Chapitre 4
Définition des données

1. Introduction

Le langage de définition des données s'appuie sur trois ordres : `CREATE`, `ALTER` et `DROP` déclinés pour chaque objet de la base de données. Chaque objet peut correspondre soit à un objet physique, donc à des fichiers dans les systèmes de fichiers, soit à un objet logique, et donc seulement une définition stockée dans le catalogue de la base de données.

L'exécution de ces ordres implique nécessairement un verrouillage exclusif des objets concernés. Ce verrouillage étant exclusif, aucun autre verrou en lecture ou en écriture ne peut être maintenu. Ceci ne pose pas de problème lors de la création d'un objet puisqu'aucune session concurrente ne peut l'avoir réclamé mais lors de la suppression d'un objet, il faut attendre que tous les verrous existants soient relâchés. En ce qui concerne une base de données par exemple, il ne doit pas y avoir de sessions connectées à cette base.

L'impact le plus important concerne la modification d'un objet, car le verrou est conservé durant tout le temps d'exécution de la commande, et ceci peut donc avoir un impact sur l'exécution de requêtes concurrentes.

Les ordres de définition des données doivent donc être exécutés avec précaution si possible en choisissant une fenêtre de temps opportune.

2. Les espaces de tables

Un espace de tables est un répertoire d'un système de fichiers, dans lequel PostgreSQL écrit les fichiers des tables et des index. Par défaut, PostgreSQL dispose d'un espace de tables situé dans le répertoire du groupe de bases de données. Il est possible de créer d'autres espaces de tables, permettant ainsi à l'administrateur de choisir l'emplacement du stockage d'une table ou d'un index.

Plusieurs motifs peuvent amener un administrateur à créer un espace de tables :

– La partition ne dispose plus de suffisamment d'espace libre. Les espaces de tables permettent dans ce cas d'utiliser plusieurs disques durs différents pour un même groupe de base de données, sans utiliser de systèmes de volumes logiques, comme LVM dans les systèmes GNU/Linux.
– L'utilisation des tables et des index provoque une saturation des écritures et des lectures du sous-système disque où est situé l'espace de tables par défaut. Même sur des systèmes performants, la montée en charge d'une application peut amener l'administrateur à créer d'autres espaces de tables, sur des sous-systèmes disques différents. Ainsi, les écritures et les lectures de fichiers de tables et d'index sont réparties sur plusieurs supports physiques, améliorant les performances.

Un espace de tables est donc un outil, utilisable par l'administrateur du serveur de bases de données, permettant d'intervenir sur l'emplacement physique du stockage des tables et des index. L'administrateur peut donc choisir, table par table et index par index, quelle que soit la base de données, l'emplacement d'un fichier, optimisant ainsi les volumes utilisés, les écritures et les lectures.

Un espace de tables n'est pas spécifique à une base de données. Il fait partie d'un groupe de bases de données, et est, par défaut, utilisable depuis toutes les bases de données. Une gestion fine des permissions sur les espaces de tables permet à l'administrateur de maîtriser la répartition des fichiers, en fonction des bases de données et des rôles utilisés.

La première étape, avant d'initialiser l'espace de tables depuis PostgreSQL, est de créer un répertoire pour cet usage. Selon les besoins, il peut s'agir d'un répertoire sur un nouveau sous-système disques ou sur une partition d'un sous-système disques déjà présent. Les étapes suivantes permettent d'initialiser ce répertoire et de positionner les droits nécessaires pour que l'utilisateur `postgres` soit le propriétaire :

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

Si le répertoire existe déjà, il doit être vide et appartenir à l'utilisateur du système d'exploitation qui exécute l'instance de PostgreSQL.

Une fois le répertoire créé, il suffit de l'enregistrer dans l'instance de PostgreSQL, en lui donnant un nom qui permet de l'identifier. Le synopsis de la commande est le suivant :

```
postgres= # CREATE TABLESPACE nomtblspc [ OWNER nomrole ]
LOCATION 'repertoire';
```

Par défaut, l'espace de tables ainsi créé appartient à l'utilisateur qui exécute la commande. Seul un super-utilisateur peut créer un espace de tables, mais il peut transmettre l'appartenance à un autre utilisateur par l'intermédiaire de l'option `OWNER`.

Une fois que l'espace de tables existe, il peut être utilisé au moment de la création ou de la modification des tables et des index.

2.1 Modification d'un espace de tables

Il est possible de modifier un espace de tables existant. Deux paramètres sont modifiables : le nom et le propriétaire. La commande `ALTER TABLESPACE` permet de faire ces deux modifications, comme dans les synopsis suivants :

```
postgres=# ALTER TABLESPACE nomtblspc1 RENAME TO nomtblspc2 ;
postgres=# ALTER TABLESPACE nomtblspc2 OWNER TO nomrole ;
```

En fonction des caractéristiques du sous-système disques utilisés par l'espace de table, il est possible de modifier des constantes de coût `seq_page_cost` et `random_page_cost` du planificateur de requêtes, modifiant le coût de lecture d'une page sur disque :

```
postgres=# ALTER TABLESPACE nomtblspc2 set (seq_page_cost =
3, random_page_cost = 1 ) ;
postgres=# ALTER TABLESPACE nomtblspc2 reset (seq_page_cost) ;
```

Le sens de ces constantes est présenté au chapitre Exploitation.

2.2 Suppression d'un espace de tables

La suppression d'un espace de tables est possible s'il n'existe plus aucune table ni aucun index dans cet espace de tables, y compris dans une base de données autre que celle de la connexion courante.

Avant de supprimer l'espace de tables, il faut avoir déplacé dans un autre espace de tables tous les objets contenus, y compris ceux qui n'appartiennent pas à la base de données courante. Une fois que l'espace de tables est vide, l'ordre `DROP TABLESPACE` permet cette suppression, comme le montre le synopsis suivant :

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

Une fois que l'espace de tables est supprimé dans l'instance de PostgreSQL, le répertoire du système de fichiers n'est plus utile et peut à son tour être supprimé. L'option `IF EXISTS` permet de ne pas provoquer d'erreur lors de la suppression si l'espace de table n'existe pas.

3. Les bases de données

Dans une instance de PostgreSQL, une base de données est un conteneur. Elle contient des schémas, des tables, des index et tous les objets utiles à une application. Elle accueille aussi les connexions depuis les applications clientes. En effet, lorsqu'une connexion est ouverte sur une base de données particulière, il n'est pas possible d'utiliser directement des objets créés dans d'autres bases de données.

Il est donc important de répartir correctement les objets et données des applications dans les bases de données, notamment en utilisant la notion de schéma. La création d'une base de données peut s'effectuer avec l'ordre `CREATE DATABASE` ou avec la commande du système d'exploitation `createdb`.

Quelques paramètres permettent de personnaliser la création d'une base de données :

– Pour créer une base de données, il est nécessaire d'être superutilisateur ou d'avoir le privilège `CREATEDB`. En revanche, il est possible de transmettre l'appartenance à un utilisateur non privilégié, avec l'option `OWNER`. L'option de la commande `createdb` est `-O` ou `--owner`.
– La base de données `template1` sert de modèle par défaut pour la création d'une autre base de données. Pour chaque base créée avec ce modèle, une copie de `template1` est faite, pour que tous les objets créés dans `template1` sont dupliqués dans la nouvelle base. La base `template0` fonctionne de la même façon mais il n'est pas possible de créer des objets dans cette base-modèle : `template0` reste une base vierge. Il est possible de choisir la base de données servant de modèle avec l'option `TEMPLATE`. Il est bien sûr possible de choisir n'importe quelle base de données modèle existante. L'option de la commande `createdb` est `-T` ou `--template`.
– Un paramètre important lors de la création d'une base de données est le choix du jeu de caractères. Il détermine la façon dont les données seront stockées dans les tables et les index. Le jeu de caractères par défaut est déterminé à la création du groupe de base de données, mais il est possible de préciser un autre jeu de caractères avec l'option `ENCODING`. Il n'est pas possible de modifier cette option une fois la base de données créée. Pour choisir un autre encodage que celui par défaut, il est nécessaire d'utiliser la base de données modèle `template0` étant donné que les autres bases de données modèles peuvent contenir des données incompatibles avec le nouvel encodage. L'option de la commande `createdb` est `-E` ou `--encoding`.
– Les paramètres `LC_COLLATE` et `LC_CTYPE` ont un impact sur la localisation des tris des chaînes de caractères, dans les index ou lors d'utilisation de la clause `ORDER BY` d'une requête.

– L'espace de tables par défaut est celui qui a été créé à l'initialisation du groupe de bases de données. Il est possible de créer d'autres espaces de tables et de l'associer avec une base de données, avec l'option `TABLESPACE`. Mais ce choix n'est qu'un paramètre par défaut pour la création des tables et des index, qui peut ne pas être suivi. L'option de la commande `createdb` est `-D` ou `--location`.

– Le dernier paramètre, `CONNECTION LIMIT`, permet de contrôler le nombre de connexions entrantes qui, par défaut, est illimité.

Le synopsis suivant montre l'ordre SQL permettant de créer une base de données :

```
postgres=# CREATE DATABASE nom [ [ WITH ] [ OWNER [=] role ]
[ TEMPLATE [=] modele ] [ ENCODING [=] codage ] [ LC_COLLATE [=]
lc_collate ] [ LC_CTYPE [=] lc_ctype ] [ TABLESPACE [=] espacetable ]
[ CONNECTION LIMIT [=] limite_connexion ] ]
```

La commande ci-dessous permet de créer une base de données, appelée `clients`, avec le jeu de caractères UTF8 :

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

Le synopsis suivant montre la commande du système d'exploitation :

```
[postgres]# createdb [-D tablespace|--tablespace=tablespace]
[-E encodage|--encoding=encodage] [--lc-collate=locale]
[--lc-ctype=locale] [-O owner|--owner=owner]
[-T template|--template=template] [nombase] [description]
```

La création de la base `clients` peut donc s'écrire comme ceci :

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

Il est possible avec la commande `createdb` de se connecter à un serveur distant, en utilisant les mêmes options que la commande `psql`. Par exemple, la commande suivante crée une base de données sur un serveur PostgreSQL distant :

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