



### Archive Transaction Logs

If you have TSM or another location for Archiving transaction logs that is best and safest. If you are archiving to disk, these **MUST** be in a different directory from active transaction logs, and preferably in a separate filesystem. The logic for a separate filesystem is that this gives you an extra layer of protection – if you’re monitoring for full filesystems, you will catch a logging issue when it fills up your archive log filesystem, and hopefully have time to address it before it also fills your active log filesystem and makes your database unavailable.

To archive transaction log files to a path, you have to set **the LOGARCHMETH1 db cfg parameter to: DISK:/path**. If you’re setting LOGARCHMETH1 for the first time, you may also be changing from circular logging to archive logging, which requires an offline database backup, so be cautious.

### db2diag/db2dump

The db2 diagnostic log and other diagnostic files by default will be in \$INSTHOME/sqllib/db2dump. I like to have them in another filesystem – this ensures that no matter what else fills up or what other filesystem level problems are encountered, I should still get the error messages from those issues.

The location for this data is changed using **the DIAGPATH parameter in the DBM cfg**. It can be updated with an UPDATE DBM CFG command, and changes to it take effect immediately.

### Backups

I like to take backups to a different filesystem. If your filesystems are fully separate I/O paths, this can have some performance benefits. But the real reason is because the backup location is the location you’re most likely to fill up, and you don’t want a filesystem filling up because of a backup and causing an availability issue for your database.

Specify the backup filesystem on any **BACKUP DATABASE** commands you issue, including those from scripts.

### Scripts

I have seen a script go a bit haywire and capture more data than it was thought it would, and fill up the filesystem it is writing data to. For this reason, I like to keep my administrative scripts and their output on a separate filesystem from everything. This makes it much harder for a scripting mistake to cause a database availability issue.

### Others

Obviously there are other filesystems that particular designs may call for – such as a shared filesystem between two HADR servers when doing loads with copy yes. But the above are true to just about every DB2 installation.

### Separating I/O

This post does not cover storage design in depth. Usually the storage I get is a black box. A client allocates the space locally on the server or on a SAN, and I have a very hard time finding out the storage details. If I actually get to specify anything about where the storage goes or what is separate, my first priority is to separate out my active transaction logs and put them on the fastest storage I have, separate from everything else. If I have more specificity available, I like to separate my two data filesystems from my backup filesystem. After that it’s all icing. In only a very few cases do I see indexes and data separated any

more. When I started as a DBA 14 years ago, that was a core tenant of DB database storage design.

### **It is Only Temporary**

I cannot tell you how many stupid things I have been asked to do in the name of “It’s only temporary!” or “It’s only a development environment”, only to have those things become permanent fixtures in a production environment. Just today, I installed DB2 on root, with no separate filesystems at all, after making it perfectly clear I thought it was a horrible idea. My exact words were:

If you want me to move forward with the understanding that a significant percentage of our clients come to us just because of this kind of mis-configuration and the stability issues it causes, then I can. But I want you to understand what a bad idea I think it is.

There really is no such thing as “It’s only temporary” – design correctly from the start and the problems that you encounter are much easier to deal with.