D. Peralta le 4 mars 2008
Rencontres AtoutFox 2007/2008

Quel SGBDR pour VFP ?

Lors des précédentes rencontres, nous avons écrit des articles sur l'utilisation des SGBDR avec VFP. Nous avons vu les différentes techniques proposées :

- Remote View
- SPT
- Cursor Adaptor

Cet exposé va tenter de donner des réponses à une question qu'un DSI pourrait vous poser : Que dois-je choisir comme SGBDR ?

Dans de nombreux cas, les entreprises possèdent déjà un SGBDR, et pour des raisons d'administration, ne souhaitent pas voir arriver un nouveau moteur. Par contre, il se peut que le moteur en place, ne corresponde pas aux besoins énoncés par le client, et dans ce cas, c'est votre développement qui sera mis en cause en cas de disfonctionnement coté données. Vous devez donc être capable de proposer le meilleur choix pour les clients n'étant pas équipés, et de pouvoir annoncer au client, les risques liés, au cas où le SGBDR serait imposé.

Coté VFP, il n'y a pas de contrainte, puisse qu'il suffit que le SGBDR possède un driver ODBC, pour que l'échange de données se fasse dans de bonnes conditions (Remote Views et SPT), ou un driver OleDb dans le cas du Cursor Adaptor. A noter que la base de données peut être montée sur un OS autre que Windows, tel que Linux.

Afin de mieux comprendre les différents services que peuvent rendre un SGBDR, nous allons parler des transactions et des verrous.

Le problème d'un système multi-utilisateurs est de garantir la cohérence d'accès aux données, c'est-à-dire que 2 utilisateurs d'un même réseau, doivent obtenir les mêmes résultats lors de l'exécution d'une même requête. La transaction est faite pour que la base soit toujours dans un état cohérent, avant et après l'écriture d'une modification cohérente. Les verrous sont faits pour empêcher les autres utilisateurs de modifier des données pendant que vous êtes en train de les lire. Ce sont ces 2 systèmes : transactions et verrouillage, qui permettent au SGBDR de livrer des informations stables et cohérentes. Un terme est souvent employé dans les SGBDR pour décrire cet état, on parle de contraintes ACID :

- Atomicité
- Cohérence
- Isolation
- Durabilité

Nous reviendrons sur les explications de ces termes tout au long de cet exposé, mais l'on peut déjà donner quelques définitions :

L'atomicité désigne l'idée qu'une transaction est un tout, qui s'exécute soit entièrement, soit pas du tout

La cohérence d'une transaction indique que les effets d'une transaction sur les données, ne modifie pas les contraintes d'intégrité des données.

L'isolation indique que les autres utilisateurs, ne voient pas les modifications faites sur les données, tant que la transaction n'est pas validée

La durabilité indique que les modifications apportées par une transaction validée, ne seront jamais perdues

Ce qu'il faut comprendre, c'est que tout a un prix. Je ne parle pas du prix d'achat du logiciel, mais des ressources nécessaires pour assurer le fonctionnement cohérent des SGBDR. Alors, les éditeurs ont fait le choix de dire :

- mon SGBDR sera le plus rapide, quitte à ce que mes données deviennent non cohérentes
- mon SGBDR sera très sécurisé, et les contraintes ACID seront toujours respectées
- mon SGBDR doit être fait pour le maximum de monde, et devra être capable d'être soit rapide, soit posséder des contraintes ACID.

Et c'est à partir de là que commence à fuser les commentaires affligeants, que l'on peut rencontrer sur certains forums, et qui annoncent que tel SGBDR est pourris, qu'un autre, c'est de la merde, etc.. , alors que le besoin n'est même pas énoncé.

En règle générale, comme pour les langages de programmation, le meilleur SGBDR est celui que l'on connaît. Il faudrait donc tous les connaître, et les utiliser, pour avoir un avis impartial sur la question.

Un exemple : si on parle d'opération bancaire de virement de compte à compte, il est nécessaire que l'opération de débit/crédit des comptes respecte les contraintes ACID, et donc, soit l'opération est appliquée en entier, soit elle ne l'est pas. Par contre, pour afficher sur Internet, des recettes de cuisine, ou des résultats d'une course à pied, on se moque de gérer les transactions, et le point important est la rapidité d'affichage. Alors, si vous parlez de MySQL à un banquier, il vous dira que c'est de la merde, et si vous parlez de DB2 à un hébergeur, il vous dira que IBM, c'est lent et trop cher, et que c'est de l'arnaque. Le débat qui s'en suivra, sera un vrai dialogue de sourd, et chacun restera campé sur ses positions.

On va donc essayer d'y voir clair, en laissant de coté l'esprit partisan ou commercial qui inonde les déclarations disponibles sur les forums ou chez les éditeurs.

Erreurs possibles pour un SGBDR :

- Mauvaise lecture : Lorsqu'un utilisateur lit un enregistrement qui est en cours de modification par un autre utilisateur. On appelle également cela un Dirty Read.
- Lecture non reproductible : Lorsque dans une même transaction, un utilisateur lit deux valeurs différentes dans une même colonne sur une même ligne.
- Fantôme : Un utilisateur récupère un jeu d'enregistrements, mais un autre modifie un de ces enregistrements durant la lecture du 1er. Le 1er utilisateur verra des fantômes.

J'insiste lourdement sur le fait, que pour certaines applications, ces erreurs ne sont pas gênantes, mais dans notre cas, nous partirons du principe que cela pose problème.

Nous allons jouer sur les isolations des transactions pour obtenir le résultat escompté, sachant que plus les contraintes ACID seront respectées, plus les ressources seront sollicitées.

On peut définir 4 grands niveaux d'isolation des transactions. Des SGBDR puissants possèdent un nombre plus important de réglages possibles, mais nous nous limiterons aux généralités. (norme)

- READ UNCOMMITTED : les 3 types d'erreurs peuvent se produire
- READ COMMITED : les dirty read sont impossibles, mais les 2 autres types d'erreur sont possibles
- REPEATABLE READ : seule les erreurs fantômes peuvent arriver
- SERIALIZABLE : Aucune erreur ne peut arriver.

A noter que sous PostGreSQL, seuls 2 niveaux sont exploités (READCOMMITED et SERIALIZABLE)

Qui peut le plus peut le moins ? : Si notre SGBDR le permet, pourquoi ne pas se placer en mode SERIALIZABLE par défaut ? Comme vous l'aurez certainement compris, le mode serializable implique qu'une transaction peut commencer, seulement lorsque la précédente sera terminée. Si je prends le cas de mon site de recettes de cuisine, cela ne permettra pas d'avoir de nombreuses connexions simultanées. Par contre, pour une appli LAN sensible, avec une dizaine d'utilisateurs connectés, ce mode garantit une cohérence maximale.

Chaque niveau d'isolation déterminera le nombre de verrous à poser sur la base. C'est cette pose de verrous qui est gourmande en ressources et en temps. Alors, les éditeurs ont imaginé des systèmes pour effectuer les verrouillages nécessaires, dans un temps le plus court possible.

On trouve 3 grands types de verrous :

- Verrous de table (flock)
- Verrous exclusifs de ligne (rlock)
- Verrous partagés de ligne (???)

Un verrou partagé de ligne, lorsqu'il est posé sur un enregistrement, permet qu'un autre utilisateur puisse lire, mais ne puisse pas modifier. Un verrou exclusif interdit la lecture comme l'écriture par un autre utilisateur.

Sous SQL Server on trouve également des verrous au niveau de la page. (groupe de lignes dans un bloc de 8000 octets)

Il existe les verrous implicites, posés par le SGBDR, et les verrous explicites, posés par le programme. Attention toutefois aux deadLocks lors du verrouillage explicite.

Exemple de DeadLock :
Début de Transaction n° 1

```
UPDATE comptes SET balance = balance + 100.00 WHERE no_compte = 11111;
```

Transaction n° 2

```
UPDATE comptes SET balance = balance + 100.00 WHERE no_compte = 22222;
UPDATE comptes SET balance = balance - 100.00 WHERE no_compte = 11111;
```

Fin de Transaction n° 1

```
UPDATE comptes SET balance = balance - 100.00 WHERE no_compte = 22222;
```

En complément il existe dans certains SGBDR, des verrous sur les index.

Test comparatif !!! :

La norme ANSI demande que le mode par défaut des SGBDR soit en SERIALISABLE. VFP, est par défaut en READ UNCOMMITED. Alors, faire des comparatifs de rapidité entre un SGBDR ANSI et des tables VFP est forcément ridicule.

Quelques SGBDR :

Tout d'abord, on trouve toute la série des Express. Ce sont des SGBDR du commerce, qui sont bridés et livrés gratuitement. On trouve SQL Express, Oracle Express, DB2 Express.

Je pense que l'on peut comparer ces versions avec les versions dites 30 jours. En effet, si les commerciaux des marques respectives on décidé de donner leurs produits, c'est que des études montrent que le volume de données stockées sur un système de bases de données,

double tous les 4 ans. Il arrivera donc un moment où la version Express ne sera plus suffisante, et que naturellement, c'est le SGBDR du même éditeur qui sera choisi. Il y a une course à la restriction qui s'est mis en place, et c'est la possibilité de stockage qui a été mis en avant. Coté MS, le MSDE (SQL 2000) est passé de 2Go à 4Go dans sa formule Express 2005, idem pour Oracle, et on est même à un stockage illimité pour DB2. Ce qu'il faut regarder, c'est qu'un SGBDR doit avoir à sa disposition autant de RAM que sa plus grosse base de données. Ainsi, une base de 3Go, dit avoir à sa disposition 3Go de RAM. Donc, dans le même temps où MS est passé de 2Go à 4 Go de data, la taille de la RAM exploitée est passé de 2Go à 1Go, et de 2 processeurs à 1 seul processeur. Idem pour Oracle et IBM. (1Go de RAM, et 1 Proc). D'un autre coté, si notre serveur est équipé d'un Windows 2003 serveur standard, on ne pourra pas dédier plus de 3 Go au SGBDR, quel qu'il soit. Le jour où votre client verra gonfler sa base de données, et aura besoin de s'équiper en produit payants, vous devrez lui imposer un OS serveur Enterprise dans le cas de MS SQL Server, ou d'un Linux pour Oracle ou DB2. Pour information, 1 licence utilisateur DB2 est de l'ordre de 1000 euros. On comprend donc mieux l'enjeu de ces produits gratuits. A noter que seul MS SQL Server n'est disponible qu'en OS Windows. Un autre SGBDR payant, et ne possédant pas de version Express, est à signaler, c'est Sysbase qui l'édite dans sa gamme Avantage. Ce dernier a la possibilité d'exploiter des fichiers DBF, en les liant aux moteurs de base de données. Je n'ai toujours pas compris l'intérêt, car s'il est recommandé de passer des DBF à un SGBDR, l'inverse est très rare.

Parmi les SGBDR du monde libre, nous ne parlerons pas de SQLite, qui n'est pas fait pour être mis en production, mais uniquement pour faciliter l'échange de fichiers lors du développement. Au dire de l'éditeur lui-même, on peut rencontrer des blocages en mode multi-utilisateurs. Donc, il reste 2 produits dignes d'intérêt : PostGreSQL et MySQL. Tous 2 peuvent être installés sur l'OS de votre choix, mais le mode de licence est différent. Seul PostGreSQL est totalement gratuit. Par contre, il peut être rassurant de savoir qu'une société commerciale est capable de vous venir en aide en cas de gros soucis. C'est ce que fait MySQL AB avec MySQL, en proposant des contrats de dépannage, à 4000 euros / an dans sa version Gold.

Si on essaie de classer ces SGBDR dans des catégories liées aux contraintes ACID, on peut mettre d'un coté MS, Oracle, IBM, PostGreSQL, et MySQL InnoDB, et d'un autre, MySQL MyISAM.

Spécificités :

Parmi les spécificités marquantes, on peut noter dans le cas de MySQL InnoDB et Oracle, qu'un système de lecture des lignes verrouillées à été mis en place. Lorsqu'un verrouillage à la ligne est posé, une copie de cette ligne est mise en mémoire, et c'est cette duplication qui sera lue tant que le verrou ne sera pas ôté. Cela permet de ne pas avoir de requête en attente de libération, et donc une vitesse accrue en lecture, sur des systèmes chargés. Ce mode de fonctionnement se rapproche du niveau SNAPSHOT de SQL Server 2005.

Du coté de MySQL MyISAM, le seul verrouillage existant est le verrouillage de table. Cela veut dire qu'une lecture ne peut se faire lors d'une écriture. Pour éviter les blocages, les mises à jour peuvent être différées, et ne s'effectuer que lorsqu'il n'y a plus de lecture en attente. On constate donc que ce SGBDR est fait pour optimiser les lectures, et ne pas faire de mise à jour.

Coté buffer, les SGBDR ont leurs techniques pour améliorer les performances. Il faut savoir qu'une lecture en RAM est 1000 fois plus rapide qu'une lecture sur disque. Certain SGBDR stocke en RAM le contenu des tables, d'autres stockent et index, et enfin MyISAM

stocke les résultats des requêtes. Il est évident que la requête sera rejouée si une donnée a changée dans la base. En fait, lors du verrouillage de table pour une modification, le buffer de requête est vidé. Cela prouve encore une fois que MySQL MyISAM est fait pour lire rapidement, mais pas pour écrire.

Conclusion :

Impossible de dire quel est le SGBDR le meilleur pour vos besoins, mais désormais, vous devriez pouvoir en discuter avec vos clients.