Optimization and new cache management

By 4D S.A.

TN 06-34

Introduction

データベースのパフォーマンスをほんとうに最適化する上で、キャッシュ管理に熟達することは実に欠かせない項目です。キャッシュの内部メカニズムおよびストラクチャデザインに及ぼす影響を理解することにより、最適のキャッシュ設定ができるようになります。このテクニカルノートは、バージョン 2004 から導入されたキャッシュの管理方法を説明し、以前のバージョンとの違いを際立たせる目的で執筆されました。

Reader prerequisites

読者は、インデックス、レコードのロードおよびアンロード、セレクションといった概念に通じていることが求められます。文中、トランザクションやアロケーションテーブルについても言及していますが、これらに関する高度な理解は特に必要ありません。付加的な内容は、随時テクニカルノートの中で説明されます。

Terminology

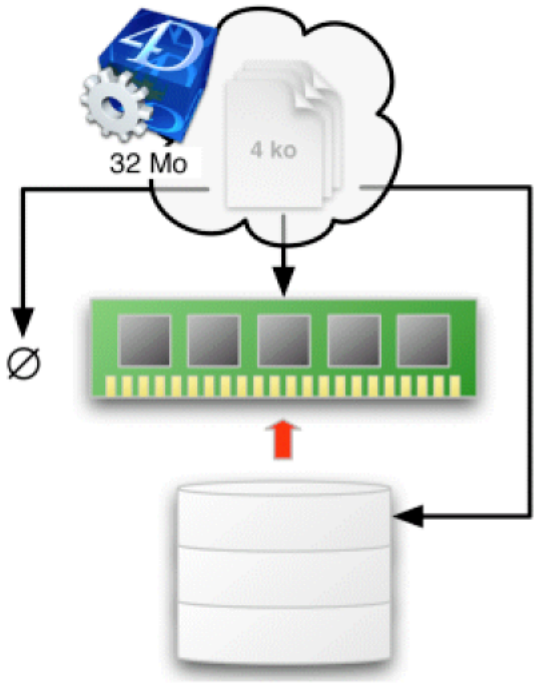
このテクニカルノートでは、「4D」はすべての 4D アプリケーションを指して用いられています。

Main memory

メインメモリは、データの計算、転送、処理を実行する過程で使用されます。いかなるデータもメインメモリを経ないで処理されることはありません。Windows 2000/XP および Mac OS X では、OS のメモリマネージャによってメインメモリが割り当てられます。割り当てられるのは可能な限り物理メモリです。物理メモリが足りなくなると、OS はもっとも使用度の低かったメモリブロックをディスクに保存し、他の操作のためにメモリ空間を確保するようになります。

この種のメモリは、仮想メモリと呼ばれます。仮想メモリで使用されるメモリアドレスは、物理アドレスではなく、RAM またはディスクに存在するブロック、あるいは未割り当てのブロックに対するリファレンスです。メモリマネージャは、仮想メモリブロックをページと呼ばれるサブブロックに分割しており、特定のページを使用するときにはじめて RAM またはメモリのどちらかをページに割り当てます。

いずれの OS においても、ハードディスクの使用はユーザの目から隠されており、たいていアプリケーションが必要とするときには RAM のページが割り当てられるようになっています。



メモリマネージャはメモリブロックをページで管理している。ページの実体は、RAM、ディスク、または未割り当ての領域。アプリケーションがブロックを使用するときには、RAM が確保できるように OS が工面するため、実行中のプロセスに行き渡るだけの RAM がある限り、アプリケーションが使用するページは RAM のページである。

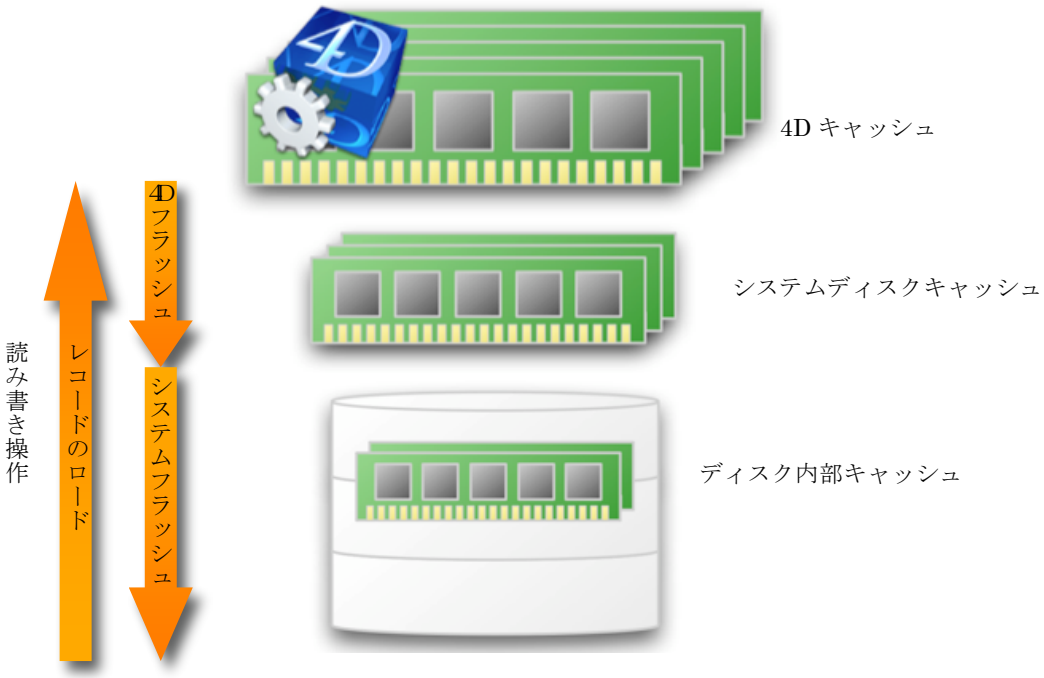
1.5GB の RAM が利用できるマシンで、いろいろなアプリケーションが合計 2GB メモリを要求したとしても、すぐにページスワッピングが発生するわけではありません。スワッピングが発生するのは、割り当てられたメモリの合計値が 1.5GB に到達した時点からです。関係するページが実際に使用されていないのであれば(つまりアイドルプロセス)、スワッピングが発生してもパフォーマンスが低下するわけではありません。マシンの実行プロセス数が多くなるにつれ、スワッピングによるパフォーマンスの低下が目立つようになります。

Disk access

OS はメモリアクセスだけでなく、ハードディスクアクセスも管理しており、どちらのプラットフォームでもその方法はよく似ています。

現行のハードディスクは、RAM と比較した場合、データアクセスに 50 から 2000 倍の時間を要します(データ量が小さいほどその差は顕著)。この問題を緩和するため、メーカーはディスクに内部キャッシュメモリ(4 から 16MB)を付加していますが、そのサイズも充分とはいえないため、OS 自身がハードディスクのためのキャッシュをメインメモリに持っています。

これらのキャッシュには、最後に要求されたブロック、またはもっとも頻繁に要求されたブロックが収納されています。キャッシュのディスクに対する書き込みは、一定時間ごと、またはアプリケーションから書き込みの要求があったときに実行されます。



ディスクの内部キャッシュはサイズが充分でないため、OS は付加的なキャッシュとしてメインメモリの一部を利用している。4D のキャッシュはディスクキャッシュよりもサイズが大きく、また収納されたオブジェクトの種類に応じて異なる優先度を適用するため、キャッシュとしてより特化された利用ができるようになっている。

ある種の操作、たとえばファイルコピーなどでは、データをキャッシュに保存しておく意味がありません。そのような場合、不必要なキャッシュ使用を避けるために、アプリケーションはノンキャッシュアクセスを実行するようになっています。

それ以外の操作、たとえばループでデータが繰り返し利用される場合などは、キャッシュを使用することによってパフォーマンスの劇的な向上が期待できます。

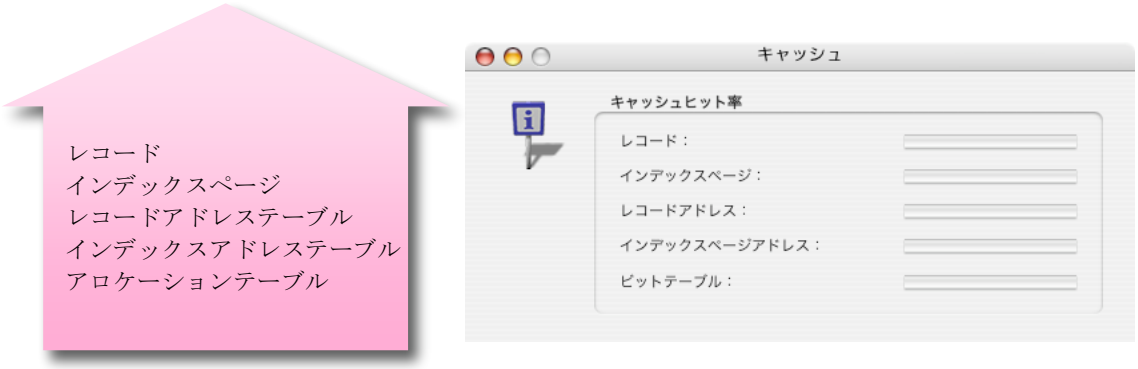
4D Cache

過去のテクニカルノートなどでも説明されているように、4D はメインメモリにアドレスアロケーションテーブル(インデックスおよびレコード)を構築し、さらに実際のインデックスとレコードもメモリに置くようになっています。このキャッシュ、つまり 4D のキャッシュは、データベースの規模がシステムキャッシュよりも大きくなるとパフォーマンスに反映されます。一時的なデータ、たとえばカレントセレクションやトランザクションなどをメモリに置くことができるためです。4D キャッシュは、デベロッパがある程度は制御できるという点でも優れています。

<table>
<tr><td colspan="2">レコード</td><td>セレクション</td><td>トランザクション</td></tr>
<tr><td rowspan="3">インデックスページ</td><td>レコードアドレステーブル</td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>インデックスアドレステーブル</td></tr>
<tr><td>アロケーションテーブル</td><td colspan="2">レコードマーカ</td></tr>
<tr><td colspan="2">データファイルオブジェクト</td><td colspan="2">一時的なオブジェクト</td></tr>
</table>

4D キャッシュに書き込まれるオブジェクト。データファイルに由来するものは、変更がなければ破棄される。その他のものは、一時ファイルに保存することができる。

キャッシュのサイズが不足すると、オブジェクトは特定の優先順位にしたがってアンロードされてゆきます。アンロードされたオブジェクトが占有していた領域には、同等あるいはより優先度の高いオブジェクトがロードされます。通常、アンロードされるのはレコードだけです。極端な場合、インデックスページやアドレステーブルがアンロードされるかもしれません。



優先順位(下のものほど優先順位が高く、上のものほど優先順位が低い)。このリストは 4D Server で表示されるキャッシュ情報と一致する。

キャッシュの内容をディスクに書き込む操作は、一般に「キャッシュのフラッシュ」と呼ばれています。フラッシュは、環境設定で定められた周期で定期的に実行され、その他ランゲージの FLUSH BUFFER コマンド、あるいはユーザモードのファイルメニューにある「ディスクに保存」を選択することによっても実行されます。

定期的にフラッシュを実行する主な目的は、停電や強制終了といった不意のハプニングからデータを守ることにあります。データに少しでも変更があれば、稼働中はフラッシュが定期的に実行されるため、ある程度の安心感が得られます。フラッシュの間隔が短すぎると、ピーク使用時のパフォーマンスが低下します。5 分から 30 分がフラッシュの妥当な実行間隔です。

キャッシュの設定に関係のあるパラメータは、実行間隔だけではありません。とはいえ、システムキャッシュの効果が操作の種類によって異なるように、4D キャッシュもデータベースの操作内容にかなりの影響を受けます。それではどのようにキャッシュの使用を最適化できるのかを考えてみましょう。

Optimal cache size

最適のキャッシュサイズは、データベースが保存するオブジェクトの種類にかかっています。標準の優先順位では、アロケーションテーブルが基本的に常時ロードされています。アロケーションテーブルはほとんどすべての操作に関係しているためです。アロケーションテーブルがアンロードされるようなことになれば、全面的にパフォーマンスが低下します。

アドレステーブルとインデックスページも基本的にキャッシュに置かれているべきです。インデックスフィールドに対するクエリが最適化されるのは、検索にディスクアクセスが関係しているときだけです。パフォーマンスは、実際にはクエリで返されるレコード数にも依存しています。クエリで返されるレコード数が多ければ、それだけ全体の所要時間におけるディスクアクセススピードの占める意味は少なくなります。

レコードのアドレステーブルもキャッシュに置かれているべきです。実行するクエリがインデックスによるクエリばかりであれば、アドレステーブルの重要性は低くなりますが、シーケンシャルクエリやセット・セレクションの操作をするときにアドレステーブルが必要です。最高のパフォーマンスは、アドレステーブルがキャッシュにロードされているときに得られます。

問題はレコードデータです。繰り返し使用されるデータほどキャッシュ効果が高いことはすでに述べました。言い換えるならば、繰り返し使用するレコードはキャッシュし、一度しか使用しないレコードはキャッシュしないのが理想的です。100 万件のレコードをロードすれば、キャッシュマネージャはレコードの数だけ処理時間を必要としますが、次に同じレコードを使用

することがあれば、そのときにキャッシュした当初の時間的ロスを取り返すことができます。

レコードキャッシュは、メモリのページ化と同じように処理されるため、サイズ不足でパフォーマンスが低下するかもしれません。。仮にすべてのレコードが均等に使用され、それらすべてを収納するだけのキャッシュサイズがない場合、頻繁にロード・アンロードが発生します。そうなってしまうと、むしろキャッシュなどないほうが良いかもしれません。

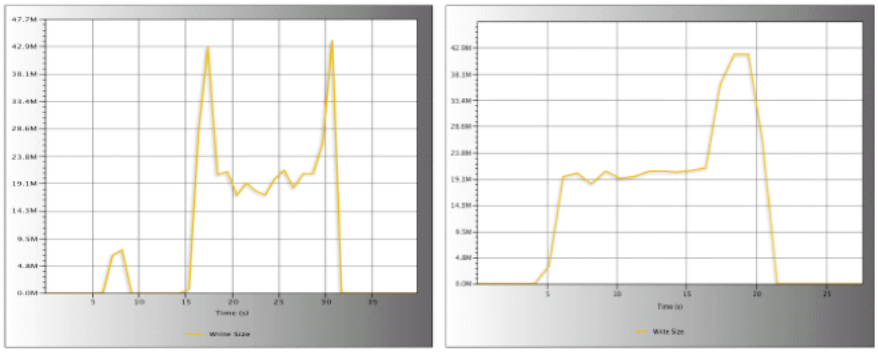
最後にトランザクションとセレクションについて考慮する必要があります。トランザクションによるフラッシュを回避するためには、トランザクションが必要とする最大サイズが計算できなければなりません。セレクションは、レコードごとに 4 バイトを使用するため、よほど小さなキャッシュ、あるいはよほど大きなデータベースでなければ、問題にならないでしょう。

Optimizations in 4D 2004

バージョン 2004 では、最大 8GB の RAM が搭載でき、最高 50MB/秒の速度でハードディスクデータ転送が可能な現行のハードウェアが活かせるように、メモリとキャッシュの使用が Mac OS X と Windows XP の両方で最適化されました。

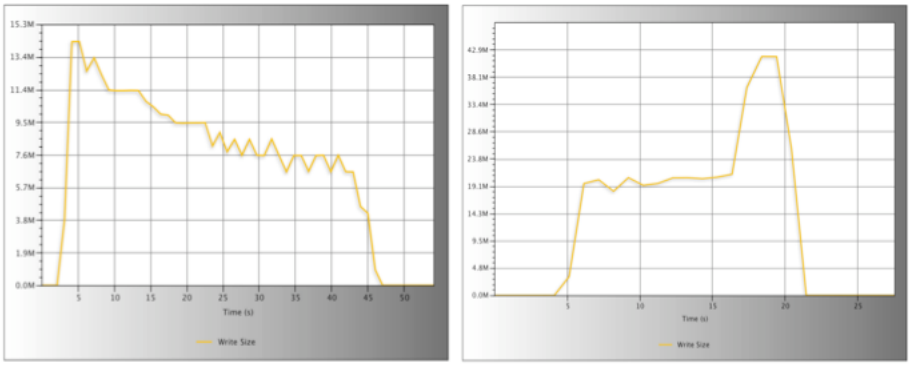
第一の改良点は、キャッシュサイズの設定方法です。以前のバージョンでは、大きなキャッシュサイズにおける断片化が全体的なパフォーマンスに望ましくない影響を及ぼすことがありました。バージョン 2004 では、1GB のキャッシュサイズであっても、断片化が問題になることはありません。

第二の改良点は、キャッシュの動作そのものの最適化です。4D 2003 では、SET DATABASE PARAMETER のセレクタ 26 を使用することにより、キャッシュの書き込みを最適化することができましたが、4D 2004 でこの操作をする必要はまずありません。4D 2004 では、フラッシュの際に発生するディスクアクセスの回数が最適化されており、前述のセレクタ 26 はかえってマイナス効果になる恐れがあります。データベースパラメータの変更が効果的なのは、データファイルが極端に断片化されている場合など、特殊なケースに限られます。



バージョン 2004 で 400MB のキャッシュをセレクタ 26 有効(左)、無効(右)それぞれの設定で実行した結果。垂直軸は瞬間転送速度(MB/秒)、水平軸は時間(秒)を表わしている。最適化オプションにより瞬間転送速度は向上するが、アロケーションテーブルのフラッシュによる再計算処理が直後に続いていることが分かる。ピーク書き込みの後に谷間があるのは、断片化の証拠。

キャッシュの書き込み自体も改良され、理論上の最高値に近い転送速度が得られるようになりました。256MB のデータを保存する場合、4D 2003 では 60MB/秒のハードディスクで 10 ないし 15MB/秒の平均転送速度が限度でしたが(Power Mac G5)、バージョン 2004 では 40 ないし 45MB/秒の速度に達するようになりました。データベースに何ら変更を加えることなく、どんなデータベースであってもアップグレードするだけでこの最適化の効果が実感できます。



バージョン 2003 のフラッシュとバージョン 2004 のフラッシュを比較した結果。

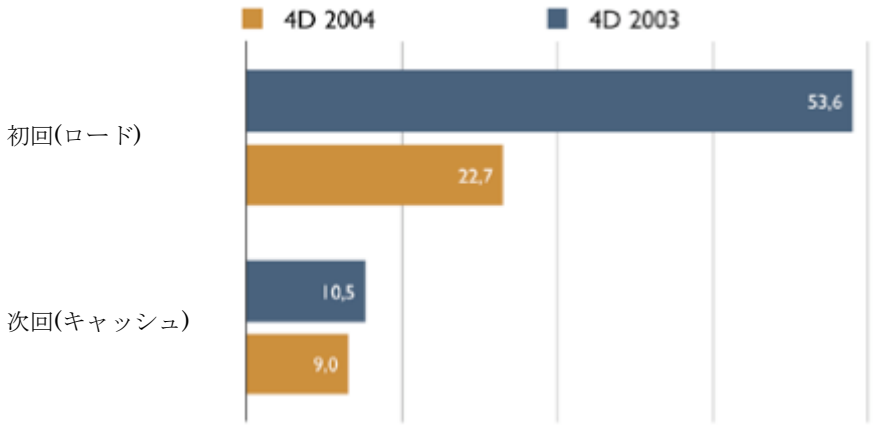
レコードをロードする方法も最適化されました。レコードが 4D キャッシュにはなく、ロードしなければならない場合、システムキャッシュが活用されるようになった結果、ハイエンドマシンでは最高 50%のパフォーマンスアップが得られるようになりました。やはり、データベースを 4D 2004 にアップグレードするだけで、すぐにこの最適化の恩恵にあずかることができます。

APPLY TO SELECTION を 100 万件のレコードで実行

データファイルサイズ：130MB

キャッシュサイズ：160MB

搭載メモリ：512MB



所要時間単位：秒

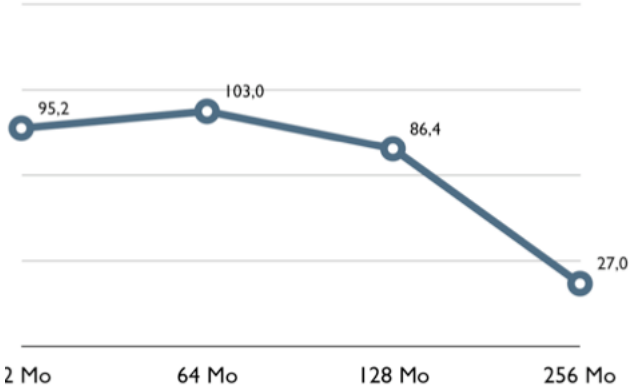
バージョン 2003 と 2004 でデータがロードされる場合と 4D キャッシュが使用される場合の APPLY TO SELECTION 所要時間。前者 58%、後者 14%の速度改善を計測。

APPLY TO SELECTION を 300 万件のレコードで実行

データファイルサイズ：390MB

搭載メモリ：512MB

所要時間単位：秒



キャッシュサイズ

異なるキャッシュサイズで計測したバージョン 2004 の APPLY TO SELECTION 所要時間。キャッシュサイズを増量した結果、ロード時間の増加によりわずかに全体の所要時間が長くなるものの、やがて速度向上による時間短縮がそれを上回ることが分かる。このテストでは、256MB 以降、所要時間は変わらなくなった。

Setting the cache

レコードが基本的に一度しか使用されないのであれば、インデックスとアドレステーブルを収納するだけのキャッシュを確保することが主要な関心事となります。その点を調べるには、ランタイムエクスプローラを開き、インデックスページヒット率が 100%に近い数値、レコードヒット率が 20%以下の数値を示していることを確認してください。かなり大きなセレクションやトランザクションを使用しているのであれば、それぞれが高いヒット率であることも確認しておいてください。

ランタイムエクスプローラのキャッシュ統計情報。

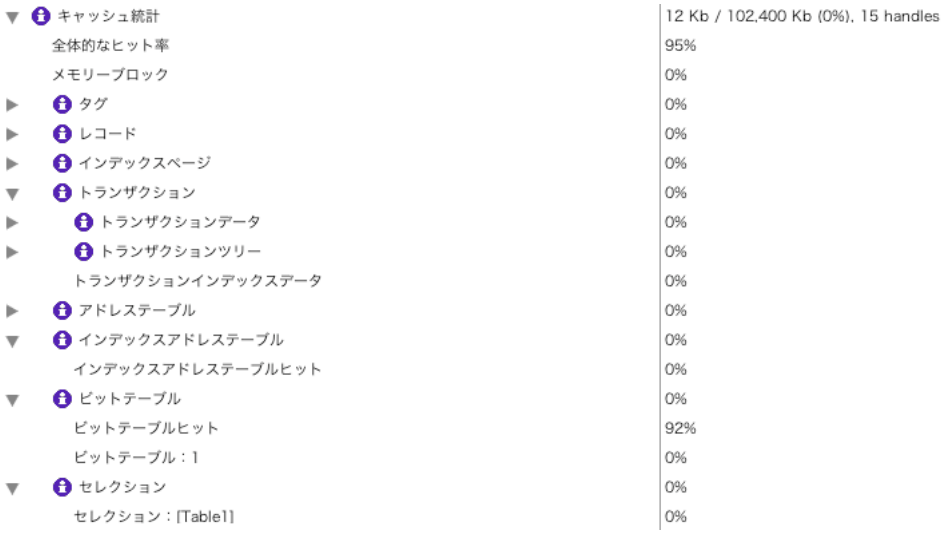
各テーブルのレコードが繰り返し使用されるのであれば、全レコードをキャッシュに収納してしまうのが得策です。(シーケンシャルロード/アンロードのために 10%程度の余裕を残すようにしてください。)この場合、高いレコードヒット率(100%近く)を維持するだけの充分なキャッシュサイズを確保する必要があります。

加えてトランザクションデータも 100%近い数値になるようにしてください。

セレクションの統計は、動作状況モニタリングを有効にした後、ランタイムエクスプローラのキャッシュ統計アイテムを収縮/展開して表示を更新すれば表示されるようになります。

モニタリングを有効にすると、アドレステーブルの利用状況を監視できるようになります。統計値が実際的な精度に達していないのであれば(大きなキャッシュにわずかなレコード数がロードされている場合など)、コンテキストメニューから「フィールド&テーブル番号を表示」オプションを選択することにより、キャッシュに収納されているオブジェクトの数が占有しているメモリサイズと共に表示されるようになります。

キャッシュサイズが充分であれば、100%に近いキャッシュヒット率が得られるはずです。データベースがすべてのレコードをロードするためには、ある程度の時間が必要です。



ランタイムエクスプローラの「動作状況モニタリングを有効」オプション。

同じリストで「フィールド&テーブル番号を表示」オプションを有効にした結果。
オブジェクトとハンドルの数で統計が表示される。

あるレコードが他のレコードよりも頻繁に使用されるようであれば、前者は(10%のゆとりを残して)キャッシュし、滅多に使用しないレコードでキャッシュが一杯になるような操作の後には(FLUSH BUFFER コマンドで)キャッシュをフラッシュするようにしてください。そうすれば、ランタイムエクスプローラには頻繁に使用されるデータが保存されたテーブルに関する情報だけが表示されるようになります。大量(10 万件以上)のレコードを削除した後や大きなトランザクションの後にはレコードマーカが占有していた領域を解放するためにフラッシュが必要です。>

データベースはどれも固有の性格を有しています。最適なサイズを見極めるためのテスト実施をためらうべきではありません。インデックスはすべてキャッシュに収納されるていることを確認し、もしも収まりきらないようであれば、ほんとうに必要なところでだけインデックスを使用するようにしてください。

Conclusion

このテクニカルノートでは、利用可能なメモリサイズおよびデータベースの性格に基づき、最適な 4D キャッシュサイズを見極める方法が説明されました。実行中のデータベースでキャッシュの使用状況を調べる方法、バージョン 2004 で施された改良点についても取り上げました。

最後になりましたが、データベースは OS のもとで実行されている以上、正常に動作できるだけの RAM が OS のために残されているべき点を忘れないようにしてください。RAM 不足により OS のパフォーマンスが低下するようなことがあれば、データベースもその影響を免れることはできません。こうした制限や注意事項に配慮さえしていれば、必要なだけのメモリサイズを 4D キャッシュに割り当てることができます。