

Secteur Tertiaire Informatique  
Filière « Etude et développement »

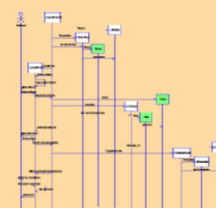
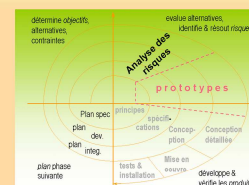
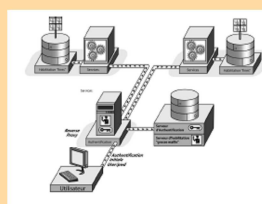
Séquence « Développer des composants dans le langage de la base de données »

Mise en œuvre des Transactions  
SQL Server

Apprentissage

Mise en pratique

Evaluation





# TABLE DES MATIERES

Table des matières .....	3
1 Transaction.....	5
1.1 Syntaxe relative aux transactions.....	6
1.1.1 Exemple sans transaction.....	6
1.1.2 Exemple avec transaction explicite .....	7
2 Gestion des verrouillages .....	8
2.1 Principe du verrouillage transactionnel.....	8
2.2 Verrous et Niveau d'isolation des transactions .....	9
2.2.1 Un exemple pour comprendre.....	9
2.2.2 Les verrous en bref.....	11
2.2.3 Niveau d'isolation des transactions .....	12
3 Le journal des transactions .....	14
3.1 Les modes de récupération.....	15
3.1.1 Mode de récupération simple.....	15
3.1.2 Mode de restauration complète et de récupération utilisant les journaux de transactions.....	15

## Objectifs

Ce document présente les principes des transactions et des verrous proposés par SQL Server pour assurer l'intégrité des données lors des mises à jour d'informations.

## Pré requis

Maîtriser les requêtes SQL et les bases de la programmation en langage Transact-SQL.

## Outils de développement

SQL Server (2008 au plus).

## Méthodologie

## Mode d'emploi

Symboles utilisés :



Renvoie à des supports de cours, des livres ou à la documentation en ligne constructeur.



Propose des exercices ou des mises en situation pratiques.



Point important qui mérite d'être souligné !

## Ressources

## Lectures conseillées

# 1 TRANSACTION

Une transaction est une unité logique de traitements regroupant un ensemble de commandes SQL élémentaires. Ces opérations doivent être soit exécutées entièrement, soit pas du tout, permettant ainsi à la base de données de passer d'un état cohérent à un nouvel état cohérent. Les transactions sont donc un moyen d'assurer l'intégrité des données lors des mises à jour d'informations.

SQL comporte un moteur transactionnel, une transaction correspondant à la modification d'une donnée. On parle, pour évoquer un système transactionnel, de système OLTP (On Line Transaction Processing).

Les transactions peuvent être implicites ou déclarées explicitement afin de recouvrir un **ensemble de modifications solidaires**.

La définition explicite de transactions permet de s'assurer de la complétude d'une demande de modification pouvant comporter plusieurs opérations de modification sur plusieurs lignes de plusieurs tables.

Il est ainsi possible de garantir une bonne cohérence à la base de données.

Pour expliciter ce propos, prenons l'exemple d'une transaction bancaire :

Elle est toujours composée du débit d'un compte et du crédit d'un autre compte. Il convient donc de s'assurer que les deux opérations ont bien été effectuées. Si l'une des deux opérations n'aboutit pas, nous aurons alors un système « bancal ». Si nous ne spécifions pas explicitement que ces deux opérations constituent une seule transaction, le risque d'erreur demeure.

Les données d'une transaction sont validées dans leur ensemble par l'application d'un ordre **COMMIT**, ou invalidées (elles reprennent alors leur état initial) par l'ordre **ROLLBACK**.

On appelle ce mécanisme la validation en deux temps.

La validation transactionnelle s'appuie sur le journal des transactions associé à toute base de données.

Une transaction est caractérisée par les critères **ACID**

- **Atomique** : si une des instructions échoue, toute la transaction échoue
- **Cohérente**, car la base de données est dans un état cohérent avant et après la transaction, c'est-à-dire respectant les règles de structuration énoncées.
- **Isolée** : Les données sont verrouillées : il n'est pas possible depuis une autre transaction de visualiser les données en cours de modification dans une transaction.
- **Durable** : les modifications apportées à la base de données par une transaction sont validées

## 1.1 SYNTAXE RELATIVE AUX TRANSACTIONS

BEGIN TRAN ou BEGIN TRANSACTION marque le point de référence du début de la transaction. La transaction peut éventuellement être nommée.

Les instructions émises dans le cadre la transaction sont verrouillées par le système et l'accès aux données sous-jacentes restreint pour les autres connexions.

Les données sont déverrouillées lors de l'exécution :

- d'un ordre de validation (application des modifications) COMMIT
- d'un ordre d'invalidation (retour version précédente des données) ROLL BACK ou relatif à une erreur système.

Les transactions peuvent être imbriquées.

Les transactions peuvent être nommées et être ainsi référencées plus facilement. Toutefois dans le cadre de transactions imbriquées, seule la transaction la plus externe peut être nommée.

L'utilisation de la clause WITH MARK ['description'] lors de la déclaration d'une transaction indique qu'elle est marquée dans le journal des transactions. Si WITH MARK est utilisé, un nom de transaction doit être spécifié.

WITH MARK permet de restaurer un journal de transactions par rapport à une marque nommée.

L'exemple suivant illustre l'intérêt et les conditions de mise ne place d'une transaction. Il s'agit ici de l'insertion de lignes de commandes dans la table [Order details] du comptoir anglais.

La clé primaire de la table [Order Details] est constituée de deux colonnes [OrderID] et [ProductID].

Deux lignes d'une même commande ne peuvent donc pas porter sur le même article.

### 1.1.1 Exemple sans transaction

Dans cet exemple, 3 lignes seront insérées et 1 rejetée.

Nous nous trouverons donc avec une commande dans le système non conforme à l'originale.

```
INSERT INTO [Order Details] ([OrderID], [ProductID], [UnitPrice], [Quantity], [Discount])
VALUES (11099, 1, 10,10, 0.2)
INSERT INTO [Order Details] ([OrderID], [ProductID], [UnitPrice], [Quantity], [Discount])
VALUES (11099, 2, 10,10, 0.2)
INSERT INTO [Order Details] ([OrderID], [ProductID], [UnitPrice], [Quantity], [Discount])
VALUES (11099, 3, 10,10, 0.1)
INSERT INTO [Order Details] ([OrderID], [ProductID], [UnitPrice], [Quantity], [Discount])
VALUES (11099, 2, 10,10, 0.1)
```

Figure 1 : Insertion de lignes de commandes

(1 ligne(s) affectée(s))

(1 ligne(s) affectée(s))

(1 ligne(s) affectée(s))

Serveur : Msg 2627, Niveau 14, État 1, Ligne 1

Violation de la contrainte PRIMARY KEY 'PK\_Order\_Details'.

Impossible d'insérer une clé en double dans l'objet 'Order Details'.

L'instruction a été arrêtée.

	OrderID	ProductID	UnitPrice	Quantity	Discount	TS
1	11099	1	10.0000	10	0.2	0x00000000000005AB7
2	11099	2	10.0000	10	0.2	0x00000000000005AB8
3	11099	3	10.0000	10	0.1	0x00000000000005AB9

Figure 2 : Messages et résultat

### 1.1.2 Exemple avec transaction explicite

Dans cet exemple, la différence réside dans le fait qu'aucune ligne ne sera insérée en cas de problèmes. La cohérence des informations reste ainsi assurée.

```
P-BOSTON\SQL... - TRAN-01.sql  Résumé
USE COMPTOIRANGLAIS
GO
-- Suppression des lignes existantes
SET NOCOUNT ON
DELETE FROM DBO.[ORDER DETAILS] WHERE ORDERID = 11099
SET NOCOUNT OFF
BEGIN TRAN
BEGIN TRY
INSERT INTO [Order Details] (OrderID,ProductID,UnitPrice,Quantity,Discount)
VALUES (11099,1,10,10,0.2)
INSERT INTO [Order Details] (OrderID,ProductID,UnitPrice,Quantity,Discount)
VALUES (11099,2,10,10,0.2)
INSERT INTO [Order Details] (OrderID,ProductID,UnitPrice,Quantity,Discount)
VALUES (11099,3,10,10,0.2)
INSERT INTO [Order Details] (OrderID,ProductID,UnitPrice,Quantity,Discount)
VALUES (11099,1,10,10,0.2)
COMMIT TRAN
END TRY
BEGIN CATCH
ROLLBACK TRAN
END CATCH
SELECT * FROM dbo.[Order Details] Where OrderID = 11099
```

Figure 3 : Recours aux transactions explicites

## 2 GESTION DES VERROUILLAGES

### 2.1 PRINCIPE DU VERROUILLAGE TRANSACTIONNEL

Le système recourt à la mise en place de verrous pour réduire les préjudices susceptibles d'affecter les ressources en cours de modification.

Ainsi, lorsqu'un utilisateur effectue des transactions sur une ressource, SQL Server n'autorise pas d'autres utilisateurs à effectuer sur cette ressource des opérations qui nuiraient aux dépendances de l'utilisateur détenteur du verrou. Les verrous sont gérés de manière interne par le logiciel système et sont placés et libérés en fonction des actions entreprises par l'utilisateur.

Il s'agit là d'un système complexe que nous n'aborderons pas en détail mais dont il faut comprendre les enjeux et les conséquences éventuelles sur le niveau de performance de votre système.

Les verrous peuvent être appliqués à différents niveaux de la base de données depuis la ligne (niveau le plus fin) jusqu'à la table voire la base de données dans son ensemble (par exemple lors d'une opération de restauration, un verrou exclusif est mis en place sur la base de données).

Vous pouvez modifier la nature des verrous et leur impact, en modifiant le niveau d'isolation des transactions. Mais avant d'aborder ce sujet, nous allons, par un schéma, visualiser une situation qui illustre les situations de blocages engendrés par ces mécanismes de verrouillage et les demandes d'accès concurrentielles à une même ressource.

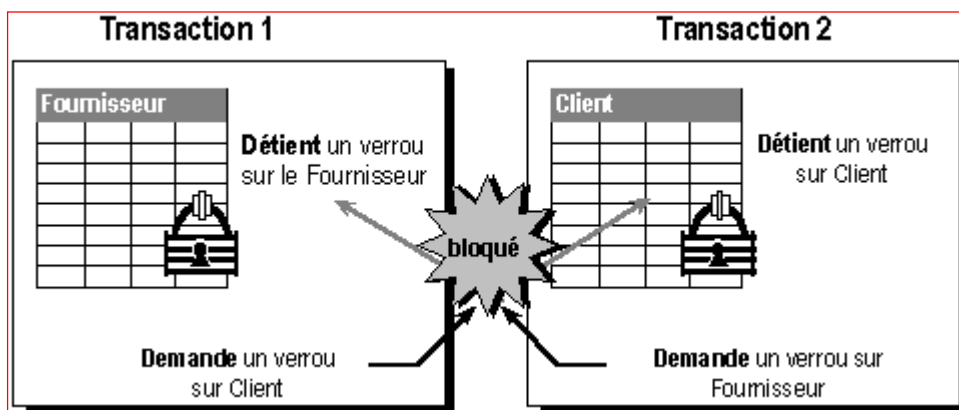


Figure 4 : Exemple de blocage sur accès concurrents à une même ressource

La transaction 1 a posé un verrou sur le fournisseur et demande un verrou sur le client sur laquelle la transaction 2 est déjà détentrice d'un verrou et qui demande à son tour un verrou sur le fournisseur. Il s'agit là d'une situation où les deux demandes se bloquent : on parle d'étreinte fatale...Le système peut mettre fin à cette situation de blocage en annulant la requête du thread victime du verrou (déterminée par des règles complexes visant à tuer le « plus faible »...): on parle alors de DEADLOCK.



## 2.2 VERROUS ET NIVEAU D'ISOLATION DES TRANSACTIONS

Il existe plusieurs modes de verrouillage : partagé, mise à jour et exclusif. Le mode de verrouillage indique le niveau de dépendance de la connexion sur l'objet verrouillé.

SQL Server contrôle l'interaction des modes de verrouillage. Par exemple, il est impossible d'obtenir un verrou exclusif si d'autres connexions disposent de verrous partagés sur la ressource.

Les verrous sont maintenus le temps nécessaire pour protéger la ressource au niveau demandé mais la durée des verrous partagés utilisés pour protéger les lectures dépend des niveaux d'isolement de la transaction.

Les verrous exclusifs utilisés pour protéger les mises à jour sont maintenus jusqu'à la fin de la transaction.

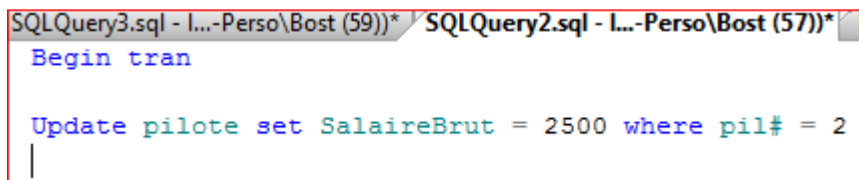
Si une connexion essaye d'appliquer un verrou qui entre en conflit avec un verrou placé par une autre connexion, elle est bloquée jusqu'à ce que le verrou en conflit soit libéré et que la connexion applique le verrou demandé ou si le délai d'attente de la connexion expire.

Par défaut, il n'y a pas de délai d'attente, mais certaines applications en définissent un pour éviter d'attendre indéfiniment.

L'attribution des verrous se fait selon la règle du premier arrivé premier servi.

### 2.2.1 Un exemple pour comprendre

La première transaction demande la mise à jour du salaire d'un pilote.



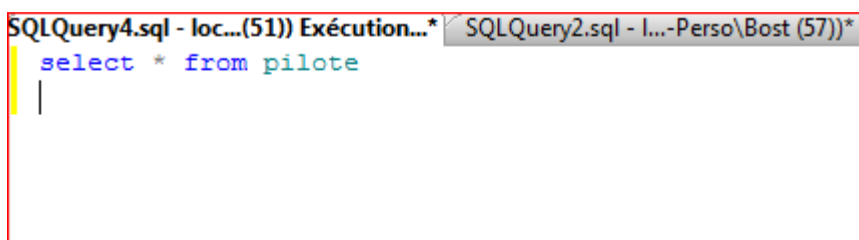
```
SQLQuery3.sql - I...-Perso\Bost (59))* SQLQuery2.sql - I...-Perso\Bost (57))*  
Begin tran  
  
Update pilote set SalaireBrut = 2500 where pil# = 2  
|
```

Figure 5 : Première transaction : verrou exclusif / MAJ

Une deuxième transaction demande la liste des pilotes.

Cette demande reste sans effet...

Elle ne sera exécutée que lors de la libération du verrou posé sur la table pilote par la première transaction.



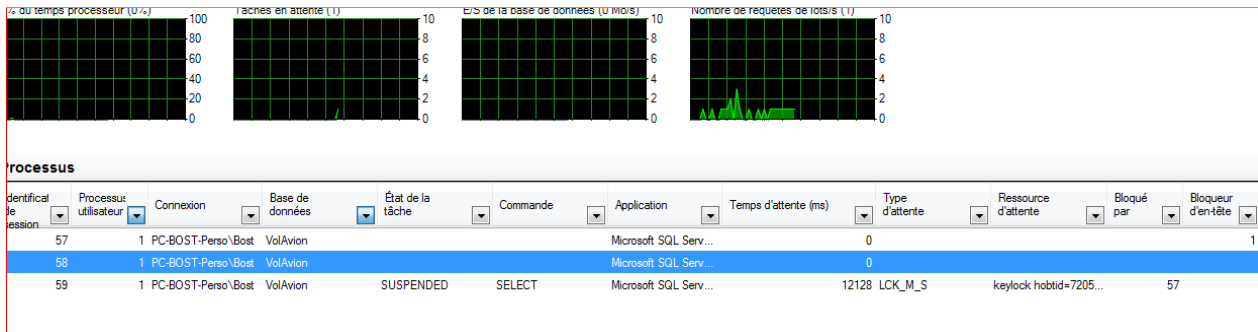
```
SQLQuery4.sql - loc...(51)) Exécution...* SQLQuery2.sql - I...-Perso\Bost (57))*  
select * from pilote  
|
```

Figure 6 : Demande extraction bloquée par Transaction 1

Par l'intermédiaire de l'onglet Moniteur d'activité du logiciel Enterprise Manager, présent dans la barre des tâches standard :

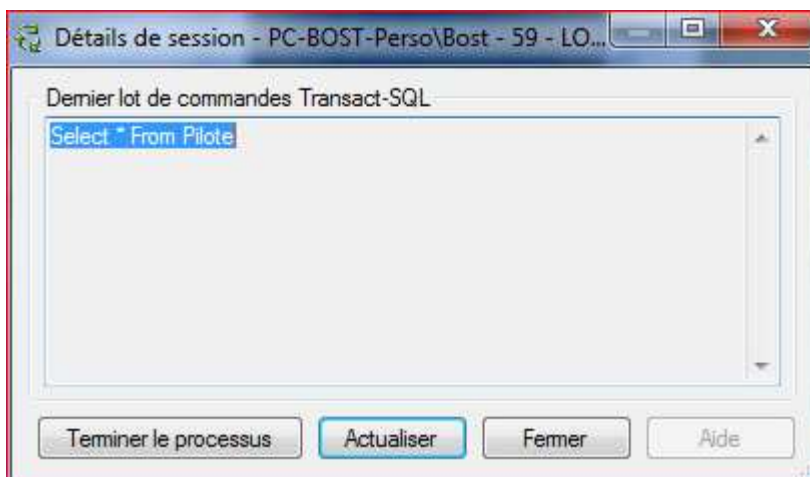


Nous pouvons comprendre la nature des opérations en cours et des verrous posés par le processus exécutant la requête de mise à jour.



État de la tâche	Commande	Application	Temps d'attente (ms)	Type d'attente
		Microsoft SQL Serv...	0	
		Microsoft SQL Serv...	0	
SUSPENDED	SELECT	Microsoft SQL Serv...	142230	LCK_M_S

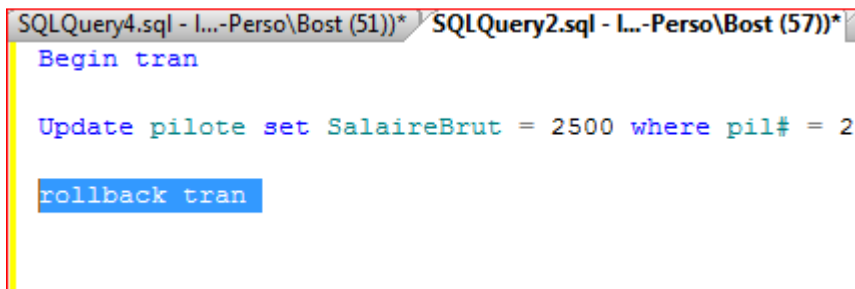
Nous pouvons aussi avoir le détail des commandes exécutées.



Que faut-il faire pour débloquent la situation et permettre à la requête d'extraction de s'exécuter ?

Mais c'est bien sûr ! Terminer la transaction 1 par un COMMIT !

Après exécution du COMMIT TRAN, l'extraction est réalisée :

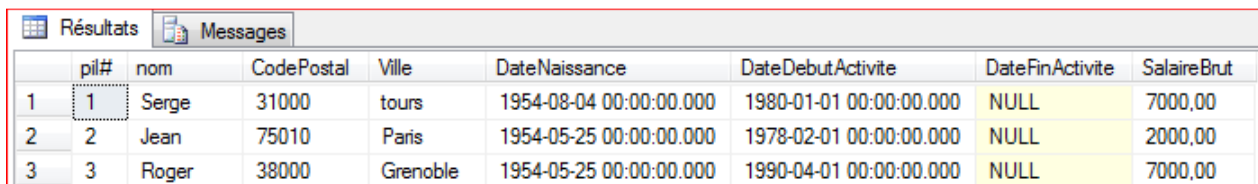


```
SQLQuery4.sql - I...-Perso\Bost (51))* SQLQuery2.sql - I...-Perso\Bost (57))*
Begin tran

Update pilote set SalaireBrut = 2500 where pil# = 2

rollback tran
```

Figure 7 : Fin de Transaction 1 – abandon par Rollback



	pil#	nom	CodePostal	Ville	DateNaissance	DateDebutActivite	DateFinActivite	SalaireBrut
1	1	Serge	31000	tours	1954-08-04 00:00:00.000	1980-01-01 00:00:00.000	NULL	7000,00
2	2	Jean	75010	Paris	1954-05-25 00:00:00.000	1978-02-01 00:00:00.000	NULL	2000,00
3	3	Roger	38000	Grenoble	1954-05-25 00:00:00.000	1990-04-01 00:00:00.000	NULL	7000,00

Figure 8 : La deuxième transaction est libérée et la liste réalisée

**A noter** : il est donc essentiel de ne pas maintenir des transactions trop longues ou qui exigent une intervention de l'utilisateur, comme c'est le cas ici ! Il s'agit évidemment d'un cas d'école...

### 2.2.2 Les verrous en bref

Les verrous empêchent les conflits de mise à jour :

- Ils permettent la sérialisation des transactions de façon à ce qu'une seule personne à la fois puisse modifier un élément de données ; dans un système d réservation de places, les verrous garantissent que chaque place n'est attribuée qu'à une seule personne.
- SQL Server définit et ajuste de manière dynamique le niveau de verrouillage approprié pendant une transaction : il est également possible de contrôler manuellement le mode d'utilisation de certains verrous.
- Les verrous sont nécessaires aux transactions concurrentes pour permettre aux utilisateurs d'accéder et de mettre à jour les données en même temps.

Le contrôle de la concurrence permet de s'assurer que les modifications apportées par un utilisateur ne vont pas à l'encontre de celles apportées par un autre.

- – Le contrôle pessimiste verrouille les données qui sont lues en vue d'une mise à jour, empêchant tout autre utilisateur de modifier ces données
- – Le contrôle optimiste ne verrouille pas les données : si les données ont été modifiées depuis la lecture, un message d'annulation est envoyé à l'utilisateur.

Le choix entre ces deux types de contrôle est effectué en fonction de l'importance du risque de conflit de données et de l'évaluation du coût de verrouillage des données (nombre d'utilisateurs, nombre de transactions, temps de réponse ...)

Le type de contrôle sera explicité en spécifiant le niveau d'isolement de la transaction.

### 2.2.3 Niveau d'isolation des transactions

Lors de transactions concurrentes, des verrous peuvent être posés pour éviter les situations suivantes :

- – .Mise à jour perdue : une mise à jour peut être perdue lorsqu'une transaction écrase les modifications effectuées par une autre transaction
- – Lecture incorrecte : Se produit lorsqu'une transaction lit des données non validées provenant d'une autre transaction
- – Lecture non renouvelable : Se produit lorsque, une transaction devant lire la même ligne plusieurs fois, la ligne est modifiée par une autre transaction et donc produit des valeurs différentes à chaque lecture
- – Lectures fantômes : Se produit lorsqu'une autre transaction insert une ligne au sein de la transaction en cours

Un verrou partagé appliqué à une ressource par une première transaction autorise l'acquisition par une deuxième transaction d'un verrou partagé sur cette ressource, même si la première transaction n'est pas terminée. Un verrou partagé sur une ligne est libéré dès la lecture de la ligne suivante.

SQL Server utilise des verrous exclusifs pour les modifications de données (INSERT, UPDATE, DELETE). Une seule transaction peut acquérir un verrou exclusif sur une ressource ; une transaction ne peut pas acquérir un verrou exclusif sur une ressource tant qu'il existe des verrous partagés ; une transaction ne peut pas acquérir un verrou partagé sur une ressource déjà pourvue d'un verrou exclusif.

SQL Server permet de définir un niveau d'isolement qui protège une transaction vis-à-vis des autres.

La syntaxe est la suivante

```
SET TRANSACTION ISOLATION LEVEL {READ COMMITTED | READ UNCOMMITTED |  
REPEATABLE READ | SNAPSHOT | SERIALIZABLE}
```

## **Read UnCommitted**

Indique à SQL Server de ne pas placer de verrous partagés : une transaction peut lire des données modifiées non encore validées par une autre transaction.

Des lectures incorrectes peuvent se produire.

## **Read Committed** (Option par défaut)

Indique à SQL Server d'utiliser des verrous partagés pendant la lecture : une transaction ne peut pas lire les données modifiées mais non validées d'une autre transaction.

La lecture incorrecte ne peut pas se produire.

## **Repeatable Read**

SQL Server place des verrous partagés sur toutes les données lues par chaque instruction de la transaction et les maintient jusqu'à la fin de la transaction : une transaction ne peut pas lire des données modifiées mais pas encore validées par une autre transaction, et ne peut modifier les données lues par la transaction active tant que celle-ci n'est pas terminée.

La lecture incorrecte et la lecture non renouvelable ne peuvent pas se produire.

## **Snapshot**

Une transaction n'a pas accès aux modifications de données apportées par une autre transaction : les données vues par la première transaction sont les données antérieures au début de la deuxième transaction.

## **Serializable**

Une transaction ne peut pas lire des données modifiées mais pas encore validées par une autre transaction, et ne peut modifier les données lues par la transaction active tant que celle-ci n'est pas terminée.

Une transaction ne peut pas insérer de nouvelles lignes avec des valeurs de clés comprises dans le groupe de clés lues par des instructions de la transaction active, tant que celle-ci n'est pas terminée.

La lecture incorrecte la lecture non renouvelable et les lectures fantômes ne peuvent pas se produire.

## En conclusion :

Le niveau d'isolement spécifie le comportement de verrouillage par défaut de toutes les instructions de la session (connexion).

Plus le niveau d'isolement est élevé, plus les verrous sont maintenus longtemps et plus ils sont restrictifs.

La commande DBCC USEROPTIONS renvoie, entre autres informations, le niveau d'isolement de la session.

Il est possible de définir la durée maximale pendant laquelle SQL Server permet à une transaction d'attendre le déverrouillage d'une ressource bloquée.

La variable globale @@lock\_timeout donne le temps d'attente défini.

SET LOCK\_TIMEOUT délai\_attente positionne le délai, en millisecondes, avant que ne soit renvoyé un message d'erreur (-1 indique qu'il n'y a pas de délai – par défaut)

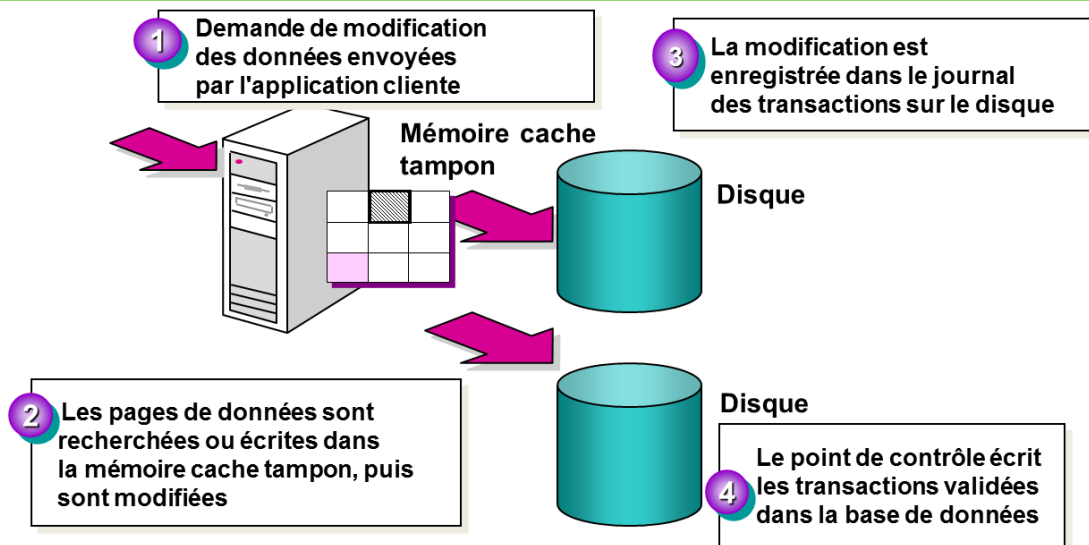
### 3 LE JOURNAL DES TRANSACTIONS

Le journal des transactions enregistre les modifications de données au fur et à mesure qu'elles sont effectuées.

Lorsqu'une modification de données est envoyée par l'application, si les pages de données concernées n'ont pas été chargées en mémoire cache à partir du disque lors d'une précédente modification, elles sont alors amenées en mémoire.

Chaque instruction est enregistrée dans le journal et la modification est effectuée en mémoire.

#### Principe de la validation transactionnelle



04/02/2016

Le processus de point de contrôle reporte périodiquement dans la base de données toutes les modifications effectuées

A un instant  $t$ , il existe dans le journal :

1. des transactions validées avant le dernier point de contrôle et donc reportées dans la base
2. des transactions validées après le dernier point de contrôle, et donc non reportées dans la base
3. des transactions non validées

Les conditions 1 et 2 n'existent que si vous avez choisi un mode de récupération de données autre que simple.

Dans ce cas, en cas de défaillance du système, le processus de restauration automatique utilise le journal des transactions pour reprendre toutes les transactions validées non reportées dans la base, et annuler les transactions non validées.

### 3.1 LES MODES DE RECUPERATION

#### 3.1.1 Mode de récupération simple

Le mode de récupération simple réduit les tâches d'administration du journal des transactions car celui-ci n'est pas sauvegardé. Il entraîne un risque de perte de travail assez élevé en cas d'endommagement de la base de données. Les données ne sont récupérables que jusqu'à la sauvegarde la plus récente des données perdues. Par conséquent, en mode de récupération simple, les intervalles de sauvegarde doivent être suffisamment courts pour empêcher la perte d'un volume significatif de données. Toutefois, les intervalles doivent être suffisamment longs pour que la charge de sauvegarde n'affecte pas la production. L'insertion de sauvegardes différentielles dans la stratégie de sauvegarde permet de réduire la surcharge.

En règle générale, pour une base de données utilisateur, le mode de récupération simple est utile pour les bases de données de test et de développement ou pour celles contenant essentiellement des données accessibles en lecture seule, telles qu'un Data Warehouse.

Le mode de récupération simple ne convient pas aux systèmes de production où la perte de récentes modifications est inacceptable. Dans ce cas, nous recommandons l'utilisation du mode de restauration complète.

#### 3.1.2 Mode de restauration complète et de récupération utilisant les journaux de transactions

Le mode de restauration complète et le mode de récupération utilisant les journaux de transactions offrent beaucoup plus de protection pour les données que le mode de récupération simple. Ces deux modes dépendent de la sauvegarde du journal des transactions pour permettre une récupération complète et empêcher une perte de travail dans le plus grand nombre possible de scénarios d'échec.

##### **Mode de restauration complète**

Fournit le mode de maintenance de base de données normal pour les bases de données où la durabilité des transactions est nécessaire.

Des sauvegardes du journal sont requises. Ce mode consigne complètement toutes les transactions et conserve les enregistrements du journal des transactions après leur sauvegarde. Le mode de restauration complète permet de récupérer une base de données jusqu'au moment de la défaillance, en partant du principe que la fin du journal peut être sauvegardée après la défaillance. Le mode de restauration complète prend également en charge la restauration des pages de données individuelles.

##### **Mode de récupération utilisant les journaux de transactions**

Ce mode de récupération consigne en bloc la plupart des opérations en bloc. Il a été conçu uniquement comme complément au mode de restauration complète.

Pour certaines opérations en bloc à grande échelle, telles qu'une importation en bloc ou une création d'index, le basculement temporaire en mode de récupération utilisant les journaux de transactions augmente les performances et réduit la consommation d'espace du journal.

Des sauvegardes du journal sont encore requises. Comme le mode de restauration complète, le mode de récupération utilisant les journaux de transactions conserve les enregistrements du journal des transactions après leur sauvegarde.

En revanche, il génère des sauvegardes du journal plus volumineuses et un risque accentué de perte de travail, car le mode de récupération utilisant les journaux de transactions ne prend pas en charge la récupération dans le temps. Pour plus d'informations, consultez Sauvegarde avec le mode de récupération utilisant les journaux de transactions.

Il s'agit de programmes déclenchés automatiquement lors d'opérations de mises à jour sur une table.



## **CREDITS**

### **ŒUVRE COLLECTIVE DE l'AFPA**

**Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services**

#### **Equipe de conception (IF, formateur, mediatiseur)**

B. Hézard - Formateur

V. Bost - Formateur

Ch. Perrachon – Ingénieure de formation

**Date de mise à jour : 8/02/16**

## **Reproduction interdite**

Article L 122-4 du code de la propriété intellectuelle.

« Toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droits ou ayants cause est illicite. Il en est de même pour la traduction, l'adaptation ou la reproduction par un art ou un procédé quelconque. »

Mise en œuvre des Transactions – SQL Server

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »