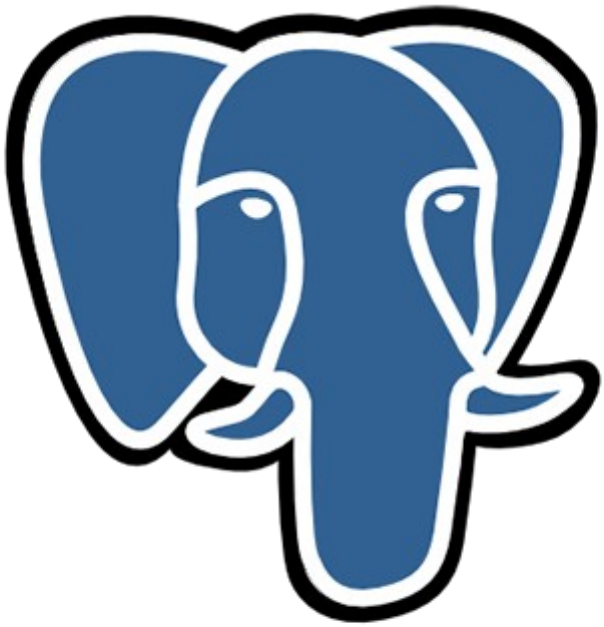
Sharding y réplica en Postgresql



PostgreSQL

Índice de contenido

1. Objetivo

El objetivo de este proyecto es el de montar un cluster de base de datos con postgresql que sea capaz de fragmentar los datos en varios servidores y a su vez estos servidores mandarán los datos que les llegan a otro servidor para realizar una réplica exacta.

Otro objetivo sería el de dejar preparado un cluster en el que, a parte del sharding y la réplica, quepa la posibilidad de añadir más funcionalidades como alta disponibilidad o balanceador de carga a través de la herramienta Pgpool-II que se usará en este proyecto.

Los componentes que se usarán para realizar la práctica serán los siguientes:

- Vagrant

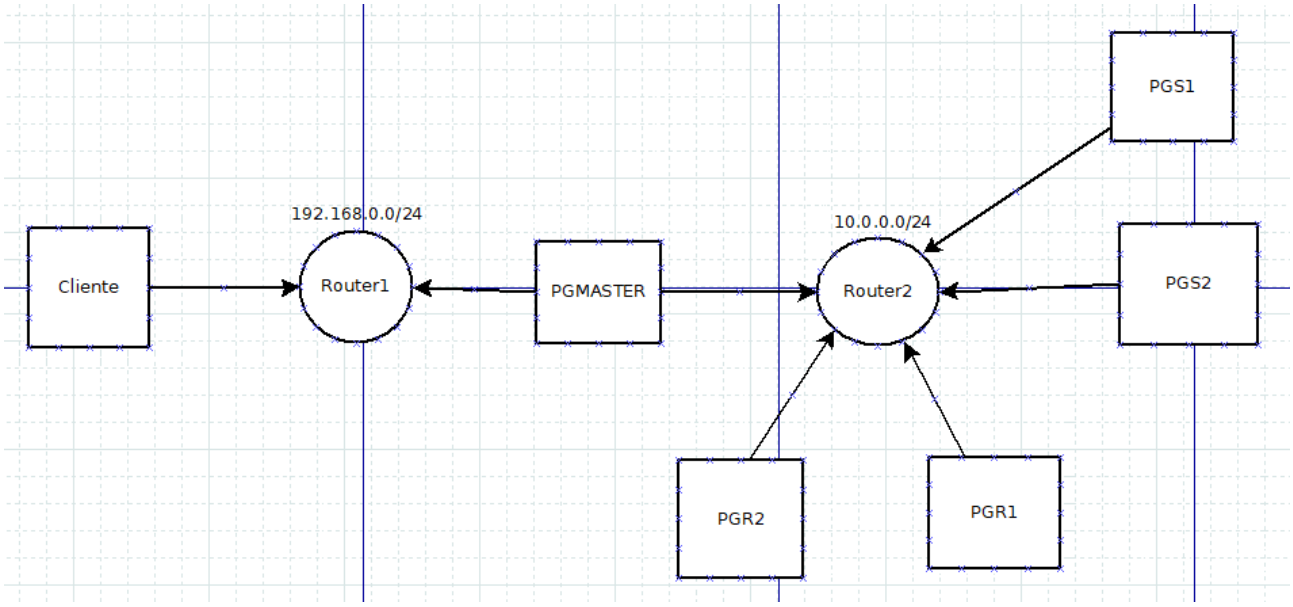
- Centos 7

- Postgresql 9.5

- Citus 6.3

- Pgpool-II

2. Descripción del esquema



En el esquema tendremos dos router y 6 hosts:

- **Router1** tendrá la red 192.168.0.0/24 y dará ip a través de DHCP. De él saldrán los nodos cliente y Pgmaster
- **Router2** tendrá la red 10.0.0.0/24. De él saldrán los nodos Pgmaster, pgs1, pgs2, pgr1, pgr2
- **Cliente**

  - eth1 => Recibirá una ip por DHCP de Router1
- **Pgmaster** tendrá dos interfaces de red:

  - eth1 => Estará conectado a Router1 y recibirá ip por DHCP

  - eth2 => Tendrá una ip fija 10.0.0.1
- **PGS1**

  - eth1 => Tendrá una ip fija 10.0.0.2
- **PGS2**

  - eth1 => Tendrá una ip fija 10.0.0.3
- **PGR1**

  - eth1 => Tendrá una ip fija 10.0.0.4
- **PGR2**

  - eth1 => Tendrá una ip fija 10.0.0.5

3. Descripción de los nodos

- **CLIENTE**=> Se usará simplemente para que se conecte al servidor Pgmaster.

- **PGMASTER** => Será el servidor maestro, el que se encarga de repartir los datos que le llegan de los clientes entre los otros dos servidores postgresql. Citus será la herramienta usada para realizar el sharding.
- **PGS1** => Tendrá instalada la extensión citus y la herramienta Pgpool-II con la que crearemos una réplica con el servidor PGR1

- **PGS2** => Tendrá instalada la extensión citus y la herramienta Pgpool-II con la que crearemos una réplica con el servidor PGR2

- **PGR1** => Tendrá instalada la extensión citus y la herramienta Pgpool-II. Esta máquina servirá para guardar la réplica de PGS1.

- **PGR2** => Tendrá instalada la extensión citus y la herramienta Pgpool-II. Esta máquina servirá para guardar la réplica de PGS2.

4. Descripción de los componentes

Vagrant

Vagrant es una herramienta para la creación y configuración de entornos de desarrollo virtualizados. Originalmente se desarrolló para VirtualBox y sistemas de configuración tales como Chef, Salt y Puppet. Sin embargo desde la versión 1.1 Vagrant es capaz de trabajar con múltiples proveedores, como Vmware, Amazon EC2, LXC, DigitalOcean, etc.Aunque Vagrant se ha desarrollado en Ruby se puede usar en multitud de proyectos escritos en otros lenguajes, tales como PHP,Python, Java, C# y JavaScript.3.4

Centos 7

Es un sistema operativo de código abierto, basado en la distribución Red Hat Enterprise Linux, operándose de manera similar, y cuyo objetivo es ofrecer al usuario un software de "clase empresarial" gratuito. Se define como robusto, estable y fácil de instalar y utilizar. Desde la versión 5, cada lanzamiento recibe soporte durante diez años, por lo que la actual versión 7 recibirá actualizaciones de seguridad hasta el 30 de junio de 2024.

Postgresql 9.5

Es un Sistema de gestión de bases de datos relacional, distribuido bajo licencia BSD y con su código fuente libre. Es el sistema de gestión de bases de datos de código abierto más potente, utiliza un modelo cliente servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afecta el resto y el sistema continúa funcionando.

Citus 6.2

Citus es una extensión de Postgresql con el que podemos hacer escalado horizontal con múltiples máquinas a través de sharding y la réplica. Nosotros nos centraremos en el sharding.

Pgpool-II

Pgpool-II es un middleware que trabaja entre servidores y clientes de PostgreSQL.

5. Sharding y Citus

5.1. Descripción y funcionamiento

El sharding es una técnica que consiste en particionar los datos de una base de datos agrupándolos en diferentes nodos, es decir, los datos de una tabla concreta o de una base de datos completa pueden estar repartidos en dos o más servidores al mismo tiempo, por lo que el usuario, al realizar una consulta se accederá a ambos servidores y este devolverá los datos solicitados como si no estuviese fragmentada la base de datos por lo que el cliente no nota nada.

Con sharding conseguimos un escalado horizontal por lo que podemos conseguir un escalado prácticamente infinito ya que podemos agregar tantos nodos como necesitemos.

Se debe realizar sharding cuando el volumen de datos es demasiado grande para una tabla concreta o una base de datos completa. Cuando una tabla es demasiado grande, más lentos son los accesos. También es bueno hacer sharding cuando el volumen de escrituras es demasiado grande para un único servidor.

Como regla general no es necesario hacer sharding desde un principio pero sí es necesario planificarlo.

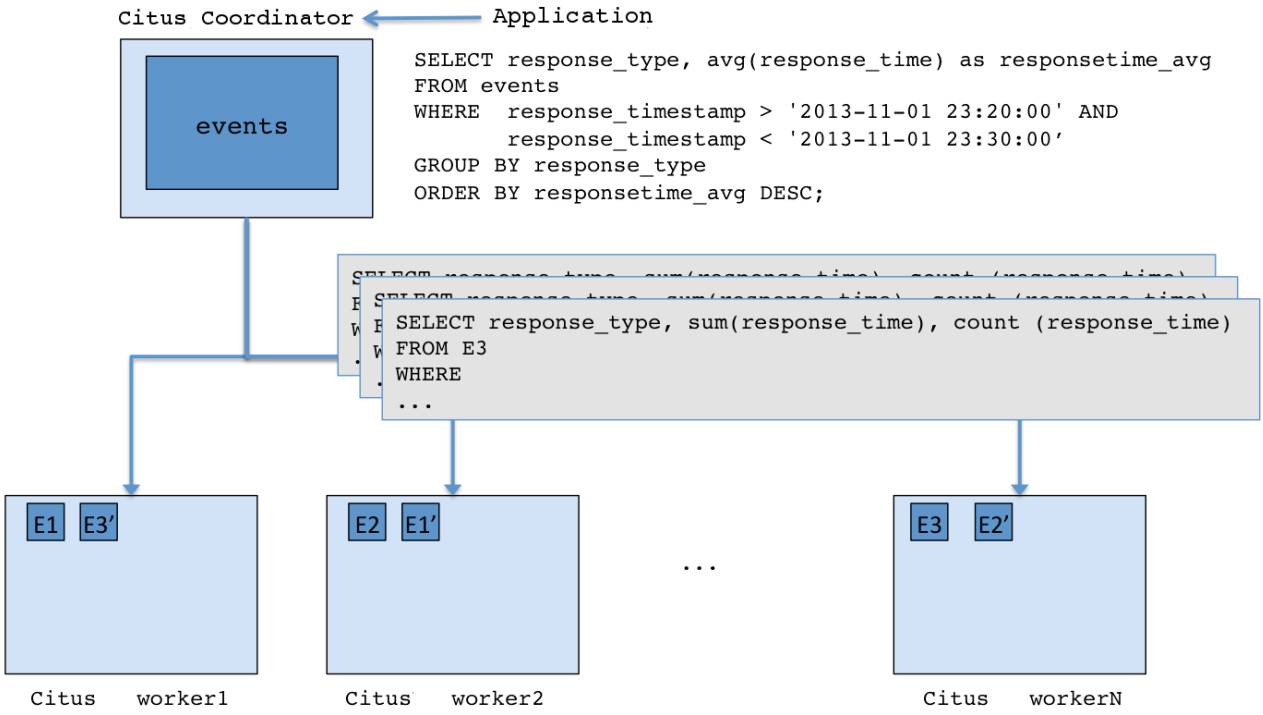
5.2. Citus

5.2.1. Citus

Citus será la herramienta que se encargará de realizar el sharding en este proyecto. Como ya hemos indicado anteriormente, citus es una extensión de código abierto para PostgreSQL con la que podremos hacer un escalado horizontal.

5.2.2. Arquitectura

Las consultas entrantes se procesan en paralelo entre los servidores.



5.2.2.1. Coordinador

El coordinador es el que se encarga de guardar los metadatos de los “worker” y los metadatos del sharding. No guarda las tablas en sí pero permite trabajar con ellas directamente desde el coordinador

5.2.2.2. Logical Sharding

Citus utiliza una arquitectura modular de bloques similar al que usa Haddop, pero usa tablas de PostgreSQL en lugar de archivos. El coordinador contiene tablas con los metadatos necesarios para rastrear a los worker y las ubicaciones de los fragmentos.

5.2.2.3. Tablas con los metadatos

El coordinador de Citus tiene varias tablas que contienen tanto datos de los worker como de los fragmentos de la base de datos. Estas tablas también tienen estadísticas como por ejemplo el tamaño de las tablas, estas tablas también se pueden replicar y restaurar por si el coordinador falla.

5.2.2.4. Consultas

Cuando enviamos una consulta, el coordinador divide en fragmentos más pequeños esta consulta y cada fragmentos puede correr independientemente sobre cada worker. Cuando los workers devuelven los datos el coordinador los fusiona y se lo muestra al usuario final.

6. Pgpool-II

6.1. Descripción

Pgpool-II es un middleware que funciona entre un servidor postgresql y un cliente. Está bajo la licencia BSD y contiene las siguientes características:

- Agrupación de Conexiones

Pgpool-II guarda las conexiones a los servidores postgresql y las reutiliza cada vez que aparece una nueva conexión con las mismas propiedades (nombre de usuario, base de datos, protocolo). Con esto se consigue que se reduzca la sobrecarga de conexiones y mejora el rendimiento del sistema.

- Réplica

Al poder administrar varios servidores postgresql, pgpool permite crear copias de seguridad en tiempo real entre estos srevidores. Con esto conseguimos que el servicio pueda continuar si uno de los servidores falla.

- Balanceador de carga

Si estamos replicando la base de datos con pgpool-II podemos activar un balanceador de carga para que reducirla carga de trabajo en los servidores postgresql.

- Límite de conexiones simultaneas

Desde el fichero de configuración de pgpool podemos configurar la cantidad máximas de conexiones simultaneas.

- Alta disponibilidad

A partir de las réplicas podemos crear más fácilmente un sistema de alta disponibilidad, por lo que, si uno de los servidores falla, podemos seguir trabajando con normalidad.

7. Configuración Nodo PGMASTER

7.1. Instalación Postgresql9.5 y Citus

Editamos el fichero hosts que se encuentra en etc y añadimos tanto pgs1 como pgs2.

```
[root@pgmaster vagrant]# vi /etc/hosts

127.0.0.1       pgmaster        pgmaster
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.0.0.2        pgs1
10.0.0.3        pgs2
```

Comprobamos que tienen conectividad.

```
[root@pgmaster vagrant]# ping pgs1
PING pgs1 (10.0.0.2) 56(84) bytes of data.
64 bytes from pgs1 (10.0.0.2): icmp_seq=1 ttl=64 time=0.440 ms
64 bytes from pgs1 (10.0.0.2): icmp_seq=2 ttl=64 time=0.289 ms
64 bytes from pgs1 (10.0.0.2): icmp_seq=3 ttl=64 time=0.264 ms
^C
--- pgs1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.264/0.331/0.440/0.077 ms
[root@pgmaster vagrant]# ping pgs2
PING pgs2 (10.0.0.3) 56(84) bytes of data.
64 bytes from pgs2 (10.0.0.3): icmp_seq=1 ttl=64 time=0.588 ms
64 bytes from pgs2 (10.0.0.3): icmp_seq=2 ttl=64 time=0.475 ms
64 bytes from pgs2 (10.0.0.3): icmp_seq=3 ttl=64 time=0.312 ms
^C
--- pgs2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.312/0.458/0.588/0.114 ms
```

Añadimos el repositorio de la versión 9.5 de postgresql.

```
yum install -y https://download.postgresql.org/pub/repos/yum/9.5/redhat/rhel-7-
x86_64/pgdg-centos95-9.5-2.noarch.rpm
```

Instalamos el paquete postgresql95-server.

```
yum install postgresql95-server -y
```

Creamos una base de datos.

```
[root@pgmaster vagrant]# /usr/pgsql-9.5/bin/postgresql95-setup initdb
Initializing database ... OK
```

Hacemos que postgresql se inicie automáticamente al iniciar la máquina.

```
[root@pgmaster vagrant]# systemctl enable postgresql-9.5
Created symlink from /etc/systemd/system/multi-user.target.wants/postgresql-
9.5.service to /usr/lib/systemd/system/postgresql-9.5.service.
```

Por último, iniciamos el servicio de postgresql.

```
[root@pgmaster vagrant]# systemctl start postgresql-9.5
```

Una vez que tengamos instalado postgresql9.5 tenemos que instalar citus tal y como nos dice la documentación oficial.

Añadimos el repositorio de citus.

```
[root@pgmaster vagrant]# curl https://install.citusdata.com/community/rpm.sh |
sudo bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  7303  100  7303    0     0  13821      0 --:--:-- --:--:-- --:--:-- 13831
Detected operating system as centos/7.
Checking for curl...
Detected curl...
Checking for postgresql10-server...
Installing pgdg10 repo... Failed to set locale, defaulting to C
done.
Downloading repository file:
https://repos.citusdata.com/community/config_file.repo?
os=centos&dist=7&source=script... done.
Installing pygpgme to verify GPG signatures... done.
Installing yum-utils... done.
Generating yum cache for citusdata_community... done.

The repository is set up! You can now install packages.
```

Instalamos citus desde el repositorio instalado anteriormente.

```
[root@pgmaster vagrant]# yum install -y citus_95.x86_64
```

7.2. Configuración Sharding

Editamos el fichero de configuración postgresql.conf y añadimos lo siguiente.

```
[root@pgmaster vagrant]# vi /var/lib/pgsql/9.5/data/postgresql.conf

listen_addresses = '*' #Hacemos que postgres sea accesible desde cualquier ip.
shared_preload_libraries = 'citus' #Hacemos que cargue la librería citus cuando
reiniciemos el servicio.
```

A continuación, editamos el fichero pg_hba.conf y lo dejamos de la siguiente forma.

```
vi /var/lib/pgsql/9.5/data/pg_hba.conf
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all             all             10.0.0.0/24             trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
```

Reiniciamos el servicio postgresql.

```
[root@pgmaster vagrant]# service postgresql-9.5 restart
Redirecting to /bin/systemctl restart postgresql-9.5.service
[root@pgmaster vagrant]#
```

A continuación, debemos crear la extensión de citus. Para ello usamos el comando “create extension”.

```
postgres=# create extension citus;
CREATE EXTENSION
```

Para comprobar que se ha creado correctamente usamos el comando “\dx”.

```
postgres=# \dx
                  List of installed extensions
  Name   | Version |   Schema   |         Description
---------+---------+------------+------------------------------
 citus   | 6.2-4   | pg_catalog | Citus distributed database
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)
```

Tenemos que decir con qué nodos va a trabajar pgmaster para el sharding. Para ello, creamos un archivo en /var/lib/pgsql/9.5/data/ indicandole los hosts y el puerto de los dos servidores postgresql.

```
[root@pgmaster vagrant]# vi /var/lib/pgsql/9.5/data/pg_worker_list.conf
pgs1    9999
pgs2    9999
```

Nos aseguramos de que el propietario del fichero sea el usuario postgres.

```
chown postgres:postgres /var/lib/pgsql/9.5/data/pg_worker_list.conf
```

También podemos añadir los nodos realizando lo siguiente. Prefiero hacerlo de este modo ya que no es necesario reiniciar postgresql

```
select * from master_add_node('pgs1', 9999);
select * from master_add_node('pgs2', 9999);
```

Reiniciamos postgresql.

```
[root@pgmaster vagrant]# service postgresql-9.5 restart
Redirecting to /bin/systemctl restart postgresql-9.5.service
```

Para comprobar que ambos hosts se han añadido hacemos la siguiente consulta.

```
postgres=# SELECT * FROM master_get_active_worker_nodes();
 node_name | node_port
-----------+-----------
 pgs2      |      9999
 pgs1      |      9999
```

8. Configuración Nodo PGS1

8.1. Instalación Postgresql9.5, Citus y Pgpool-II

Editamos el fichero hosts que se encuentra en etc y añadimos tanto pgmaster como pgr1.

```
[root@pgs1 vagrant]# vi /etc/hosts

127.0.0.1       pgmaster        pgmaster
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.0.0.1        pgmaster
10.0.0.4        pgr1
```

Añadimos el repositorio de la versión 9.5 de postgresql.

```
[root@pgs1 vagrant]# yum install
https://download.postgresql.org/pub/repos/yum/9.5/redhat/rhel-7-x86_64/pgdg-
centos95-9.5-2.noarch.rpm
```

Instalamos el paquete postgresql95-server.

```
[root@pgs1 vagrant]# yum -y install postgresql95-server
```

Creamos una base de datos.

```
[root@pgs1 vagrant]# /usr/pgsql-9.5/bin/postgresql95-setup initdb
Initializing database ... OK
```

Hacemos que postgresql se inicie automáticamente al iniciar la máquina.

```
[root@pgs1 vagrant]# systemctl enable postgresql-9.5
Created symlink from /etc/systemd/system/multi-user.target.wants/postgresql-
9.5.service to /usr/lib/systemd/system/postgresql-9.5.service.
```

Por último, iniciamos el servicio de postgresql.

```
[root@pgs1 vagrant]# systemctl start postgresql-9.5
```

Al igual que en pgmaster, instalamos citus según la documentación oficial. Añadimos el repositorio de citus.

```
[root@pgs1 vagrant]# curl https://install.citusdata.com/community/rpm.sh | sudo
bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  7303  100  7303    0     0  25661      0 --:--:-- --:--:-- --:--:-- 25624
Detected operating system as centos/7.
Checking for curl...
Detected curl...
Checking for postgresql10-server...
Installing pgdg10 repo... Failed to set locale, defaulting to C
done.
Downloading repository file:
https://repos.citusdata.com/community/config_file.repo?
os=centos&dist=7&source=script... done.
Installing pygpgme to verify GPG signatures... done.
Installing yum-utils... done.
Generating yum cache for citusdata_community... done.

The repository is set up! You can now install packages.
```

Instalamos citus desde el repositorio instalado anteriormente.

```
[root@pgs1 vagrant]# yum install -y citus_95.x86_64
```

Por último, instalamos pgpool-II para la versión 9.5 de postgresql.

```
yum install pgpool-II-95.x86_64
```

Habilitamos pgpool para que se inicie automáticamente al arrancar la máquina.

```
systemctl enable pgpool-II-95
```

8.2. Configuración Sharding

Editamos el fichero de configuración postgresql.conf y añadimos lo siguiente.

```
[root@pgs1 vagrant]# vi /var/lib/pgsql/9.5/data/postgresql.conf
listen_addresses = '*' #Hacemos que postgres sea accesible desde cualquier ip.
shared_preload_libraries = 'citus' #Hacemos que cargue la librería citus cuando
reiniciemos el servicio.
```

A continuación, editamos el fichero pg_hba.conf y lo dejamos de la siguiente forma.

```
[root@pgs1 vagrant]# vi /var/lib/pgsql/9.5/data/pg_hba.conf
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all             all             10.0.0.0/24             trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
```

Reiniciamos el servicio postgresql.

```
[root@pgs1 vagrant]# service postgresql-9.5 restart
Redirecting to /bin/systemctl restart postgresql-9.5.service
```

A continuación, debemos crear la extensión de citus. Para ello usamos el comando “create extension”.

```
postgres=# create extension citus;
CREATE EXTENSION
```

Para comprobar que se ha creado correctamente usamos el comando “\dx”.

```
postgres=# \dx
                 List of installed extensions
  Name   | Version |   Schema   |         Description
---------+---------+------------+------------------------------
 citus   | 6.2-4   | pg_catalog | Citus distributed database
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)
```

8.2. Configuración Pgpool-II

Una vez lo tengamos instalado nos colocamos en /etc/pgpool-II-95 y editamos el fichero pgpool.conf editando lo siguiente.

```
[root@pgs1 pgpool-II-95]# cd /etc/pgpool-II-95/
[root@pgs1 pgpool-II-95]# cp pgpool.conf.sample pgpool.conf
[root@pgs1 pgpool-II-95]# vi pgpool.conf

listen_addresses = '*'
port = 9999

backend_hostname0 = 'pgs1'
backend_port0 = 5432
backend_data_directory0 = '/var/lib/pgsql/9.5/data'

backend_hostname1 = 'pgr1'
backend_port1 = 5432
backend_data_directory1 = '/var/lib/pgsql/9.5/data'

replication_mode = true
```

Por último, reiniciamos el servicio pgpool-II

```
systemctl restart pgpool-II-95
```

9. Configuración Nodo PGS2

9.1. Instalación Postgresql9.5, Citus y Pgpool-II

Editamos el fichero hosts que se encuentra en etc y añadimos tanto pgmaster como pgr2.

```
[root@pgs2 vagrant]# vi /etc/hosts

127.0.0.1       pgmaster        pgmaster
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.0.0.1        pgmaster
10.0.0.5        pgr2
```

Añadimos el repositorio de la versión 9.5 de postgresql.

```
[root@pgs2 vagrant]# yum install
https://download.postgresql.org/pub/repos/yum/9.5/redhat/rhel-7-x86_64/pgdg-
centos95-9.5-2.noarch.rpm
```

Instalamos el paquete postgresql95-server.

```
[root@pgs2 vagrant]# yum install postgresql95-server
```

Creamos una base de datos.

```
[root@pgs2 vagrant]# /usr/pgsql-9.5/bin/postgresql95-setup initdb
Initializing database ... OK
```

Hacemos que postgresql se inicie automáticamente al iniciar la máquina.

```
[root@pgs2 vagrant]# systemctl enable postgresql-9.5
Created symlink from /etc/systemd/system/multi-user.target.wants/postgresql-
9.5.service to /usr/lib/systemd/system/postgresql-9.5.service.
```

Por último, iniciamos el servicio de postgresql.

```
[root@pgs2 vagrant]# systemctl start postgresql-9.5
```

Al igual que en pgmaster, instalamos citus según la documentación oficial. Añadimos el repositorio de citus.

```
[root@pgs2 vagrant]# curl https://install.citusdata.com/community/rpm.sh | sudo
bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  7303  100  7303    0     0  16264      0 --:--:-- --:--:-- --:--:-- 16301
Detected operating system as centos/7.
Checking for curl...
Detected curl...
Checking for postgresql10-server...
Installing pgdg10 repo... done.
Downloading repository file:
https://repos.citusdata.com/community/config_file.repo?
os=centos&dist=7&source=script... done.
Installing pygpgme to verify GPG signatures... done.
Installing yum-utils... done.
Generating yum cache for citusdata_community... done.
The repository is set up! You can now install packages.
```

Instalamos citus desde el repositorio instalado anteriormente.

```
[root@pgs2 vagrant]# yum install -y citus_95.x86_64
```

9.2. Configuración Sharding

Editamos el fichero de configuración postgresql.conf y añadimos lo siguiente.

```
[root@pgs2 vagrant]# vi /var/lib/pgsql/9.5/data/postgresql.conf
listen_addresses = '*' #Hacemos que postgres sea accesible desde cualquier ip.
shared_preload_libraries = 'citus' #Hacemos que cargue la librería citus cuando
reiniciemos el servicio.
```

A continuación, editamos el fichero pg_hba.conf y lo dejamos de la siguiente forma.

```
[root@pgs2 vagrant]# vi /var/lib/pgsql/9.5/data/pg_hba.conf
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all             all             10.0.0.0/24             trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
```

Reiniciamos el servicio postgresql.

```
[root@pgs2 vagrant]# service postgresql-9.5 restart
Redirecting to /bin/systemctl restart postgresql-9.5.service
```

A continuación, debemos crear la extensión de citus. Para ello usamos el comando “create extension”.

```
postgres=# create extension citus;
CREATE EXTENSION
```

Para comprobar que se ha creado correctamente usamos el comando “\dx”.

```
postgres=# \dx
                 List of installed extensions
  Name   | Version |   Schema   |         Description
---------+---------+------------+------------------------------
 citus   | 6.2-4   | pg_catalog | Citus distributed database
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)
```

9.2. Configuración Pgpool-II

Una vez lo tengamos instalado nos colocamos en /etc/pgpool-II-95 y editamos el fichero pgpool.conf editando lo siguiente.

```
[root@pgs2 pgpool-II-95]# cd /etc/pgpool-II-95/
[root@pgs2 pgpool-II-95]# cp pgpool.conf.sample pgpool.conf
[root@pgs2 pgpool-II-95]# vi pgpool.conf

listen_addresses = '*'
port = 9999

backend_hostname0 = 'pgs2'
backend_port0 = 5432
backend_data_directory0 = '/var/lib/pgsql/9.5/data'

backend_hostname1 = 'pgr2'
backend_port1 = 5432
backend_data_directory1 = '/var/lib/pgsql/9.5/data'

replication_mode = true
```

Por último, reiniciamos el servicio pgpool-II

```
systemctl restart pgpool-II-95
```

10. Configuración Nodo PGR1

10.1. Instalación Postgresql9.5 y Pgpool-II

Editamos el fichero hosts que se encuentra en etc y añadimos tanto pgmaster como pgs1.

```
[root@pgr1 vagrant]# vi /etc/hosts

127.0.0.1       pgmaster        pgmaster
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.0.0.1        pgmaster
10.0.0.2        pgs1
```

Añadimos el repositorio de la versión 9.5 de postgresql.

```
[root@pgr1 vagrant]# yum install
https://download.postgresql.org/pub/repos/yum/9.5/redhat/rhel-7-x86_64/pgdg-
centos95-9.5-2.noarch.rpm
```

Instalamos el paquete postgresql95-server.

```
[root@pgr1 vagrant]# yum install postgresql95-server
```

Creamos una base de datos.

```
[root@pgr1 vagrant]# /usr/pgsql-9.5/bin/postgresql95-setup initdb
Initializing database ... OK
```

Hacemos que postgresql se inicie automáticamente al iniciar la máquina.

```
[root@pgr1 vagrant]# systemctl enable postgresql-9.5
Created symlink from /etc/systemd/system/multi-user.target.wants/postgresql-
9.5.service to /usr/lib/systemd/system/postgresql-9.5.service.
```

Iniciamos el servicio de postgresql.

```
[root@pgr1 vagrant]# systemctl start postgresql-9.5
```

Instalamos pgpool-II para la versión 9.5 de postgresql.

```
[root@pgr1 vagrant]# yum install pgpool-II-95.x86_64
```

Habilitamos pgpool para que se inicie automáticamente al arrancar la máquina.

```
[root@pgr1 vagrant]# systemctl enable pgpool-II-95
```

10.2. Configuración Sharding

Editamos el fichero de configuración postgresql.conf y añadimos lo siguiente.

```
[root@pgr1 vagrant]# vi /var/lib/pgsql/9.5/data/postgresql.conf
listen_addresses = '*' #Hacemos que postgres sea accesible desde cualquier ip.
shared_preload_libraries = 'citus' #Hacemos que cargue la librería citus cuando
reiniciemos el servicio.
```

A continuación, editamos el fichero pg_hba.conf y lo dejamos de la siguiente forma.

```
[root@pgr1 vagrant]# vi /var/lib/pgsql/9.5/data/pg_hba.conf
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all             all             10.0.0.0/24             trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
```

Reiniciamos el servicio postgresql.

```
[root@pgr1 vagrant]# service postgresql-9.5 restart
Redirecting to /bin/systemctl restart postgresql-9.5.service
```

A continuación, debemos crear la extensión de citus. Para ello usamos el comando “create extension”.

```
postgres=# create extension citus;
CREATE EXTENSION
```

Para comprobar que se ha creado correctamente usamos el comando “\dx”.

```
postgres=# \dx
                 List of installed extensions
  Name   | Version |   Schema   |         Description
---------+---------+------------+------------------------------
 citus   | 6.2-4   | pg_catalog | Citus distributed database
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)
```

10.3. Configuración Pgpool-II

Una vez lo tengamos instalado nos colocamos en /etc/pgpool-II-95 y editamos el fichero pgpool.conf editando lo siguiente.

```
[root@pgr1 pgpool-II-95]# cd /etc/pgpool-II-95/
[root@pgr1 pgpool-II-95]# cp pgpool.conf.sample pgpool.conf
```

```
[root@pgr1 pgpool-II-95]# vi pgpool.conf
listen_addresses = '*'
port = 9999

backend_hostname0 = 'pgr1'
backend_port0 = 5432
backend_data_directory0 = '/var/lib/pgsql/9.5/data'

backend_hostname1 = 'pgs1'
backend_port1 = 5432
backend_data_directory1 = '/var/lib/pgsql/9.5/data'

replication_mode = true
```

Por último, reiniciamos el servicio pgpool-II

```
systemctl restart pgpool-II-95
```

11. Configuración Nodo PGR2

11.1. Instalación Postgresql9.5 y Pgpool-II

Editamos el fichero hosts que se encuentra en etc y añadimos tanto pgmaster como pgs2.

```
[root@pgr2 vagrant]# vi /etc/hosts

127.0.0.1       pgmaster        pgmaster
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.0.0.1        pgmaster
10.0.0.3        pgs2
```

Añadimos el repositorio de la versión 9.5 de postgresql.

```
[root@pgr2 vagrant]# yum install
https://download.postgresql.org/pub/repos/yum/9.5/redhat/rhel-7-x86_64/pgdg-
centos95-9.5-2.noarch.rpm
```

Instalamos el paquete postgresql95-server.

```
[root@pgr2 vagrant]# yum install postgresql95-server
```

Creamos una base de datos.

```
[root@pgr2 vagrant]# /usr/pgsql-9.5/bin/postgresql95-setup initdb
Initializing database ... OK
```

Hacemos que postgresql se inicie automáticamente al iniciar la máquina.

```
[root@pgr2 vagrant]# systemctl enable postgresql-9.5
Created symlink from /etc/systemd/system/multi-user.target.wants/postgresql-
9.5.service to /usr/lib/systemd/system/postgresql-9.5.service.
```

Por último, iniciamos el servicio de postgresql.

```
[root@pgr2 vagrant]# systemctl start postgresql-9.5
```

Instalamos pgpool-II para la versión 9.5 de postgresql.

```
[root@pgr2 vagrant]# yum install pgpool-II-95.x86_64
```

Habilitamos pgpool para que se inicie automáticamente al arrancar la máquina.

```
[root@pgr2 vagrant]# systemctl enable pgpool-II-95
```

11.2. Configuración Sharding

Editamos el fichero de configuración postgresql.conf y añadimos lo siguiente.

```
[root@pgr2 vagrant]# vi /var/lib/pgsql/9.5/data/postgresql.conf
listen_addresses = '*' #Hacemos que postgres sea accesible desde cualquier ip.
shared_preload_libraries = 'citus' #Hacemos que cargue la librería citus cuando
reiniciemos el servicio.
```

A continuación, editamos el fichero pg_hba.conf y lo dejamos de la siguiente forma.

```
[root@pgr2 vagrant]# vi /var/lib/pgsql/9.5/data/pg_hba.conf
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all             all             10.0.0.0/24             trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
```

Reiniciamos el servicio postgresql.

```
[root@pgr2 vagrant]# service postgresql-9.5 restart
Redirecting to /bin/systemctl restart postgresql-9.5.service
```

A continuación, debemos crear la extensión de citus. Para ello usamos el comando “create extension”.

```
postgres=# create extension citus;
CREATE EXTENSION
```

Para comprobar que se ha creado correctamente usamos el comando “\dx”.

```
postgres=# \dx
                 List of installed extensions
  Name   | Version |   Schema   |         Description
---------+---------+------------+------------------------------
 citus   | 6.2-4   | pg_catalog | Citus distributed database
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)
```

11.3. Configuración Pgpool-II

Una vez lo tengamos instalado nos colocamos en /etc/pgpool-II-95 y editamos el fichero pgpool.conf editando lo siguiente.

```
[root@pgs2 pgpool-II-95]# cd /etc/pgpool-II-95/
[root@pgs2 pgpool-II-95]# cp pgpool.conf.sample pgpool.conf
```

```
[root@pgs2 pgpool-II-95]# vi pgpool.conf
listen_addresses = '*'
port = 9999

backend_hostname0 = 'pgr2'
backend_port0 = 5432
backend_data_directory0 = '/var/lib/pgsql/9.5/data'

backend_hostname1 = 'pgs2'
backend_port1 = 5432
backend_data_directory1 = '/var/lib/pgsql/9.5/data'

replication_mode = true
```

Por último, reiniciamos el servicio pgpool-II

```
systemctl restart pgpool-II-95
```

12. Comprobaciones

Usaremos los siguientes datos para hacer las comprobaciones.

```
create table Asociaciones(
CIF               varchar(9),
Denominacion      varchar(100),
Direccion         varchar(100),
Provincia         varchar(100),
Declarada         boolean,
Fecha_Fundacion date,
constraint pk_Asociacion primary key(CIF),
constraint CIF_ok check(CIF ~'^[K,Q,S]{1}[0-9]{7}[A-Z]' or CIF ~'^[A,B,E,H]{1}
[0-9]{8}' or CIF ~'^[C,D,F,G,I,J,L-P,R,T-Z]{1}[0-9]{7}[A-Z0-9]')
);
insert into Asociaciones values(
      'G81164105',
      'Acción Contra el Hambre',
      'Duque de Sevilla 3,28002',
      'Madrid',
      True,
      '04/05/2001'
);

insert into Asociaciones values(
      'G58277534',
      'Médicos sin Fronteras',
      'Nou de la Rambla 26, 08001',
      'Barcelona',
      True,
      '21/02/1971'
);

insert into Asociaciones values(
      'G83323683',
      '1 Kilo de Ayuda',
      'Pedro Unanue 14, 28045',
      'Madrid',
      True,
      '06/11/2009'
);

insert into Asociaciones values(
      'R2800560A',
      'Cáritas',
      'Embajadores 162, 28045',
      'Madrid',
      True,
      '13/06/1897'
);

insert into Asociaciones values(
      'Q2866001G',
      'Cruz Roja',
      'Avenida de la Cruz Roja S/N, 41009',
      'Sevilla',
      True,
      '15/04/1970'
);
```

Lo primero que debemos hacer crear la tabla.

```
postgres=# create table Asociaciones(
postgres(# CIF varchar(9),
postgres(# Denominacion varchar(100),
postgres(# Direccion
postgres(# Direccion varchar(100),
postgres(# Provincia
postgres(# Provincia varchar(100),
postgres(# Declarada
postgres(# Declarada boolean,
postgres(# Fecha_Fundacion date,
postgres(# constraint pk_Asociacion primary key(CIF),
postgres(# constraint CIF_ok check(CIF ~'^[K,Q,S]{1}[0-9]{7}[A-Z]' or CIF
~'^[A,B,E,H]{1}[0-9]{8}' or CIF ~'^[C,D,F,G,I,J,L-P,R,T-Z]{1}[0-9]{7}[A-Z0-9]')
postgres(# );
CREATE TABLE
postgres=#
```

A continuación, debemos hacer que se pueda fragmentar la tabla. Para ello, le indicamos con master_create_distributed_table la tabla, la primary key y el tipo, que pueden ser dos, hash o append.

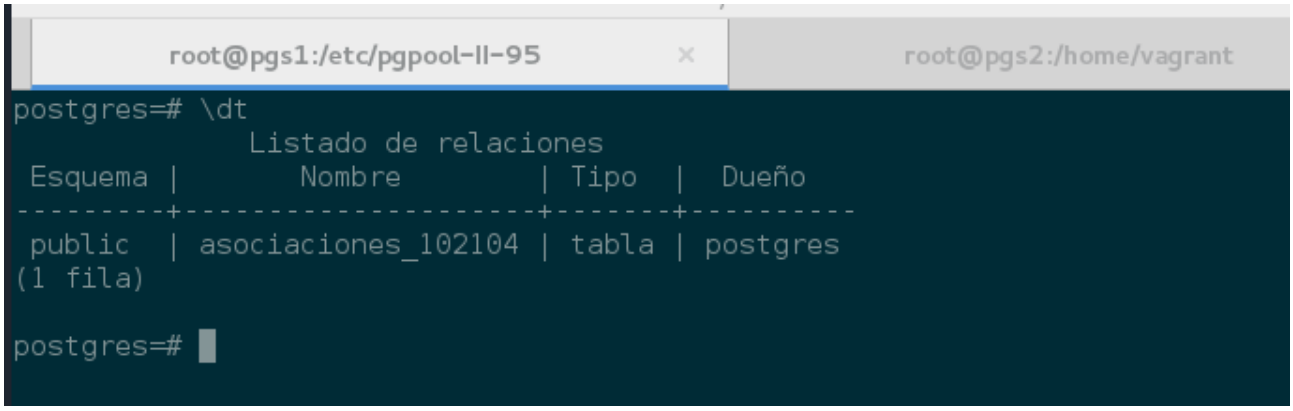
```
postgres=# SELECT master_create_distributed_table('Asociaciones', 'cif',
'hash');
 master_create_distributed_table
---------------------------------

(1 row)
```

Después de convertir la tabla en un tabla distribuida tenemos que fragmentarla entre los nodos. Para ello usamos el comando master_create_worker_shards en el que le tenemos que indicar el nombre de la tabla que queremos fragmentar el número de fragmentaciones y si queremos réplica de esta fragmentación(Con un 1 le indicamos que no queremos réplica ya que para eso se encarga pgpool).

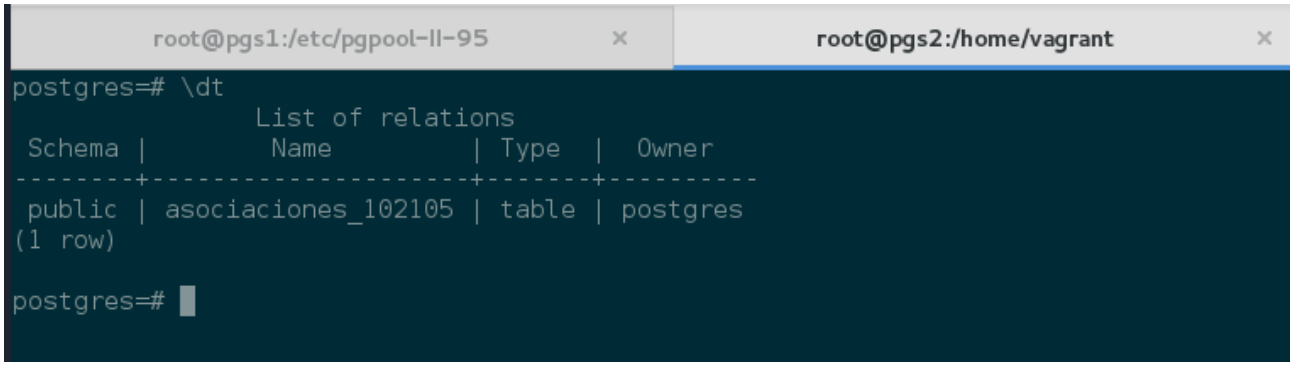
```
postgres=# SELECT master_create_worker_shards('Asociaciones', 2, 1);
 master_create_worker_shards
-----------------------------

(1 row)
```

Pasamos a comprobar PSG1 y PSG2 para ver si se han creado las tablas correctamente.

PGS1

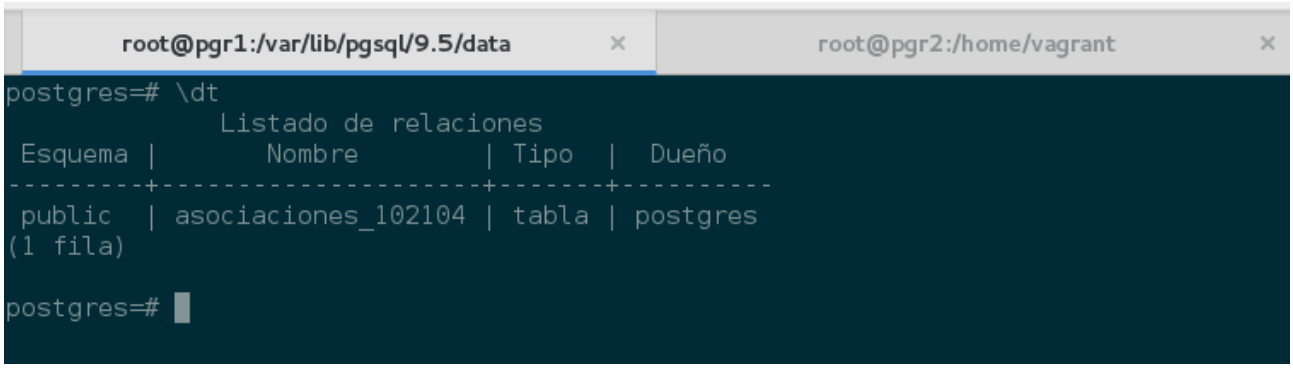


PGS2

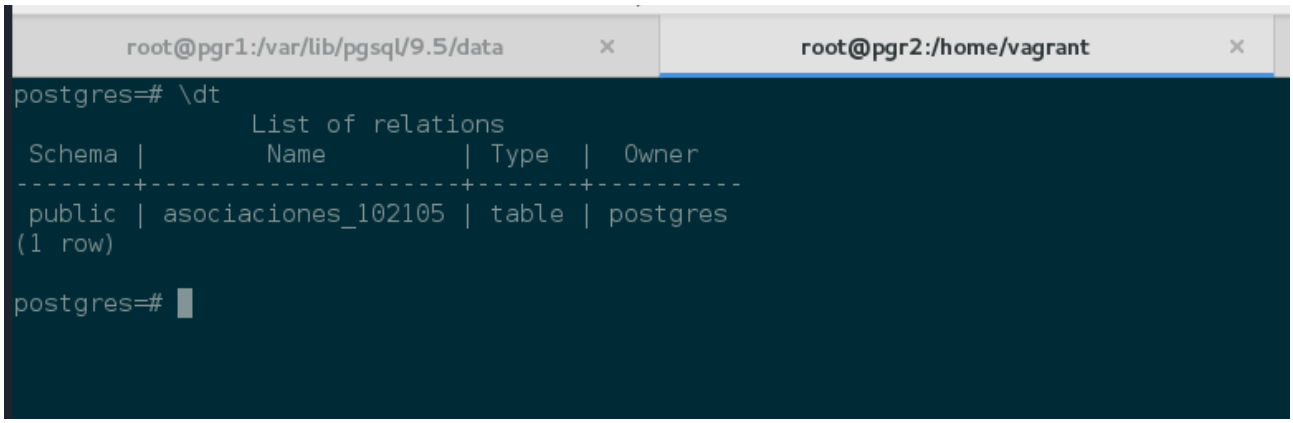


Como podemos ver la tabla Asociaciones se ha fragmentado correctamente entre los dos nodos. Ahora pasamos a comprobar si en PSR1 y en PSR2 se ha hecho la réplica correctamente.

PGR1



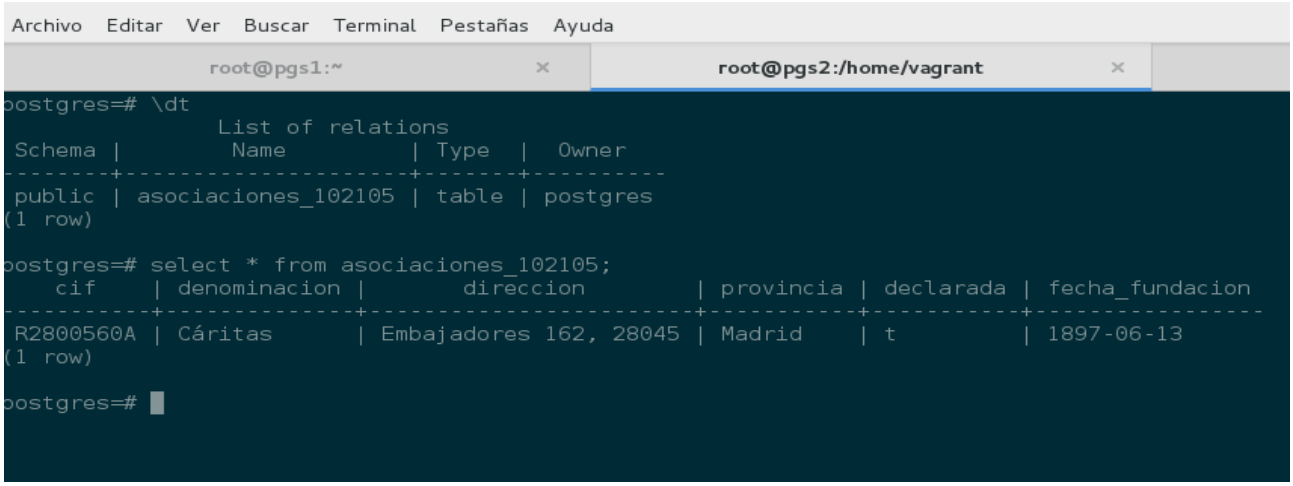
PGR2



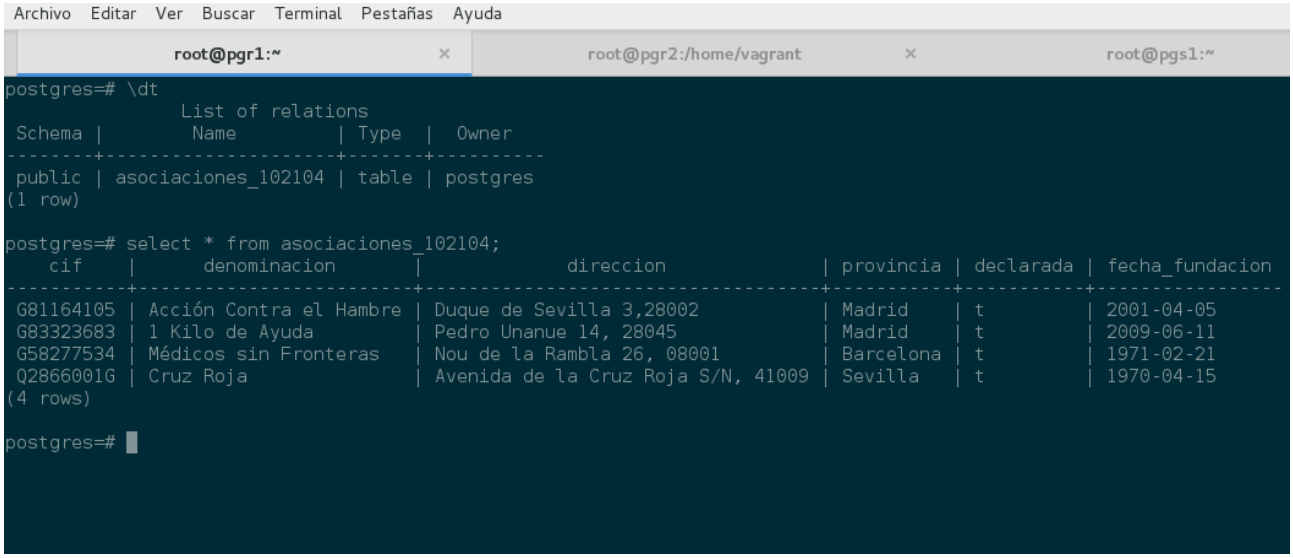
Después de comprobar que las tablas se han replicado correctamente pasamos a insertar los datos. Cuando estén todos insertados podemos ver desde cada nodo cuantos insert se han añadido en la tabla asociaciones de PGS1 y PGS2 a su vez también vamos a comprobar si se ha hecho la réplica correctamente.

PGS1

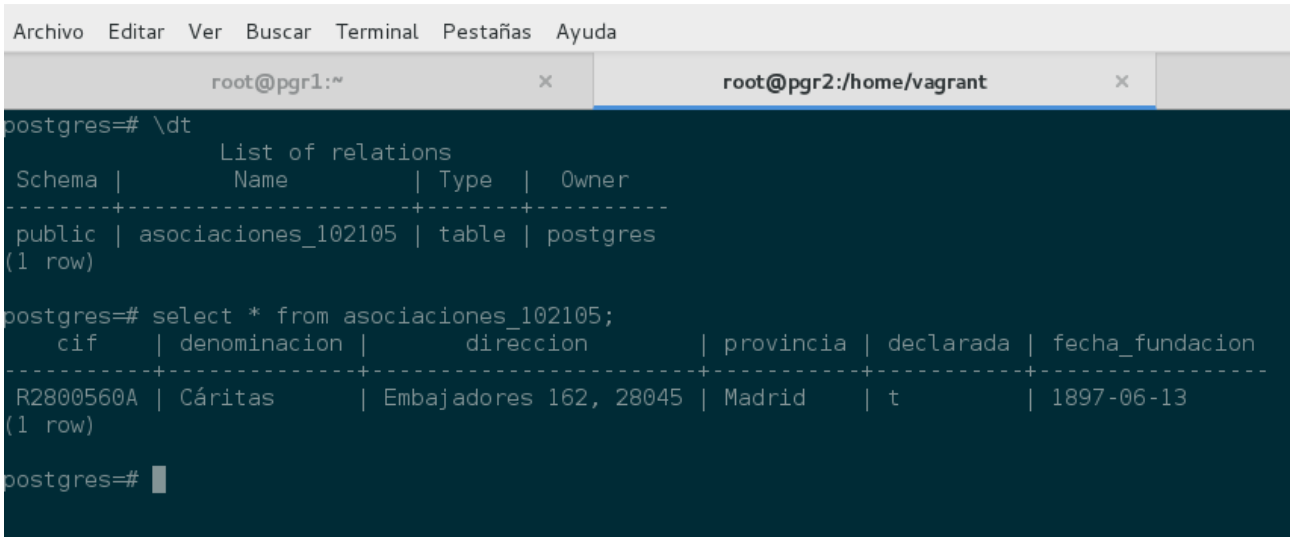
PGS2



PGR1



PGR2



Desde Pgmaster podemos ver en que host está un dato concreto de la siguiente forma.