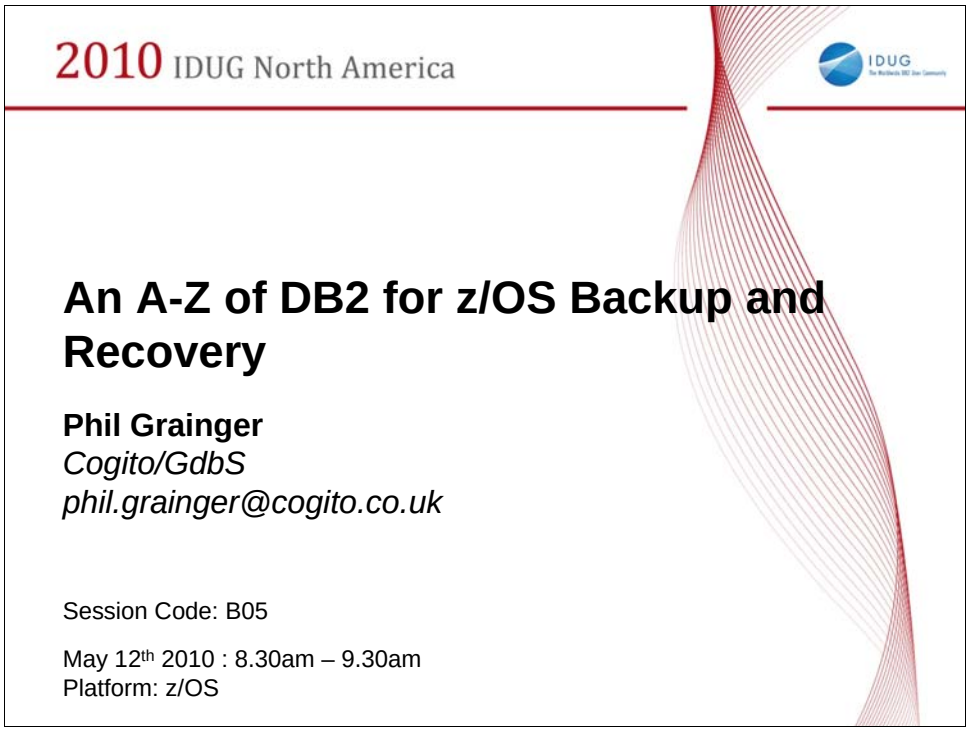
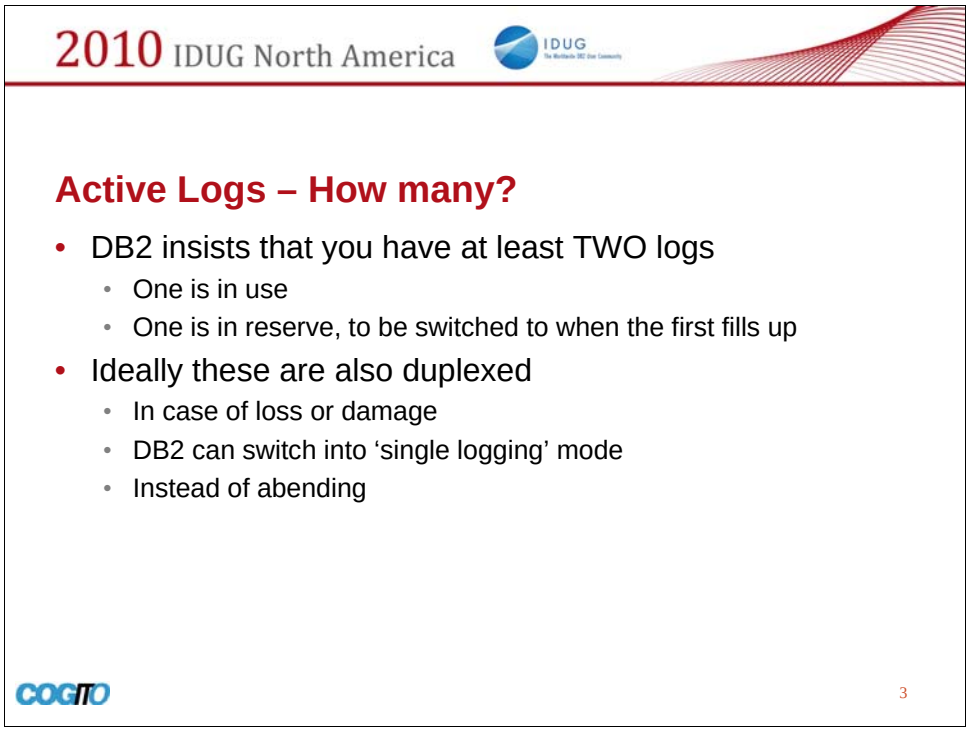


There are lots (and lots!) of things connected with DB2 for z/OS backup and recovery – we'll look at many of them in this presentation, in alphabetical order

As well as the technical nuts and bolts (active and archive logs, hardware services, image copies and system backup & restore) we will also spend time considering the less technical, but equally important management considerations (business & mandated requirements, strategies etc.)

There are lots (and lots!) of things connected with DB2 for z/OS backup and recovery – we'll look at many of them in this presentation, in alphabetical order

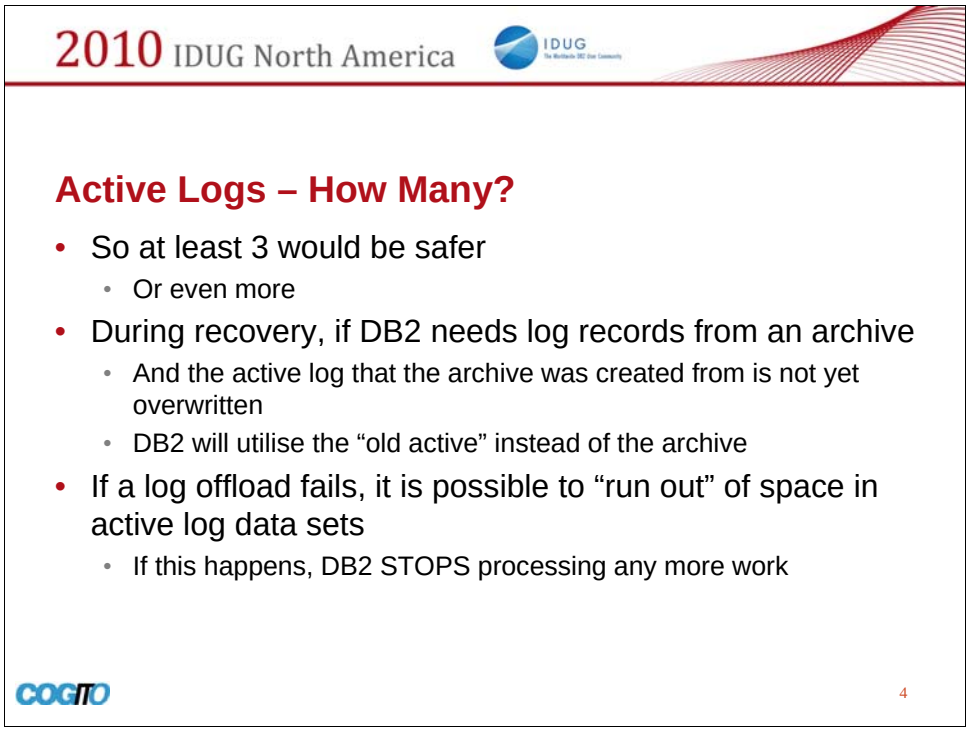
As well as the technical nuts and bolts (active and archive logs, hardware services, image copies and system backup & restore) we will also spend time considering the less technical, but equally important management considerations (business & mandated requirements, strategies etc.)



The active logs are a key part of normal DB2 operation. EVERY change (with only a few "log no" exceptions) will be recorded in the DB2 active log.

DB2 insists on AT LEAST two active logs – one in use and one ready to take over when the first fills up. It is good practice (see later) to have more than two though

Ideally the active logs should also be duplexed. If DB2 cannot write to any active logs, it WILL abend. Duplexing removes a single point of failure

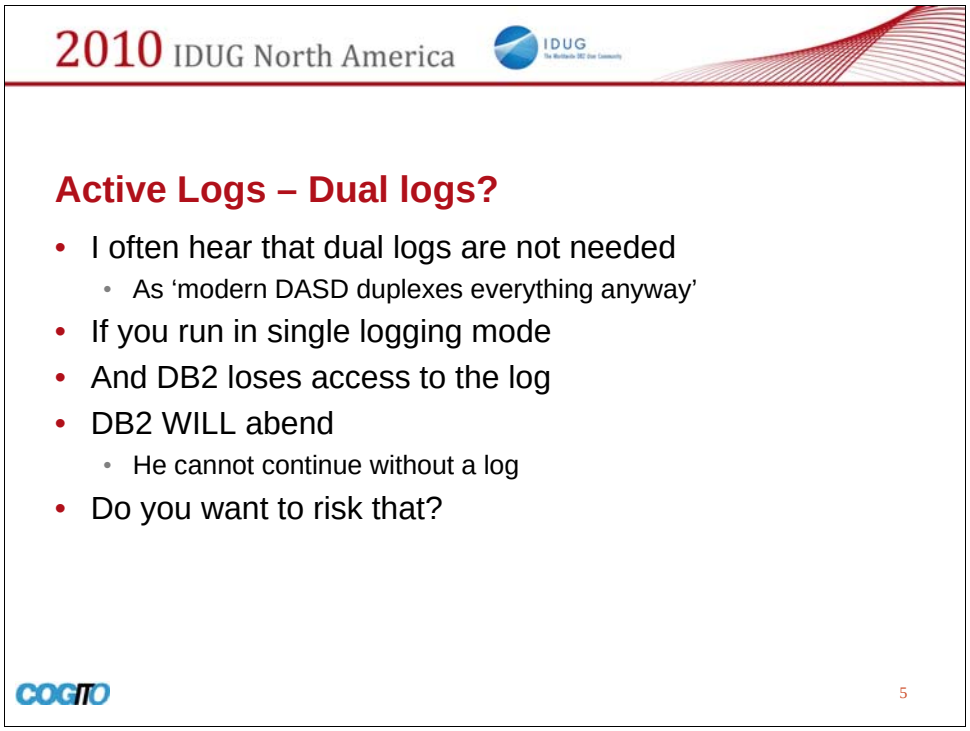


The problem with only having two active logs, is what happens if one fills and then is unable to be offloaded. This means that DB2 cannot switch BACK to this log when the second one fills up

When that does happen, DB2 issues message "DSNJ111E OUT OF SPACE IN ACTIVE LOG DATA SETS" and stops processing more work. The only way out of this impasse is to either force an archive of a full active log or to forcibly terminate DB2 by cancelling the IRLM. You cannot perform a normal DB2 shutdown as there is nowhere for DB2 to write the shutdown checkpoints!

If you feel you may be at risk of this, keep a VERY careful eye out for "DSNJ110E LAST COPYn ACTIVE LOG DATA SET IS nnn PERCENT FULL" – this is your alert that you ARE running out of active log space!

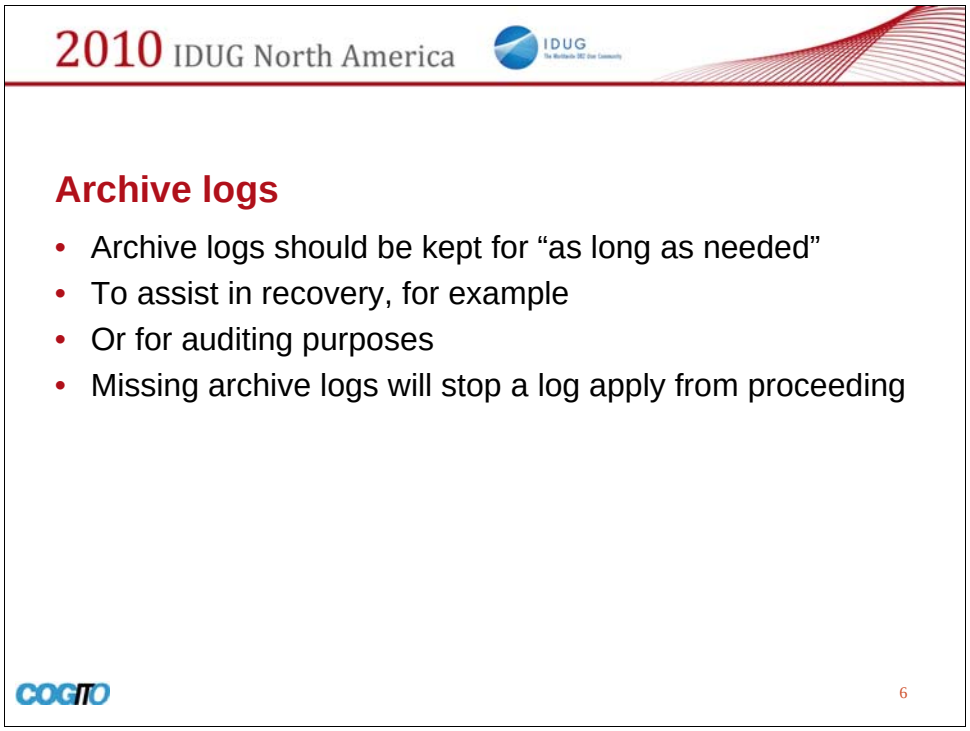
Also, if you have more active log datasets, it is possible for DB2 recovery to use them to recover data – the log apply processes know whether an old active log dataset still contains data needed, and processing a DASD active log will be faster than a (possibly) tape mounted archive



But do you really need to keep dual archive logs when todays DASD is fully "fault tolerant" and probably duplexes everything anyway (not to mention the provision of RAID technology to recreate lost data)

Well, DB2 doesn't KNOW that you have duplexed it's single log so a failure to write to that log WILL cause a DB2 abend, regardless of what is happening in your DASD

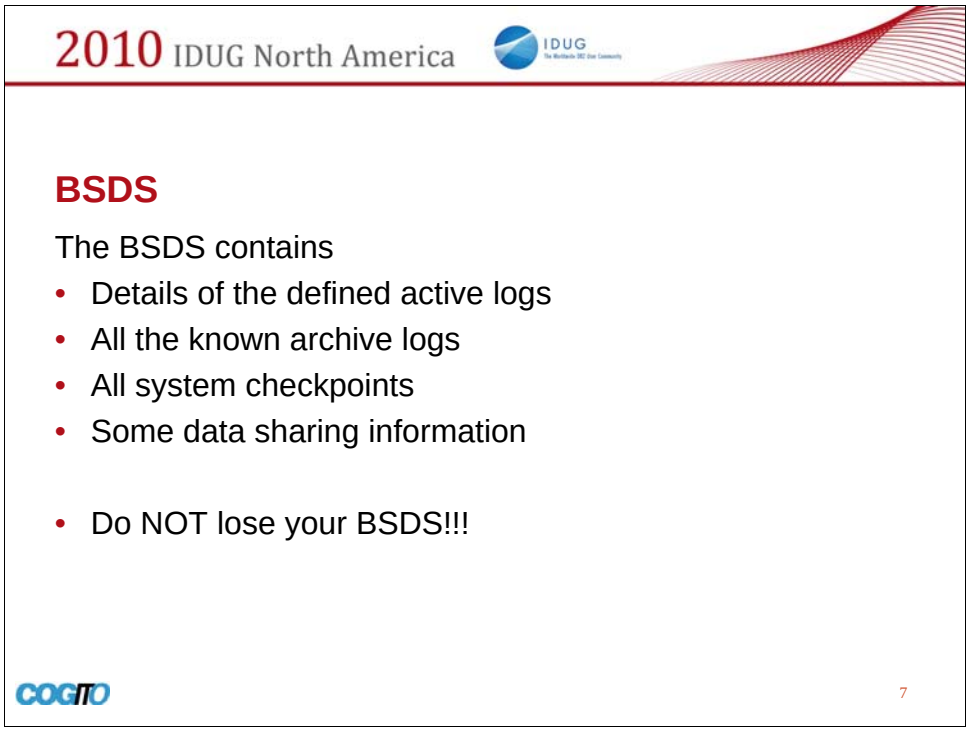
Do you want to risk that?



There really isn’t such a situation as “having too many archive logs” but there certainly is a problem with “not having enough”

Mane sure that you have ALL the archive logs you need for ANY recovery eventuality – including where you may need to fall back from one recovery position to an earlier one and roll forward through the logs

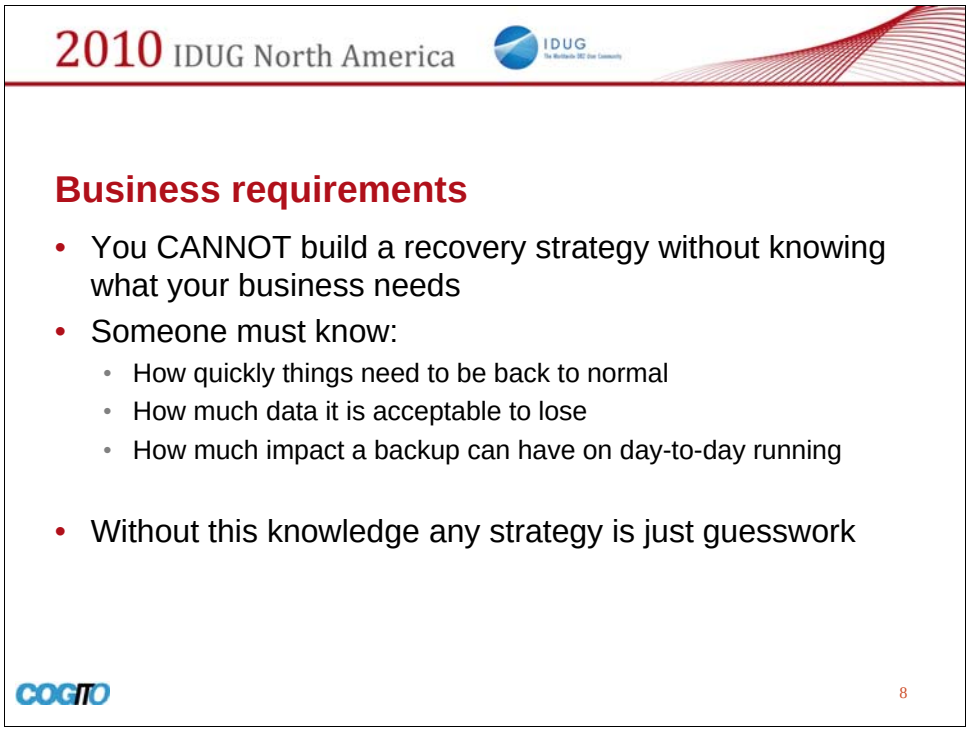
Auditors sometimes also have a knack of asking “What did this data look like 6 months ago?”…..



The Boot Trap Data Set (BSDS) is THE single most important part of your DB2 subsystem. It not only contains information about all of the active AND archive logs, it also contains the current highest log RBA, as well as lots of information about data sharing groups (not just the current LRSN)

If you lose your BSDSs for some reason then you HAVE lost your DB2 subsystem until you can recreate and recover them

Also, DB2 insists that you have TWO BSDSs and they MUST be the same AT ALL TIMES – if DB2 detects a discrepancy in the BSDSs at startup, it will not start until the problem is resolved



Far too many DBAs rush off and build a recovery strategy without considering what the BUSINESS needs

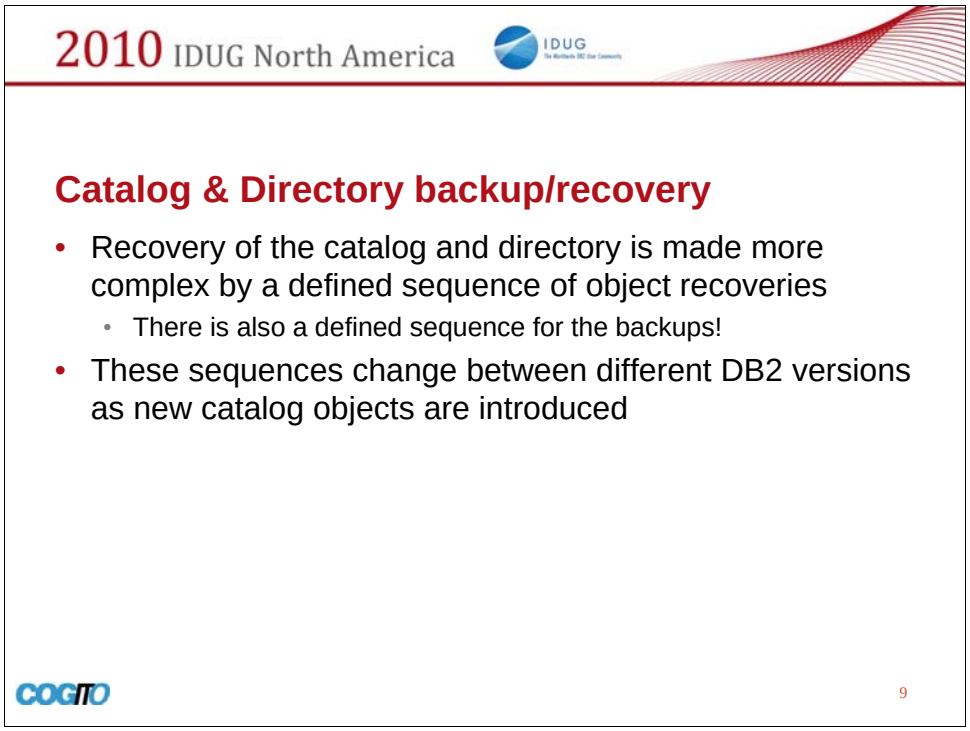
What results is, usually, a strategy that treats all DB2 user objects equally and probably recovers them in alphabetical order

As far as a backup strategy, that sometimes becomes nothing more complicated (or more useful) than "how much can we back up in the time available?"

The business MUST be the drivers. Only they know how much data (if any) can be lost and how much time can be expended on a recovery. Only once those facts are known can backup and recovery strategies be developed.

Of course, it may be that the strategy to fulfil their requirements will not work with the competing availability needs of the application. Now it's time for a debate – either the availability needs must be relaxed or the backup/recovery ideals must be relaxed

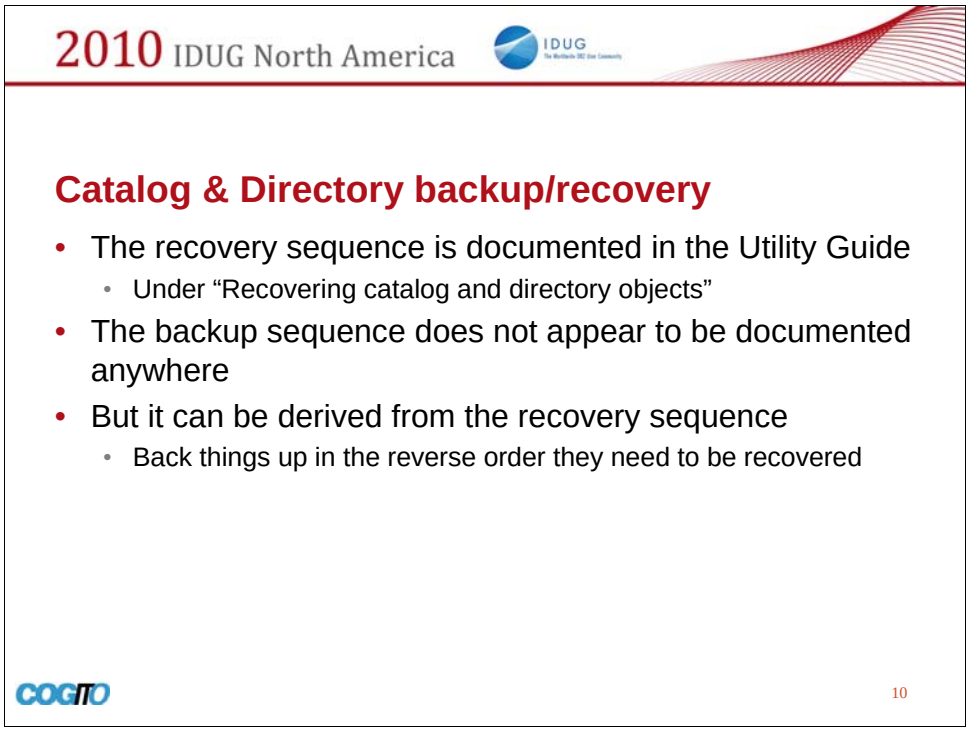
Again, these are NOT decisions for the DBA!



Most DBAs worst nightmare is a catalog/directory recovery

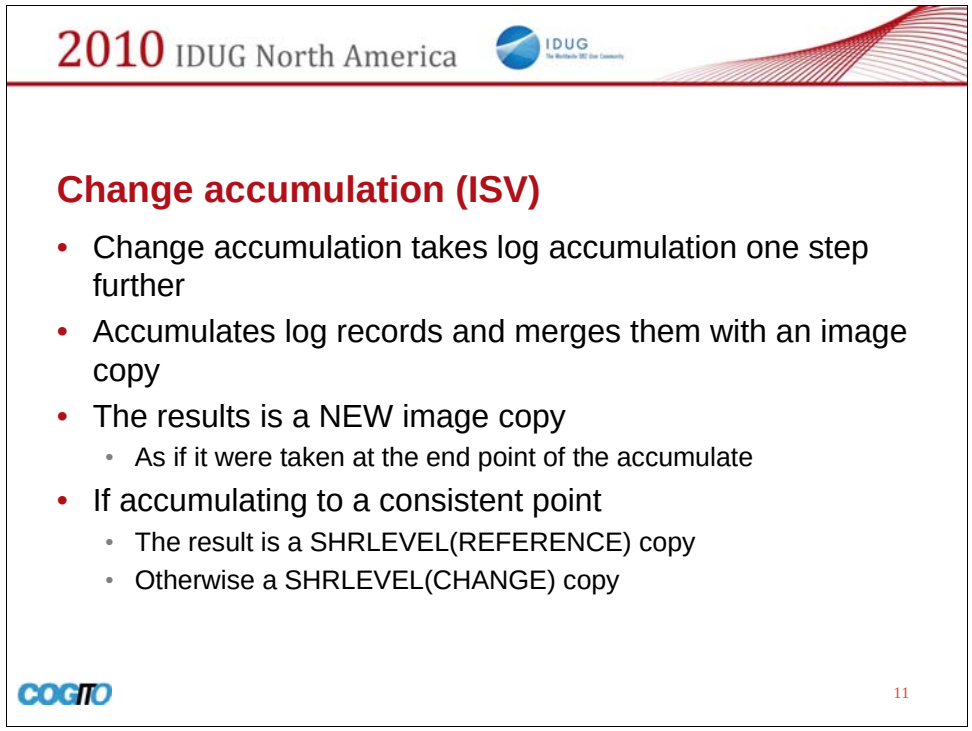
After all, user data is easy – you just code a RECOVER statement

The core of DB2 though requires much more thought and includes steps that you don't normally perform – not only that, but things can change with new releases of DB2 (and even with application of DB2 maintenance)



Step 1 would be “Read the documentation”

There is a defined sequence for catalog and directory recovery and you need to make sure that your backup sequence reflects this



Whilst DB2 only provides for the copying of an entire object (a FULL copy) or the pages changed since the last copy (an INCREMENTAL), independent software vendors (ISVs) also offer the ability to accumulate log records reflecting changes to objects and to merge those with existing image copies to make NEW copies

If the accumulation of changes to an object is performed to a point of consistency, this is equivalent to taking a full copy of that object with SHRLEVEL REFERENCE. But the big advantage is that no access is required to the actual object to create this new copy

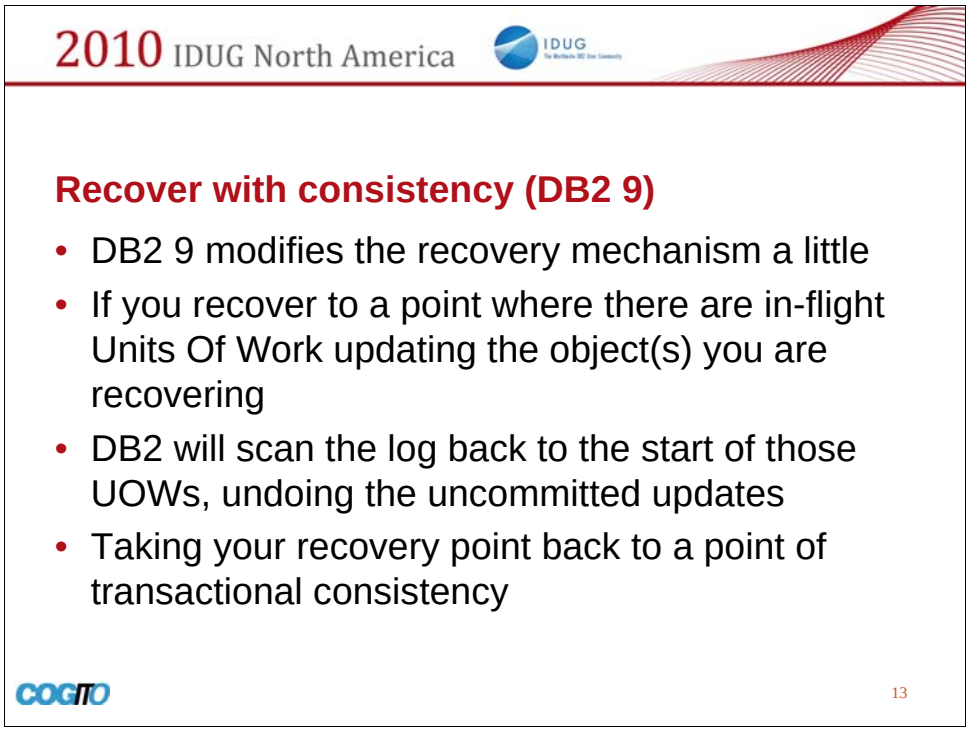
So, it is a non-intrusive way of creating new, consistent, copies

Recover with consistency (DB2 9)

- DB2 has always allowed you to recover to ANY RBA (or LRSN) you wish
- It is up to YOU to recover to a sensible (aka consistent) point
- In fact, DB2 (prior to DB2 9) will allow you to recover into the middle of a unit or work
- You could have uncommitted updates in your recovered object
  - It's what you asked for so you got it!

Before DB2 9, it was possible to recover to ANY log RBA you chose. If that RBA happened to be in the middle of an uncommitted unit of work, then DB2 would (as requested) recover your data back to that point

You might then be left thinking all was OK, whereas in fact you had recovered your data back to a point of Inconsistency



If you do the same thing with DB2 9, then DB2 will recognise that you have recovered into an in-flight unit of work, and figure out how to back OUT those changes that have not (yet) been committed

This means

1.You WILL have recovered your data back to a consistent point, but

2.You will NOT have recovered your data back to quite where you thought you had!

For these reasons, it is still much MUCH better to recover to a KNOWN point of consistency

Deferred restart

- If your subsystem availability is critical, the restart delay may be unacceptable
  - Even though things are much faster than they used to be
- In this case, the availability of the subsystem may be more important than the availability of individual tables

When DB2 is restarted after an abnormal failure there will normally be a delay whilst DB2 figures out what was happening at the time of the failure and works to undo any uncommitted changes and ensure that all committed changes are indeed propagated to the data pages

For a DB2 subsystem that has critical applications and data, it can sometimes be an issue to make the whole subsystem to wait for recovery of a relatively small amount of data

The restart of the subsystem may then be more important that the correction of all of the data

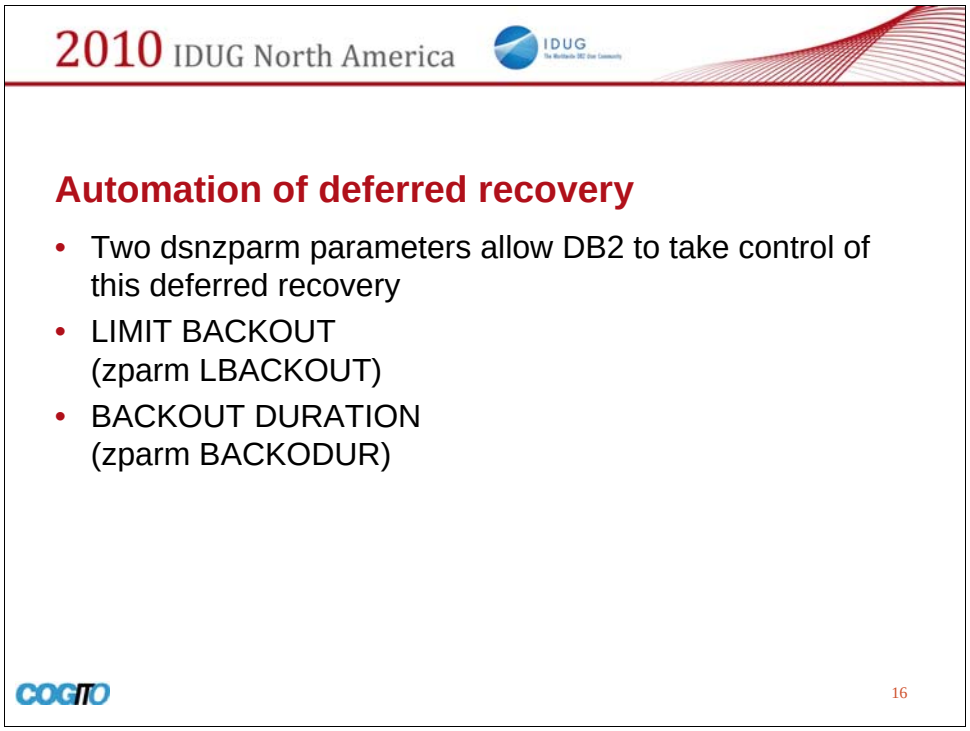
Deferred restart

- If so, dsnzparm can be changed to speficy DEFER,ALL
  - Instead of the more usual RESTART,ALL
- This means that during an abnormal restart
- DB2 will still look for in-flight UOWs
- BUT will not recover them
  - They are placed in LPL instead

If speedy restart IS the highest priority, then it is possible to change dsnzparm to specify DEFER ALL instead of RESTART ALL

This means no recovery will take place (apart from catalog and directory) during restart – they will be placed in recovery pending instead

If there are some high priority items, then the RESTART parm can be amended to list the page sets to be recovered during restart processing



Historically, these objects to be recovered were processed following a –START DATABASE command

Latterly though, this whole control is handed to DB2 – by two dsnzparm items

LBACKOUT and BACKODUR tell DB2 whether to continue to recover objects after a specified interval or whether to defer that recovery to after restart

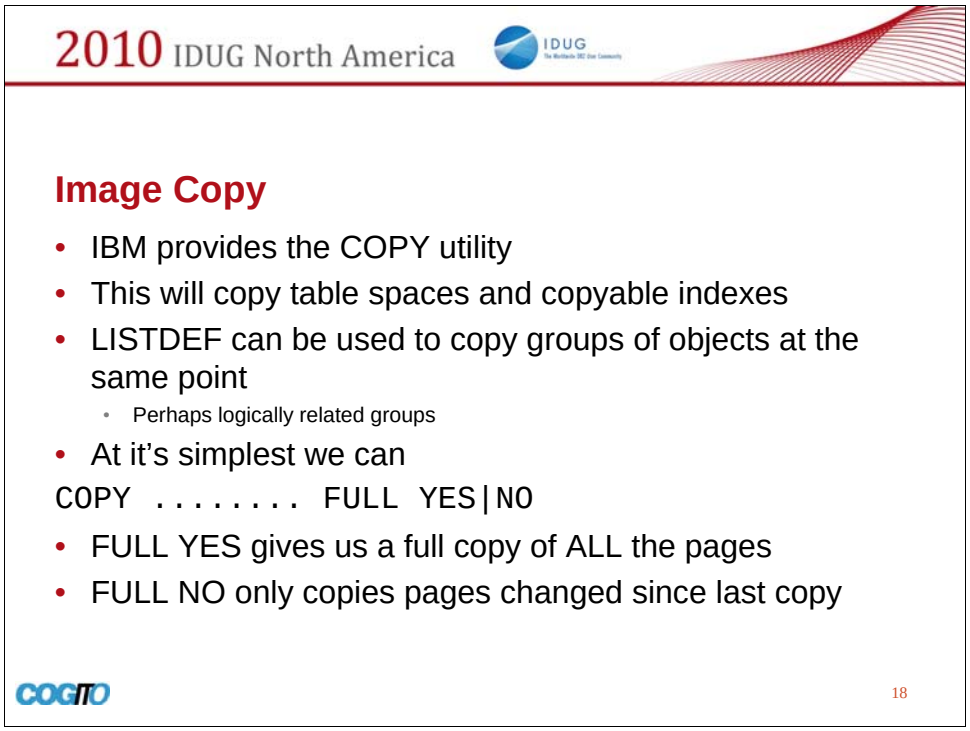
Hardware services

- Don't forget the possibilities for hardware assisted backups
  - And recoveries
- DB2 can utilise these for snapshot copies
  - And for BACKUP/RESTORE SYSTEM, of course
- ISVs also make use of these capabilities

Don't forget that backup and recovery is not just a DB2 issue – there may be things that you can do outside of DB2 to provide solutions to your businesses recovery requirements

DB2s new BACKUP and RESTORE SYSTEM utilities also use the hardware facilities to take rapid, effectively outage free, backups

ISVs may also provide support – check with them regularly as things do change



The Image Copy is really the basic building block of DB2 backup and recovery

You can copy TABLES or copyable INDEXES

Copies can be FULL (all pages are copied) or INCREMENTAL (only changed pages are copied)

COPY can be combined with LISTDEF to copy sets of objects with a single copy statement – TEMPLATE also allows target copy dataset names to be defined based on the contents of the LISTDEF

Incremental Copy

- In the past, incremental copies rapidly became slower than full copies
  - Because FULL YES could take advantage of prefetch
- Today, FULL NO can use LIST PREFETCH
- Benchmark to see where the breakpoint is
  - It can still be slower to incrementally copy a large proportion of a page set

Incremental copies, because they are copying a subset of pages, may not get quite the same benefits from sequential prefetch as full copies

Do your own benchmarks to see where the break point is – sometimes it may be quicker to take a FULL copy than an INCREMENTAL if a large proportion of pages have changed. It may also be possible to use CHANGELIMIT to automatically control whether a FULL or INCREMENTAL is taken

Incremental Copy - Changelimit

- You can also specify

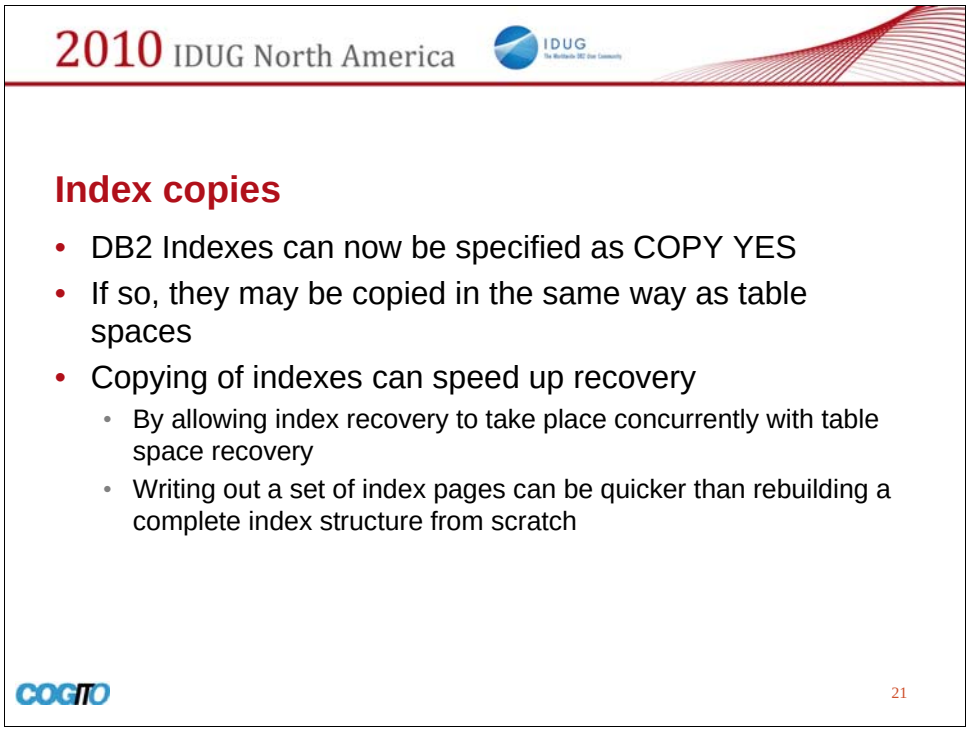
```
... CHANGELIMIT (percent1, percent2) ....
```

- If less than *percent1*% of the pages have changed
  NO copy will be taken
- If more than *percent2*% of the pages have changed
  A full copy will be taken
- Otherwise
  An incremental copy will be taken

With the CHANGELIMIT keyword, you can instruct DB2 what sort of copy you wish to be taken

- No copy
- Incremental copy
- Full copy

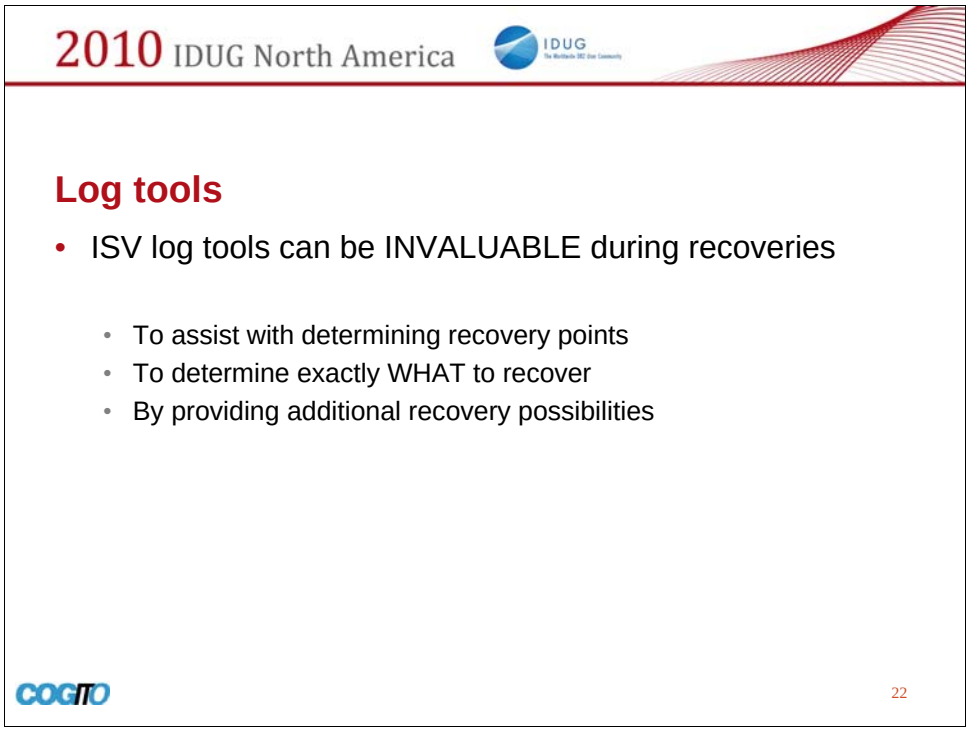
But be careful of relying on CHANGELIMIT for all copies – it could be possible that a fairly static object is NEVER copied! Perhaps a good idea to schedule a full copy periodically regardless of CHANGELIMIT



It is now possible to take copies of indexes, but ONLY if they have been defined as COPY YES

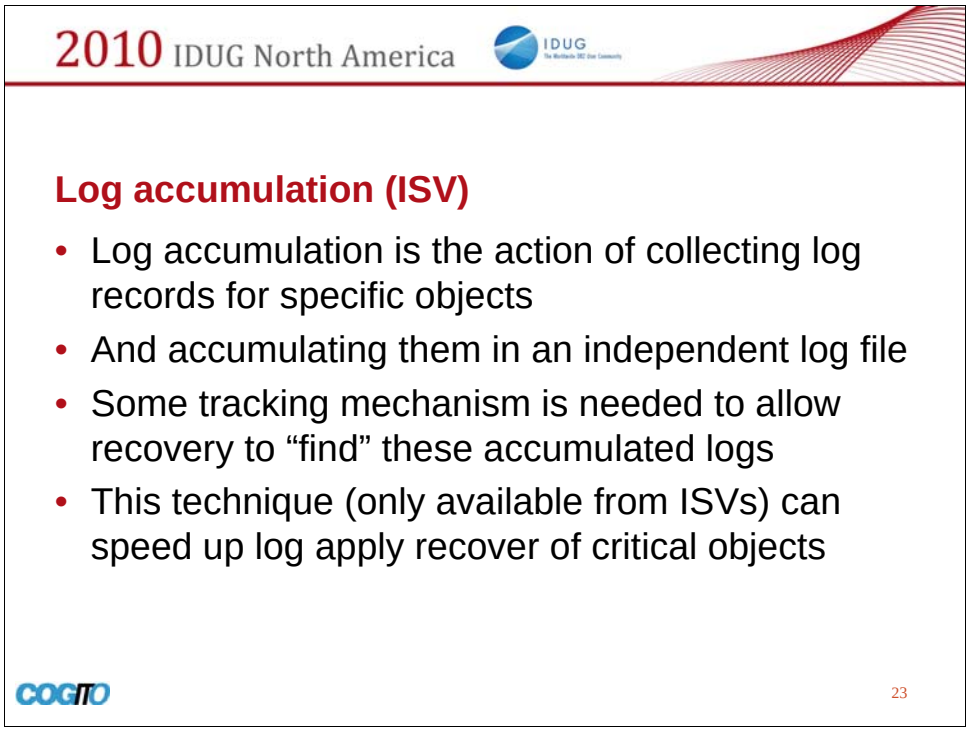
If not, they can only be REBUILT from the corresponding table space data

However, recovery of an index from an image copy can not only be quicker, but can be carried out in parallel with the table space recoveries



Don’t underestimate the value of the DB2 log when performing recoveries

Not only can it help to determine HOW MUCH data needs to be recovered (this is invaluable when recovering from a data corruption) as well as suggesting alternative recovery scenarios



Some ISVs also provide the ability to do LOG ACCUMULATION (see CHANGE ACCUMULATION earlier for something similar)

Log accumulation collects together log records for critical tables and saves them in a set of proprietary log files. These can then be used by the same ISVs recovery utility to speed recovery of those critical objects

It is generally quicker as the log files ONLY contain log data pertinent to the objects being recovered

Logging and Not Logging

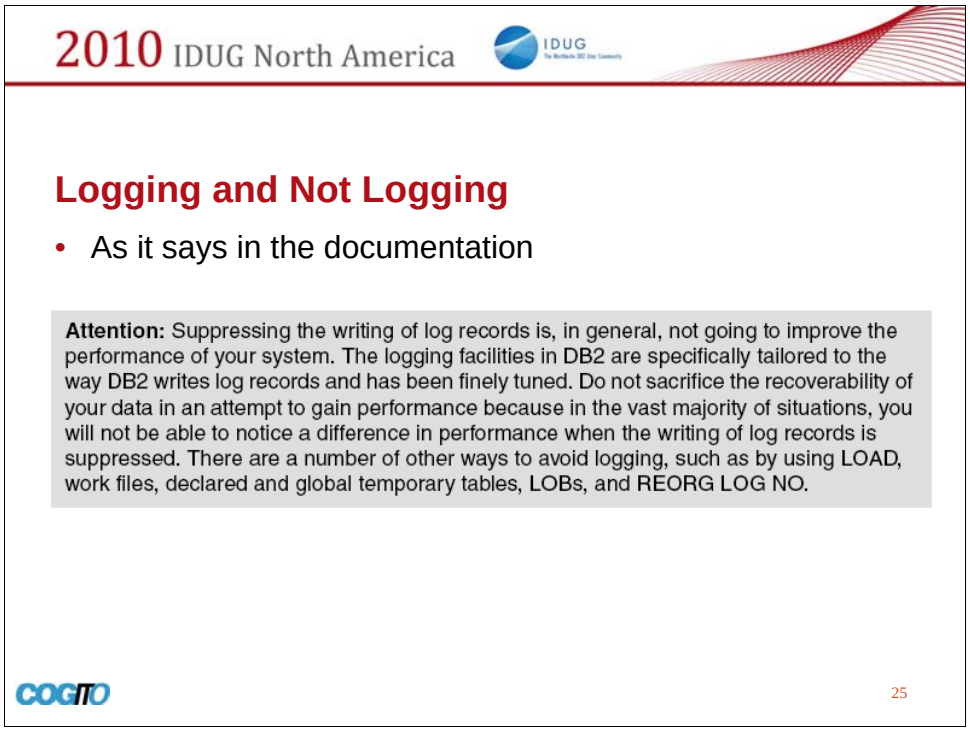
- DB2 already had NOT LOGGED for LOB table spaces
- DB2 9 allows this for XML and normal table spaces as well
  - And propagates this to their indexes
- Objects can be created NOT LOGGED
  - Or be altered to/from NOT LOGGED
- This is NOT a performance option
- It is dangerous

Logging of DB2 changes is no longer compulsory – you can turn it off for selected objects by setting NOT LOGGED

Note, logging for indexes is dependent on the logging set for their tables

PLEASE BE CAREFUL

This is NOT a “go faster” switch – it is to avoid logging when you don’t NEED the changes



Read this note carefully before going near NOT LOGGED objects!

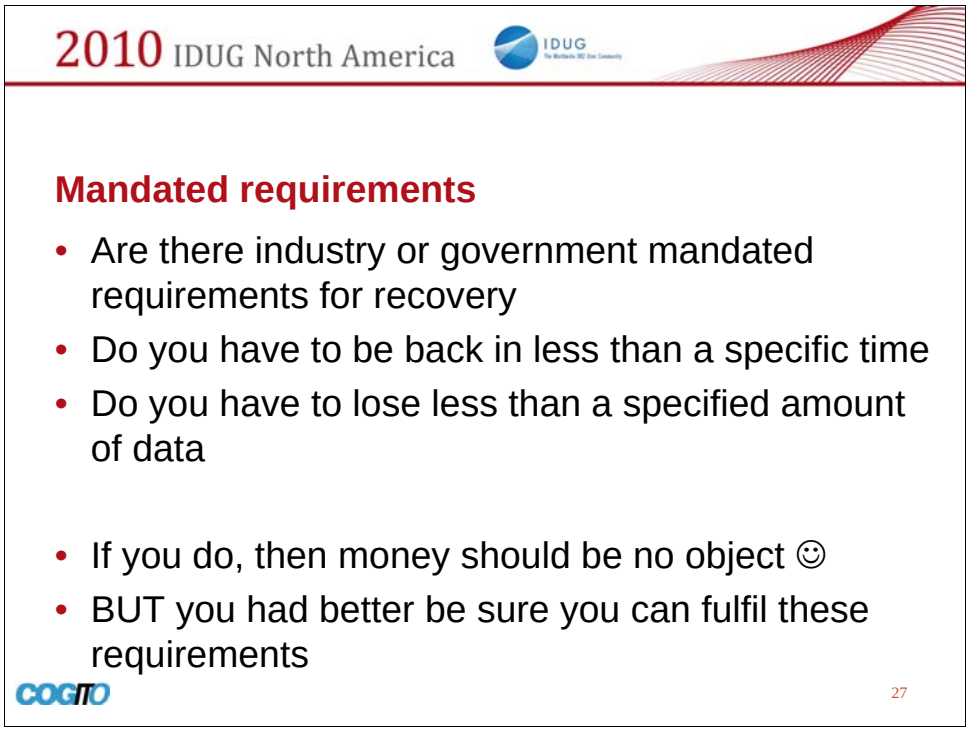
Logical/SQL recovery

- Similar to change backout
- It is possible to read log records to forward recover objects
- Normally, DB2 will do this
- But if you have a badly corrupted file system, an alternative may be needed
- Also, if you are recovering a dropped object
- Changes to internal ids (obid, psid etc.) may make DB2 log apply invalid

So-called “logical recovery” makes use of generated SQL (or equivalents) to perform a recovery action that is not otherwise supported

This can be either forward recovery OR backward recovery. This method makes it possible to “roll back” committed updates, ideal for correcting logic errors

This type of recovery can also extend to DDL – permitting recreation of dropped objects, but in this case the recovery is complicated by the fact that the internal IDs of the object(s) may have changed making traditional DB2 recovery of the data impossible



Are there any mandated requirements around data loss, service unavailability and the like?

If so, your business WILL have to find the money to provide for a robust solution

Having said that, you'd better be 100% sure you can fulfil your obligations

Incremental copy – MERGECOPY

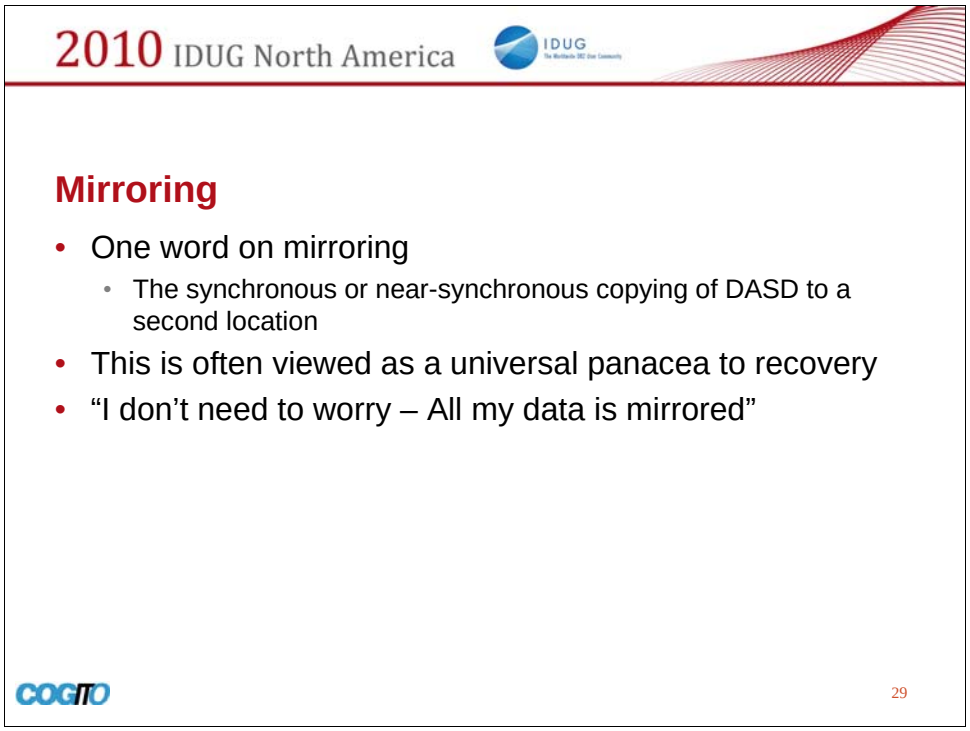
- The MERGECOPY utility can do two things

1. Merge a series of incremental copies to form one consolidated incremental copy
2. Merge incremental copies with a full copy to make a new full copy

- Either of these techniques will help to speed up recovery where multiple incremental copies are involved

If you are using incremental copies, you need to be aware that ALL necessary incrementals must be mounted AT THE SAME TIME during recovery

One way around this restriction, is to use the MERGECOPY utility to merge multiple incremental copies into a single one (or even to consolidate a full copy with subsequent incrementals to make a new full copy)



Mirroring is often suggested to be the "solution" to all data protection worries

Snag is, if you suffer a data corruption, the mirror will efficiently propagate your corruption

Mirroring

- But what about data corruptions?
- Many (most?) recoveries are done purely to recover from logically damaged data
- Mirroring just spreads the corruption across your duplicated DASD
- So mirroring only protects from PHYSICAL damage or loss
- And if you accidentally drop an object
  - The drop will also be mirrored!

So, mirroring only protects from PHYSICAL damage or loss, not LOGICAL damage (corruption) or loss (“oops – dropped the wrong table”)

Parallelism in copy

- Either code parallelism into the jobs that you create
  - And run multiple jobs simultaneously
- Or utilise the parallelism inherent in processing sets of objects
  - Either with IBMs LIST definitions
  - Or ISV wildcarding
- Make sure you understand how to control the degrees of parallelism though

Once you have identified what you need to copy and how often, the next challenge is usually to ensure that the copy runs in as short a time as possible

This normally means by the use of parallel processing

Utilise the parallelism inherent in your utilities (either IBM Listdefs or ISV wildcards)

Also, make sure that you understand how to control the degrees of parallelism and don't "overdo" it

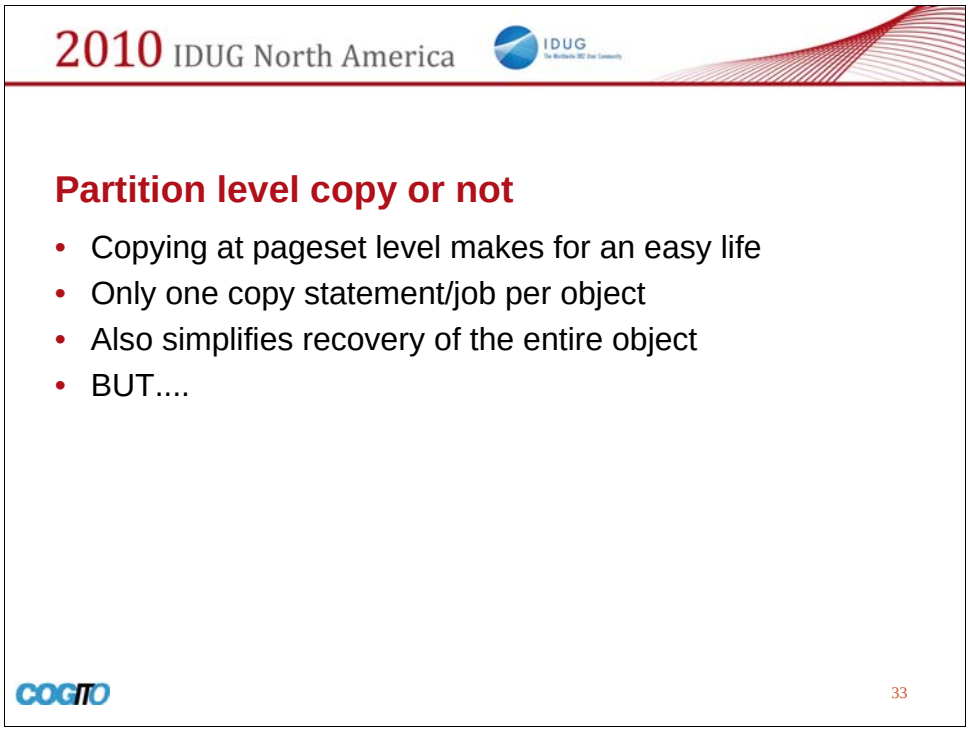
Parallelism in recovery

- Parallelism in recovery is perhaps even more important
- You are definitely now concerned about minimising elapsed time
- Parallelism can be:
  - Multiple partitions
  - Multiple page sets
  - Table spaces and Indexes
  - Any combination of the above

Again, during recover, possibly THE most important thing is to be done as fast as possible (OK, second most important – the MOST important thing is to recover the right data to the right point in time!)

You will have people standing over your shoulder and a business desperate to get back IN business

Make use of as much parallelism as you can



With partitioned objects, you can take copies at partition level OR at page set level

Whilst copying page sets makes for an easy life (one copy statement per table space for example), there is much flexibility to be gained from partition level copies

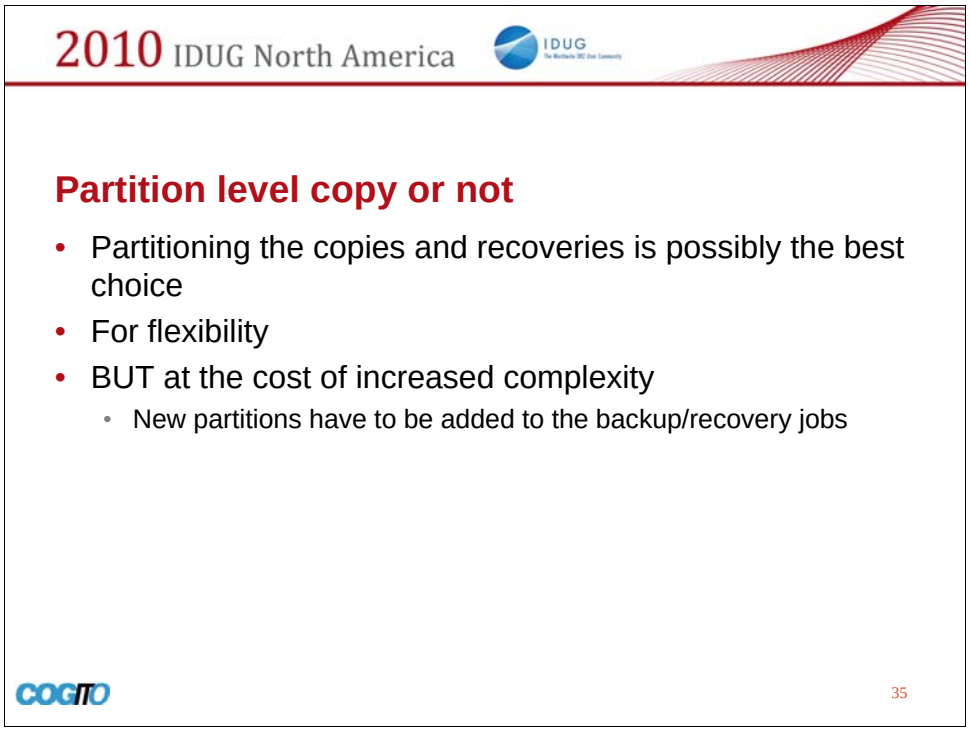
Partition level copy or not

- Copying at partition level increases flexibility
- ANY partition can be recovered independently
- Copies can also be run in parallel
  - Reducing the elapsed time of the copy
- Recoveries can also be run in parallel
  - Reducing the elapsed time of the recovery

You can choose whether to recover ALL partitions or just a sub set – with page set level copies, you cannot recover individual parts

You can recover partitions IN PARALLEL (see before)

You can copy partitions IN PARALLEL



Really, there should be no choice – COPY (AND RECOVER) AT PARTITON LEVEL

OK, it's marginally more complex, but the added flexibility makes this choice a winner

Priorities

- Most IT staff consider a recovery sequence of:
  - Catalog + Directory
  - ISV utilities
  - User data
- To be sufficient for the needs of the business

Going back to something we said earlier, when determining your recovery priorities, of course the DB2 catalog and directory is the most important, followed possibly by any tools that you will need to perform your application recovery

BUT

Priorities

- But this does not take into account the differing business requirements
- Some APPLICATIONS may be more critical than others
- Some applications may have critical OBJECTS
- Only the business knows these things
  - But IT knows how they interrelate
  - Business may require an object to be recovered as a high priority
    - But IT knows other objects are pre-requisites

Do you have multiple business applications in a single DB2 subsystem?

Do they all have the same recovery needs?

Are they all equally important?

Would you even know if one of them was critical??

Priorities

- These categorisations may also then drive revisions of the backup strategy
- Do these high priority objects require special treatment
  - More frequent full copies?
  - Change or Log accumulation processes?
  - Concurrent copies?
  - Mirroring?

As we said before, the business needs to provide input to determine whether some applications are more important than others

The high priority applications (or even individual objects) may need specific backup strategies as well

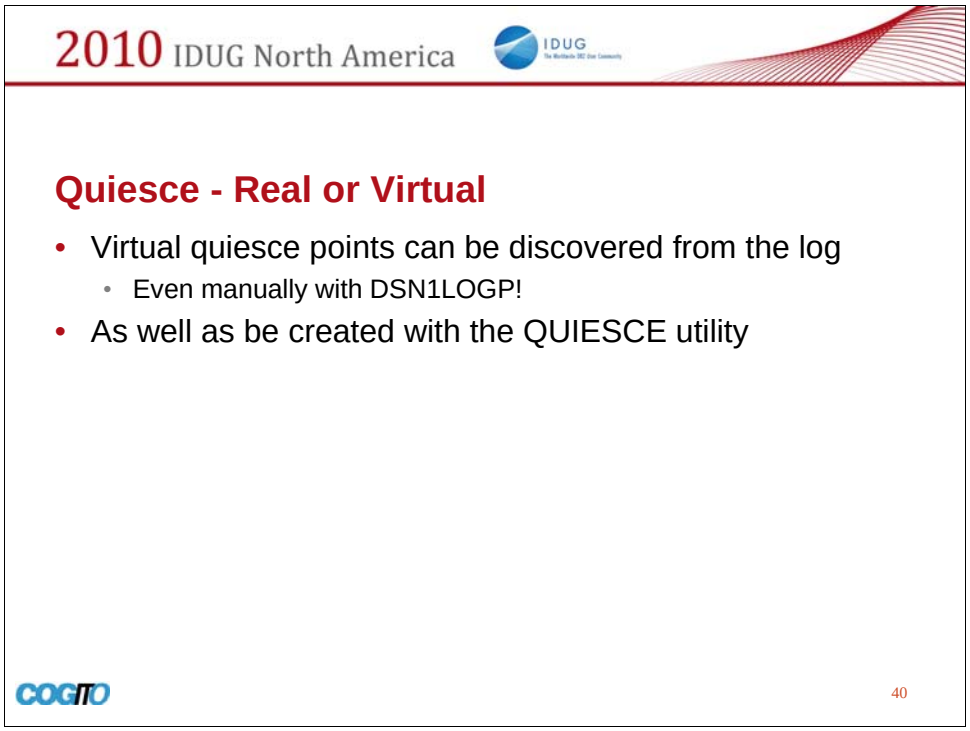
Quiesce

- A quiesce point is a point in the DB2 log where an object
  - Or more usually a set of objects
- Is not currently being updated by an in-flight unit of work
- In other words
- Is a point of CONSISTENCY to which objects may be recovered
- The quiesce utility is normally used to determine and register quiesce points

A quiesce point is a point in the log where there is no update activity in progress for an object or a set of objects

A quiesce point is thus a point of consistency for recovery purposes

Quiesce points are determined and registered in SYSCOPY by use of the QUIESCE utility



However, it is also possible to determine quiesce points “after the fact” by examining the DB2 log

Either manually with DSN1LOGP (painfully slow and not easy) or using a log analysis tool (from IBM or ISVs)

I’ve taken to calling this discovered quiesce points “virtual quiesce” as they are points in the log where you WOULD have been able to take a quiesce if you had tried

DB2 Restart

- If the prior shut down was NOT normal
  - i.e. there is no shut down check point
- DB2 knows that something is wrong
- Now the log must be scanned back to the most recent checkpoint
- This shows which UOWs were in flight
- Now each and every one of these must be undone before restart can be completed

During restart, DB2 will know whether the shutdown was an orderly one or an abnormal one (there will either be a shutdown checkpoint or not)

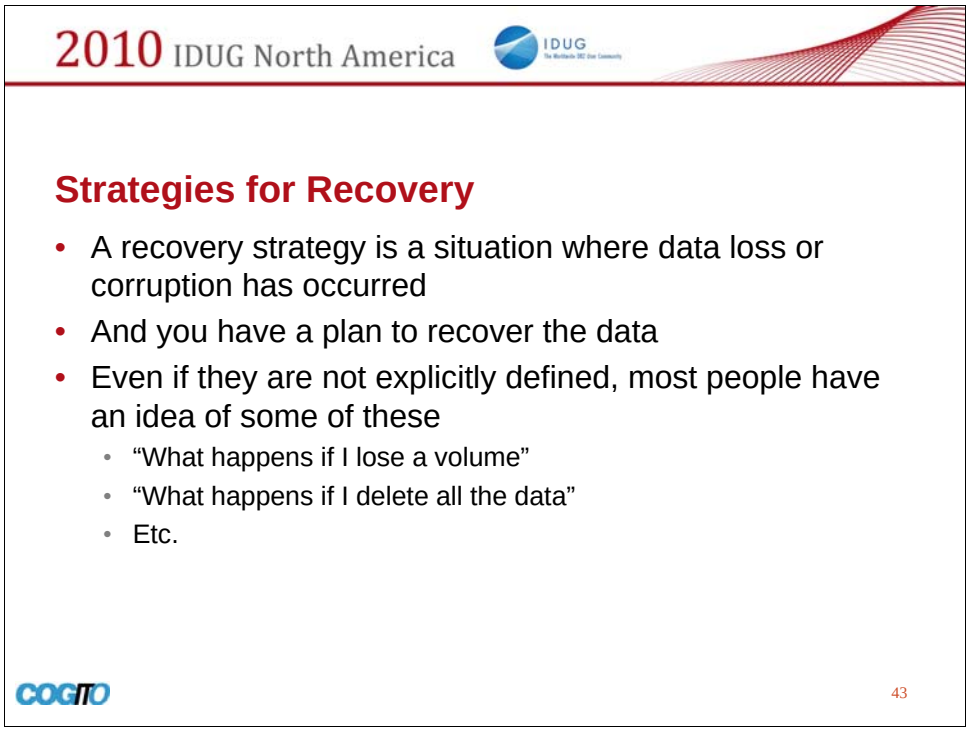
If the prior shutdown was abnormal, during restart processing DB2 has to figure out what was going on at the time of failure, which unitos of work were in flight and hence had uncommitted changes pending and determining whether these must be committed or rolled back

DB2 Restart

- As DB2 is processing these, compensation log records are also being created
  - In case DB2 fails during the restart
  - There has to be some way of picking up the pieces and starting again
- Once all of the in-flights have been recovered, the DB2 subsystem is now available

However – what happens if DB2 abends again before this abnormal restart is complete?

DB2 has to have some way of determining how far through the recovery process it had got last time – so during any abnormal restart, DB2 will write out MORE log records (compensation log records) leaving an “audit trail” of the in-flight recovery process



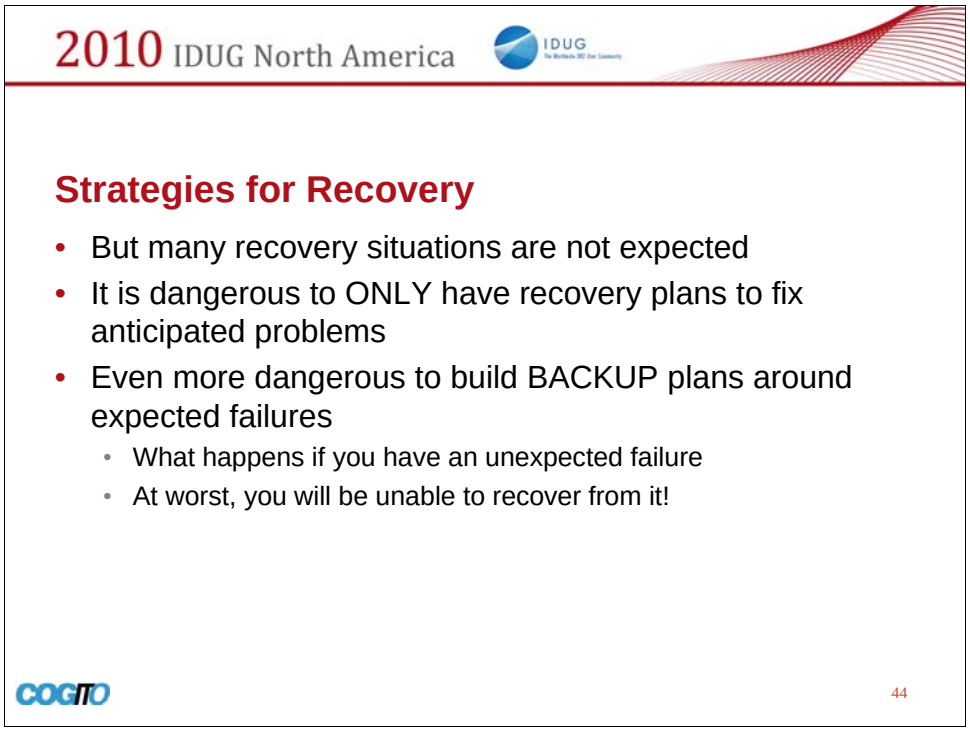
We've already talked enough about priorities and we have touched on the subject of strategies, but just what IS a recovery strategy

Well, I suppose there are two types

1.Recovery from an expected event

2.Recovery from an unexpected event

The former are relatively easy to plan for – after all, you will know what has happened and what you have lost

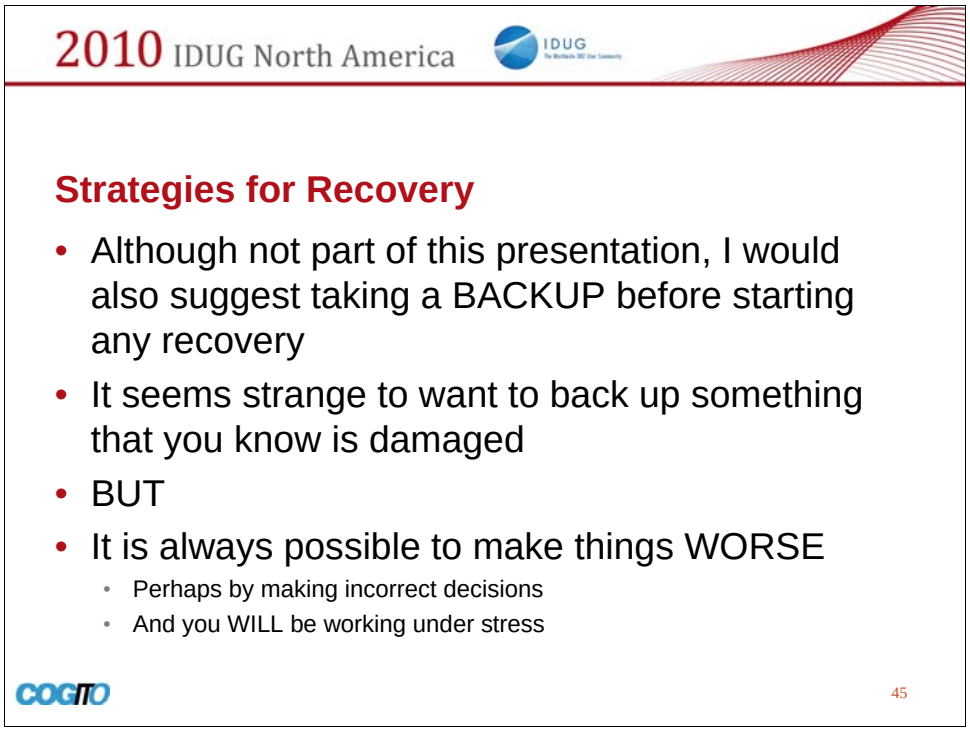


The snag comes with unexpected events

Because they are unexpected you cannot plan a strategy to recover from them

The only safe strategy you can create is to plan for the COMPLETE loss of EVERYTHING – hopefully then your loss will be a subset of that, and you can use a subset of your strategy

The one situation you do not want to be faced with is trying to create a new recovery strategy “on the fly” after you have had your data loss



There is one thing I would highly recommend though

If you have had some sort of disastrous data los, PLEASE before you start ANY recovery activity take some sort of backup of your current situation

This might seem an odd thing to do, but believe me – in the heat of the moment, it IS possible to make things even worse than they are already

If you make an incorrect decision and dig yourself even deeper into the hole you are trying to get yourself out of

With a “before you start” backup, at least you can get back to where you started and try again

Otherwise, now you are faced with building a strategy to get yourself out of your self inflicted mess before you can turn your attention to the real recovery again

SYSCOPY housekeeping

- SYSCOPY can quickly grow to an unmanageable size
- The MODIFY RECOVERY utility is used to keep things under control
- MODIFY RECOVERY will remove entries from SYSCOPY based on specific criteria

As you will know, all image copies and quiesce points (and quite a lot of other utility operations) are all logged in the SYSCOPY catalog table

The MODIFY RECOVERY utility is used to clean out unwanted entries from SYSCOPY

SYSCOPY housekeeping

- DELETE .... AGE() or DATE()
- Will delete rows that are older than the specified number of days or specific date will be deleted

Records can either be deleted by DATE or by AGE

DATE removes records created prior to a certain date

AGE removed records older than a certain number of days

SYSCOPY housekeeping

- RETAIN .... LAST(), LOGLIMIT or GDGLIMIT
- Retains the
  - Last N entries in SYCOPY
  - All entries that are more recent than the oldest archive log timestamp
  - All entries that are more generations behind than the GDG holding the copied object will support

Useful recent additions to MODIFY RECOVERY allow the records kept in SYSCOPY to be synchronised with any GDGs that are used to actually create the copies

Either retain the LAST(n) entries

Or retain as many entries as needed to match the specific GDGLIMIT

System level point-in-time recovery

- DB2 Version 8 introduced BACKUP and RESTORE SYSTEM utilities
- To support backing up (and restoring) a full DB2 subsystem
  - Either data only or data and catalog/system data

To better assist with the planning (and execution) of system level recoveries, IBM introduced the BACKUP and RESTORE SYSTEM utilities

These use DB2 logic, together with dfsms to take minimally disruptive copies of entire DB2 subsystems so that they can later be restored to a specific point in time

Changes at DB2 9 enabled recovery TO CURRENT with RESTORE SYSTEM as well

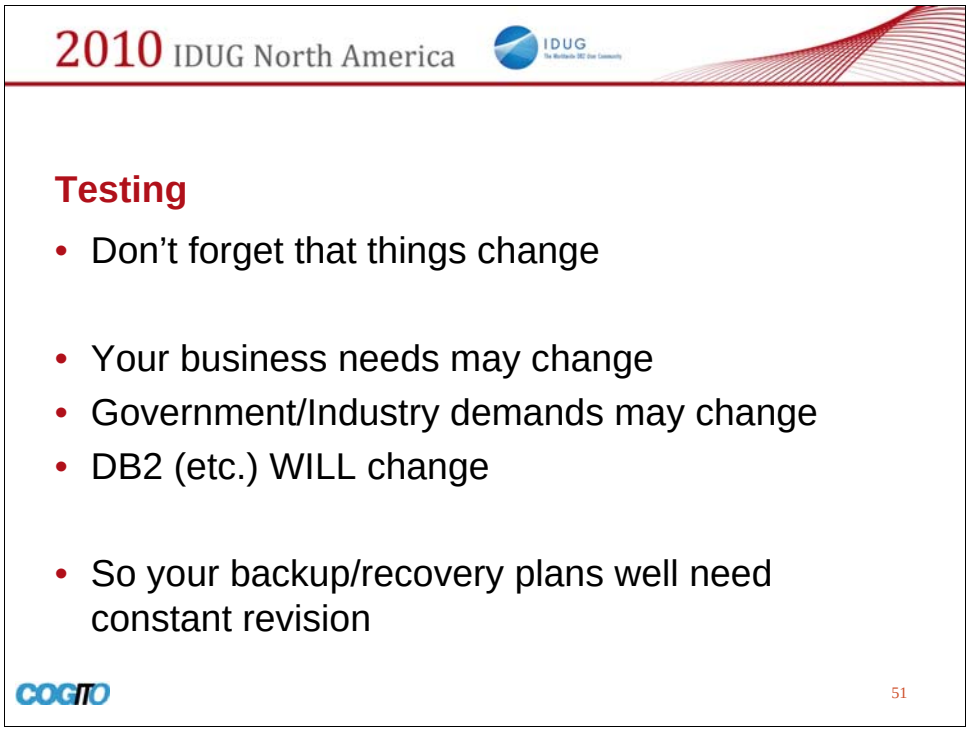
System level point-in-time recovery

- However, there is a string of pre-requisites
- Z/OS V1R5 or above
- Disk controllers supporting ESS FlashCopy API
- Specific DB2 named HSM constructs
- Defined SMS target copy groups
- DB2 must be running in at least V8 New Function Mode

There are many pre-requisites before these utilities can be used, as they rely on DFDSShsm services and hardware assists

Also, all of the DB2 pagesets must be defined to SMS and the SMS storage pools must be named in a specific way

Finally, DB2 must be running in Version 8 New Function Mode



Possibly the most important thing to take from this presentation is the necessity for testing

And more testing

And even more testing

Unfortunately our DB2 world (and the world in general) is NOT static, and things do change

It may be that a new version of DB2 (or even DB2 maintenance) will affect your recovery plans and either render them inoperable or may introduce new steps or subtle changes

Questions/Thoughts?

Phil Grainger
phil.grainger@cogito.co.uk

B05: DB2 for z/OS Backup and Recovery