1

Instance

La notion d'instance est certainement celle à laquelle un utilisateur se voit confronté le plus rapidement quand il débute avec PostgreSQL. Il est donc important de commencer par la clarifier.

1.1. Concept général

L'instance est l'objet de base du système PostgreSQL. Les anglophones parlent plus généralement de *cluster*. Ce n'est pas un terme spécifique au monde des bases de données. La traduction stricte d'un cluster est un *groupe*.

Au niveau de PostgreSQL, une instance (ou un cluster) est un groupe de bases de données. C'est une définition assez simpliste mais tout à fait convenable. Une instance PostgreSQL peut contenir plusieurs bases de données. Mais cette propriété n'est pas la seule définition d'une instance.

En fait, une instance est un répertoire de fichiers. Ces fichiers sont créés à l'origine par un outil appelé `initdb` qui est fourni dans la distribution PostgreSQL. Ils ont différentes finalités : fichiers de configuration, fichiers de données, fichiers temporaires de travail, fichiers de traces, etc.

1.2. Au niveau du système d'exploitation

Lorsque le serveur PostgreSQL est démarré, l'instance désigne plutôt un ensemble d'objets au niveau du système d'exploitation :

- un utilisateur ;
- des processus ;
- la mémoire consommée par ces processus ;
- les ports de communication ouverts par ces processus ;
- et le répertoire de fichiers.

Un utilisateur système est utilisé pour exécuter le serveur PostgreSQL. Il n'est pas forcément dédié à l'instance mais, pour des raisons de sécurité, c'est quelquefois le cas.

PostgreSQL utilise plusieurs processus, chacun s'occupant d'une activité particulière. Par exemple, il existe deux processus pour les écritures en tâche de fond des fichiers de données, un processus pour l'envoi des données de réplication sur un serveur secondaire, etc. Ces différents processus sont chargés de travailler sur le même répertoire de fichiers, ce qui fait qu'ils appartiennent à la même instance.

Note *› Voir le chapitre Architecture des processus pour une explication complète sur les processus d'un serveur PostgreSQL.*

Chacun de ces processus va consommer de la mémoire. Une partie de cette mémoire est partagée entre les différents processus et une autre partie est privée à chaque processus spécifique. Par contre, il n'y a pas de mémoire partagée entre deux instances. Chaque instance est très fortement isolée des autres.

Note *› Voir le chapitre Architecture mémoire pour une explication complète sur l'utilisation de la mémoire par un serveur PostgreSQL.*

PostgreSQL est un système client/serveur. Un client a besoin d'extraire et de stocker des données. Le serveur se charge de ce travail d'extraction et de stockage pour que le client puisse se focaliser sur son travail spécifique. Le client se connecte au serveur, les deux discutent suivant un protocole spécifique à PostgreSQL. Le client envoie des ordres écrits dans le langage standardisé SQL. Le serveur exécute chaque ordre du client, renvoyant à ce dernier un statut et, assez fréquemment, des données. Pour que la connexion se fasse, le serveur écoute sur un port des interfaces réseau de la machine. Comme deux processus ne peuvent pas écouter sur le même port, chaque instance dispose de son propre port d'écoute.

Note *› Voir le chapitre Protocole de communication pour une explication complète sur le protocole de communication client/serveur d'un serveur PostgreSQL et le chapitre Gestion des connexions pour la gestion des connexions.*

Le répertoire de fichiers est spécifique à une instance. Ce répertoire contient les fichiers de données des différentes bases, les segments des journaux de transactions, les fichiers de configuration et tout un ensemble de fichiers de métadonnées. Tous ces fichiers sont nécessaires au bon fonctionnement de PostgreSQL.

Note *› Voir le chapitre Fichiers pour une explication complète sur les répertoires et fichiers utilisés par un serveur PostgreSQL.*

De ce fait, une instance PostgreSQL est un ensemble processus/mémoire/port d'écoute/répertoire de fichiers.

1.3. Au niveau du système de base de données

Au niveau de la base de données, une instance est composée d'objets globaux :

- des bases de données ;
- des rôles ;
- des tablespaces.

Les bases de données contiennent différents types d'objets. Certains de ces objets, comme les tables, les index ou les vues matérialisées, peuvent contenir des données. Tous les objets sont accessibles, suivant les droits disponibles, uniquement lorsqu'un utilisateur est connecté à la base de données où ils ont été créés. Chaque base est un objet hermétique dans le sens où il n'est pas possible d'accéder aux objets d'une autre base que celle à laquelle l'utilisateur est connecté.

Les rôles permettent la gestion des droits, que ce soit au niveau de la connexion ou au niveau des objets et des données. Les rôles sont soit des utilisateurs soit des groupes, suivant leur définition.

Les tablespaces permettent de disposer d'autres répertoires de données que le répertoire principal.

Note › *Voir le chapitre Gestion des objets pour une explication complète sur les objets en base proposés par un serveur PostgreSQL.*

Certains objets d'une base de données ont besoin d'être traités par des opérations de maintenance.

Note › *Voir le chapitre Maintenance pour une explication complète sur les opérations de maintenance disponibles avec un serveur PostgreSQL.*

Les données de ces objets sont soumises à des mesures de sécurité :

- des droits sont positionnables pour garantir l'accès aux données uniquement aux personnes autorisées ;
- une sauvegarde permet de s'assurer que les données sont disponibles sur plusieurs serveurs pour qu'en cas de perte du serveur principal, il soit possible de restaurer les données.

PostgreSQL dispose de nombreux types de droits applicables sur les différents objets qu'une instance peut contenir. Ces droits, définis par la norme SQL, sont positionnables par rôle. PostgreSQL dispose aussi de la notion de labels de sécurité. Ces labels sont vérifiés par des mécanismes externes de gestion de droits.

Note *› Voir le chapitre Sécurité pour une explication complète sur la gestion de la sécurité avec PostgreSQL.*

PostgreSQL propose plusieurs solutions de sauvegarde :

- une sauvegarde logique aussi appelée système d'import/export ;
- une sauvegarde physique à froid ;
- une sauvegarde physique à chaud, avec archivage des journaux de transactions.

Note *› Voir le chapitre Sauvegarde et restauration pour une explication complète sur les différents types de sauvegarde avec PostgreSQL.*

1.4. Groupes d'instances

On peut voir les groupes d'instances de deux façons : plusieurs instances liées par le fait qu'elles se trouvent sur la même machine et plusieurs instances liées par le fait qu'elles se trouvent dans le même ensemble de réplication (donc généralement réparties sur plusieurs machines). Les anglophones parlent là aussi de cluster, mais il s'agit dans ce cas d'un *cluster de réplication*.

En effet, il est tout à fait possible d'avoir plusieurs instances PostgreSQL sur la même machine, physique ou virtuelle. Il existe plusieurs bonnes raisons pour le faire et autant pour ne pas le faire. Mais techniquement, c'est possible.

Chaque instance sur la même machine dispose de son répertoire de données, de son port d'écoute, de ses processus. Cependant, si les processus sont différents, les exécutables peuvent être les mêmes. Ils peuvent aussi être différents, notamment dans le but d'avoir des instances provenant de versions différentes de PostgreSQL.

Quand plusieurs instances sont installées sur la même machine, l'utilisateur système qui lance les instances peut être le même. Il peut aussi être différent.

Quand les instances sont réparties sur plusieurs machines différentes tout en faisant partie d'un même groupe, il s'agit généralement d'un cluster de réplication. PostgreSQL propose un système de réplication physique asymétrique, asynchrone ou synchrone, et un système de réplication logique. La mise en cascade des serveurs est possible. De même, il est possible de se connecter aux serveurs secondaires, s'ils autorisent la connexion. Dans ce cas, ils sont accessibles en lecture seule.

Note *› Voir le chapitre Réplication pour une explication complète sur la réplication avec PostgreSQL.*