Capítulo 4
Definición de los datos

1. Introducción

El lenguaje de definición de los datos se basa en tres sentencias: `CREATE`, `ALTER` y `DROP`, que se aplican a cada objeto de la base de datos. Cada objeto se puede corresponder con un objeto físico, es decir, un archivo en los sistemas de archivos, o a un objeto lógico, es decir, una definición almacenada en el catálogo de la base de datos.

La ejecución de estas sentencias implica necesariamente un bloqueo exclusivo de los objetos afectados. Con este bloqueo exclusivo, no se puede atender ningún otro bloqueo en modo lectura o escritura. Esto no supone ningún problema durante la creación de un objeto, porque ninguna sesión actual lo puede haber reclamado, pero durante la eliminación de un objeto hay que esperar a que todos los bloqueos existentes se liberen. En lo que respecta a una base de datos, por ejemplo, no debe haber sesiones conectadas a esta base de datos.

El impacto más importante se produce durante la modificación de un objeto porque el bloqueo permanece durante todo el tiempo de ejecución del comando y, por lo tanto, esto puede tener un impacto en la ejecución de consultas concurrentes.

En consecuencia, las sentencias de definición de los datos se deben ejecutar con precaución, eligiendo si es posible una ventana de tiempo oportuna.

2. Los espacios de tablas

Un espacio de tablas es un directorio de un sistema de archivos, en el que PostgreSQL escribe los archivos de las tablas y de los índices. Por defecto, PostgreSQL dispone de un espacio de tablas ubicado en el directorio del grupo de bases de datos. Es posible crear otros espacios de tablas, que permiten al administrador seleccionar de esta manera la ubicación del almacenamiento de una tabla o de un índice.

Hay varios motivos que pueden hacer que un administrador cree un espacio de tablas:

– La partición ya no dispone de suficiente espacio libre. En este caso, los espacios de tablas permiten utilizar varios discos duros diferentes para un mismo grupo de base de datos, sin utilizar sistemas de volúmenes lógicos, como LVM en los sistemas GNU/Linux.

– La utilización de las tablas y de los índices provoca una saturación de las escrituras y lecturas del subsistema de disco en el que se ubica el espacio de tablas por defecto. Incluso en los sistemas de alto rendimiento, el escalado de una aplicación puede hacer que el administrador cree otros espacios de tablas en los subsistemas de discos diferentes. De esta manera, las escrituras y lecturas de archivos de tablas e índices se reparten en varios soportes físicos, mejorando el rendimiento.

Por lo tanto, un espacio de tablas es una herramienta que se puede utilizar por el administrador del servidor de bases de datos, que permite intervenir sobre la ubicación física del almacenamiento de las tablas y de los índices. Esto significa que el administrador puede elegir, tabla por tabla e índice por índice, independientemente de la base de datos, la ubicación de un archivo, optimizando de esta manera los volúmenes utilizados, las escrituras y las lecturas.

Un espacio de tablas no es específico a una base de datos. Forma parte de un grupo de bases de datos por defecto, se utiliza desde todas las bases de datos. Una administración fina de los permisos sobre los espacios de tablas permite al administrador controlar el reparto de los archivos, en función de las bases de datos y los roles utilizados.

La primera etapa antes de inicializar el espacio de tablas desde PostgreSQL es crear un directorio para este uso. Según las necesidades, se puede tratar de un directorio sobre un nuevo subsistema de discos o en una partición de un subsistema de discos ya presente. Las siguientes etapas permiten inicializar este directorio y asignar los permisos necesarios para que el usuario `postgres` sea el propietario:

```
[root]# mkdir -p /data2/postgres/tblspc2/
[root]# chown postgres:postgres /data2/postgres/tblspc2/
[root]# chmod 700 /data2/postgres/tblspc2/
```

Si el directorio ya existe, se debe vaciar y pertence al usuario del sistema operativo, que ejecuta la instancia de PostgreSQL.

Cuando se crea el directorio, es suficiente con guardarlo en la instancia de PostgreSQL, dándole un nombre que permita identificarlo. La sinopsis del comando es la siguiente:

```
postgres= # CREATE TABLESPACE nombretblspc [ OWNER nombrerol ]
LOCATION 'directorio';
```

Por defecto, el espacio de tablas creado de esta manera pertenece al usuario que ejecuta el comando. Solo un superusuario puede crear un espacio de tablas, pero puede transmitir la pertenencia a otro usuario por medio de la opción `OWNER`.

Una vez que existe el espacio de tablas, se puede utilizar cuando se crean o modifican las tablas e índices.

2.1 Modificación de un espacio de tablas

Es posible modificar un espacio de tablas existente. Hay dos argumentos modificables: el nombre y el propietario. El comando `ALTER TABLESPACE` permite hacer estas dos modificaciones, como en la siguiente sinopsis:

```
postgres=# ALTER TABLESPACE nombretblspc1 RENAME TO nombretblspc2;
postgres=# ALTER TABLESPACE nombretblspc2 OWNER TO nombrerol;
```

En función de las características del subsistema de discos utilizados por el espacio de tabla, es posible modificar las constantes de coste `seq_page_cost` y `random_page_cost` del planificador de consultas, modificando el coste de lectura de una página en disco:

```
postgres=# ALTER TABLESPACE nombretblspc2 set (seq_page_cost =
3, random_page_cost = 1 );
postgres=# ALTER TABLESPACE nombretblspc2 reset (seq_page_cost);
```

El significado de estas constantes se presenta en el capítulo Explotación.

2.2 Eliminación de un espacio de tablas

La eliminación de un espacio de tablas es posible si no existe ninguna tabla ni índice en este espacio de tablas, incluidos en una base de datos diferente a la de la conexión actual.

Antes de eliminar el espacio de tablas, hay que haber movido a otro espacio de tablas todos los objetos contenidos, incluidos los que no pertenezcan a la base de datos actual. Una vez que el espacio de tablas está vacío, la sentencia `DROP TABLESPACE` permite esta eliminación, como muestra la siguiente sinopsis:

```
postgres=# DROP TABLESPACE [ IF EXISTS ] nombretblspc;
```

Una vez que el espacio de tablas se elimina en la instancia de PostgreSQL, el directorio del sistema de archivos ya no es útil y se puede eliminar. La opción `IF EXISTS` evita que se provoque un error durante la eliminación si el espacio de tablas no existe.

3. Las bases de datos

En una instancia de PostgreSQL, una base de datos es un contenedor. Contiene los esquemas, las tablas, los índices y todos los objetos útiles para una aplicación. También recibe las conexiones desde las aplicaciones cliente. En efecto, cuando se abre una conexión en una base de datos particular, no es posible utilizar directamente los objetos creados en otras bases de datos.

Por lo tanto, es importante repartir correctamente los objetos y los datos de las aplicaciones en las bases de datos, principalmente utilizando la noción de esquema. La creación de una base de datos se puede realizar con la sentencia `CREATE DATABASE` o con el comando del sistema operativo `createdb`.

Algunos argumentos permiten personalizar la creación de una base de datos:

- Para crear una base de datos, es necesario ser superusuario o tener el permiso `CREATEDB`. Por el contrario, es posible transmitir la pertenencia a un usuario que no tiene permisos con la opción `OWNER`. La opción del comando `createdb` es `-O` o `--owner`.
- La base de datos `template1` sirve de modelo por defecto para la creación de otra base de datos. Para cada base de datos creada con este modelo, se hace una copia de `template1`, para que todos los objetos creados en `template1` se dupliquen en la nueva base. La base `template0` funciona de la misma manera, pero no es posible crear objetos en esta base-modelo: `template0` es una base de datos virgen. Es posible seleccionar la base de datos que sirve de modelo con la opción `TEMPLATE` y, por supuesto, es posible seleccionar cualquier base de datos modelo existente. La opción del comando `createdb` es `-T` o `--template`.
- Un argumento importante durante la creación de una base de datos es la elección del juego de caracteres. Determina la manera en la que los datos se almacenarán en las tablas y los índices. El juego de caracteres por defecto se determina durante la creación del grupo de base de datos, pero es posible indicar otro juego de caracteres con la opción `ENCODING`. No es posible modificar esta opción una vez que la base de datos se crea. Para seleccionar otra codificación diferente a la dada por defecto, es necesario utilizar la base de datos modelo `template0`, dando por hecho que el resto de las bases de datos modelo pueden contener datos incompatibles con la nueva codificación. La opción del comando `createdb` es `-E` o `--encoding`.
- Los argumentos `LC_COLLATE` y `LC_CTYPE` tienen un impacto en la localización de las ordenaciones de las cadenas de caracteres, los índices o durante la utilización de la sentencia `ORDER BY` de una consulta.

– El espacio de tablas por defecto es el que se ha creado durante la inicialización del grupo de bases de datos. Es posible crear otros espacios de tablas y asociarlos con una base de datos con la opción `TABLESPACE`. Pero esta opción solo es un argumento por defecto para la creación de las tablas e índices, que puede no servir. La opción del comando `createdb` es `-D` o `--location`.
– El último argumento, `CONNECTION LIMIT`, permite controlar el número de conexiones entrantes, que es ilimitado por defecto.

La siguiente sinopsis muestra la sentencia SQL que permite crear una base de datos:

```
postgres=# CREATE DATABASE nombre [ [ WITH ] [ OWNER [=] rol ]
[ TEMPLATE [=] modelo ] [ ENCODING [=] codificación ] [ LC_COLLATE [=]
lc_collate ] [ LC_CTYPE [=] lc_ctipo ] [ TABLESPACE [=] espaciotabla ]
[ CONNECTION LIMIT [=] límite_conexión ] ]
```

El siguiente comando permite crear una base de datos llamada `clientes` con el juego de caracteres UTF8:

```
postgres=# CREATE DATABASE clientes OWNER sebl ENCODING 'UTF8';
```

La siguiente sinopsis muestra el comando del sistema operativo:

```
[postgres]# createdb [-D tablaespacio|--tablespace=tablaespacio]
[-E codificación|--encoding=codificación] [--lc-collate=locale]
[--lc-ctype=locale] [-O owner|--owner=owner]
[-T modelo|--template= modelo] [nombrebase] [descripción]
```

Por lo tanto, la creación de la base `clientes` se puede escribir como se muestra a continuación:

```
[postgres]# createdb -O sebl clientes
```

Con el comando `createdb`, es posible conectarse a un servidor remoto utilizando las mismas opciones que el comando `psql`. Por ejemplo, el siguiente comando crea una base de datos en un servidor PostgreSQL remoto:

```
[root]# createdb -E UTF8 -h 192.168.0.3 -p 5432 -U postgres clientes
```