# Bases de Données - TP 8
# EIDD

---

**Requêtes récursives en SQL**

---

## 1 Jeu de données

Nous travaillerons avec les données fournies par le Syndicat des Transports en Île-de-France (STIF), au format GTFS, que vous avez modélisées et chargées lors du TP3. Si ce n'est pas déjà fait, suivez les étapes indiquées dans la Section 2 du TP3 pour charger ces données (et relisez cette section pour vous rappeler la structure de ces données).

**IMPORTANT** : Dans la recherche de trajets, on considérera possible un changement de transport en commun entre deux arrêts quelconques qui se trouvent dans la même zone (c'est-à-dire deux arrêts qui ont la même station parent).

## 2 Requêtes sur la base

Pour chacune des questions suivantes, proposer une requête SQL qui renvoie les informations demandées ; ne pas hésiter à construire les requêtes petit à petit pour vérifier qu'elles renvoient des informations cohérentes.

Il peut être utile (voire nécessaire) d'ajouter des indexes sur les tables pour rendre l'exécution des requêtes plus efficace.

Pour vous aider, il est indiqué après chaque question le nombre de résultats que cette requête est censée renvoyer.

Pour désactiver la pagination du résultat dans le client en ligne de commande de PostgreSQL utiliser la commande \pset pager off au début de votre session psql. Vous pouvez utiliser la commande \timing afin de comparer les temps d'exécution avec et sans index. N'hésitez pas non plus à préfixer vos requêtes avec le mot clef EXPLAIN pour voir le plan d'exécution de votre requête.

1. Sélectionner l'ensemble des champs de la table stop pour :
   - les stops avec location_type=0, limiter la requête à 20 lignes (donc sélectionner les vraies stations)

     (20 résultats)

   - les stops avec location_type=0 avec un champ parent_station = NULL (attention en SQL on compare avec NULL par la condition « is NULL »)

     (0 résultats)

   - les stops avec location_type=1 avec un champ parent_station non NULL

     (0 résultats)

   - les stops avec location_type=1 avec un champ parent_station = NULL, limiter la requête à 20 lignes (donc sélectionner les zones d'arrêts)

     (20 résultats)

On en déduit qu'il y a deux façons alternatives de reconnaître les zones d'arrêts :

- Les arrêts qui sont parent d'un autre arrêt et n'ont pas de parent ;

- Les arrêts qui ont location_type = 1 ;

2. Quelles sont les zones d'arrêts présents dans la table stop_times ?
(0 résultats)

3. Combien y-a-t-il de stops des vrais arrêts (c'est-à-dire des arrêts qui ne représentent pas des zones d'arrêts) ?
(36517 résultats)

4. Combien y-a-t-il de stops de zones d'arrêts ?
(15228 résultats)

5. Quels sont les arrêts de transport en commun qui se trouvent dans la zone de "Sucy - Bonneuil" ? Pour chaque arrêt, renvoyer son identifiant et son nom. Les résultats doivent être triés par nom.
(12 résultats)

6. Quels sont les lignes de transport en commun (c.-à-d. les routes) qui ont un arrêt dans la zone de "Sucy - Bonneuil" ? Pour chaque ligne, renvoyer le nom long de la ligne et son type (route_type). Les résultats doivent être triés.
(10 résultats)

7. Quels sont les arrêts accessibles en prenant un transport en commun dans la zone de "Sucy - Bonneuil", sans changement ? Renvoyer à la fois le nom de l'arrêt et le nom long de la ligne de transport permettant d'y accéder. Les résultats doivent être triés par nom de ligne, puis par nom d'arrêt.
(107 résultats)

8. Refaire la requête précédente, en sélectionnant cette fois ci uniquement les arrêts accessibles en train. Se rappeler que le type de transport en commun est décrit par l'attribut route_type de la table routes.
(1 résultat, la base est incomplète)

9. Créer une vue intitulée **stop_sequence** qui sélectionne sans doublons les stops_id de départ (d_stop_id) et les stop_id d'arrivée (a_stop_id) entre 2 arrêts dont la séquence est séparée d'un seul trajet (d_stop_sequence = a_stop_sequence – 1) pour un même trajet.

10. Compter le nombre de lignes de cette vue. Afficher les 20 premières lignes
(301 résultats)

11. Créer une requête basée sur la vue précédente qui liste tous les couples arrêt de départ, arrêt d'arrivée (d_stop_id, a_stop_id) quel que soit le nombre de changement et d'arrêts parcourus. On se basera sur une requête récursive WITH RECURSIVE access(d_stop_id, a_stop_id) AS … , limiter la requête à 20 lignes. (Voir https://www.sqlservertutorial.net/sql-server-basics/sql-server-recursive-cte/)

12. Compter le nombre de lignes de cette requête
(7765 résultats)

13. Utiliser la requête précédente mais afficher cette fois le nom des arrêts de départ et d'arrivée en les ordonnant par nom des arrêts de départ puis d'arrivée. Limiter la requête à 20 lignes.

14. Modifier la requête précédente pour afficher toutes les stations accessibles depuis la zone de "Sucy - Bonneuil".
(162 résultats)

15. Afficher le plan d'exécution de la requête précédente (explain ANALYZE).

16. Créer une vue **matérialisée** intitulée m_stop_sequence basée sur la même requête que la vue stop_sequence.

17. Réutiliser la requête récursive sur la vue matérialisée et afficher son plan d’exécution.

18. Créer deux index sur la vue matérialisée m_stop_sequence, un pour le champ d_stop_id et un pour le champ a_stop_id.

19. Ré-exécuter la requête récursive et constater la différence de temps d’exécution.