Alexis Araujo
But info 3

Stations et Trains (RER et métro) en îlle de France Version 2

Code couleur :

- primary key
- Clé étrangère

Schéma Proposition des tables :

1. **Table "STOPS"** :
    - "stop_id" : Identifiant unique de la station.
    - “stop_name" : Nom de la station.
    - "stop_lon" : Coordonnée géographique : Latitude de la station.
    - "stop_lat" : La longitude de la station.
    - “OperatorName”
    - “Nom_commune” : Nom de la commune
    - “Code_insee” : Code insee

Table stops qui stock les stations.

2. **Table "ROUTES"** :
    - "route_id" : Identifiant unique de la ligne.
    - "route_short_name" : Nom de la ligne (e.x : RER B)
    - “route_long_name”

Table ligne qui stock les lignes du réseaux.

3. **Table "TRIPS"** :
    - "trip_id" : Identifiant unique du train.
    - "route_id" : Nom ou numéro du train (e.x : Train A).
    - "trip_headsign"

4. **Table "STOP_TIMES" :** (pour faire la liaison entre le train et la station car un train appartient à une ligne mais il passe pas par toutes les stations de la ligne)
    - "trip_id" : Clé étrangère : l'ID du train.
    - "stop_id" : Clé étrangère : l'ID de la station.
    - "stop_sequence"
    - "arrival_time" : Heure de passage du train
    - “departure_time”

Script Création de la base de données :

```
-- Suppression des tables existantes
DROP TABLE Routes;
DROP TABLE Trips;
DROP TABLE StopTimes;
DROP TABLE Stops;
DROP TABLE DetailedStops;

-- Création de la table Routes
CREATE TABLE Route(
    route_id VARCHAR(14) PRIMARY KEY,
    route_short_name VARCHAR(14),
    route_long_name VARCHAR(20)
);

-- Création de la table Trips
CREATE TABLE Trips (
```

```
    route_id VARCHAR(14) REFERENCES Routes(route_id),
    trip_id VARCHAR(49) PRIMARY KEY,
    trip_headsign VARCHAR(27)
);

-- Création de la table StopTimes
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)
);

-- Création de la table Stops
CREATE TABLE Stops (
    stop_id VARCHAR(30) PRIMARY KEY,
    stop_name VARCHAR(52),
    parent_station VARCHAR(30)
);

-- Création de la 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)
);

-- Ajout de la contrainte de clé étrangère sur DetailedStops
ALTER TABLE DetailedStops
ADD CONSTRAINT fk_detailedstops_routes
FOREIGN KEY (route_id) REFERENCES Routes(route_id);
```

Script transformation de données :

Voir le script dans le répertoire : “ main_extraction_stations “

Requêtes :

1. Etant donnée une ligne X (e.g. RER B), quelles sont ses stations?

```
SELECT Stops.*
FROM Routes
JOIN Trips ON Routes.route_id = Trips.route_id
JOIN StopTimes ON Trips.trip_id = StopTimes.trip_id
JOIN Stops ON StopTimes.stop_id = Stops.stop_id
WHERE Routes.route_short_name = 'RER B';
```

2. Quelles sont toutes les stations au sud (géographiquement) d'une station X donnée?

```
SELECT *
FROM Stops
WHERE stop_lat < (SELECT stop_lat FROM Stops WHERE stop_id =
'X_station_id');
```

3. Quelles sont toutes les stations au sud d'une station X donnée et qui sont desservies par le même train?

```
SELECT DISTINCT Stops.*
FROM Stops
JOIN StopTimes AS CurrentStopTimes ON Stops.stop_id =
CurrentStopTimes.stop_id
JOIN StopTimes AS XStationStopTimes ON XStationStopTimes.trip_id =
CurrentStopTimes.trip_id
WHERE XStationStopTimes.stop_id = 'X_station_id'
AND Stops.stop_lat < (SELECT stop_lat FROM Stops WHERE stop_id =
'X_station_id');
```

4. Etant donnée 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).

```
SELECT DISTINCT 'Direct Connection Exists' AS ConnectionStatus
FROM StopTimes AS XStopTimes
JOIN StopTimes AS YStopTimes ON XStopTimes.trip_id = YStopTimes.trip_id
WHERE XStopTimes.stop_id = 'Robinson_station_id'
  AND YStopTimes.stop_id = 'Massy-Palaiseau_station_id';
```

5. Étant donné deux stations X et Y de la même ligne, combien de stations les séparent? (attention: requˆete imprécise)

```
SELECT COUNT(*) - 1 AS StationsBetween
FROM StopTimes AS XStopTimes
JOIN StopTimes AS YStopTimes ON XStopTimes.trip_id = YStopTimes.trip_id
WHERE XStopTimes.stop_id = 'X_station_id'
AND YStopTimes.stop_id = 'Y_station_id'
  AND XStopTimes.stop_sequence < YStopTimes.stop_sequence;
```

6. Etant donnée deux stations X et Y pour lesquelles il existe une connexion directe, quelle est la durée minimale d'un trajet de X à Y ?

```
SELECT MIN(duration) AS MinDuration
FROM (
    SELECT
        XStopTimes.trip_id,
        (YStopTimes.arrival_time - XStopTimes.departure_time) AS duration
    FROM StopTimes AS XStopTimes
    JOIN StopTimes AS YStopTimes ON XStopTimes.trip_id = YStopTimes.trip_id
    WHERE XStopTimes.stop_id = 'X_station_id'
      AND YStopTimes.stop_id = 'Y_station_id'
      AND XStopTimes.departure_time < YStopTimes.arrival_time
) AS Durations;
```

7. Etant donnée 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 ?

```
SELECT DISTINCT 'Connection Exists' AS ConnectionStatus
FROM StopTimes AS XStopTimes
JOIN StopTimes AS TransferStopTimes ON XStopTimes.trip_id =
TransferStopTimes.trip_id
JOIN StopTimes AS YStopTimes ON TransferStopTimes.trip_id =
YStopTimes.trip_id
WHERE XStopTimes.stop_id = 'X_station_id'
  AND TransferStopTimes.stop_id <> 'X_station_id'
  AND YStopTimes.stop_id = 'Y_station_id'
  AND TransferStopTimes.departure_time > XStopTimes.departure_time
  AND YStopTimes.arrival_time > TransferStopTimes.arrival_time;
```

8. Combien de changements de ligne pour aller de la station X à la station Y ?

```
WITH RecursiveTripPaths AS (
    SELECT
        1 AS ChangeCount,
        XStopTimes.trip_id,
        YStopTimes.trip_id AS NextTrip,
        YStopTimes.departure_time AS NextDeparture
```

```
    FROM StopTimes AS XStopTimes
    JOIN StopTimes AS YStopTimes ON XStopTimes.stop_id = 'X_station_id'
                            AND YStopTimes.stop_id = 'Y_station_id'
                            AND XStopTimes.trip_id = YStopTimes.trip_id

    UNION ALL

    SELECT
        rt.ChangeCount + 1,
        rt.NextTrip,
        st.trip_id AS NextTrip,
        st.departure_time AS NextDeparture
    FROM RecursiveTripPaths AS rt
    JOIN StopTimes AS st ON rt.NextTrip = st.trip_id
)
SELECT MIN(ChangeCount) AS MinChangeCount
FROM RecursiveTripPaths;
```

9. Quel est le nombre maximal de changements pour un trajet entre deux stations ? (pire de cas)

```
WITH RecursiveTripPaths AS (
    SELECT
        1 AS ChangeCount,
        XStopTimes.trip_id,
        YStopTimes.trip_id AS NextTrip,
        YStopTimes.departure_time AS NextDeparture
    FROM StopTimes AS XStopTimes
    JOIN StopTimes AS YStopTimes ON XStopTimes.stop_id = 'X_station_id'
                            AND YStopTimes.stop_id = 'Y_station_id'
                            AND XStopTimes.trip_id = YStopTimes.trip_id

    UNION ALL

    SELECT
        rt.ChangeCount + 1,
        rt.NextTrip,
        st.trip_id AS NextTrip,
        st.departure_time AS NextDeparture
    FROM RecursiveTripPaths AS rt
    JOIN StopTimes AS st ON rt.NextTrip = st.trip_id
)
SELECT MIN(ChangeCount) AS MinChangeCount
FROM RecursiveTripPaths;
```

10. Quelles sont les stations X et Y dont le trajet le plus simple demande un nombre de changements maximal ?

```
WITH RecursiveTripPaths AS (
    SELECT
        XStopTimes.stop_id AS X_station_id,
        YStopTimes.stop_id AS Y_station_id,
        1 AS ChangeCount,
        XStopTimes.trip_id AS CurrentTrip,
        YStopTimes.trip_id AS NextTrip,
        YStopTimes.departure_time AS NextDeparture
    FROM StopTimes AS XStopTimes
    JOIN StopTimes AS YStopTimes ON XStopTimes.trip_id = YStopTimes.trip_id
    WHERE XStopTimes.stop_id <> YStopTimes.stop_id

    UNION ALL

    SELECT
        rt.X_station_id,
        rt.Y_station_id,
        rt.ChangeCount + 1,
        rt.NextTrip,
        st.trip_id AS NextTrip,
        st.departure_time AS NextDeparture
    FROM RecursiveTripPaths AS rt
    JOIN StopTimes AS st ON rt.NextTrip = st.trip_id
)
SELECT X_station_id, Y_station_id, MAX(ChangeCount) AS MaxChangeCount
FROM RecursiveTripPaths
GROUP BY X_station_id, Y_station_id;
```