# TP 5 – Evaluation de requêtes et optimisation

Pour cette partie du cours, nous travaillerons sous postgreSQL. Si vous travaillez sur votre compte étudiant, référez-vous aux consignes sur le elearning pour vous connecter à votre base. Si vous travaillez sur votre ordinateur personnel, débrouillez vous pour avoir une installation fonctionelle **avant** les TPs.

## 1 Importation et index

1. Nous allons travailler sur les données utilisées en L3 dans le TP sur les index. Récupérez les fichiers `clients.sql` et `clients_data.csv`. Le script `clients.sql` vous permet de créer et remplir une table `client`. Explorez un peu le schéma ainsi que le contenu de la base. Vous aurez besoin d'analyser les valeurs présentes dans la table pour expliquer les comportements observés (nombre de valeurs distinctes, valeurs répétées, valeurs uniques, nulls etc). Dans un premier temps, vous devrez calculer les informations dont vous avez besoin "à la main" avec des requêtes. (La semaine prochaine, on étudiera les contenus la table `pg_statistic`.)

2. Lors de la création de la table, vous n'avez pas créé d'index, pourtant il en existe un. Vérifiez et expliquez.

3. Lancez la commande ANALYZE afin de mettre à jour les statistiques.

4. Testez la commande `EXPLAIN` sur la requête toute simple qui énumère le contenu d'une table. Documentez vous dans la doc sur la signification des informations `cost, rows, width` renvoyées par la commande `EXPLAIN`.

Remarques: Dans ce TP, nous ne nous focaliserons **pas** sur l'option `ANALYZE` de `EXPLAIN`. Gardez bien en tête le fait que (1) les données d'`EXPLAIN` sont des estimations (2) les coûts sont calculés à partir de valeurs de références dont l'échelle est arbitraire.

5. Créez maintenant des scripts en prenant exemple sur `clients.sql` permettant chacun de créer des nouvelles tables ayant le même schéma et les mêmes données que la table `client`. L'une s'appellera `client_btree` aura en plus deux index `btree`: un sur la colonne `age` et un sur `tel`. Un autre s'appellera `client_hash` aura en plus deux index `hash`: un sur la colonne `age` et un sur `tel`. Vous pouvez aussi faire une table `client_both` avec les index `btree` et `hash`.

Les créations d'index devront être faites avant de remplir la table.

6. Relancez ANALYZE après la création de vos tables afin de mettre à jour les statistiques.

## 2 Opérateurs de sélection

L'objectif de cette partie est d'observer, sur des requêtes simples **sans jointure**, les choix faits par le système au niveau des plans d'exécution. Vous expliquerez l'utilisation ou non des différents

index selon le type de requêtes et mettrez en évidence les différents opérateurs de sélection de postgreSQL.

Vous trouverez ci-dessous des suggestions de requêtes. Mais n'hésitez pas à pousser les tests plus loin et/ou à utiliser d'autres requêtes. L'important est de faire apparitre des comportements variés et intéressants à expliquer.

*Les résultats variant selon les instances de bases et les stats construites par le système, il est normal que vous n'obteniez pas forcément tous le même résultat à une même requête.*

7. Choisir une valeur x non nulle de `age` qui apparait plus de 1000 fois dans la table. Observer le query plan de la requête qui extrait les tuples ayant x pour valeur de `age`.
8. Choisir une valeur x non nulle de `tel` qui apparait plus d'une fois dans la table. Observer le query plan de la requête qui extrait les tuples ayant x pour valeur de `tel`.
9. Extraire les tuples dont `age` appartient à un intervale de valeurs. Essayer d'obtenir plusieurs plans différents.
10. Extraire les tuples ayant un null pour `age`.
11. Extraire les tuples ayant une valeur de `age` non null.
12. Faire une sélection sur la disjonction d'une valeur de `age` et d'une valeur de `tel`.
13. Sur la table `client_btree`, trouvez une requête mettant en avant l'opérateur `BitmapAnd`.

# 3 Jointures

Nous allons maintenant nous intérsser aux opérateurs de jointure. Dans chacun des cas suivants, proposez une requête calculant le résultat et analysez le plan d'exécution proposé par le query planner. N'hésitez pas à observer le comportement d'autres requêtes (sur la base magasin par exemple).

N'hésitez pas à dessiner l'arbre correspondant au plan d'exécution lorsqu'il devient compliqué.

14. La jointure de la table client avec elle même sur l'attribut `age`. Est-ce que le plan d'exécution change avec l'index utilisé ?
15. La jointure de la table client avec elle même sur l'attribut `age`, triée par l'âge de c1. Est-ce que le plan d'exécution change avec l'index utilisé ?
16. La jointure de la table client avec elle même avec la condition `c1.age < c2.age`. Est-ce que le plan d'exécution change avec l'index utilisé ?

# 4 Autres opérateurs

L'objectif de cette partie est d'observer d'autres opérateurs tels que les opérateurs ensemblistes, d'agrégats, etc. Proposez des requêtes mettant en évidence l'utilisation des différents opérateurs.

Vous pouvez écrire des requêtes sur la base client ou bien étudiez les plans d'exécution résultant de vos réponses au TP d'échauffement sur elearning.