Les journées
SQL Server 2013

Un événement organisé par GUSS

#JSS2013

Les journées

SQL Server 2013

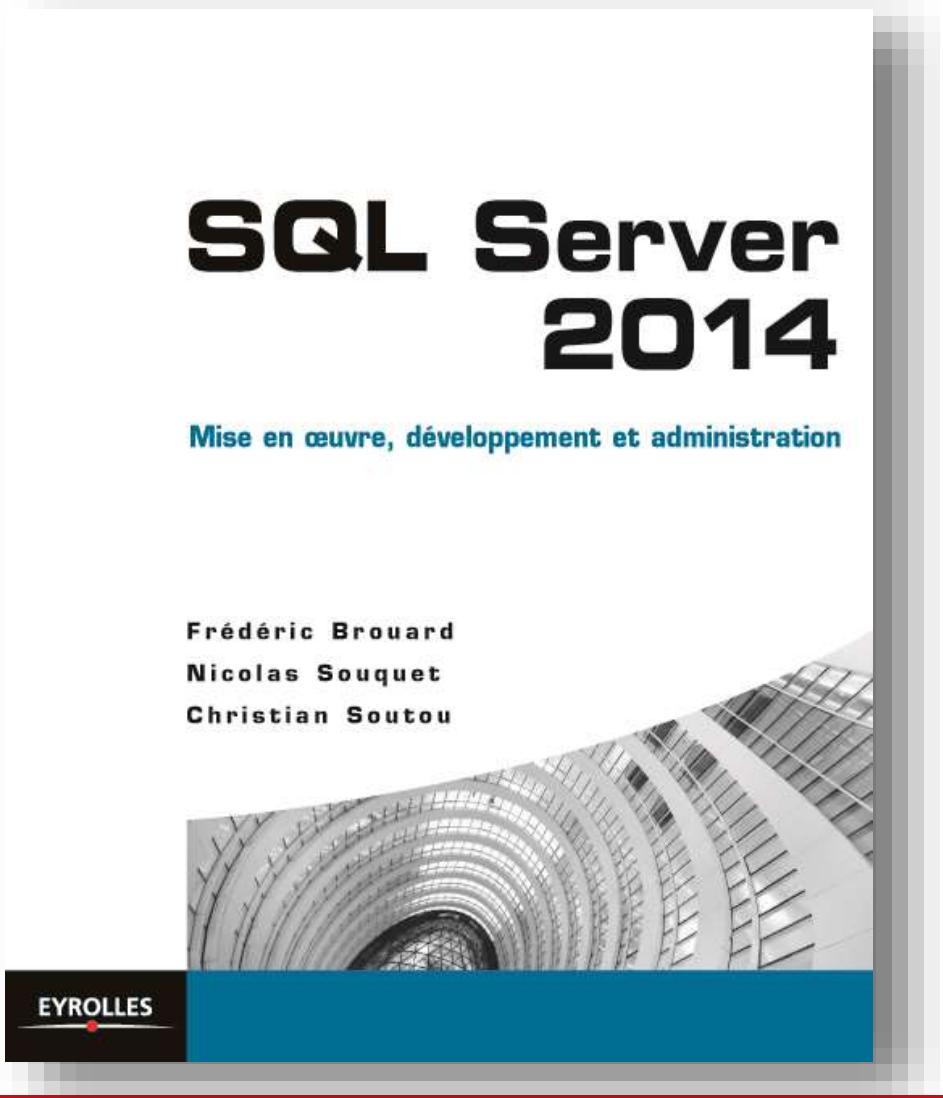
Les Verrous

Arian Papillon, MVP SQL Server
Frédéric Brouard, MVP SQL Server

- Arian Papillon
  - a.papillon@datafly.fr

- Frédéric Brouard
  - sqlpro@sqlspot.com

Bientôt dans les bacs



Merci à nos sponsors

Les journées SQL Server

Agenda

- SGBD et concurrence d'accès
- Qu'est-ce qu'un verrou ?
- Transactions et niveaux d'isolation
- L'étreinte fatale ou verrou de la mort
- Diagnostiquer les verrous, attentes, blocages
- Limiter la contention et les deadlocks

SGBD : Système Multi-Utilisateurs

- Plusieurs utilisateurs accèdent aux données et les mettent à jour
- Il est nécessaire de garantir la stabilité des données afin que des accès simultanés aux mêmes données n'interfèrent pas…
- Le verrou est le mécanisme de base pour assurer la stabilité des données

Comment gérer les accès concurrents ?

- Pessimiste : blocage à priori
  - Les lectures bloquent les écritures
  - Les écritures bloquent tout
  - Mode par défaut dans SQL Server
- Optimiste : vérification à postériori
  - Basé sur un « versionning » des lignes
  - Aussi pris en charge par SQL Server

Il ne reste plus que 1 exemplaire(s) en stock

Sans verrouillage, que se passe-t-il ?

ANOMALIES TRANSACTIONNELLES :

- Le dernier qui met à jour gagne : les mises à jour précédentes sont écrasées (perte de mise à jour)
- Lors des lectures, si une mise à jour est en cours, il peut y avoir : lectures « sales », lectures non répétables, lignes fantômes…
- Le temps de lire toutes les lignes d'une table, un autre utilisateur peut changer les valeurs de certaines lignes.
  - Exemple : deux produits identiques mis successivement dans un même panier, pourraient avoir deux prix différents !

Qu'est-ce qu'un verrou ?

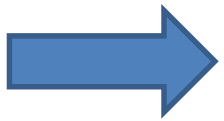
- Le mécanisme utilisé pour contrôler l'accès à une même donnée par de multiples utilisateurs
- Avant qu'une transaction n'effectue une modification, elle doit s'assurer un accès exclusif à la donnée

Qu'est-ce qu'un verrou

- Ressources « verrouillables » et granularité
  - Base de données
  - Table
  - Partition
  - Page
  - Ligne
  - Clé (index)
  - Range de clés
  - *Schéma*
  - *Extension*
  - *Fichier*
- Mode de verrouillage
  - X : exclusif
  - S : partagé
  - U : update
  - I : intention (IS, IX)
  - SIX/SIU/UIX : partagés avec intention
- Durée :
  - Lecture
  - Ordre SQL
  - Transaction

Compatibilité des modes de verrouillage

Verrou à poser



Verrou existant

	IS	S	U	IX	SIX	X
IS						
S						
U						
IX						
SIX						
X						

Blocages et attentes

- Lorsqu'une ressource est verrouillée et qu'une autre transaction nécessite la pose d'un verrou en conflit sur cette même ressource : attente.
- Par défaut, l'attente est infinie…
  - SET LOCK_TIMEOUT configuré pour la session permet l'abandon (et l'erreur 1222)

Escalade de verrous

- Le verrou est une ressource serveur à économiser
  - Coûte environ 100 octets
- SQL Server choisit la granularité de verrouillage adaptée à la requête à exécuter (ligne, page, partition ou table)
- Pendant le traitement, de nombreux verrous sont acquis.
- Au fil de l'exécution, limitation de la charge par escalade (ligne ou page > table…) :
  - Conversion des verrous d'intention
  - Lorsqu'il atteint environ 5000 locks pour une instruction, en cas d'échec tentative tous les 1250 locks (valeurs approximatives)

Démo

Mécanismes logiques

- On peut contrôler la concurrence d'accès, par :
  - Une transaction explicite
  - Le niveau d'isolation des transactions

Transaction

- Concerne toute opération unitaire de lecture ou écriture (implicite)
- Lorsque plusieurs tables sont concernées, les opérations doivent s'effectuer en « tout ou rien » (complètement ou pas du tout – transaction explicite).
- Implicite :
  - Toute commande SQL (DDL, DML, DCL) est une transaction
  - Commit automatique par défaut (SET IMPLICIT_TRANSACTIONS)
- ou explicite :
  - Démarrage par BEGIN TRANSACTION
  - Finalisation par COMMIT ou ROLLBACK

Les anomalies transactionnelles potentielles

- Lectures impropres (sales, *dirty*, chaos…)
- Lecture non répétables
- Lecture fantôme

Les niveaux d'isolation

- Sans contrôle
  - READ UNCOMMITTED
- Pessimiste
  - READ COMMITTED
  - REPEATABLE READ
  - SERIALIZABLE
- Optimiste
  - READ COMMITTED SNAPSHOT
  - SNAPSHOT

Gérer son niveau d'isolation

- Au niveau session :
  - SET TRANSACTION ISOLATION LEVEL
- Au niveau table / ordre SQL :
  - Hints de table
    - NOLOCK | READUNCOMMITTED
    - READCOMMITTED
    - REPEATABLEREAD
    - HOLDLOCK | SERIALIZABLE

```
UPDATE Production.Product WITH (SERIALIZABLE) …
```

Démo

Dead lock : l'étreinte fatale

Interblocage de ressources :

- Concerne au moins 2 transactions simultanées
- Survient si au moins une transaction concerne deux ressources
- L'une des transactions doit écrire

Dead lock : l'étreinte fatale

Comportement :

- Chaque transaction attend la libération des ressources détenues par l'autre
- Détecté par le moniteur de verrouillage
- Annule automatiquement la moins coûteuse
  - SET DEADLOCK_PRIORITY

Démo

Détecter les attentes de verrous

- Wait Stats : statistiques d'attente depuis le démarrage
  - DMV sys.dm_os_wait_stats
- Attentes de verrous > 30 secondes
  - XEvents : SystemHealth
- Activité globale de verrouillage
  - Analyseur de performances (perfmon)
  - Compteurs Lock Requests/s, Lock Waits/s, Lock Wait Time (ms), Number of Deadlocks/s
- Activité en cours
  - sp_lock, sp_who, moniteur d'activité
  - DMV sys.dm_tran_locks
- Captures détaillées des verrous posés et escalades
  - Profiler
  - Sessions XEvents

Détecter les blocages

- Capturer les informations sur les verrous bloquants : **Block Process Report**
  - Configurer le seuil de déclenchement dans les propriétés de l'instance
- Recueil des informations avec :
  - profiler
  - notification d'évènement
  - XEvents

Détecter les deadlocks

- Trace Flags 1204 et 1222
- Capturer les informations sur les Deadlocks
  - Deadlock Graph
- Capture avec :
  - profiler
  - notification d'évènement
  - Xevents : évènement présent dans System Health

Démo

Contrôler la granularité ou l'escalade

- Hints de table
  - PAGLOCK | ROWLOCK | TABLOCK (et UPDLOCK | XLOCK | TABLOCKX)
- Options d'index
  - ALLOW_ROW_LOCKS
  - ALLOW_PAGE_LOCKS
- ALTER TABLE SET (LOCK_ESCALATION = { AUTO | TABLE | DISABLE })
- Laisser un lock IX positionné sur une table

```
BEGIN TRAN
SELECT * FROM TABLE1 WITH (UPDLOCK,SERIALIZABLE) WHERE 1=0
WAITFOR DELAY ... COMMIT
```

- TraceFlags (global ou session) à utiliser avec précaution
  - 1224 : désactive jusqu'à 40% mémoire utilisée, puis réactive
  - 1211 : désactive totalement l'escalade jusqu'à saturation mémoire et erreur

Bonnes pratiques

- Pour limiter la contention et éviter les blocages
  - Faire des transactions les plus courtes possible :
    - Éviter de « sur transactionner »
    - Minimiser le code transactionné
    - Faire les transactions côté serveur (procédures stockées)

  N'oubliez pas que certains verrous ne sont levés qu'en finalisation (COMMIT, ROLLBACK)

Bonnes pratiques

- Pour limiter la contention et éviter les blocages
  - Normalisation :
    - Tables obèses vs tables fines

    Plus une table est « grosse », plus elle concentre de mise à jour

  Corréler avec : sys.dm_db_index_usage_stats / SUM(user_updates)
  - Remodéliser, partitionner (vertical, horizontal).

Bonnes pratiques…

- Pour limiter la contention et éviter les blocages
  - Indexation, optimisation
    - Plans de requête : éviter les parcours de table ou d'index grâce à une indexation efficace

      Les index servent aussi pour les mises à jour !
    - Récrire les requêtes pour diminuer le temps de traitement

Bonnes pratiques…

- Pour limiter la contention et éviter les blocages
  - Écriture du code :
    - Éviter le code non ensembliste :
      - Boucle
      - Curseur
      - Variable table, table temporaires, fonctions table
    - Faire les transactions explicites côté serveur !

Bonnes pratiques…

- Pour limiter la contention et éviter les blocages
  - Diminuer la « force » du verrouillage
    - Descendre le niveau d'isolation global
    - Adapter pour certaines tables
  - Utiliser l'UPDATE pour lecture

Bonnes pratiques…

- Pour limiter le verrou mortel
  - Écriture du code :
    - Placer les mises à jour avant les lectures
    - Séquencer les opérations dans le même ordre logique d'objets

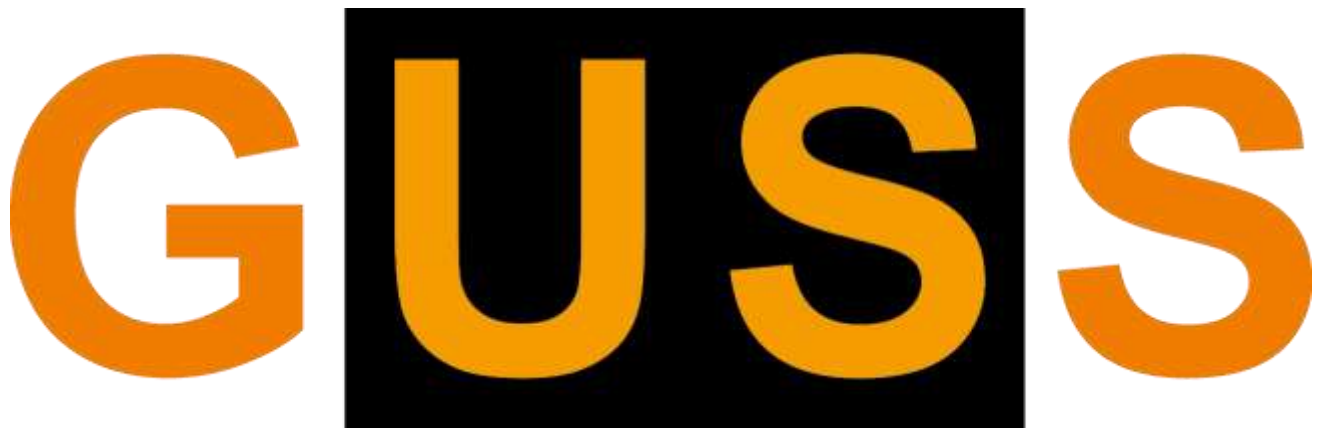
Téléchargez la présentation

- Blog Datafly : http://blog.datafly.pro/post/les-verrous
- Blog SQLPro : http://blog.developpez.com/sqlpro/p/ms-sql-server/les-verrous

- Site GUSS : www.guss.pro

mssql.fr
L'actualité technique MS SQL Server en France (et ailleurs)

Questions & Réponses


GUSS