UNPSJB

LICENCIATURA EN SISTEMAS OPGCPI

BASES DE DATOS II

Trabajo Práctico 5

Datawarehouse

Cátedra
Lic. Gabriel Ingravallo
Lic. Cristian Parisse

Integrantes:
Luciano Serruya Aloisi
Pablo Toledo Margalef

11 de junio de 2018

Índice

1. Alta Disponibilidad, Balanceo de Carga, y Réplica

Estudie las formas de implementar Alta Disponibilidad, Balanceo de carga y replicación en postgres (capitulo 26 de la ayuda), escriba un resumen y un cuadro explicativo de cada alternativa planteada en la documentación, detallando el software que se debe utilizar para implementarlo.

El motor de bases de datos *PostgreSQL* detalla en su documentación distintas formas posibles de implementar los conceptos de **Alta disponibilidad**, **Balanceo de carga**, y **Replicación**. Antes de detallar brevemente cada uno, se definen los conceptos de

- **Alta disponibilidad**: propiedad de las *Bases de Datos Distribuidas* (*BDD*) que permite tener acceso a una relación *r* cuando uno de los sitios que contenga dicha relación falle [6]. También se puede definir como la probabilidad de que el sistema esté accesible durante un periodo de tiempo [2].
- **Balanceo de carga**: propiedad que permite que varios servidores provean los mismos datos [4]
- **Replicación**: mantener copias idénticas de la relación almacenadas en distintos sitios que conforman la *BDD* [6]

Hace falta agregar que en la gran mayoría de las soluciones propuestas por el motor se habla de una arquitectura de *BDD* de tipo *maestro-esclavo*, donde existe un único servidor que es el que escribe en la base de datos (maestro ó primario), y otros que solamente llevan registro de los cambios (esclavos o *en espera*); se subdividen los servidores esclavos en "servidores tibios" (*warm*) y "servidores calientes" (*hot*). A los primeros solamente se puede conectar cuando se cae el servidor principal, y toman su puesto, mientras que a los segundos siempre se puede conectar, para realizar consultas de solo lectura.

Las distintas soluciones que presenta *Postgres* en su documentación son las siguientes:

1. Tolerancia a fallos por disco compartido (*Shared Disk Failover*): se mantiene una única copia de la base de datos que es accedida por múltiples servidores. Si el servidor principal falla, el servidor esclavo puede realizar una recuperación como si se tratase de una caída del sistema.

2. Replicación del sistema de archivos (*File System Replication*): todos los cambios realizados al sistema de archivos de la base de datos son duplicados en otro sistema de archivos ubicado en otra máquina. La copia debe garantizar la consistencia de los datos entre ambas versiones.

3. Envío de la bitácora de transacciones (*Transaction Log Shipping*): los servidores esclavos se pueden mantener actualizados enviándoles registros de la bitácora de lectura adelantada. Si el servidor principal se cae, los esclavos pueden tomar su puesto rápidamente ya que tienen casi toda la información.

4. Replicación *maestro-esclavo* basado en *triggers* (*Trigger-Based Master-Standby Replication*): las consultas de modificación son enviadas al servidor maestro, para que este asíncronamente envíe los cambios a los esclavos (que solamente pueden realizar consultas de lectura). Este tipo de replicación se puede implementar con el módulo **Slony-I**.

5. *Middleware* de replicación basado en sentencias (*Statement-Based Replication Middleware*): se intercepta cada sentencia SQL y se la envía a cada uno de los servidores que componen la *BDD* (cada uno opera independientemente). Las cosultas de escritura son enviadas a todos los servidores, así se mantienen actualizados, mientras que las de lecturas pueden ser enviadas a uno solo (se balancea el trabajo de lectura). Se puede implementar con **Pgpool-II** y con **Continuent Tungsten**

6. Replicación asíncrona de múltiples maestros (*Asynchronous Multimaster Replication*): cada servidor trabaja independientemente del resto, y periódicamente se comunican entre sí para identificar transacciones que entran en conflicto. Estos conflictos pueden ser resueltos por el usuario o establecer de antemano ciertas reglas de resolución. Puede ser implementado con el módulo **Bucardo**

7. Replicación síncrona de múltiples maestros (*Synchronous Multimaster Replication*): cualquier servidor puede aceptar consultas de escritura, y los cambios son transmitidos desde el servidor que realizó la actulización hacia el resto *antes de que finalice la transacción.* Consultas de lectura pueden ser ejecutadas por cualquier servidor. El motor no provee este tipo de replicación, pero puede ser implementado mediante el *protocolo de compromiso en dos fases.*

Solución	¿Asíncrono?	Cuello de botella	Software adicional
Tolerancia a fallos por disco compartido	✗	Fallos del disco	-
Replicación del sistema de archivos	✗	Integridad de la copia	-
Envío de la bitácora de transacciones	Ambas	Velocidad de la red	
Replicación *maestro-esclavo* basado en *triggers*	✓	Velocidad de la red	Slony-I
***Middleware* de replicación basado en sentencias**	✓	Sincronización entre servidores	Pgpool-II, Continuent Tungsten
Replicación asíncrona de múltiples maestros	✓	Sincronización entre servidores y resolución de conflictos	Bucardo
Replicación síncrona de múltiples maestros	✗	Sincronización entre servidores	-

1.1. Replicación en *PostgreSQL*: *Slony-I*

Slony-I es un módulo para *PostgreSQL* que implementa *un sistema de replicación "maestro con múltiples esclavos"* [3]. Soporta **réplica en cascada** (un nodo sirve a otro nodo), y es tolerante a fallos.

Algunos conceptos que maneja el módulo son:

- **Cluster**: conjunto de instancias de bases de datos involucradas en la replicación.
- **Nodo**: integrantes del cluster.
- **Conjunto de réplica**: conjunto de objetos de la base de datos a ser replicados; pueden haber varios en un cluster.
- **Origen**: nodo principal (maestro); es el único en el que se puede escribir.
- **Suscriptores**: nodos esclavos; reciben datos de la réplica.
- **Proveedor**: nodo esclavo que provee de datos a los otros nodos esclavos (actúa como un nodo maestro pero no se permite ninguna escritura en él).

La versión 9.0 de *PostgreSQL* incorpora *replicación por flujos*, sin embargo hay ciertos casos de uso donde se necesitaría una herramiento como *Slony-I* ya que las técnicas nativas del motor no alcanzan [7]:

1. Interactuar con distintas versiones de *Postgres*: *Slony-I* soporta esta interacción, mientras que la replicación propia del DBMS no.

2. Replicación parcial: la replicación basada en *logs de escritura adelantada* replica todo.

3. Implementar comportamiento adicional en los esclavos.

2. Fragmentos de la base *aviones*

Sobre la base de datos de aviones, crear fragmentos para los pilotos de Trelew, pilotos de Madryn, y todos los pilotos. Hacer esquema de fragmentación y de asignación

La definición de cada fragmento se puede encontrar en los archivos ***pilotos_de_madryn.sql***, ***pilotos_de_trelew.sql***, y ***todos_pilotos.sql*** dentro del directorio *fragmentos_aviones*. Cada consulta incluye una descripción y nombre de fragmento.

Sitio Trelew	Sitio Madryn	Sitio Todos
localidad_trelew	***localidad_madryn***	***localidad_todos***
piloto_trelew	***piloto_madryn***	***piloto_todos***
trabajador_trelew	***trabajador_madryn***	***trabajador_todos***
piloto_avion_trelew	***piloto_avion_madryn***	***piloto_avion_todos***
avion_trelew	***avion_madryn***	***avion_todos***
modelo_avion_trelew	***modelo_avion_madryn***	***modelo_avion_todos***

Cuadro 1: Esquema de asignación

3. Bases de Datos Distribuidas con *dblink*

3.1. Introducción

El módulo *dblink* permite hacer conexiones a otras bases de de datos *PostgreSQL* desde la sesión actual [5]. Para utilizarlo hace falta crear la extensión en la base de datos en la que se lo vaya a usar

```
1 universidad=# CREATE EXTENSION dblink;
2 CREATE EXTENSION
```

A continuación se explicarán brevemente las primitivas de *dblink* que se utilizarán posteriormente en este laboratorio

- *dblink_connect*: establecer una conexión con otra base de datos; la conexión puede ser nombrada o no. Mínimamente se necesita pasar como parámetro el nombre de la base de datos a la que se intenta conectar, si es que está en el mismo servidor. Caso contrario, se le debe indicar la dirección IP del servidor, puerto de escucha, y credenciales (usuario y contraseña).
- *dblink_disconnect*: cierra una conexión (nombrada) con otra base de datos.
- *dblink*: ejecuta una consulta en una base de datos remota. Se le debe pasar como parámetros la conexión (ya sea el nombre, o toda la cadena de conexión), y la consulta que se quiera realizar. También se debe definir el formato de la tabla que devolverá la consulta

```
1     universidad=# SELECT * FROM dblink('conn_cursos', 'SELECT
        nombre FROM cursos') AS t(nombre varchar(50));
```

Consulta remota al sitio de cursos

- *dblink_exec*: ejecuta una consulta remota pero no devuelve resultados. Al igual que la primitiva anterior, se le debe pasar la conexión y la consulta.

3.2. Implementación

Con el módulo dbLink de Postgres establecer una conexión entre dos servidores posibles; en el servidor local crear las tablas "alumnos" e "inscriptos", en el remoto crear la tabla "cursos"

Para la implementación de la base de datos distribuidas, se optó por crear dos *servicios* mediante contenedores de Docker®. Cada uno corre una servidor de *PostgreSQL 9.6* con un usuario con privilegios de superusuario. La definición y el manejo de estos servicios se llevó a cabo utilizando Docker-Compose® [1].

El servicio nombrado "sitio-alumnos" hará de servidor local, mientras que "sitio-cursos" será el remoto.

Cada servicio se crea con los siguientes parámetros:

- Usuario: *admin*
- Contraseña: *admin*
- Nombre de la base de datos: *universidad*

3.2.1. Creación de las tablas

Para crear las tablas en el servidor local y el remoto, se ejecutó el siguiente script en el sitio de alumnos

[1] Para ver cómo están definidos, referirse al archivo *docker-compose.yml* y al archivo *README.md*

```sql
CREATE TABLE alumnos (
    dni integer PRIMARY KEY,
    nombre varchar(50)
);

-- Conectar al servidor 172.20.0.20 (servidor remoto, sitio con los cursos) con las credenciales admin:admin
SELECT dblink_connect('conn_cursos', 'hostaddr=172.20.0.20 dbname=universidad user=admin password=admin');

-- Utilizar la conexión creada anteriormente para ejecutar una consulta remota
SELECT dblink_exec('conn_cursos', 'CREATE TABLE cursos (id_curso integer PRIMARY KEY, nombre varchar(50), id_profesor integer, departamento integer)');

CREATE TABLE inscriptos (
    dni_alumno integer REFERENCES alumnos (dni),
    id_curso integer,
    anio integer,
    nota integer,
    PRIMARY KEY (dni_alumno, id_curso, anio)
);

-- Cerrar la conexión con la BD remota
SELECT dblink_disconnect('conn_cursos');
```

Definición de las tablas (el script SQL es *crear_tablas_locales_remotas.sql*)

3.2.2. Carga de datos

Los scripts generados para la carga de datos fueron creados utilizando el sitio [1]. Debido la extensión de los scripts no se mostrarán en este informe, pero están disponibles en el directorio *cargas/*, cada uno con un nombre representativo según su tabla.

Para la tabla *inscriptos*, se cargaron datos que pueden existir en la tabla *cursos*, y otros que no (según se pedía en el enunciado).

3.3. Consultas

Ya poblada las tablas que componen la base de datos distribuidas, se pueden ejecutar consultas con datos locales y datos remotos utilizando el módulo *dblink*.

3.3.1. Cursos con inscriptos

```
SELECT dblink_connect('conn_cursos', 'hostaddr=172.20.0.20 user=
   admin password=admin dbname=universidad');

SELECT
    distinct nombre
FROM
    dblink('conn_cursos', 'SELECT id_curso, nombre FROM cursos')
    AS t(id_curso integer, nombre text)
JOIN inscriptos AS ins
    ON t.id_curso=ins.id_curso;

SELECT dblink_disconnect('conn_cursos');
```

Cursos con algún inscripto (el script es *cursos_con_inscriptos.sql*)

En la consulta anterior, se muestran una única vez los nombres de los cursos (desde el servidor remoto) que tengan al menos un inscripto registrado (en e servidor local).

3.3.2. Alumnos recursantes

```
-- Alumnos recursantes

SELECT dblink_connect('conn_cursos', 'hostaddr=172.20.0.20 user=
   admin password=admin dbname=universidad');

SELECT
    al.nombre AS "Alumno",
    ins.dni_alumno AS "DNI",
    cur.nombre AS "Curso",
    COUNT(ins.id_curso) AS "Cuántas veces"
FROM inscriptos AS ins
JOIN alumnos AS al
    ON ins.dni_alumno = al.dni
JOIN (SELECT * FROM dblink('conn_cursos', 'SELECT id_curso, nombre
   FROM cursos') AS t(id_curso integer, nombre text)) AS cur
    ON ins.id_curso=cur.id_curso
GROUP BY
    al.nombre,
    ins.dni_alumno,
    cur.nombre
HAVING COUNT(ins.id_curso) > 1
ORDER BY al.nombre;

SELECT dblink_disconnect('conn_cursos');
```

Alumnos que cursaron más de una vez una materia (el script es *recursantes.sql*)

Los alumnos recursantes estarán registrados varias veces con un mismo curso en la tabla local *inscriptos*.

3.3.3. Verificar la existencia de los cursos en la tabla *inscriptos*

Para recorrer la tabla de inscripciones y verificar si cada curso registrado existe en la tabla remota *cursos*, se creó una función *verificar_curso_inscripto()* que itera sobre la tabla (valiéndose de la posibilidad que provee *Pl/PgSQL* para iterar sobre el resultado de una consulta en un ciclo *FOR*) y consulta en la tabla remota si existe ese curso. Debido a la extensión de la función, no se presentará en este informe, pero se la puede encontrar en el archivo ***verificar_curso_inscripto.sql***

Referencias

[1] Benjamin Keen. *Sitio generador de datos*. 2018. URL: http://www.generatedata.com/.

[2] Ramez Elmasri; Shamkant B. Navathe. *Fundamentos de Sistemas de Bases de Datos*. Quinta edición. Pearson Education, 2007.

[3] PostgreSQL. *Entrada de Slony-I en la Wiki de PostgreSQL*. 2016. URL: https://wiki.postgresql.org/wiki/Slony.

[4] PostgreSQL. «PostgreSQL 9.6.9 Documentation». En: The PostgreSQL Global Development Group, 2018. Cap. 26. High Availability, Load Balancing, and Replication.

[5] PostgreSQL. «PostgreSQL 9.6.9 Documentation». En: The PostgreSQL Global Development Group, 2018. Cap. Appendix F. Additional Supplied Modules.

[6] Abraham Silberschatz. *Fundamentos de Bases de Datos*. Cuarta edición. McGraw-Hill, 2002.

[7] Slony. *Sitio de Slony*. 2010. URL: http://www.slony.info/.