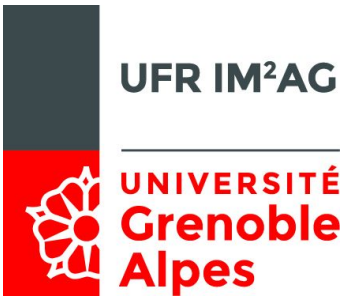
TROUVIN Paul
BLONDEL Quentin
DEDIEU-MEILLE Andréas
PERRIN Kyllian



Projet Airchance

Scénarios

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

Planification d'une InstanceVol :

Déroulement d'une transaction :

- choix du vol (ligne)
- choix de l'avion
- choix du personnel affecté :
    - choix du PNT
    - choix du PNC
- Insertion des lignes dans les tables concernées

Exemple de concurrence : imaginons que nous voulons insérer une nouvelle instance dans la table InstanceVol et qu'on affecte le PN n°46 à cette instance de vol (table EmployeInstanceVol). Si dans le même temps une autre transaction fait une insertion/modification pour affecter ce même PN à une instance de vol aux même dates, cela va poser problème. En effet le PN sera toujours vu comme disponible par notre première transaction (lecture fantôme). Le seul moyen de régler cette lecture fantôme est de placer la création d'une instance de vol en mode **serializable**.

Supprimer une InstanceVol existante :

Lors de la suppression d'une instance de la table InstanceVol, on va au niveau applicatif re-affecter chaque client dans l'ordre de la qualité de la classe. S'il n'y a aucune place de la même classe, ils seront affectés à une classe plus basse. S'il n'y a aucun avion disponible, les réservations seront supprimées. Notre application ne prend pas en charge la partie remboursement du trajet.

À partir de la, pour chaque client on va boucler en suivant le régime suivant :

- On prend un client avec la classe la plus haute possible,
- On prend l'avion avec de la place au plus tôt, et on y affecte le client.

Comme pour la modification, la suppression d'une instance de la table InstanceVol devra être **serializable**.

En effet, dans le mode **read uncommitted**, on risque, si on ne confirme pas la suppression du vol, d'avoir une lecture sale pour le client. Et dans le mode **read committed**, on veut éviter toute écriture sale afin d'éviter le soucis des PN disponibles.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Créer un nouveau vol (ligne, dans notre table Vol) :

Déroulement d'une transaction :

- Entrée origine et destination
- entrée distance et durée
- choix type de Vol
- choix nombres de places OU poids (en fonction du type de vol)

Afin de créer un vol, c'est assez simple même s'il faut garder une certaine cohérence, on vérifie que le point de départ soit différent du point d'arrivée, le reste des informations est à la discrétion du créateur de la ligne. On peut donc utiliser le **read uncommitted**.

Modifier/supprimer une réservation :

La modification d'une réservation est représentée par la création d'une nouvelle réservation et la suppression de l'ancienne.

Le tout est en **read committed** ainsi chaque requête de la transaction ne voit que les modifications validées avant son exécution. Ainsi, il ne sera pas possible de lire une nouvelle instance d'InstanceVol non commit.

Si deux personnes réservent en même temps leur place, un trigger prémunit d'une écriture sale en vérifiant qu'aucune place n'est réservée deux fois. Si cela arrive, le dernier utilisateur qui commit se verra refuser sa réservation car quelqu'un vient tout juste de la réserver.

Réserver une place sur une InstanceVol :

Déroulement d'une transaction de réservation :

- Login du client qui veut prendre une InstanceVol (signifie que le client est déjà en base)
- Recherche des Instances vol :
  - choix de ville départ, ville arrivée (Vol)
  - choix de la dateDepart (InstanceVol)
  - choix du type de Vol (Vol)
  - choix de la catégorie de place
  - Afficher les InstanceVol qui ont leur placeRest cohérent avec la catégorie entrée
- Choix de l'instance Vol
- Choix de la place en fonction de l'InstanceVol choisi
- Enregistrer nouveau Réservation Passager et ResaVolPlace

Nous avons choisis du **read committed** pour réserver une place.

Imaginons que nous voulons créer une nouvelle réservation. Il faut choisir avec des select la ville départ, ville arrivée, dateDepart, type de Vol, catégorie de place et finalement les InstanceVol.
Si une opération de suppression se déroule sur un InstanceVol en même temps, cette opération sera serializable comme énoncé plus haut et va donc bloquer la réservation.
Si la réservation continue malgré cela, il va tenter un insert sur une InstanceVol qui n'existe plus, et va donc jeter une erreur car la table ResaVolPlace a pour clé externe numInstance.
Si la requête de suppression passe après celle de réservation, aucun problème car la suppression va réaffecter les réservations.

2e scénario : si 2 clients veulent réserver la même place. Etant du read committed, les transactions vont lire les même informations. La première transaction qui va faire un insert va être valide, la seconde jettera une erreur oracle sur les contraintes d'intégrité. Elle pourra être attrapée, et on pourra dans l'application afficher une message d'erreur dans le genre "La place que vous avez sélectionné n'est plus disponible".

Confirmation de l'arrivée d'un vol :

Lors de l'arrivée d'un vol, il y à deux choses à faire, modifier l'état de l'instanceVol et affecter l'heure d'arrivée du vol, mais cela est effectué par un trigger, mais il faudra bien faire attention de ne pas commit avant d'avoir rentrer les deux

La transaction peut se faire en **Read Uncommitted**, il n'est pas important de vérifier la cohérence de la base lors de cet update car on ne fait que modifier un état.