Chargement d'un enregistrement (I)

Par
Tim TONOOKA, Technicien Support Clients 4DUS
Note technique 4D-200007-21-FR
Version 1

Date 1 Juillet 2000

Résumé

Cette note technique (1er volet sur trois) initie aux subtilités du mécanisme de chargement des enregistrements en mémoire.
Les volets suivants vous permettront d'appréhender complètement le processus.

4D Notes techniques

Introduction

Dans les versions précédentes de 4D, le mécanisme du chargement des enregistrements ne posait pas de problème particulier car les objets utilisés n'occupaient pas une grande place en mémoire.

L'apparition des BLOBS (Binary Large Objects), en version 6, qui peuvent atteindre une taille maximale de 2Go, méritent de connaître le principe de l'allocation mémoire réalisé par 4D lors du chargement des enregistrements.

Connaître ce principe vous permettra d'optimiser le taux d'occupation de la mémoire ainsi que la manière d'améliorer le trafic réseau dans le cadre de l'utilisation de 4D serveur.

Présentation

Cette note technique est décomposée de la manière suivante :

VI La différence de comportement du mécanisme de chargement des enregistrements entre 4D mono poste et 4D serveur.

VII Explication sur la quantité de la mémoire nécessaire pour le chargement et la manipulation d'enregistrement de grande taille.

1) Durant l'opération de chargement d'enregistrement.
2) Durant l'opération de sauvegarde d'un enregistrement existant.
3) La sauvegarde d'un nouvel enregistrement.

Poids d'un enregistrement

Comment un enregistrement peut-il atteindre un poids important ?

Les types de données : alpha, réel, entier, entier long, date, heure et booléen ont un poids faible (par exemple : un maximum de 255 caractères pour une variable alpha). Pour les 4 types suivants, leur taille (poids) peut-être beaucoup plus importante.

Champ BLOB : peut atteindre 2 Go de données.

Champ IMAGE : peut atteindre 2 Go de données

Champ TEXTE : un champ texte peut contenir 32000 caractères au maximum. Si une table contient plusieurs champs de texte (par exemple 32), vous obtiendrez des enregistrements d'un poids proche du méga bytes. Il faut savoir qu'une table peut contenir, au maximum, 511 champs.

Champs RACINE et sous-tables : les enregistrements, dans une sous-table, associés à un enregistrement père, font partie intégrante de l'enregistrement. Il est essentiel de connaître ce mécanisme qui n'existe pas dans d'autres bases de données (ces sous-enregistrements sont considérés comme des enregistrements simples). Tous les types de champs, pour les sous-tables, sont acceptés (sauf le type champ RACINE), cependant il n'est pas recommandé d'utiliser les types suivants : Texte, Image, Blob, dans les sous-tables, de façon à éviter l'alourdissement de l'enregistrement père pour les raisons expliquées ci dessus.

Si vous utilisez une base contenant des enregistrements de petite taille et que vous allouez une quantité de mémoire raisonnable à 4D, vous ne rencontrerez pas de problème. Par contre si votre base comporte des champs de type Blob ou Texte et que vos enregistrements évoluent en taille, cela risque de poser des soucis à 4D dans le cadre de la manipulation de ces enregistrements.

Cette note technique examine le mécanisme de fonctionnement de chargement des enregistrements et la réaction de 4D en fonction de la taille de la mémoire qui lui a été allouée.

I - Comment 4D découpe la mémoire qui lui a été allouée ?

4D découpe celle-ci en trois sections : une partie réservée au moteur, une zone contenant le cache (les données) et enfin une partie que l'on appelle la mémoire principale.

1 - La partie mémoire moteur.

Au moment du lancement de 4D cette partie est occupée par le moteur de 4D (4D et certaines ressources). Ces objets, chargés en mémoire, ne sont pas purgeables et restent présents pendant toute la durée de vie de 4D. Ce qui revient à dire que cet emplacement ne peut être alloué à une autre application pendant que 4D est lancé.

2 - La zone du cache.

Au démarrage de 4D un bloc de mémoire, insécable et contigu, est alloué. Il constitue le cache. Celui-ci contient :

- Les enregistrements qui proviennent ou partent du disque dur, la table des bits "bit map"
- Les tables d'adresses des enregistrements (primaires et secondaires)
- Les pages d'index (primaires et secondaires).

3 - La mémoire principale (User HEAP).

Cette partie gère différents objets qui sont utilisés par l'application et qui sont chargés ou déchargés du disque selon leur utilisation. Cette zone contient :

- Une copie de l'enregistrement courant (dite partielle) de chacunes des tables de chaque process. Nous verrons son rôle un peu plus tard.
- Tous les objets de structure (formulaires, méthodes, énumérations) ainsi que les sélections courantes, les sélections temporaires et les ensembles.
- Toutes les variables interprocess et process, ainsi que les variables locales qui ont les types suivants : Blob, Texte, Image et Tableaux.
- Le code des plug-ins.
- Différentes ressources.
- Une copie de la fenêtre de 4D qui sera utilisée, quand l'option "rafraîchissement plus rapide" des propriétés de la base sera cochée, pour améliorer les performances de l'affichage

- La pile associée (STACK) à chaque process qui stocke les variables locales (en mode compilé seulement), le contexte du process qui assure le lien entre les différentes méthodes, certaines variables internes à 4D.

Le deuxième volet de cette note technique précise, de manière détaillée, l'occupation et le découpage des parties mémoires ainsi qu'une méthode permettant de mesurer et d'ajuster au mieux ces allocations pour optimiser les performances de l'application.

II - De quelle façon 4D manipule les enregistrements ?

Les enregistrements d'une base de données sont physiquement enregistrés dans un fichier qui se trouve sur le disque de l'ordinateur. La vitesse d'exécution du CPU est largement supérieure au mécanisme d'accès au disque qui est géré par les pilotes. Par conséquent, il a fallu trouver une solution pour optimiser la vitesse de transfert.

1 - Qu'est ce qu'un cache et quel est son rôle ?

L'une des solutions, pour améliorer les performances, est de réserver une partie de la mémoire pour faciliter les accès aux disques. Celle-ci sera appelée CACHE. Elle stockera les enregistrements en attente de transfert dans le sens disque vers CPU (lecture) ou vis versa (écriture). Ceci a pour effet de réduire les accès vers le disque car un certain nombre d'enregistrements se trouveront toujours présents dans le cache.

Plusieurs caches sont présents sur une machine. Par exemple : le cache réservé au disque qui se trouve sur le contrôleur de celui-ci et un cache CPU qui se trouve en RAM (ils sont tous les deux maintenus par le système ou le hardware). Les données transitent par ces différents caches.

Il faut souligner, à cause du mécanisme des caches, que si vous réalisez des manipulations sur les enregistrements comme la sauvegarde (avec 4D), une partie de ces enregistrements se trouve toujours dans le cache et n'est pas obligatoirement enregistrée physiquement sur le disque.

C'est le système qui gère ce mécanisme. C'est pour cela qu'en cas de coupure d'électricité, vous pouvez perdre le contenu des caches qui ne sont pas encore écrits, et par conséquent, perdre des données. Si vous voulez éviter ce type de problème, il faut mettre en place un système de BACKUP et protéger la machine par un onduleur.

Les données transitent d'un cache à l'autre grâce au système. 4D maintient son propre cache. Il faut souligner, dans le cadre d'un 4D serveur, que le cache de 4D se trouve sur la machine serveur et qu'il n'y a aucun autre cache sur les postes clients.

Quand vous effectuez des manipulations sur les enregistrements (comme le chargement), 4D vérifie, dans un premier temps, si l'enregistrement est présent dans le cache, sinon il le récupère sur le disque et le met dans le cache.

2 - Comment sont gérés les enregistrements ?

Lorsque 4D charge un enregistrement en mémoire, il charge deux copies de celui-ci indépendamment de la copie primaire qui se trouve dans le cache.

Nous parlerons donc d'une copie primaire (celle qui reste inchangée dans le cache), d'une autre copie qui forme l'enregistrement courant et d'une troisième copie (partielle) qui sert de référence aux éventuelles modifications apportées à un enregistrement.

Voici quelques détails sur ces différentes copies:

a) La copie primaire.

Elle est entièrement contenue dans le cache.

b) La copie désignée comme enregistrement courant.

Une partie est contenue dans le cache tandis que le reste se trouve en mémoire principale (User Heap). Nous verrons ce mécanisme en détail un peu plus tard.

c) La copie partielle.

Elle s'appelle aussi copie de référence. Elle permet, lorsque vous utilisez la fonction "Ancien", de faire la comparaison pour détecter les différences (modifications apportées). Celle-ci ne contient pas les champs : Texte, Image et Blob. Ces champs ne sont pas dupliqués pour des raisons d'encombrement de la mémoire et d'optimisation.
Cette troisième copie se trouve dans la mémoire principale (User heap).

Le mécanisme de la gestion des enregistrements est différent suivant la version de 4D. La différence se situe entre 4D serveur et tous les autres produits (4D monoposte, 4D runtime, 4D runtime classic, 4D engine). Ce qui revient à dire que le mécanisme est différent si les enregistrements transitent par un réseau.

Lorsqu'un enregistrement est chargé en mémoire, pour un process et une table de données, il devient l'enregistrement courant de la table pour ce process.
Il n'existe qu'un seul enregistrement courant par table.

Versions monopostes:

Le détail du mécanisme correspond aux paragraphes a), b), c), ci-dessus.

Versions multi postes:

En mode client serveur le mécanisme de chargement des enregistrements est partagé entre la machine client et la machine serveur.

Deux cas peuvent se présenter :

- Les enregistrements sont utilisés uniquement sur le serveur pour les triggers et les procédures stockées. Le détail du mécanisme correspond aux paragraphes a), b), c), ci-dessus.
- Les enregistrements sont utilisés côté client.

Nous vous rappelons que le client ne possède pas de cache. Ainsi quand un enregistrement doit être chargé sur le client, il est recopié du cache (du serveur) dans la mémoire principale (user heap) de l'utilisateur via le réseau. Ce qui veut dire que l'enregistrement se trouve au niveau de la RAM du client et que les éventuelles modifications se feront sur son poste et ne transiteront sur le réseau qu'au moment de la sauvegarde de l'enregistrement.

3 - Quels sont les facteurs qui déclenchent la sauvegarde d'un enregistrement ?

Lorsque l'enregistrement se trouve dans le cache et dans la mémoire principale (user heap), il subit certaines modifications et est déchargé de la mémoire uniquement dans les conditions suivantes :

- Une procédure stockée donne l'ordre de sauvegarde de l'enregistrement.
- L'enregistrement courant associé à la table et a un process, change.
- Le process qui possède l'enregistrement se termine.

4 - Quelle copie d'enregistrement est envoyée lors d'un accès simultané par différents process?

Si un autre 4D client (ou un autre process sur la même machine) veut accéder à un enregistrement en cours de modification, une copie est envoyée à ce client (ou process). Alors une question se pose : d'où provient la copie ?

Mécanisme :

Il faut savoir que cette copie est l'image de la dernière sauvegarde de l'enregistrement sur le disque. En d'autres termes, la copie provient soit du cache (copie primaire du serveur pour le cas du serveur client), si l'enregistrement est présent dans celui-ci, soit du disque dur.
Il faut aussi en déduire que si un autre utilisateur charge l'enregistrement que vous êtes en train de modifier. Il ne verra les modifications que si vous les sauvegardez avant qu'il charge l'enregistrement. Comment cela fonctionne ?

Mécanisme :

Lorsque vous sauvegardez un enregistrement, à partir de 4D client, c'est la copie de celui-ci (qui se trouve dans la mémoire principale " user heap " de votre machine) qui est envoyée au serveur via le réseau. Puis, s'il y a un trigger avec l'évènement adéquat, il est exécuté. Enfin l'enregistrement est réellement considéré comme sauvegardé et c'est donc cette copie (copie primaire) qui sera envoyée aux utilisateurs.

5 - Présentation des triggers

Un trigger est rattaché à une table. Il est représenté sous forme d'une méthode qui accueillera les différents tests sur les évènements moteurs.

Ci-dessous nous allons vous présenter le rôle des triggers et les évènements moteurs auxquels ils réagissent ainsi que leur mécanisme.

a) Les évènements moteurs des triggers sont :

- Sur la sauvegarde d'un nouvel enregistrement : exécutés si un enregistrement vient d'être créé.
- Sur la sauvegarde d'un enregistrement : exécutés si un enregistrement vient d'être modifié.
- Sur la suppression d'un enregistrement : exécutés si un enregistrement est supprimé.
- Sur le chargement d'un enregistrement : exécutés après le chargement d'un enregistrement.

b) Leurs rôles :

Les triggers sont utilisés pour renforcer l'intégrité de votre base et permettre la mise à jour des enregistrements, juste avant leur sauvegarde sur les disques, mais après la phase de validation.

Il faut entendre par cela que ce n'est pas le développeur (par programmation) ou l'utilisateur (par l'action de validation) qui déclenche un trigger, mais le moteur de 4D qui, juste avant l'écriture dans le cache de l'enregistrement, active le mécanisme.

Le mécanisme des triggers :

Un trigger se déclenchera lors de la manipulation d'enregistrement. Ce mécanisme intervient au niveau le plus bas du moteur de 4D et n'est géré que par le moteur. Cela veut dire qu'il est impossible d'intervenir dans ce processus.

La seule possibilité donnée au développeur est le déclenchement (ou non) du trigger grâce à l'onglet " triggers " des propriétés de la table. Comme nous l'avons vu plus haut, le trigger est réceptif à quatre évènements moteurs.

Il travaillera sur une copie de l'enregistrement comme suit :

- Sur la copie provenant du cache dans le cas de l'événement moteur sur " chargement enregistrement ".
- Sur la copie qui remplacera celle du cache dans le cas des autres événements moteurs.

Cas particulier : les versions multipostes.

Une copie de l'enregistrement est envoyée au serveur. Puis le serveur la traite et exécute le trigger. Ceci provoque alors une différence entre la copie du client et celle du serveur.

À partir de la version 605 de 4D, 4D renvoie une copie au poste client, via le réseau, pour remplacer celle qui se trouve dans la mémoire principale (user heap) du client, mais ne provoque pas l'évènement moteur sur " chargement enregistrement ".

Remarque :
Si vous utilisez la commande "LIBERER ENREGISTREMENT" à partir de 4D client, cela a pour effet de libérer l'enregistrement courant sur le client, mais l'image de l'enregistrement qui se trouve dans le cache et dans la mémoire principale (heap user) du serveur n'est pas déchargée même si nous tentons de réduire la sélection à zéro sur le 4D client.

III - Explication sur le mécanisme de chargement des enregistrements

1 - Notion d'intégrité.

Lors du chargement d'un enregistrement (en mode lecture écriture), il faut comprendre principalement le fonctionnement du mécanisme de verrouillage des enregistrements. Celui-ci est géré par 4D afin d'éviter que deux utilisateurs modifient le même enregistrement au même moment.

Généralement, les développeurs utilisent la commande " CHARGER ENREGISTREMENT " dans une boucle comme le montre le code ci-dessous :

```
CHARGER  ENREGISTREMENT (nom Table)

Tant  que (Enregistrement  verrouille(nom Table))
     CHARGER  ENREGISTREMENT (nom Table)

Fin  tant  que
```

Cette méthode vous permet de savoir que l'enregistrement n'est pas verrouillé et qu'il a été chargé.

Mise en garde :
Il faut toujours avoir un cache assez grand. Il peut arriver que, dans certains cas très exceptionnels, 4D n'arrive pas à purger son cache permettant de libérer assez de place pour accueillir un nouvel enregistrement. Dans ce cas, 4D peut s'interrompre inopinément ou ne pas signaler d'erreur. Ce dernier cas entraînant un comportement anormal de l'application.

Pour être à l'abri de ce genre de désagrément, il faut toujours s'assurer d'avoir un cache assez important. Sa taille est déterminée en fonction du poids des enregistrements qui seront chargés.
Nous verrons ceci plus en détail dans le deuxième volet de cette note.

2 - Divers

Le fait d'avoir un enregistrement courant sur une table de données ne veut pas dire que celui-ci est chargé en mémoire. C'est un mécanisme normal qui peut se produire dans le cas suivant : un enregistrement est chargé et vous avez exécuté la commande " LIBERER ENREGISTREMENT ".

Si vous créez un nouvel enregistrement, il est d'abord conçu dans la RAM, sur le poste client, dans le cadre client serveur, puis sur la machine où se trouve 4D pour les autres applications.

Dans ce cas si vous utilisez la fonction " Enregistrement chargé " de la version 6.5 avant de sauvegarder l'enregistrement, cette fonction renverra la valeur "Vrai" car l'intégralité de l'enregistrement se trouve chargée en mémoire.

Quand 4D lit une valeur dans un champ deux cas sont possibles :

- Le champ est indexé, 4D consulte l'index dans le cache associé au champ.

- Le champ n'est pas indexé, 4D charge l'enregistrement complet.

En conclusion, il faut comprendre que le fait de charger un enregistrement provoque l'écriture en mémoire de plusieurs images de celui-ci. Ce mécanisme engendre donc une occupation très importante de la mémoire dont la croissance est proportionnelle à la taille de l'enregistrement.

Ce taux d'occupation est aussi inversement proportionnel aux performances, c'est-à-dire que plus la mémoire est occupée, plus les performances sont faibles.

IV - Les triggers et l'événement moteur "sur chargement"

1 - Généralités.

L'événement moteur sera exécuté à chaque fois qu'un enregistrement de la table, associé au trigger, sera chargé.

Afin d'optimiser le fonctionnement de 4D, l'événement moteur, sur le chargement de l'enregistrement, ne déclenche JAMAIS l'appel du trigger lors de l'utilisation des fonctions pouvant, éventuellement (mais pas systématiquement), tirer parti d'un index.

En effet, si l'index est utilisé, les enregistrements ne sont pas chargés. Inversement, si l'index n'est pas utilisé (par exemple si le champ traité n'est pas indexé), les enregistrements sont chargés.Cette incertitude quant à l'appel du trigger ne permet pas de l'exploiter de manière fiable.

Cet évènement couvre toutes les situations où un enregistrement courant est chargé à partir du fichier de données, à l'exception des fonctions listées ci-dessous. Nous vous rappelons que cet événement moteur ne concerne pas la création d'enregistrement.

- Recherches : elles sont effectuées par l'utilisateur (éditeur standard) et les commandes **CHERCHER**, **CHERCHER DANS SELECTION**.

- Tris : ils sont effectués par l'utilisateur (éditeur standard) et la commande **TRIER**.

- Fonctions statistiques : **Somme**, **Moyenne**, **Min**, **Max**, **Ecart type**, **Variance**, **Somme des carres**.

- Commandes : **JOINTURE**, **SELECTION RETOUR**.

2 - Dans quel process s'exécute un trigger ?

Deux possibilités sont à considérer :

- Versions monopostes : le trigger s'exécute dans le process qui a provoqué l'appel du trigger. Il faut sous entendre que le trigger, non seulement, se déroule dans le process, mais aussi qu'il est situé dans le contexte du process.

- Versions multipostes : quand un client se connecte, deux process se créent, l'un sur le client, l'autre sur le serveur. Le trigger s'exécute sur le serveur dans la partie du process dédiée au client.

V - Comment 4D manipule les enregistrements quand la mémoire est restreinte ?

Si vous parcourez le manuel langage de 4D, vous constaterez qu'il existe beaucoup de commandes ou fonctions qui concernent le traitement des enregistrements.
Il faut toujours avoir à l'esprit que 4D doit avoir de la place en mémoire pour charger les enregistrements dans leur intégralité.

Le manque de place mémoire, pour charger des enregistrements, peut être imputé à différents facteurs comme des enregistrements trop volumineux et trop nombreux, mais aussi par un autre phénomène que l'on appelle " fuite mémoire ".

Ceci est provoqué par des manipulations d'objets (au sens large du terme) dans le code. Voici quelques exemples qui peuvent être à l'origine de la fuite :

- Oublier de libérer la place après avoir créé et utilisé des tableaux, des ensembles, des sélections temporaires, des variables Textes ou Blobs.

- Oublier de dépiler des enregistrements préalablement empilés

Cette liste n'est pas exhaustive et d'autres phénomènes peuvent engendrer des fuites mémoires.

Pour détecter les fuites mémoires, l'explorateur d'exécution récemment apparu avec la version 6.5 de 4D peut vous aider.Vous pouvez le faire apparaître à l'aide des touches CTRL SHIF F9 sur PC et POMME SHIFT F9 sur MAC.Grâce à l'onglet "évaluation", vous obtiendrez la visualisation de l'état de la mémoire, et des informations sur le taux d'occupation des objets 4D.

Note : voir utilitaire GREEN SOFWARE sous windows et ZONE RANGER sous MACOS

Si vous voulez gérer efficacement ce genre d'erreur, il faut installer une méthode de gestion des erreurs à l'aide de la commande APPELER SUR ERREUR.

Quand 4D rencontre ce type d'erreur, il active la méthode de gestion des erreurs, quand elle existe, ou il gère lui même l'erreur. Mais, dans tous les cas, il interrompt le traitement du process courant afin d'éviter d'endommager les données. Ce comportement est identique à celui d'actionner le bouton "stopper exécution" de la fenêtre du mode " trace ".

Si vous installez une méthode de gestion des erreurs, 4D vous laisse l'entière responsabilité de la gestion, c'est-à-dire que vous devez canaliser tous les messages de 4D ainsi que vos propres messages.

Il faut savoir que, dans cette méthode, 4D vous donne la possibilité de créer vos propres messages d'erreur. Pour de plus amples informations, concernant cette partie, veuillez vous référer à la documentation.

VI - La différence de comportement du mécanisme de chargement des enregistrements entre 4D monoposte et 4D serveur.

Le mécanisme de chargement des enregistrements est différent selon que nous soyons en mode monoposte ou en mode client serveur.

Versions monopostes.

Quand vous utilisez des commandes qui vont charger des enregistrements, 4D examine s'il a la place nécessaire disponible en fonction de la place mémoire qui lui a été allouée. Ce mécanisme d'évaluation peut être faussé car 4D peut avoir une action de chargement dans différents process, ce qui peut rendre difficile de savoir exactement la place restant disponible.

Si la place est disponible, pour le chargement d'un enregistrement, 4D le charge. La fonction "Enregistrement chargé" retournera "Vrai", la fonction " Enregistrement trouvé" retournera le numéro de l'enregistrement.

L'influence des triggers :

Si la table associée à l'enregistrement possède un trigger sur " chargement enregistrement ", 4D l'exécutera à condition qu'il ait suffisamment de mémoire. Dans le cas contraire, 4D provoquera une erreur -108 (mémoire insuffisante).

Ce qu'il faut retenir :

- Si 4D ne possède pas assez de mémoire pour charger un enregistrement, il ne tentera pas de le charger et donc ne provoquera pas d'erreur.

- Si 4D, par le biais du mécanisme d'évaluation de la place disponible, estime qu'il peut charger l'enregistrement, vous pouvez obtenir les messages " mémoire saturée " ou " mémoire insuffisante " qui seront provoqués par le contexte de l'enregistrement, par exemple par un trigger.

- Si vous faites varier la quantité de la mémoire alloué à 4D en l'augmentant, il est possible que vous ne rencontriez plus de problèmes de chargement de vos enregistrements, mais un phénomène de swapping (transfert des données de la RAM vers le disque) peut s'installer (lié à la mémoire virtuelle), il fera tomber les performances de 4D. En résumé, il faut trouver un juste milieu entre le nombre d'enregistrements présents en mémoire et la taille à allouée à 4D.

Versions multipostes.

Le principe de chargement des enregistrements reste le même, mais il est réparti sur deux machines : la partie serveur et la partie cliente.

a) Pour le serveur

Elle correspond exactement à la partie monoposte que nous avons vu ci-dessus. Il faut simplement préciser que si une erreur apparaît (exemple erreur -108), le message est affiché sur le serveur. Dans ce cas, tous les clients actifs sont gelés (il est impossible de travailler).

Si un nouveau client tente de se connecter à cet instant, il sera gelé sur la fenêtre d'ouverture de la base. Ce phénomène restera en l'état tant qu'il n'y aura pas eu d'intervention sur le serveur pour répondre au message d'erreur provoqué par le chargement d'enregistrement. Après cet incident, il est fortement conseillé d'arrêter ou de relancer 4D.

Pour éviter d'avoir des messages d'erreurs sur la partie serveur, il est possible d'installer une méthode de gestion des erreurs à l'aide de la commande " APPELER SUR ERREUR ", dans la méthode base "sur démarrage serveur". Une méthode de gestion des erreurs s'applique à un process.

Cas particulier:

Les triggers et les méthodes bases partagent la même méthode de gestion des erreurs située sur le serveur. C'est précisément ce qui se passe lorsque vous lancez une méthode de gestion des erreurs dans la méthode base sur "démarrage serveur".

Si le client ne possède pas assez de mémoire, l'erreur -108 peut être générée. Elle sera généralement suivie par une erreur ?10002.

Ces erreurs seront gérées par 4D ou par votre propre procédure d'erreur.

L’erreur -10033 peut apparaître lors d’une demande du client vers le serveur. Ceci est la conséquence d’une désynchronisation entre le client et le serveur.

VII - Explication sur la quantité de mémoire nécessaire pour le chargement et la manipulation d'enregistrements de grande taille.

Quand vous manipulez des enregistrements importants en taille, en client serveur, il faut essayer de connaître le nombre de clients simultanés qui vont accéder à ces enregistrements car, suivant les cas, l'occupation de la mémoire pourra être très différente.

La version monoposte pose moins de soucis sauf si vous créez de multiples process. Dans cette hypothèse, cela est comparable au mode client serveur.

Pour avoir une idée de l'occupation mémoire, il faut compter le nombre de copies, que 4D génère, pour chaque enregistrement courant utilisé. Ce nombre ne comprend pas les copies dites partielles qui permettent d'utiliser la fonction " Ancien ".

Rappel :

Nous parlerons donc de copie complète pour un enregistrement et de copie partielle pour la copie de référence qui est nécessaire à la fonction " Ancien ".

Pour ce calcul, il faut aussi tenir compte du phénomène de découpage mémoire.
En effet, lors de la recopie de l'enregistrement dans la mémoire principale (user heap),4D travaille avec des blocs de mémoire insécables. Cela peut provoquer des conflits car, même s'il y a assez de mémoire disponible, cela ne veut pas dire que cette mémoire peut accueillir des blocs de mémoire insécables.

Le système en gère une partie, mais il ne peut pas déplacer des blocs de mémoire verrouillés. Ceci peut donc poser des difficultés lors du chargement de champs comme, par exemple des BLOBS, qui peuvent être de taille importante.

Présentation du nombre de copies présentes en mémoire et indication sur leur emplacement.

1 - Durant l’opération de chargement d’enregistrement.

Versions multipostes.

a) Côté serveur.

Dans le cache se trouve :

- Une copie primaire complète, compactée et unique.

- Une copie décompactée pour le process demandeur, dont les données fixes seront présentes dans le cache et dont les données variables (qui sont les champs Blob Image) seront stockées en mémoire principale.Cette copie permettra l'exécution des triggers.

- Une copie temporaire compressée qui est transmise au client puis effacée après l'opération de transfert.

Remarque :

Il y aura autant de copies décompactées que de process en exécution qui auront demandé le chargement de l'enregistrement.

b) Côté client.

Deux copies complètes seront présentes dans la mémoire principale (user heap).

- Une copie temporaire qui est compressée et réceptionnée par le client, puis effacée après l'opération de transfert

- Une copie qui forme l'enregistrement courant.

Versions monoposte ou procédure stockée.

Dans le cache se trouve :

- Une copie primaire complète, compactée et unique.

- Une copie décompactée pour le process demandeur dont les données fixes seront présentes dans le cache et dont les données variables (qui sont les champs Blob, Texte, Image) seront stockées en mémoire principale.Cette copie permettra l'exécution des triggers.

Remarque : il y aura autant de copies décompactées que de process en exécution qui auront demandé le chargement de l'enregistrement.

2 - Durant l'opération de sauvegarde d'un enregistrement existant.

Versions multipostes.

a) Côté serveur :

Dans le cache sont déjà présentes:

- Une copie primaire complète compactée et unique.

- Une copie décompactée pour le process demandeur

Durant la sauvegarde, une copie compactée, temporaire, reçue du client, pour le process demandeur, qui :

- S'il existe un trigger, sera décompactée et remplacera la copie décompactée, ensuite le trigger s'exécutera. Cette copie sera à nouveau compactée et remplacera la copie primaire.

- S'il n'existe pas de trigger, remplacera la copie primaire.

Remarque : il y aura autant de copies décompactées que de process en exécution qui auront demandé la sauvegarde de l'enregistrement.

b) Côté client :

Deux copies complètes seront présentes dans la mémoire principale (user heap).

- Une qui formera l'enregistrement courant,

- L'autre temporaire qui sera compactée et envoyée au serveur, puis effacée après l'opération de transfert

Versions monopostes ou procédure stockée.

Dans le cache sont déjà présentes :

- Une copie primaire complète, compactée et unique.

- Une copie décompactée pour le process demandeur.

Durant la sauvegarde, une copie temporaire, compactée pour le process demandeur qui :

- S'il existe un trigger, sera décompactée et remplacera la copie décompactée, ensuite le trigger s'exécutera. Cette copie sera à nouveau compactée et remplacera la copie primaire.

- S'il n'existe pas de trigger, remplacera la copie primaire.

Remarque : il y aura autant de copies décompactées que de process en exécution qui auront demandé la sauvegarde de l'enregistrement.

3 - La sauvegarde d'un nouvel enregistrement.

Versions multipostes.

a) Côté serveur :

Dans le cache :

Durant la sauvegarde, une copie compactée, temporaire, reçue du client, pour le process demandeur, qui :

- S'il existe un trigger, sera décompacté et remplacera la copie décompactée, puis le trigger s'exécutera. Cette copie sera à nouveau compactée et deviendra la copie primaire.

- S'il n'existe pas de trigger, deviendra la copie primaire.

b) Côté client :

Deux copies complètes seront présentes dans la mémoire principale (user heap).

- Une qui formera l'enregistrement courant,

- L'autre temporaire qui sera compactée et envoyée au serveur, puis effacée après l'opération de transfert.

Versions monopostes ou procédure stockée.

Dans le cache :

Durant la sauvegarde une copie temporaire, compactée, pour le process demandeur qui :

- S'il existe un trigger, sera décompactée et remplacera la copie décompactée, ensuite le trigger s'exécutera. Cette copie sera à nouveau compactée et remplacera la copie primaire.

- S'il n'existe pas de trigger, remplacera la copie primaire.

Remarque : il y aura autant de copies décompactées que de process en exécution qui auront demandé la sauvegarde de l'enregistrement.

4 - Détruire les enregistrements

Le fait de supprimer un enregistrement provoque son chargement s'il n'est pas en mémoire.
Aucune copie, dans ce cas, n'est présente. Après la suppression, toutes les copies, si elles existent, sont détruites tant sur le serveur que chez le client.