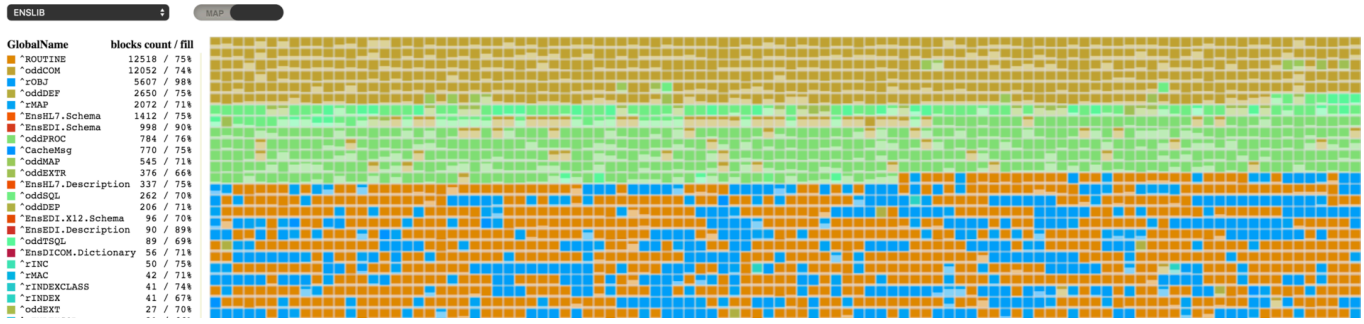
記事
Toshihiko Minamoto · 2021年2月8日 10m read

Caché データベースブロックの内部構造、パート 3。

これで 3 記事目になります (パート 1 と パート 2 をご覧ください) が、引き続き Caché データベースの内部構造をご紹介いたします。 今回は、興味深い内容をいくつかご紹介し、私の Caché Blocks Explorer プロジェクトを使って作業の生産性をアップさせる方法について説明します。



この画像に表示されているものに見覚えがあるという方はたくさんおられると思います (クリック可能)。 グローバルの断片化した状態を視覚化する必要があったとき、様々なディスクデフラグツールが私の頭をよぎりました。 なんとか、そういったツールと同等の効果を発揮する製品を開発できたと願っています。

このユーティリティには複数のブロックで構成されたマップが表示されています。 四角はそれぞれブロックを表し、その色はレジェンドセクションに表示されている特定のグローバルに対応しています。 また、ブロックそのものは格納されているデータの量を示しているため、マップにさっと目を通すだけで、データベース全体の使用量を素早く推定できます。 グローバルレベルのブロックとマップレベルのブロックは未だ実装されていませんので、空のブロック同様に白のブロックとして表示されます。

データベースを選択すれば、すぐにブロックのマップが読み込みを開始します。 情報はそれぞれ順番にではなく、ブロックツリーに並ぶブロックの順番に従って読み込まれるため、そのプロセスは以下の画像のようになる可能性があります。

データベースは、前回の記事で使用したものを引き続き使用しましょう。
グローバルは必要ありませんので、すべて削除しています。 また、SAMPLES データベースの Sample クラスパッケージを基に新しいデータを生成しました。 そのために、HABR と名付けた私のネームスペースへのパッケージマッピングを設定しています。

以下のようにデータ生成コマンドを実行しました。

```
do ##class(Sample.Utils).Generate(20000)
```

マップには以下の結果が表示されました。

ブロックが埋まり始める場所はファイルの先頭でないことにお気づきでしょうか。 ブロック 16 からトップレベルのポインタブロックが、そしてブロック 50 からデータブロックが始まります。 デフォルト値は 16 と 50 ですが、必要に応じて変更できます。 ポインタブロックの先頭は SYS.Database クラスの*NewGlobalPointerBlock* プロパティで定義されます。これにより、新しいグローバルのデフォルト値が設定されます。 既存のグローバルの場合は、*PointerBlock* プロパティを使って%Library.GlobalEdit クラスの中で変更できます。 一連のデータブロックを開始するブロックは、SYS.Database クラスの*NewGlobalGrowthBlock* プロパティに指定されます。 個別のグローバルの場合は、%Library.GlobalEdit クラスの *GrowthBlock* プロパティを使っても同じことができます。 こういったプロパティの変更は、トップポインタブロックやデータブロックの現在の位置には一切影響しないため、まだデータを格納していないグローバルに対してだけ行うことが理に適っていると言えます。

ここで、989 個のブロックを持つ **^Sample.PersonD** グローバルは 83% 埋まっており、それに次いで 573 個のブロックを持つ **^Sample.PersonI** グローバルは 70% 埋まっていることが分かります。 どのグローバルを選択しても、それに割り当てられたブロックを確認できます。 ^Sample.PersonI グローバルを選択すると、ほぼ空になっているブロックがいくつかあるのが分かります。 また、これら 2 つのグローバルに属するブロックが混合していることも分かります。 実はこれには理由があります。 新しいオブジェクトが作成されると、これら 2 つのグローバルは、片方はデータで、もう片方は Sample.Person テーブルのインデックスで埋まってしまうのです。

テストデータがいくつか手に入ったところで、Caché が提供するデータベース管理機能を活用して結果を確認することができます。 まずは、データを少し取り除いて、いかにもデータを追加したり、削除したりするアクティビティが実行されているかのように見せます。 ランダムなデータをいくつか削除するコードを実行します。

```
set id=""
set first=$order(^Sample.PersonD(""),1)
set last=$order(^Sample.PersonD(""),-1)
for id=first:$random(5)+1:last {
    do ##class(Sample.Person).%DeleteId(id)
}
```

このコードを実行すると、以下の結果が表示されます。 空のブロックがいくつかある一方で、64～67% 埋まっているブロックもあります。

このデータベースは、%SYS ネームスペースから ^DATABASE ツールを使って操作することができます。 では、その機能をいくつか使ってみましょう。

まずは、ブロックがほとんど埋まっていないので、データベース内のすべてのグローバルを圧縮したらどうなるか試してみましょう。

ご覧のとおり、圧縮したことで、占有率を必要な 90% の値に限りなく近づけることができました。 この結果、空であったブロックは他のブロックから移動されてきたデータでいっぱいになりました。 データベースのグローバルは、^DATABASE ツール (アイテム 7) を使うか、以下のコマンドにデータベースへのパスを 1 つ目のパラメーターとして渡して実行すれば圧縮できます。

```
do ##class(SYS.Database).CompactDatabase("c:\intersystems\ensemble\mgr\habr\")
```

また、すべての空のブロックをデータベースの最後に移動することもできます。 これは、例えば、膨大なデータを削除したからデータベースを圧縮したいというときに必要となるかもしれません。
このデモとして、私たちのテスト用データベースからデータを削除する作業をもう一度実行してみましょう。

```
    set gn=$name(^Sample.PersonD)
    set first=$order(@gn@(""),1)
    set last=$order(@gn@(""),-1)
    for i=1:1:10 {
        set id=$random(last)+first
        write !,id
        set count=0
        for {
            set id=$order(@gn@(id))
            quit:id=""
            do ##class(Sample.Person).%DeleteId(id)
            quit:$increment(count)>1000
        }
    }
```

以下はデータを削除した結果です。

空のブロックがいくつかあるのが分かります。 Caché
では、こういった空のブロックをデータベースファイルの末尾に移動して圧縮することができます。
空のブロックを移動するには、システムネームスペースにある SYS.Database クラスの FileCompact メソッドを使うか、^DATABASE ツール (アイテム 13) を利用しましょう。
このメソッドには、データベースへのパス、ファイルの末尾の理想的な空きスケース (デフォルトは 0)、戻り値パラメーター (最終的な空きスペース) の 3 つのパラメーターを渡すことができます。

```
do ##class(SYS.Database).FileCompact("c:\intersystems\ensemble\mgr\habr\",999)
```

結果、空のブロックがなくなりました。 先頭のブロックは、設定通りに
(上位のトップレベルポインタとデータブロックを開始する位置として) 置かれているだけなので、考慮しません。

デフラグ (最適化)

これでグローバルを最適化する作業に取りかかれます。
このプロセスを実行すると、各グローバルのブロックの順番が並び替えられます。 デフラグを実行するには、データベースファイルの最後に、ある程度の空きスペースが必要になる場合があり、それが必要な状況ではスペースが追加される可能性があります。 このプロセスは、^DATABASE ツールのアイテム 14
から始めるか、以下のコマンドを実行して始めることができます。

```
d ##class(SYS.Database).Defragment("c:\intersystems\ensemble\mgr\habr\")
```

空きスペースを増やす

グローバルがきちんと並べられているのはいいのですが、
どうやら、デフラグによってデータベースファイルのスペースが余分に使われてしまったようです。

このスペースを開放するには、^DATABASE ツールのアイテム 12 を使用するか、以下のコマンドを実行します。

```
d ##class(SYS.Database).ReturnUnusedSpace("c:\intersystems\ensemble\mgr\habr\")
```

データベースの占有スペースは大幅に減りましたが、データベースファイルには空きスケースが 1 MB
しか残っていません。 ブロックを移動し、空きスペースを作ることによってデータベース内のグローバルをデフラグし、空きスペースを管理するという可能性が発表されたのはつい最近の話です。
それ
までは、
データベースをデフラグしてデータベースファイルのサイズを縮小する必要があったときは、^GBLOCKCOPY
ツールを使う必要がありました。
このツールは、ソースデータベースの各ブロックを新しく作成されたデータベースに 1
つずつコピーするもので、ユーザーはコピーするグローバルを選択できました。
このツールは現在も使用可能です。

#システム管理 #データベース #Caché

ソースURL:
https://jp.community.intersystems.com/post/cach%C3%A9-%E3%83%87%E3%83%BC%E3%82%BF%E3%83%99%E3%83%BC%E3%82%B9%E3%83%96%E3%83%AD%E3%83%83%E3%82%AF%E3%81%AE%E5%86%85%E9%83%A8%E6%A7%8B%E9%80%A0%E3%80%81%E3%83%91%E3%83%BC%E3%83%88-3%E3%80%82