3.2. Administración de la base de datos

En estos días, las bases de datos han cobrado vital importancia. No sólo por ser los almacenes que guardan la actividad diaria de la empresa, sino también porque la disponibilidad de información se ha convertido en una ventaja competitiva para las organizaciones. De aquí que, mantener en óptimo funcionamiento el sistema manejador de bases de datos es una tarea prioritaria de todo centro de informática.

El responsable de asegurar la funcionalidad y eficiencia de una base de datos organizacional es conocido como Administrador de Bases de Datos (Database Administrador, DBA). Las principales tareas que realiza el DBA son: la administración del servidor de bases de datos (instalación, configuración, monitoreo y actualización del sistema manejador de bases de datos), seguridad, respaldo y recuperación, importación y exportación de datos, y ajustes de rendimiento.

A continuación conocerás un poco más fondo algunas de las actividades que debe realizar un administrador de bases de datos.

3.2.1. Administración del servidor

Lo más importante para un DBA es mantener disponibles las bases de datos para los usuarios y aplicaciones que los necesiten. Es por esto que necesita vigilar y programar tareas automáticas que le permitan generar alertas ante cualquier posible falla del sistema.

Otro aspecto importante a considerar es el tiempo necesario para realizar tareas rutinarias de administración y durante el cual, muchas veces el sistema debe estar detenido. Esto era muy común en el pasado, hoy en día los manejadores de bases de datos han mejorado mucho y han logrado disminuir este tiempo.

Por ejemplo, para respaldar una base de datos, muchos manejadores necesitan que la base esté fuera de uso, es decir, sin acceso de ningún usuario. Si se trata de la base de datos de un sistema de transacciones en línea (OLTP), ese tiempo significa la negación del servicio a los usuarios y resulta costoso. El respaldo, entonces, se prefiere en las horas o días de menor uso de la base de datos. El DBA debe tener en cuenta estos aspectos para asegurar la disponibilidad de las bases de datos.

Instalación del DBMS

No se debe pensar en esta tarea como algo simple. Es necesario conocer y entender los prerrequisitos de instalación (versión del sistema operativo, cantidad de memoria, tipo de procesador). Además, es sumamente importante leer el manual de instalación para conocer los detalles necesarios para que el sistema quede en óptimas condiciones.

Configuración del DBMS

Todo DBMS cuenta con parámetros configurables que modifican su comportamiento. Ejemplos de ellos son: número de tablas disponibles al mismo tiempo, cantidad de memoria caché, cantidad de memoria para tablas temporales, número máximo de usuarios que pueden estar conectados, etc. En el manual de cada manejador se encuentran explicados estos parámetros.

Los parámetros son ajustables generalmente de dos formas. Pueden ser establecidos mediante comandos o a través de archivos de configuración. Si se trata de modificar archivos, es altamente recomendable respaldar el archivo de configuración original antes de modificarlo. En el caso de PostgreSQL, el archivo que guarda toda la configuración del servidor es el archivo postgresql.conf.

Actualización del DBMS

Los DBMS no están a salvo de un acelerado cambio de versiones, que incluyen mejoras al software y a las interfaces gráficas de administración. Al parecer, el

tiempo promedio de vida de una versión de software es, a lo mucho, de dieciocho meses. Estas nuevas versiones corrigen errores de programación, optimizan procesos y mejoran la seguridad de los sistemas, por tanto, el DBA necesita conocer de su liberación.

Lo importante es conocer si conviene cambiar de versión y, de ser así, qué impacto tendrá en nuestras aplicaciones. Muchas veces, el cambio de versión de un DBMS acarrea consigo cambios en la programación de las aplicaciones y de la misma base de datos. También es importante conocer los pasos necesarios para realizar estas actualizaciones, ya que una base de datos puede dejar de funcionar completamente en una nueva versión si no se siguen los pasos de actualización correctos.

Iniciar y detener el servidor de PostgreSQL

Iniciando PostgreSQL

Para iniciar el servidor de bases de datos utilizamos el siguiente comando:

```
pg_ctl start [-w] [-D DATADIR] [-s] [-l FILENAME] [-o "OPTIONS"]
```

`[-w].-` Espera el fin de la instrucción para regresar el prompt.

`[-D DATADIR].-` Especifica el directorio con las bases de datos.

`[-s].-` Suprime los mensajes de salida a la pantalla.

`[-l FILENAME].-` Especifica un archivo en donde se registrará la actividad de la base de datos.

Deteniendo PostgreSQL

El comando para detener el servidor de PostgreSQL es el siguiente.

```
pg_ctl stop [-W] [-D DATADIR] [-s] [-m SHUTDOWN-MODE]
```

`[-W].` No espera el fin de la instrucción para regresar el prompt.

`[-m SHUTDOWN-MODE].` Indica el modo de detener el servidor:

- `Smart` – Espera la desconexión de todos los clientes.
- `Fast` – Detiene el servidor sin esperar la desconexión de los clientes.
- `Immediate` – Es más abrupto que fast y provoca que la base inicie en modo recovery la siguiente vez.

3.2.2. Administración del catálogo

El **catálogo** consiste en el conjunto de tablas de sistema que guardan información sobre los objetos de la base de datos. Así, podemos encontrar una tabla que describa los datos de todas las tablas de usuario; también, una tabla que describa los datos de las bases de datos; otra tabla que describa los usuarios; etc. Todo DBA necesita conocer con que **tablas de catálogo** cuenta, en caso de que necesite información específica de los objetos de la base. Dado que son tablas, es posible hacer consultas basadas en la instrucción SELECT con el fin de personalizar la información del catálogo. Por ejemplo en PostgreSQL, algunas tablas de sistema son: pg_constraint, pg_database, pg_group, pg_indexes, pg_tables y pg_user. Para ver las tablas del catálogo puedes utilizar el comando \dS, de la siguiente forma:

```
TEST=# \dS
                   Listado de relaciones
   Schema   |           Nombre           |   Tipo   |  Due±o
------------+----------------------------+----------+----------
 pg_catalog | pg_aggregate               | tabla    | postgres
 pg_catalog | pg_attrdef                 | tabla    | postgres
 pg_catalog | pg_attribute               | tabla    | postgres
 pg_catalog | pg_cast                    | tabla    | postgres
 pg_catalog | pg_class                   | tabla    | postgres
 pg_catalog | pg_constraint              | tabla    | postgres
 pg_catalog | pg_database                | tabla    | postgres
 pg_catalog | pg_group                   | tabla    | postgres
 pg_catalog | pg_index                   | tabla    | postgres
 pg_catalog | pg_indexes                 | vista    | postgres
 pg_catalog | pg_tables                  | vista    | postgres
 pg_catalog | pg_tablespace              | tabla    | postgres
 pg_catalog | pg_trigger                 | tabla    | postgres
 pg_catalog | pg_type                    | tabla    | postgres
 pg_catalog | pg_user                    | vista    | postgres
```

Manejar la información del catálogo le permite al DBA administrar mejor los objetos de la base de datos. Imagínate la cantidad de tablas, vistas y usuarios que debe tener una organización. Con el tiempo, es necesario eliminar tablas y objetos que ya nos son utilizados o que fueron usados para realizar pruebas y desarrollo. Un buen DBA automatiza estas laboras de administración mediante procedimientos almacenados y consultas que utilizan el catálogo.

3.2.3. Seguridad

Una de las responsabilidades más importantes del DBA es asegurar que sólo los usuarios autorizados puedan entrar a la base de datos. De igual manera tiene que prevenir que los usuarios no vean o modifiquen datos para los que no tienen autorización.

Por lo general, estas actividades se realizan mediante instrucciones SQL (GRANT y REVOKE), creación de vistas para ocultar datos confidenciales y configuraciones de acceso al sistema de bases de datos.

A continuación vamos a revisar la administración de usuarios y privilegio con el fin de que tengas idea clara de esta labor.

Administración de usuarios y grupos

El uso de usuarios y grupos permiten controlar el acceso a los objetos de la base de datos. En PostgreSQL existen usuarios y grupos distintos a los usuarios y grupos del sistema operativo. Toda conexión al servidor debe ser hecha con un usuario que pertenece a uno o varios grupos. Los usuarios tienen o no derecho de realizar acciones en el servidor. Los grupos permiten simplificar el trabajo de administrar varios derechos.

Usuarios

Todo usuario se identifica con un *username* que puede ser independiente de su cuenta de sistema operativo (*system account*). Tiene además un ID de sistema llamado *sysid* y una contraseña (*password*). El ID de sistema sirve para asociar al usuario como propietario (*owner*) de los objetos dentro de la base.

Existen derechos llamados *globales* que determinan si un usuario puede crear o eliminar bases de datos del servidor. También indican si el usuario es *superusuario* (tiene todos los derechos en todas las bases).

Toda la información de los usuarios se guarda en la tabla de sistema pg_shadow. Para ver los usuarios podemos utilizar la vista pg_user. Por ejemplo:

```
TEST=# SELECT * FROM pg_user;
 usename  | usesysid | usecreatedb | usesuper | usecatupd |  passwd  |  valuntil| useconfig
----------+----------+-------------+----------+-----------+----------+----------+----------
 postgres |        1 | t           | t        | t         | ******** |          |
 (2 rows)
TEST=# select * from pg_shadow;
 usename  | usesysid | usecreatedb | usesuper | usecatupd | passwd | valuntil | useconfig
----------+----------+-------------+----------+-----------+--------+----------+-----------
 postgres |        1 | t           | t        | t         |        |          |
(1 row)
```

Crear usuarios

Hay dos métodos: con el comando CREATE USER de SQL o con el comando createuser de PostgreSQL.

Crear usuarios con SQL

```
Syntax:
CREATE USER username [ [ WITH ] option [ ... ] ]
where option can be:
      SYSID uid
    | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
    | CREATEDB | NOCREATEDB
    | CREATEUSER | NOCREATEUSER
    | IN GROUP groupname [, ...]
    | VALID UNTIL 'abstime'
```

Para crear un *superuser* lo hacemos incluyendo la opción CREATEUSER que además de permitir crear usuarios, implícitamente convierte al usuario en súper usuario.

Crear usuarios con createuser

Se ejecuta desde la línea de comando del sistema operativo. Es interactivo y permite definir algunas opciones. También puede utilizarse indicando opciones pertinentes para la creación del usuario. Por ejemplo:

```
postgres> createuser
Enter name of user to add: borrame
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) n
CREATE USER
```

Modificar usuarios

Podemos modificar todas las opciones con las que se creo el usuario excepto la del sysid.

```
Syntax:
ALTER USER username [ [ WITH ] option [ ... ] ]
where option can be:
      [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
    | CREATEDB | NOCREATEDB
    | CREATEUSER | NOCREATEUSER
    | VALID UNTIL 'abstime'
```

Eliminar usuarios

No podemos borrar usuarios si son propietarios de alguna base de datos. Tenemos también dos formas de hacerlo: con SQL y con el comando ***dropuser***.

Por ejemplo:

Borremos al usuario que acabamos de crear desde la línea de comando

```
postgres> dropuser
Enter name of user to delete: borrame
DROP USER
```

Grupos

Creando grupos

Utilizamos el comando CREATE GROUP de la siguiente manera:

```
Syntax:
CREATE GROUP groupname [WITH
USER username [, …] ]
```

Por ejemplo.

```
CREATE GROUP desarrollo
      WITH USER cmendezc;
```

Borrando grupos

```
Syntax:
DROP GROUP groupname
```

Asociando usuarios a grupos

```
Syntax:
ALTER TABLE groupname
      { ADD | DROP } USER username [, ...]
```

Privilegios

PostgreSQL mantiene un conjunto de listas de control de acceso (ACL's) en donde se almacenan los privilegios que tienen los grupos y usuarios respecto a los objetos de la base de datos. Los principales privilegios son: SELECT, INSERT, UPDATE y DELETE.

Asignando y quitando privilegios

Los derechos o privilegios se asignan con GRANT y se quitan con REVOKE. Ambos tienen la misma sintaxis.

```
Syntax:
GRANT privilege [, ...] ON object [, ...]
      TO {PUBLIC | username | GROUP groupname}
REVOKE privilege [, ...] ON object [, ...]
      TO {PUBLIC | username | GROUP groupname}
```

Por ejemplo.

```
GRANT ALL ON tautor, tlibro, ttema
      TO cmendezc;
```

Revisando privilegios

Podemos revisar los privilegios de un objeto con el comando \z y el nombre del objeto.

3.2.4. Respaldos

Hemos sido redundantes al hablar de la importancia que tiene para una organización la información contenida en sus bases de datos. Sufrir daños o pérdidas de información son situaciones muy costosas para las empresas, desafortunadamente, ninguna está exenta de que esto le pueda suceder. Todo manejador de bases de datos en funcionamiento conlleva la posibilidad de una

falla de hardware o software, de un ataque fructífero o hasta de un desastre natural que lo dañe por completo. Muchas veces, un error humano también puede causar modificaciones dañinas a la información de una base de datos.

El DBA debe estar preparado para responder a estas situaciones y a la consiguiente recuperación y restauración de la información correcta. Esto lo hace mediante una estrategia de respaldo que permita, en el menor tiempo posible, recuperar al sistema de una falla. Para la formulación de esta estrategia, será necesario tomar en cuenta aspectos como el tiempo de recuperación de una falla, el tipo de respaldo, el medio de almacenamiento de los respaldos, la periodicidad de los respaldos y los simulacros de falla que permitirán saber si la estrategia es óptima.

Respaldo

Utilizamos el comando pg_dump. Podemos respaldar en formato de texto, formato comprimido (gzip) y a un archivo tipo tar. Si lo hacemos en archivo de texto, es como si generáramos los scripts de creación y actualización de la base de datos y sus objetos. Algunas opciones del comando pg_dump se muestran a continuación:

`-a` Respalda datos y no el esquema.

`-b` Respalda objetos largos.

`-C` El comando CREATE DATABASE se incluye en el respaldo.

`-f FILENAME` Especifica el archivo en donde se guardará el respaldo.

`-F {c | t | p}` Indica el tipo de respaldo:

- `c` – gzip
- `t` – tar
- `p` – texto plano

`-o` Respalda los oid.

`-O` Elimina el *owner* en el respaldo.

`-s` Respalda sólo el esquema.

`-t` TABLE Indica que se respalde sólo una tabla.

Recuperación

Utilizamos el comando pg_restore. Casi todas las opciones son las mismas que usamos en pg_dump, así que lo mejor es poner las mismas opciones que se utilizaron para respaldar. Algunas opciones adicionales son:

`-i` Sólo restaura los índices.

`-f` Sólo restaura funciones.

`-s` Sólo restaura el esquema.

`-T NAME` Sólo restaura el trigger indicado en NAME.

3.2.5. Importación y exportación de datos

Actualmente, todos los DBMS cuentan con opciones de importación y exportación de datos. Éstas responden a las necesidades de intercambio electrónico de información. Por ejemplo, muchas empresas entregan su información fiscal al SAT por medio de archivos de texto. Para lograr esto, se necesita exportar la información desde el manejador de bases de datos.

Importar y exportar datos en PostgreSQL

Para copiar (importar) datos desde un archivo hacia una de las tablas de nuestra base de datos utilizamos el comando COPY. Debemos usar un archivo de tipo texto ASCII separado por comas o tabuladores.

```
Syntax:
COPY table [ ( column [, ...] ) ]
    FROM { 'filename' | stdin }
    [ [ WITH ]
          [ BINARY ]
          [ OIDS ]
          [ DELIMITER [ AS ] 'delimiter' ]
          [ NULL [ AS ] 'null string' ] ]
```

`[ BINARY ].` Indica que la entrada de datos proviene de un archivo binario creado previamente con el mismo comando COPY.

3.2.6. Monitoreo del sistema

Por ejemplo, algunos manejadores de bases de datos definen el tamaño de una base de datos desde que ésta es creada. Con forme pasa el tiempo y se van acumulando datos, por consiguiente el espacio reservado para la base va disminuyendo. Es imprescindible que el DBA realice un monitoreo constante del espacio libre faltante ya que si el espacio se termina, la base de datos deja de funcionar y los servicios de información se detienen. Muchos de estos mismos sistemas, permiten crear **tareas de vigilancia** que mandan una alerta en línea o por correo electrónico cuando el espacio libre llega a cierto porcentaje mínimo.

3.2.7. Programación de tareas rutinarias

La tarea rutinaria por excelencia es el respaldo de los datos. En el pasado, los DBA tenían que trabajar de noche y días festivos con el fin de realizar los respaldos de las bases de datos. Eso ya cambió, ya que hoy contamos con herramientas de software cada vez más sofisticadas. Hoy en día, la práctica general es dejar programado el respaldo de las bases mediante una tarea (*task*) o un plan de mantenimiento. Es entonces el propio servidor quien se encarga de ejecutar en el momento preciso el respaldo y entregar un reporte de su resultado. Este reporte puede ir al correo electrónico del DBA o a una bitácora del sistema.

3.2.8. Ajustes de configuración de rendimiento

El rendimiento de una base de datos está determinado por el grado de satisfacción con el que un sistema administrador de bases de datos (*Database Management System*, DBMS) responde a las peticiones de información de un usuario. Al menos los siguientes factores intervienen en el rendimiento de un manejador: carga de trabajo (*workload*), capacidad de procesamiento (*throughput*), recursos, optimización, y contención.

La carga de trabajo es una combinación de las transacciones que atiende el DBMS, los trabajos programados, las peticiones de acceso a los datos, consultas a los datos y los comandos directos sobre el sistema. La carga de trabajo no es la

misma a todas horas y es importante para el DBA determinar los momentos de mayor carga.

Otro elemento que define el rendimiento de un DBMS es la **capacidad de procesamiento** de datos a nivel de hardware y de software. Está determinada por la velocidad del CPU, capacidades de procesamiento paralelo y eficiencia del sistema operativo. La capacidad de procesamiento tiene mucho que ver con los recursos con los que cuenta el sistema de cómputo, estos son, espacio en disco y cantidad de memoria.

La **optimización** se refiere al análisis de las peticiones a la base de datos con el fin de disminuir su costo de acceso a los datos. Se trata de modificar la manera en como se formulan las peticiones SQL con la idea de disminuir el tiempo de acceso a los datos.

Por su parte, la **contención** se refiere a la necesidad de administrar las peticiones de distintos usurarios a los mismos recursos al mismo tiempo.

El DBA necesita resolver problemas de rendimiento mediante una estrategia de monitoreo y afinación (*tunning*). Finalmente, es necesario decir que la actividad de mejorar el rendimiento de una base de datos puede ser compartida con otros especialistas en tecnología, como expertos en sistemas operativos, expertos en hardware y expertos en redes.