

“As a DBA, where do I start?” by [Mike Walsh \(@mike_walsh\)](#)

- **Materials:** <http://www.straightpathsq.com/where>
- “Restore To ...” backups are important so there is a selection of restore options when something does go wrong. Backups should be done on a regular basis and securely stored.
- Keep prepared SQL code to check the backup and recovery statuses of servers.
- Have database policies for standards, conventions and security. These policies should cover both the Database Administrators and Developers. (Play nice together!)
- Vendor questionnaires can help solve potential problems.
 - “How does X interface with our server?”
 - “How are you insuring the protection of our data and the security of the connection?”
- Should define clear Groups and Roles – don’t forget to limit who has access to the permissions!

“Getting starting blogging and technical speaking,” by [Kendal Van Dyke \(@sqldb\)](#)

- **Materials:** <http://www.sqlsaturday.com/viewsession.aspx?sat=33&sessionid=1288>
- Determine why you want to blog, to what audience, the general topic(s) and so forth. Determining the host of the blog is another consideration, but is more of a personal preference.
- Before you start the blog for public release, have some posts (~10) already in the pipeline so you can keep posting and get a good start.
 - A full pipeline is a good tactic against writer’s block preventing you to continue posting.
 - “Top Ten ...”, “Three Things ...” and such are helpful posts and can be expanded or explained in later posts.
- The personality of the writer should always show in the writing, but your target audience and reason for blogging should determine how much is personal versus professional.
 - Ask a favorite blogger for some tips or criticisms if you want some helpful feedback.
- Do you want a schedule? If so, make sure to stick to it, and if you go for an extended break, due to a trip or something, announce that you will be on break until you can post again. If you’re pipeline has some ready posts, you can still adhere to the schedule even while away.
- For additional help, check out [Brent Ozar’s “How to Start a Blog”](#) (December 2008)
- *For any notes on Technical Speaking, please refer directly to the presentation.*

“Tips For The Lazy But Driven DBA,” by **Timothy Ford** (@sqlagentman)

- **Materials:** <http://thesqlagentman.com/presentationfiles/>
- Five reports that help make the work easier:
 1. Missed Backups
 2. Login Changes
 3. SQL Agent Job Failures
 4. Log File Concerns
 5. Instances, Databases, and Applications
- Pre-made reporting and/or maintenance scripts in general make the work easier.
- Shrinking your logs will cause fragmentation, so be sure it is what you **need** to do, not just **recommended**.
- The default settings are not always the best **practical** practice, so make sure the server settings suit the environment and load needs.
- Quick-to-access SQL templates can be a time saver; adding parameters to it, even more so.
 - If you make the template into a functional call, beware of optional parameter problems. (*Refer to “Bad SQL” further down for an explanation.*)

“SQL Server Data Compression 101,” by **Patrick LeBlanc**

- **Information:** <http://www.sqlsaturday.com/viewsession.aspx?sat=33&sessionid=1066>
- “using compression = [ROW/PAGE/none]” (*Congratulations, class could be over now!*)
- Sp_configure: set default compression to 1
- The **ONLY** trade-off for compression is the CPU time/cycles.
- Backing up to and restoring from compressed files can be quicker than non-compressed files, but again the trade-off is if you can spare the extra processing power required.
- Depending on your records, backing up by ROW or PAGE can be smaller. Some tables may benefit more from PAGE compression, which does ROW compression followed by two additional compression methods, while others may benefit from just ROW.
 - Using SQL Server compression estimates you can do quick reviews to which would be recommend. (*I am inquiring about the code examples of this...*)
- ... (*I plan to write more at a later time, but I took poor notes –being more of a developer – and haven’t had found the materials yet to expand on my very condensed notes.*)

“You Can Improve Your Own SQL Code,” by [Mike Walsh](#) (@mike_walsh)

- **Materials:** <http://www.straightpathsq.com/ucandoit/>
- Continue to do code reviews to optimize performance. This doesn't mean you need to distrust Developer code, but improvements can almost always be made.
 - *As a Developer myself, I know to allot some review time every couple of months to look for processing, resource and time enhancements – I know I can't be perfect always.*
- Just because you've achieved the desired result set, don't say you're done; instead review the performance of the code to make sure that the code isn't doing more work (CPU, memory, disk reads, etc...) than it needs to.
 - Is the SQL scanning, seeking or using keys? Can this be improved by tweaking the code?
- Plan for larger scale data sources so future expansion doesn't upset desired SQL performance.

“Bad SQL,” by [Geoff Hiten](#)

- **Materials:** <http://www.sqlsaturday.com/viewsession.aspx?sat=33&sessionid=1040>
- Make the SQL something easy to follow for yourself and others; otherwise you will have editing or expansion problems later on. Documenting long SQL is good practice as well.
 - “FROM a,b,c WHERE a.*,b.*,c.*” → “FROM a JOIN b ON b.* JOIN c ON c.*”
 - “FROM a JOIN B WHERE a.* OR b.*” → “FROM a WHERE a.* UNION FROM b WHERE b.*”
- Optional parameters in a procedure (function *for developers*) will increase the number of reads unless cache is cleared every run. This is caused because the procedure is optimized after the first call using the parameters used then and any calls with different parameters will cause a much larger number of reads due to the cached, “optimized” procedure.
 - Ex: We have a procedure (P) that accepts two optional, integer parameters. If you call P(null,<int>) the procedure is optimized for it and cached, so calling anything but P(null,<int>) will result in more logical reads than necessary. For best results, make the four separate procedures; you can always cater to the Developers by creating another that will pick which one to call using the optional parameters procedure.
- There can be a performance difference when using DISTINCT, instead of GROUP BY, along with a JOIN. The reason is because the DISTINCT will still gather and pass all records while only showing the distinct returned records while GROUP BY will actually only return the distinct ones.
 - *I made a very hasty note that “this is true unless the sub-tables actually do have multiple records of the key or index listed,” but I am unsure if that was completely accurate.*
- Finally, don't be an idiot when writing code. *Okay, this is more of my own note, but it was the beginning basis of the class.* The ideal code would use the most minimal amount of time, CPU, memory and reads, but just knowing enough not to maximize these is an improvement.
 - Look back at “You Can Improve Your Own SQL Code” above for additional tips.