EVERYTHING YOU DID OR DIDN'T WANT TO KNOW ABOUT OPEN FILE BACKUP

Normally any discussion of backup and restore is a pretty dry topic unless you've just had a data losing experience. Perhaps even drier still is talking about alternative methods to process open files during backup! While some may feel that this subject is "esoteric," it is nevertheless important to understand the two major approaches at processing files that are exclusively locked during backup to better insure a recovery after a failure. For simplicities sake, we call the methods the "Locked File Backup" or LFB approach and the "Application Program Interface" or API approach.

Both approaches have their pros and cons. The LFB method is generic in nature and doesn't really care what files are open while the API method uses programming routines supplied by the appropriate software author that typically guaranty file integrity and other options not available when using LFB technology. Bottom line, your organization may need to implement both generic LFB and API based agents for clean backups.

Backing up a Lotus Notes database is a prime example of where LFB technology has been required. This is because previous to Domino, there were no APIs allowing third party developers to create API based agents. LFB technology also has two approaches. The first is stand-alone while the second is integrated technology. An example of a stand-alone LFB version is from St. Bernard Software called "Open File Manager". This software is generic in nature and is configured for the backup product used but requires support from two separate vendors. An integrated example is like the LFB agent from UltraBac.com where it's an UltraBac version 5.5 backup option.

API based backup and restore agents are very different from LFB technology and are used for specific databases. An originating software author, like Microsoft, first creates the database technology and then develops a set of programmable calls (APIs) so that third party companies can safely interface with the databases for whatever the reason. This is typically so that other applications can read and write to the database for application integration and also to allow backup while the database files are open and active.

While LFB technology can be used to backup relational databases like Microsoft's SQL and Exchange, it is not advisable to use this technology for this purpose when there is an optional API based agent available. To justify this statement, I will attempt to explain how these technologies work along with explaining how at least one backup product from UltraBac.com has a built-in "Open Shared" file backup function that is included in the basic software. UltraBac will also be referenced for integrated locked file backup and API based agents in this discussion.

1. Stand-Alone Locked File Backup

Generic LFB software is designed to maintain a static image of open files on a disk partition until they have either been backed up or until a verification pass has been completed where a block of data on the backup media is compared against the corresponding block of data on disk.

Files that are related on a disk partition are treated as a logical set for backup purposes provided they are not split over multiple volumes.

When the backup program attempts to process the first open file within a disk partition, the LFB process waits for a pre-determined pause in write activity before deciding that it is OK to continue (all related files are synchronized with no pending partial writes). A pre-write cache is dynamically allocated by creating a standard disk file that will be used for all open files within the partition to be processed by the backup program.

After this, any file write operations go directly to the correct file while a copy of the data is placed by LFB software in the pre-write cache. When the backup application gets to a part of a file that has been changed while the backup is in progress, the LFB software uses the original data from the pre-write cache to satisfy the read requests. This results in the subsequent file written to the backup media to look exactly like it did at the time when backup started. The LFB process allows active users uninterrupted computer operation during the backup process.

Stand-alone LFB can release the pre-write data held in cache for a file after it has been backed up, after any optional verify operation, or upon completion of the backup program.

Note that the restorability of a database backup depends on the integrity of the files at the time of the pause and therefore cannot be 100% verified for integrity using LFB technology.

2. Built-in "Open Shared" File Handling"

UltraBac has a built-in feature since original release in 1995, for backing up "Open Shared" files automatically with an open status entry written to its backup report and is included in the basic license fee. An example of this functionality would be where a user has a Text.Doc or Access DB that is a flat file format open at the time of the backup. UltraBac comes along processing files in sequential order, freezes the file, backs it up and then releases it. Any changes that have taken place in the file while the backup/verify occurs is stored in paged memory and will re-synchronize when the file is released. This cannot be used to back up locked files (opened exclusively by an application) and should not be used to back up relational database files even when opened in shared mode.

3. Integrated Locked File Backup Agent

UltraBac's LFB agent was designed for backup of exclusively locked files such as Excel, FoxPro, Btrieve, Lotus Notes and other accounting databases and mail applications. It is similar in functionality to other available LFB software where it waits for a pause in the activity of the application, but is different in that it freezes all the files on a disk at once, and then starts the backup of those files. This automatically guaranties that any related files will always be in-sync for backup. The software places any new changes into a Pre-Cache location until the backup/verify is finished, and then releases any such changes to re-synchronize the backed up files automatically. While this agent can be used to backup any type of locked file, note that it cannot determine the accuracy or integrity of a file and is an all or nothing approach. LFB technology is not advocated for use on relational database backups when API based agents are available and must not be used when such databases are split over multiple volumes. Additionally, large-scale relational databases with hundreds of thousands or millions of records should not be backed up using LFB technology. Relying on a pause to assume a quiescent state for all related files when dealing with large databases is laden with risk.

UltraBac's LFB functionality is integrated into the basic software and has a single point of program development and technical support.

4. API Based Backup Agents

API based agents for the backup and restore of relational databases are created by integrating the application program interface calls (API's) supplied by their original developers (like for SQL and Exchange from Microsoft) to handle specific database situations and requirements.

In essence, the API's link all files that are related to the database, freezes the disk(s), backs up the entire database or just specific logs and verifies the integrity of the files being backed up. Users may remain active and use applications that read and write to the database. Such changes are made into a Paged memory location that will be released and re-synchronized when the backup/verify operation is complete. A third party backup product, like UltraBac, simply harnesses these API's for seamless operation.

The major advantages in using an API based agent to backup and restore databases are:

- Option to perform a full database dump.
- Option to perform an incremental by backing up the differential log.
- Automatically backing up a database split over multiple partitions or disks.
- OEM verification of database file integrity upon backup.
- Option to perform a full database restore.
- Option to perform an incremental or differential log restore.
- Option to perform individual database table restores.

In comparison, LFB technology only allows backing up and restoring a complete database (all or nothing).

In summary, both LFB and API based agents have their place in an organization's backup strategy. They are two very different tools that sometimes appear to do the same job. Please note, however, that this similarity in operation is only on the surface! In reality, these two methods use very different technology. An example of backing up a valuable relational database using LFB technology would be a little like playing Russian roulette with your business's data. Just like you cannot guaranty that a chamber will come up empty, you cannot verify the integrity of a file as it is being backed up and thus significantly increase your chance of a future restore operation failure with every additional backup. Using a worst-case scenario (that has happened many times to real businesses), a corrupt database could be backed up for days or weeks without detection until every backup tape was contaminated! Too often this is only discovered when attempting to recover data from some event requiring a restore operation.

Investing in backup is the same as buying business continuation insurance and any consultant or backup guru worth his "salt" would advocate buying the best hardware and software available that would provide the best chance for a successful recovery for not if, but when a failure strikes. API based agents, when available, should always be the preferred approach for backing up files that are open. LFB technology should be reserved to pick up all other open files during backup! This strategy may impact the bottom line with a higher initial cost but like any good insurance program, it offers a big payback when considering the alternative to being able to successfully restore a critical database file and continue doing business as usual.

Written by Morgan Edwards, President and CEO of UltraBac.com.

###