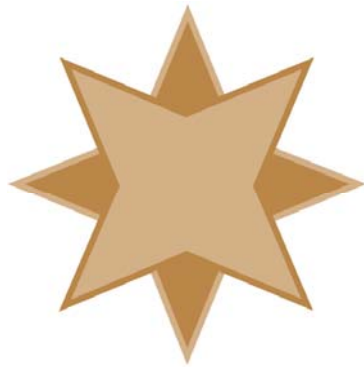


Using Continuous Operations Mode for Proper Backups

A White Paper From



GOLDSTAR
SOFTWARE

www.GoldstarSoftware.com

For more information, see our web site at
<http://www.goldstarsoftware.com>

Using Continuous Operations Mode for Proper Backups

Last Updated: July 2010

Ensuring the Validity of your Backup

There are many types of "valid" backups. The trick is to know what level of integrity your backup system is providing today and what type you really need to protect your company's data assets. As the level of integrity increases, so do the costs and limitations that are imposed on the backup. Like anything else in life, it important to know the tradeoffs involved when selecting a strategy. Note that the issues presented herein are problems with any database, not just Btrieve databases.

The first part of this article looks at the various types of integrity that can be attained. The second part discusses achieving database integrity within Btrieve databases.

Media Integrity: This refers to the problem of backing up files to a given device that has failed but does not report errors properly. The backup software says that everything is A-OK, but every tape is blank.

Remember: You do **not** do backups just to do backups. You do backups to do *restores*. Any backup scheme should have periodic (weekly or monthly) testing of the backup media, including the restoration of a random file and comparing the resulting data with the original.

Backup Integrity: Many backup packages can access files even though they may be open by the OS. If your backup software reports a "complete backup" with no errors, then you have a copy of the entire database on the tape.

Unfortunately, there is no guarantee that it will be a usable copy, as it is just a copy of the files as the backup software got to them. Remember that open Btrieve files can change rapidly, at many different points in the file. By the time the backup software gets through the first half of the file, the second half may have been changed hundreds of times. You may have the entire file on the tape, but it may not be a proper snapshot.

One devastating problem that can occur is that when the Page Allocation Table is backed up, it lists several free-space pages that actually contain valid data by the time the backup gets to them. Restoring this file will almost definitely result in lost data or a Status 2.

Data File Integrity: This indicates that **each data file** on the backup media is a complete representation of the file at a given instant in time. Some backup packages which claim to back up open files act as data caches which store the original data for a file in a memory cache and allow the data on disk to be changed real-time.

This solution works very well, but if there are too many changed pages to keep in memory, or if the server crashes during the backup, these caches typically fail and revert to a normal backup, giving essentially no protection at all.

Database Integrity: Database integrity is defined as having a complete snapshot of **all data files in the database** from a given instant in time. It can be achieved through the use of a more powerful data cache package capable of treating groups of files as a single unit. Of course, when many files are in a group and are being actively modified, there is a greater likelihood of the cache being overrun with changes, invalidating the entire backup. Database integrity for a Btrieve database can also be achieved through the use of Btrieve's continuous operations mode.

Application Data Integrity: This is the hardest to control. Many database applications do not utilize transactions. Instead, they lock individual records in multiple files and then update each of them in turn.

A good example of this is an inventory application that supports storing parts in multiple locations. It uses a location file that contains locations, quantities, and pointers to the inventory part file. For performance reasons, the total inventory quantity is also stored in the inventory file. Then, when new parts are added to inventory, the application increases the values for both the inventory part file and the inventory location file. What happens if the application updates the inventory part file, but before it can update the inventory location file, the database is "snapshotted" onto a backup either by continuous mode or a data cache? In this case, if these files must be restored, the total inventory quantity will no longer match the sum of the inventory location quantities.

This application-level integrity is the hardest to ensure, as it requires either the extensive use of transactions or that all users exit the application before the backup. Luckily, it is also the easiest to recover from, since there is no actual corruption in the files themselves, and applications which can be affected by this often have "fix" programs to adjust the quantities. Of course, this also means that it may be impossible to detect a problem of this nature until someone is comparing reports or doing auditing at year-end and notices the discrepancy. (Just hope it doesn't affect your paycheck stub!)

In the next section, we will discuss the options for using Continuous Operations Mode to maintain database integrity during your backup.

What is Continuous Operations Mode?

Every Pervasive database engine since Btrieve 6.0 supports the use of *Continuous Operations Mode* for files. Continuous Operations Mode (or ContOps), enables you to take a snapshot backup of your database files while they are in use by your application, essentially eliminating downtime related to backups.

Continuous Operations Mode works by freezing each data file in its current state and reopening it in a shared, read-only mode so that the backup software can easily access it. Then, the engine creates and opens a *delta file*, which uses the same base file name but an extension of "^^^", to which all file changes are written. When data is read from the file, the engine checks the delta file first to see if an updated page is available. If a new page is not found, the page is retrieved from the main data file instead.

When the files are removed from continuous mode, the original file is opened in read-write mode once again, and changed pages from the delta file are committed back to the primary file. Once this completes, the delta files are deleted and the file is marked as being open in "normal" mode once again.

Using Continuous Operations Mode to Attain Database Integrity

Having a solid backup is imperative to properly recover from a disaster. In the previous issue, we reviewed various levels of data integrity that can be obtained by a backup system. In this issue, we delve more deeply into the mechanics of continuous operations mode with the Btrieve and Scalable SQL database engines, so that you can maintain full database integrity as you prepare for those inevitable disasters to strike.

In order to utilize continuous operations mode for an entire database, four prerequisites must be met.

1. The database must reside completely on a single server. If a set of files is split across two or more servers, then only the files on a single server can be guaranteed to retain full **database integrity**, while the rest of the files will have **data file integrity** only.
2. Your database engine should be as fully patched as possible. Some versions (especially very old 6.10 engines) have been known to have problems with continuous operations mode.
3. Your database files must all be in the 6.x or higher format. Continuous Operations mode is not supported on 5.x or earlier files.
4. For NetWare servers, you must use a BUTIL.NLM file which is dated 12/11/1995 or later. Earlier versions had problems with a task ID that prevented BUTIL from being able to remove files from continuous mode in certain situations.

To ensure that the entire database gets placed into continuous mode at the same time, you need to create a text file that contains a list of all files in the database with complete pathnames (including volume names on NetWare and server-side drive letters on Windows Servers). Each file name should be on its own line, like this:

```
C:\PVSW\DEMODATA\PERSON.MKD  
C:\PVSW\DEMODATA\FACULTY.MKD  
...
```

Make sure that all pathnames in this file refer to valid database files on the server, or problems may result. You may wish to also include data dictionary (DDF) files in this list, especially if they are being actively changed. The command "DIR /B /S" may be helpful in building this file. This file will be referred to as DATAFILE.TXT in the subsequent discussion.

Once the file DATAFILE.TXT is prepared, you should test the continuous operations mode process. Make a full backup of your files with all users out of the database and verify that it worked. (This is a good idea before ANY major upgrade or testing of new

Information Provided By **Goldstar Software Inc.**

<http://www.goldstarsoftware.com>

features.) Use the following command from the server console to start continuous mode:

```
BUTIL -STARTBU @DATAFILE.TXT
```

On NetWare Servers, you may need to add the "LOAD" keyword before each line, and you will also need to provide the volume name for the location of DATAFILE.TXT. For Windows servers, specify the path to the DATAFILE.TXT on the local machine, such as C:\DATAFILE.TXT. Note the time it takes for the command prompt to come back. This tells you how long it will take to put the database into continuous operations mode, and should be used in your backup job to insert a pause before starting the backup.

Next, look in the directory, and see all of the files with the ^^^ extension. This indicates that the files are properly logging operations in continuous mode. If you do NOT see these files for every database file, you may need to modify the DATAFILE.TXT file list.

To remove the files from continuous mode, execute the following command from the server console:

```
BUTIL -ENDBU @DATAFILE.TXT
```

This command will roll the changed pages from each delta files into the corresponding primary files and then delete the delta. If the database was very active during this time, you may see the server run at a higher utilization while this process completes. You will also be able to watch as the delta files disappear from the directory as they are processed.

The process of rolling files into and out of continuous mode can be time consuming, especially for large databases with many active changes. It is important to schedule these functions for a quiet moment in the day. Outside of a minor "pause", the users should see no impact from a typical continuous operations mode process. (Note that if there is a substantial number of database updates, inserts, or deletes while the system is in ContOps mode, the server will likely get very sluggish, as the extra load from ContOps is busy handling the writes to the delta file. For this reason, we do NOT recommend doing any write-intensive processing during a backup.)

Automating Continuous Operations Mode

Once you understand the process itself, it is a simple matter to automate it from the server console. You simply need to execute the proper BUTIL command before and after the backup process. Several server-based backup products like *ARCserve* and *BackupEXEC* provide this functionality as a base part of the package and can even provide a built-in delay for the process to complete before it continues. The advantage to this solution is simplicity. The disadvantage is that the exact time for rollout may vary with the length of the backup itself, and users will not know when to limit their usage.

If your backup package does not support executing server commands, or if it is running on a different server or a workstation, then you will need to issue the commands either manually or from an command scheduler utility. Windows users can leverage the built-in

Information Provided By **Goldstar Software Inc.**

<http://www.goldstarsoftware.com>

AT or SCHEDULES functions, while NetWare and Linux users can use the CRON utility.

Testing Continuous Operations Mode

Once you have achieved a good backup, how can you fully test it? Assume a major disaster (tornado, earthquake, hurricane, fire, or flood) just wiped out your entire building. Now, get your server up and running with **only** your off-site backup tape. Remember to include the time needed to order new hardware, software, tape drive, etc. Companies who can be running again in less than a few days are very rare, indeed.

When the files have been recovered, you should first delete any delta files. (Remember that delta files are still rapidly changing, and should not really be included in the backup.) Then, check the timestamps on the files -- they should all be exactly the same -- the time of the START of the backup. If you find any files that are NOT the same timestamp, then this file may not be a Btrieve file, or if it is, then it was not properly included in the DATAFILE.TXT file, and you should remediate that before the next backup cycle.

Once you have the data recovered, you should try running your application to verify that all is well. You may also wish to use a tool like Goldstar Software's KeyCheck to run through every record in some of your more critical files to ensure that the files remain intact.

What To Do When Things Go Bump In the Night

Everything is great -- right until something goes terribly wrong in the middle of your backup. (Which, as Murphy would say, is the **ONLY** time that things will go wrong.) So, when the power goes out and the server crashes, what should you do?

Option 1: First, Kill All the Deltas

If you simply restart the server and delete the delta files, the system will think it was just restored from backup and recover gracefully. All data changes since the STARTBU will be lost, but all files are in a known, consistent state.

Option 2: Allow the Deltas to Recover

If you leave the files alone, then the roll-in process will start on each file which is opened. Note that the deltas may be partial or may have partial transactions in them, so choose wisely. What happens if you don't open up all files during the day? The next evening, your STARTBU will fail, because not all files could be put into continuous mode (because some still had deltas existing). You **MUST** do something which causes every file to open and resolve the deltas prior to the next backup. That something may be one of the following:

1. Open each file. A BUTIL -STAT will do the trick, or even Goldstar Software's LISTSTAT which accepts wildcards. This works very well on older 6.x engines, but not so well on the newer ones like Pervasive.SQL 2000 which only roll files in while the file is actually open.
2. Open each file & HOLD IT OPEN. Use the Function Executor (WBEXEC) or even WBEXEC32 to open each file & hold them open until the system finishes

Information Provided By **Goldstar Software Inc.**

<http://www.goldstarsoftware.com>

the roll-in. This may be required in some cases where #1 doesn't work. A new utility called HoldOpen to hold these files open is available [here](#).

3. Open each file, read the first record, then update the first record in place. Hold the file open until the roll-in completes. This is required in places where #1 & #2 don't work.

Hopefully, Pervasive will work out the issues with the roll-in process, but until then, we are stuck with what we have. In any event, if both the delta and the live file are getting updated timestamps, then the roll-in process is continuing. Let it finish as soon as possible -- leave people out of the applications if you must to allow the process to complete.

Once all deltas have been rolled in, THEN you can start your next backup cycle.

The Pervasive Backup Agent

For users of Pervasive.SQL V8 and newer server engines, and for the Pervasive PSQL v10 Workgroup Engine, there is another tool available that simplifies this process greatly. The Pervasive Backup Agent, with a list price of only \$249, can automate the entire Continuous Operations mode process for you. When the Backup Agent goes into backup mode, it talks to the database engine to determine the list of active database files, eliminating the need to create or maintain a file list. It can be easily integrated with the backup software in a single command (PVBACKUP -ON). Even better, if there is a server crash, it will automatically hold the database files open for you for up to 30 minutes when the system restarts, giving the database time to recover from the failure and roll in the changes.

With all of these benefits, the Pervasive Backup Agent is well worth the cost. One downside of this tool is the lack of a "white list", so it is possible that database files can get copied without going into Continuous Operations mode, and then have the file open up WHILE the copy is running, causing Status 46 problems. (See our separate paper on this topic.) Luckily, you can emulate the white list via a simple batch file with a bunch of BUTIL -STAT commands, one for each database file. Of course, if you are creating a list of files for BUTIL, you could also use ContOps manually and save the cost of BA, too.

About the Author: Bill Bach is the founder and President of Goldstar Software Inc., a firm dedicated to providing installation, configuration, optimization, and troubleshooting services for Btrieve and Pervasive.SQL engines and applications. For more information, visit www.goldstarsoftware.com.