Paradox File Corruption

by Brian Long

It seems that corruption in Paradox tables and index files is still rife, if the following *Delphi Clinic* question is any measure:

I read your summary of precautions regarding Paradox table and index corruption in multi-user applications on page 52 of Issue 17, implemented all the measures, but I still get corrupted tables and index files every now and again. Are there any more things you can think of?

Well, indeed there are. Since writing that entry I have been collecting other useful corruption avoiding tips and techniques that I will list here. First of all though, Table 1 shows a summary of what was highlighted in Issue 17.

Another useful piece of advice is to ensure you set the `Session` object's `PrivateDir` property to some suitable private directory. Perhaps a subdirectory of your application's directory. You can get your application's directory with `ExtractFilePath(Application.ExeName)`. Avoid leaving `PrivateDir` blank as it will default to the current directory, which will more than likely be the application directory. Letting the BDE use your application directory as its temporary working directory can lead to *Lock file has grown too big* exceptions. Also, avoid pointing `PrivateDir` at a root directory, make sure it points at a subdirectory.

One final point on the private directory subject is that the private directory should have access to a reasonable amount of disk space. A query linked across tables to extract data may take up at least *three times* the size of largest table for its temporary __*.* files.

If your application is terminated unexpectedly, your BDE private directory might have its temporary files left in place, instead of being deleted as they would normally be by the BDE. If you think this might be a problem, you could write a section of code that executes at the beginning of your application, before any tables are opened, that deletes files from the private directory conforming to the specification __qb*.*.

Since the `Local Share` setting is so crucial, one idea is to only let the program run if `Local Share` has a value of `True`. Your main form's `OnCreate` handler could call a routine like that shown in Listing 1. In Delphi 2 or later you can also read `Local Share` in a more VCL-esque fashion by making use of

```
Session.GetConfigParams(
  '\System\Init',
  SomeTStringsObject)
```

and then checking

```
SomeTStringsObject.Values[
  'LOCAL SHARE']
```

Another piece of general advice for applications that may be editing, deleting and adding many records is to periodically pack the Paradox tables to save them spreading across your hard disk. Issue 9, p63 has code for doing this.

New information needs to be added to this list to ensure that networked machines do not cause problems by 'clever' buffering of any sort. One of the prime reasons for the problem is that two applications accessing the same table have conflicting views of what is really in the table because each machine is caching changes to a certain degree.

Unfortunately there are many levels of this caching. Consequently there are many system settings that you need to give appropriate values to in order to avoid its potentially harmful impact to the application. This problem is more general than just Paradox applications: many vendors have the same issues and so these settings can help many applications to work in a more resilient manner. However, given the context of this article, I will only be referring to BDE-driven Paradox data applications.

For all Windows 95 machines running BDE applications accessing Paradox or dBASE data or containing that data you should take the following steps. Firstly, make sure you have at least version

➤ *Table 1*

If there is a possibility of a form being closed with a table still open, in the OnClose event handler, ensure either Post or Cancel is called. If you open your tables programmatically, then you could also close them in a form's OnClose or OnCloseQuery event handler.
In the BDE Administrator or BDE Configuration program, set Local Share to True in the System settings.
Don't store tables on drive letters manufactured with the DOS SUBST command.
Don't store NET files in a root directory, some BDE revisions apparently did not handle this too well.
Call dbiSaveChanges in the TTable's AfterPost event handler.
If your version of the BDE supports it, call dbiUseIdleTime in your Application's OnIdle event handler. This is easier than the above point, and apparently provides the automated equivalent of it. The API was removed in BDE 4, with the suggestion of using dbiSaveChanges instead.
In Delphi 1, write a wm_EndSession message handler for your main form and call Halt in it. This ensures the BDE closes down and hopefully flushes all its buffers.

```
procedure CheckLocalShare;
var ASYSConfig: SYSConfig;
begin
 {$ifdef Win32}
  { Ensure BDE is initialised }
  Session.Open;
 {$endif}
  if (DbiGetSysConfig(ASYSConfig) = DbiErr_None) and
    not ASYSConfig.bLocalShare then begin
    ShowMessage('BDE''s LOCAL SHARE flag must be TRUE for this ' +
      'program to run. Ask your System Administrator to do this for ' +
      'you.'#13#13'This program will not continue until this change ' +
      'has been made and all BDE applications have been restarted');
   {$ifdef Win32}
    Application.ShowMainForm := False;
   {$endif}
    Application.Terminate;
  end
end;
```

➤ *Listing 1*

4.00.1116 of VREDIR.VXD (156,773 bytes, 11th Sep 97, 11:16) and version 4.00.1112 of VNETSUP.VXD (17,595 bytes, 30th May 97, 11:12). These can be installed with the patch program vrdupd.exe located at http://support.microsoft.com/Download/support/mslfiles/ and fix a problem in Windows 95 where an application shares data with a Windows NT server, as described in Microsoft's Knowledge Base articles Q148367 and Q174371. Note that Microsoft's articles get the date stamp of VREDIR.VXD wrong and suggest it is 2 Jun 97.

Next, `VREDIR` must also be set up correctly: ensure that the binary registry value

```
HKey_Local_Machine\System\
  CurrentControlSet\Services\VxD\
  VREDIR\DiscardCacheOnOpen
```

is set to `01`. The machine will need rebooting for this setting to take effect.

Plus, Windows 95 caching should be disabled. To do this, launch the `System Properties` dialog by holding down the `Alt` key and double clicking on `My Computer` (or by holding down the `Windows` key and pressing `Pause`). Click the `Performance` tab, press the `File System...` button and click on the `Troubleshooting` tab. Check the options: `Disable write-behind caching for all drives`. This corresponds to setting this `DWord` value to `0`:

```
HKey_Local_Machine\System\
  CurrentControlSet\Control\
  FileSystem\DriveWriteBehind
```

In addition to the above setting there is some suggestion that the `Disable synchronous buffer commits` and `Disable new file sharing and locking semantics` options should also be checked, although the help for these options does not really uphold this. This equates to setting `AsyncFileCommit` to `1` and `SoftCompatMode` to `0`.

Windows 95 machines running Novell networking software should set the registry entries shown in Table 2 (setting these when Novell is not installed does no harm).

On Windows NT machines that run the BDE, or contain Paradox or dBASE files, the settings in Table 3 should be applied to disable opportunistic locking. The machine will need rebooting for these settings to take effect. Opportunistic locking is explained in Microsoft's Knowledge Base article Q129202, and this quote is taken from the Windows NT Resource Kit: 'This setting specifies whether the server allows clients to use oplocks on files. Oplocks are a significant performance enhancement, but have the potential to cause lost cached data on some networks, particularly wide area networks.' Some of the problems introduced by oplocks are described in articles Q134637, Q124916, and Q126026.

If you are running Windows for Workgroups 3.1x, you should disable disk write caching. In SYSTEM.INI in the `[386Enh]` section and add an entry to disable 32-bit write-caching on all drives that have shared data on them,

Type	Key	Value
DWord	HKey_Local_Machine\System\CurrentControlSet\Services\VxD\NWREDIR\ReadCaching	0
String	HKey_Local_Machine\Network\Novell\System Config\Netware Dos Requester\Cache Writes	No
String	HKey_Local_Machine\Network\Novell\System Config\Netware Dos Requester\Opportunistic Locking	No

➤ *Above: Table 2*

➤ *Below: Table 3*

Type	Key	Value
DWord	HKey_Local_Machine\System\CurrentControlSet\Services\LanmanServer\Parameters\EnableOpLocks	0
DWord	HKey_Local_Machine\System\CurrentControlSet\Services\LanmanServer\Parameters\EnableOpLockForceClose *This is probably redundant because of the setting above, but you can set it to be on the safe side*	1
DWord	HKey_Local_Machine\System\CurrentControlSet\Services\LanmanServer\Parameters\CachedOpenLimit	0
DWord	HKey_Local_Machine\System\CurrentControlSet\Services\LanmanWorkStation\Parameters\UseOpportunisticLocking	0
DWord	HKey_Local_Machine\System\CurrentControlSet\Services\LanmanWorkStation\Parameters\UtilizeNtCaching	0
For Windows NT Workstation's running Novell NetWare :		
DWord	HKey_Local_Machine\System\CurrentControlSet\Services\NWCWorkstation\Parameters\DisablePopup	0

eg `ForceLazyOff=CDE`. Microsoft KnowledgeBase article Q107645 describes this option and article Q108109 emphasises that `[386Enh]` is the correct section, not `[vcache]` as mentioned in the Resource Kit documentation. If AUTOEXEC.BAT loads the SMARTDRV.EXE program, make sure the `/X` parameter is used to disable write-behind caching.

The sample project DBStuff.Dpr on this month's disk contains some code which will check all the appropriate entries depending on whether you are running your application on Windows 95, NT or 3.1. In fact, so potentially important are the registry settings that if they are not found to have the correct values, the program sets them and suggests that the user reboots. DBStuff.Dpr compiles in all versions of Delphi from 1 to 4.

The reboot dialog comes from `RestartDialog`, an undocumented Shell32 API described in the *Restarting Windows* entry in Issue 40's *Delphi Clinic*. Since I am not passing any text to the API I don't need to worry about the Unicode issue highlighted in that write-up.

Listing 2 shows the form's `OnCreate` handler, the main registry checking routine and the Windows 95 checking code. You can see that several helper routines are not listed, including the code that checks the version information of the network redirector files. Refer to the project on the disk for all the missing code. One helper routine listed in full is `CheckRegistryEntry`. This verifies both numeric and string registry values (passed as a `Variant` parameter), setting them if they are either wrong or missing. If a change was made, the `Reboot Required` flag is set to `True`.

It is important to remember that these operating system settings must be set on *all* machines either running BDE applications or containing BDE data. Similarly the BDE must have `Local Share` set to `True` on *all* installations. If a machine gets upgraded, or has the operating system re-installed, or has a new version of the BDE set up, some of these settings may need to be fixed again. The same applies if a new machine is added to the network.

Of course if this approach is taken, you also need to remember to set the settings appropriately on any file server machines used to store your data files, but which do not have any Delphi applications running on them.

One important point on this whole subject is that it helps enormously if you educate and train your users. It is not unheard

➤ *Listing 2*

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  CheckOKForParadoxAppToRun
end;
...
procedure CheckRegistryIsAcceptable;
begin
 {$ifdef Win32}
  case Win32Platform of
    VER_PLATFORM_WIN32_WINDOWS : CheckWin95Registry;
    VER_PLATFORM_WIN32_NT      : CheckWinNTRegistry;
  end;
  if RebootRequired then
    //Use standard Win32 reboot dialog
    RestartDialog(0, nil, ew_RestartWindows)
 {$else}
  CheckWin31Registry;
  if RebootRequired then begin
    ShowMessage('Some system settings have been changed '+
      '- Windows needs to restart');
    ExitWindows(ew_RestartWindows, 0)
  end
 {$endif}
end;
...
procedure CheckRegistryEntry(Reg: TRegistry;
  const Path, Value: String; const Default, Desired:
  Variant; Size: Byte);
var
  TmpInt: Cardinal;
  TmpStr: String;
begin
  with Reg do
    if OpenKey(Path, True) then
      try
        case VarType(Desired) of
          varInteger:
            begin
              TmpInt := Default;
              if ValueExists(Value) then
                ReadBinaryData(Value, TmpInt, Size);
              if TmpInt = Default then begin
                TmpInt := Desired;
                WriteBinaryData(Value, TmpInt, Size);
                RebootRequired := True
              end
            end;
          varString:
            begin
              TmpStr := Default;
              if ValueExists(Value) then
                TmpStr := ReadString(Value);
              if TmpStr = Default then begin
                TmpStr := Desired;
                WriteString(Value, TmpStr);
                RebootRequired := True
              end
            end
        end
      finally
        CloseKey
      end
end;
const
  Control  = 'System\CurrentControlSet\Control\';
  Services = 'System\CurrentControlSet\Services\';
procedure CheckWin95Registry;
var
  Reg: TRegistry;
const
  DOSRequester =
    'Network\Novell\System Config\Netware Dos Requester';
begin
  Reg := TRegistry.Create;
  try
    Reg.RootKey := HKey_Local_Machine;
    //Fix VREDIR.VxD settings
    CheckRegistryEntry(Reg, Services + 'VxD\VREDIR',
      'DiscardCacheOnOpen', 0, 1, SizeOf(Byte));
    //Fix NWREDIR.VxD settings
    CheckRegistryEntry(Reg, Services + 'VxD\NWREDIR',
      'ReadCaching', 1, 0, SizeOf(Byte));
    //Fix Novell settings
    CheckRegistryEntry(Reg, DOSRequester, 'Cache Writes',
      'Yes', 'No', 0);
    CheckRegistryEntry(Reg, DOSRequester, 'Opportunistic
      Locking', 'Yes', 'No', 0);
    //Fix FileSystem troubleshooting settings
    CheckRegistryEntry(Reg, Control + 'FileSystem',
      'DriveWriteBehind',
      $FFFFFFFF, 0, SizeOf(Longint));
   {$define AllOptionsThatPeopleSuggest}
   {$ifdef AllOptionsThatPeopleSuggest}
    CheckRegistryEntry(Reg, Control + 'FileSystem',
      'SoftCompatMode', 1, 0, SizeOf(Longint));
    CheckRegistryEntry(Reg, Control + 'FileSystem',
     'AsyncFileCommit', 0, 1, SizeOf(Byte));
   {$endif}
  finally
    Reg.Free
  end
end;
...
procedure CheckOKForParadoxAppToRun;
begin
 {$ifdef Win32}
  //Only Win95 redirector files need checking
  if Win32Platform = VER_PLATFORM_WIN32_WINDOWS then
    CheckRedirector;
 {$endif}
  CheckRegistryIsAcceptable;
  CheckLocalShare;
end;
```

of for users to pull the power cable from their PCs to plug into a kettle, or to terminate applications with the `Ctrl+Alt+Del` three-fingered salute.

See the boxout on the right for other references which discuss some of these issues, sometimes in the context of totally non-Inprise, non-Paradox systems.

One final point to make is that if your Paradox tables do get corrupted, you can make use of TUtility.DLL to try and repair them. A BDE 5 compatible version of this DLL can be found at:

```
www.inprise.com/devsupport/bde/utilities.html
```

You can also find a sample Delphi application there that can be used against arbitrary Paradox tables to try and fix them.

Thanks go to Will Watts, Nick Moon, David Rose, Nick Spurrier, Andy Race and Jack Birrell for some of the information used in this article.

Brian Long is an independent consultant and trainer. You can reach him at brian@blong.com

References

Inprise Technical Information sheet TI3342 which offers more views on the Paradox corruption issue; note the file name is case sensitive:
www.inprise.com/devsupport/bde/ti_list/TI3342.html

Q148367 discusses the Win95 redirector bug; to read any of the other MS KnowledgeBase articles you can build up a URL using the same pattern as this one:
http://support.microsoft.com/support/kb/articles/q148/3/67.asp

Discussion of the Windows 95 redirector problem and NT opportunistic locking:
www.turbopower.com/newsletters/0798/default.asp

Discussion of Novell settings to avoid data loss:
www.daccess.com/Support/Integrating_DF_with_NetWare.htm

cc:Mail technical paper 144293 discussing various Windows registry settings that need to be set correctly to avoid data loss:
http://orionweb.lotus.com/sims/c8f6_546.htm

cc:Mail technical paper 139714 discussing preferred NetWare settings to avoid data loss; if you use NetWare you should read this as it describes many preferred NetWare configuration settings not discussed here:
http://orionweb.lotus.com/sims/8a76_546.htm

Article 2917002 describes a Novell read caching problem; article 2911461 describes potential data loss issues using Windows 95 against a NetWare 3.11 server:
http://support.novell.com/search/kb_index.htm