Réalisé par : Loïc LAMOUR, Younes HITMI et Matthieu GUIARD DEXET

Analyse et Description de projet

LocoLog

I- Reformulation complète de la description du projet

Notre projet de base de données relationnelle , consiste à la gestion des infrastructures ferroviaires et des trajets interconnectés. Les différents points importants de ce projet sont le modèle de la base de données et les requêtes à effectuer sur celle-ci.

Pour le modèle, il contient 3 grands ensembles qui sont les lignes, les gares et les trains. Explicitons les différents liens entre ces 3 ensembles. Nous établissons que les trains effectuent des trajets entre deux gares nécessairement consécutives à une date et heure précises et que ces trajets ont une durée. De plus, nous considérons que chaque ligne est définie par les gares par lesquelles elle passe. Cela nous permet donc de connaître les horaires pour chaque ligne, gare et train.

Ensuite, il peut aussi exister des maintenances et incidents pour les lignes, gares et trains. Celles-ci sont définies par l'endroit impacté, leur type, une date et heure de début, une durée, et dans le cas où l'élément impacté est une ligne, les deux gares entre lesquelles les trajets sont impactés.

Quand aux requêtes, notre base de données doit pouvoir :

- Donner des trajets (pas nécessairement entre deux gares consécutives) qui optimisent les temps d'attente et/ou le nombre de correspondances.
- Déterminer s'il est nécessaire d'effectuer des maintenances sur certains trains en fonction de leurs heures cumulées de trajet ou des incidents signalés.
- Déterminer les gares pouvant être saturées en fonction des heures de pointe.
- Déterminer de nouvelles lignes permettant d'optimiser les correspondances.
- Trouver des trajets alternatifs lorsqu'un incident a lieu.
- Déterminer si certains trains peuvent être réaffectés pour couvrir une panne.
- Déterminer les endroits sur lesquels un incident aurait un grand impact sur la ponctualité globale du réseau ferroviaire.

II- Explications des requêtes SQL

- Requête 1 : Cette requête récursive a pour principal objectif de trouver les trajets les plus optimisés (en nombre de correspondances et en temps d'attente) pour aller d'une gare de départ donnée à une gare d'arrivée.

La première partie (avant le UNION ALL) initialise notre cas de base. La requête d'initialisation sélectionne tous les trajets directs (à 0 correspondances donc) à partir de la gare de départ. L'attribut "nb_correspondances" est initialisé à 0, pour que l'on puisse l'incrémenter de manière récursive à chaque itération. Les attributs sequence_trains est une liste des trains empruntés durant le voyage, et 'duree_totale' représente la somme des durées des segments (trajets intermédiaires) et des temps d'attente.

La deuxième partie de la requête (après le UNION ALL) incrémente de 1 le nombre de correspondances à chaque appel, et la durée totale est additionnée à la durée du trajet actuel ainsi qu'à la différence entre l'heure de départ de ce trajet et la somme de l'heure de départ de dernier trajet et de sa durée. Le train utilisé pour le nouveau trajet (trajet actuel) est ajouté à la liste des trains empruntés. Tous ces attributs sont sélectionnés sur la table générée par une jointure au niveau des id de gare de départ et d'arrivée des trajets successifs.

La clause WHERE permet de vérifier que l'heure de départ du dernier trajet est bien postérieure à l'heure d'arrivée du dernier, et de limiter les boucles infinies liées aux conditions de la récursion.

Les résultats sont ainsi filtrés de manière à ne garder que les trajets qui atteignent la destination définie, en les classant par ordre croissant de nombre de correspondance, puis de durée totale de trajet (en inversant les critères, le résultat reste inchangé). Le nombre d'options est limité à 1 pour ne garder que la meilleure option.

- Requête 2 : Cette requête a pour but de trouver les trains nécessitant maintenance, ceux dépassant le seuil heures cumulées et ceux ayant subi un incident et forcément n'étant pas déjà en maintenance.

  Pour réaliser cela, on retrouve un premier SELECT, qui va nous donner les id des trains de la tables Trains à l'aide de la clause WHERE vérifiant que le seuil d'heures cumulées est dépassé et que ces trains là ne sont pas actuellement en maintenance , pour cela on vérifie que l'id du train n'est pas dans le résultat d'un sous requête qui correspond à l'ensemble des trains en cours de maintenance.

  Ensuite pour le second SELECT ,qui va nous donner les id des trains de la tables Incidents_de_Train à l'aide de la clause WHERE que ces trains là ne sont pas actuellement en maintenance , pour cela on vérifie que l'id du train n'est pas dans le résultat d'un sous requête qui correspond à l'ensemble des trains en cours de maintenance.

- Requête 3 : Cette requête a pour objectif de déterminer si des gares sont saturées en heures de pointe, et s' il y en a, quelles sont ces gares. Le seuil de saturation est fixé à 60% d'utilisation de la capacité. Passé ce seuil, la gare est "saturée". En lançant la requête sur pgAdmin, on déduit que la seule gare saturée dans la plage horaire 7-9h et 17-19h est la gare Saint-Charles à Marseille.

Cette requête, en partant du principe qu'une gare accueille au maximum 3 trains par heure, avec 4 heures de pointe au total, compte le nombre de trains qui arrivent/partent de chaque gare et le divisent par la capacité pour multiplier le résultat par 100. Cela donne un taux en % de saturation de la gare.

- Requête 4 :Cette requête a pour but de savoir comment intégrer une nouvelle ligne ferroviaire dans le réseau existant tout en optimisant les correspondances, c'est-à-dire on va regarder les trajets dont le nombre de correspondances dépasse un certain seuil dans notre cas 2. On va ensuite choisir un trajet pour lequel on va proposer la création d'une nouvelle ligne entre les deux gares concernées.

  Pour réaliser cela , on commence par démarrer une transaction , puis on commence notre requête avec de la récursivité , un WITH RECURSIVE nous permettant d'obtenir tous les chemins disponibles dans notre réseau ferroviaire.

  Ensuite , sur tous ces chemins disponibles on effectue une requête qui nous permet d'obtenir pour chaque trajet possible , la possibilité effectuant le minimum de distance et en groupant et ordonnant par distance, départ , arrivée et l'horaire. Puis on détermine , un id pour une nouvelle ligne en prenant l'id maximum parmi les lignes actuelles du réseau ferroviaire.

  On détermine également , un id pour un nouveau train en prenant l'id maximum parmi les trains actuels du réseau ferroviaire.

  Par la suite, on effectue une requête sur les trajets possibles avec minimum de distance , de manière à obtenir ceux dépassant 2 correspondances.

  Pour finir pour le résultat final on se limite à un élément qui correspond à notre proposition de ligne et si tout s'est passé sans erreur on peut terminer la transaction et COMMIT.

- Requête 5 : Cette requête a pour but de déterminer les trajets alternatifs disponibles à partir de l'heure courante (ce qui ne sera pas le cas ici pour tester) en prenant en compte les incidents courants.

  Pour cela, 3 fonctions ont été créées. Tout d'abord la fonction garesComprises, qui va renvoyer la liste des gares comprises entre deux gares données, sur une ligne donnée avec aussi la gare de départ et la gare d'arrivée. La deuxième fonction, trajetDansChemin, renvoie un booléen en fonction de l'appartenance d'un trajet à un ensemble de gares. La troisième fonction, trajetsImpactesParIncidents, renvoie une table de tous les trajets impactés par un incident donné sur une ligne en fonction de la date et l'heure auxquelles se produit l'incident et les dates et heures des trajets.

  La requête est suivie d'un WITH RECURSIVE, CheminsSansIncidentsDisponibles, qui renvoie une table de tous les trajets possibles et disponibles sans passer par les trajets impactés par les incidents courants.

  Finalement, la requête sélectionne les trajets calculés précédemment en prenant seulement le trajet le plus court en cas de doublons sur les mêmes horaires, et en

les ordonnant par distance en nombre de gares intermédiaires, puis par gare de départ, d'arrivée, de date, d'heure, de durée, etc.

- Requête 6 :Cette requête a pour but de déterminer les trains qui peuvent être réaffectés pour couvrir une panne sur une autre ligne , c'est-à-dire obtenir les trains disponibles au moment donné.

  Pour cela , on effectue une première requête qui permet d'obtenir les trains disponibles parmi les maintenances de trains c'est-à-dire ceux dont la maintenance est terminée.

  Ensuite , sur cette première requête on en effectue une seconde qui nous permet d'obtenir ceux qui n'effectuent pas de trajet actuellement.

  Finalement cela nous permet d'obtenir les trains disponibles.

- Requête 7 : Cette requête a pour but de déterminer les incidents qui ont le plus grand impact sur la ponctualité globale du réseau ferroviaire, autrement dit, les endroits où un incident (train/gare/ligne) serait le plus problématique pour la ponctualité globale du réseau.

  La requête crée ainsi 3 vues, pour chaque type d'incidents, où elles donnent les gares/trains/lignes et le nombre de passages/utilisations pour chacun. Elles ordonnent aussi par nombre de passages/utilisations pour avoir les gares/trains/lignes les plus important(e)s, donc sur lequel(le)s un incident aurait le plus d'impact.

III- Description des jeux de données utilisés

Les jeux de données utilisés étaient principalement focalisés sur la table Trajets.

On peut justifier cela par le fait qu'une grande partie des requêtes aient besoin des données de la table trajets. Plus spécifiquement les questions 3, 4 et 7. La question 3 nécessite beaucoup de données sur une plage horaire précise, les heures de pointe. Pour la question 4 , on a besoin d'un ensemble de trajet différents pour que ce soit cohérent avec une utilisation réelle. Enfin pour la question 7 on a également besoin d'un ensemble de trajet différents pour que ce soit cohérent avec une utilisation réel et d'avoir une gare qui est beaucoup plus utilisée de manière à avoir des incidents plus impactants que d'autres.

On a également ajouté des données dans les tables incidents de trains et maintenance de trains pour les questions 2 et 6 qui nécessite d'avoir des trains disponibles c'est-à-dire des trains dont la maintenance est terminée et des trains nécessitant une maintenance c'est-à-dire ayant subi un incident mais pas encore en maintenance.

Réalisé par : Loïc LAMOUR, Younes HITMI et Matthieu GUIARD DEXET

IV- Analyse critique du déroulement du projet

- Découpage / Répartition du travail en équipe :

  Pour ce qui est des décisions au niveau structurel de la base de données (Schéma EA et schéma relationnel), elles ont été prises par l'ensemble du groupe. C'est un travail collaboratif qui a été fait durant les TDS. Il en est de même pour les scripts de peuplement et de création de tables.

  En ce qui concerne les différents scripts de requêtes (7 au total), nous avons essayé de les répartir équitablement. Deux membres en ont écrit 2, et un membre en a écrit 3. Chaque membre à choisi les requêtes qu'il voulait écrire.

- Difficultés rencontrées :

  Difficultés à créer des situations réalistes et qui "arrangent" nos simulations. Pour tester les saturations des gares aux heures de pointe par exemple, il est assez difficile de produire des scripts de peuplement assez riches pour simuler des saturations dans des gares tout en gardant un cohérence spatio-temporelle (des trains qui arrivent et partent à des heures cohérentes laissant place à d'autres trains qui arrivent ensuite). La création et la manipulation de données qui normalement sont produites automatiquement par des événements réels s'avèrent assez compliquées. C'est ce qui pousse, entre autres, à entrer des données plus favorables aux situations que nous recherchons. Cela facilite les tests des requêtes et la manipulation des données.

- Améliorations éventuelles de l'organisation des tâches :

  R.A.S

- Réussites du projet et éléments clés :

  Les règles de normalisation ont été respectées et la structure de notre base de données est bonne. La mise en place d'un bon schéma relationnel a permis une bonne compréhension / visualisation de la base de données, ce qui a facilité l'écriture des requêtes. C'est la partie la plus importante du projet, et elle a défini la suite de ce dernier.

- « Et si c'était à refaire ? » :

  Demain , si on devait refaire le projet, on ferait des améliorations au niveau de notre organisation. Notamment dès le début lors de la création du schéma entité-association et schéma relationnel , l'amélioration serait prendre beaucoup plus en compte les requêtes à effectuer pour facilement effectuer celle-ci par la suite.

Réalisé par : Loïc LAMOUR, Younes HITMI et Matthieu GUIARD DEXET