Product Documentation

Peregrine

AssetCenter

Optimisation

Part No. DAC-441-FR35

Peregrine SYSTEMS®

Build 17

Pour toute information technique sur ce produit ou pour faire la demande d'une documentation sur un produit dont vous possédez la licence, veuillez contacter le support client Peregrine Systems, Inc. en envoyant un e-mail à l'adresse suivante : support@peregrine.com.

Pour tout commentaire ou suggestion à propos du présent document, veuillez contacter le département des publications techniques de Peregrine Systems, Inc. en envoyant un e-mail à l'adresse suivante : doc_comments@peregrine.com.

Cette édition s'applique à la version 4.4 du programme sous contrat de licence

AssetCenter

Peregrine Systems, Inc.
3611 Valley Centre Drive San Diego, CA 92130
858.481.5000
Fax 858.481.1751
www.peregrine.com

PEREGRINE

Table des matières

PEREGRINE

Introduction

A quoi sert ce manuel

Ce manuel décrit des stratégies d'optimisation de AssetCenter. En particulier, il étudie certaines pistes pour réduire les goulets d'étranglement occasionnés par :

- le réseau,
- le serveur de base de données,
- le client AssetCenter,
- le moteur de base de données

A qui s'adresse ce manuel

Ce manuel s'adresse en priorité aux administrateurs de AssetCenter et de la base de données. Il contient des informations et décrit des procédures qui, si elles sont mal comprises, mal implémentées ou appliquées à mauvais escient peuvent entraîner des problèmes allant d'impacts négatifs de performance à la perte ou à la corruption de données.

Avertissement :

Ce manuel a pour objectif d'aider les administrateurs à identifier et résoudre les problèmes de performances associés au logiciel et/ou à la base de données. Il ne s'agit pas d'un guide exhaustif sur l'optimisation des performances d'une base de données. Les stratégies d'optimisation de AssetCenter décrites dans ce document se veulent aussi complète que possible. Elles ne constituent cependant pas l'intégralité des possibilités d'optimisation disponibles. Peregrine Systems, Inc. ne saurait être tenu pour responsable des dégâts ou pertes de données occasionnés à la suite d'une mauvaise utilisation des informations contenues dans ce manuel. Si vous n'êtes pas certain de bien comprendre une procédure ou de ne pas en mesurer toutes les conséquences, ne l'appliquez pas. En tout état de cause, il est fortement recommandé de réaliser une copie de sauvegarde complète de votre environnement (base de données, personnalisations,...) avant d'effectuer toute modification de votre implémentation.

Rappel sur l'architecture de AssetCenter

AssetCenter repose sur une architecture deux-tier. Le client AssetCenter se connecte directement à la base de données et rentre dans la catégorie de ce qu'on appelle communément un "client lourd" qui s'exécute et réside entièrement sur un poste. Ce type d'architecture offre généralement de meilleures performances dans le cas d'un réseau LAN rapide. A contrario, la chute de performance est géométrique sur un réseau lent. Concrètement si la latence du réseau est deux fois plus importante que celle recommandée de 10 ms, la chute de performance sera généralement d'un facteur quatre.

Un des composants clés de AssetCenter est AssetCenter Serveur. Ce serveur n'échange aucune donnée avec le client. Il se connecte à la base de données séparément et réalise en tâche de fond toutes sortes d'opérations de maintenance et de surveillance. Par exemple : une alarme placée sur la date d'expiration d'un contrat provoque l'envoi d'un mail de rappel 30 jours avant l'échéance. AssetCenter traite également les workflows, le calcul des loyers ainsi que la validation de la clé d'autorisation de la base de données. La plupart des optimisations décrites dans ce document ne concernent pas AssetCenter Serveur.

Goulets d'étranglement

Dans l'utilisation courante de AssetCenter, on distingue principalement quatre goulets d'étranglements potentiels : le réseau, le client, le serveur de base de données et le moteur de base de données.

Les problèmes de performances associés au réseau surviennent, dans l'immense majorité des cas, lorsque AssetCenter est utilisé sur un réseau dont les caractéristiques sont inférieures (en terme de latence par exemple) à celles recommandées par Peregrine Systems, Inc.

Les problèmes de performances perçus comme provenant du client sont en fait très souvent des problèmes liés à la base de données et/ou au réseau. Il s'agit fréquemment d'une requête dont le résultat met du temps à parvenir au client, soit du fait de la lenteur du réseau, soit parce que la base de données tarde à répondre. Les problèmes associés au serveur de base de données sont en règle générale liés à un manque de puissance ou de ressources de la machine qui fait tourner le serveur. C'est la raison pour laquelle les pré-requis matériels pour une implémentation de AssetCenter doivent être étudiés de très près et très tôt.

Les problèmes associés au moteur de base de données sont les plus fréquents. Dans ce cas de figure, le SGBD ne répond pas suffisament rapidement aux requêtes qui lui sont envoyées. La caus peut en être imputée à une variété de facteurs tels que : problème d'optimisation de la base de données, requêtes inutilement complexes, mauvais choix d'optimisation, ressources matérielles insuffisantes, ...

 Note :

Même si le matériel sur lequel tourne le moteur et/ou le serveur de base de données est largement suffisant en termes de puissance, le moteur et/ou le serveur de base de données ne sont pas nécessairement configurés pour tirer le meilleur parti de cette puissance. Dans le même ordre d'idée, si plusieurs applications partagent le même serveur de base de données, gardez à l'esprit que les ressources seront partagées entre les applications et que les performances peuvent s'en ressentir.

Contenu du manuel

Chapitre Optimiser un client

Ce chapitre traite de l'optimisation d'un poste client.

Chapitre Optimiser la base de données

Ce chapitre traite de l'optimisation de la base de données.

Conventions utilisées dans ce manuel

Les conventions de notation suivantes sont utilisées tout au long de ce manuel :

Convention	Description
`Code Java Script`	Exemple de code ou de commande
`Police fixe`	Commande DOS, paramètre de fonction ou formatage de données.
...	Portion de code ou de commande omise.
Note : Informations complémentaires...	Note à valeur informative
IMPORTANT : Soyez vigilants...	Information importante pour l'utilisateur
Astuce : Astuce d'utilisation....	Astuce
Avertissement : Attention	Information extrêmement importante pour l'utilisateur
Objet	Objet de l'interface graphique de AssetCenter : un menu, une entrée de menu, un onglet ou un bouton.

Les conventions suivantes sont également appliquées :

- Les étapes que vous êtes invités à suivre dans un ordre défini sont présentées sous la forme d'une liste à puce numérotée. Par exemple :
  1. Première étape

2 Seconde étape

3 Troisième et dernière étape

- Toutes les figures et les tables sont numérotés en fonction du chapitre dans lequel ils se trouvent et de leur ordre d'apparition à l'intérieur du dit chapitre. Par exemple, le titre du quatrième tableau du chapitre deux sera préfixé par la mention **Tableau 2-4**.

CHAPITRE 1

Optimiser un client

Ce chapitre explore les différentes stratégies d'optimisation des performances de AssetCenter. Il propose une analyse de certains mécanismes internes de AssetCenter qui peuvent avoir un impact sur les performances et permet de diagnostiquer plus efficacement vos problèmes tout en y apportant des pistes de résolution.

Latence du réseau

Comme nous l'avons vu précédemment, la qualité du réseau peut avoir un impact direct sur les performances de l'application. La latence recommandée pour une bonne utilisation de AssetCenter doit se situer sous la barre des 10 millisecondes.

Vue en liste et vue hiérarchique

AssetCenter permet de voir les enregistrements d'une table selon deux modes bien distincts :

- Une vue hiérarchique des enregistrements de la table. Cette vue permet de naviguer au sein de la hiérarchie des enregistrements en dépliant les branches

de l'arborescence. Vous pouvez ainsi visualiser les relations parents-enfants entre les enregistrements. Ce type de vue est disponible pour toutes les tables qui possèdent un champ **FullName**, par exemple : la table des **Services et Personnes** (**amEmplDept**), la table des **Eléments de parc** (**amPortfolio**), la table des **Localisations** (**amLocation**), etc.

- Une vue en liste, où les enregistrements sont affichés à plat sans notion de hiérarchie.

La vue hiérarchique est beaucoup plus coûteuse en termes de performances que la vue en liste. Sur un réseau rapide et sur une table de taille moyenne, les performances de ce type de vue sont bonnes. Plus le réseau est lent et/ou la table est importante et plus les performances sont dégradées.

Deux solutions immédiates d'optimisation s'offrent à vous :

1. Visualiser les tables en vue en liste si le réseau est lent et/ou si la table est de taille importante.
2. Modifier les paramètres d'affichage par défaut des listes hiérarchiques de façon à ce qu'elle s'affichent initialement en vue repliée. Ceci permet d'accélérer l'affichage initial de la liste. Pour ce faire, choisissez la valeur **Oui** pour l'option **Branches refermées par défaut** de AssetCenter (**Edition/ Options/ Listes/ Listes principales** et **Edition/ Options/ Listes/ Autres listes**).

Mécanisme de chargement des listes

Lorsque l'option **Branches refermées par défaut** est activée, AssetCenter utilise le mécanisme suivant pour afficher les listes (l'exemple ci-après concerne l'ouverture en vue hiérarchique de la table des **Services et Personnes**) :

1. AssetCenter envoie une requête pour récupérer tous les enregistrements situés au plus haut niveau de la hiérarchie (ceux pour lesquels le champ **sLvl** vaut **0**) et dont le **Nom complet** (**FullName**) est supérieur au premier enregistrement de la liste. La requête envoyée à la base de données est la suivante :

```
SELECT E1.lIconId, E1.lEmplDeptId, E1.MrMrs, E1.Name, E1.FirstName, E1.
Title, E1.Phone, E1.Fax, E1.FullName FROM amEmplDept E1 WHERE E1.sLvl=0
AND E1.FullName >='/Admin,,ADMIN/' ORDER BY E1.FullName
```

2. Lorsqu'une branche de l'arborescence est dépliée, une nouvelle requête est envoyée à la base de données. Cette requête va récupérer les enregistrements de niveau hiérarchique directement supérieur et dont le nom complet contient celui du parent. Par exemple (dans ce cas, la branche Taltek de l'arborescence a été dépliée) :

```
SELECT E1.lIconId, E1.lEmplDeptId, E1.MrMrs, E1.Name, E1.FirstName, E1.
Title, E1.Phone, E1.Fax, E1.FullName FROM amEmplDept E1 WHERE E1.FullNa
me LIKE '/Taltek/%' ESCAPE '\\' AND E1.sLvl=1 ORDER BY E1.FullName
```

A chaque fois qu'une branche est dépliée, une requête de ce type est envoyée par AssetCenter. Si l'enregistrement que vous recherchez est situé bas dans la hiérarchie, vous multiplierez les requêtes envoyées à la base de données.

Par comparaison, la même table ouverte en vue en liste, déclenche l'envoi d'une seule et unique requête du type :

```
SELECT E1.lIconId, E1.lEmplDeptId, E1.MrMrs, E1.Name, E1.FirstName, E1.Tit
le, E1.Phone, E1.Fax, E1.FullName FROM amEmplDept E1 ORDER BY E1.lEmplDept
Id
```

Tous les enregistrements renvoyés par la requête sont utilisés pour peupler la liste, à concurrence du nombre d'enregistrements à charger, tel qu'il est défini dans les options de AssetCenter.

Chargement des listes

Plusieurs options sont disponibles pour définir le nombre d'enregistrements initialement chargés dans une liste. En réduisant ou limitant ce nombre vous pouvez améliorer le temps de réponse à l'affichage d'une liste. Les options concernées se trouvent dans les sections **Edition/ Options/ Listes/ Listes principales** et **Edition/ Options/ Listes/ Autres listes**. Deux options sont disponibles :

- **Ne pas charger pendant plus de** : cette option définit la durée maximale (en millisecondes) pendant laquelle AssetCenter essaie de peupler la liste. Quand cette durée est dépassée, l'opération est interrompue et la liste est peuplée avec les enregistrement récupérés. Par défaut cette durée est de 5000 millisecondes.
- **Ne pas charger plus de** : cette option définit le nombre maximal d'enregistrements chargés dans une liste. Par défaut 200 enregistrements sont chargés.

 Note :

Ce nombre correspond également au nombre d'enregistrements additionnels qui seront chargés à chaque fois qu'un utilisateur clique sur l'icône + d'une liste.

Les valeurs par défaut de ces deux options ont été étudiées pour satisfaire la grande majorité des cas. Néanmoins, dans des cas particuliers où la configuration de la liste est complexe, ces valeurs peuvent s'avérer pénalisantes en terme de performances. Dans des cas de problèmes de performances à l'affichage des listes, il peut donc être utile de :

- Réduire la valeur de l'option **Ne pas charger plus de** pour la porter à 30 ou 50 par exemple.
- Modifier la valeur du paramètre **ArrayFetchingSize** de AssetCenter. La valeur de ce paramètre spécifie le nombre d'enregistrements récupérés par AssetCenter dans chaque paquet réseau. Pour des raisons de performance, cette valeur doit être égale à celle définie dans l'option **Ne pas charger plus de** plus un. Par exemple, si la valeur de l'option **Ne pas charger plus de** est 200, celle du paramètre **ArrayFetchingSize** doit être 201. Cette incrémentation permet de tenir compte de l'enregistrement NULL, renvoyé dans le résultat de la requête, mais qui n'est pas affiché par AssetCenter.

  Pour modifier la valeur du paramètre **ArrayFetchingSize** :

  1. Chargez le fichier **`amdb.ini`** dans un éditeur de texte,
  2. Dans la section correspondant à la déclaration de la connexion pour votre base de données, repérez le paramètre **ArrayFetchingSize**. Ci-dessous un exemple pour la base de données **ACDemo432en** :

```
[ACDemo432en]
Engine=SQLAnywhere
Location=ACDemo432en
EngineLogin=itam
EnginePassword=7BC2423F1B1F6A42E2B3A27E396F52C3A9AD6828582AED88DAC2F
53E9A369BC03E6393911903124254200200
ReadOnly=0
CacheSize=5120000
ArrayFetchingSize=201
AmApiDll=C:\PROGRA~1\PEREGR~1\ASSETC~1\ASSETC~2\bin\aamapi43.dll
```

  3. Modifiez la valeur du paramètre et sauvegardez vos modifications.

Note :

L'ajustement de ce paramètre peut également s'avérer particulièrement utile dans le cas où AssetCenter est utilisé sur un réseau WAN (Wide Area Network).

Autocomplétion

Par défaut, AssetCenter recourt à un mécanisme d'autocomplétion pour aider l'utilisateur dans la saisie des données. L'autocomplétion fonctionne de la manière suivante : lorsque l'utilisateur arrête la saisie dans un enregistrement lié, AssetCenter respecte une pause définie dans les options de l'application (500 millisecondes par défaut) puis effectue une requête sur la base de données pour retrouver l'enregistrement correspondant à ce que l'utilisateur a commencé à saisir.

Ce mécanisme est particulièrement utile dans bien des cas. Lorsque vous saisissez le nom de l'utilisteur d'un élément de parc, il permet de retrouver et de compléter le nom de la personne à partir de quelques lettres. Le désavantage de cette option est que, pendant toute la durée d'envoi, d'exécution et de récupération des résultats de la requête, l'utilisateur ne peut continuer la saisie. Si le réseau est lent et/ou que la table sur laquelle pointe le lien est volumineuse, l'impact sur les performances peut être lourd. Dans ces cas de figure, il est recommandé de débrayer le mécanisme d'autocomplétion. Pour ce faire :

1. Sélectionnez le menu **Edition/ Options/ Navigation/ Choix d'enregistrements liés/ Autocomplétion au bout de**,
2. Cliquez sur le nombre situé à droite de cette option,
3. Saisissez un nombre très important (par exemple 1000000) pour prévenir l'exécution du mécanisme,
4. Cliquez sur **OK**.

Restrictions d'accès

Une restriction d'accès est un des constituants d'un profil utilisateur d'AssetCenter. Il correspond à un filtre des enregistrements d'une table. Par exemple, vous pouvez faire en sorte qu'un technicien ne puisse accéder qu'aux biens de son service, à l'exclusion de tous les autres.

De par leur nature et leur capacité à sécuriser une implémentation, les restrictions d'accès sont, à juste titre, fréquemment utilisées. Elles engendrent cependant des modifications substantielles des requêtes qui sont envoyées à la base de données lorsqu'un utilisateur ouvre un écran. Pour cette raison, les restrictions

d'accès, ou plus exactement les restrictions d'accès mal définies ou mal implémentées sont fréquemment la source de problèmes de performances.

Les restrictions d'accès sont toujours dépendantes du contexte d'une table et portent sur la lecture et sur l'écriture (ajout ou modification) d'enregistrements. Les restrictions en lecture empêchent un utilisateur de visualiser une catégorie définie d'enregistrements. L'extension naturelle de ce mécanisme est que cet utilisateur ne pourra modifier ces enregistrements qu'il ne peut pas visualiser. Les restrictions en écriture empêchent un utilisateur de modifier ou de créer une certaine catégorie d'enregistrements mais n'affectent pas la visualisation.

Mécanisme des restrictions d'accès

- Les restrictions en lecture sont appliquées par l'ajout d'une clause WHERE à la requête envoyée à la base de données. Par exemple, prenons une restriction d'accès qui définit qu'un utilisateur ne peut visualiser que les centres de coûts dont il est responsable. La requête envoyée est du type :

```
SELECT C1.lCostId, C1.Title, C1.AcctNo FROM amCostCenter C1, amEmplDept
amEmplDept_CurrentUser WHERE amEmplDept_CurrentUser.lEmplDeptId = C1.lS
upervId AND amEmplDept_CurrentUser.lEmplDeptId = 25323 ORDER BY C1.lCos
tId
```

  Ce cas de figure simple et typique n'est pas coûteux en termes de performance. Mais si la requête est plus complexe, et en particulier si elle contient une ou plusieurs sous-requêtes, le moteur de base de données mettra plus de temps à la traîter et à envoyer le résultat à AssetCenter ce qui se traduira par une lenteur perçue à l'affichage de l'écran. Là encore, la volumétrie de la table sur laquelle porte la requête joue un rôle important.
- Les restrictions en écriture fonctionnent de manière légèrement différente. Lorsqu'un utilisateur tente de modifier un enregistrement d'une table possédant une ou plusieurs restrictions d'accès en écriture, AssetCenter envoie une requête SELECT séparée à la base de données qui explicite la restriction d'accès dans sa clause WHERE. Si l'enregistrement que l'utilisateur tente de modifier n'est pas renvoyé, un message informe l'utilisateur qu'il ne possède pas les droits suffisants pour modifier l'enregistrement et les modifications sont annulées au niveau base de données. Si l'enregistrement est renvoyé un ordre UPDATE est envoyé à la base de données et les modifications sont validées. Le traffic généré est donc potentiellement deux fois plus important (une requête de test et une requête de modification). De plus, comme pour les restrictions en lecture, la complexité des requêtes et

la volumétrie des tables impliquées sont aussi un facteur de dégradation des performances.

IMPORTANT :

Il est donc impératif d'analyser le plus finement possible les besoins en restrictions d'accès et de les implémenter de la façon la plus rationnelle, la plus simple et la plus minimaliste possible.

Propriétés des objets

Les propriétés des objets (champs, liens,...) de la base de données peuvent avoir un impact significatif sur les performances.

Note :

Ces propriétés sont accessibles au travers du menu contextuel **Configurer l'objet** ou en utilisant AssetCenter Database Administrator.

Les propriétés d'affichage

Les objets de la base de données de AssetCenter, notamment les champs et les liens, possèdent des propriétés qui conditionnent leur mode d'affichage. Ces propriétés sont :

- **Obligatoire** : le label du champ est rouge et le champ doit être renseigné pour valider l'enregistrement,
- **Lecture seule** : le champ est grisé,
- **Hors contexte** : le champ n'est pas affiché.

Ces propriétés peuvent prendre les valeurs suivantes :

- Oui
- Non
- Script

  Dans ce dernier cas, un script Basic interprété détermine au final la valeur de la propriété.

Une valeur Oui ou Non pour une propriété n'a aucune incidence sur les performances, aucune évaluation n'étant requise. Le cas est différent pour un

script. Lorsqu'un enregistrement est chargé ou mis à jour, tous les scripts associés à l'enregistrement s'exécutent. Les scripts simples qui travaillent sur des données locales sont relativement anodins pour les performances. Par contre, les scripts qui interrogent la base de données (par exemple en utilisant des fonctions du type **AmDbGetLong()**) vont générer des requêtes et ont un impact direct sur les performances. Il en va de même lors du référencement de la valeur d'un champ ou lien distant dans un script, par exemple :

```
[CostCenter.Supervisor.Location.City]
```

En résumé :

- Analysez précisément le besoin de scripter une propriété,
- Gardez à l'esprit que tout script effectuant une requête sur la base de données peut impacter les performances,
- Optimisez vos scripts au maximum afin qu'ils soient le plus efficace possible.

L'historisation

AssetCenter offre la possibilité de conserver l'historique des modifications apportées à un objet de la base. Cette fonctionnalité est accessible au travers de la propriété **Historisé** des champs ou des liens. Le mécanisme d'historisation est le suivant :

1. Un utilisateur modifie un enregistrement. Si cette modification porte sur un objet dont la propriété **Historisé** est évaluée à 1 (i.e elle a pour valeur **Oui** ou un script renvoie la valeur 1), l'agent d'historisation est déclenché.
2. L'agent d'historisation crée un enregistrement dans la table des historiques (**amHistory**) et stocke notamment l'ancienne et la nouvelle valeur de l'objet.

Tout comme les propriétés **Obligatoire**, **Lecture seule** et **Hors contexte** précédemment évoquées, la propriété **Historisé** peut être source de problèmes de performances si un script complexe (ou effectuant des requêtes sur la base de données) lui est associé. Un autre impact sur les performances est directement lié au fait qu'une requête d'insertion dans la table des historiques est envoyée à la base de données pour chaque objet historisé. Dans de bonnes conditions, s'il n'y a que quelques objets historisés, cela ne pose pas de problème particuliers. Mais la situation est différente dans le cas où des dizaines d'objets historisés sont modifiés. Un autre facteur de dégradation des performances provient du fait que le temps d'insertion d'un enregistrement dans une table est directement fonction du volume de la table. Plus celle-ci contient d'enregistrement et plus l'insertion d'un nouvel enregistrement est longue. Dans les cas où un administrateur choisirait de conserver l'historique sur un nombre conséquent

d'objets, il n'est pas rare d'observer plusieurs dizaines de millions d'enregistrements dans la table des historiques.

Il est donc conseillé :

- De ne conserver l'historique des objets qu'après une analyse précise des besoins. Il ne faut pas y recourir de façon systématique.
- De décider à l'avance d'un délai de purge acceptable au delà duquel un engistrement est effacé de la table des historiques. Vous pouvez par exemple écrire un workflow qui efface toutes les historiques de plus de six mois.

La validité d'un enregistrement

Les scripts de validité sont utilisés par AssetCenter pour contrôler certains critères inhérents à l'enregistrement avant son insertion dans la base de données. Par exemple, ce type de script est utilisé pour contrôler que la date de fin d'un contrat saisie par l'utilisateur n'est pas antérieure à la date de début du contrat. Si tel est le cas, l'enregistrement n'est pas inséré et un message d'erreur est envoyé à l'utilisateur.

Ces scripts sont assujettis au mêmes contraintes de performances que celles précédemment évoquées pour les propriétés des objets.

Les caractéristiques

Les caractéristiques sont une fonctionnalité historique de AssetCenter. Introduites dans les versions antérieures à la 4.0.0, elles permettaient de définir des informations additionnelles pour des enregistrements, à une époque ou le schéma de la base de données n'était pas extensible pour l'utilisateur. Bien que puissantes, les caractéristiques peuvent être coûteuses en termes de performance. Si l'on prend comme exemple la table des localisations, trois tables sont impactées lors de l'utilisation des caractéristiques, comme le montre le schéma ci-après :

La table des localisations (amLocation) ne contient aucune information sur les caractéristiques. Elle est simplement la cible d'une table de valeurs (amFVLocation) associée qui contient la valeur des caractéristiques pour les enregistrements de la table des localisations. La table des caractéristiques (amFeature) contient, quant à elle, toutes les données physiques de la caractéristique, telles que son nom SQL, son Label, etc.

A chaque fois qu'une caractéristique est associée à un enregistrement, au moins une jointure suplémentaire est nécessaire pour mettre en place le mécanisme, écrire ou retrouver les données de la caractéristique. L'impact sur les performances peut être important, et d'autant plus dans le cas où la caractéristique est affichée dans la liste.

Depuis la version 4.0.0 de AssetCenter, il est possible de modifier directement le schéma de la base de données. L'utilisation des caractéristiques se justifie donc de moins en moins. Cependant, vous pouvez choisir d'y avoir recours dans les deux cas suivants :

1. Quand l'information ne doit porter que sur un petit pourcentage (en pratique moins de 1%, mais ce chiffre varie grandement en fonction des implémentations) des enregistrements d'une table, il est plus intéressant d'utiliser une caractéristique que de créer un champ supplémentaire dans la table. Une caractéristique n'occupe de l'espace de stockage que si elle possède une valeur (sauf dans le cas notable ou elle possède la propriété **Forcer l'affichage**). Un champ supplémentaire utilise de l'espace de stockage qu'il contienne ou non des données. Dans le cas qui nous intéresse, l'utilisation d'un champ peut conduire à une perte massive de l'espace de stockage.
2. Le processus de modification de la structure de la base de données est simple mais n'est pas anodin. Si un problème survient pendant la mise à jour de la structure de la base de production, comme un problème réseau, il est possible que la base de données se retrouve dans un état intermédiaire incohérent. C'est la raison pour laquelle Peregrine Systems recommande :
   - de réaliser une copie de sauvegarde de la base avant toute modification de structure,
   - de planifier les changements et de confier leur réalisation à des personnes expérimentées.

   Il peut être alors utile dans la phase de planification des modifications d'utiliser temporairement une caractéristique, puis lors de la modification effective de la structure de la base de transférer les données de la caractéristique vers le champ ou le lien nouvellement créé.

Configuration des listes

La très grande majorité des problèmes de performance est imputable à des problèmes de configuration des listes. Lorsqu'un utilisateur ouvre un écran de l'application, AssetCenter envoie une requête à la base pour récupérer les

données. Cette requête est générée sur la base de plusieurs paramètres, dont la configuration des listes (colonnes présentes dans la liste, tris, filtres, etc.), les restrictions d'accès, le type de vue,...

Les colonnes présentes dans la liste ont une importance particulière puisqu'elles seront utilisées dans la clause SELECT de la requête. Le corollaire de cette analyse est que les performances sont optimales quand les colonnes correspondent à des champs présents dans la table sur laquelle porte la requête. Dès qu'une colonne correspond à un champ d'une autre table, une jointure doit être réalisée dans la requête, ce qui impacte les performances.

 Note :

Ceci est également vrai pour les caractéristiques, pour les raisons évoquées précédemment.

Il est également recommandé d'analyser vos besoins avant d'ajouter les éléments suivants à la configuration d'une liste :

- Un champ calculé : les champs calculés de type Basic sont évalués pour chaque enregistrement présent dans la liste. Plus le nombre d'enregistrements est élevé et plus les performances vont s'en trouver dégradées.
- La chaîne de description **self** : pour certaines tables, la chaîne de description est complexe (par exemple sur la table des éléments de parc) et peut requérir une ou plusieurs jointures externes.

Les tris

Lorsqu'une liste est triée sur la valeur d'un champ ou d'un lien, le moteur de base de données réordonne les résultats qui sont renvoyés à AssetCenter. Cette opération est peu coûteuse en termes de performances lorsque l'objet est indexé au niveau de la base de données. Schématiquement, voici une liste des stratégies de tri applicables à une liste, de la moins pénalisante à la plus coûteuse pour les performances :

1. Tri sur un champ indexé de la table
2. Tri sur un champ indexé d'une table distante d'un lien de la table d'origine (par exemple **User.BarCode** si l'on considère la liste des éléments de parc)
3. Tri sur un champ non-indexé de la table
4. Tri sur plusieurs champs indexés de la table
5. Tri sur plusieurs champs non-indexés de la table

6 Tri sur plusieurs champs (indexés ou non) d'une table distante d'un lien de la table d'origine
7 Tri sur la chaîne de description d'une table liée
8 Tri sur des champs (indexés ou non) d'une table distante de plus d'un lien de la table d'origine

Les filtres

Les filtres sur les listes se traduisent directement par une clause WHERE au niveau de la requête envoyée par AssetCenter à la base de données. Pour des raisons similaires à celles invoquées dans le cas des tris, la complexité de la requête et les champs ou liens qu'elle fait intervenir (particulièrement s'ils s'agit de champs ou liens distants) impacte les performances. Les filtres doivent donc être bien pensés et bien écrits pour ne pas pénaliser les performances. Les quelques règles suivantes peuvent s'avérer utiles lors de l'écriture des filtres :

- Dans la mesure du possible, il vaut mieux utiliser la clause AND que la clause OR
- L'utilisation des clauses LIKE et NOT LIKE est à éviter. Si vous devez y avoir recours, utilisez un caractère joker en fin de comparaison et un champ indexé. Par exemple, il vaut mieux écrire :

```
FullName LIKE '/Taltek/%'
```

que :

```
FullName LIKE '%/Taltek/%'
```

Les caractères joker en début de chaîne pour la comparaison forcent un recherche complète de la base de données, ce qui peut s'avérer très pénalisant.

- Il est préférable d'utiliser les clauses IN et NOT IN avec des constantes. Par exemple :

```
Name IN ('Thomas', 'James')
```

L'utilisation de ces clauses sur une sous-requêtes impacte fortement les performances.

CHAPITRE 2

Optimiser la base de données

Ce chapitre traite de l'optimisation de la base de données sur laquelle opère AssetCenter. Il ne saurait se substituer à tous les ouvrages qui ont été écrits sur le sujet mais décrit néanmoins quelques possibilités d'optimisation liées à AssetCenter. En tout état de cause, il est recommandé de laisser votre administrateur de bases de données réaliser toutes les opérations de cette nature.

A propos du matériel

Déterminer les pré-requis matériels adaptés pour le serveur de base de données est particulièrement difficile. Ce choix est fonction d'un nombre de paramètres importants qui varient grandement d'une implémentation à une autre, tels que le nombre d'utilisateurs concurrents, le volume des données, le moteur de base de données, etc.

La mémoire vive

Une des règles de base est que la mémoire vive est le facteur le plus déterminant en termes de performances du moteur de la base de donénes. La mémoire vive sert notamment à conserver un cache des requêtes exécutées et des données.

Plus la mémoire vive est importante sur un système et plus les chances que les données requises par l'application soient conservées dans le cache (et donc directement accessible sans traitement) sont importantes. Les informations liées aux utilisateurs, et en particulier les informations de connexion, sont également stockées en mémoire vive. Plus le nombre d'utilisateurs est grand et plus vos besoins en mémoire vive seront importants.

Les processeurs

Le nombre et la puissance des processeurs est également un facteur déterminant de la performance. Certains moteurs de bases de données savent tirer parti d'une configuration multi-processeurs et réalisent leur calculs en parallèle. Il est donc important de valider ce point lors du choix de votre configuration.

Les disques durs

Les performances des disques durs sont également à prendre en compte. Le système doit être équilibré : il ne sert à rien d'avoir à disposition de grandes quantités de mémoire vive et plusieurs processeurs rapides si le ou les disques durs sont lents, car ils constituerton le goulet d'étranglement du système. Idéalement, les disques durs doivent être aussi rapides que possible, quelque soit le protocole de transmission des données qu'ils utilisent : SCSI ou IDE (bien que le SCSI soit généralement préféré). Une configuration des disques durs en RAID (Redundant Array of Independent Disks) est très fortement conseillée pour optimiser les performances sur des systèmes à plusieurs disques durs :

- Une configuration en RAID 1 est un minimum. Dans ce genre de configuration, chaque disque du système possède un disque mirroir qui est son exact duplicat. Les performances en lecture sont grandement améliorées car les deux disques peuvent être lus en parallèle (ce qui est particulièrement utile dans le cas de AssetCenter car la majorité des requêtes ne concernent que la lecture des données). L'effet de bord intéressant est que si l'un des disques est endommagé ou détruit, il n'y a pas de perte des données. Il suffit de remplacer le disque défaillant qui répliquera les informations de son disque mirroir. En contrepartie, il n'y a aucune amélioration en écriture sur ce genre de système et le nombre de disques durs doit toujours être pair.
- Une configuration en RAID 10 offre les meilleures performances. Elle combine les techniques de duplication et d'écriture en striping (les données sont écrites simultanément et de façon fragmentée entre les disques, ce qui génère une

amélioration très significative des performances en écriture) mais nécessite un investissement financier beaucoup plus important.

Le réseau

Comme il a été dit à de maintes reprises, les performances du réseau sont très importantes. Plus le réseau est rapide, plus la bande passante est importante et plus le nombre d'utilisateurs concurrents peut être élevé.

A propos des requêtes

Pour mieux comprendre l'impact des requêtes sur les performances, il est utile de détailler la procédure d'émission d'une requête par AssetCenter :

1. AssetCenter construit la requête et l'envoie à la base de données. Tous les paramètres de la requête ne sont pas envoyés. Les paramètres sont les valeurs de la requête dont le moteur à besoin pour déterminer le résultat de la requête. Par exemple, dans la clause WHERE suivante :

   ```
   WHERE lEmplDeptId=12345
   ```

   12345 est le paramètre.
2. L'optimiseur de requêtes de la base de données calcule ce qui est appelé un **Plan d'accès** (Access Plan) aux données, qui détermine la meilleure stratégie à adopter, au niveau du moteur de base de données, pour honorer la requête. Ce plan d'accès est renvoyé à AssetCenter. A ce niveau, les paramètres n'ont pas d'importance. Pour reprendre l'exemple précédent, tant que l'optimiseur de requêtes sait qu'il doit recevoir une valeur pour **lEmplDeptId**, il est en mesure de construire un plan d'accès approprié (par exemple en utilisant l'index sur cette valeur).
3. AssetCenter associe les paramètres à la requête et la renvoie au moteur de base de données.
4. Le moteur de base de données exécute la requête en utilisant le plan d'accès prédéfini, récupère le résultat et le renvoie à AssetCenter.

Note :

Sur certains moteurs de base de données, les étapes 1 et 2 ne sont pas réalisées. La requête est alors complètement et directement envoyée au moteur, sans phase de détermination d'un plan d'accès. L'impact sur les performances peut être important.

A propos des moteurs de base de données

Cette section contient quelques suggestions qui demeurent pertinentes indépendamment du moteur de base de données utilisé. L'efficacité de ces suggestions dépend néanmoins du moteur utilisé.

- Tout d'abord, les index et les données doivent être conservés dans des tablespaces différents. AssetCenter utilise fortement les index et les performances sont généralement améliorées s'ils sont séparés des données au niveau de la base. Le gain de performance est encore plus significatif quand les deux tablespaces résident sur deux disques durs physiquement séparés car la base de données pourra alors y accéder simultanément.
- Il est également recommandé de mettre régulièrement à jour les statistiques sur les tables. Cette opération peut être soit réalisée directement par l'administrateur de la base de données dans le cadre de ses opérations classiques de maintenance, soit par AssetCenter Serveur. Peregrine Systems, Inc recommande de laisser AssetCenter Serveur s'acquitter de cette tâche une fois par semaine. Les statistiques sur les tables permettent de déterminer la fréquence d'utilisation des données, ce qui permet à l'optimiseur d'améliorer l'accès aux données les plus fréquemment utilisées.
- Les index doivent être reconstruits régulièrement, tout particulièrement si les opérations d'insertion (INSERT) et d'effacement (DELETE) dans la base sont fréquents. Au fur et à mesure de l'exploitation, les index deviennent moins efficaces du fait de leur taille et du manque de continuité des données qu'ils contiennent. La reconstruction permet de régler ce problème. Si vous réalisez principalement des mises à jour sur les enregistrements (UPDATE), cette opération peut être réalisée uniquement tous les trimestres. Dans la majorité des cas, il est recommandé de demander à l'administrateur de base de données d'effectuer une reconstruction des index tous les mois.
- Des performances générales dégradées peuvent également être le symptome d'une recours à la mémoire virtuelle sur le serveur. Lorsque la mémoire

physique de la machine ne suffit plus, les systèmes d'exploitation peuvent utiliser un fichier d'échange (swapfile) utilisé comme de la mémoire virtuelle. Il y a alors un échange permanent entre ce fichier et la mémoire physique. Or les opérations de lecture et d'écriture sur un disque dur sont infiniment plus lentes que les mêmes opérations réalisées en mémoire. Si vous êtes confrontés à ce problème, vous pouvez soit augmenter la mémoire vive de la machine, soit allouer plus de mémoire vive au moteur de base de données.

Eliminer les locks (verrous) et les deadlocks (interblocages)

Lorsque plusieurs utilisateurs effectuent des opérations similaires résultant dans des transactions volumineuses impliquant les compteurs, il existe une forte probabilité que des transactions concurrentes provoquent un verrou ou un interblocage sur la table amCounter. Cette section traite des pistes de résolutions de ce problème sur les principaux moteurs de bases de données.

Avertissement :

Ces informations sont à l'usage des administrateurs de bases de données confirmés.

Microsoft SQL Server

La solution proposée ci-après déporte les compteurs dans d'autres tables de façon à minimiser les risques de verrous.

Etape 1 - Identification

Les compteurs qui posent problème sont très majoritairement ceux dont le compte est le plus élevé. La première étape consiste à les identifier. Pour ce faire :

1. Lancez l'analyseur de requêtes de Microsoft SQL Server,
2. Exécutez la requête suivante sur la base de données AssetCenter :

```
select * from amcounter
```

Le résultat de cette requête vous donne la valeur courante des compteurs stockés dans la table.

Note :

Pour le reste des étapes, nous utiliserons les compteurs **amItemReceived_ItemNo**, **amAsset_AssetTag**, **amTicket_TicketNo** et **amExpenseLine_ItemNo**.

Etape 2 - Copie de sauvegarde

Effectuez une copie de sauvegarde de la procédure stockée **up_GetCounterVal**.

Etape 3 - Création des tables pour les compteurs

Lancez un ordre CREATE TABLE pour chacun des compteurs qui doivent être déportés, en utilisant le format suivant :

```
CREATE TABLE [itam].[<NewTableName>] (
     [lValue] [int] IDENTITY (<IdentityStart>, 1) NOT NULL ,
     [luserspid] [int] NOT NULL
) ON [PRIMARY]
GO
```

Dans ce script :

- **NewTableName** est le nom de la table à créer, généralement du type **cnt_<Nom d'origine du compteur>**.
- **IdentityStart** est la valeur actuelle de la table d'origine du compteur.

Une requête SQL typique pour générer ces scripts sur la base de données est :

```
select 'CREATE TABLE [itam].[cnt_'+Identifier+'] ([lValue] [int] IDENTITY
('+ltrim(rtrim(str(lValue)))+',1) NOT NULL ,[luserspid] [int] NOT NULL)  O
N [PRIMARY]' as sqlquery from amCounter where lCounterid<>0
```

Note :

Dans cet exemple, on suppose que le propriétaire des tables est **itam**. Modifiez cette partie du script si nécessaire.

En prenant les exemples de compteurs définis précédemment, les ordres lancés seront les suivants:

```
CREATE TABLE [itam].[cnt_amAsset_AssetTag] (
     [lValue] [int] IDENTITY (17697, 1) NOT NULL ,
     [luserspid] [int] NOT NULL
) ON [PRIMARY]
GO
```

```
CREATE TABLE [itam].[cnt_amExpenseLine_ItemNo] (
     [lValue] [int] IDENTITY (10735, 1) NOT NULL ,
     [luserspid] [int] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [itam].[cnt_amItemReceived_ItemNo] (
     [lValue] [int] IDENTITY (4, 1) NOT NULL ,
     [luserspid] [int] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [itam].[cnt_amticket_ticketno] (
     [lValue] [int] IDENTITY (140000, 1) NOT NULL ,
     [luserspid] [int] NOT NULL
) ON [PRIMARY]
GO
```

Etape 4 - Création d'une procédure stockée

Il s'agit ici de créer la procédure stockée qui gère les compteurs. Nous vous invitons à utliser l'exemple ci-dessous en remplaçant le nom des compteurs par ceux qui vous intéressent et en adaptant la fin du script en fonction du nombre de compteurs que vous devez traiter. Veillez également à ce que le nom des compteurs apparaisse dans la clause NOT IN(), utilisée pour différencier les compteurs qui utilisent leur propre table de ceux qui utilisent la table **amCounter**.

```
sp_dropmessage 50895
GO
sp_addmessage 50895, 16,'up_GetCounterVal increment is %d which is not sup
ported when modified for non-locking optimization. Please edit the AMDB.IN
I file on your computer and add ImportCounterCache=1 in the current databa
se connexion description'
GO
CREATE PROCEDURE up_GetCounterVal @CounterName varchar(64), @CounterIncrem
ent int AS
BEGIN
  DECLARE @CounterIdent int
  'Change the NOT IN() list in this next line to reflect the names of the
new counters in your database.
  IF @CounterName NOT IN ('amItemReceived_ItemNo', 'amAsset_AssetTag', 'am
Ticket_TicketNo', 'amExpenseLine_ItemNo')
  BEGIN
    DECLARE amCounterLock CURSOR FOR SELECT lCounterId FROM amCounter WHER
E Identifier = @CounterName FOR UPDATE
    SELECT @CounterIdent = MAX(lCounterId) FROM amCounter WHERE Identifier
= @CounterName
    IF @CounterIdent IS NULL
      INSERT INTO amCounter (lCounterId, Identifier, Description, lValue,
dtLastModif) SELECT MAX(lCounterId) + 1, @CounterName, @CounterName, @Coun
terIncrement, GetDate() FROM amCounter
    ELSE
```

```
    BEGIN
        OPEN amCounterLock
        UPDATE amCounter SET lValue = lValue + @CounterIncrement, dtLastMo
dif = GetDate() WHERE lCounterId = @CounterIdent
    END
    SELECT lValue FROM amCounter WHERE Identifier = @CounterName
    IF @CounterIdent IS NOT NULL
      CLOSE amCounterLock
    /* END IF*/
    DEALLOCATE amCounterLock
  END
  ELSE
  BEGIN
  IF @CounterIncrement <> 1
    RAISERROR (50895, 17, @CounterIncrement)
  /* END IF */

-- See generation sql below
  --Add an IF block (next 5 lines) for EACH new counter table. Insert the
original counter name in all the bold faced areas.
  IF @CounterName = 'amTicket_TicketNo'
    BEGIN
      insert into cnt_amTicket_TicketNo (luserspid) values(@@SPID)
      SELECT max(lValue) FROM cnt_amTicket_TicketNo WITH (NOLOCK) where lu
serspid=@@SPID
    END
  IF @CounterName = 'amAsset_AssetTag'
    BEGIN
      insert into cnt_amAsset_AssetTag (luserspid) values(@@SPID)
      SELECT max(lValue) FROM cnt_amAsset_AssetTag WITH (NOLOCK) where lus
erspid=@@SPID
    END
  IF @CounterName = 'amExpenseLine_ItemNo'
    BEGIN
      insert into cnt_amExpenseLine_ItemNo (luserspid) values(@@SPID)
      SELECT max(lValue) FROM cnt_amExpenseLine_ItemNo WITH (NOLOCK) where
luserspid=@@SPID
    END
  IF @CounterName = 'amItemReceived_ItemNo'
    BEGIN
      insert into cnt_amItemReceived_ItemNo (luserspid) values(@@SPID)
      SELECT max(lValue) FROM cnt_amItemReceived_ItemNo WITH (NOLOCK) wher
e luserspid=@@SPID
    END
-- end of generation sql below
  END
END
GO
```

Astuce :

Le contenu de la clause NOT IN() et les segments IF utilisés pour chaque compteur peuvent être générés en utilisant respectivement les requêtes suivantes :

```
select '''cnt_'+Identifier+''', ' as sqlquery from amCounter where lCounterid<>0
```

et

```
select '  IF @CounterName =''cnt_'+Identifier+ '''
BEGIN
insert into  cnt_'+Identifier+' (luserspid) values(@@SPID)
SELECT max(lValue) FROM  cnt_'+Identifier+' WITH (NOLOCK) where luserspid=@@SPID
END' as sqlquery from amCounter where lCounterid<>0
```

Etape 5 - Test de la procédure stockée

Exécutez le script suivant :

```
exec up_getcounterval 'amAsset_AssetTag',1
SELECT COUNT(*) from  cnt_amAsset_AssetTag
exec up_getcounterval 'amItemReceived_ItemNo',1
SELECT COUNT(*) from  cnt_amItemReceived_ItemNo
exec up_getcounterval 'amTicket_TicketNo',1
SELECT COUNT(*) from  cnt_amticket_ticketno
exec up_getcounterval 'amExpenseLine_ItemNo',1
SELECT COUNT(*) from  cnt_amExpenseLine_ItemNo
```

Chaque nouvelle table de compteurs doit théoriquement avoir un enregistrement.

Etape 6 - Modification de la procédure stockée up_GetIt

Il peut y avoir également des verrous sur la génération des identifiants. Pour minimiser ce problème, modifiez la procédure stockée **up_GetId** comme suit :

```
DROP procedure up_GetId
GO
CREATE procedure up_GetId as
insert into LastId (Value)  values(@@SPID)
return 1+(SELECT max(IdSeed) FROM lastid  WITH (NOLOCK) where value=@@SPID)*30
/*
create procedure up_GetId as   if (select count(*) from LastId WITH (READUNCOMMITTED) )>20 delete from LastId  insert into LastId (Value) values(@@SPID)  return 1+@@IDENTITY*30
*/
GO
```

Etape 7 - Création d'une procédure stockée de nettoyage

Il s'agit de construire la procédure stockée qui nettoie régulièrement les tables de compteurs quand aucune insertion n'est en cours.

1 Récupérez l'identifiant pour chaque table de compteur en utilisant la requête suivante :

```
select id, name from sysobjects where name like 'cnt%'
```

2 Le résultat de cette requête est du type :

```
id          name
----------  ------------------------
1793441463  cnt_amAsset_AssetTag
2001442204  cnt_amExpenseLine_ItemNo
1745441292  cnt_amItemReceived_ItemNo
1505440437  cnt_amticket_ticketno
```

3 Générez la procédure stockée de nettoyage :

```
To generate all the main code block below for all deletes, run the sql
query
select 'delete from [cnt_'+Identifier+'] WITH (TABLOCKX)' as sqlquery f
rom amCounter where lCounterid<>0


DROP PROCEDURE AM_Cleanspeccounters
GO
CREATE PROCEDURE AM_Cleanspeccounters AS
BEGIN
     declare @locknb int

     set nocount on

     --while 1=1



     'Insert a code block (Next 1 line) for EACH new counter. Insert th
e appropriate ID or counter name in the bold faced areas.
     delete from cnt_amAsset_AssetTag  with (TABLOCKX)

     delete from cnt_amExpenseLine_ItemNo  with (TABLOCKX)

     delete from cnt_amItemReceived_ItemNo with (TABLOCKX)

     delete from cnt_amticket_ticketno with (TABLOCKX)

...

--Next line is to be commented when you define a new SQL Agent Task to
kick off the procedure each time on the server (it is in fact recommend
ed).
--waitfor delay '000:05:00'
```

```
--      'here the stored procedure is fired each 5 minutes

--      CONTINUE
--    end
end
```

4 Définisser un nouvel agent SQL qui démarre cette procédure toutes les heures sur le serveur.

5 Vérifiez le bon fonctionnement de l'ensemble. Après le passage de cette procédure, les tables de compteurs doivent être vides.

Etape 8 - Modification du fichier `amdb.ini`

1 Ouvrez le fichier **`amdb.ini`** de AssetCenter,

2 Identifiez la section **[Connexion]** de ce fichier et ajoutez la ligne suivante dans cette section :

```
ImportCounterCache=1
```

3 Répétez cette étape sur tous les clients qui accèdent à la base de données.

IMPORTANT :

Cette étape est cruciale et doit impérativement être réalisée.

Oracle

La solution proposée consiste à modifier la procédure stockée **UP_GETCOUNTERVALUE**.

Cette solution permet d'améliorer les performances dans les assistants et dans les applications externes qui utilisent les APIs AssetCenter. Des transactions volumineuses contenant des insertions d'enregistrements et d'objets faisant appel aux compteurs peuvent engendrer des verrous. Vous pouvez monitorer les verrous sur les compteurs en lançant l'ordre SQL suivant dans une vue **v$session** (en étant connecté comme administrateur) :

```
Select username, machine, sid, serial# from v$session where lockwait != Nu
ll;
```

Si la valeur renvoyée par cet ordre est importante, vous pouvez avoir recours à la solution décrite dans la suite de cette section. L'objectif de cette solution est de remplacer les compteurs par des séquences (qui ne sont pas sujettes aux verrous).

Note :

Si vous implémentez cette solution, les valeurs qui apparaîtront pour les compteurs dans AssetCenter (menu Outils/ Administration/ Compteurs) seront erronées.

Etape 1 - Préparation

1. Vérifiez que personne n'utilise la base de données. Ce pré-requis est indispensable.
2. Réalisez une copie de sauvegarde de la base de données et testez cette copie.

Etape 2 - Remplacement des compteurs

1. Connectez-vous à SQLPlus en tant que propriétaire de la base AssetCenter
2. Identifiez les compteurs à remplacer en exécutant la requête suivante :

```
Select * from amcounter;
```

3. Déplacez les compteurs vers des séquences :

```
SET CHARWIDTH 200;
SET PAGESIZE 9999;
SET HEADING OFF;
SET ECHO OFF;
Spool seqgen.sql

SELECT 'CREATE SEQUENCE ' || IDENTIFIER || ' INCREMENT BY 1 NOMAXVALUE
MINVALUE '|| (LVALUE + 1) || ' NOCYCLE CACHE 20 ORDER;' AS TEXT FROM AM
COUNTER where identifier IS NOT NULL;

Spool off
```

Vous obtenez un résultat similaire à celui ci-après :

```
CREATE SEQUENCE amEmplDept_BarCode INCREMENT BY 1 NOMAXVALUE MINVALUE 6
0155 NOCYCLE CACHE 20 ORDER;  CREATE SEQUENCE amLocation_BarCode INCREM
ENT BY 1 NOMAXVALUE MINVALUE 31 NOCYCLE CACHE 20 ORDER;  CREATE SEQUENC
E amCategory_BarCode INCREMENT BY 1 NOMAXVALUE MINVALUE 101 NOCYCLE CAC
HE 20 ORDER;  CREATE SEQUENCE amProduct_CatalogRef INCREMENT BY 1 NOMAX
VALUE MINVALUE 34076 NOCYCLE CACHE 20 ORDER; ...
CREATE SEQUENCE amOutputEvent_EventNo INCREMENT BY 1 NOMAXVALUE MINVALU
E 11 NOCYCLE CACHE 20 ORDER;

49 rows selected.
```

4. Le fichier **seqgen.sql** peut alors être édité pour ne conserver que les clauses CREATE SEQUENCE

Etape 3 - Mise à jour de la procédure stockée

Il s'agit de remplacer la procédure stockée **UP_GETCOUNTERVALUE** pour utiliser les séquences.

1 Exécutez un script de génération des appels des séquences dans la procédure stockée. Par exemple :

```
spool UP_GETCOUNTERVAL_gen.SQL
SET CHARWIDTH 200;
SELECT 'ELSIF CounterName = '||CHR(39)|| IDENTIFIER ||CHR(39)||' THEN '

|| ' SELECT '||IDENTIFIER||'.NEXTVAL INTO CounterValue FROM DUAL;'
FROM AMCOUNTER;
spool off;
```

Vous obtenez un résultat similaire à celui ci-après :

```
ELSIF CounterName = '' THEN  SELECT .NEXTVAL INTO CounterValue FROM DUA
L;
ELSIF CounterName = 'amEmplDept_BarCode' THEN  SELECT amEmplDept_BarCod
e.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amLocation_BarCode' THEN  SELECT amLocation_BarCod
e.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amCategory_BarCode' THEN  SELECT amCategory_BarCod
e.NEXTVAL INTO CounterValue FROM DUAL;
...
```

2 Effectuez les modifications suivantes sur le résultat généré :

1 Modifiez le script de façon à ce qu'il traite correctement les conditions IF

2 Effacez la ligne qui traite un compteur vide :

```
ELSIF CounterName = '' THEN  SELECT .NEXTVAL INTO CounterValue FROM
DUAL;
```

3 Ajoutez la déclaration et la fin de la procédure stockée.

Le résultat final est similaire à celui ci-après :

```
CREATE OR REPLACE PROCEDURE UP_GETCOUNTERVAL (CounterName IN VARCHAR
2, CounterIncrement IN NUMBER, CounterValue OUT NUMBER) AS
CounterIdent NUMBER;
BEGIN
IF CounterIncrement <> 1 THEN
RAISE_APPLICATION_ERROR (-20004, 'Error in counter increment with th
e Up_getcounterval stored procedure altered to minimize locking, You
must use a counter increment of 1, please add the importcountercache
=1 in the amdb.ini file, in the parameters of your current database
connection');
ELSIF CounterName = 'amEmplDept_BarCode' THEN  SELECT amEmplDept_Bar
Code.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amLocation_BarCode' THEN  SELECT amLocation_Bar
Code.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amCategory_BarCode' THEN  SELECT amCategory_Bar
Code.NEXTVAL INTO CounterValue FROM DUAL;
```

```
ELSIF CounterName = 'amProduct_CatalogRef' THEN  SELECT amProduct_Ca
talogRef.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amProduct_BarCode' THEN  SELECT amProduct_BarCo
de.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amProdCompo_ItemNo' THEN  SELECT amProdCompo_It
emNo.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amAsset_AssetTag' THEN  SELECT amAsset_AssetTag
.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amConnection_CnxNo' THEN  SELECT amConnection_C
nxNo.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amCompany_Code' THEN  SELECT amCompany_Code.NEX
TVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amBudget_BudgetNo' THEN  SELECT amBudget_Budget
No.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amContract_Ref' THEN  SELECT amContract_Ref.NEX
TVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amConsUse_ItemNo' THEN  SELECT amConsUse_ItemNo
.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amStock_Code' THEN  SELECT amStock_Code.NEXTVAL
INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amProdReserv_ItemNo' THEN  SELECT amProdReserv_
ItemNo.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amProject_Code' THEN  SELECT amProject_Code.NEX
TVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amTicket_TicketNo' THEN  SELECT amTicket_Ticket
No.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amWorkOrder_WONo' THEN  SELECT amWorkOrder_WONo
.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amNews_Name' THEN  SELECT amNews_Name.NEXTVAL I
NTO CounterValue FROM DUAL;
ELSIF CounterName = 'amKnowlBase_Code' THEN  SELECT amKnowlBase_Code
.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amDecisionTree_Code' THEN  SELECT amDecisionTre
e_Code.NEXTVAL INTO CounterValue FROM DUAL
ELSIF CounterName = 'amDownTimePeriod_Code' THEN  SELECT amDownTimeP
eriod_Code.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amRequest_ReqNumber' THEN  SELECT amRequest_Req
Number.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amReqLine_ItemNo' THEN  SELECT amReqLine_ItemNo
.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amEstimate_EstimNumber' THEN  SELECT amEstimate
_EstimNumber.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amEstimLine_ItemNo' THEN  SELECT amEstimLine_It
emNo.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amPOrder_PONumber' THEN  SELECT amPOrder_PONumb
er.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amPOrdLine_ItemNo' THEN  SELECT amPOrdLine_Item
No.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amDeliv_DelivNumber' THEN  SELECT amDeliv_Deliv
Number.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amItemReceived_ItemNo' THEN  SELECT amItemRecei
ved_ItemNo.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amItemReturned_ItemNo' THEN  SELECT amItemRetur
ned_ItemNo.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amInvoice_InvoiceNumber' THEN  SELECT amInvoice
_InvoiceNumber.NEXTVAL INTO CounterValue FROM DUAL;
```

```
ELSIF CounterName = 'amAdjustment_ItemNo' THEN  SELECT amAdjustment_
ItemNo.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amFieldAdjust_ItemNo' THEN  SELECT amFieldAdjus
t_ItemNo.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amFieldAdjustModel_ItemNo' THEN  SELECT amField
AdjustModel_ItemNo.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amReturnEnv_Code' THEN  SELECT amReturnEnv_Code
.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amLoan_Code' THEN  SELECT amLoan_Code.NEXTVAL I
NTO CounterValue FROM DUAL;
ELSIF CounterName = 'amCostCenter_Code' THEN  SELECT amCostCenter_Co
de.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amExpenseLine_ItemNo' THEN  SELECT amExpenseLin
e_ItemNo.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amWfScheme_Ref' THEN  SELECT amWfScheme_Ref.NEX
TVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amWfOrgRole_Ref' THEN  SELECT amWfOrgRole_Ref.N
EXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amWfActivity_Ref' THEN  SELECT amWfActivity_Ref
.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amWfActivAlarm_Ref' THEN  SELECT amWfActivAlarm
_Ref.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amWfEvent_Ref' THEN  SELECT amWfEvent_Ref.NEXTV
AL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amWfTransition_Ref' THEN  SELECT amWfTransition
_Ref.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amWfWorkItem_Ref' THEN  SELECT amWfWorkItem_Ref
.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amWfSyncPoint_Ref' THEN  SELECT amWfSyncPoint_R
ef.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amWfInstance_Ref' THEN  SELECT amWfInstance_Ref
.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amInputEvent_EventNo' THEN  SELECT amInputEvent
_EventNo.NEXTVAL INTO CounterValue FROM DUAL;
ELSIF CounterName = 'amOutputEvent_EventNo' THEN  SELECT amOutputEve
nt_EventNo.NEXTVAL INTO CounterValue FROM DUAL;
ELSE
SELECT MAX(lCounterId) INTO CounterIdent FROM amCounter WHERE Identi
fier = CounterName;
IF CounterIdent IS NULL THEN
SELECT MAX(lCounterId)+1 INTO CounterIdent FROM amCounter;
INSERT INTO amCounter (lCounterId, Identifier, Description, lValue,
dtLastModif) VALUES (CounterIdent, CounterName, CounterName, Counter
Increment, SYSDATE);
ELSE
UPDATE amCounter SET lValue = lValue + CounterIncrement, dtLastModif
= SYSDATE WHERE lCounterId = CounterIdent;
END IF;
SELECT lValue INTO CounterValue FROM amCounter WHERE Identifier = Co
unterName;
END IF;
END;

/
```

3 Faites une copie de sauvegarde de la procédure **UP_GETCOUNTERVALUE** originale.
4 Mettez à jour la procédure stocké en en utilisant le script précédemment généré.

Etape 4 - Modification du fichier `amdb.ini`

1 Ouvrez le fichier **`amdb.ini`** de AssetCenter,
2 Identifiez la section [**Connexion**] de ce fichier et ajoutez la ligne suivante dans cette section :

```
ImportCounterCache=1
```

3 Répétez cette étape sur tous les clients qui accèdent à la base de données.

IMPORTANT :

Cette étape est cruciale et doit impérativement être réalisée.

DB2

La solution proposée consiste à modifier la procédure stockée **UP_GETCOUNTERVALUE**.

Cette solution permet d'améliorer les performances dans les assistants et dans les applications externes qui utilisent les APIs AssetCenter. Des transactions volumineuses contenant des insertions d'enregistrements et d'objets faisant appel aux compteurs peuvent engendrer des verrous.

Si la valeur renvoyée par cet ordre est importante, vous pouvez avoir recours à la solution décrite dans la suite de cette section. L'objectif de cette solution est de remplacer les compteurs par des séquences (qui ne sont pas sujettes aux verrous).

Note :

Si vous implémentez cette solution, les valeurs qui apparaîtront pour les compteurs dans AssetCenter (menu Outils/ Administration/ Compteurs) seront erronées.

Etape 1 - Préparation

1. Vérifiez que personne n'utilise la base de données. Ce pré-requis est indispensable.
2. Réalisez une copie de sauvegarde de la base de données et testez cette copie.

Etape 2 - Remplacement des compteurs

1. Connectez-vous à CLP en tant que propriétaire de la base AssetCenter
2. Identifiez les compteurs à remplacer en exécutant la requête suivante :

```
Select * from amcounter;
```

3. Déplacez les compteurs vers des séquences :

```
SELECT 'CREATE SEQUENCE ' || IDENTIFIER || ' INCREMENT BY 1 NOMAXVALUE
START WITH '|| CHAR(LVALUE + 1) || ' NOCYCLE CACHE 20 ORDER' AS TEXT FR
OM AMCOUNTER where identifier IS NOT NULL;
```

Vous obtenez un résultat similaire à celui ci-après :

```
CREATE SEQUENCE amInputEvent_EventNo INCREMENT BY 1 NOMAXVALUE START WI
TH 1001 NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amOutputEvent_EventNo INCREMENT BY 1 NOMAXVALUE START W
ITH 1001 NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amWfActivAlarm_Ref INCREMENT BY 1 NOMAXVALUE START WITH
1001 NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amWfActivity_Ref INCREMENT BY 1 NOMAXVALUE START WITH 1
005 NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amWfEvent_Ref INCREMENT BY 1 NOMAXVALUE START WITH 1011
NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amWfInstance_Ref INCREMENT BY 1 NOMAXVALUE START WITH 1
029 NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amWfOrgRole_Ref INCREMENT BY 1 NOMAXVALUE START WITH 10
01 NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amWfScheme_Ref INCREMENT BY 1 NOMAXVALUE START WITH 101
0 NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amWfTransition_Ref INCREMENT BY 1 NOMAXVALUE START WITH
1009 NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amWfWorkItem_Ref INCREMENT BY 1 NOMAXVALUE START WITH 1
029 NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amCable_CableTag INCREMENT BY 1 NOMAXVALUE START WITH 1
001 NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amCableLink_BarCode INCREMENT BY 1 NOMAXVALUE START WIT
H 1001 NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amColorCode_Code INCREMENT BY 1 NOMAXVALUE START WITH 1
001 NOCYCLE CACHE 20 ORDER
CREATE SEQUENCE amTraceHistory_BarCode INCREMENT BY 1 NOMAXVALUE START
WITH 1001 NOCYCLE CACHE 20 ORDER
......
73  rows selected.
```

4 Le fichier **seqgen.sql** peut alors être édité pour ne conserver que les clauses CREATE SEQUENCE

Etape 3 - Mise à jour de la procédure stockée

Il s'agit de remplacer la procédure stockée **UP_GETCOUNTERVALUE** pour utiliser les séquences.

1 Exécutez un script de génération des appels des séquences dans la procédure stockée. Par exemple :

```
db2 => SELECT 'ELSEIF CounterName = '||CHR(39)|| IDENTIFIER ||CHR(39)||
' THEN Values NEXTVAL FOR '||IDENTIFIER||' INTO CounterValue ' FROM AMC
OUNTER
```

Vous obtenez un résultat similaire à celui ci-après :

```
ELSEIF CounterName = 'amInputEvent_EventNo' THEN Values NEXTVAL FOR amI
nputEvent_EventNo INTO CounterValue;
ELSEIF CounterName = 'amOutputEvent_EventNo' THEN Values NEXTVAL FOR am
OutputEvent_EventNo INTO CounterValue;
ELSEIF CounterName = 'amWfActivAlarm_Ref' THEN Values NEXTVAL FOR amWfA
ctivAlarm_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfActivity_Ref' THEN Values NEXTVAL FOR amWfAct
ivity_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfEvent_Ref' THEN Values NEXTVAL FOR amWfEvent_
Ref INTO CounterValue;
ELSEIF CounterName = 'amWfInstance_Ref' THEN Values NEXTVAL FOR amWfIns
tance_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfOrgRole_Ref' THEN Values NEXTVAL FOR amWfOrgR
ole_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfScheme_Ref' THEN Values NEXTVAL FOR amWfSchem
e_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfTransition_Ref' THEN Values NEXTVAL FOR amWfT
ransition_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfWorkItem_Ref' THEN Values NEXTVAL FOR amWfWor
kItem_Ref INTO CounterValue;
ELSEIF CounterName = 'amCable_CableTag' THEN Values NEXTVAL FOR amCable
_CableTag INTO CounterValue;
ELSEIF CounterName = 'amCableLink_BarCode' THEN Values NEXTVAL FOR amCa
bleLink_BarCode INTO CounterValue;
ELSEIF CounterName = 'amColorCode_Code' THEN Values NEXTVAL FOR amColor
Code_Code INTO CounterValue;
ELSEIF CounterName = 'amTraceHistory_BarCode' THEN Values NEXTVAL FOR a
mTraceHistory_BarCode INTO CounterValue;
ELSEIF CounterName = 'amCatalog_Code' THEN Values NEXTVAL FOR amCatalog
_Code INTO CounterValue;
ELSEIF CounterName = 'InternalRef' THEN Values NEXTVAL FOR InternalRef
INTO CounterValue;
ELSEIF CounterName = 'amContract_Ref' THEN Values NEXTVAL FOR amContrac
t_Ref INTO CounterValue;
.....
```

2 Effectuez les modifications suivantes sur le résultat généré :

1. Modifiez le script de façon à ce qu'il traite correctement les conditions IF
2. Effacez la ligne qui traite un compteur vide
3. Ajoutez la déclaration et la fin de la procédure stockée.

Le résultat final est similaire à celui ci-après :

```
CREATE PROCEDURE UP_GETCOUNTERVAL (IN CounterName VARCHAR(64), IN Co
unterIncrement INTEGER, OUT CounterValue INTEGER) LANGUAGE SQL BEGIN
DECLARE CounterIdent INTEGER; DECLARE ERR_MSG VARCHAR(255);
IF CounterIncrement <> 1 THEN
SELECT 'Up_getcounterval stored procedure altered to minimize lockin
g, You must use a counter increment of 1, please add the importcount
ercache=1 in the amdb.ini file, in the parameters of your current da
tabase connection' INTO ERR_MSG FROM SYSIBM.SYSDUMMY1;
SIGNAL SQLSTATE '75002' SET MESSAGE_TEXT = ERR_MSG;
ELSEIF CounterName = 'amInputEvent_EventNo' THEN Values NEXTVAL FOR
amInputEvent_EventNo INTO CounterValue;
ELSEIF CounterName = 'amOutputEvent_EventNo' THEN Values NEXTVAL FOR
amOutputEvent_EventNo INTO CounterValue;
ELSEIF CounterName = 'amWfActivAlarm_Ref' THEN Values NEXTVAL FOR am
WfActivAlarm_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfActivity_Ref' THEN Values NEXTVAL FOR amWf
Activity_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfEvent_Ref' THEN Values NEXTVAL FOR amWfEve
nt_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfInstance_Ref' THEN Values NEXTVAL FOR amWf
Instance_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfOrgRole_Ref' THEN Values NEXTVAL FOR amWfO
rgRole_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfScheme_Ref' THEN Values NEXTVAL FOR amWfSc
heme_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfTransition_Ref' THEN Values NEXTVAL FOR am
WfTransition_Ref INTO CounterValue;
ELSEIF CounterName = 'amWfWorkItem_Ref' THEN Values NEXTVAL FOR amWf
WorkItem_Ref INTO CounterValue;
ELSEIF CounterName = 'amCable_CableTag' THEN Values NEXTVAL FOR amCa
ble_CableTag INTO CounterValue;
ELSEIF CounterName = 'amCableLink_BarCode' THEN Values NEXTVAL FOR a
mCableLink_BarCode INTO CounterValue;
ELSEIF CounterName = 'amColorCode_Code' THEN Values NEXTVAL FOR amCo
lorCode_Code INTO CounterValue;
ELSEIF CounterName = 'amTraceHistory_BarCode' THEN Values NEXTVAL FO
R amTraceHistory_BarCode INTO CounterValue;
ELSEIF CounterName = 'amCatalog_Code' THEN Values NEXTVAL FOR amCata
log_Code INTO CounterValue;
ELSEIF CounterName = 'InternalRef' THEN Values NEXTVAL FOR InternalR
ef INTO CounterValue;
ELSEIF CounterName = 'amContract_Ref' THEN Values NEXTVAL FOR amCont
ract_Ref INTO CounterValue;
ELSE SELECT MAX(lCounterId) INTO CounterIdent FROM amCounter WHERE I
dentifier = CounterName;
IF CounterIdent IS NULL THEN SELECT MAX(lCounterId)+1 INTO CounterId
ent FROM amCounter;
INSERT INTO amCounter (lCounterId, Identifier, Description, lValue,
dtLastModif) VALUES (CounterIdent, CounterName, CounterName, Counter
```

```
Increment, CURRENT TIMESTAMP);
ELSE UPDATE amCounter SET lValue = lValue + CounterIncrement, dtLast
Modif = CURRENT TIMESTAMP WHERE lCounterId =CounterIdent;
END IF;
SELECT lValue INTO CounterValue FROM amCounter WHERE Identifier = Co
unterName;
END IF;
END
GO
```

3 Faites une copie de sauvegarde de la procédure **UP_GETCOUNTERVALUE** originale.

4 Mettez à jour la procédure stocké en en utilisant le script précédemment généré.

Etape 4 - Modification du fichier `amdb.ini`

1 Ouvrez le fichier **`amdb.ini`** de AssetCenter,

2 Identifiez la section **[Connexion]** de ce fichier et ajoutez la ligne suivante dans cette section :

```
ImportCounterCache=1
```

3 Répétez cette étape sur tous les clients qui accèdent à la base de données.

IMPORTANT :

Cette étape est cruciale et doit impérativement être réalisée.


Peregrine
SYSTEMS