**Titulación: Grado en Ingeniería Informática y Sistemas de Información**

**Curso: 2019-2020. Convocatoria Ordinaria de Junio**

**Asignatura: Bases de Datos Avanzadas – Laboratorio**

**Practica 4: Replicación e Implementación de una Base de Datos Distribuida.**

**ALUMNO 1:**

**Nombre y Apellidos: ISABEL MARTÍNEZ GÓMEZ**

**DNI: 06027983M**

**ALUMNO 2:**

**Nombre y Apellidos: JAVIER GARCÍA JIMÉNEZ**

**DNI: 09099503J**

**Fecha: 08/06/2020**

**Profesor Responsable: JOSÉ CARLOS HOLGADO**

Mediante la entrega de este fichero los alumnos aseguran que cumplen con la normativa de autoría de trabajos de la Universidad de Alcalá, y declaran éste como un trabajo original y propio.

En caso de ser detectada copia, se puntuará **TODA** la asignatura como Suspenso – Cero.

**Plazos**

Tarea online: Semana 27 de Abril y 4 de Mayo.

Entrega de práctica: Día 08 de Junio de 2020. Aula Virtual

Documento a entregar: Este mismo fichero con los pasos de la implementación de la replicación y la base de datos distribuida, las pruebas realizadas de su funcionamiento; y los ficheros de configuración del maestro y del esclavo utilizados en replicación; y de la configuración de los servidores de la base de datos distribuida. Obligatorio. Se debe de entregar en un ZIP comprimido: **DNI'sdelosAlumnos\_PECL4.zip**

**AMBOS ALUMNOS DEBEN ENTREGAR EL FICHERO EN LA PLATAFORMA.**

**Introducción**

El contenido de esta práctica versa sobre la Replicación de Bases de Datos con PostgreSQL e introducción a las bases de datos distribuidas. Concretamente se va a utilizar los servicios de replicación de bases de datos que tiene PostgreSQL. Para ello se utilizará PostgreSQL 12.x con soporte para replicación. Se prohíbe el uso de cualquier otro programa externo a PostgreSQL para realizar la replicación, como puede ser Slony.

También se va a diseñar e implementar una pequeña base de datos distribuida. Una base de datos distribuida es una base de datos lógica compuesta por varios nodos (equipos) situados en un lugar determinado, cuyos datos almacenados son diferentes; pero que todos ellos forman una base de datos lógica. Generalmente, los datos se reparten entre los nodos dependiendo de donde se utilizan más frecuentemente.

El escenario que se pretende realizar se muestra en el siguiente esquema:







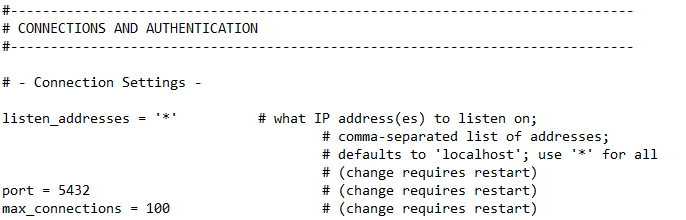
Se van a necesitar 4 máquinas: 2 maestros y 2 esclavos. Cada maestro puede ser un ordenador de cada miembro del grupo con una base de datos de unas tiendas en concreto (TIENDA1 y TIENDA2). Dentro de cada maestro se puede instalar una máquina virtual, que se corresponderá con el esclavo que se encarga de replicar la base de datos que tiene cada maestro, es decir, hace una copia o backup continuo de la base de datos TIENDA1 o de la base de datos TIENDA2.

Se debe de entregar una memoria descriptiva detallada que posea como mínimo los siguientes puntos:

1. Configuración de cada uno de los nodos maestros de la base de datos de TIENDA1 y TIENDA2 para que se puedan recibir y realizar consultas sobre la base de datos que no tienen implementadas localmente.

Para configurar cada uno de los nodos maestros de la base de datos de Tienda1 y Tienda2 para que puedan recibir y realizar consultas sobre la base de datos que no tienen implementadas localmente, debemos modificar parámetros en el archivo postgresql.conf y pg\_hba.conf.

Para permitir que los nodos maestros puedan recibir consultas de otros equipos, tenemos que asegurarnos de que los parámetros *listen\_addresses* y *max\_connections* del archivo postgresql.conf estén bien configurados:



Listen\_addresses especifica las direcciones TCP/IP en las que el servidor (maestro) estará escuchando las conexiones de las aplicaciones cliente (esclavo). Especificamos el comando listen\_addresses igual a \*, de forma que el servidor escuche a todas las interfaces IP disponibles.

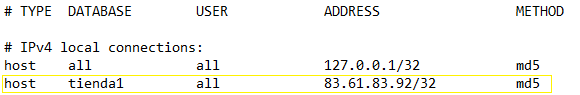
Port es el puerto TCP en el que está escuchando el servidor. El puerto por defecto de postgres es el 5432.

Por último, max\_connections determina el número máximo de conexiones simultáneas al servidor de la base de datos. En nuestro caso, hemos dejado el valor por defecto que es 100.

Una vez que hemos comprobado que todo esté correctamente configurado en el archivo postgresql.conf pasaremos a modificar el archivo pg\_hba.conf almacenado también en el directorio de datos del clúster de la base de datos y en el que especificaremos las direcciones IPs que tendrán permitido el acceso a la base de datos. Esto lo haremos en los dos maestros.

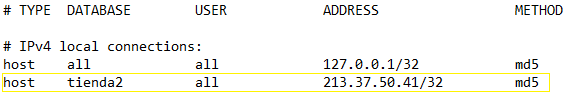
**Maestro 1 con tienda1:**

En el Maestro 1 se ha añadido una fila en la que permitimos que el Maestro 2 con IP pública 83.61.83.92 pueda conectarse a la base de datos tienda1 que reside en el host del Maestro 1 con autenticación md5.



**Maestro 2 con tienda2:**

En el Maestro 2 se ha añadido una fila en la que permitimos que el Maestro 1 con IP pública 213.37.50.41 pueda conectarse a la base de datos tienda2 que reside en el host del Maestro 2 con autenticación md5.



Una vez que hemos modificado y guardado los cambios en ambos ordenadores seguiremos cuatro pasos para prepararnos para el acceso remoto utilizando postgres\_fdw:

Primero, hemos instalado la extensión postgres\_fdw que permite acceder a los datos almacenados en servidores externos PostgreSQL.

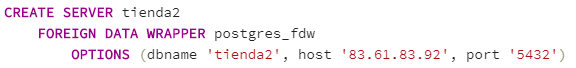
Para crear esta extensión utilizaremos en ambos maestros el siguiente comando:



Después, creamos un objeto de servidor externo en cada uno de los maestros para representar la base de datos remota a la que queremos conectarnos desde nuestro maestro y especificamos la información de conexión.

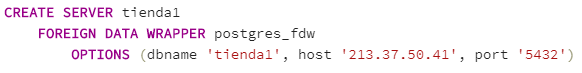
**Maestro 1 con tienda1:**

En el caso del Maestro 1, se crea un objeto de servidor externo tienda2 que es la base de datos remota a la que el Maestro 1 quiere conectarse, para ello se especifica la información de conexión: nombre de la base de datos, la IP pública y el puerto del Maestro 2.



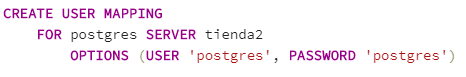
**Maestro 2 con tienda2:**

Por otra parte, el Maestro 2 crea un objeto de servidor externo tienda1 que es la base de datos remota a la que el Maestro 2 quiere conectarse, para ello se especifica la información de conexión: nombre de la base de datos, la IP pública y el puerto del Maestro 1.

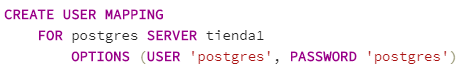


El tercer paso es la creación de un user mapping para cada usuario de la base de datos al que queremos permitir el acceso a cada servidor externo. Para ello tendremos que especificar el nombre del usuario remoto y la contraseña en nuestro caso postgres en ambas.

**Maestro 1 con tienda1:**



**Maestro 2 con tienda2:**

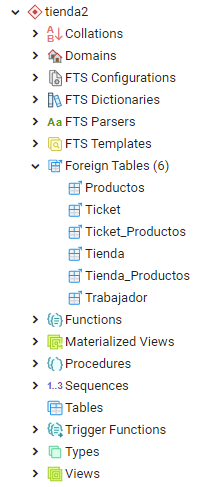


Por último, se ha creado un schema en cada maestro, se ha cambiado el nombre del schema public por tienda 1 y tienda 2 respectivamente y se han importado las tablas del servidor externo en el schema de cada maestro.

**Maestro 1 con tienda1:**





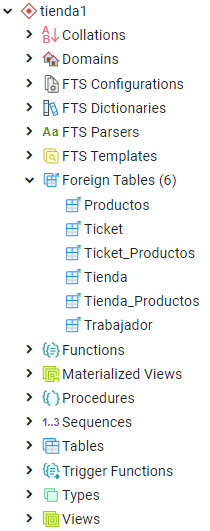
Como se puede observar en la siguiente captura, podemos ver como en el schema tienda2 del Maestro 1 se han importado todas las tablas que tiene el Maestro 2 en su base de datos.

**Maestro 2 con tienda 2:**





En el caso del Maestro 2, podemos ver como en el schema tienda1 que hemos creado se han importado todas las tablas que tiene el Maestro 1 en su base de datos tienda1 y se han añadido a la sección “Foreign Tables”.



Para conseguir lo anterior hemos tenido que hacer algunas configuraciones en el router en ambos ordenadores:

Por una parte, hemos tenido que añadir una entrada en la configuración del router, específicamente en la sección de “Redirección de Puertos”, para que se permita la conexión del maestro remoto en el puerto asignado a postgres que es el 5432.

**Maestro 1:**

En el campo LAN IP se ha especificado la IP privada del Maestro 1, que la podemos encontrar en el cmd de Windows con el comando ipconfig y en el campo puerto LAN y puerto público se ha especificado el puerto donde se encuentra postgres.

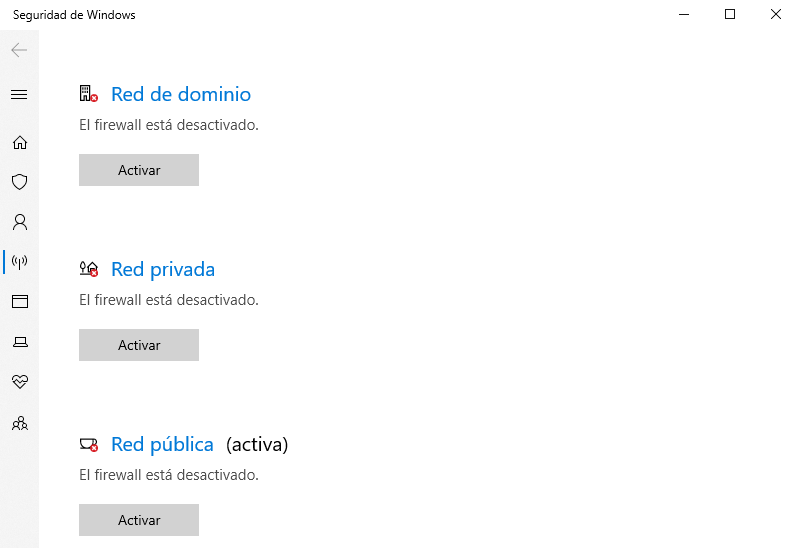


**Maestro 2:**

En el campo Dirección IP se ha especificado la IP privada del Maestro 2, que la podemos encontrar en el cmd de Windows con el comando ipconfig y en el campo Interno Inicio/Fin y Externo Inicio/Fin se ha especificado el puerto donde se encuentra postgres.



Además, para la realización de la práctica se ha desactivado el firewall en ambos ordenadores para poder realizar la práctica sin ningún tipo de problema.



1. Configuración completa de los equipos para estar en modo de replicación. Configuración del nodo maestro. Tipos de nodos maestros, diferencias en el modo de funcionamiento y tipo elegido. Tipos de nodos esclavos, diferencias en el modo de funcionamiento y tipo elegido, etc.

Para realizar la configuración completa de los equipos para estar en modo replicación, debemos realizar ciertas configuraciones tanto en los maestros como en los esclavos.

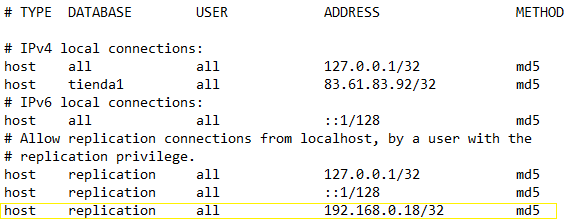
Poner dos ordenadores en modo replicación no es ni más ni menos que conectar dos ordenadores y configurarlos para que todos los cambios realizados en la base de datos del nodo maestro se realicen al mismo tiempo en el nodo esclavo. Esto nos sirve para mantener la seguridad y la integridad de los datos de nuestra base de datos.

Comenzaremos a configurar los nodos maestros:

Primero añadimos en el archivo pg\_hba.conf la IP privada del nodo esclavo y añadimos un campo indicando que está en modo replicación.

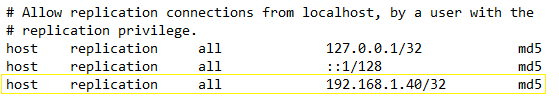
**Maestro 1:**

Como se puede ver resaltado en el archivo pg\_hba del Maestro 1, se ha añadido una fila con la dirección IP privada del Esclavo 1, 192.168.0.18, y cuya base de datos está en modo replicación.



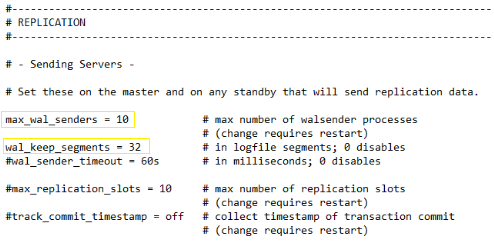
**Maestro 2:**

Como se puede ver resaltado en el archivo pg\_hba del Maestro 2, se ha añadido una fila con la dirección IP privada del Esclavo 2, 192.168.1.40, y cuya base de datos está en modo replicación.

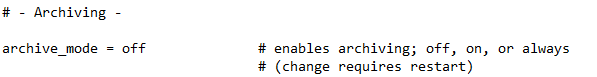


A continuación, pasamos a modificar el archivo postgresql.conf de los maestros. En este archivo vamos a modificar diferentes parámetros para permitir la replicación entre el maestro y el esclavo.

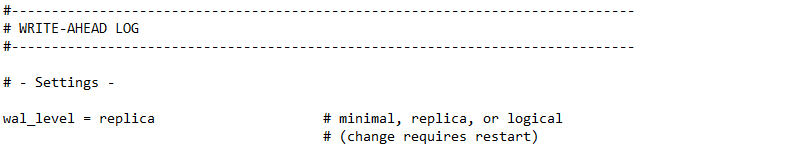
Hay que asegurarnos de que *max\_wal\_senders* tenga establecido un valor suficientemente grande en el archivo de configuración del maestro por lo que establecemos *max\_wal\_senders=10* y también asignaremos el tamaño del buffer de los fragmentos de WAL a 32 en el parámetro *wal\_keep\_segments.*



Por otra parte, ponemos el parámetro *archive\_mode* a “off” para actuar en modo Streaming Replication.



Por último, pondremos *wal\_level=replica* para activar la replicación del WAL. Además esto es necesario para posteriormente activar el hot\_standby en el esclavo.



Una vez configurado el archivo postgresql.conf, pasaremos a realizar un backup del maestro. Esto se hace para pasar el backup al esclavo y conectar ambos equipos con la base de datos en el mismo estado.

Para realizar el backup, ejecutaremos el comando pg\_basebackup con una serie de parámetros que indican:

* -h localhost → indica que nuestro backup se está realizando en este mismo equipo
* -p 5432 → indica el puerto del servidor
* -U postgres → indica que el usuario que está realizando dicho backup es postgres
* -D “C:\BackupMaestro” → indica la ruta de la carpeta donde se van a guardar los datos de dicho backup.
* -R: este parámetro crea el archivo standby.signal y añade ajustes de conexión al archivo postgresql.auto.conf para facilitar la configuración de un servidor en espera.

Ahora realizamos el backup en los dos maestros:



Con esto, ya habríamos preparado la configuración de los nodos maestros y pasamos a configurar los nodos esclavos.

Lo primero que tenemos que hacer es detener el servidor postgresql de los nodos esclavos:



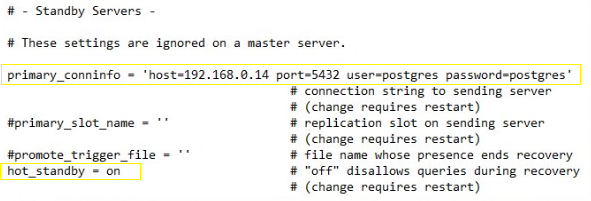
Una vez que hemos detenido el servidor en ambos esclavos, borramos todo lo que hay en la carpeta data de ambos esclavos y copiamos el backup que habíamos realizado en los nodos maestros.

A continuación, modificaremos el archivo *postgresql.conf* de ambos esclavos y modificaremos dos parámetros *primary\_conninfo* y *hot\_standby*.

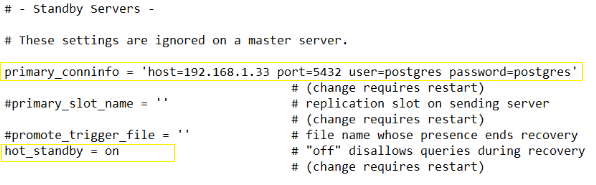
En el parámetro *primary\_conninfo* especificaremos el host con la IP privada del maestro, el puerto del maestro, el usuario y la contraseña.

Por otra parte, el parámetro *hot\_standby* lo ponemos a “on” para que el usuario pueda conectarse y además hacer consultas durante el *recovery.*

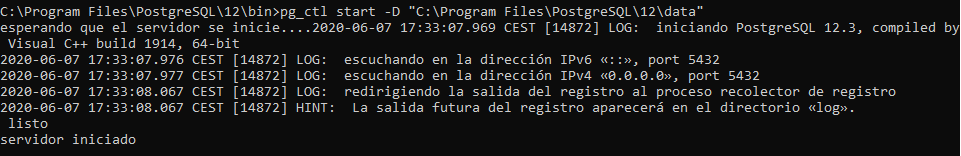
**Esclavo 1:**



**Esclavo 2:**

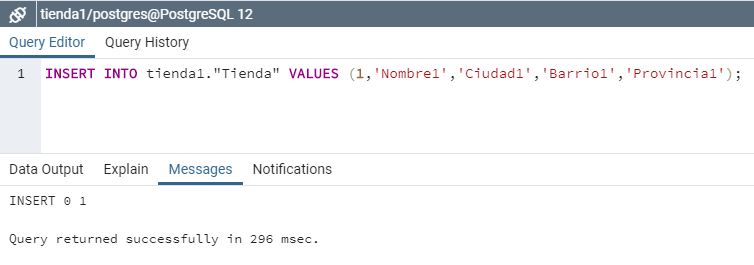


Ahora que ya hemos configurado el archivo postgresql.conf en los dos esclavos, iniciaremos el servidor postgres en ambos ordenadores con pg\_ctl start:



Ya habríamos terminado la configuración y por tanto ya debería de funcionar la replicación entre los maestros y sus esclavos. Vamos a probar que funciona haciendo una inserción de una tupla en la tabla Tienda del Maestro y posteriormente haciendo una lectura para ver si el Esclavo es capaz de ver lo que el Maestro ha escrito:

**Maestro 1 hace inserción de una tupla en la tabla Tienda:**

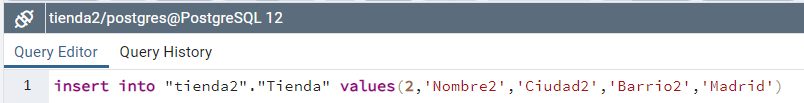


**Esclavo 1 leyendo sobre la tabla Tienda:**

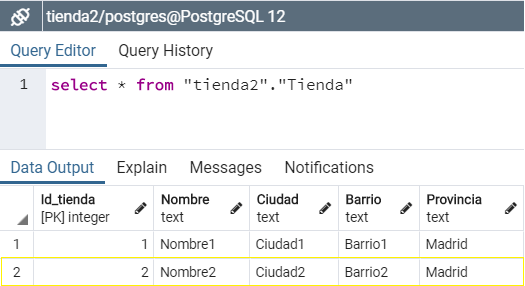
Como se puede observar el esclavo puede ver los cambios que hay en la base de datos tienda1.



**Maestro 2 insertando una tupla en la tabla Tienda**



**Esclavo 2 leyendo la tupla insertada por el Maestro 2 en la tabla Tienda**



Vamos a pasar a ver cuáles son los tipos de nodos maestros/esclavos que posee postgres. Eso se representa mediante diversas configuraciones de sistemas de replicación, que explicaremos a continuación:

* **Shared Disk Failover**: esta configuración evita los gastos de sincronización a través de una única copia de la base de datos. Utiliza un array de disco compartido por múltiples servidores. En caso de fallo del maestro, el esclavo es capaz de montar y comenzar la base de datos como si se estuviera recuperando de una caída.
* **File System Replication**: Se trata de un sistema de replicación donde todos los cambios realizados en el maestro son replicados en el esclavo, el cual es un elemento que se encuentra leyendo en otro ordenador. Para preservar la consistencia, se obliga a que los datos en el esclavo sean copiados en el mismo orden en el que fueron copiados en el maestro.
* **Write-Ahead Log Shipping**: en este sistema los esclavos se mantienen actualizados constantemente leyendo un registro de escritura anticipada (WAL). Si el maestro falla gracias a estas lecturas el esclavo posee prácticamente toda la información del nodo maestro y puede reiniciar rápidamente la base de datos.

Se puede implementar mediante File-based log shipping o mediante Streaming replication.

* + **File-based log shipping**: en este sistema existe una carpeta compartida en la que el maestro irá escribiendo los cambios que se realizan en la base de datos. Dicha carpeta es accesible por los esclavos que podrán copiar lo que se ha hecho en la base de datos del maestro a través de dichos cambios. El mayor inconveniente de este sistema es que hay muchos retrasos al tener que escribir los cambios en la carpeta compartida.
  + **Streaming replication**: el esclavo se conecta al maestro y éste le transmite los archivos WAL directamente. Al no existir una carpeta compartida como en el caso anterior, este sistema es mucho más rápido al no tener que realizar operaciones de escritura y lectura en disco. En este caso existirá un buffer que almacena los fragmentos de WAL.
* **Logical Replication**: en este sistema se permite que un servidor de base de datos envíe modificaciones de datos a través de un canal a otro servidor. Esto acaba construyendo un conjunto de modificaciones de datos extraídos del WAL. Permite también que los cambios de las tablas individuales sean replicados. Este sistema no requiere una designación especial de un maestro y un esclavo, sino que permite que los datos fluyan en múltiples direcciones.
* **Trigger-Based Master-Standby Replication**: este sistema manda todas las consultas de modificación de datos al maestro. Una vez ocurre esto, el maestro envía de forma asíncrona dichos datos al esclavo. El esclavo en este sistema se encuentra en estado read-only, y sólo puede resolver consultas de lectura.
* **SQL-Based Replication Middleware**: con este sistema un programa intercepta todas las consultas SQL y las manda a uno o todos los servidores. Cada servidor opera de manera individual. Si se produce una consulta de lectura/escritura ésta debe ser enviada a todos los servidores para que puedan realizar los cambios necesarios, en cambio, si se trata de una consulta de sólo lectura, puede ser leída simplemente por un servidor, ya que no realiza cambios.
* **Asynchronous Multimaster Replication**: este sistema es apropiado para servidores que no se encuentran conectados regularmente o tienen unos enlaces de comunicación inconsistentes o lentos. Aquí, cada servidor trabaja de manera independiente y periódicamente se comunica con los otros servidores para identificar posibles conflictos entre transacciones.
* **Synchronous Multimaster Replication**: en este caso cada servidor puede aceptar consultas de escritura, y los datos modificados se transmiten desde el servidor original al resto de servidores antes de que cada transacción sea terminada. Esto conlleva problemas de lentitud y bloqueo si hay una alta actividad de escritura.
* **Data Partitioning**: este sistema particiona los datos en conjuntos. Cada conjunto puede ser modificado por un único servidor.
* **Multiple-Server Parallel Query Execution**: este sistema permite que múltiples servidores trabajen de forma concurrente en la misma consulta.

En nuestro caso hemos elegido el sistema Write-Ahead Log Shipping y en especial el modo Streaming replication para configurar nuestros maestros y esclavos.

1. Operaciones que se pueden realizar en cada tipo de equipo de red. Provocar situaciones de caída de los nodos y observar mensajes, acciones correctoras a realizar para volver el sistema a un estado consistente.

Como hemos visto anteriormente, hemos configurado dos nodos maestros y dos nodos esclavos. Además de diferenciarse en su modo de configuración y en su nombre, ambos tipos de equipos dentro de nuestra red poseen diferentes operaciones que pueden realizar.

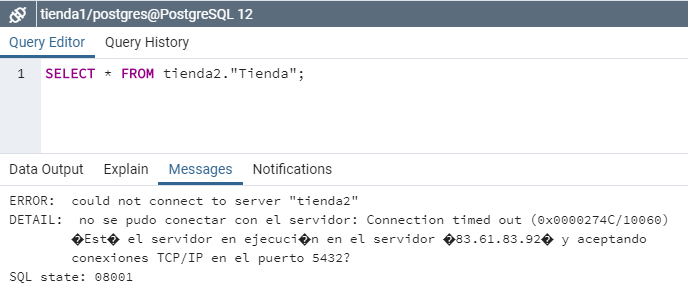
En cuanto a los nodos maestros, estos pueden realizar todo tipo de operaciones dentro de la base de datos, tanto seleccionar, insertar, borrar o actualizar datos.

Sin embargo, si nos vamos a un esclavo nos encontramos con una situación diferente. En los nodos esclavos solo podemos realizar operaciones de selección. Esto es porque el nodo esclavo se encuentra en modo lectura de la base de datos, por tanto, sólo le está permitido leer datos de esta, y si intentamos realizar cualquier tipo de operación que viole esa restricción de sólo lectura, nos encontraremos con un error.

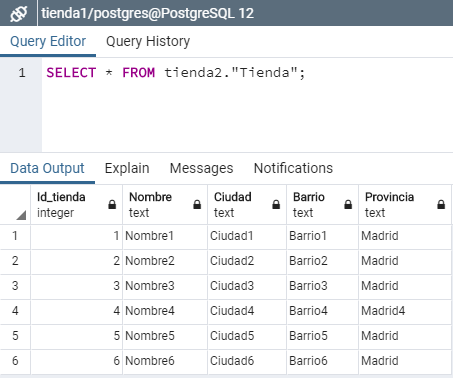
Ahora vamos a pasar a ver qué ocurriría en caso de caída de los nodos.

* Caída de nodo esclavo:
  + Si se produjese la caída de un nodo esclavo, no supondría ningún problema para la integridad de la base de datos, ya que simplemente se dedica a leer datos producidos por su nodo maestro, por lo que la base de datos tendría un comportamiento absolutamente normal. El único problema que podría suponer la caída de un nodo esclavo es si a continuación se produjese una caída del nodo maestro, pero ¿por qué? Bien, esto nos supondría problemas, ya que se podría producir una inconsistencia de datos cuya consecuencia es la pérdida de los mismos de forma irrecuperable.
* Caída de nodo maestro:
  + En cuanto a la caída de un nodo maestro sí que conlleva más problemas en nuestro sistema. Lo primero sería que el nodo esclavo dejaría de recibir datos, por lo que se quedaría intentando conectar al nodo maestro sin obtener respuesta alguna, aunque en cuanto a su funcionamiento no se vería afectado, ya que solo se ha caído su nodo maestro. Como añadido, si intentamos introducir datos desde el otro nodo maestro, al nodo maestro caído, nos encontraríamos con un error de conexión producido por la caída del mismo.

La siguiente imagen representa el error obtenido si intentamos hacer una selección desde el maestro que no se ha caído de las tablas pertenecientes al maestro que está caído. En este caso, el Maestro 2 que tiene la base de datos tienda2 está caído y por tanto si el Maestro 1 intenta leer información sobre la tabla Tienda disponible en la tienda2, nos encontramos con el siguiente error:

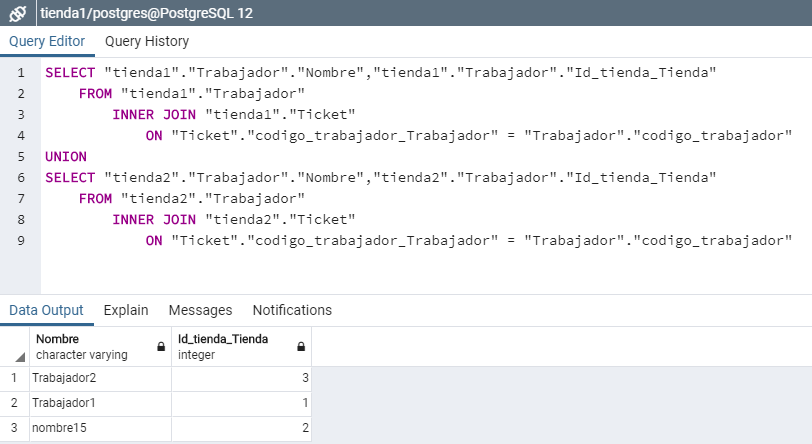


Si volvemos a iniciar el maestro caído y volvemos a intentar leer la tabla Tienda de la base de datos del Maestro 2 desde el Maestro 1, este es el resultado a la misma consulta anterior:

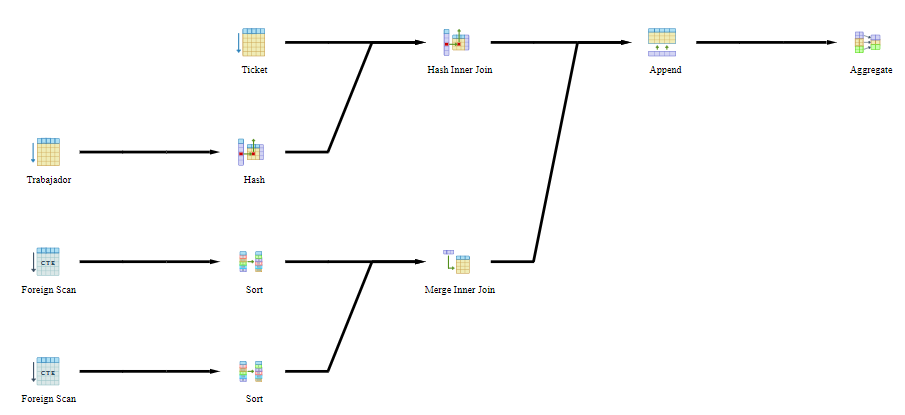


1. Insertar datos en cada una de las bases de datos del MAESTRO1 y del MAESTRO2. Realizar una consulta sobre el MAESTRO1 que permita obtener el nombre de todos los trabajadores junto con su tienda en la que trabajan que hayan realizado por lo menos una venta de algún producto en toda la base de datos distribuida (MAESTRO1 + MAESTRO2). Explicar cómo se resuelve la consulta y su plan de ejecución.

Para obtener el nombre de todos los trabajadores junto con la tienda en la que trabajan y que hayan realizado al menos una venta de algún producto en toda la base de datos se ha realizado la siguiente consulta en la que hacemos un UNION para seleccionar todas las tuplas que salen de la base de datos del Maestro 1 (tienda1) y también las tuplas que cumplen la condición de la base de datos del Maestro 2 (tienda2):



El plan de ejecución realizado en el Maestro 1 se muestra en el siguiente esquema donde podemos ver que las tablas del Maestro 2 son cargadas como tablas foráneas (Foreign scan), son ordenadas y posteriormente se añaden a las que cumplían la condición de la consulta en las tablas locales de la base de datos tienda1 del Maestro 1.



1. Si el nodo MAESTRO1 se quedase inservible, ¿Qué acciones habría que realizar para poder usar completamente la base de datos en su modo de funcionamiento normal? ¿Cuál sería la nueva configuración de los nodos que quedan?

En caso de que el nodo maestro se cayese de forma definitiva y no pudiese volver a ser utilizado, tendríamos que hacer que el nodo esclavo pasase a ser el nodo maestro. Para ello deberíamos modificar el nodo esclavo ya que se encuentra en estado de solo lectura de la base de datos y eso debe cambiar. Para ello, se ejecutaría el comando pg\_ctl promote.

Además, habría que realizar una nueva configuración de este nuevo nodo maestro y el nodo Maestro 2 a través de los archivos pg\_hba.conf correspondientes, ya que ahora las condiciones de conexión han cambiado. El nodo Maestro 1 ha cambiado y, por tanto, hay que rehacer todos los pasos necesarios para conectar el nuevo nodo Maestro 1 con el nodo Maestro 2. Además habría que modificar el servidor remoto del Maestro 2 y Esclavo 2 a través del comando alter server sustituyendo la IP del servidor antiguo por la nueva IP para que puedan conectarse a la base de datos del nuevo Maestro 1.

1. Según el método propuesto por PostgreSQL, ¿podría haber inconsistencias en los datos entre la base de datos del nodo maestro y la base de datos del nodo esclavo? ¿Por qué?

Es posible que haya inconsistencias con el método propuesto por PostgreSQL.

Una posible situación de inconsistencia podría ser cuando se produce un error de conexión en el maestro y éste sigue insertando, actualizando o eliminando datos en la base de datos ya que estas modificaciones no estarían visibles para el otro maestro y su esclavo como hemos visto en la cuestión 3 donde en el caso de que se cayera el servidor postgres ningún equipo puede acceder a él, sin embargo, cuando la conexión se recupera, las modificaciones son replicadas en todos los equipos conectados al maestro resolviéndose cualquier posible inconsistencia.

Por otra parte, al ser una base de datos distribuida puede haber posibles retrasos entre la modificación de la base de datos de un maestro y su replicación en su esclavo, por ello puede producirse una inconsistencia durante un mínimo periodo de tiempo que es casi inapreciable. Esto es debido a que estamos utilizando el modo Streaming replication y los cambios que hace un maestro se replican de forma casi inmediata en los demás equipos. Sin embargo, si hubiésemos utilizado otro modo, quizás los retrasos podrían haber sido mucho mayores y producirse inconsistencias durante un periodo de tiempo mucho mayor que con el modo Streaming replication.

1. Conclusiones.

En esta práctica hemos aprendido sobre las bases de datos distribuidas en postgres, tanto como funcionan, como diferentes modos junto con sus ventajas e inconvenientes. Al haber realizado la implementación de una base de datos distribuida se ha podido analizar de manera precisa cómo funcionan estas bases de datos y cuáles son los problemas que presenta su implementación.

En primer lugar, hemos aprendido sobre cómo implementar una base de datos distribuida en postgres, a través de la modificación de los archivos pg\_hba.conf y postgresql.conf, así como a configuración de aspectos de red para permitir que dos dispositivos remotos se puedan conectar entre sí.

Al implementarla se puede observar las ventajas que una base de datos de este tipo puede tener, como mayor seguridad ante caídas, ya que si se cae un nodo aún existe otro nodo operativo, por lo que solo se perdería la posible lectura de un conjunto de datos. Además de esta manera se permite la distribución de una base de datos en dos ordenadores, lo que hace que su carga se vea dividida en dos, algo que puede resultar más eficiente.

Además, se han implementado unos nodos esclavos para aumentar la integridad y seguridad de los datos ante caídas. A través de la modificación de los dos archivos mencionados anteriormente, hemos podido implementar unos nodos esclavos que se encargan de leer todo lo que se modifica en los maestros.

Gracias a estos nodos esclavos, la base de datos está protegida en cuanto a caídas, ya que los datos residen en dos sitios diferentes, y si se cae el nodo maestro, el nodo esclavo seguirá operativo en la mayoría de los casos.

Por tanto, en esta práctica se han podido comprobar todas las ventajas que puede tener una base de datos distribuida en postgres, así como el camino a seguir para realizar su implementación.

La memoria debe ser especialmente detallada y exhaustiva sobre los pasos que el alumno ha realizado y mostrar evidencias de que ha funcionado el sistema.

**Bibliografía**

* Capítulo: 20.1. The pg\_hba.conf File
* Capítulo 25: Backup and Restore.
* Capítulo 26: High Avaliability, Load Balancing, and Replication.
* Appendix F: Additional Supplied Modules. F.33. Postgres\_fdw