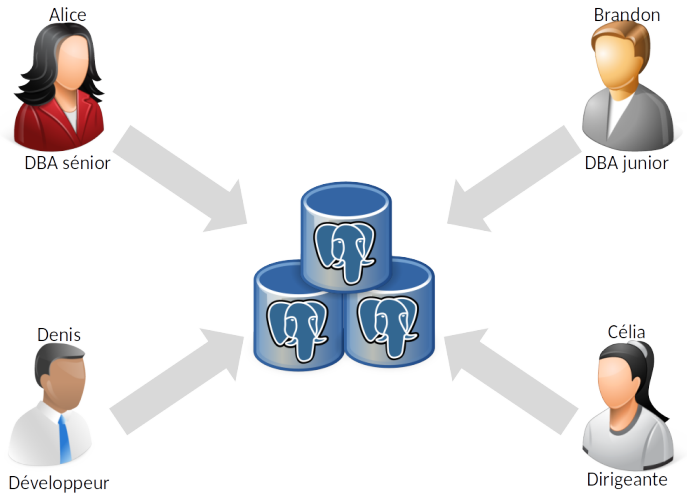
M2 TIW
Administration des Systèmes et des Bases de Données
TIW3-ASBD

Sujets des TP pour l'administration des BD
`https://perso.liris.cnrs.fr/romuald.thion/files/Enseignement/TIW3/`
Version 4.3.0



Travaux Pratiques

TP1 (deux séances) - Configuration et déploiement de bases PostgreSQL *via* Ansible
TP2 - Restauration des Bases de Données
TP3 - Réplication pour haute disponibilité
TP4 (deux séances) - Optimisation de requêtes

L. Avrot – N. Lumineau – R. Thion

Préambule

Objectifs

La réalisation de ces TP d'administration des Bases de Données vous permettra :

1. d'acquérir les compétences techniques vous permettant d'être opérationnel·le pour installer et configurer une base de données de type PostgreSQL,
2. de construire un script de configuration automatique de bases PostgreSQL *via* Ansible,
3. d'acquérir les compétences techniques pour optimiser l'exécution de requêtes interrogeant des tables de grandes tailles,
4. d'acquérir les compétences techniques pour intervenir lors de pertes de données en restaurant une base PostgreSQL,
5. d'acquérir les compétences techniques pour mettre en place une stratégie de réplication des données pour en améliorer la disponibilité.

Informations pratiques

Organisation

Pour réaliser ce TP, vous pouvez travailler seul·e ou en binôme (les trinômes sont interdits). Les séances de TP sont les suivantes

— Lundi 28/10 14h 17h15 : TP1
— Mercredi 30/10 14h 17h15 : TP1
— Lundi 18/11 14h 17h15 : TP2
— Lundi 25/11 14h 17h15 : TP3
— Lundi 16/12 14h 17h15 : TP4
— Mardi 17/12 14h 17h15 : TP4

Modalités d'évaluation

Pour chaque partie du TP, vous devrez rendre un rapport dit d'intervention qui synthétise, en au plus 4 pages, les actions réalisées et les résultats pertinents de la partie. Les scripts SQL, les résultats complets et autres informations techniques doivent être fournies en annexe. Chaque rapport précisera de façon succincte le contexte et les éléments techniques du TP (IP, dates, personnes etc.). Les objectifs principaux du rapport sont de garder trace de l'activité, d'en rendre compte et de permettre de la reproduire, pour cela, il faut donc s'assurer que les annexes sont bien *exhaustives et réutilisables*.

Les rapports sont à rendre sur TOMUSS dans les cases idoines aux dates suivantes :

— TP1 : dimanche 3 novembre 23h59
— TP2 : dimanche 1er décembre 23h59
— TP3 : dimanche 1er décembre 23h59
— TP4 : dimanche 22 décembre 23h59

A propos de la connexion à la VM.

Le TP se déroulera sur les machines virtuelles (VM) hébergées dans le *cloud* du département informatique. Les certificats *.pem* pour établir la connexion SSH ont été transmis par mail. Chaque binôme a deux VM, les adresses IP des VM qui vous sont attribuées se trouvent sur TOMUSS.

A propos de la configuration de la VM.

Une fois connecté·e, il est nécessaire de modifier :

— le fichier `/etc/hosts` pour associer votre IP au nom de votre VM.

Informations pratiques

Documentation de référence pour PostgreSQL :
`https://www.postgresql.org/docs/current/`
Téléchargement de PostgreSQL :
`https://www.postgresql.org/download/linux/ubuntu/`
Pour la mise en place d'ansible :
`https://blog.2ndquadrant.com/ansible-loves-postgresql/`
Documentation pédagogique PostgreSQL UCBL :
`https://forge.univ-lyon1.fr/bd-pedago/bd-pedago/`
Modules ansible :
`https://docs.ansible.com/ansible/latest/modules/modules_by_category.html`
Formations Dalibo (pdf en bas de page de chaque module) :
`https://www.dalibo.com/en/formations`

TP1 - Configuration et déploiement de BD

L'objectif de ce TP est de vous familiariser avec l'installation et le déploiement d'instances de bases de données de type PostgreSQL.

1 Configuration manuelle d'une base PostgreSQL

1.1 Scénario

Bob a été embauché comme DBA dans l'entreprise en prévision du départ dans les mois à venir d'Alice qui est l'actuelle BDA. Cette entreprise doit gérer les données d'un projet interne *via* une base PostgreSQL. Pour pouvoir mettre en place la base de données, Charlie vous présente brièvement le projet.

« Pour votre première mission, je vous mets sur le projet P1BP. Nous avons besoin d'une base de données PostgreSQL opérationnelle dans laquelle de très nombreuses insertions seront effectuées par lots très régulièrement. Il faudra limiter le nombre de connexions à 10 clients. Pour finir, nos filiales auront besoin de se connecter à distance à la base. Pour les accès distants, notre DSI nous impose TLS et l'authentification `scram-sha256` uniquement. Il faut aussi supprimer les droits par défaut sur le schéma '`public`'.
Comme vous êtes nouveau dans l'entreprise, je vous invite à vous rapprocher d'Alice qui sera votre référent technique. »

« Très bien, je me rapproche tout de suite d'Alice pour en savoir un peu plus sur les standards de l'entreprise au niveau des configurations. »

« Pour réaliser ton job, tu devras installer la dernière version de PostgreSQL et je t'invite à chercher la documentation d'installation sur l'OS qui te sera affecté. Il faudra aussi chercher la documentation pour savoir comment *'tuner'* la base. Tu créeras un utilisateur '`betatesteur`' et tu lui associeras une base '`testbd`' dans laquelle tu créeras la table '`tableTest`'. La table sera peuplée avec un jeu d'essai d'au moins 3 tuples, idéalement plusieurs milliers générés avec `generate_series`. Ensuite tu créeras l'utilisateur '`clientWeb`' qui n'aura que les droits de `SELECT`, `INSERT` et `UPDATE` sur les données. Pour gagner beaucoup de temps à la prochaine installation, tu penseras à te faire un mémento sous la forme d'un rapport textuel des commandes et des résultats attendus de toutes tes actions. »

1.2 Travail à réaliser

Il vous est demandé de réaliser l'installation et la configuration de la base PostgreSQL en vous basant sur les recommandations du DBA senior.

Rendu attendu
Votre rapport devra au moins contenir les informations suivantes :

- Les fichiers de configuration (`postgresql.conf` et `pg_hba.conf` notamment) correctement rédigés [1]
- Les commandes SQL et leurs résultats pour gérer les utilisateurs, les bases et les contenus demandés.

1. Par exemple, `https://mydbanotebook.org/post/conf-files/`

2 Configuration automatique de plusieurs instances PostgreSQL

2.1 Scénario

Suite à la réalisation et au succès du projet P1BP, Bob est affecté à un projet de plus grande ampleur, nommé PnBP. La dirigeante vous donne quelques informations.

« Vous allez travailler sur un projet PnBP. Dans ce projet, nous devons mettre en production notre solution logicielle pour une vingtaine de clients. Pour des raisons de confidentialité, tous ces clients souhaitent un hébergement dédié dans notre *datacenter*. Pour chaque base, nous n'avons encore pas beaucoup d'informations. Le nombre d'utilisateurs et la quantité de données à stocker ne sont pas encore clairement définis. Comme toujours, il faut que vous preniez les précautions nécessaires pour réduire les risques de perte de données. Pour pouvoir faire les premiers tests, il me faudrait les 2 premières instances PostgreSQL au plus vite. »

Face au problème, Bob se rapproche du *DBA senior* qui le conseille.

« Pour ce projet, il est nécessaire d'automatiser le processus d'installation, de configuration et de déploiement de tes bases. Face à une telle voilure d'instances, tu devras être vigilant pour garantir la standardisation de tes installations afin de gérer efficacement tes fichiers de configuration en cas d'intervention.
Réutilise ta procédure d'installation manuelle de PostgreSQL pour pouvoir te rendre compte des tâches à automatiser.
Dans l'entreprise, notre outil d'automatisation est ansible. Il va falloir que tu te réalises un script d'installation automatique de PostgreSQL *via* ansible. »

2.2 Travail à réaliser

Il vous est demandé de réaliser une procédure d'installation automatisée d'une instance PostgreSQL *via* un master ansible. Vous exploiterez au mieux les recommandations du DBA senior.

Extrait du standard de l'entreprise

- mots de passe par défaut `mdp:postgresql`
- nom du cluster : `main`
- nom de la base de données : `PnBP`
- nom de l'utilisateur applicatif : `PnBP`
- répertoire du cluster : `/var/lib/postgresql/<version>/<cluster>`
- répertoire des fichiers de configuration : `/etc/postgresql/<version>/<cluster>`

Rendu attendu
Votre rapport de TP1 devra au moins contenir, pour cette partie, les informations suivantes :

- le script ansible contenant la procédure d'installation automatisée.

TP2 – Restauration de données

L'objectif de ce TP est de vous mettre en face d'un problème de perte de données nécessitant la restauration d'une base.

1 Mise en place du système de sauvegarde

1.1 Scénario

Suite au travail réalisé par Bob , Alice vient voir Bob .

« Je viens de voir le travail que tu as effectué sur le projet PnBP. C'est très bien ! Par contre, je n'ai rien vue en place pour gérer la restauration des données en cas de défaillance ou d'erreurs utilisateurs. Il faudrait que tu t'en occupe rapidement. »

« Quels sont les outils utilisés pour pouvoir faire les restauration ? »

« Il va falloir que tu installes *pgbackrest*, et que tu créés ta procédure de restauration à partir de la dernière sauvegarde ou à partir d'une action malencontreusement exécutée. »

1.2 Travail à réaliser

Dans cette partie du TP, il vous est demandé :

- d'installer `pgbackrest`
- de peupler votre base `via pg_bench`
- de dénombrer pour chaque table son nombre de tuples
- d'effectuer une sauvegarde
- d'exécuter la requête D1 : **DELETE FROM** pgbench_tellers;
- d'exécuter une nouvelle sauvegarde
- d'exécuter la requête D2 : **DELETE FROM** pgbench_accounts **WHERE** bid = 2;
- d'afficher le le nombre de tuples de la table `pgbench_accounts`
- de restaurer la base dans l'état précédent la modification effectuée par la requête D2
- d'afficher le le nombre de tuples de la table `pgbench_accounts`
- de restaurer la base dans l'état précédent la modification effectuée par la requête D1
- d'afficher le le nombre de tuples de chaque table

Pour réaliser ce travail, vous utiliserez l'instance que vous avez créée lors du TP1 *via Ansible*.

Rendu attendu
Votre rapport de TP devra contenir, pour cette partie, des scripts exécutés et des copies d'écran validant les exécutions.

1.3 Informations pratiques

1.3.1 A propos de pgbackrest

Vous devez installer manuellement la dernière version de *pgbackrest*[2].

1.3.2 A propos de pg_bench

`pg_bench`[3] est un utilitaire fourni avec PostgreSQL pour simuler des transactions et que vous utiliserez pour simuler une charge de production. Il permet de simuler plusieurs connexions de clients. Il faut d'abord initialiser la base avec l'option *-i*, puis lancer l'outil en paramétrant le nombre de clients et la durée de traitement. Exemple : pour le TP, vous pourrez par exemple utiliser en tant qu'utilisateur 'postgres' la commande :

```
pgbench -i -s 60 <nomBase>
```

1.3.3 Extrait du standard de l'entreprise

— dans *pgbackrest*, le nom du cluster est '`demo-`' suivi du nom de la VM (*e.g., 'demo-asbd-01'*).

— dans *pgbackrest*, le 'stanza' a le même nom que le cluster.

2 Restauration depuis une action donnée

2.1 Scénario

David le développeur débarque dans le bureau de Bob complètement paniqué.

« J'ai perdu une 100000 comptes dans la table '`pgbench_accounts`' de la base de données du projet PnBP ! Je ne sais pas qui a fait le `DELETE` et je ne veux pas savoir, par contre, il fait que tu me retrouves quand le requête a été exécutée et que tu me restaures les comptes au plus vite. »

« OK ! Je vais regarder dans les logs au plus vite `via pg_waldump`. »

2.2 Travail à réaliser

Pour cette partie du TP, il vous est demandé :

— d'identifier la date où la requête D2 a été exécutée en utilisant `pg_waldump`

— de restaurer la base pour rétablir les comptes de `pgbench_accounts` manquants

Rendu attendu
Votre rapport de TP3 devra au moins contenir, pour cette partie, les copies d'écran montrant l'état de la base avant et après restauration.

2.3 Informations pratiques

2.3.1 A propos de pg_waldump

L'utilitaire *pg_waldump*[4] permet de fournir une vue lisible des logs. Pour pouvoir identifier des actions telles qu'une suppression survenue dans une table, il est nécessaire :

— d'identifier les identifiants du tablespace, de la base et de la table. Ce type d'information peut s'obtenir *via* une requête SQL comme ci-après

2. Voir `http://pgbackrest.org/user-guide.html`
3. Voir `https://www.postgresql.org/docs/current/pgbench.html`
4. Voir `https://www.postgresql.org/docs/current/static/pgwaldump.html`

— d'isoler dans un fichier les instructions **DELETE** via un pipeline entre le résultat du '`pg_waldump`', des grep sur le mot clé **DELETE** et sur les identifiants de la table ('`rel oidTablespace/oidBase/oidTable`').

```
SELECT
  COALESCE(tbs.oid, db.dattablespace) AS tablespace,
  db.oid AS database,
  t.relfilenode AS table
FROM pg_class t LEFT OUTER JOIN pg_tablespace tbs
  ON t.reltablespace=tbs.oid
  CROSS JOIN pg_database db
WHERE t.relname='<nom-de-ma-table>'
  AND db.datname=current_database();
```

2.3.2 Extrait du standard de l'entreprise

— Pour pouvoir travailler sur les log, il est recommandé de les stocker dans une répertoire à part '`/tmp/WAL`' dans lequel les logs pourront être dézipés, renommés si nécessaire, manipulés.

3 Épilogue

Suite aux restaurations réussies, Alice vient féliciter Bob .

« Félicitations ! Tu as eu les bons réflexes pour la gestion de crise sur le projet PadBol. David a eu de la chance de ne pas avoir fait cette boulette sur un projet reposant sur une base ORACLE. »

« Pourquoi ? Il y a bien RMAN pour la restauration de données sur ORACLE. »

« Effectivement ! Par contre, si tu envisages d'utiliser certaines fonctionnalités comme le *flashback* de RMAN par exemple, il faudra vérifier nos contrats pour s'assurer que lesdites fonctionnalités sont comprises dans ce qui a été payé. Ça fait partie de notre boulot de DBA. »

TP3 – Réplication physique : performance et haute disponibilité

L'objectif de ce TP consiste à mettre en place une stratégie de réplication physique des objets d'une base de données PostgreSQL pour en améliorer la disponibilité.

4 Réplication physique

4.1 Scénario

Charlie et Alice débarquent dans le bureau de Bob .

Charlie

Dirigeante

Alice

DBA sénior

« Je viens d'avoir un de nos clients du projet 'PnBP'. Malgré l'optimisation, il continue à observer des latences peu satisfaisante. Alice vient de m'expliquer que le problème viendrait sûrement du trop grand nombre d'utilisateurs interrogeant la base. »

« Il me semble nécessaire de répartir la charge sur au moins un autre serveur en mettant en place une stratégie de réplication. La base en cours de production serait la copie primaire et il faudrait installer une copie secondaire. »

« OK ! je pars sur de la réplication physique de la base. »

Bob

DBA junior

4.2 Travail à réaliser

Il vous est demander de faire :

1. la mise en place des copies secondaires de la base de données que vous avez peuplée durant le TP précédent.
2. le paramétrage pour effectuer une gestion synchrone de la réplication.
3. l'insertion de tuples dans la copie primaire et visualiser les mises à jour dans les copies secondaires.

Rendu attendu
Votre rapport de TP, devra au moins contenir les copies d'écran montrant les commandes exécutées et l'état des tables observées.

4.3 Informations pratiques

Chapter 26. High Availability, Load Balancing, and Replication :
`https://www.postgresql.org/docs/current/high-availability.html`

How to manage Replication and Failover in Postgres Version 12 without recovery.conf file :
`https://www.enterprisedb.com/postgres-tutorials/how-manage-replication-and-failover-postgres-ver`

TP4 – Optimisation de requêtes

L'objectif de ce TP est d'explorer les possibilités d'optimisation de requêtes sur PostgreSQL. Les schémas et données sont accessibles au format CSV sur le serveur `https://perso.liris.cnrs.fr/romuald.thion/files/Enseignement/TIW3/`. Après avoir créées les tables correspondantes avec les fichiers `Object-pg.sql` et `Source-pg.sql`, vous pourrez importer les données avec des commandes similaires à la suivante :

```
\COPY object FROM PROGRAM
'wget -O -  https://perso.liris.cnrs.fr/[...]/Object-001.gz  | gzip -dc'
with (format csv, null 'NULL');
```

1 Pratique du SQL moderne

TBD

2 Impact de l'indexation sur les requêtes en lecture

2.1 Scénario

Charlie et David débarquent dans le bureau de Bob .

« Je viens d'avoir le client du projet 'AstroD'. Il observe des latences sur le traitement de certaines requêtes. Est-ce qu'il y aurait moyen d'améliorer les choses ? stp. »

« Il y a sûrement moyen de faire de l'optimisation. Quelles sont les requêtes qui posent problème ? »

« La base contient deux relations *objects* et *sources*.
Parmi les requêtes qui posent un problème, il y a :
La requête R1 qui est : "Pour chaque objet de la relation *objects*, donner le nombre de sources de la relation *sources* correspondant à cet objet".
La requête R2 qui est : "Donner les paires de sources qui ont la même valeur de RA arrondi à 0.1 près".
La requête R3 qui est : "Donner les objets qui ne sont pas associés à une source". »

« OK ! Je vous fais un audit de la situation actuelle pour le traitement de ces trois requêtes. Ensuite, je créerai quelques index et je vous ferai un rapport pour faire le point. »

2.2 Travail à réaliser

Il vous est demander de faire :

1. l'étude des plans d'exécution des requêtes R1, R2 et R3 et estimation des coûts associés. (Utiliser la commande `EXPLAIN ANALYZE`.)
2. créer les index qui vous semblent les plus appropriés pour chacune des requêtes. Lebut est de ne pas en créer trop.
3. l'étude des plans d'exécution avec index des requêtes R1, R2 et R3 et estimation des coûts associés.

Pour de la documentation sur les index, le site de Markus Winand vous est recommendé : `https://use-the-index-luke.com/`.

Rendu attendu
Votre rapport de TP, pour cette partie, devra au moins contenir les différents plans d'exécution et vos commentaires sur les observations.

3 Impact de l'indexation sur les requêtes en écriture

3.1 Scénario

Alice qui a pris connaissance de l'audit sur le projet AstroD vient voir Bob

"C'est du bon travail ! Bravo. Par contre, je suis surprise qu'il n'y ait pas un mot sur la taille des index sur le disque ni sur l'impact de tes index sur les requêtes en écriture."

"Gloups ! Euh... effectivement il faudrait regarder ces aspects."

3.2 Travail à réaliser

À la différence d'ORACLE, notez que pour POSTGRESQL toutes les créations et suppressions d'objets sont transactionnelles. Vous pouvez donc annuler une suppresion d'index (par exemple) avec un rollback. Servez-vous en pour comparer les plans d'exécution avec et sans index.

Il vous est demander de :

1. déterminer la taille de chaque index qui a été créé dans la section précédente.
2. écrire une requête SQL de mise à jour E1 qui applique la fonction $f : x \rightarrow 2x^2$ à l'attribut RA de la table *objects*.
3. évaluer le temps d'exécution de la requête E1.
4. évaluer à nouveau le temps d'exécution de la requête E1, après avoir supprimé vos index.
5. comparer les résultats obtenus aux deux étapes précédentes.
6. définir vos conclusions d'audit.

Rendu attendu
Votre rapport de TP, pour cette partie, devra présenter votre audit.

3.3 Informations pratiques

— L'extension HypoPG `https://hypopg.readthedocs.io/en/latest/`
— Différentes méthodes pour un *anti join* `https://gist.github.com/fengb/058e2378e4931b69b869`