Stations et Trains (RER et métro) en Île de France

October 27, 2023

Abstract

Création d'une base de données relationnelle à partir des données en accès libre qui se trouvent ici.
Le but c'est de mettre en évidence les limitations du langage SQL et du modelé relationnel de BD.

1 Requêtes

L'application a comme but de répondre aux questions de ce type:

1. Étant donné une ligne X (e.g. RER B), quelles sont ses stations?
2. Quelles sont toutes les stations au sud (géographiquement) d'une station X donnée?
3. Quelles sont toutes les stations au sud d'une station X donnée et qui sont desservies par le même train?
4. Étant donné deux stations X et Y, y-a-t-il une connexion directe de X à Y (un train qui va de X à Y)? (*question pas triviale, prendre comme exemple les stations Robinson et Massy-Palaiseau*).
5. Étant donné deux stations X et Y de la même ligne, combien de stations les séparent? (*attention: requête imprécise)*
6. Étant donné deux stations X et Y pour lesquelles il existe une connexion directe, quelle est la durée minimale d'un trajet de X à Y?
7. Étant donné deux stations X et Y pour lesquelles il n'existe pas une connexion directe, y-a-t-il une connexion avec un seul changement entre les deux stations?
8. Combien de changements de ligne pour aller de la station X à la station Y?
9. Quel est le nombre maximal de changements pour un trajet entre deux stations? (*pire de cas*)
10. Quelles sont les stations X et Y dont le trajet le plus simple demande un nombre de changements maximal?

Syntaxe SQL recommandée

- Les clauses WITH (également connues sous le nom de CTE = Expression de Table Commune) fonctionnent comme des vues temporaires.
- Elles permettent de décomposer des requêtes plus complexes en blocs simples à utiliser et à réutiliser si nécessaire.
- Elles peuvent améliorer la lisibilité en mettant l'accent sur une interprétation plus procédure d'une requête dans un flux de travail (similaire aux programmes Java, C, python) .
- Particulièrement utile si vous avez besoin de faire référence à une table dérivée plusieurs fois dans une seule requête ou si vous effectuez le même calcul plusieurs fois sur plusieurs composants de la requête (= mémorisation).

2 Schéma

Demande 1: À ce point il faut proposer le schéma d'une nouvelle base de données relationnelle capable de répondre aux questions énoncées précédemment. Il faut donc définir les relations, leurs leurs attributs et leurs clés primaires. La décision doit prendre en compte les données fournies par la région. Elles peuvent être consultées ou télé-chargées ici. Dans le compte rendu, justifier vos choix et analyser les avantages et les désavantages du schéma proposé.

Solution partielle: Schéma proposé en classe

STOPS(**stop_id**, stop_name, stop_lon stop_lat OperatorName Nom_commune Code_insee)
La tables des stations.

ROUTE(**route_id**, route_short_name, route_long_name)
ROUTE = Ligne e.g., RER A, RER B, etc.

TRIPS(route_id, **trip_id**, trip_headsign)
TRIPS = Type de trajet en train e.g. le train MONA sur l'RER C

STOP_TIMES(**trip_id, stop_id, stop_sequence, arrival_time,** departure_time)
stop_sequence $\in [0, n]$ donne l'ordre de la station sur le trajet.

Demande 2: Donner les requêtes SQL pour les questions de la Section 1 ou expliquer pourquoi la question ne peut pas s'exprimer dans le langage SQL.

3 Les demandes

3.1 Demande 1:

Des scripts de création de la base de données.

Solution

```
1
2  drop table Routes;
3  drop table Trips;
4  drop table StopTimes;
5  drop table Stops;
6  drop table DetailedStops;
7
8
9  CREATE TABLE Routes (
10     route_id VARCHAR(14) PRIMARY KEY,
11     route_short_name VARCHAR(14),
12     route_long_name VARCHAR(20)
13 );
14
15
16 CREATE TABLE Trips (
17     trip_id VARCHAR(49) PRIMARY KEY,
18     route_id VARCHAR(14) REFERENCES Routes(route_id),
19     trip_headsign VARCHAR(27)
20 );
21
22
23 CREATE TABLE StopTimes (
```

```
    trip_id VARCHAR(60),
    stop_id VARCHAR(30),
    stop_sequence SMALLINT CHECK (stop_sequence >= 0 AND stop_sequence < 50),
    arrival_time TIME,
    departure_time TIME,
    PRIMARY KEY (trip_id, stop_id)
);

CREATE TABLE Stops (
    stop_id VARCHAR(30) PRIMARY KEY,
    stop_name VARCHAR(52),
    parent_station VARCHAR(30)
);


drop table DetailedStops;
CREATE TABLE DetailedStops (
    route_id VARCHAR(12),
    stop_id VARCHAR(30),
    stop_name VARCHAR(52),
    stop_lon FLOAT8,
    stop_lat FLOAT8,
    OperatorName VARCHAR(5),
    Nom_commune VARCHAR(40),
    Code_insee VARCHAR(5),
    PRIMARY KEY (route_id, stop_id)
);

ALTER TABLE DetailedStops
ADD CONSTRAINT fk_detailedstops_routes
FOREIGN KEY (route_id) REFERENCES Routes(route_id);
```

3.2 Demande 2:

Un programme ou script de transformation des données brutes en vue de leur insertion dans la base de données. Le compte rendu doit préciser (1) les fichiers d'entrée, (2) les commandes pour obtenir les résultats (3) l'emplacement des résultats.

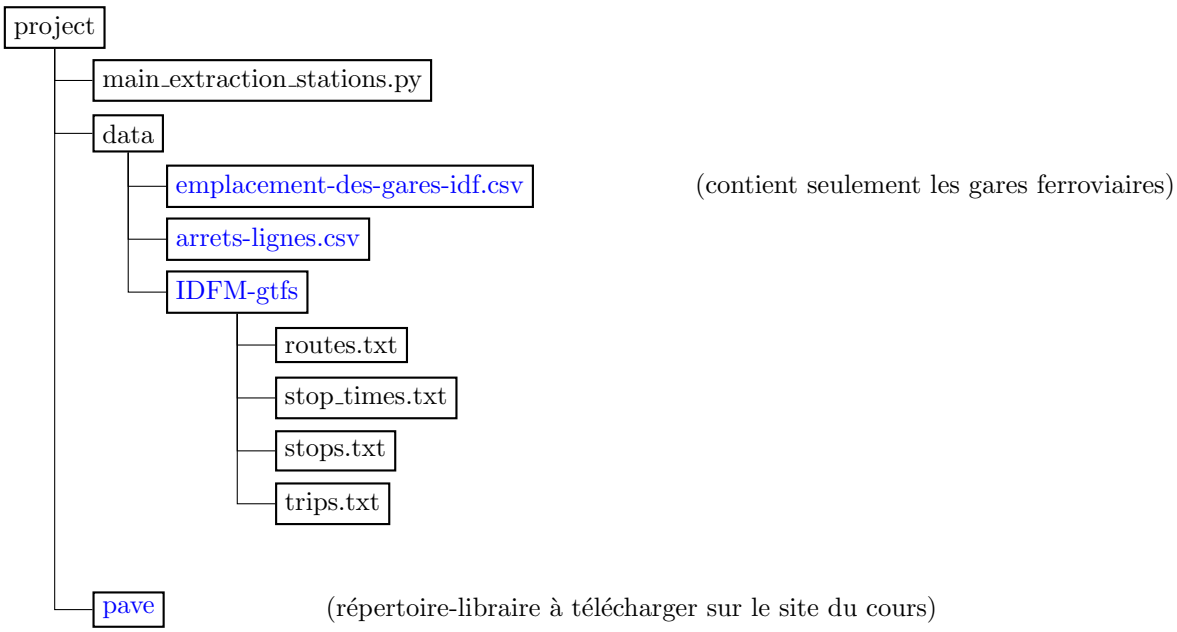

Figure 1: La structure du répertoire de travail.

Solution:

1. **Les données brutes** J'ai télé-chargé deux fichiers et un répertoire fournis pas la région Île de France En suivant les liens vous pouvez les télé-chargér aussi:

`.data/emplacement-des-gares-idf.csv`
`.data/arrets-lignes.csv`
`.data/IDFM-gtfs`

   Dans le document `opendata gtfs.pdf` qui est dans le répertoire, donne des plus amples informations concernant la structure des fichiers fournis pas la région Île de France.

2. **Le script Python de transformation.** J'ai adapté et étendu le moteur d'évaluation de requêtes algébriques que nous avons développé l'année dernière. Il comporte le répertoire `pave` (une librairie Python), et le script `main_extraction_stations.py`. Vous pouvez les télécharger ici (le lien vers mon répertoire partagé). Le script `main_extraction_stations.py` va extraire les données correspondant aux réseau ferroviaire (routes, trains, stations). Ce script a besoin que les données télé-chargées soit dans un répertoire appelé `data`. Figure 1 présente la structure attendue par le script. Pour exécuter le script, la commande est:

   ```
   python3 main_extraction_stations.py
   ```

   L'exécution va créer le répertoire `./new-dataset` ou seront déposé les fichiers résultat. Figure 2 présente la structure du nouveau répertoire.

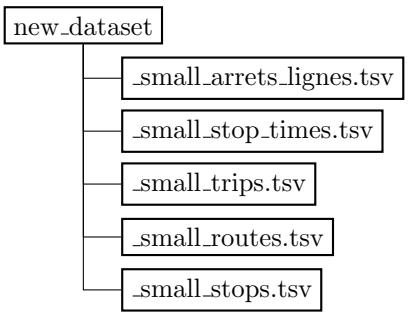

Figure 2: Répertoire résultat `new_dataset`

3.3 Demande 3:

Des scripts d'insertion des données transformées dans la nouvelle base de données.

Solution partielle: Les heures dans le fichier stop times n'etait pas dans le bon format. En ligne de commande, j'ai du faire les remplacements suivants.

```
$ sed -i '' $'s/\t24:/\t00:/g' ./_small_stop_times.tsv
$ sed -i '' $'s/\t26:/\t02:/g' ./_small_stop_times.tsv
$ sed -i '' $'s/\t25:/\t01:/g' ./_small_stop_times.tsv
$ sed -i '' $'s/\t27:/\t03:/g' ./_small_stop_times.tsv
```

```
COPY Routes(route_id, route_short_name, route_long_name)
FROM '/path/to/_small_routes.tsv' WITH DELIMITER E'\t' HEADER;

COPY Trips(route_id, trip_id, trip_headsign)
FROM '/path/to/_small_trips.tsv' WITH DELIMITER E'\t' HEADER;

COPY stopTimes(trip_id, stop_id, stop_sequence, arrival_time, departure_time)
FROM '/path/to/_small_stop_times.tsv' WITH DELIMITER E'\t' HEADER;

COPY stops(stop_id, stop_name, parent_station)
FROM '/path/to/_small_stop_times.tsv' WITH DELIMITER E'\t' HEADER;

COPY DetailedStops(route_id,
    stop_id,
    stop_name,
    stop_lon,
    stop_lat,
    OperatorName,
    Nom_commune,
    Code_insee)
FROM '/path/to/_small_arrets_lignes.tsv' WITH DELIMITER E'\t' HEADER;
```

3.4 Demande 4:

Exprimer en SQL les dix requêtes données dans la première section.

4 Installations

1. Installer PostgreSQL (préférable car il y a un serveur) ou SQLite.