破損したデータファイルを復旧する

By Thang Nguyen, Technical Support Engineer, 4D Inc.
Technical Note 05-09

（原題: Recovering a Damaged Data File）

概要

4種類のデータ復元方法を紹介する。4D Toolsを使用する方法、データファイルを再インデックスする方法、レコードを再び保存する方法、新規データファイルにデータを書き出す方法である。

4D社が第一に推奨するのはバックアップファイルとログファイルの統合による復元である。その方法で復元できない場合にのみ、以下の方法を試すべきである。その場合も、バックアップをとってから始めるべきであるのはいうまでもない。また、診断のみで復元にはいたらない方法も含まれている。

4D Toolsを使用する方法

4Dに付属の4D Toolsは、データのチェック、修復機能を備えたユーティリティプログラムである。ダメージの度合いにもよるが、4D Toolsを使用して修復できる場合もある。使用するには、問題のストラクチャファイルを4D Toolsで開き、問題のデータファイルを指定する。メンテナンスタブをクリックし、データ検査のオプションを選択する。

すべてを検査オプションは、レコードとインデックスの両方をすべて検査する。時間はかかるが、通常はこちらのオプションを選択する。

レコード検査オプションは、テーブルを指定することができるので、巨大なデータベースの特定のテーブルをチェックしたい場合には便利である。

インデックス検査オプションは、特定のインデックス、あるいはすべてのインデックスを検査したい場合に使用する。

検査の結果、問題が見つかった場合は、修復タブをクリックして修復のオプションを選択する。

すべてを復旧すると、時間はかかるが、すべてのレコードとインデックスが解析され、必要に応じて修復される。

レコード修復オプションは、テーブルを指定することができるので、巨大なデータベースの特定のテーブルを修復したい場合には便利である。

インデックス修復オプションは、特定のインデックス、あるいはすべてのインデックスを修復したい場合に使用する。

タグによる復旧

タグによる修復を試みる前に、再インデックス、およびレコードの読み込み/保存による方法を試すのが望ましい。再インデックスによる方法がうまくいかない場合には、この方法を試すことができる。

このオプションは旧データファイルを修復するのではなく、新たなデータファイルを作成する。したがってディスクの空スペースに注意する必要がある。

通常、長さが不揃いである各レコードの位置を特定するために4Dはアドレステーブルを使用している。このアドレステーブルが損傷をうけた場合に、各レコードのヘッダに書き込まれているタグを使用して、アドレステーブルを復元を試みるのがこのオプションである。（タグに関する詳細は、Tech Note 01-29および01-30を参照。）

タグによる修復を選択すると、ダイアログが表示されるので、新しいデータファイルの場所と名前を指定する。セグメントが複数あるなら、このときそのことも設定する。しない場合は、修復の途中でセグメントの作成を促すダイアログが表示される。

再インデックスによる修復

ストラクチャのコピーを作成し、新規データファイルを作成する。すべてのリレートを外し、各フィールドのインデックス属性を外す。再起動後、コピーのストラクチャで元のデータファイルを開いて閉じるとデータファイルのインデックスがなくなる。最後に元のストラクチャでインデックスのないデータファイルを開くと再インデックスができる。

データの読み込み/保存による修復

特定の動作、たとえばクエリなどを実行するとDBが落ちるのに、4D Toolsでは問題が発見できない場合には、データの読み込み/保存による方法が有効な場合がある。データをひとつずつ読み込んで、保存し直せば、問題のあるデータを特定することができる。以下のコードは、問題の解決を保証するものではないが、各レコードを読み込んで保存し、成功したものを"Records Logs"ファイルに書き出すものである。

```
`----Massage method---
C_BLOB($Blob;$BlobBuffer)
C_BOOLEAN($Boolean;$BooleanBuffer)
C_DATE($date;$DateBuffer)
C_INTEGER($Integer;$IntegerBuffer)
C_LONGINT($Longint;$LongintBuffer)
C_PICTURE($Picture;$PictureBuffer)
C_REAL($Real;$RealBuffer)
C_STRING(20;$String;$AlphaBuffer)
C_TEXT($Text;$TextBuffer)
C_TIME($Time;$TimeBuffer)
C_LONGINT($j)
C_TEXT($tableName)
C_TIME($vhDoc)

$vhDoc:=Create  document("Record  Logs")
If (OK=1)
For ($j;1;Count  tables)
ALL RECORDS(Table($j)->)
For ($k;1;Count  fields(Table($j)))
Case  of
\ (Type(Field($j;$k)->)=ls  Alpha Field )
ALL RECORDS(Table($j)->)
For ($i;1;Records  in selection(Table($j)->))
$AlphaBuffer:=Field($j;$k)-> Field($j;$k)->:=$string
SAVE  RECORD(Table($j)->) Field($j;$k)->:=$AlphaBuffer
```

```
SAVE  RECORD(Table($j)->)
$tableName:=Table  name($j)
SEND  PACKET($vhDoc;"OK    "+"Table: "+$tableName+"    "+"Field:
"+String($k)+"   "+"Record: "+String($i)+Char(13))  `  Write one word in the
document
NEXT  RECORD(Table($j)->)
End for
\ (Type(Field($j;$k)->)=ls Text )
ALL RECORDS(Table($j)->)
For ($i;1;Records  in selection(Table($j)->))
$TextBuffer:=Field($j;$k)->
Field($j;$k)->:=$Text
SAVE  RECORD(Table($j)->)
Field($j;$k)->:=$TextBuffer
SAVE  RECORD(Table($j)->)
$tableName:=Table  name($j)
SEND  PACKET($vhDoc;"OK    "+"Table: "+$tableName+"    "+"Field:
"+String($k)+"   "+"Record: "+String($i)+Char(13))  `  Write one word in the
document
NEXT  RECORD(Table($j)->)
End for
\ (Type(Field($j;$k)->)=ls Integer )
ALL RECORDS(Table($j)->)
For ($i;1;Records  in selection(Table($j)->))
$IntegerBuffer:=Field($j;$k)->
Field($j;$k)->:=$Integer
SAVE  RECORD(Table($j)->)
Field($j;$k)->:=$IntegerBuffer
SAVE  RECORD(Table($j)->)
$tableName:=Table  name($j)
SEND  PACKET($vhDoc;"OK    "+"Table: "+$tableName+"    "+"Field:
"+String($k)+"   "+"Record: "+String($i)+Char(13))  `  Write one word in the
document
NEXT  RECORD(Table($j)->)
End for
```

```
\ (Type(Field($j;$k)->)=ls LongInt )
ALL RECORDS(Table($j)->)
For ($i;1;Records  in selection(Table($j)->))
$LongintBuffer:=Field($j;$k)->
Field($j;$k)->:=$Longint
SAVE  RECORD(Table($j)->)
Field($j;$k)->:=$LongintBuffer
SAVE  RECORD(Table($j)->)
$tableName:=Table  name($j)
SEND  PACKET($vhDoc;"OK    "+"Table: "+$tableName+"    "+"Field:
"+String($k)+"    "+"Record: "+String($i)+Char(13))  `  Write one word in the
document
NEXT  RECORD(Table($j)->)
End for
\ (Type(Field($j;$k)->)=ls  Boolean )
ALL RECORDS(Table($j)->)
For ($i;1;Records  in selection(Table($j)->))
$BooleanBuffer:=Field($j;$k)->
Field($j;$k)->:=$Boolean
SAVE  RECORD(Table($j)->)
Field($j;$k)->:=$BooleanBuffer
SAVE  RECORD(Table($j)->)
$tableName:=Table  name($j)
SEND  PACKET($vhDoc;"OK    "+"Table: "+$tableName+"    "+"Field:
"+String($k)+"    "+"Record: "+String($i)+Char(13))  `  Write one word in the
document
NEXT  RECORD(Table($j)->)
End for
\ (Type(Field($j;$k)->)=ls Date )
ALL RECORDS(Table($j)->) For ($i;1;Records  in selection(Table($j)->))
$DateBuffer:=Field($j;$k)->
Field($j;$k)->:=$date
SAVE  RECORD(Table($j)->)
Field($j;$k)->:=$DateBuffer
SAVE  RECORD(Table($j)->)
```

```
$tableName:=Table  name($j)
SEND  PACKET($vhDoc;"OK    "+"Table: "+$tableName+"   "+"Field:
"+String($k)+"   "+"Record: "+String($i)+Char(13))  `  Write one word in the
document
NEXT  RECORD(Table($j)->)
End for
\ (Type(Field($j;$k)->)=ls Time )
ALL RECORDS(Table($j)->)
For ($i;1;Records  in selection(Table($j)->))
$TimeBuffer:=Field($j;$k)->
Field($j;$k)->:=$Time
SAVE  RECORD(Table($j)->)
Field($j;$k)->:=$TimeBuffer
SAVE  RECORD(Table($j)->)
$tableName:=Table  name($j)
SEND  PACKET($vhDoc;"OK    "+"Table: "+$tableName+"   "+"Field:
"+String($k)+"   "+"Record: "+String($i)+Char(13))  `  Write one word in the
document
NEXT  RECORD(Table($j)->)
End for
\ (Type(Field($j;$k)->)=ls Picture )
ALL RECORDS(Table($j)->)
For ($i;1;Records  in selection(Table($j)->))
$PictureBuffer:=Field($j;$k)->
Field($j;$k)->:=$Picture
SAVE  RECORD(Table($j)->)
Field($j;$k)->:=$PictureBuffer
SAVE  RECORD(Table($j)->)
$tableName:=Table  name($j)
SEND  PACKET($vhDoc;"OK    "+"Table: "+$tableName+"   "+"Field:
"+String($k)+"   "+"Record: "+String($i)+Char(13))  `  Write one word in the
document
NEXT  RECORD(Table($j)->)
End for
\ (Type(Field($j;$k)->)=ls Real )
```

```
ALL RECORDS(Table($j)->)
For ($i;1;Count  fields(Table($j)))
$RealBuffer:=Field($j;$k)->
Field($j;$k)->:=$Real
SAVE  RECORD(Table($j)->)
Field($j;$k)->:=$RealBuffer
SAVE  RECORD(Table($j)->)
$tableName:=Table  name($j)
SEND  PACKET($vhDoc;"OK    "+"Table: "+$tableName+"    "+"Field:
"+String($k)+"  "+"Record: "+String($i)+Char(13))  `  Write one word in the
document
NEXT  RECORD(Table($j)->)
End for
\ (Type(Field($j;$k)->)=ls  BLOB )
ALL RECORDS(Table($j)->)
For ($i;1;Records  in selection(Table($j)->))
$BlobBuffer:=Field($j;$k)->
Field($j;$k)->:=$Blob
SAVE  RECORD(Table($j)->)
Field($j;$k)->:=$BlobBuffer
SAVE  RECORD(Table($j)->)
$tableName:=Table  name($j)
SEND  PACKET($vhDoc;"OK    "+"Table: "+$tableName+"    "+"Field:
"+String($k)+"    "+"Record: "+String($i)+Char(13))  `  Write one word in the
document
NEXT  RECORD(Table($j)->)
End for
\ (Type(Field($j;$k)->)=ls Subtable )   `do  nothing
End case
End for
End for
SEND  PACKET($vhDoc;"All Records are OK"+Char(13))  `  Write one word
in the document
CLOSE  DOCUMENT($vhDoc)  `  Close the document
Else ALERT("Cannot create a Records Log")
```

```
End if
```

データ書き出しによる方法

以上の方法による復旧がどれも成功しない場合の最後の手段が、データのエクスポート/インポートによる方法である。データファイルのサイズによっては非常に時間がかかる方法であるので覚悟が必要な上、いくつかの注意点がある。コマンドSEND RECORDとRECEIVE RECORDを使用すると、ほとんど場合、問題のあるレコードも一緒にエクスポート/インポートされるので無意味になってしまう。もっとも確実な方法は、ユーザモードで標準のインポート/エクスポートダイアログを使用する方法である。形式には、タブ区切りのテキスト形式か、XMLを指定する。（書き出し/読み込みに関する詳細については4th Dimensionユーザリファレンスを参照。）

この方法を使用すれば、各フィールドにつき、データはテキストに変換されるので、変換に成功すればその時点でエラーを起こすデータを除去することができる。