BACKUP PROBLEMS AND SOLUTIONS

From the precon "The Lifecycle Approach: Maintenance Problems & Solutions", by Sean McCown and Jen McCown. Please do not publish this document.

We're going to solve backup issues/needs today, using Minion Backup as the platform. Of course, you can use MB, or use something else to solve these same problems…MB is just what we're using for demonstrations.

PROBLEM: EXCLUDE DBS

We have just a couple of databases that should NEVER be backed up. Or, there are a couple of databases that should NEVER get a certain kind of backup. Examples:

- Temporary or duplicate databases that are going to be destroyed anyway.
- Databases in log shipping scenarios.

Solution: Your system should create backup exclusions using **table based parameters** ("uncoding"), instead of hardcoding or soft-coding (using SP parameters in jobs).

Also note:

- Your system should automatically *not* generate log backup statements for databases in SIMPLE recovery mode. Those statements would generate errors!
- The table based parameter system we recommend handles new and "retired" databases seamlessly through the use of default values. Having default values (in MB's case, the "MinionDefault" row) means the system must have rules for settings hierarchy: default values apply for any database that doesn't have DB-specific configurations defined.

PROBLEM: ORDER DBS

We have some VERY IMPORTANT databases that need to be backed up first! Examples:

- One or more databases have an external mechanism that copies off the backup file to restore, or to save to an offsite data center.
- One database is extremely active, and we want the log backup to happen right at the start of the log backup batch (so it's not growing while waiting on the other log backups to finish).

Solution: Provide for backup ordering, again in a settings table. MB doubles down on this by providing a way to define groups of databases, order those backups, and define the ordering within groups, too.

Also note:

- When ordering log backups, take into account that a log backup shouldn't show up in the process' resulting order list IF (a) it's in SIMPLE mode, or (b) the database hasn't had a full backup yet.
- A flexible enough system allows for some really creative problem solving. With MB, for example, you can create an SP to dynamically order the backups in any way you like. The SP can run before each batch of that type, if you put the SP call in the BatchPreCode column for that backup type's schedule (in Minion.BackupSettingsServer).

PROBLEM: READ ONLY DBS

Read only databases in general do not need to be backed up on the same schedule (or indeed, as often) as read-write databases do. Examples:

- RO1 is a read only database that hasn't been written to in years, and never will be.
- RO2 is a read only database that is altered to be a read-write database only once a year.

Solution: Provide an easy way to treat read only databases differently. As usual, this should be uncoded, using table based parameters (not hardcoded or coded as parameters in jobs). The table in question is Minion.BackupSettingsServer; each schedule can back up all databases (ReadOnly=1), just read-write databases (ReadOnly=2), or just read only databases (ReadOnly=3).

Also note:

- We've been hitting the "use table based parameters!" message really hard, because it's very important. However, most of our demos in this session use the SP parameters. That's fine for a demo standpoint, but why would we include that for MB in the first place? Answer: because you also need the ability to run the occasional manual backup; the system *must* provide for that. (Plus, some people just won't listen to the "use table parameters!" message, and insist on multiple jobs with SP calls. Hey, to each his or her own, right?)
- Later, we'll see how to set up different schedules for read only databases (say, once a month instead of daily or weekly).

PROBLEM: BACKUP PATH

Traditionally, it's a giant pain to change the backup path: you get to update paths in a dozen jobs or more on each and every server. Examples of why the backup path would change:

- One drive runs out of space
- The naming convention changes.
- A new SAN is installed.
- We need to back up different databases (or different backup types) to separate locations.
- Nobody knows, it just has to change.

Solution: Store backup paths in a table (of course), *separately* from the normal backup configuration. Any change to the backup path is a simple UPDATE statement away.

Also note:

- Storing the paths separately from the backup settings is database design 101: one kind of backup for one database may have more than one path (due to mirroring, striping, or as we'll see later, copy and move operations). So, paths must be stored separately.
- Note that this also lets us set a different backup file retention for *every backup and every backup type*, if we want to. So I get to keep my FULLs for a month, DIFFs for two days, and LOGs for 24 hours if I want to, and all that configuration is sitting in a table – not hidden in 10 different jobs.

PROBLEM: MIRROR BACKUPS

Native SQL Server allows you to mirror a backup. How do you set that up in a backup solution?

Solution: Well of course we put that configuration in a table. But instead of creating a whole new table, we put it in a logical place: the paths table. After all, all that's needed to set up a mirrored backup is the addition of "MIRROR TO DISK = '*location*'" in the backup statement. Examples of a requirement for mirroring:

- For some reason your company insists on mirrored backups instead of just copying the backup file. (I'm a little biased.)

PROBLEM: BACKUP TUNING

You have a database that takes a *long long* time to back up. Examples:

- The traditional huge database that's still backing up at 9am.
- A highly busy database whose log grows almost faster than it can be backed up during the day.
- Any database that you want to *restore* quickly.

What's to be done?

Solution: Backup tuning. Better yet, dynamic backup tuning. Better still, *timed* dynamic backup tuning...configured in a table. Backup tuning is the process of throwing lots of resources at the backup so it goes really fast.

Teaching you backup tuning is beyond the scope of this session, but you should know that tuning settings should change for a database as the database grows. The Minion.BackupTuningThresholds table lets you set all that up. (This, by the way, would be absolute agony to manage using SP parameters and multiple jobs, and impossible to reasonably make dynamic.)

Also note:

- When you tune your backups, you're also determining the settings to use for faster restores.
- Find the backup tuning session on the www.MidnightDBA.com Events page.

Also-also note: Provide yourself a way to test your settings!

- In Minion Backup, BackupStmtGet tests your path and tuning settings.
- And the SP parameter @StmtOnly is there to test ordering settings, and exclusions.
- Later, we'll see @TestDateTime, which tests schedule settings.

PROBLEM: SHRINKLOG

If the log grows abnormally large, you may not want the file to stay that big. Examples:

- If the log file(s) get too big, they may choke out other log files on the drive and bring databases down.
- You want to keep the log to a reasonable size, to keep restore times down.

Solution: Enable and set automatic log shrink operations, when a lot file exceeds a configured threshold. (Make sure that you also configure a target size, so as not to overshrink the file.) In MB, this is done in the Minion.BackupSettings table.

Also note:

- Shrinking logs is considered bad, and it can be if it’s way way overused. But sometimes it’s totally necessary. If your log chokes out other databases because it has 90GB of unused space, you *have* to shrink it to get things back on track. Now we're just arguing over whether you're going to get out of bed to do it after it's already caused problems, or have the system take care of it before it becomes an issue.

PROBLEM: COPY

We need backup file(s) duplicated in more than one location. Examples:

- You need to copy the backup file to a development/QA instance, or even to another data center.
- You're having network issues with mirrored backups, so you're copying the files instead. This is more stable because the backup completes first.
- You need to copy the files to more than one location on a static schedule (like, daily).
- You need to dynamically change the location you're copying the files to. This could be based off of the day, or even the data center.

Solution: Your backup solution should provide for copying backups, which lets you accomplish the same thing without extra risk to the primary backup. In MB, this is configured in Minion.BackupSettings and Minion.BackupSettingsPath.

PROBLEM: SERVERLABEL

The location of the backup traditionally depends, in part, on the server name where the database resides. But if the DB is in a failover scenario of any kind, that path could change.

Solution: Define a server label to be used by all nodes of the scenario. Or, don’t configure the backup path to depend on the server name at all. Or, both.

PROBLEM: DATACENTER

In a failover scenario, the backup routine may need to change one of a number of configuration settings once the database has failed over. For example: changing the backup path used.

Solution: Create code to detect which node the database is on, and alter (or enable/disable) settings as needed. Have this code run before each batch of backups.

PROBLEM: CERTIFICATES

Certificate backups are often badly overlooked.

Solution: Back up your certificates. Make it as automatic and seamless as possible. In MB, once you configure server certificate backups, they are backed up every time the master database is backed up.

PROBLEM: MISSING BACKUPS

Periodically, one or more backups in a batch will fail. Examples:

- You’re out of space for backups.
- There are random network issues.

- Someone turned of SQL Agent again. (I'm looking at you, Al.)

Solution: Set up a second schedule to check for failed backups, and rerun them.

Also note:

- Without an automated "retry" mechanism, your choices are to have the NOC wake you up, or restart the backups when you get in to the office. Of course, by the time you come in it may be too late because the business day has already started and you can't afford to drag down the system. And you could throw off your deletion schedule drastically.

PROBLEM: LIVE INSIGHT

Backups are running long. Is it stuck? Is it progressing?

Solution: Get live insight into the currently running backups. The way we've done it in MB, a job starts off with every backup, and updates the status in the log tables.

PROBLEM: SCHEDULER

We're firmly against managing backup schedules in jobs, because it's too onerous to manage.

Solution: Keep your schedule in a table, of course, with flexible backup schedule options. In MB, see the Minion.BackupSettingsServer table.

REFERENCES

- Precon site: minionware.net/dbmaintprecon/
- Download free backup and maintenance:
    - minionware.net/backup
    - minionware.net/reindex
    - minionware.net/checkdb (soon)
- Sean: www.Twitter.com/KenpoDBA
- Jen: www.Twitter.com/MidnightDBA