Verrous

Transaction

Introductions aux Bases de Données

Nathanaël Martel

Verrous / Lock

Problème :

- **l'utilisateur 1 récupère un enregistrement en vue de le modifier**
- **l'utilisateur 2 récupère le même enregistrement en vue de le modifier**
- **l'utilisateur 1 enregistre son nouvelle enregistrement**
- **l'utilisateur 2 enregistre son nouvelle enregistrement**

Verrous / Lock

Problème :

- **La base ne reflète pas des modifications des l'utilisateur 1, au mieux ils seront ignorées…**
- **… dans certains cas, les données peuvent être incohérente**

Verrous / Lock

Solutions

- **Avant de faire ce type de modification, il est possible de le signaler à la base en lui demandant de verrouiller tel enregistrement ou tel table**

Verrous / Lock

- l'utilisateur 1 demande un verrous sur un enregistrement, il est accordé
- l'utilisateur 2 demande un verrous sur un enregistrement, il est refusé
- l'utilisateur 1 enregistre son nouvelle enregistrement et libère le verrou
- l'utilisateur 2 peut alors demander un verrous

Verrous / Lock

Nouveau problème :

- **Que se passe-t-il si l'utilisateur 1 à poser un verrou sur une table t1 et que l'utilisateur 2 à poser un verrou sur une table t2,**
- **Et que pour finir leur transaction l'utilisateur 1 a besoin de la table t2 et l'utilisateur 2 a besoin de la table t1**
- **On arrive a un «deadlock»**

Verrous / Lock

Gestion des « deadlock » :

- Il faut que les verrous soit suffisamment important sans être trop bloquant…
- La solution est a voir au cas par cas en fonction du modèle
- Les verrous peuvent être mise en place dans l'application plutôt que dans la base.

Transaction

Problème

- **Pour assurer la cohérence des données certaines transaction doivent être faite ensemble et si l'une échoue (verrous sur une table, enregistrement inexistant…), il faut pouvoir annuler l'ensemble**

Transaction

Solution

- **Signaler à la base que nous avons affaire à une transaction, un ensemble de requêtes indissociables :**

```
BEGIN TRANSACTION
UPDATE PropertyForRent
SET StaffNo = 'SN99'
WHERE StaffNo = 'SG37';
DELETE FROM Staff WHERE StaffNo = 'SG37';

COMMIT TRANSACTION
```

Transaction

Avantage

- **Atomicity - each unit of work is *indivisible*; “all-or-nothing” (transactions that don't complete must be undone or “rolled-back”)**
- **Consistency - a transaction transforms the database from one consistent state into another (intermediates may be inconsistent)**
- **Isolation - each transaction effectively executes independently - one transaction should not see the inconsistent/incomplete state of another transaction**
- **Durability - once a transaction is complete, its effects cannot be undone or lost (it can only be “undone” with a *compensating transaction*)**

Conclusion

Il est possible de mettre des mécanisme en place pour s’assurer de la cohérence des données