

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: Javier Martín Gómez _____

DNI: 47231977M _____

ALUMNO 2:

Nombre y Apellidos: Alberto González Martínez _____

DNI: 09072311F _____

-

-Fecha: 8-6-2020 _____

Profesor _____ **Responsable:** _____ **Oscar**
Gutiérrez _____

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 18 de Mayo de 2020 (**provisional**). 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

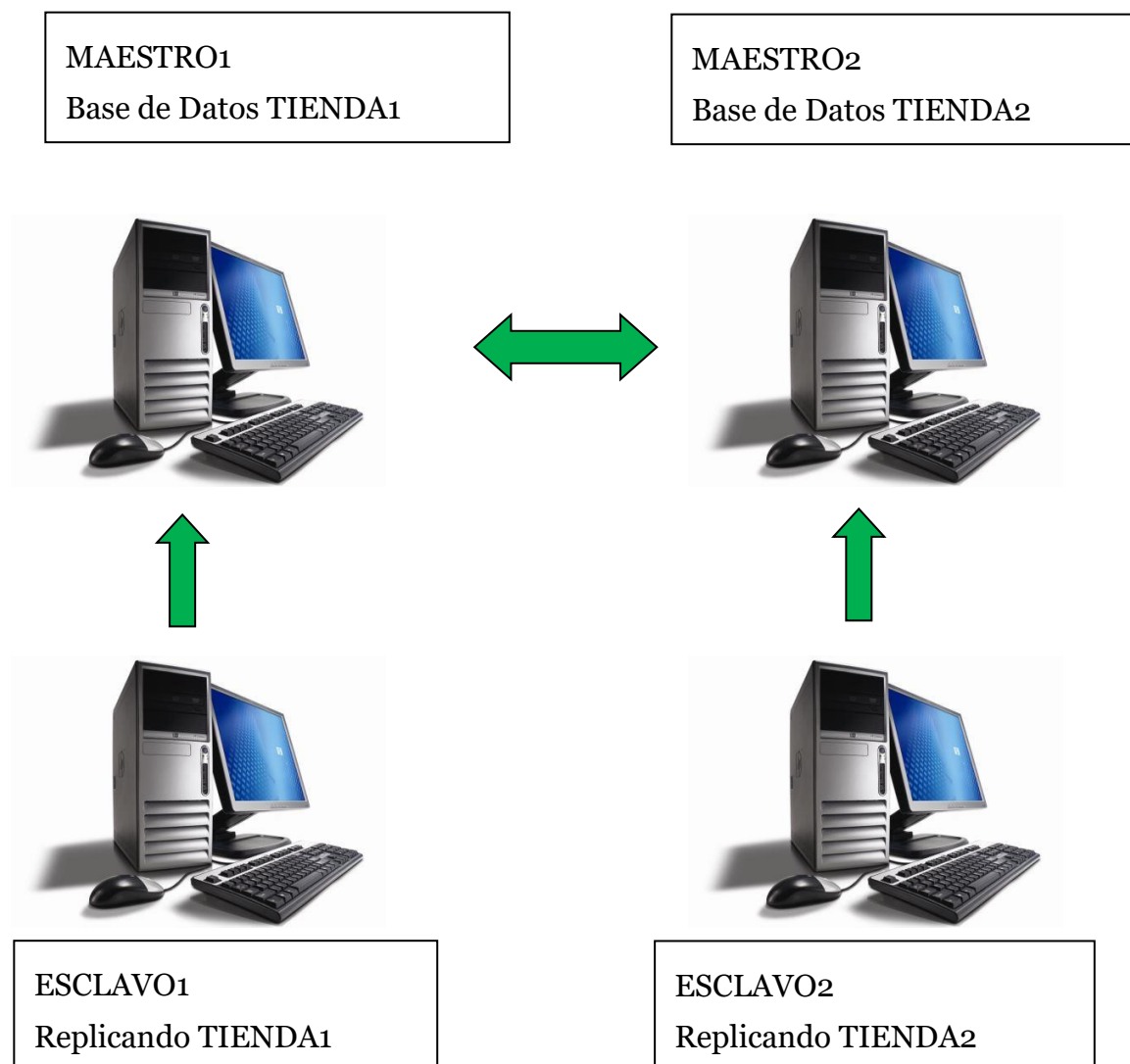
AMBOS ALUMNOS DEBEN ENTREGAR EL FICHERO EN LA PLATAFORMA.

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.

Conectamos nuestros routers abriendo los puertos y accediendo a la IP de cada uno.

Tabla actual de mapeo de puertos

	Nombre	Protocolo	Puerto/Rango Externo	Puerto/Rango Interno	Dirección IP	Activar
✖	Postgres	TCP+UDP	5432	5432	192.168.1.69	<input checked="" type="checkbox"/>

Editar

Nombre

Postgres

Protocolo

TCP

Puerto/Rango Externo

5432:5432

Puerto/Rango Interno

5432:5432

Dirección IP

192.168.1.69

Activar

ON

1/1

Configuramos el archivo pg_hba.conf para acceder a nuestras bases de datos, introduciendo la IP pública de nuestro compañero.

```
# IPv4 local connections:
host    all             all             127.0.0.1/32      md5
host    all             all             83.57.35.56/32    md5

# TYPE  DATABASE      USER      ADDRESS            METHOD

# IPv4 local connections:
host    all             all             127.0.0.1/32      md5
host    all             all             88.16.172.227/32   md5
# IPv6 local connections:
host    all             all             ::1/128           md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
host    replication     all             127.0.0.1/32      md5
host    replication     all             ::1/128           md5
```

Una vez establecidas las IPs, creamos un servidor con la IP de nuestro compañero. Creamos la extensión:

```
1 --Creamos la extension
2 CREATE EXTENSION postgres_fdw;
```

Creamos un servidor con la IP de nuestro compañero:

```
|
create server tienda_alberto foreign data wrapper postgres_fdw
options(host '83.57.35.56', dbname 'Tienda2', port '5432');
```

Importamos la base de datos de nuestro compañero:

```
create user mapping for postgres server tienda_alberto options(user 'postgres', password 'postgres');

create schema Tienda2;

import foreign schema public from server tienda_alberto into Tienda2;
```

```
11 CREATE EXTENSION postgres_fdw;
12
13 CREATE SERVER server_tienda1
14     FOREIGN DATA WRAPPER postgres_fdw
15     OPTIONS (host '88.16.172.227', port '5432', dbname 'Tienda1');
16
17 CREATE USER MAPPING FOR postgres
18     SERVER server_tienda1
19     OPTIONS (user 'postgres', password 'postgres');
20
21 CREATE SCHEMA tienda1;
22
23 IMPORT FOREIGN SCHEMA PUBLIC FROM SERVER server_tienda1 INTO tienda1;
```

Si hacemos una consulta en Tienda, nos aparecen los datos de nuestro compañero. Consulta de la base de datos de Alberto desde la máquina de Javier:

```

16 select * from Tienda2."Tienda";
17
18

```

Data Output Explain Messages Notifications

	Id_tienda integer	Nombre text	Ciudad text	Barrio text	Provincia text
1	1	Tienda1	Madrid	Vicalvaro	Madrid
2	2	Tienda2	Madrid	Alcala	Madrid
3	3	Tienda3	A Coruña	A Coruña	Galicia
4	4	Tienda4	Murcia	Plaza Cas...	Murcia
5	5	Tienda5	Huelva	Rojas	Andalucia
6	6	Tienda6	Logroño	Avenida	La Rioja
7	7	Tienda7	Torrejon	Calle Iugo	Madrid
8	8	Tienda8	Alcala de H...	Santos ni...	Madrid

Consulta de la base de datos de Javier desde la máquina de Alberto

```

24
25 SELECT * FROM tienda1."Tienda";
26

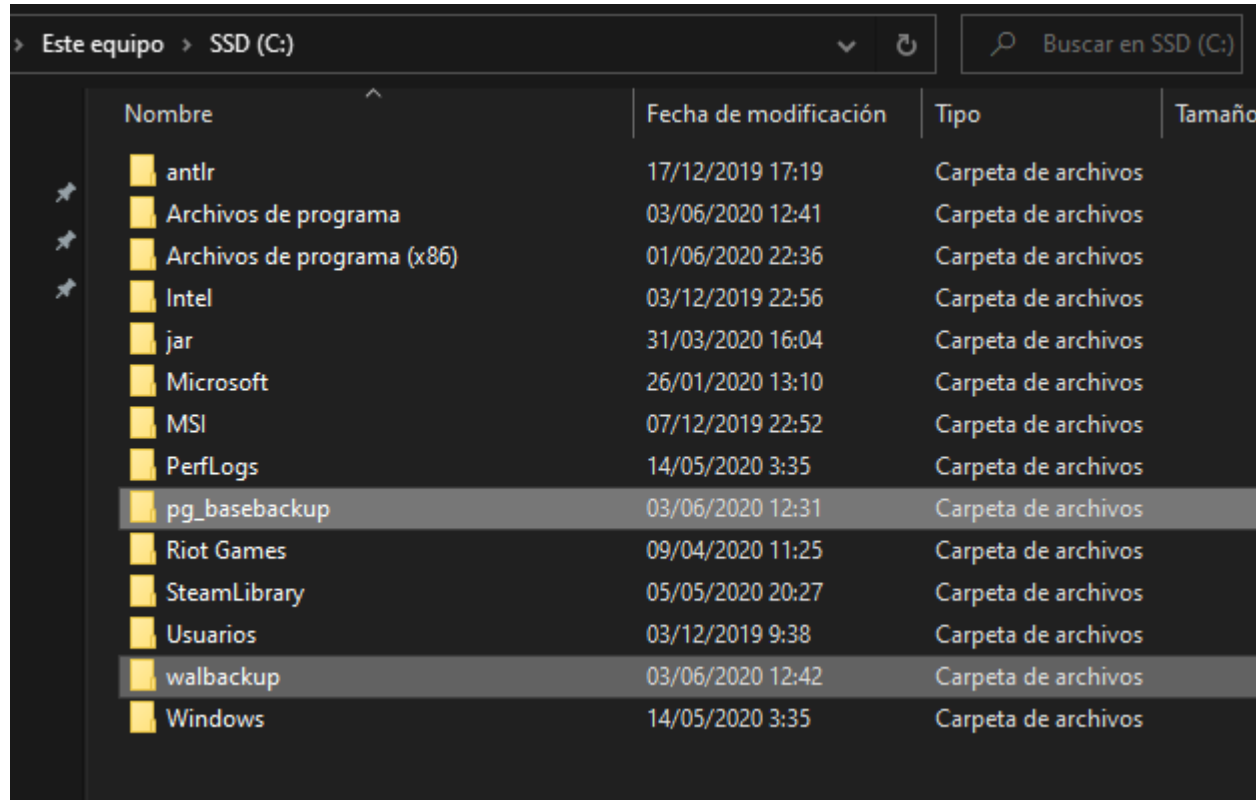
```

Data Output Explain Messages Notifications

	Id_tienda integer	Nombre text	Ciudad text	Barrio text	Provincia text
1	1	Tienda1	Madrid	Vicalvaro	Madrid
2	31145	Tienda Alcala	Cadiz	Puerto de...	Andalucia
3	6789	Mediamarkt	Coslada	Central	Madrid
4	58	Tienda47	Manzanares	Colina	Madrid
5	59	Tienda50	Manzanares	Barrio2	Madrid
6	60	Tienda51	Manzanares	Barrio3	Madrid
7	2	Tienda2	Barcelona	Canaletas	Catalunya
8	1000	Tienda3	Sevilla	Giralda	Andalucia
9	2000	Tienda4	Madrid	Barrio Sal...	Madrid

2. 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 la configuración del modo replicación, tenemos que crear una copia de seguridad del nodo maestro y enviarla al esclavo. Primero creamos las dos carpetas siguientes, una para guardar la copia de seguridad y otra para el wal:



Nombre	Fecha de modificación	Tipo	Tamaño
antlr	17/12/2019 17:19	Carpeta de archivos	
Archivos de programa	03/06/2020 12:41	Carpeta de archivos	
Archivos de programa (x86)	01/06/2020 22:36	Carpeta de archivos	
Intel	03/12/2019 22:56	Carpeta de archivos	
jar	31/03/2020 16:04	Carpeta de archivos	
Microsoft	26/01/2020 13:10	Carpeta de archivos	
MSI	07/12/2019 22:52	Carpeta de archivos	
PerfLogs	14/05/2020 3:35	Carpeta de archivos	
pg_basebackup	03/06/2020 12:31	Carpeta de archivos	
Riot Games	09/04/2020 11:25	Carpeta de archivos	
SteamLibrary	05/05/2020 20:27	Carpeta de archivos	
Usuarios	03/12/2019 9:38	Carpeta de archivos	
walbackup	03/06/2020 12:42	Carpeta de archivos	
Windows	14/05/2020 3:35	Carpeta de archivos	

En postgresql.conf, cambiamos lo siguiente:

```
# - Archiving -
```

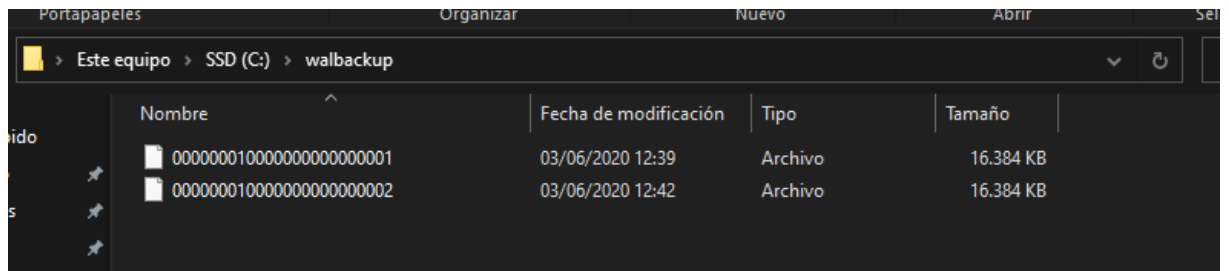
```
archive_mode = on
```

```
archive_command = 'copy "%p" "C:\\walbackup\\%f"'
```

```
# - Settings -
```

```
wal_level = replica
```

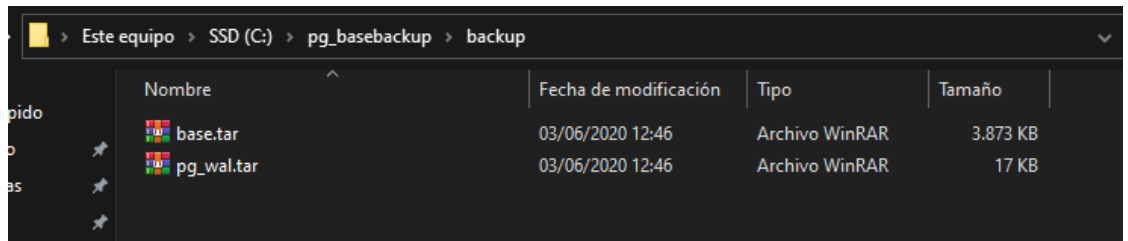
Vemos que se copian bien los archivos del wal:



Introducimos lo siguiente en la consola para crear la copia:

```
C:\>cd "Program Files"
C:\Program Files>cd PostgreSQL
C:\Program Files\PostgreSQL>CD 12
C:\Program Files\PostgreSQL\12>cd bin
C:\Program Files\PostgreSQL\12\bin>pg_basebackup -D "C:\pg_basebackup\backup" -Ft -z -U postgres
Contraseña:
```

Vemos que se crea correctamente:

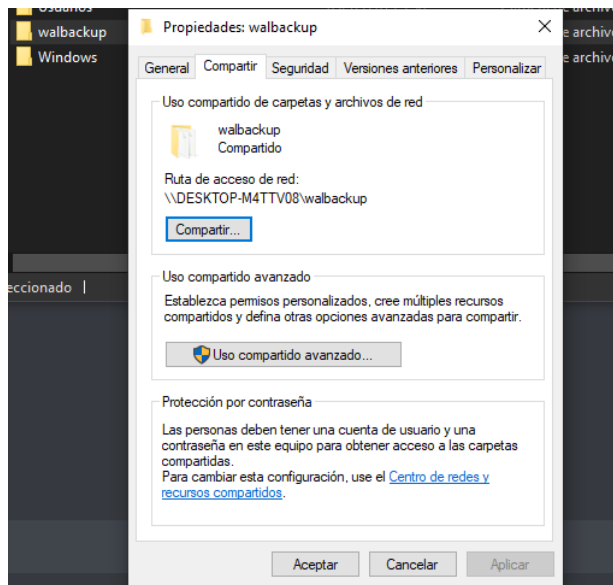


Ahora, en el esclavo, hacemos los pasos de recuperación. Detenemos el servidor, cambiamos el nombre a la carpeta data y restauramos la copia de seguridad del maestro que será el nuevo data (clúster):

En postgresql.conf comentamos las líneas que habíamos cambiado anteriormente y restauramos el wal introduciendo lo siguiente:

```
restore_command = 'copy "\\DESKTOP-M4TTV08\walbackup\%f" "%p"'
```

La dirección introducida es la de la carpeta compartida del wal del maestro, a la que accedemos desde el esclavo:



Ahora, para la replicación en la línea de primary_conninfo, tenemos que introducir lo siguiente para conectarnos al maestro:

```
primary_conninfo = 'host=192.168.1.50 port=5432 user=postgres password=postgres'
# (change requires restart)
```

En el maestro, en el pg_hba.conf introducimos la IP del esclavo (192.168.1.39) para confirmar la replicación:

#	TYPE	DATABASE	USER	ADDRESS	METHOD
# IPv4 local connections:					
host	all		all	127.0.0.1/32	md5
host	all		all	88.16.172.227/32	md5
# IPv6 local connections:					
host	all		all	:::1/128	md5
# Allow replication connections from localhost, by a user with the					
# replication privilege.					
host	replication		all	192.168.1.39/32	md5
host	replication		all	127.0.0.1/32	md5
host	replication		all	:::1/128	md5

En el esclavo creamos los archivos recovery.signal y standby.signal y reiniciamos el servidor.

En el log, vemos que todo ha funcionado perfectamente:

```
2020-06-04 13:13:21.218 CEST [8564] LOG: el sistema de bases de datos fue interrumpido; última vez en funcionamiento en 2020-06-04 13:06:57 CEST
2020-06-04 13:13:23.018 CEST [8564] LOG: entrando al modo standby
2020-06-04 13:13:23.166 CEST [8564] LOG: el sistema de bases de datos no fue apagado apropiadamente; se está efectuando la recuperación automática
2020-06-04 13:13:23.221 CEST [8564] LOG: redo comienza en 0/1600000A0
2020-06-04 13:13:23.222 CEST [8564] LOG: largo de registro no válido en 0/1600000D8: se esperaba 24, se obtuvo 0
2020-06-04 13:13:23.252 CEST [8564] LOG: el estado de recuperación consistente fue alcanzado en 0/1600000D8
2020-06-04 13:13:23.255 CEST [648] LOG: el sistema de bases de datos está listo para aceptar conexiones de sólo lectura
2020-06-04 13:13:23.537 CEST [4568] LOG: iniciando el flujo de WAL desde el primario en 0/160000000 en el timeline 2
La conexión coincidió con la línea 83 de pg_hba.conf: «host all all :::1/128 md5»
```

Ahora, si en el maestro hacemos una modificación de una tabla, nos aparecerá también en el esclavo:

Maestro:


```
28
29 insert into "Tienda" values(13,'Corte Ingles','Madrid','Barrio Salamanca','Galicia');
30
```

Data Output Explain Messages Notifications

INSERT 0 1

Query returned successfully in 33 msec.

Esclavo:

```
1 select * from "Tienda";
```

	Id_tienda [PK] integer	Nombre text	Ciudad text	Barrio text	Provincia text
1	1	Tienda1	Madrid	Vicalvaro	Madrid
2	2	Tienda2	Madrid	Alcala	Madrid
3	3	Tienda3	A Coruña	A Coruña	Galicia
4	4	Tienda4	Murcia	Plaza Cas...	Murcia
5	5	Tienda5	Huelva	Rojas	Andalucia
6	6	Tienda6	Logroño	Avenida	La Rioja
7	7	Tienda7	Torrejon	Calle lugo	Madrid
8	8	Tienda8	Alcala de H...	Santos ni...	Madrid
9	9	Alcampo	Guadalajara	Avenida G...	Madrid
10	10	MediaMarkt	Guadalajara	Avenida G...	Madrid
11	11	Carrefour	Vallecas	Calle Gra...	Madrid
12	12	Carrefour	Lugo	Preto	Galicia
13	13	Corte Ingles	Madrid	Barrio Sal...	Galicia

Realizamos los mismos pasos para el otro maestro (192.168.1.69) y su esclavo (192.168.1.48).

Existen dos tipos de nodos maestros: unos encargados de las operaciones de lectura y escritura en otros nodos y otros dedicados a la configuración, diagnóstico y mantenimiento de otro nodo. En nuestro caso se han utilizado los de lectura y escritura.

En cuanto a los nodos esclavos solo puede haber cambios en cuanto a la conexión, ya que solo pueden hacer operaciones de lectura.

Hemos elegido el método Write-Ahead Log Shipping cuya comunicación es a través del WAL, donde se utiliza una carpeta compartida en la red local a la que acceden el esclavo y maestro.

En la siguiente tabla podemos ver las diferentes soluciones con sus características:

Feature	Shared Disk Failover	File System Replication	Write-Ahead Log Shipping	Logical Replication	Trigger-Based Master-Standby Replication	Statement-Based Replication Middleware	Asynchronous Multimaster Replication	Synchronous Multimaster Replication
Most common implementations	NAS	DRBD	built-in streaming replication	built-in logical replication, pglogical	Londiste, Slony	pgpool-II	Bucardo	
Communication method	shared disk	disk blocks	WAL	logical decoding	table rows	SQL	table rows	table rows and row locks
No special hardware required		•	•	•	•	•	•	•
Allows multiple master servers				•		•	•	•
No master server overhead	•		•	•		•		
No waiting for multiple servers	•		with sync off	with sync off	•		•	
Master failure will never lose data	•	•	with sync on	with sync on		•		•
Replicas accept read-only queries			with hot standby	•	•	•	•	•
Per-table granularity				•	•		•	•
No conflict resolution necessary	•	•	•		•			•

- 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.

De maestro a maestro:

Select: se pueden hacer operaciones de selección de maestro a maestro

```

16 select * from Tienda2|"Tienda";
17
18
19
20

```

Data Output

Explain

Messages

Notifications

	Id_tienda integer	Nombre text	Ciudad text	Barrio text	Provincia text
5	5	Tienda5	Huelva	Rojas	Andalucia
6	6	Tienda6	Logroño	Avenida	La Rioja
7	7	Tienda7	Torrejon	Calle lugo	Madrid
8	8	Tienda8	Alcala de H...	Santos ni...	Madrid
9	9	Alcampo	Guadalajara	Avenida G...	Madrid
10	10	MediaMarkt	Guadalajara	Avenida G...	Madrid
11	11	Carrefour	Vallecas	Calle Gra...	Madrid
12	12	Carrefour	Lugo	Preto	Galicia
13	13	Corte Ingles	Madrid	Barrio Sal...	Galicia

Insert: se pueden hacer operaciones de inserción de maestro a maestro

```
27 INSERT INTO tiendal."Tienda" VALUES(33, 'Alcampo', 'Camarma', 'Calle Era', 'Madrid');
```

Data Output Explain Messages Notifications

INSERT 0 1

Query returned successfully in 110 msec.

```
25 SELECT * FROM tiendal."Tienda";
26
27 INSERT INTO tiendal."Tienda" VALUES(33, 'Alcampo', 'Camarma', 'Calle Era', 'Madrid');
```

Data Output Explain Messages Notifications

	Id_tienda integer	Nombre text	Cludad text	Barrio text	Provincia text
1	1	Tienda1	Madrid	Vicalvaro	Madrid
2	31145	Tienda Alcala	Cadiz	Puerto de...	Andalucia
3	6789	Mediamarkt	Coslada	Central	Madrid
4	58	Tienda47	Manzanares	Colina	Madrid
5	59	Tienda50	Manzanares	Barrio2	Madrid
6	60	Tienda51	Manzanares	Barrio3	Madrid
7	2	Tienda2	Barcelona	Canaletas	Catalunya
8	33	Alcampo	Camarma	Calle Era	Madrid
9	1000	Tienda3	Sevilla	Giralda	Andalucia
10	2000	Tienda4	Madrid	Barrio Sal...	Madrid

Delete: se pueden hacer operaciones de eliminación de maestro a maestro

```
29 DELETE FROM tiendal."Tienda" WHERE "Nombre" = 'Alcampo';
30
```

Data Output Explain Messages Notifications

DELETE 1

Query returned successfully in 181 msec.

Update: se pueden hacer operaciones de modificación de maestro a maestro

```
31 UPDATE tiendal."Tienda" SET "Nombre" = 'Corte Ingles' WHERE "Nombre" = 'Tiendal';
```

Data Output Explain Messages Notifications

UPDATE 1

Query returned successfully in 60 msec.

Resultado del delete y update:

	Id_tienda integer	Nombre text	Ciudad text	Barrio text	Provincia text
1	31145	Tienda Alcala	Cadiz	Puerto de...	Andalucia
2	6789	Mediamarkt	Coslada	Central	Madrid
3	58	Tienda47	Manzanares	Colina	Madrid
4	59	Tienda50	Manzanares	Barrio2	Madrid
5	60	Tienda51	Manzanares	Barrio3	Madrid
6	2	Tienda2	Barcelona	Canaletas	Catalunya
7	1	Corte Ingles	Madrid	Vicalvaro	Madrid
8	1000	Tienda3	Sevilla	Giralda	Andalucia
9	2000	Tienda4	Madrid	Barrio Sal...	Madrid

De maestro a esclavo: el esclavo replica la base de datos del maestro. El maestro hace modificaciones en su base de datos y el esclavo accede a ellas, por lo que no tiene sentido decir que el maestro realiza operaciones en el esclavo.

De esclavo a maestro: el esclavo accede a la información de la base de datos del maestro, por lo que sí puede hacer operaciones de selección:

```
1 select * from "Tienda";
```

	Id_tienda [PK] integer	Nombre text	Ciudad text	Barrio text	Provincia text
1	1	Tienda1	Madrid	Vicalvaro	Madrid
2	2	Tienda2	Madrid	Alcala	Madrid
3	3	Tienda3	A Coruña	A Coruña	Galicia
4	4	Tienda4	Murcia	Plaza Cas...	Murcia
5	5	Tienda5	Huelva	Rojas	Andalucia
6	6	Tienda6	Logroño	Avenida	La Rioja
7	7	Tienda7	Torrejon	Calle lugo	Madrid
8	8	Tienda8	Alcala de H...	Santos ni...	Madrid
9	9	Alcampo	Guadalajara	Avenida G...	Madrid
10	10	MediaMarkt	Guadalajara	Avenida G...	Madrid
11	11	Carrefour	Vallecas	Calle Gra...	Madrid
12	12	Carrefour	Lugo	Preto	Galicia
13	13	Corte Ingles	Madrid	Barrio Sal...	Galicia

El resto de las operaciones no se pueden realizar, ya que solo puede consultar la información:

```
3 INSERT INTO "Tienda" VALUES(33, 'Alcampo', 'Camarma', 'Calle Era', 'Madrid');
4
```

Data Output Explain Messages Notifications

ERROR: no se puede ejecutar INSERT en una transacción de sólo lectura
SQL state: 25006

```
5 DELETE FROM "Tienda" WHERE "Nombre" = 'Alcampo';
6
```

Data Output Explain Messages Notifications

ERROR: no se puede ejecutar DELETE en una transacción de sólo lectura
SQL state: 25006

```
7 UPDATE "Tienda" SET "Nombre" = 'Corte Ingles' WHERE "Nombre" = 'Tienda1';
8
```

Data Output Explain Messages Notifications

ERROR: no se puede ejecutar UPDATE en una transacción de sólo lectura
SQL state: 25006

Desconectamos el servidor de un maestro y vemos las respuestas de su esclavo y del otro maestro al intentar conectarse a él. En el otro maestro si intentamos hacer alguna operación, nos dirá que no se ha podido conectar al servidor:

Data Output Explain Messages Notifications

ERROR: could not connect to server "tienda_alberto"
DETAIL: no se pudo conectar con el servidor: Connection timed out (0x0000274C/10060)
Est el servidor en ejecución en el servidor 83.57.35.56 y aceptando conexiones TCP/IP en el puerto 5432?
SQL state: 08001

En el log nos aparece acceso denegado:

```
2020-06-05 13:42:11.124 CEST [2284] SENTENCIA: select * from Tienda2."Tienda";
Acceso denegado.

2020-06-05 13:42:14.580 CEST [14820] LOG: la orden de archivado falló con código de retorno 1
2020-06-05 13:42:14.580 CEST [14820] DETALLE: La orden fallida era: «copy "pg_wal\0000000200000009000000E4" "C:\wal_backup\0000000200000009000000E4"»
Acceso denegado.

2020-06-05 13:42:15.606 CEST [14820] LOG: la orden de archivado falló con código de retorno 1
2020-06-05 13:42:15.606 CEST [14820] DETALLE: La orden fallida era: «copy "pg_wal\0000000200000009000000E4" "C:\wal_backup\0000000200000009000000E4"»
Acceso denegado.

2020-06-05 13:42:16.625 CEST [14820] LOG: la orden de archivado falló con código de retorno 1
2020-06-05 13:42:16.625 CEST [14820] DETALLE: La orden fallida era: «copy "pg_wal\0000000200000009000000E4" "C:\wal_backup\0000000200000009000000E4"»
2020-06-05 13:42:16.625 CEST [14820] WARNING: el archivado del archivo de WAL «0000000200000009000000E4» falló demasiadas veces, se tratará de nuevo más tarde
```

En el esclavo al intentar hacer una operación de selección nos aparece lo siguiente:

```

2  select * from "Tienda";
3

```

	Id_tienda [PK] integer	Nombre text	Ciudad text	Barrio text	Provincia text
1	1	Tienda1	Madrid	Vicalvaro	Madrid
2	2	Tienda2	Madrid	Alcala	Madrid
3	3	Tienda3	A Coruña	A Coruña	Galicia
4	4	Tienda4	Murcia	Plaza Cas...	Murcia
5	5	Tienda5	Huelva	Rojas	Andalucía
6	6	Tienda6	Logroño	Avenida	La Rioja
7	7	Tienda7	Torrejon	Calle lugo	Madrid
8	8	Tienda8	Alcala de H...	Santos ni...	Madrid
9	9	Alcampo	Guadalajara	Avenida G...	Madrid
10	10	MediaMarkt	Guadalajara	Avenida G...	Madrid
11	11	Carrefour	Vallecas	Calle Gra...	Madrid
12	12	Carrefour	Lugo	Preto	Galicia
13	13	Corte Ingles	Madrid	Barrio Sal...	Galicia

Como se puede apreciar, se accede a los datos sin problemas, ya que el esclavo es una copia de la base de datos del maestro, por lo que el esclavo podrá seguir accediendo a la información de la última copia del maestro. En el log, podemos ver que la replicación ha terminado y que se ha perdido la conexión con el maestro:

```

2020-06-05 13:46:03.933 CEST [17112] LOG:  replicación terminada por el servidor primario
2020-06-05 13:46:03.933 CEST [17112] DETALLE:  Se alcanzó el fin de WAL en el timeline 2 en la posición 0/170000A0.
2020-06-05 13:46:03.941 CEST [17112] FATAL:  no se pudo enviar el mensaje fin-de-flujo al primario: no hay COPY alguno en ejecución
2020-06-05 13:46:04.183 CEST [8564] LOG:  registro con prev-link 0/13000028 incorrecto en 0/170000A0
2020-06-05 13:46:25.321 CEST [7540] FATAL:  no se pudo conectar al servidor primario: no se pudo conectar con el servidor: Connection timed out (0x0000274C/10060)
¿Está el servidor en ejecución en el servidor «192.168.1.50» y aceptando conexiones TCP/IP en el puerto 5432?
2020-06-05 13:46:40.686 CEST [6412] LOG:  iniciando el flujo de WAL desde el primario en 0/17000000 en el timeline 2

```

Si se cae el nodo esclavo, no tendrá ningún impacto en el funcionamiento del sistema, ya que solo es una copia del maestro. Si se recupera después, se podrán actualizar los datos utilizando recovery, el modo de recuperación ya explicado.

Para corregir la situación de caída de un nodo maestro y convertir al esclavo en maestro, tenemos que utilizar el comando `pg_ctl promote` (manualmente o con un trigger), esto hace que acabe el modo standby y se convierta en un nodo independiente, y así, podrá realizar todo tipo de operaciones. También, si el antiguo maestro se recupera, hay que comunicarle que ya no es maestro para evitar inconsistencias.

4. 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

Insertamos datos en cada una de las tablas de los dos maestros. La consulta es la siguiente:

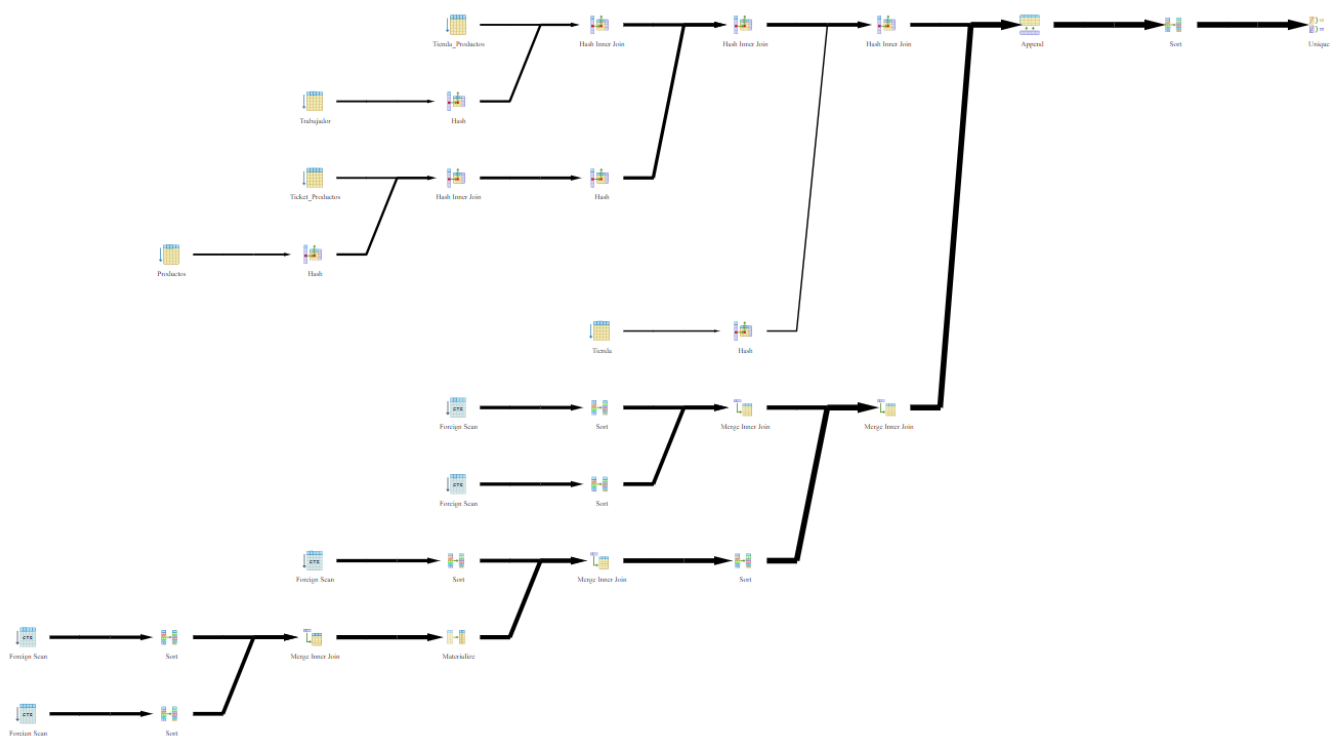
```
(select "Trabajador"."Nombre","Tienda"."Nombre" from "Trabajador" inner join "Tienda" on "Id_tienda"="Id_tienda_Tienda"
inner join "Tienda_Productos" on "Id_tienda"="Tienda_Productos"."Id_tienda_Tienda" inner join "Productos"
on "Codigo de barras_Productos"="Codigo de barras"
inner join "Ticket_Productos" on "Ticket_Productos"."Codigo de barras_Productos"="Codigo de barras")
union
(select Tienda3."Trabajador"."Nombre",Tienda3."Tienda"."Nombre" from Tienda3."Trabajador"
inner join Tienda3."Tienda" on "Id_tienda"="Id_tienda_Tienda"
inner join Tienda3."Tienda_Productos" on "Id_tienda"=Tienda3."Tienda_Productos"."Id_tienda_Tienda"
inner join Tienda3."Productos"
on "Codigo de barras_Productos"="Codigo de barras"
inner join Tienda3."Ticket_Productos" on Tienda3."Ticket_Productos"."Codigo de barras_Productos"="Codigo de barras")
;
```

Donde la primera parte de la unión es la la base de datos del maestro actual y la segunda la del otro maestro. En la segunda parte de la consulta, Tienda3 es la base de datos del otro maestro (tabla foránea) y al estar ambas bases de datos estructuradas de la misma manera, hacemos las mismas operaciones. Con la unión de ambas consultas se mostrarán la información de ambas bases de datos.

El resultado es el siguiente:

	Nombre character varying	Nombre text
1	Carmen	Tienda2
2	Celia	Tienda2
3	Fernando	Tienda50
4	Marta	Tienda1
5	Pedro	Tienda4
6	Roberto	Tienda3
7	Siro	Tienda1
8	Siro	Tienda3

El plan de ejecución es el siguiente:



En el plan de ejecución, se puede ver que se utiliza la operación foreign scan, que consulta la información de la otra base de datos.

5. 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?

Si el nodo MAESTRO1 falla, el esclavo empezará con el procedimiento de failover. Si el failover ocurre, el esclavo se convierte en maestro y el antiguo maestro se reinicia. Debe haber un mecanismo que informe al antiguo maestro que ya no es maestro (STONITH), necesario para evitar situaciones donde ambos sistemas se piensan que son el maestro y esto puede llevar a una confusión y pérdida de datos. Para activar el failover, podemos usar `pg_ctl promote`.

Después de hacer el `pg_ctl`, ya puede actuar como maestro, también puede estar conectado con el MAESTRO2 y mantener los datos de la base de datos distribuida.

Ahora, el nuevo maestro hay que configurarlo para conectarlo al otro maestro. Hacemos los mismos pasos que realizamos anteriormente (abrir el puerto del router, poner las IPs en el archivo `pg_hba.conf`, crear un servidor con la IP del otro maestro...).

Podemos tener dos configuraciones nuevas: una con el anterior esclavo ahora como maestro y el MAESTRO2 con su esclavo igual que estaban anteriormente. Y otra, con el anterior esclavo como maestro, el anterior

maestro como su esclavo y el MAESTRO2 con su esclavo igual que estaban anteriormente. En la segunda hay que tener cuidado de avisar al nuevo esclavo (anteriormente como maestro) de que ya no es maestro.

6. 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?

El sistema puede funcionar de forma asíncrona, por lo que sí puede haber inconsistencias. Estas inconsistencias pueden ser debidas a retrasos en la propagación de los datos a otros servidores, por lo que hay veces que incluso se pueden perder algunas transacciones. Por ejemplo, si el maestro hace una actualización en la base de datos, es posible que el esclavo aún no pueda acceder a esa información, ya que puede que el WAL no se haya actualizado aún en el esclavo debido a un retraso en la sincronización de los datos. La comunicación asíncrona se utiliza cuando la síncrona es demasiado lenta.

Si se utiliza una comunicación síncrona se garantiza la consistencia de los datos, ya que una transacción de modificación no es confirmada hasta que el resto de los servidores la hayan confirmado.

7. Conclusiones.

En esta práctica hemos creado una base de datos distribuida utilizando nodos maestros y nodos esclavos. Los nodos maestros se comunican entre ellos estando cada uno de ellos en diferentes redes y pudiendo acceder a la información de la base de datos del otro maestro. Los nodos esclavos, acceden a leer información de la base de datos de un nodo maestro estando ambos en la misma red local.

Este tipo de base de datos distribuida tiene varios beneficios, como una mayor seguridad ante caídas, ya que, si un nodo se cae, podemos acceder a los datos del nodo caído a través de otro nodo. También, otro beneficio es la facilidad para atender a varios clientes, al no centrar las peticiones únicamente en un nodo, aumentando la velocidad, ya que podemos distribuir la carga, evitando saturaciones.

Se ha visto también, cuál es el método más interesante dependiendo de los requerimiento: asíncrono si queremos más velocidad a cambio de una posible pérdida de datos o síncrono si queremos asegurarnos de que los datos llegan de forma correcta.

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 Availability, Load Balancing, and Replication.

- Appendix F: Additional Supplied Modules. F.33. Postgres_fdw