

Backup and Recovery Strategies for Your SQLBase Databases

By Mike Vandine

Database recovery is data administration's response to Murphy's law. Inevitably, databases are damaged or lost because of some system failure, human error, hardware failure, incorrect or invalid data, program error, computer virus, or natural catastrophe.

Since an organization depends so heavily on its database(s), there must be a strategy in place to restore a database quickly and accurately after loss or damage, and since the recovery of your data is only as good as the backups that have been done, this strategy must also include the consistent backing up of the data as well.

The following basic facilities must be in place to insure accurate recovery:

- 1. Backups of the data sufficient to bring a database back to a consistent state regardless of the current state of the database.*
- 2. Logging of transactions and database changes.*
- 3. Recovery of the database with sufficient options to restore the data with minimal data and time loss.*

I will be explaining the options of each of these facilities that are available for the SQLBase database. Each of these options should be reviewed with your particular situation in mind. You must select the best options for your operation, based on various factors, including acceptable loss/down time, amount of disk space available, and the size of the databases.

What is a backup?

A backup consists of the database (copied as a .BKP file) and all associated log files required to bring the .BKP file to a consistent state at the time of the backup. Keep in mind that, regardless of the type of backup that you do, the data that is backed up must be uncorrupted, so it is a good practice to do a 'check database' before the backup and only do the backup if the database is in a stable state.

Backups can be done either online using SQLBase backup software (while the SQLBase server is running and users are attached to the database) or offline using standard operating system commands, ie. COPY (while the SQLBase server is down or the database is deinstalled) then a 'set nextlog' command. The option chosen most often is the online backup, since most operations find it difficult or inconvenient to find a time when all users are off all databases so the SQLBase server can be shut down gracefully or all databases can be deinstalled. This is necessary because the SQLBase server holds a database as an open file from the time the first person connects to it until the database is deinstalled or the SQLBase server is brought down.

What are log files, why do I need them and why won't they go away?!?

SQLBase uses transaction log files to hold before and after images of changes to the database as well as periodic checkpoints for transaction control. They are used for crash recovery (automatically done after a power failure, server crash, etc.), rollback (when changes made to the database are not required or a failure causes SQLBase to roll back a transaction) and data recovery (restores of databases).

Note well that a database consists of the .DBS file and these log files. If the log files are deleted, the database is made useless! By default, SQLBase automatically deletes log files whenever they are backed up or when they are not needed for crash recovery or rollback. This behavior may be changed by setting LOGBACKUP ON for a database using SQLTalk. This will specify that the log files are to be kept until they are backed up by a 'backup logs' command. This is the only way that these logs can be deleted! This option must be turned on if you are doing a 'backup database' or 'backup logs'. Conversely, there is no need to set this on unless you intend to use 'backup logs' and incorporate a 'rollforward' capability into your recovery strategy, ie. you will only do 'snapshot's.

You can specify the maximum size of the log files to be created for each database by setting the LOGFILESIZE using SQLTalk (default is 1 megabyte). Logs are created at a minimum size and grow to the limit. They can be pre-built at the maximum size specified by setting the LOGFILEPREALLOC on for the database using SQLTalk (default is off). Some ramifications are that if you create a lot of log files,



you can set the maximum size of the logs to be large (say 5 MB) and pre-allocate the file size. This will result in less I/O required to build new files and expand the existing file.

Log files will only be deleted if eligible for deletion whenever a 'release log' or 'backup logs' is done if they are not 'pinned'. A log file is pinned:

1. By a current transaction logged to the log file. Can only be unpinned by a commit or rollback by that transaction.
2. If a last-but-one checkpoint was done using the log file or a previous log file. For example, if a checkpoint was done in 6.log then that log and all subsequent logs will be pinned. This can be unpinned by doing a 'release log'.
3. If LOGBACKUP is on and the log file has not been backed up. Only a 'backup logs' will unpin this. Doing a 'backup snapshot' will have no effect on the pinning of this log file.

The log files that have been backed up will have to be manually deleted on a regular basis. The only log files that must be kept are the ones that directly reference the .BKP file in the backup area. For example, if you do a snapshot backup on Friday, all log files prior to Friday's backup may be deleted. Be warned: these log files can eat up a lot of disk space.

Online backups

The advantage of doing online backups is that there is no disruption to users currently accessing the database. This is extremely important for operations which require 24 hour access to the database. The format of the online backup is:

```
BACKUP DATABASE
      LOGS          - Select one of these options
      SNAPSHOT
FROM database-name - Not required if attached to the database
TO directory-name or Netware-path
ON CLIENT          - Select one of these options
  SERVER
```

There are two basic options for an online backup. The first is a 'backup snapshot' (complete in itself) and the second is a 'backup database', a 'release log', then a 'backup logs' (all three are required for a consistent backup).

Regardless of the option you choose, the sequence of events of a backup is the creation of the .BKP file in the chosen backup area (backup database), the rollover of the most recent log file, since that log contains all the information about the current backup (release log), and a copy of all pertinent 'unpinned' log files before the current log to the chosen backup area (backup logs). The snapshot option automatically does these three steps for you.

When performing a backup, the destination for the data can either be the client machine or the network itself. This is specified in the 'directory-name' option of the backup command. The major thing to be aware of is the difference between the 'on server' and 'on client' options of the backup command. The 'on client' option will pull all data and logs to the PC, then push the data to the specified directory (even if it's going to the network server). The 'on server' option will do the backup directly on the server without pulling the data to the client first. This can make a world of difference in the speed of your backups as well as reducing network traffic! If you are using the 'on client' option, ie. storing the data on your PC, the directory would be specified as a drive and directory, for example, C:\SQLBASE\BACKUPS\DB1 or a network mapped drive, ie. F:\BACKUPS\DB1. If, however, you are using the 'on server' option and you're using a Novell server, the directory must be a Netware server path. For example, if you have your databases backed up in a structure of: SERVER1 \SYS:SQLBASE \BACKUPS \DB1 a snapshot for the DB1 database could look like:

```
BACKUP SNAPSHOT FROM DB1 TO \SQLBASE\BACKUPS\DB1 ON SERVER;
```



Keep in mind that you do not have to have write or connect access to the server or the directory where the data is stored. The server will quite happily store a backup in the server root directory if you ask it to, so be careful where you tell the backups to go! Also don't make the mistake of putting the backups into a subdirectory of the database itself. When a database is dropped (prior to a restore), the whole database directory is deleted including any subdirectories. Bye-bye backups! Ouch!

Backup Snapshot

Backup snapshot is by far the easiest method of backing up your databases. It copies the database and the logs necessary to restore the database to a consistent state. Before backing up the logs it forces a log rollover so the current active log is included in the backup. This type of backup only requires one command to back up your data and only one command to restore your data. This would typically be done once in the evening before the server tape backup and possibly again during the day. A drawback would be that snapshots of a very large database can be very time-consuming if done often.

Backup Database, Release Log, Backup Logs

This method of backing up your data is a bit more complicated but for recovery flexibility is well worth the extra effort. It requires backing up just the database file, releasing the log file (since it has the information about the current backup), and then doing a backup logs. Caution: never back up a database without backing up the logs. This method of backup allows you to back up the database and its logs infrequently, then do a backup of the logs only on a periodic basis, i.e. several times during the day. This would allow for recovering a database and then doing a rollforward of the log files to recover any work that has occurred during the day. How often these log backups would be taken depends on the amount of disk space available and how much data you can afford to lose. This can be a real disk space eater. This method is great for large databases since the time-consuming backup of the database file is done infrequently and the periodic log file backup is done quickly. The disadvantage is that the log files will hang around until they are backed up.

Segmented Backups for Databases Greater Than 2 GB

In the past, a SQLBase database that was larger than 2 GB had to be partitioned. With the 7.5.0 release of SQLBase, this restriction was lifted for databases on the non-Novell operating systems. They now have the capability of growing to 256 GB without being partitioned. However, this restriction only applies to the .dbs file and not to temporary files or to the .bkp file. In order to copy with this restriction on the .bkp file, a new segmented backup was introduced.

You just need to set up a control file and put it in the same directory as the backups. No changes have to be made to the backup statements or programs running. The SQLBase server sees that there is a control file in the destination directory and segments the backup accordingly. The control file name has the same name as the database to be backed up with a .bcf extension. You can use any ascii editor to create this file (ie. notepad.exe).

The format of the control file is:

```
FILEPREFIX    prefix
DIR           directory to back up the segment to    SIZE size in MB
```

You can specify multiple DIR statements to break up the database into chunks less than 2 GB each. Note that the sum of all the SIZE statements must be large enough to back up the whole database. For example:

```
FILEPREFIX MIKE
DIR C:\BACKUPS\MIKE SIZE 1000
DIR C:\BACKUPS\MIKE SIZE 1000
DIR C:\BACKUPS\MIKE SIZE 500
```

would allow for three files in the C:\BACKUPS\MIKE directory:

```
MIKE.1 1 GB
MIKE.2 1 GB
MIKE.3 .5 GB
```



Please note that the files are only created if the database requires the space. In the example above, if the database was only 1.8 GB, the segments would be 1gb and .8gb and no third file would be produced.

What Software Should You Use To Do Backups?

You can use the Gupta -supplied SQLTalk to do the backup command shown, you can use SQLConsole (which uses C/API calls) to schedule your backups or you can write your own program to do the backups to incorporate any special requirements that your site may have, ie. do a check database before backing up.

SQLConsole - This product is supplied by Gupta and, along with its other excellent monitoring features, allows you to schedule backups including check database and update statistics. This is quite a sophisticated product with lots of backup options.

BACKUP.ZIP - This is a freeware program (available from the author) that reads an .INI file to determine what servers and the databases on each server to back up. It uses the C/API routines to connect to each server and database specified and does a 'backup snapshot' with the 'on server' option. This program can be used 'as is' or can easily be modified to add any requirements you have. There is also a program available that does a 'backup database' as well.

Backup Gotcha's

I've hopefully covered most of the gotcha's regarding online backups in the text above. However, to recap:

1. Do a 'check database' before backing up a database. If the check database fails, don't back it up over the top of an uncorrupted backup.
2. If not just doing a snapshot, check that the backup sequence is correct, ie. 'backup database', 'release log', 'backup logs'.
3. Use the 'on server' option whenever possible to save bunches of time!
4. You must use a Netware server path rather than a mapped drive when the 'on server' option is used with Novell servers.
5. Don't delete ANY log files from the current log directory, specified in the logdir= statement in the server's sql.ini file (or the dbdir= statement if no logdir= has been specified).
6. Periodically delete old log files in the backup area to save disk space. Don't delete any that pertain to the current .BKP file. It is safe to delete any .LOG files in the backup area that have a date/time less than the current .BKP file.
7. A 'backup snapshot' will not unpin logs if LOGBACKUP is set on. You must do a 'backup logs' to unpin them.
8. Don't put the backups in a subdirectory of the database directory.
9. If the client machine that started a backup gets rebooted in the middle of the snapshot, you may get a message stating that a backup is already in progress when trying to back up the database again. This can occur if you connect to the server before the previous session has had a chance to clear itself. You can get around this by killing the session that aborted with SQLConsole or by bringing down the server and bringing it back up.

Offline backups

The advantage of doing an offline backup is that the data can be backed up directly to the archive media. This can save lots of disk space because the amount of disk space required to hold a backup is the same as the size of the original database.

To actually do the backup, use an operating system command or utility (for example, the 'copy' command or ARCServe backups) to back up the database .DBS file and all .LOG files in the database directory. If LOGBACKUP is not on, there will be no log files to copy. If it is on, however, you must tell the SQLBase engine that the log files have been backed up or they will remain 'pinned'. This is done



with the 'nextlog' command using Gupta's supplied SQLTalk. You must be connected to the database to do this. The format is:

```
SET NEXTLOG [integer]
```

The number given is the next log to be backed up. For example, if the last backup copied 4.LOG and 5.LOG to the archive media, do a 'set nextlog 6'.

The disadvantage of an offline backup is that the database has to be deinstalled or the SQLBase server must be brought down. This is because the SQLBase server attaches to the database file upon the first connect to it and doesn't release the attachment until one of these events occur. This means that all users must be disconnected from the database to be deinstalled or all users must be disconnected to bring down the whole SQLBase server. Finding a suitable time or disconnecting errant users who forgot to log off when they went home can be a real problem! How is this done?

ABTUSR.ZIP - This is a freeware program available from the author that aborts users from a specified database. The parameters are passed to the program in the 'run' command line.

Recovering Your Databases

What is a restore?

A restore consists of a .BKP file and all .LOG files backed up since the original backup. The .BKP file is useless without the associated logs. A restore is essentially a three part event:

1. Copy the .BKP file to the database directory.
2. Copy the .LOG files to the database directory.
3. Run a two-pass (redo and undo) rollforward process against the new .DBS file.

The format of the restore command is:

The format of the online backup is:

```
RESTORE DATABASE
        LOGS          - select one of these options
        SNAPSHOT
FROM    directory-name or Netware-path
ON      CLIENT        - select one of these options
        SERVER
TO      database-name
```

If you choose 'restore snapshot', no further action is required, since all of the steps in the recovery happen automatically. If you choose the 'restore database' option you must then do the rollforward process. Note that, as in the backup phase, the directory must be a Netware path rather than a mapped drive if the 'on server' option is specified for a Novell server and that the 'on server' option greatly reduces the restore time. The rollforward consists of applying changes made after the database backup to bring the database to a consistent state. The log files contain information on all transactions, both successful and rolled back. The rollforward makes two passes through the log files. In the 'redo' pass, SQLBase locates the starting point of all transactions and applies all successful transactions to the backup. In the 'undo' pass any rolled back transactions are reversed. At the end of the rollback process the database is in a completely consistent state with no 'pending' transactions. The format of the rollforward command is:

```
ROLLFORWARD database-name
TO      END
        BACKUP      - select one of these options
        TIME datetime
END      - select one of these options
```

CONTINUE

The 'rollforward to backup' option will recover all committed work up to the point of the backup completion of the database. This is equivalent to a 'restore snapshot'. This does not continue recovering any additional .LOG files that may have been backed up after the original backup.

The 'rollforward to time' option allows you to recover data to a particular point in time. This is great for rolling back large bits of committed work that shouldn't have happened. For example, if your new whiz programmer just deleted half your database then committed it, you can restore the database then rollforward to the time just before the commit.

The 'rollforward to end' walks through all the log files beginning with the first log file from the backup and continuing until the next numbered .LOG file cannot be found. This recovers as much work as possible. When the last log file has been processed, you will get a message that the next log file cannot be found. This is normal! Just do a 'rollforward end' command to end the rollforward and recovery is complete.

Please note that you must have ALL the database log files and must apply them in order or the rollback will fail. If you are missing any logs, the rollforward operation will stop and request the missing log. If it is available, you can do a 'restore logs' command to make the log available and then a 'rollforward continue' command to proceed. If the log file is not available, do a 'rollforward end' to end the process. The database will only be recovered up to the last log file processed!

Restore Gotcha's

A couple of gotcha's to watch out for when restoring your databases:

1. When a 'restore snapshot' is done you may get the message 'cannot connect to the database'. This is essentially a timing problem. When a restore is done, the old database is purged and the .BKP file is copied from the backup directory to the database directory and installed. The restore process then tries to connect to the database to apply the logs. If there is a lot of activity on the system or the SQLBase server is slow to advertise the newly installed database, the restore process cannot connect to it. The solution is to increase the CONNECTTIMEOUT value in the router section of your sql.ini file. CONNECTTIMEOUT specifies the amount of time (in seconds) to wait after failing to connect to a server before attempting to connect again. For example, if you need to set connecttimeout to 20 in order to successfully connect to a database you've just created in the same session:
connecttimeout=20.

2. If you have been caught with just a .BKP file and no logs, you're pretty much dead in the water. There are, however, a couple of things to try before giving in to despair. They sometimes work, but nothing is guaranteed. Let's consider that the database is called MIKE.BKP. Using SQLTalk, do a set server command and then do the following:

```
RESTORE DATABASE FROM [backup path] ON SERVER TO MIKE;  
<<Database has been restored. Use rollforward to complete recovery>>  
is the message returned.
```

```
ROLLFORWARD MIKE TO BACKUP;
```

(Note that 'to backup' is the only one that seems to work)

```
<<You must restore the database first>> is the message returned.
```

```
ROLLFORWARD MIKE TO BACKUP; (Yes, run it again!)
```

```
<<Rollforward completed>> is the message returned.
```

```
CONNECT MIKE 1 username/password;
```

```
<<Connected to MIKE>> is the message returned.
```

Do a check database to find out the status of your database.

Scenarios

Here are some comparisons of the different methods of backup to give an idea of the disk space requirements/recovery possibilities. We will assume that the log file size is standard 1 MB beginning



with 1.LOG, logs are pre-built at full size and that all transactions are committed when the backups take place.

Snapshot method:

Transactions consuming 4 log files are completed.
Before the snapshot:

Database Area		Backup Area	
4.LOG	1 MB		
DB1.DBS	6 MB		

Note that logs 1 through 3 in the database area are released since all transactions have been committed.

After the snapshot:

Database Area		Backup Area	
5.LOG	1 MB	4.LOG	1 MB
DB1.DBS	6 MB	DB1.BKP	6 MB

Three more log files of transactions are completed.

Database Area		Backup Area	
8.LOG	1 MB	4.LOG	1 MB
DB1.DBS	9 MB	DB1.BKP	6 MB

If the system crashed now, restore capabilities are only available to the previous backup. No recovery of the last three logs worth of transactions are possible because continuous logs are not available (logs 5 through 7 in the database area were released since transactions were committed and not needed for crash recovery or rollback).

After the snapshot:

Database Area		Backup Area	
9.LOG	1 MB	4.LOG	1 MB
DB1.DBS	9 MB	8.LOG	1 MB
		DB1.BKP	9 MB

Note that 4.LOG in the backup area is now eligible for deletion, since those transactions are included in the DB1.BKP file and its date/time will be earlier than the DB1.BKP file.

Backup Database and Logs method:

Transactions consuming 4 log files are completed. Before the backup:

Database Area		Backup Area	
---------------	--	-------------	--

1.LOG	1 MB
2.LOG	1 MB
3.LOG	1 MB
4.LOG	1 MB
DB1.DBS	6 MB

Note that all log files exist in the database area until they are backed up with a 'backup logs' command even though all transactions are committed.

After the backup database and backup logs:

Database Area		Backup Area	
5.LOG	1 MB	1.LOG	1 MB
DB1.DBS	6 MB	2.LOG	1 MB
		3.LOG	1 MB
		4.LOG	1 MB
		DB1.BKP	6 MB

Note that since the log files have been backed up and there are no uncommitted transactions, they are automatically deleted from the database area.

Three more log files of transactions are completed.

Database Area		Backup Area	
5.LOG	1 MB	1.LOG	1 MB
6.LOG	1 MB	2.LOG	1 MB
7.LOG	1 MB	3.LOG	1 MB
8.LOG	1 MB	4.LOG	1 MB
DB1.BKP	9 MB	DB1.BKP	6 MB

If the system crashed now, restore capabilities are available to the latest transaction by restoring the .BKP file from the backup area, applying logs 1 thru 4 from the backup area, then applying logs 5 through 8 from the database area.

After the release log and log backup is done:

Database Area		Backup Area	
9.LOG	1 MB	1.LOG	1 MB
DB1.DBS	9 MB	2.LOG	1 MB
		3.LOG	1 MB
		4.LOG	1 MB
		5.LOG	1 MB



6.LOG	1 MB
7.LOG	1 MB
8.LOG	1 MB
DB1.BKP	6 MB

Note that no log files in the backup area are eligible for deletion since they ALL must be applied to the DB1.BKP file in the backup area to make it consistent with the DB1.DBS file in the database area.

Three more logs are created and another database and log backup is done:

Database Area		Backup Area	
12.LOG	1 MB	1.LOG	1 MB
DB1.DBS	11 MB	2.LOG	1 MB
		3.LOG	1 MB
		4.LOG	1 MB
		5.LOG	1 MB
		6.LOG	1 MB
		7.LOG	1 MB
		8.LOG	1 MB
		9.LOG	1 MB
		10.LOG	1 MB
		11.LOG	1 MB
		DB1.BKP	11 MB

Note that once the backup database and logs is done, logs 1 through 10 in the backup area could be released since the new .BKP file would include those transactions and their date/time will be earlier than the DB1.BKP file.

In Conclusion

If you haven't had to recover your data yet, you're one of the lucky ones. However, the statistics indicate that at some time you will have to! The time to develop a data recovery strategy is now, before you need to get your data back.

Michael Vandine is currently a Senior Technical Consultant for Gupta Technologies and resides in sunny Perth, Western Australia. He's used the Gupta products heavily for the last 11 years and is a Gupta Certified SQLBase DBA. He's also a member of the TeamAssist organization on the Centura newsgroups forum. He can be reached by e-mail at mike.vandine@centurasoft.com for comments or questions or to request one of the freeware programs listed in this whitepaper.