Fusion GTFS

Dans l’état actuel, la production des fichiers GTFS s’appuie fortement sur les données opérationnelles, et présente ainsi l’information aux usagers sous la forme d’aller et retour, alors que bon nombre de ces lignes ne sont qu’un voyage ininterrompu. Or, du point de vue de l’usager, cette notion d’aller et de retour est contre-intuitive et rend plus complexe la présentation de l’information sur le service réellement effectué. Cette confusion est entre autre très présente dans l’application Chrono, celle-ci, dans son état actuel, tentant de présenter l’information à la manière des horaires papiers. D’un point de vue cartographique, l’interprétation pour les usagers est particulièrement présente, les noms des destinations intermédiaires pour les boucles n’étant pas nécessairement en parfaite correspondance avec les données opérationnelles (exemple : La ligne 81 a comme destination le boulevard du Fort-Saint-Louis, mais le terminus franchissable dans les données est situé à l’arrêt Marie-Victorin/de la Barre).

Dans le but d’offrir une meilleure marge de manœuvre dans l’utilisation des données GTFS, un travail préliminaire a été fait afin de fusionner les aller et retour des lignes en boucles dans les flux de données publiés. Le résultat de ces travaux est une application légère s’appuyant presqu’uniquement sur du SQL et permettant de convertir un GTFS avec terminus franchissable vers un GTFS avec les aller/retour fusionnés.

# Logique de fusion

Le GTFS, dans sa forme actuelle, contient toutes les informations nécessaires afin d’effectuer la fusion. Trois des fichiers doivent être fusionnés, soit trips, stop\_times, et shapes.

## trips

Pour les lignes en boucle, trois principaux éléments doivent être fusionnés dans la table des trips. En premier lieu, il importe de fusionner les aller et retour, qui est l’objectif fondamental de ce projet. Pour accomplir cette tâche, il suffit de fusionner selon la colonne block\_id, qui indique la continuité des opérations d’un véhicule créant ainsi un terminus traversable. La structure fondamentale des routes étant changée par cette fusion, il est aussi important de modifier la direction de ces départs. Pour ce faire, toutes les routes ayant des tracés en boucles se voient attribuer la direction 0, puisque les tracés deviennent unidirectionnels suite à la fusion. En troisième lieu, les trips headsign doivent aussi être changés, afin d’incorporer les deux valeurs disponibles pour l’aller et le retour. En reprenant l’exemple de la ligne 81, le nom de la destination devient Terminus Longueuil via Boulevard du Fort-Saint-Louis, ce qui permet de fusionner l’aller et le retour sans perte d’information.

Il est à noter qu’il importe ici de modifier les valeurs de trip\_id, block\_id et shape\_id, ceux-ci étant erronés une fois la fusion complétée.

### Algorithme

1) Déterminer l’heure de départ d’un trip

2) Ordonner les trips par block\_id et leur attribuer un numéro de séquence sur le block

3) Déterminer les blocks ayant plus d’un trip

4) Déterminer quelles lignes utilisent des blocks avec plusieurs trips

5) Déterminer les trip\_headsign individuels et les fusionner

6) Algorithmes pour fusionner les stop\_times et shapes

7) Attribuer les nouvelles valeurs de trip\_id,block\_id,shape\_id,direction, et headsign

8) Fusionner avec les trips non modifiés

## stop\_times

La fusion des stop\_times découle de la fusion des trips. Quatres éléments fondamentaux sont à modifier, soit l’élimination d’arrêts desservis de façon consécutive dans le même block, le calcul des distances parcourues le long du shape, l’insertion de valeurs de stop\_headsign pour combler le vide engendré par la fusion des allers et retours et bien sûr l’identifiant du trip.

L’identification et l’élimination d’arrêts doublés est rendu nécessaire en raison de la structure actuelle des données. En reprenant l’exemple de la ligne 81, la fin du parcours retour se fait à l’arrêt Marie-Victorin/de la Barre. Or, cet arrêt est présent dans les deux trips, en descente seulement dans le voyage de retour, et en montée seulement dans le voyage aller. Le processus de fusion doit donc éliminer un de ces doublons, en associant les bonnes valeurs de temps et de type d’embarquement au stop\_time restant. On prendra donc le temps d’arrivée du voyage retour, le temps de départ du voyage aller, on attribuera le code 0 pour les montées/descentes, alors que les autres variables pourront être conservées.

Les distances parcourues sur le shape impliquent de classer les arrêts selon leur numéro de séquence de voyage ainsi que leur propre numéro de séquence et de calculer les différentiels de distance parcourus déjà existant dans la colonne shape\_dist\_traveled. On peut par la suite recalculer la distance totale parcourue à ce point.

Finalement, la valeur de headsign est intégrée en utilisant celle-déjà existante, ou en remplaçant par celle du trip avant la fusion, permettant ainsi de fournir l’information telle qu’elle existe déjà présentement (exemple de la ligne 81 : le headsign est boulevard du Fort-Saint-Louis jusqu’à cet arrêt, pour devenir Terminus Longueuil par la suite, alors que le trip\_headsign est Terminus Longueuil via Boulevard du Fort-Saint-Louis, présentant l’information à deux niveaux).

### Algorithme

1) Déterminer les stop\_times qui font partie de trips qui seront fusionnés

2) Déterminer les points de bascule entre le retour et l’aller

3) Éliminer l’arrêt doublon en combinant ses informations dans un seul stop\_time

4) Recalculer les distances parcourues à l’aide de différentiels sur l’ensemble du trip fusionné

5) Associer des valeurs de headsign en utilisant soit la valeur actuelle, soit celle du trip d’origine non fusionné

6) Redéfinir la valeur de trip\_id pour lier au nouveau trip fusionné

## shapes

Un trip ne pouvant être lié qu’à un seul shape, il est essentiel, pour les trips fusionnés, les shapes doivent eux aussi être fusionnés. À l’image des stop\_times, les principaux éléments de la fusion concernent l’élimination de doublons et le calcul des distances parcourues sur le nouveau shape. La même logique s’applique ici, avec l’ordonnancement des points formant la ligne et l’élimination d’un des deux points identiques consécutifs, suivi du calcul de la différence de distance parcourue entre chaque point et de la sommation de ceux-ci pour obtenir la nouvelle distance parcourue. Les nouveaux shapes devront aussi se voir attribuer un nouvel identifiant, soit un composite des deux shapes fusionnés.

### Algorithme

1) Déterminer les shapes à fusionner

2) Séquencer les points des shapes à fusionner et éliminer les doublons

3) Calculer les différentiels de distance parcourus

4) Calculer la somme de la distance parcourue pour chaque point en faisant la sommation des différentiels

5) Attribuer les nouvelles valeurs de shape\_id à partager avec la table trips

# Utilisation de l’outil

La procédure de fusion est programmée en SQL directement, sans intervention d’un langage de programmation. Cette décision a été prise dans l’optique de conserver au maximum la portabilité de la procédure développée. Malgré tout, afin d’automatiser le processus, un arrimage sur le langage ruby et la plateforme rails a aussi été complété, notamment pour l’importation du GTFS d’origine et l’exportation vers le GTFS fusionné. La plateforme rails est fondamentalement superflue, mais présente certains outils rendant plus facile le développement, notamment le système de migrations permettant de facilement créer la structure de la base de données.

La procédure SQL est dans le dialecte PostgreSQL, rendant possible certaines incompatibilités avec d’autres dialectes. Notamment, pour éviter le stockage de tables intermédiaires, celle-ci fait largement usage de Common Table Expression (CTE), qui permettent de créer des tables temporaires en mémoire et de les réutiliser dans des requêtes subséquentes. En plus des CTE, de nombreuses Window Functions sont utilisées, celles-ci permettant de faire des opérations sur les voisinages d’une entrée dans la base de données. Ces fonctions permettent notamment de déterminer aisément les doublons en observant l’entrée suivante (lead) sur une partition de données. Elles permettent aussi d’aisément calculer les distances à partir des différentiels, une somme pouvant être faite sur l’ensemble des éléments précédents d’une partition.

# Commandes d’utilisation

L’application Rails dans son état actuel offre deux options pour l’export des données vers le format fusionné, soit une importation et une exportation en deux temps avec les commandes suivantes (dans un terminal, dans le dossier de l’application)

rake import\_from\_csv[C:\\chemin\\dossier\\gtfs]

rake merge\_trips\_on\_block[‘lignes\_a\_fusionner’,C:\\Dossier\\de\\sortie]

où lignes\_a\_fusionner est une liste de lignes séparée par des espaces (ex : ‘1 6 81’)

Une commande permettant l’export complet, d’un CSV d’entrée vers un CSV de sortie est aussi fournie afin de limiter l’exécution à une seule opération

rake merge\_gtfs[C:\\chemin\\dossier\\gtfs, ‘lignes\_a\_fusionner’]

où lignes\_a\_fusionner est une liste de lignes séparée par des espaces (ex : ‘1 6 81’)