ESGI - 2024/2025 - Requêtage SQL - Projet

Voici quelques conseils à garder à l'esprit dans votre quête de la base de données parfaite :

- Certaines requêtes sont plus difficiles que les autres, n'hésitez pas à les faire plus tard et à passer à une autre question.
- Prenez le temps de lire chaque question afin de vous assurer que votre code répond au besoin. Pour les requêtes SQL et les vues, vous devez respecter les noms de colonnes donnés en exemple, sinon tous les tests échoueront.

Introduction

Vous devrez créer puis manipuler une base de données. Le but est de gérer un système de transport public (type RATP). Pour se concentrer sur la manipulation de la base de données, vous serez amenés à faire plusieurs choses.

Vous aurez à votre disposition un fichier `data.sql` qui vous aidera à remplir votre base avec des données cohérentes.

Scripts pour la base de données

Vous devrez fournir un script SQL pour créer les tables de votre base de données.

Ce fichier doit être nommé `init_database.sql`. Vous devrez aussi fournir un fichier SQL par niveau avec vos requêtes, vues, etc. à l'intérieur.

Chaque fichier doit être nommé `level_X.sql` (X étant le numéro du niveau).

Tous ces fichiers doivent se trouver dans un répertoire src à la racine de votre dossier de rendu.

Résumé

- 1 fichier SQL pour créer vos tables.
- 1 fichier SQL par niveau avec votre travail à l'intérieur.

Veillez à ces points importants sur lesquels l'évaluation sera basée :

- Évidemment, validité des résultats des procédures stockées.
- Performance des procédures (induisant la qualité de la modélisation).
- Qualité du code fourni.

Objectif

L'objectif de la base de données et de toutes les fonctionnalités associées est de gérer un système de transport public.

Les fonctionnalités attendues sont assez simples à comprendre : il y a tout d'abord la gestion des **lignes**, des **stations**, des **arrets** et des **horaires**, c'est-à-dire tout le réseau physique.

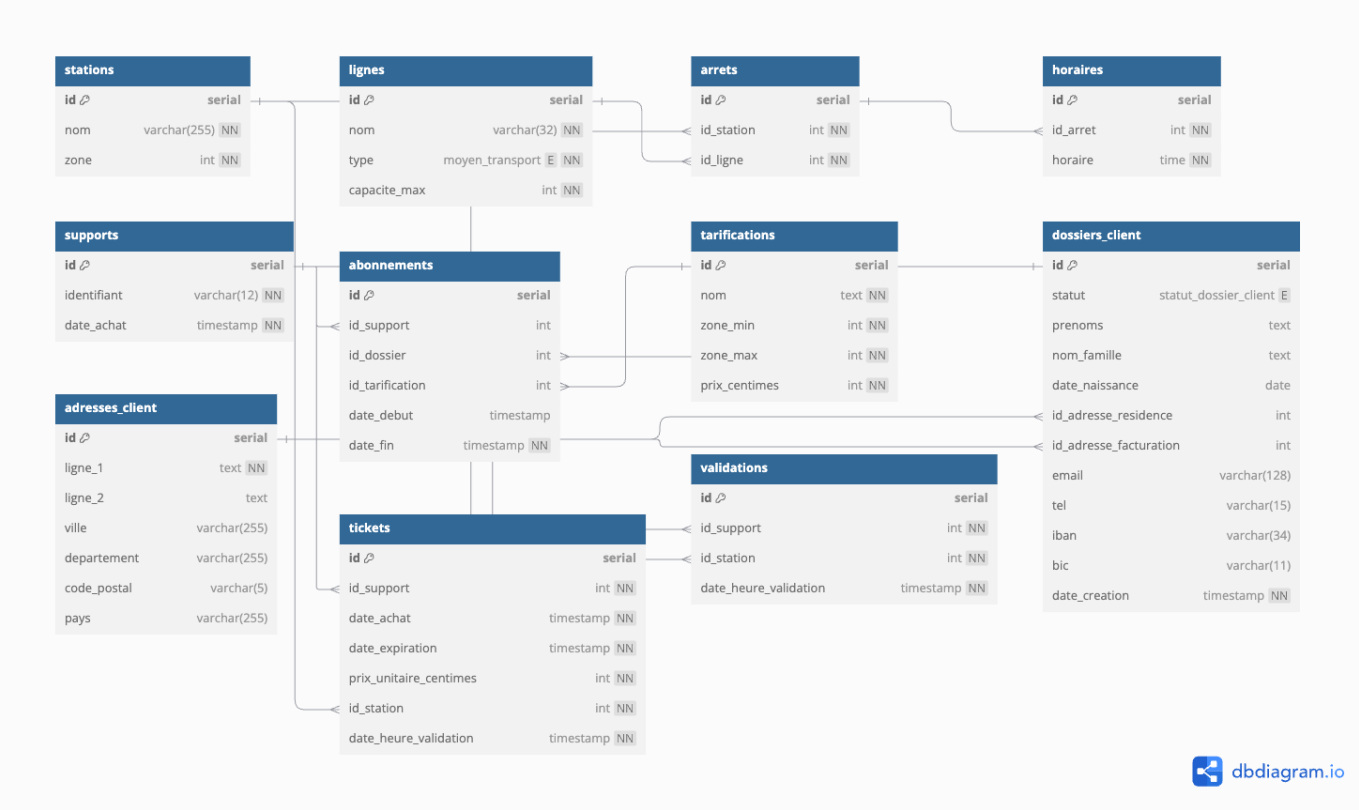
Ensuite, vous devrez gérer tous les **supports** en circulation sur le réseau, leurs **abonnements** ou leurs **tickets** et leurs **déplacements** matérialisés par des **validations**.

Les **supports** sont liés à des **dossiers clients**.

Spécifications fonctionnelles

Une entreprise régissant un réseau de transport souhaite moderniser son système d'information.

Vous devez représenter le schema de base de données ci-dessous dans votre base en respectant les contraintes fournies :



> E signifie Enum, les valeurs attendues sont les suivantes :
>
> - `moyen_transport` : `metro`, `rer`
> - `statut_dossier_client` : `incomplet`, `validation`, `validé`, `rejeté`

Niveau 1

nombre de dossiers incomplets

Ecrivez une requête qui renvoie le nombre de dossiers incomplets dans un colonne nommée `nb_dossiers_incomplets`.

stations desservies par chaque ligne

Ecrivez une requête qui renvoie les lignes suivies des stations qu'elles desservent dans l'ordre alphabatique dans 2 colonnes `ligne` et `stations`.

nombre de stations par moyen de transport

Ecrivez une requête qui retourne le nombre de stations par moyen de transport de celle qui en a le plus, à celle qui en a le moins, dans 2 colonnes `moyen_transport`, `nb_stations`.

abonnements qui expirent à la fin de janvier 2025

Ecrivez une requête qui liste le nombre d'abonnements par tarification qui expirent à la fin du mois de janvier 2025, on souhaite récupérer ces informations dans 2 colonnes `nom_tarification`, `nb_abonnements` de la tarification ayant le moins de d'abonnements qui expirent, à celle qui en a le plus.

dossiers en validation

Ecrivez une vue SQL `dossiers_en_validation` qui renvoie toutes les informations des dossiers en validation du plus ancien au plus récent.

Niveau 2

stations doubles

Listez dans une colonne `station` les stations desservies par au moins un metro **et** un RER. Triez par ordre alphabétique.

forfaits populaires

Affichez dans des colonnes `nom_forfait`, `nb_abonnements` les 3 forfaits les plus populaires, du plus populaire au moins populaire. La popularité d'un forfait se base sur le nombre d'abonnements actifs pour ce même forfait.

capacité moyenne de chaque station

Dans une requête SQL, affichez la capacité moyenne de chaque station listée dans l'ordre alphabétique dans 2 colonnes `station` et `capacite_moy` (formule : `somme des capacités max des lignes d'une station / nombre de lignes dans la station`)

nombre d'abonnés par département

Ecrivez une vue SQL `abonnes_par_departement` qui affiche pour chaque département et code postal, le nombre d'abonnés y habitant (`departement`, `code_postal`, `nb_abonnes`). Ordonnez les résultats par code postal.

usagers par tranches d'âge

Dans une requête SQL, retournez le nombre total d'usagers par tranche d'âge :

- - 18 ans (`moins_18`)
- 18-25 ans (`18_24`)
- 25-40 ans (`25_40`)
- 40-60 ans (`40-60`)
- + 60 ans (`plus_60`)

Vous devrez retourner une colonne par tranche d'âge.

stations les plus fréquentées

Listez dans une vue `frequentation_stations`, dans 2 colonnes `station` et `frequentation`, les 10 stations les plus fréquentées depuis toujours. La fréquentation se mesure par le nombre de validation. Triez les de la plus fréquentée à la moins fréquentée.

Niveau 3

chiffre d'affaires des ventes de tickets par mois sur l'année 2024

Ecrivez une requête qui retourne le chiffre d'affaires par mois de l'année 2024. Vous devrez retourner dans 2 colonnes `mois` et `chiffre_affaires`, le mois en toutes lettres et le chiffre d'affaires en euros (*sans* le signe €).

lignes de transports à Nation à 17:28:16 +- 4 minutes

Dans une requête SQL, listez par ordre de passage, les lignes de transports qui passent à la station Nation autour de 17 heures 28 minutes et 16 secondes à plus-ou-moins 4 minutes.

nombre moyen de validation par mois par types d'abonnement

Récupérez dans une requête SQL dans des colonnes `moy_validation` et `abonnement`, la moyenne des validations par mois selon les différents abonnements (tarifications). Triez par moyennes décroissantes puis par nom de tarification dans l'ordre alphabétique.

moyenne des passages par jour de la semaine sur les 12 derniers mois

Créez une vue permettant de voir la moyenne de passagers par jour de la semaine (lundi, mardi, ... dimanche) sur les 12 derniers mois. La vue devra avoir 2 colonnes : `moy_passagers`, `jour_semaine`. Triez par jour de la semaine.

taux de remplissage moyen des lignes

Créez une vue permettant d'avoir le taux de remplissage moyen de chaque ligne de transport sur le réseau. Le taux de remplissage étant `nombre moyen de passagers par ligne par jour / nb de train par jour sur la ligne / capacite max ligne`. Considérez pour connaître le nombre de train par jour d'une ligne, qu'il y'a un metro toutes les 6 minutes entre 5:00 du matin et 1:30 du matin (le lendemain) et un RER toutes les 15 minutes de 5:00 à 1:30 aussi. Renvoyez le résultat dans 2 colonnes `taux_remplissage` et `nom_ligne`.

> 💡 Rappel, un taux est un pourcentage.

Niveau 4

parts des passagers ayant un abonnement, contre ceux voyageant avec des tickets (supports)

Renvoyez dans une requête SQL le pourcentage de passagers avec un abonnement contre ceux avec des tickets. Considérez de manière unique les passagers en fonction du support utilisé. Votre résultat sera retourné dans 2 colonnes `part_abonnement`, `part_ticket`.

nombre de nouveaux abonnements par mois en 2024

Grâce à une requête SQL, dans une colonne `nb_nvx_abo`, retournez le nombre d'abonnements créés par mois, sur des supports qui n'en avaient pas avant, en 2024.

montant total économisé par les usagers ayant un abonnement s'ils avaient dû acheter un ticket

Calculez le montant total économisé par les usagers ayant un abonnement comparé au prix qu'ils auraient dû payer en achetant des tickets pour chacuns de leurs voyages. Donnez un montant total en euros dans une colonne `montant_economise_euros`, ce dernier ne peut pas être négatif (il peut être égal à 0).

heure la plus affluante par station

Rédigez une vue permettant de connaître **l'heure** la plus affluante (celle où il y'a le plus de validations) pour chaque station. Affichez 2 colonnes `nom_station`, `heure_affluante`. Triez de la plus affluante à la moins affluante.

nombre d'abonnements actifs par tranche de zone

Rédigez une vue qui retourne dans 3 colonnes `zone_min`, `zone_max`, `nb_abonnements`, le nombre d'abonnements actifs par tranche de zone. Triez par nombre d'abonnements décroissants, puis par zone_min, puis par zone_max.

Bon courage !