# 3.-Bases de datos distribuidas con SQL

En esta práctica veremos como usar la extensión de PostgreSQL Citus para almacenar y consultar una base de datos en un cluster de varias máquinas. La información necesaria para esta práctica está disponible en la documentación de Citus en la siguiente URL:

https://docs.citusdata.com/en/v10.0/

## 3.1 Instalación de Citus en Ubuntu

Este paso no es necesario hacerlo, ya que Citus ya está instalado en las máquinas "GeiBD" y "GreiBDServerBase". Los pasos necesarios para la instalación son los siguientes. Primero configuramos el repositorio.

```
curl https://install.citusdata.com/community/deb.sh | sudo bash
```

Instalamos PostgreSQL y Citus.

```
sudo apt-get -y install postgresql-13-citus-10.0
```

Configuramos la precarga de la extensión Citus en PostgreSQL. Se añade la extensión Citus a la lista de extensiones para precargar en el archivo "/etc/postgresql/13/main/postgresql.conf". El siguiente comando hace este paso.

```
sudo pg_conftool 13 main set shared_preload_libraries citus
```

Configuramos las conexiones de red para PostgreSQL. Primero modificamos la variable "listen_addresses" en "/etc/postgresql/13/main/postgresql.conf".

```
sudo pg_conftool 13 main set listen_addresses '*'
```

Para simplificar la configuración, damos permisos para conexión a postgresql a cualquier usuario sin password (hay información en la documentación de citus para restringir estos accesos). Configuramos las siguientes líneas del archivo "/etc/postgresql/13/main/pg_hba.conf" (reemplazar abajo la base de la ip de la subred si fuese necesario 192.168.56.0).

```
sudo nano /etc/postgresql/13/main/pg_hba.conf

host all all 192.168.56.0/24 trust
host all all 127.0.0.1/32 trust
```

Reiniciamos el servidor para actualizar la configuración y habilitamos su inicio automático

```
sudo service postgresql restart
sudo update-rc.d postgresql enable
```

Creamos la extensión citus en la base de datos postgres y habilitamos un password para el usuario postgres

```
sudo -i -u postgres psql -c "CREATE EXTENSION citus"
sudo -i -u postgres psql -c "ALTER USER postgres WITH PASSWORD 'greibd2021'"
```

## 3.2 Creación del cluster virtual

Para crear el cluster utilizaremos la máquina virtual "GreiBD" como nodo coordinador y dos copias de "GreiBDServerBase" que clonaremos como nodos worker.

Para clonar la máquina elegir la opción "clonar" en Virtual Box, elegir un nombre para la máquina (citus2 y citus3), elegir una carpeta en donde almacenar la copia de la máquina y asegurarse de generar nueva dirección mac para la tarjeta de red (para no tener problemas a la hora de tener todas las máquinas en la misma red local). Elegir finalmente la opción de clonación completa.

Después de clonar cada máquina, debemos cambiar el nombre y el identificador en el sistema operativo. Primero cambiamos el nombre de la máquina

```
sudo hostnamectl set-hostname citus2
sudo hostnamectl
```

Después borramos el identificador y volvemos a generarlo.

```
sudo rm /etc/machine-id
sudo dbus-uuidgen --ensure=/etc/machine-id
dbus-uuidgen --ensure
```

Anotamos la IP de la máquina (consultar usando ifconfig) y reiniciamos.

```
sudo shutdown -r now
```

Repetimos estos pasos para las tres máquinas.

Una vez tenemos las tres máquinas funcionando, para simplificar el acceso a las bases de datos vamos a configurar conexiones desde DBeaver.

Añadimos la IPs y puertos de los dos nodos worker (citus2 y citus3) al coordinador (GreiBD). Para esto ejecutamos las dos instrucciones siguientes (¡Reemplazar las ips de las máquinas!)

```
SELECT * from citus_add_node('192.168.56.9', 5432);
SELECT * from citus_add_node('192.168.56.10', 5432);
```

Verificamos que la instalación es correcta.

```
SELECT * FROM citus_get_active_worker_nodes();
```

## 3.3 Ejemplo de uso: base de datos de películas.

Primero creamos el esquema de la base de datos, para eso ejecutamos el siguiente comando. Podemos usar un cliente ssh que nos permita copiar y pegar para facilitar el trabajo.

```
chmod 664 /home/alumnogreibd/BDGE/datos/PeliculasSchema.sql
sudo -i -u postgres psql -f /home/alumnogreibd/BDGE/datos/PeliculasSchema.sql
```

Ahora vamos a distribuir las tablas. Elegiremos el identificador de la película para distribuir la tabla de películas y las tablas de las relaciones. Las demás tablas serán tablas de referencias, que replicaremos en todos los nodos. Es importante seguir el orden correcto en la distribución de las tablas para evitar errores derivados de las claves foráneas.

```
--tablas de referencia
SELECT create_reference_table('colecciones');
SELECT create_reference_table('generos');
SELECT create_reference_table('idiomas');
SELECT create_reference_table('paises');
SELECT create_reference_table('personas');
SELECT create_reference_table('productoras');

--tablas distribuídas
SELECT create_distributed_table('peliculas', 'id');
SELECT create_distributed_table('pelicula_genero', 'pelicula', colocate_with => 'peliculas');
SELECT create_distributed_table('pelicula_idioma_hablado', 'pelicula', colocate_with => 'peliculas');
SELECT create_distributed_table('pelicula_pais', 'pelicula', colocate_with => 'peliculas');
SELECT create_distributed_table('pelicula_personal', 'pelicula', colocate_with => 'peliculas');
SELECT create_distributed_table('pelicula_productora', 'pelicula', colocate_with => 'peliculas');
SELECT create_distributed_table('pelicula_reparto', 'pelicula', colocate_with => 'peliculas');
```

Ya podemos ejecutar las instrucciones de inserción de datos (puede tardar un rato dada la cantidad de datos a insertar. Ahora la tabla de personas tiene que replicarla en dos nodos).

```
chmod 664 /home/alumnogreibd/BDGE/datos/*.csv
sudo -i -u postgres psql -c "\copy colecciones from /home/alumnogreibd/BDGE/datos/colecciones.csv csv"
sudo -i -u postgres psql -c "\copy generos from /home/alumnogreibd/BDGE/datos/generos.csv csv"
sudo -i -u postgres psql -c "\copy idiomas from /home/alumnogreibd/BDGE/datos/idiomas.csv csv"
sudo -i -u postgres psql -c "\copy paises from /home/alumnogreibd/BDGE/datos/paises.csv csv"
sudo -i -u postgres psql -c "\copy personas from /home/alumnogreibd/BDGE/datos/personas.csv csv"
sudo -i -u postgres psql -c "\copy productoras from /home/alumnogreibd/BDGE/datos/productoras.csv csv"
sudo -i -u postgres psql -c "\copy peliculas from /home/alumnogreibd/BDGE/datos/peliculas.csv csv"
sudo -i -u postgres psql -c "\copy pelicula_genero from /home/alumnogreibd/BDGE/datos/pelicula_genero.csv csv"
sudo -i -u postgres psql -c "\copy pelicula_idioma_hablado from /home/alumnogreibd/BDGE/datos/pelicula_idioma_hablado.csv csv"
sudo -i -u postgres psql -c "\copy pelicula_pais from /home/alumnogreibd/BDGE/datos/pelicula_pais.csv csv"
sudo -i -u postgres psql -c "\copy pelicula_personal from /home/alumnogreibd/BDGE/datos/pelicula_personal.csv csv"
sudo -i -u postgres psql -c "\copy pelicula_productora from /home/alumnogreibd/BDGE/datos/pelicula_productora.csv csv"
sudo -i -u postgres psql -c "\copy pelicula_reparto from /home/alumnogreibd/BDGE/datos/pelicula_reparto.csv csv"
```

Ahora podemos comprobar como se han almacenado las tablas en los dos workers (citus2 y citus3).

Como observación, podemos ver que el uso de agregación en el modelo objeto-relacional, o con XML o JSON, replica todos los datos que aquí estamos replicando con las tablas de referencia. En el caso de los datos agregados, los datos de referencia se repiten para cada película, mientras que aquí se replican para cada nodo worker.

Ahora podemos probar algunas de las consultas.

1.- Obtener todas las películas dirigidas por "Ridley Scott". Ordena el resultado de forma descendiente por fecha de emisión.

```
select pels.titulo, pels.fecha_emision
from peliculas pels, pelicula_personal pp , personas per
where pels.id=pp.pelicula and pp.persona = per.id
  and per.nombre = 'Ridley Scott'
  and pp.trabajo = 'Director'
order by fecha_emision desc
```

2.- Para cada actor/actriz principal (orden < 5), obtener el número de películas en las que participó y la cantidad de beneficios que han generados dichas películas. Devuelve solo las 10 primeras filas ordenas por beneficio.

```
select per.nombre, count(pels.id) as peliculas, sum(pels.ingresos-pels.presupuesto) as beneficio
from peliculas pels, pelicula_reparto pr, personas per
where pels.id=pr.pelicula and pr.persona=per.id
  and pr.orden<5
group by per.nombre
order by beneficio desc
limit 10
```

Apagamos uno de los worker y comprobamos que el sistema ya no es capaz de responder a las consultas (podemos pausar la ejecución de la máquina virtual).

# Ejercicios

Realiza estos ejercicios y genera un documento de texto de explicación de cada paso que hagas. Entrega el documento a través del campus virtual.

1.- Probar el uso de la replicación. Para esto, borramos todas la tablas (en cascada) y las creamos de nuevo. Antes de crearlas modificar el script para eliminar las claves foráneas cláusulas references, ya que no se puede usar replicación en CITUS con claves foráneas. Antes de distribuir los datos, asignar el valor 2 a la variable de factor de replicación ("SET citus.shard_replication_factor = 2"). Cargar de nuevo los datos. Probar una de las consultas anteriores. Pausar una de las máquinas worker. Probar de nuevo la consulta. Intentar varias veces si es necesario, para que el coordinador se de cuenta que uno de los worker no responde.

2.- Buscar información sobre el formato de almacenamiento columnar en citus (https://docs.citusdata.com/en/v10.1/admin_guide/table_management.html#columnar-storage). Probar a crear la tabla "peliculas" en formato columnar y comprobar si hay ganancia en tiempo de ejecución en las consultas anteriores.

Última modificación: venres, 15 de outubro de 2021, 13:47