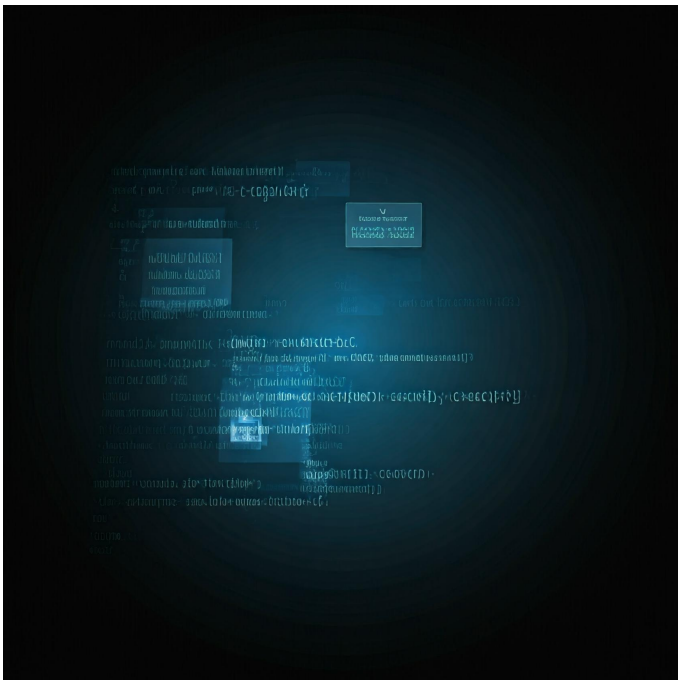
Gestion de la Fragmentation des Indexes et du Fill Factor dans SQL Server



La gestion de la fragmentation des indexes est un aspect fondamental de l'optimisation des performances dans SQL Server. Lorsque les indexes deviennent fragmentés, les opérations de lecture et d'écriture peuvent être ralenties, impactant l'ensemble des performances de la base de données. Un concept clé pour limiter cette fragmentation est le **fill factor**, qui détermine l’espace de remplissage des pages de données lors de la création ou de la reconstruction d'un index.

---

1. La Fragmentation des Indexes

La fragmentation des indexes survient lorsque les pages de données d'un index ne sont plus stockées de manière séquentielle sur le disque. Cela peut se produire pour deux raisons principales :

- **Fragmentation logique** : C’est la séparation de l'ordre logique et physique des pages de données. Lorsqu'un index est fragmenté logiquement, SQL Server doit effectuer plus de lectures pour trouver les données, ce qui ralentit les requêtes.
- **Fragmentation interne** : Elle se produit lorsque des pages de données contiennent de l'espace inutilisé. Elle survient notamment après des opérations d’insertion, de mise à jour ou de suppression, qui laissent des « trous » dans les pages.

La fragmentation peut être mesurée par le pourcentage de désordre d'un index. Plus le pourcentage est élevé, plus l’index est fragmenté, et plus cela nécessite d'opérations de maintenance.

---

2. Le Fill Factor et sa Configuration

Le **fill factor** est un paramètre configuré au moment de la création ou de la reconstruction d'un index. Il spécifie le pourcentage de remplissage de chaque page de données. Par exemple, un fill factor de 80 % signifie que SQL Server laissera 20 % d’espace libre dans chaque page, permettant ainsi d'accueillir de futures insertions sans fragmenter immédiatement les pages.

Les valeurs possibles de fill factor vont de 0 à 100 :

- **0 ou 100 %** : La page est remplie au maximum, sans espace libre. Cela maximise l'utilisation de l'espace disque, mais augmente la probabilité de fragmentation si de nouvelles insertions ou modifications surviennent.
- **1 à 99 %** : L’espace libre est laissé selon le pourcentage spécifié. Un fill factor de 80 %, par exemple, signifie que chaque page sera remplie à 80 %, laissant 20 % de marge pour les futures opérations.

Le choix du fill factor dépend du type de charges et de la nature des données :

- Pour des tables très statiques avec peu de modifications, un fill factor élevé (90–100 %) est souvent optimal.
- Pour des tables avec de nombreuses insertions, mises à jour ou suppressions, un fill factor plus bas (70–80 %) peut être préférable pour limiter la fragmentation.

---

3. Maintenance et Réorganisation des Indexes

Pour gérer la fragmentation, SQL Server offre plusieurs outils de maintenance :

- **Reconstruction des indexes** : Cela recrée les indexes, éliminant la fragmentation. Cependant, cette opération peut être coûteuse en termes de temps et de ressources.
- **Réorganisation des indexes** : Cela réduit la fragmentation sans recréer complètement l'index, en réordonnant les pages. Cette opération est moins intensive que la reconstruction et peut être faite plus fréquemment.

La fréquence de ces opérations de maintenance dépend de la nature de la fragmentation et de la taille de la base de données. Il est conseillé d’identifier les indexes fortement fragmentés et de planifier leur maintenance.

---

4. Le Fill Factor : Un Équilibre entre Espace et Performance

Le fill factor influence directement le nombre de pages de données nécessaires pour stocker un index, ce qui affecte :

- **Les performances en lecture** : Un fill factor bas peut augmenter le nombre de pages et donc les lectures nécessaires.
- **Les performances en écriture** : Avec un fill factor plus bas, la fragmentation est réduite et les insertions sont plus rapides, mais l’index utilise davantage d'espace disque.

Trouver le bon équilibre nécessite souvent des tests pour chaque environnement. Parfois, il est recommandé d'ajuster le fill factor de manière dynamique pour s'adapter aux variations de la charge de travail.

---

5. Mesurer et Suivre la Fragmentation des Indexes

SQL Server propose des vues système telles que `sys.dm_db_index_physical_stats`, qui permet d'obtenir des informations sur la fragmentation des indexes et d'ajuster la maintenance en conséquence. Voici un exemple de requête :

```
SELECT
    dbschemas.[name] AS 'Schema',
    dbtables.[name] AS 'Table',
    dbindexes.[name] AS 'Index',
    indexstats.avg_fragmentation_in_percent
FROM
    sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, NULL, 'DETAILED') AS indexstats
JOIN
    sys.tables dbtables ON dbtables.[object_id] = indexstats.[object_id]
JOIN
    sys.schemas dbschemas ON dbtables.[schema_id] = dbschemas.[schema_id]
JOIN
    sys.indexes AS dbindexes ON dbindexes.[object_id] = indexstats.[object_id]
    AND indexstats.index_id = dbindexes.index_id
ORDER BY
    indexstats.avg_fragmentation_in_percent DESC;
```

---

6. Utiliser la Solution de Maintenance des Indexes d’Ola Hallengren

La solution de maintenance d’Ola Hallengren inclut un script T-SQL qui facilite la gestion des indexes et des statistiques dans SQL Server. Elle est très prisée pour sa flexibilité et son adaptabilité à des environnements SQL de toute taille, des petites bases de données aux grandes infrastructures d'entreprise.

Installation du Script d’Ola Hallengren

1. **Téléchargement du script** : Vous pouvez télécharger le script depuis le site officiel d’Ola Hallengren, sous la section "SQL Server Maintenance Solution". Ce script comprend plusieurs procédures stockées, incluant `IndexOptimize`, qui est spécialement conçu pour la maintenance des indexes.
2. **Installation** : Une fois le script téléchargé, exécutez-le sur votre base de données SQL Server. Cela va créer plusieurs procédures stockées, notamment dans la base `master` (ou une autre base de votre choix si vous préférez).
3. **Personnalisation des paramètres** : Le script permet de spécifier divers paramètres pour ajuster la maintenance des indexes en fonction de vos besoins, comme le niveau de fragmentation pour la réorganisation et la reconstruction, les priorités de traitement, et l'exclusion de certains indexes si nécessaire.

Fonctionnalités de la Procédure `IndexOptimize`

La procédure `IndexOptimize` offre plusieurs options de maintenance pour les indexes :

- **Réorganisation d’index** : Elle effectue une réorganisation en douceur des pages fragmentées, ce qui consomme moins de ressources mais réduit efficacement la fragmentation légère.
- **Reconstruction d’index** : Si la fragmentation dépasse un seuil donné (par exemple 30 %), la procédure déclenchera une reconstruction complète des indexes, supprimant toute fragmentation mais consommant plus de ressources.
- **Mise à jour des statistiques** : La procédure met également à jour les statistiques des tables et indexes, améliorant ainsi la précision des plans d’exécution et optimisant les requêtes.

Ces paramètres sont entièrement configurables en fonction de vos seuils de fragmentation et de vos besoins de performance.

Exemple d’Exécution de la Procédure `IndexOptimize`

Une fois installée, vous pouvez programmer `IndexOptimize` en utilisant le SQL Server Agent ou exécuter la procédure manuellement. Voici un exemple de script pour exécuter la procédure avec des paramètres spécifiques :

```
EXECUTE dbo.IndexOptimize
   @Databases = 'USER_DATABASES',
   @FragmentationLow = NULL,
   @FragmentationMedium = 'INDEX_REORGANIZE',
   @FragmentationHigh = 'INDEX_REBUILD_ONLINE',
   @FragmentationLevel1 = 5,
   @FragmentationLevel2 = 30,
   @UpdateStatistics = 'ALL',
   @OnlyModifiedStatistics = 'Y',
   @LogToTable = 'Y';
```

Dans cet exemple :

- `@Databases = 'USER_DATABASES'` : Applique la maintenance à toutes les bases de données utilisateur.
- `@FragmentationMedium = 'INDEX_REORGANIZE'` : Réorganise les indexes pour une fragmentation moyenne.
- `@FragmentationHigh = 'INDEX_REBUILD_ONLINE'` : Reconstruit les indexes fragmentés fortement, en ligne si possible.
- `@UpdateStatistics = 'ALL'` : Met à jour toutes les statistiques.
- `@OnlyModifiedStatistics = 'Y'` : Ne met à jour que les statistiques modifiées.
- `@LogToTable = 'Y'` : Enregistre les opérations de maintenance dans une table pour un suivi ultérieur.

Programmation des Tâches de Maintenance

La solution d’Ola Hallengren peut être planifiée via SQL Server Agent pour automatiser la maintenance. Vous pouvez créer des tâches de maintenance régulières (quotidiennes, hebdomadaires) en fonction de vos besoins de performance et des exigences de votre système. Par exemple, une réorganisation pourrait être planifiée quotidiennement, tandis qu’une reconstruction complète pourrait être effectuée une fois par semaine pendant les périodes de faible activité.

Conclusion

La gestion de la fragmentation des indexes, couplée à une bonne configuration du fill factor, est essentielle pour maintenir les performances de SQL Server. Un fill factor bien ajusté peut réduire la fragmentation et améliorer la vitesse d'insertion tout en utilisant efficacement l'espace disque. La solution d’Ola Hallengren est un outil puissant et adaptable pour gérer cette fragmentation, permettant d’optimiser les performances en automatisant la maintenance des

---

Révision #8
★Créé 5 novembre 2024 08:38:21 par Marc Leroi

Mis à jour 5 novembre 2024 09:35:22 par Marc Leroi