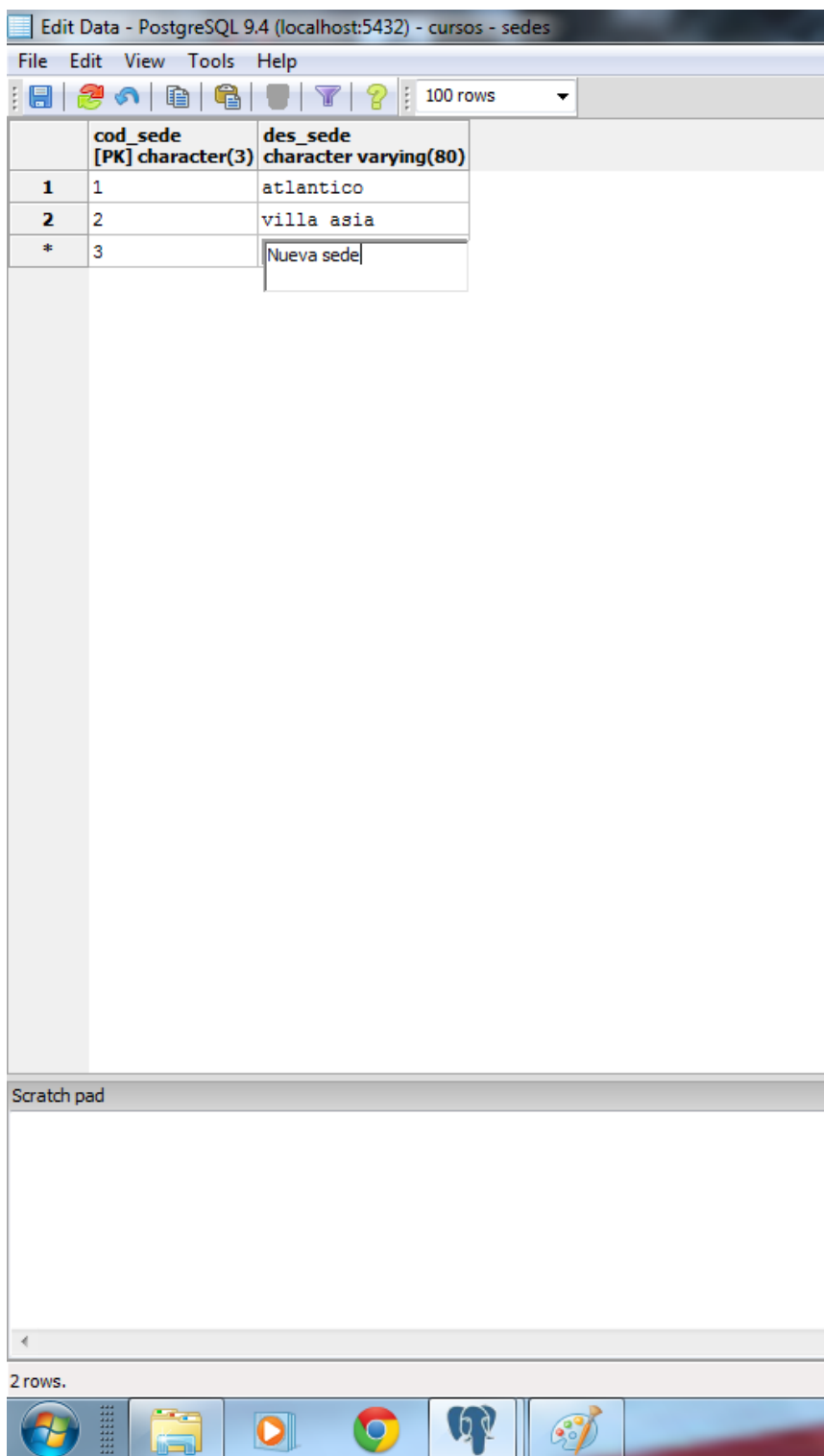
CREATE DATABASE cursos
WITH OWNER = postgres
ENCODING = 'UTF8'
TABLESPACE = pg default
```

En cual se ven todas las tablas replicadas, esto es el nodo Maestro, y de igual manera aparece en los Nodos esclavos con imagen a continuación:



Esto nos indica que la replicación se puede dar, por lo que ahora procedes a ingresar datos en nuestro nodo Maestro para observar que la replicación trabaja correctamente.

Para ello agregamos 3 sedes en la base de datos del nodo Maestro, quedando de la siguiente manera:



PostgreSQL 9.4 (localhost:5432) - cursos - sedes

File Edit View Tools Help

100 rows

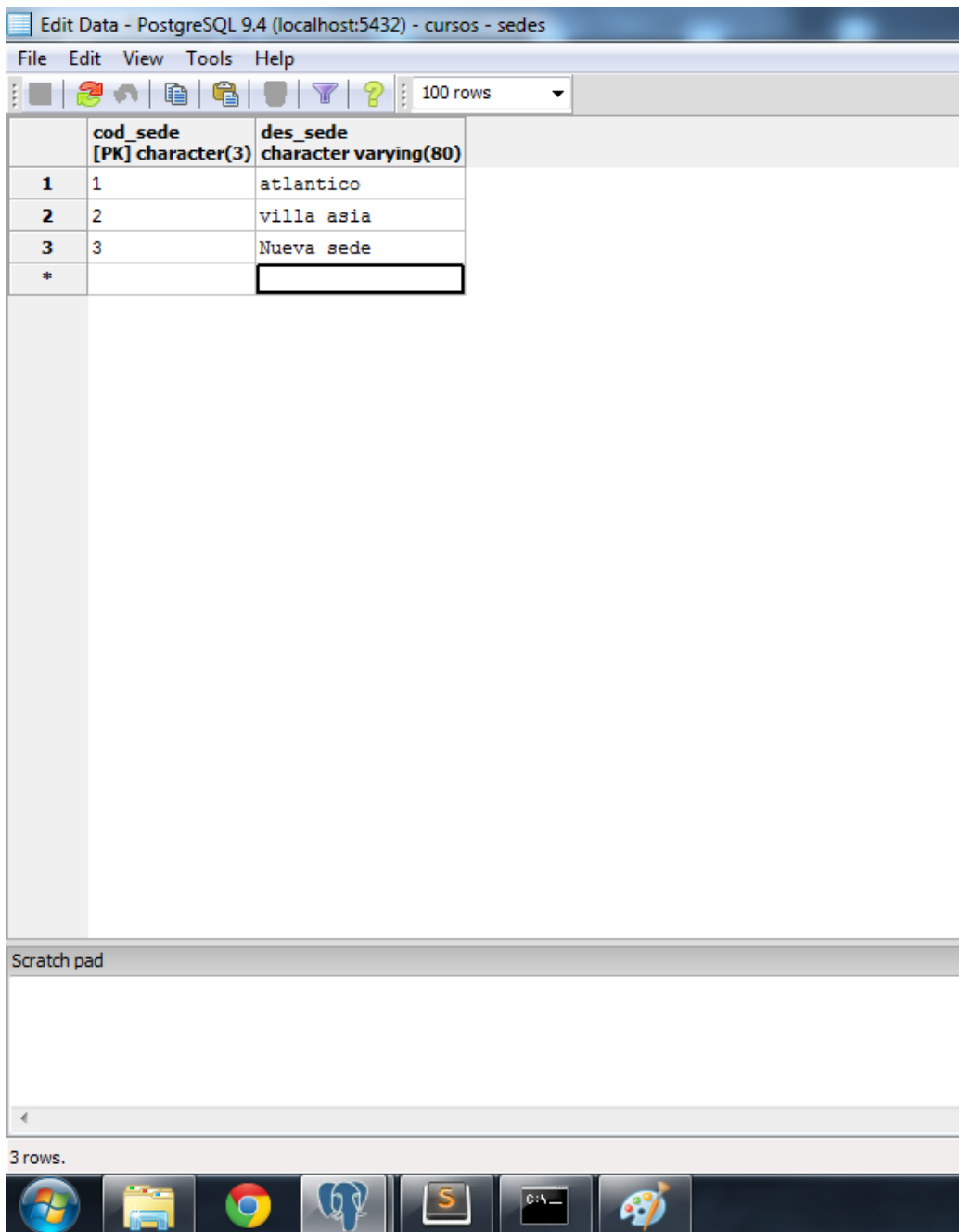
	cod_sede [PK] character(3)	des_sede character varying(80)
1	1	atlantico
2	2	villa asia
*	3	Nueva sede

Scratch pad

2 rows.

Windows taskbar icons: Start button, File Explorer, VLC media player, Google Chrome, Telegram, and Paint.

Una vez ingresadas las Sedes, procedemos a verificar en el nodo Esclavo que los datos han sido replicados:



	cod_sede [PK] character(3)	des_sede character varying(80)
1	1	atlantico
2	2	villa asia
3	3	Nueva sede
*		

Con lo que queda comprobado que la replicación ha sido exitosa.