

GESTION DES TRANSACTIONS | TRIGGERS

CHAP 4.0 H.CHARARA

Un SGBD est multi-utilisateurs >> une même donnée peut être manipulée par plusieurs utilisateurs à la fois :

- ❑ problèmes d'accès concurrents (perte de mise à jour, lecture incohérente, ...)
- ❑ comment garantir la cohérence des données ? contraintes de temps réel ?

Définition : Une transaction est une unité logique de travail : séquence d'instruction SQL soumise à une BD. Ceci implique la modification d'au moins une donnée, fait passer la BD d'un état cohérent à un autre. Une transaction prend 2 états :

- ❖ "validée" "committed"
- ❖ "échouée" "rolled back"

- Une transaction exécutée ne peut pas être annulée, l'ensemble d'instruction est une entité logique.
- Tout est exécuté ou abandonné, pour effectuer cela SQLServer gère un journal des transactions.

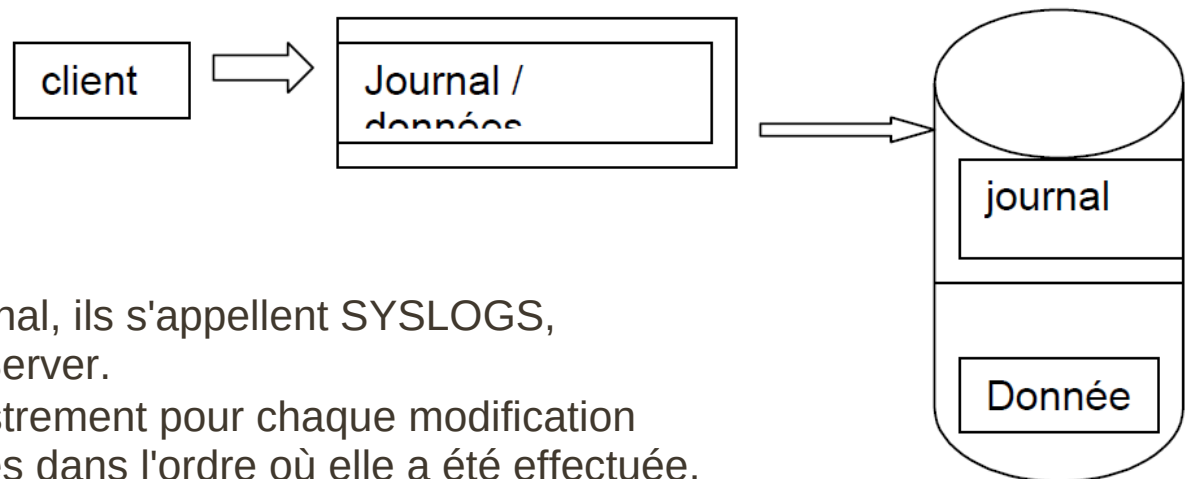
Gestion :

- Sql serveur n'accepte dans la BD que des opérations validées (committed).
- Des utilisateurs multiples ont accès à une ou plusieurs bases de données.
- Des utilisateurs multiples ne peuvent pas accéder aux mêmes données en lecture et écriture simultanément.
- SQL server protège contre les problème logiciel, matériel, panne alimentation.

Traitement des transactions :

- SQLServer gère un journal des transactions.
- SQLserver verrouille des pages de données au cours des transactions de façon à ce que d'autres utilisateurs ne puissent pas accéder à des données en cours de modification.
- SQLServer exécute une reprise automatique (recovery) lorsqu'il redémarre.
- Les instructions de contrôle des transactions déterminent à quel moment une transaction commence ou est annulée.

Mécanisme du journal des transactions



- Chaque BD possède son journal, ils s'appellent SYSLOGS, géré exclusivement par SQLServer.
- Ce journal contient un enregistrement pour chaque modification apportée à la base de données dans l'ordre où elle a été effectuée.
- SQLServer met en mémoire les opérations de lecture et d'écriture de toutes les tables y compris la table SYSLOGS, dans une zone de mémoire appelée cache de données.
- Ce cache est composé de pages.

Le flot des données est dans une transaction

Gestion du cache

- Le journal des transactions est généralement distinct des véritables données de la base.
- Nous avons un vidage du cache vers le journal à chaque fois qu'une transaction est validée ou besoin de place dans le cache.
- Si un nombre de transaction a été validée, un point de reprise (**check point**) est réalisé pour cette BD.
- Une entrée de journal est effectuée pour indiquer qu'un point de reprise a été réalisé.
- Toutes les pages du cache sont transférées sur le périphérique de journal.
- Périodiquement en fonction du temps (ex toutes 5 minutes) on effectue le checkpoint,
- Les transactions validées contenues dans le Log sont effectivement écrites en base, celles non validées restent dans le log.
- Cas de la reprise automatique : cas du redémarrage.
- Les transactions validées depuis le dernier point de reprise sont relancées (rolled forward)
- Les transactions qui n'ont pas pu être validées avant l'arrêt sont invalidées.
- Verrouillage : limite l'accès aux données lors de modification il est géré par SQLServer.
- Il est largement utilisé pour éviter tout conflit entre les transactions s'exécutant simultanément.
- Verrouillage de page : Un bloc de données est de 2Ko cela constitue la plus petite unité pouvant être lue ou écrite sur disque

`begin <tran | transaction> <nameOfTransaction>`

`commit <tran nameOfTransaction >`

`rollback <tran >`

Etats des données avant COMMIT ou ROLLBACK

- Comportement standard:
- L'état précédent de la base peut être récupéré,
- L'utilisateur courant peut voir le résultat de ses mises à jour de données,
- Les autres utilisateurs ne peuvent généralement pas voir les mises à jour effectuées par l'utilisateur courant,
- Certaines données peuvent être verrouillées,
- Les autres utilisateurs ne peuvent modifier les données verrouillées.

Après le COMMIT

- Les modifications de la base sont permanentes,
- L'état précédent de la base est définitivement perdu,
- Tous les utilisateurs peuvent voir le résultat des opérations de DML,
- Les verrous sont relâchés,
- Tous les points de sauvegarde sont effacés.

Validation de transaction partielle

- On peut découper une transaction en créant des points d'arrêts (savepoint).
- SAVEPOINT S1;
- Ces points d'arrêts permettent d'annuler seulement une partie de la transaction. La commande ROLLBACK permet de retourner à un point d'arrêts:
- ROLLBACK TO S1;

Intégrité référentielle DECLENCHEUR>> TRIGGERS

- ❖ Cas des tables liés par des clés primaires et donc étrangères dans les modèles, si une clé primaire est insérée (inserted) elle doit être unique.
 - ❖ S'il existe des clés étrangères >> ne pas autoriser la suppression de la valeur clé primaire.
 - ❖ Suppression (deleted) d'une clé primaire alors supprimer les étrangères y faisant référence.
 - ❖ Modification d'une clé primaire alors modification de toutes celles y faisant référence.
- Forme évoluée de règles utilisés pour renforcer l'intégrité de la base de données, particulièrement l'intégrité référentielle. Le plus souvent on parle de **TRIGGERS** :

Les triggers sont stockées avec les données dans la base.

- Les triggers sont déclenchés automatiquement par le noyau SQL à chaque intervention sur la table qui les supportent, pas de possibilités de passer outre quelque soit l'utilisateur.

Principes :

- Les triggers sont un type particulier de procédure mémorisée.
- Un trigger est toujours associé à une table, il est impossible de l'associer à une vue ou à une table temporaire.
- Une table peut avoir au maximum trois triggers, chacun étant déclenché en fonction de l'événement détecté sur la table :
 - Insertion de ligne, modification, suppression de ligne.
- Chaque trigger est facultatif et une table peut n'avoir aucun trigger
- D'autre part, un trigger ne s'applique qu'à une seule table.
- La suppression d'une table entraîne la destruction de ses triggers

Principe de fonctionnement :

- 2 tables virtuelles sont créées au moment de la MAJ sur la table: INSERTED, DELETED.
- Elles sont destinées à contenir les lignes de la table sur lesquelles ont été effectuées des opérations.
- Les tables INSERTED et DELETED peuvent être utilisées par le trigger pour déterminer comment le traitement doit se dérouler.
- Ce traitement est à écrire par le développeur.
- Les mécanismes d'utilisation par SQL Server de ces deux tables diffèrent en fonction de l'opération effectuée sur la table contenant le ou les triggers.

Cas de suppression d'une ligne de table (delete)

- La/les lignes supprimées sont placées dans la table temporaire DELETED et supprimées de la table réelle;
- La table DELETED et les tables de la base ne peuvent pas avoir de lignes en commun.

Cas de création d'une ligne de table (insert)

- La/les lignes nouvelles sont placées dans la table temporaire INSERTED et dans la table réelle;
- toutes les lignes de la table INSERTED apparaissent dans la table de la base.

Cas de modification d'une ligne de table (update)

- La/les lignes avant modification sont placées dans la table temporaire DELETED et
- La/les lignes après modification sont placées dans la table temporaire INSERTED et dans la table réelle

CREATION D'UN TRIGGER

Un trigger est un **objet SQL Server**, en conséquence de quoi :

- Les triggers seront gérés de manière autonome
- Chaque trigger porte un nom
- Chaque trigger possède un certain nombre de propriétés.
- Les triggers doivent être créés juste après la création de la table de sorte que l'intégrité référentielle des données soit assurée.

La création d'un trigger peut se faire de manière interactive grâce à l'interface graphique, en utilisant les procédures de SQL Server. Utiliser la procédure **create trigger** :

*CREATE TRIGGER nom de trigger ON nom de table FOR INSERT AS séquence1
d'ordres SQL FOR UPDATE AS séquence2 d'ordres SQL FOR DELETE AS séquence3 d'ordres SQL CREATE
TRIGGER nom de trigger ON nom de table FOR INSERT, UPDATE AS
séquence d'ordres SQL*

Les triggers définis sur la base de données ont 2 rôles principaux: **Contrôler** que les données manipulées **vérifient** l'intégrité référentielle, c'est-à-dire que les tables restent synchronisées.

LES TRIGGERS EN CASCADE

Une opération sur une table déclenche automatiquement, s'il existe, le trigger correspondant à l'opération. Si le trigger déclenché effectue une opération sur une autre table, les triggers associés à cette table sont alors déclenchés. C'est ce que l'on qualifie de **cascade**. Le nombre de niveaux en cascade est fonction du SGBD (16 en SQL Server).

CREATE [OR REPLACE] TRIGGER nom-trigger
BEFORE | AFTER
INSERT OR UPDATE [OF column] OR DELETE ON nom-table
[FOR EACH ROW [WHEN (condition)]]
bloc d'instructions pl/sql

–INSERT OR UPDATE [OF column] OR DELETE ON nom-table
Précise le ou les événements provoquant l'exécution du trigger

–BEFORE | AFTER
Spécifie l'exécution de la procédure comme étant avant(BEFORE) ou après (AFTER) l'exécution de l'événement déclencheur
BEFORE : vérification de l'insertion
AFTER : copie ou archivage des opérations

–[FOR EACH ROW]
Indique que le trigger est exécuté pour chaque ligne modifiée/insérée/supprimée
Si cette clause est omis, le trigger est exécuté une seul fois pour chaque commande UPDATE, INSERT, DELETE, quelque soit le nombre de lignes modifiées, insérées ou supprimées.

–[WHEN (condition)]
conditions optionnelles permettant de restreindre le déclenchement du trigger !!! Ne peut contenir de requêtes (exemple: WHEN emp.sal > 5000)