

Lesson 9 Transcript: Backup and Recovery

Slide 1: Cover

Welcome to lesson 9 of the “DB2 on Campus Lecture Series”. We are going to talk in this presentation about database logging and backup and recovery. My name is Raul Chong and I’m the DB2 on Campus Program Manager.

Slide 2: Agenda (1)

This is the agenda for today.

Slide 3: Agenda (2)

We talk about database logging first.

Slide 4: Database logging (1)

So what is database logging? Well, database logging allows me to store any changes I am making to the database: insert, update, delete, etc, etc. So what is the purpose of logging? Well, when you have to recover information, you can use the logs for that purpose. So, there are situations when sometimes there is a crash on the system and your computer goes down; what’s going to happen with the things you were doing at that time with the database? Or there may be cases where everything is working fine, but there is a human error. And... people are inserting incorrect information into the database. How can I recover from those situations? Database logging allows me to pretty much recover from these types of problems and ensures that my data in my database is in a good, consistent state, and also contains valid information. Now logging is always required in DB2, you cannot really turn it off entirely, but you can turn off logging for specific tables or columns. And if you use temporary tables (that is explained in previous lessons), then you cannot log those temporary tables.

Slide 5: Database logging (2)

So let’s take a look at this chart to explain logging in a little bit more detail.

Demo: Switching to Paint

Actually, let me show you the same chart but using this Paint application. Let’s say in this case, I have a table space with a table T1 and a table T2, and this table space is stored on a given disk, let’s say disk 1. Then we have the buffer pool which is the cache for the database. It is in memory. We have the logs which are used for recovery and they are stored on a different disk, let’s say disk 2. The reason for this is you don’t want to store your database information or your database data and the logs, which you use to recover that data in case of a crash, on the same disk. Because if for any reason the disk gets corrupted, then you are corrupting not only the data but also the data that is used to recover the data. That is why you normally want to keep that in separate disks. Now, there are two ways to store information in the logs.

One way is to issue a COMMIT statement. The way I normally explain this is by comparing it to a Word document. So, let’s say, you are working in MS Word and you’re writing a nice and important document and then you want to ensure that information that you’ve written is safe on the disk. So you will click on the Save button or you go to File > Save to make sure the information you’ve written so far is saved on the disk. Now you can compare clicking on the Save button to issuing a COMMIT statement. When you issue a COMMIT statement basically whatever you are doing in your transaction at that time (you should COMMIT), you

basically save that information into the logs and that means later on they could be recovered. The other analogy I use with MS Word is that sometimes you are working on MS Word and you are not clicking on the Save button, however, at the bottom of your screen when you are working on MS Word, it may say 'autosaving'. What does that mean? Well, you didn't click on the Save button, but automatically Word is autosaving some of this information for you. So in the same way, DB2 or a database works like this where you may not be issuing a COMMIT statement, but that information is still being saved on the logs.

Alright, now, let's say, I have a table T1 and the way it works is, in this table I have some record. I am not going to put the entire record, but let's say the name column has a value of Peter, in table T1. Now I am going to, let's say, I am going to be issuing some statement like, update T1 set name equals to Paul, where name equals to Peter. With something like this, I am trying to change the name where the current name is Peter. And what is going to happen is when I issue this command, first of all, DB2 will bring the page from this disk into memory (because most of the things are processed in memory). So here I am going to bring the page where you have the value of Peter, then I will perform the change and now this is going to be changed to say Paul, right? At the same time, in my logs, I will be storing information like this: I will be saying the old value is equal to Peter. And the new value is equal to Paul. So these are the types of information I will be storing on my logs and I can run another update statement like this one. And in this case, I would say update T1 set name equals to Mary, where name equals Paul. So in that case because I already had the page in memory, I can just simply make the change here and put the new value which is going to be Mary. And then over here, I have another entry where I am going to put that the old value was Paul and the new value is Mary. So in the logs I am recording all of this information. In the buffer pool I just have the latest information, which is that the current value for this particular record is Mary. And in the disk on the table space, this information still says Peter.

Now this looks like it is inconsistent, but the reason is that we don't need to change the value here right away because we already changed the value; we have already stored this information on the logs, so it doesn't make sense in terms of performance to have (every time we make a change), to be writing to the disk, to two disks twice. So I don't want to write to the logs, which is disk, and then I want to be writing to the table space or this table, which is also on another disk, and every time that I am doing an update or any other operation because in terms of performance it wouldn't be good. So, that's the way this works because of the performance reasons. And now let's say I have many pages that I brought to the buffer pool (which is memory) I have been using them, and eventually, I want to bring another page to memory (to the buffer pool), but because most of my pages, or because most of my buffer pool contains information that has not been externalized (meaning that the information is not yet reflected on this disk), then I cannot really, (these are basically called dirty pages); I cannot really overwrite them, so at that time there are different methods that would clean this buffer pool for you.

So in DB2, for example, we have this parameter called change_page_threshold (chnpgs_thresh), which is shown here. And by using this parameter, let's say, I set it to 80 percent. What that means is, when 80 percent of my buffer pool contains dirty pages, then at that time DB2 will reach this threshold and will flush those pages and externalize them to the disk. 'Externalize' means basically I am making the information that I had on the table space to be more accurate and more up-to-date with the information that I had in the buffer pool. So, now I have moved or changed the information, here or externalizing information, here. You would say Mary, and then all this information may be basically, or could be deleted

because the pages that were using this are no longer required to be there, and they could be used by some of the other process to bring their pages there. So, anyway, in the process, externalization means that the database now has inconsistent information and when that happens, when I have here my information in this disk, then some of the logs where contained information are no longer really required for that particular normal processing, because I already have this information stored here. As we will see later, when you have some human errors, it is good to still keep these logs even though they are already externalized on the other disk.

Back to slide

So, moving on to the next slide...

Slide 6: Types of logs

There are two specifications in terms of types of logs. The first specification is to say that there are primary logs and secondary logs. Primary logs are pre-allocated and secondary logs are the ones allocated dynamically as needed, or as I use them. For example, let's take a look now at this other slide. Let's say I have one transaction, which is transaction A, and it starts using this log. Then I have another transaction, just say transaction B, and it also starts within this log. Now transaction A, completes here, but transaction B continues here and then it continues here and so on. Well, transaction A, it starts here and it finishes here, so it was a COMMIT that finished, but transaction B continues, so that means I can't really reuse this primary log here because transaction B has not finished. I can't overwrite this log and we'll talk about circular logging later, where you may understand this better why we can't overwrite this log.

But anyway, in this case, transaction B has not finished, so if I need more logs because transaction B has not finished and I have set my primary logs to be three, then I would need to generate, dynamically, another file which would be called secondary logs and transaction B will continue here. And if transaction B still continues, then we have to use another file. Another secondary file will be generated, and then, if for any reason transaction B still needs more logs, then I will get an error from DB2 saying that there are no more logs available. So the default behavior may be to get the error message to draw back that transaction.

Now there are different parameters or many, many parameters that can help change the behavior of logs. You can also have infinite logging, for example, where you just log, depending on, basically, the size of your disk and so on, and so on. But this just explains to you further, the type of logs. These are called primary logs and these are called secondary logs.

Switch to cmd window

And these, the number of logs are determined by parameters. So, for example, if I first connect to a database, if I connect to SAMPLE, then I can run the command `db2 get db cfg`. I don't need to put the clause, for example, (because) I am already connected to the sample database. And then (since) I have a (I am using Windows but I have a UNIX simulator), so I am going to **grep** (grep command) and I am going to grep for the word 'log', and it's going to show me all the 'log' related parameters. So, for example, I have the log buffer size, this is the size of the buffer for the logs, logfilesize. So this is the size of each of the files, which is 2000... 4 KB. So it is the unit Kilobyte. So 2000 times 4KB; so it's about 8000 KB or 8MB for each of the files. Then here, LogPrimary, this is the parameter that tells me how many logs are going to be primary; in this case, there are three. And then logsecond, how many

secondary logs will I generate dynamically. So in this case it's 2 that will be generated dynamically, which is default. So these are the default values for logprimary and logsecond. And that's why you will see this example here matches what we have for logprimary and logsecond: 3 primary and 2 secondary logs.

And we can also see here this parameter called NEWLOGPATH which is pointing to DB2\node0000\SQL00002\SQLOGDIR. So if I go to that directory, DB2\node0000\SQL002\SQLOGDIR, I can see here the three primary logs. But I will not see the secondary logs because these ones are going to be created dynamically when I need them. One way to specify a good size for the logprimary would be to monitor your workload and determine when or how many secondary logs are being created. And, say 4 secondary logs are being created, and then you may want to add those 4 secondary logs to logprimary, so you don't need to be generating these secondary logs at run time because it's not good for performance.

Switch back to ppt

Anyway, so that is one specification for logs; primary and secondary. And the other specification is active logs and archive logs. Active logs are logs where the information has not yet been externalized, meaning it has not been written to the table space disk (the disk with the table space). And archive logs are the ones where all the information from those logs have been externalized, meaning the table space has the up-to-date information. And this is going back to the previous example about when we are updating from Peter, Paul and Mary. When Mary, which was the last in the table is ready to be externalized, and it's in the table space, then the log would become an archive log.

Slide 7: Types of logging

Moving on to the next slide: now we are going to talk about types of logging. Before we talked about type of logs, now we are talking about the type of logging. And there are two types of logging. We have circular logging and we have archive logging. So circular logging means you are going to be reusing the existing logs, so this would be used normally in test or development systems and because you are going to be reusing logs; basically, the archive logs are the ones where the information had already been externalized. So those ones could be overwritten because you know the information is already in the table space disk. So I don't really need those logs, so I could overwrite those logs and that is the procedure that is used in circular logging.

Now the problem with this method would be when there is a human error. Let's say, you input incorrect information into the logs and now you have overwritten those circular logs, then you have a problem in that situation because you can't go back to those old logs to get the correct information that you had in there. We are going to talk more about this in the next few slides.

And the other one is archive logging where basically you don't get rid of any of these logs. You don't overwrite any of these logs; you just store them maybe in a different media.

Slide 8: Circular logging

So, circular logging again: the way it works is that you start working with log primary one, another log primary, another log primary, and let's say if this information in P1, in this primary log, has already been externalized, and I don't need this information anymore in the log because it's already externalized in the table space disk, then I can potentially overwrite

from this log. And the same thing will continue and, again I can overwrite whatever has been already externalized. Even I can go P1, P2, P3, S1, S2 and let's say P1 now, the information has already been externalized, then I can go again and reuse it. So it will be, it's called circular logging, and again it would not be good to use in production environments. We'll talk later about crash recovery, and basically archive logs are not needed for crash recovery, so they could be overwritten. Anyway, I'll talk more about this in the next few slides.

Slide 9: Archive logging/log retain

We have archive logging, which means you are not going to get rid of any of the logs. So let's say you started with log number 12, and then you move on to log number 13, and then number 12 is archived; the information there is not really required for crash recovery because the information has been externalized. The same thing, we move on to 13, same thing happens to 13, the information is not really required; the information has been externalized. Then I moved to 15, where 15 is an active log because the information has not yet been externalized; the same with 16. Now number 12 and 13 can be moved to another media and these are archive logging or archive logs where the information has been externalized, so I can move it to a different media. Number 14 is also an archive log, but it's called ONLINE ARCHIVE because we still keep it in the same location as the other logs, but the information there is still, and has been externalized, so it could be moved at a later stage to another media. This move could be done either manually or using an automatic way of doing it, which in the past used to be called USEREXIT.

Slide 10: Launching the configuration database logging wizard

Anyway, now if you want to configure logging from the Control Center, you can right-click on the database. In this case, the database is SAMPLE and you right-click it. You can choose Configure Database Logging.

Slide 11: Database logging wizard

Here is where you can choose either what you want: circular logging or archive logging. If you choose archive logging, you would be asked to run a backup first. And we haven't talked about backup yet. We will talk about that in the next section.

Slide 12: DB2 logging parameters

These are some of the logging parameters; some of them we already talked about. You can pause in this slide and read more description about these logging parameters.

Slide 13: Logging parameters (cont'd)

There are many, many logging parameters.

Slide 14: Logging parameters (cont'd)

Again, you can look at the manuals as well to read more about these logging parameters.

Slide 15: Agenda (3)

So now we are going to move on to the next section of this presentation which is Backup.

Slide 16: Database backup

And backup is basically like taking a picture of your system and storing it into a file. I am providing here a very simple syntax of the **backup database** command.

Demo: Switch to cmd window

But you can always find more information from, for example, the command window by typing **db2 ? backup**. And then you get more information about the backup database command. And so there are a lot of choices or options that you can use.

Switch back to ppt

I am choosing a very simple example here. So here what we have is a simple thing that is backup database of the database name and optionally you can specify where you want to store it, otherwise, like any command, it will be storing the database backup to wherever you are executing this command. So let's try an example. The one that is shown here: **backup db sample to C:\backups**.

Switch to cmd window

So before you do that, you have to make sure that there is this directory 'backups'. Let me create it right now. And then I can do **db2**, well, let me just first see if I'm connecting to something. Yes. Let me terminate this connection. And clear the screen here. And let's now do a **backup DB sample to C:\backups**. Ok so right now it's... oh, I still have connection problems. Probably it's from the Control Center. So let me run the command **db2 force applications all** to make sure that I just wipe out all connections. And I can execute this several times because this is an asynchronous command. So sometimes it's good to execute it a few times. Now let's execute the backup command... and now it is executing the backup command.

Now you may say why do I need to make sure that the database has no connections? Why did I need to force all connections? Well, there are two types of ways in which you can execute commands in DB2 for backup, for reorg, runstats etc etc. I'll talk more about those utilities in other lessons. But talking again about backup, you can do a **backup offline** or a **backup online**. What does offline mean? Offline means that there are no connections on your database at the time that you are issuing the backup statement. And by default, that's the mode we are using. We are using backup database and database name, then we are doing offline backup. And that's why we are getting there error messages saying 'The database is currently in use', so that's why I have to force all connection so that the database is offline. And then I can take the backup.

Now you can also do a **backup online** and for that purpose, you'll have to add the clause online here in your backup database statement, and online means that you are basically doing operations on the database while you are taking the backup. So there may be different people connected to the database and they're doing operations on the database while you are taking the backup. You may say, ok, if I am taking the backup while there are connections and other people are making changes to the database, which means my backup image is not accurate. And, that's true. Your database image would not be accurate, but with the backup database command, there is a clause, that is called '**include logs**', so you could include the logs at the time you are taking this backup. And with this, you have the image and the logs together, which mean that if you need by any chance to do a restore using this backup, you can apply the corresponding logs and then you will have an accurate situation or an accurate image of this database at the time. So you can include the logs... when you are using an online backup.

Anyway, going back to the demo that I was doing here, I issue the backup statement and as you can see the backup was successful and it gives me a timestamp. Now if I change to the "backups" directory and issue a **dir**, I will see here the file. So this is the name of the backup.

And it also shows me the size of this file and sometimes it's a good way to, or a quick way, to take a look at the size of the database itself. So, if you want to take a look or if somebody tells you, 'hey how big is your database?', then you can take a backup (and look at the backup), and then you can estimate the size based on the size of this backup. Also you can do it from the Control Center. From the Control Center, you can see the size of a backup.

Switch to Control Center

Let me see if we can see it right now. See, from the Control Center here, I have this SAMPLE database selected. And from here, you can also see the size of the backup, approximately 126 megabytes. Anyway...

Switch back to cmd window

Now, the syntax for the backup command needs to be shown here. So there is a name of the backup, the type of the backup. Zero means it's a full backup, because you can also take backups at the table space level. Just for a table space, there are different values here you can specify. DB2, the name of the instance, then these ones; NODE would be a fixed number and CATN would be a fixed number, because we are not working with database partition feature which is on other editions of DB2, where you can have different partitions for your database. So this would be fixed for DB2 Express-C. And here you have the timestamp, all of this way up to here. And the timestamp is a way to uniquely create a filename for this image. You cannot really change, or you can't change the name of the backup, you can change it, but then you have to put the name back, the original name, if you want to restore.

Switch back to cmd ppt

So, going back to the slides, the backup was successful.

Slide 17: Example of backup file names

And I already showed you the name or the convention that is used to name a backup file. And this, on Linux, UNIX, and Windows, is the same name.

Slide 18: QuickLab #10 – Scheduling a Backup

And now you can pause and work on Quicklab #10 to schedule a backup, using the Control Center.

Slide 19: Agenda

So, moving on in the agenda, we are now going to talk about recovery.

Slide 20: Database recovery

For recovery: recovery is basically, you took a backup or you have your logs, and now you want to recover from that backup or reading the logs. This happens normally when there is a problem that happens, maybe there was a crash in your system or maybe there was incorrect information.

Slide 21: Recovery types

So there are different types of recovery. The first one is a crash or restart recovery. So what this means is, let's say, you are working on a desktop, and you are working with your database and then all of a sudden there is a blackout, so you run out of, you lose your electricity, or maybe somebody trips on the cord that was powering your desktop, so then your computer powers off or crashes because of this. And then what you do next is you plug in the plug again. You start your computer again. And then you will start the DB2 instance

again or DB2 will start again. What's going to happen is that DB2 automatically will run a command called RESTART DATABASE. Restart database means that ...or what DB2 will do at that time it will read the active logs at the time, and whatever information was committed, it will redo. Whatever it was not committed, it will undo.

So basically it will make sure (a restart database will make sure) that whatever you had saved and committed in the logs will be there for you. And whatever was not committed, meaning that actually you didn't explicitly save this information, and then it would not be that part that would be lost. But at least DB2 will bring back whatever you explicitly saved. So it's like again in the Word document, and you are working on a Word document. Whenever you click on Save, that information will be guaranteed to be saved and you can recover. But whatever you didn't save, that will not be recovered. So that's the same thing we are doing here with the crash recovery. Now for a version or image recovery, what that means is, we are running, let's say, you took a backup, and now you want to, for some reason, (you want to) restore from that backup. So you took a backup; it's like taking a picture of a database and now you restore from that image to that point in time.

And then finally, we have a 'Rollforward' recovery which is basically you took a backup and then you are going to restore from that backup but as well; you are going to 'rollforward' the logs to the latest point in time. And a rollforward recovery is normally used in production and it is normally used to recover to the closest point in time. So for example, going back to the example of human error, let's say, your user was creating an application where he had to increase the salary of all employees in the company by 10 percent, but he made a mistake. Instead of putting 10 percent, he put 100 percent. So nobody knew about this problem. And you know the application starts working; started changing this information in the database and many of the processors were using this information. And what's going to happen is you have your database... working perfectly, but the information in the database is not correct... Normally, once... everybody got their paycheck (and then everybody got a big surprise), then the DBA noticed about this problem and at that point, the database administrator will have to call this person who created this application. And then he will have to figure a way to restore the information that was available before the application was run. And then have to run, again, the application using that correct percentage for the increase on salary.

So the DBA will probably ask the application programmer, to say, when did you, when was this application scheduled? Based on that, he will probably have to find out the closest backup, so he will find a backup at the time and apply a restore on that; so we restore on that and then he would have to rollforward the logs up to a given point that is consistent. And so again, we come (again) to the topic about having archival logs and using archival logging. Because if you don't have those logs available, because, let's say, if you were circular logging, those logs are overwritten, then that's it, you cannot really rollforward; you can just do the backup and that's it. But if you had these other logs archived, then you can rollforward and get closer to the point where the human error occurred. And then once you rollforward to the closest point you can, then you can start running the application again using the right increase percentage for your salary, and then the database will be consistent again with correct information.

Slide 22: Database restore

So those are the types of recovery. And now if you want to run a restore, you can run a restore using the syntax that I provide here. Again this is a simple syntax. You can run a **DB2 ?** with the restore from the db2 command window as I did with backup to take a look at

more information on restore. Now you can say restore database (database name), from the path, this is optional. Whatever is in square brackets is optional. That's the syntax that we use. And 'taken at' is also optional. But 'taken at' would be good to put when you have several images on the same directory, so that's the way you would identify which of the several images is the one that you want. And then, well, this is what I already showed you, this is the syntax for the backup image. And you will be restored from there.

So let's do the example where I first drop the database sample, but in the previous exercise we back up the database sample, so we backed it up. We can drop it now and we restore. There are other utilities that allow you to check that database is not, the database image of the backup, the backup image is in good state as well. Well that may be a good thing to do as well in a production environment before you issue a drop. Or, in general, it may be good to do it before you issue a drop. But in our case, we are just going to go ahead and do a drop.

Switch to cmd window

So we go **db2 drop db sample**. And again we are already connected here. So let me **force applications all**.

Switch to Control Center

I think that happens because I was using the Control Center, and I checked for this.

Switch to cmd window

So let me close this Control Center as well. Let me now drop database SAMPLE. And the database SAMPLE is successfully dropped. Now so let me just connect. If I try to connect to sample, I should get an error: **db2 connect to sample**. Database sample is no longer there. Now I am going to run a restore, so I can do this: **db2 restore db sample from C:\backups**. And now in this directory, there was only one image, so if I run it like this; it should work, but just for the purposes of this demo, I am going to put the timestamp as well. So I am going to say **taken at**, put the timestamp which finishes all the way here. So I need to add that. So that's a whole timestamp, you don't need the zero-zero-one, that's milliseconds, that's not required. And then I press Enter to restore the SAMPLE database. So it may take a few seconds or minutes depending on the size of the database, but while it's restoring...

Switch back to ppt

Let's see what's the next on the PowerPoint presentation...

Slide 23: Also possible with BACKUP and RESTORE

Let's see: so these are older things that you can do with backup and restore. You can restore over an existing database. You can clone a backup image. So that, let's say, you take a backup of your database in Toronto and your computer had 2 disks, where the backup information was stored. Now, you take that backup image to China, and now you want to restore this database in this computer in China, but the computer in China only has one disk. So if you are going to restore the disk image, then it would complain because ... where is it going to put it? It cannot put the same layout or use the same layout because you only have one disk. So in this case, you can use our **redirector restore**, and basically with the redirector restore you would be prompted... where you want to store given table space. You are prompted on a table space where should I put it and you fill in the information. Now with DB2 Version 9, there is a script that can be created for you so that it reduces the amount of manual work for this redirector restore.

You can also do backups on table spaces. You don't need to backup the entire database. You can restore also only table spaces for a full backup image. You don't need to restore the entire database. You can do delta and incremental backups. You can do backup from the flash copy if the hardware supports it. And you can also recover drop tables if you turn on the option to save some information about drop tables.

Switch to the cmd window

So let's go back to the command window. And we can see now that the **restore database** completed successfully. And I can try to now connect to the SAMPLE database... and I should be able to connect to the sample database as you can see. So the backup worked. The restore worked as well.

Switch to the ppt

Slide 24: What's Next?

So with this, we have reached the end of Lesson 9, Backup and Recovery. So, congratulations for completing this lesson. And I would suggest now to continue with Lesson 10, which is Maintenance and that would be the last topic from the DBA part session of e-learning courses. Thank you and have a good day.