## Forum: WD 9
### Topic: SQL et threads
### Subject: Re: SQL et threads
PostÃ© par: R&B
Contribution le : 23/3/2005 8:22:02

Après quelques recherches, j'ai du repenser la méthode de lancement de mes requêtes.

Précision :
- le traitement en cours d'optimisation est simple. Il faut produire un état statistique sur la base de données.
Pour une sélection d'enregistrements et une période, on va, pour chacun, aller récupérer des cumuls dans différentes paires de fichiers "entête-lignes". Ces cumuls serviront de termes dans l'opération de calcul final à afficher dans l'état.

Dans le détail, pour chaque enregistrement de la sélection (pouvant aller jusqu'à 100 000 enregistrement par exemple), on va lancer dans une majorité de cas un train de 6 requêtes SQL. nutile de penser alors laisser ce traitement à 1 seconde par ligne !

Pourquoi des requêtes : parce que les informations qui servent à effectuer le cumul se trouvent à chaque fois dans deux fichiers liés (Entete : Date (pour la période), Ligne : Code cherché + Valeur cumulée). Les requêtes semblent la méthode préconisée pour ce type d'accès.
Les requêtes sont dans ce genre :
SELECT SUM(LIGNE.RUBCUMUL) AS CUMUL FROM LIGNE,TETE WHERE  TETE.IDTETE = LIGNE.IDTETE  AND LIGNE.CODE = {Param1}  AND TETE.DATE >= {Param2}  AND TETE.DATE <= {Param3}  GROUP BY LIGNE.CODE
On remarquera que le résultat est réduit à sa plus grande simplicité : une ligne et une rubrique.
L'objectif est bien à l'optimisation.

Contrainte : ce traitement étant utilisé dans deux fenêtres, nous somme partis sur une classe en charge de ces calculs et du lancement des requête.

Ce que nous avons fait :
Nous nous sommes penché sur l'exemple WinDev "Pool de thread" et en avons pris le mécanisme de lancement  des requêtes.

Un compteur de thread, un tableau de la liste des requêtes (:m_tReq) et de leur résultat (:m_tRes).
Une méthode d'initialisation de sémaphore (???) et d'arret des threads.
Une méthode de lancement de requêtes (thread_requete) et celle de sont lancement (Thread_Lance).

PROCEDURE Thread_Requete(nNumRequete,pnThread) //  // On indique que le thread s'est chargé et lancé ThreadEnvoieSignal("") // Contrôle du nombre de thread à un instant t grâce aux sémaphores (???) SémaphoreDébut("SEM_LIMITE") bResExecute est un booléen SectionCritiqueDébut(":Thread_Requete") bResExecute = HExécuteRequête(:m_tReq[nNumRequete],hRequêteDéfaut,:m_cCI,:m_dDeb,:m_dFin) SectionCritiqueFin(":Thread_Requete") SI bResExecute ALORS  SI

HLitPremier(:m_tReq[nNumRequete]) ALORS
  :m_tRes[nNumRequete]={:m_tReq[nNumRequete]+".QTE",indRubrique}  FIN FIN
HAnnuleDéclaration(:m_tReq[nNumRequete])  // On peut libérer une unité du sémaphore
SémaphoreFin("SEM_LIMITE",1) // Arrêt du thread ThreadArrête("")      PROCEDURE
Thread_Lance(pnIndice) // lance la requête de l'indice Sablier(Vrai) // On exécute le thread
ThreadExécute(:m_cNomThread+:m_nThread,threadNormal,":Thread_Requete",pnIndice,:m_nThre
ad) // On attend que le thread se soit lancé ThreadAttendSignal() // On incrémente le numéro du
thread en cours :m_nThread ++ Sablier(Faux)

Le but est de lancer les requêtes et attendre que toutes soient exécutées pour passer à la suite...
Il s'agit donc, semble-t il, de synchroniser non pas deux mais plusieurs threads (cet exemple en synchronise deux)... et sur le sujet à part la liste des fonctions relatives aux signaux, les exemples sont rares.