Projet intégrateur en gestion de données: Application VisioPad

<u>Groupe 5 :</u> GOURDON Stéphanie, AVANZINO Aurélien, OUBARI Mohammad et VARINIER Rémy
M1 MIAGE
Année 2020-2021

SOMMAIRE

I. Scénarios de concurrence

Les problèmes engendrés par le partage des données en simultané entre les différents utilisateurs sont nombreux et peuvent poser plus ou moins des difficultés:

A. ***Perte de modifications***
B. ***Non-reproductibilité des lectures***
C. ***Modification temporaire***
D. ***Lecture Fantôme***

A. Exemple de lecture fantôme :

Imaginons que deux membres d'une famille (par exemple deux frères **qu'on appelle f1 et f2** pour la suite) souhaitent voir leur grand-père à une date et à un créneau identique (*On supposera que f1 ne connaît pas les intentions de f2 et inversement*).

F1 se connecte à l'application VisioPad et se dirige dans la partie dédiée à la réservation. F1, après vérification, choisit la date et l'heure du créneau qui lui convient et confirme.
Au même moment, F2 se connecte et se dirige également dans la partie réservation et souhaite prendre un rendez-vous à la même date que F1.
Dans ce cas là, il se peut que la transaction associée à F2 ne prenne pas en compte les données ajoutées par la transaction associée à F1 et ne verra donc pas le rendez-vous pris.

B. Exemple de non-reproductibilité des lectures :

Imaginons deux membres du personnel (On les nommera respectivement M1 et M2 pour la suite).

M1 souhaite effectuer la somme totale des rendez-vous des résidants de l'unité 4 tandis que M2 ajoute un rendez-vous supplémentaire à un résident X et en annule un d'un autre résidant Y en parallèle.
Dans ce cas-là, il se peut que M1 lit la donnée de X après que ce dernier soit modifié et lit les données de Y avant que ce dernier soit modifié.
On aura donc des modifications appliquées sur certains éléments mais pas sur les autres ce qui peut fausser les résultats attendus.

C. Exemple de perte de modifications :

Imaginons deux membres du personnel (On les nommera respectivement M1 et M2 pour la suite)

On a demandé à M1 d'ajouter dans la base de données de nouvelles places dans l'EHPAD X (10 places exactement) tandis qu'on a demandé à M2 d'en retirer sur ce même EHPAD.
M1 commence à faire les modifications et M2 fait débutent les siennes juste après, dans ce cas là, il peut arriver que l'exécution des opérations soient décalées au risque alors d'obtenir des résultats erronés

Par exemple, l'EHPAD X à 100 places disponibles. M1 lit la donnée et ajoute 10 places (donc nbPlaces = 110) mais n'enregistre pas encore la modification. M2 lit aussi la donnée et retire 25 places (donc nbPlaces = 75).

M1 enregistre la modification puis M2 enregistre également.
Nous aurons alors 75 places disponibles au lieu de 85 (100+10-25) dans l'EHPAD X car M2 n'avait pas la modification effectuée par M1 lorsqu'il a lu la donnée nbPlaces.

D. Exemple de Modification temporaire :

Une transaction va modifier un élément de la BD et la transaction échoue. L'élément mis-à-jour est alors accédé par une autre transaction avant réparation.

Imaginons qu'un utilisateur prenne un rendez-vous pour voir un proche dans un EHPAD a une date et heure précise et l'enregistre, ainsi le serveur et la base de données vont commencer le processus d'écriture des modifications.
Pendant ce processus, un second client souhaite prendre également un rendez-vous à la même date, le serveur va donc envoyer au client les informations qu'il a sur cette date ainsi que les modifications en cours d'enregistrement apportées par le premier client.
Cependant après plusieurs essais, la base de données n'arrive pas à effectuer l'enregistrement. On a donc des données erronées sur l'affichage du second client car la base de données n'a pas eu le temps de faire le rollback avant la récupération

II. Opération d'accès/requêtes types

A. Lecture Fantôme

Transaction 1 : (T1)	Transaction 2 : (T2)	Database
select(RESIDENT)		
	select(RESIDENT)	
select(CRENEAUVISIO)		r C1.disponible = 1
select(RENDEZVOUS)		
update(CRENEAUVISIO)		w C1.disponible = 0
	select(CRENEAUVISIO)	r C1.disponible = 1
	Commit	
insert(RENDEZVOUS)		
Commit		

B. Non-reproductibilité des lectures

Transaction 1 : (T1)	Transaction 2 : (T2)	Database
select(RENDEZVOUS)		r nb(RENDEZVOUS) = 10
update(RENDEZVOUS)		w nb(RENDEZVOUS) = 11
	select(RENDEZVOUS)	r nb(RENDEZVOUS) = 11
	Commit	
update(RENDEZVOUS)		w nb(RENDEZVOUS) = 10
Commit		

C. Perte de modifications

Transaction 1 : (T1)	Transaction 2 : (T2)	Database
select(ETABLISSEMENT) update(ETABLISSEMENT) Commit	select(ETABLISSEMENT) update(ETABLISSEMENT) Commit	r E1.nbPlaces = 100 r E1.nbPlaces = 100 w E1.nbPlaces = 110 w E1.nbPlaces = 75

III. Choix pour résolution

Afin de résoudre les problèmes posés par la mise en place de processus en concurrence qui utilisent et modifieront la base de données.

- **Interdire** certaines possibilités via l'utilisation de contraintes au niveau BDD/Serveur ou Client
    - Exemple un Trigger en BDD empêchant la modification du créneau une fois que le rendez-vous a été validé par un personnel
- **Prioriser les types de requêtes** lors du traitement par le serveur
    - Par exemple, les requêtes de création sont prioritaires au lecture et/ou à l'update, etc…
- **Sérialiser les transactions** quand cela est possible
    - Via l'utilisation de Thread ou de fonction synchronized côté serveur qui empêchera une utilisation simultanée
- **Rafraîchir** les données côté client de manière cyclique afin que l'utilisateur puissent avoir les dernières modifications effectuées
    - Utilisation si possible d'un observable sur le serveur qui notifie les autres composants de l'IHM lorsque le serveur indique une modification
    - Ou bien un processus en arrière-plan côté IHM qui s'occupe de faire des requêtes au serveur de manière périodique pour récupérer les données