

# Harnessing the Power of the DB2 Logs

**Rich Wolfson**

**Principal Consultant**

**[richard.wolfson@ca.com](mailto:richard.wolfson@ca.com)**

September 15, 2010



- No one will disagree that the DB2 log is a vital part of DB2 and DB2 recovery
- However, once you look under the log's "hood" and start to exploit its content, many day-to-day tasks can be changed for the better
- This presentation will look at real-life scenarios of how DB2 sites are squeezing the most out of the log to manage audit control, application recovery, change propagation, application QA, and assist in disaster recovery scenarios

# Disclaimer

- This presentation explains the DB2 logging basics and how DB2 users around the world are using the Log content to perform various tasks
- This presentation does not cover Log Monitoring – nor does it provide any guidelines for how to tune DB2 logging
- No programs/samples are provided in order to accomplish what is listed in this presentation
  - A number of ISV's and IBM provide software to accomplish what is outlined
  - and the patient user can of course write these programs too

# Agenda

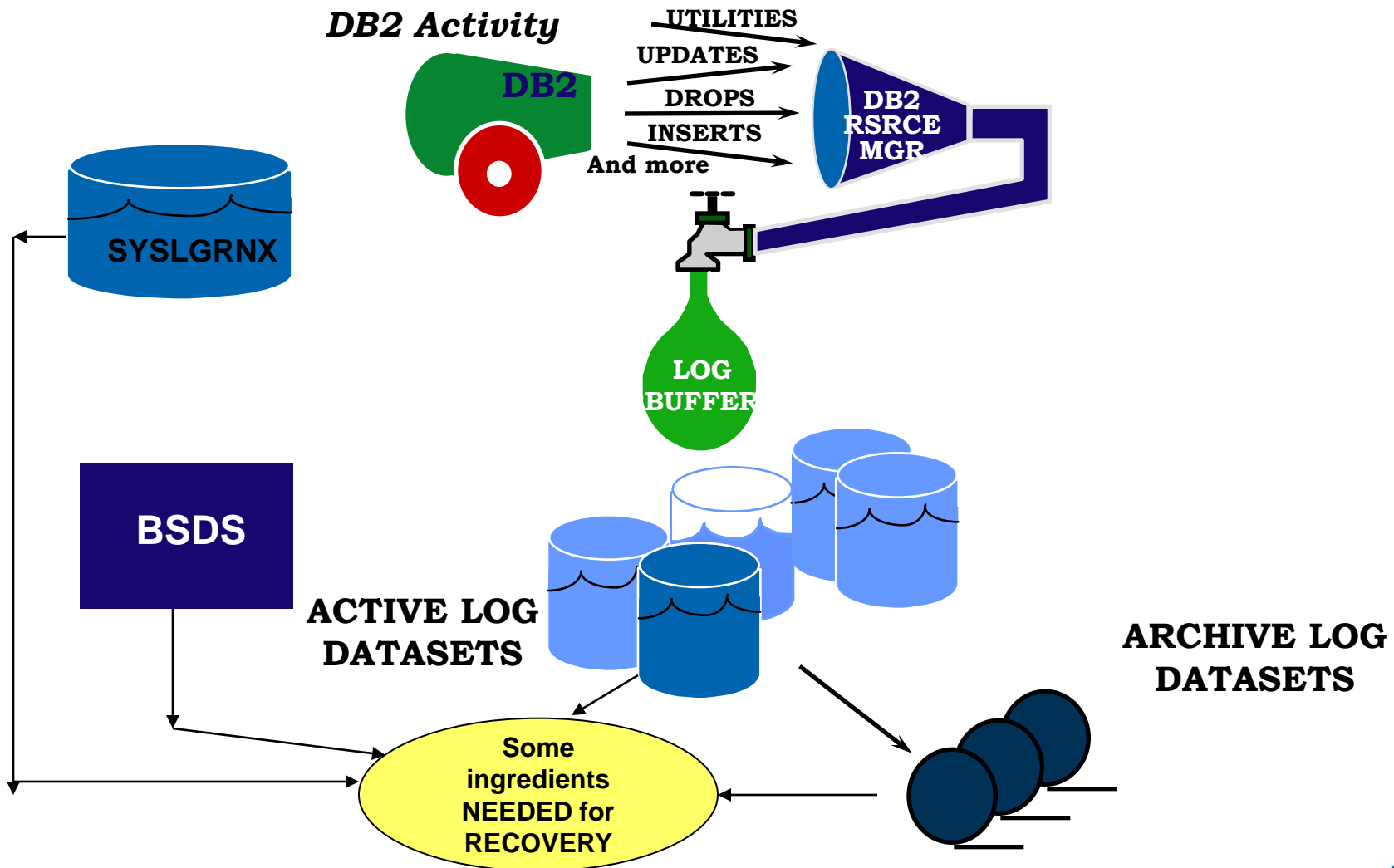
- DB2 Logging fundamentals
  - What is logged and not logged
  - Data Capture Changes (DCC) truths and myths
  - Trigger challenges
- How can DB2 Log records be used to optimize the daily job
  - Audit Control
  - Change Propagation
  - Application Recovery
  - Application Quality Assurance
  - Disaster Recovery Scenarios
  - Violations to your standards (IC, commit, rollback)

# DB2 Logging fundamentals

- Why use LOGGING – is it necessary ?
  - It's overhead
  - It costs in terms of performance, DASD, administration, clean-up .....
  - It's an insurance – just in case we get an accident
- In a perfect world
  - No need to ROLLBACK
  - No need to RECOVER
  - No program errors
  - No hardware errors
  - No power failures
  - No hurricanes, terror attacks, fraud, .....
- Let's get the MOST out of the LOG since it's here !

} *Used to be the threat*

# DB2 Logging Fundamentals



# Active Logs

- The DB2 log contains the info needed to recover the results of program execution, the contents of the database, and the DB2 subsystem...
- It does NOT contain info for accounting, statistics, traces, or performance evaluation
- DB2 records all data changes and various other significant events in a log as they occur
- This log is stored on DASD - the **active** log
- Should have at least 3 active log datasets
  - One in use
  - One offloading to archive
  - One or more available for use
- These usually duplexed to reduce possibility of integrity issues after DASD failure etc.

# Log Buffers

- Input & output buffers defined in DSNZPARM
- When an update occurs, log records are written to the log buffers
- This must be complete before DB2 regards the update as complete
- Two modes of writing buffers to the active log
  - NOWAIT - Asynchronous
  - FORCE - Synchronous at COMMIT
- Note: IMS and CICS have two FORCES
- WRITE THRESHOLD also forces a write to log

# BSDS – Bootstrap Datasets

- BSDS tells DB2 about the state of the logs
- Coded into DB2 start-up JCL
  - Only datasets other than LOADLIB required for system restart
- Contents of bootstrap
  - High RBA written to active log datasets
  - Status and RBA ranges of active logs
    - DB2 events are recorded without a date and time, but with an RBA
  - Information on available archive logs
  - Conditional restart definitions
  - Checkpoint description queue
  - DDF communication record
  - Information on members of a data sharing group

# Archive Logs

- Archive logs are off-loaded copies of an active log dataset and are registered in BSDS
  - Simply a sequential file of what was in that active log
- Offload process gets initiated when
  - An active log dataset becomes full
    - Or nearly full – recommend 85% max
  - DB2 starts and there is a full active log
  - -ARCHIVE LOG command issued (DB2 V2.3)
  - An I/O error occurs writing to an active log
  - As the last active log dataset fills up
- If dual logging you get 2 archive logs

} *Both uncommon events*

# More on Archive Logs

- Sequential datasets on DASD, MSS or tape
  - Can be managed by DFSMS or DFHSM
- Dynamically allocated using a name chosen by DB2 and a generic UNIT if on DASD
- BSDS keeps a specified number of archive logs
- Can also specify retention period
- Make sure these are high enough!
- Deleting archive does not remove entry from BSDS
  - Use change log inventory (DSNJU003)

# Yet more on Archive Logs

- There are many reasons for keeping lots of archive logs on disk
  - Performance
    - DASD archive logs can be processed by multiple recover processes... plus it's faster
  - Availability
    - You know they are there when you need them
  - Data Sharing
    - ANY member of a data sharing group may need access to any OTHER members archive logs
- But only finite space available
- Log compression can help
  - ISVs have log compression tools that can compress over 90%
  - Can be totally transparent to DB2

# Archive Logs and the BSDS

- BSDS contains a list of archive logs
- BSDS gets copied when an archive is created
  - Prefix.Bnnnnnnnn dataset name
- Note that the copied BSDS does not contain the archive log that is created along with it
- This will be needed if you need to perform disaster recovery processing
  - This requires running a change log inventory
- Some sites have an automated procedure to add the latest archive to the BSDS using the Prefix.Cnnnnnnnn dataset name

- Directory table; not accessible via SQL
- Contains the RBA when any tablespace or partition is opened or closed for update
- Note: information is also in DB2 log
- DB2 uses SYSLGRNX to speed up recovery by limiting the amount of log data which needs to be scanned
- MODIFY utility removes this information along with SYSCOPY rows

# Relative Byte Address - RBA

- DB2 events are recorded using Relative Byte Address (RBA), not dates and times
  - RBA starts at 0 at first DB2 start up
  - Increases from there ad infinitum!
  - Largest number over 280,000,000,000,000
- DB2 V4 introduced Log Record Sequence Number (LRSN) for data sharing systems
  - 6-byte hex value derived from a store clock timestamp

# Unit Of Recovery ID - URID

- Each unit of recovery is identified by the RBA of its first log record - the URID
- This is contained in every log record for that unit of recovery
- All these records are backward chained for fast recovery in a rollback situation
- Forward recoveries go through the log sequentially
  - Because of before UPDATE records

# Physical Log Records

- Physical log records are different sizes
  - Depends on how much of the CI is available
- One physical record can contain 1 or more logical records, or bits of a logical record
  - Makes reading logs complex
  - IBM provides methods to decode structure
  - See Admin Guide appendix C (version 7 & 8)
- Each physical record contains a 21 byte Log Control Internal Definition (LCID)

# Logical Log Records

- References to log records usually mean logical log records
- A record can be up to 32815 bytes long, and can span one or many physical records (i.e., VSAM CIs)
- A segment is a section of a log record in a physical log record
- Each logical record contains a Log Record Header or LRH record
  - First segment contains whole LRH
  - Subsequent segments have first two fields

# Types of Log Records



- Unit of Recovery Log Records
  - Undo/Redo
- Checkpoint Log Records
  - Log info on all open & restricted datasets, and inflight Units of Recovery
- Database Page Set Control Records
  - Control allocation, open & close of every pageset
- Other Stuff
  
- Let's examine each
  - in more detail...

# Unit of Recovery Log Records

- Most of these are undo or before images and redo or after images
- There are also several other types including
  - Begin-UR and end-UR
  - Phase 1-2 transition records
  - End phase 2 records
  - Begin abort and end abort

# Undo and Redo Records

- These are the most useful log records for most sites - they contain a guaranteed source of all update activity on the DB2 subsystem
  - REDO info is required if the work is committed and later must be recovered
  - UNDO info is used to back out work that is not committed
- Note that compressed records are stored compressed in the log
  - Decompression requires access to the decompression dictionary/module
  - What happens if this has been rebuilt?
  - Also remember that you can get 255 records on a page if it is compressed

# Insert Records

- Only get a REDO record, which contains the complete after image of the row
- DATA CAPTURE CHANGES has no impact

# Update Records

- Updates generate an UNDO and a REDO
- Standard logging contains the first updated byte to the last updated byte or to end of the row if the logged data is variable
  - Put VARCHARS at the end of the row
  - Put commonly updated columns together
- If DATA CAPTURE CHANGES is active, you get the complete before image
  - Required for change propagation

# Delete Records

- For unqualified (a.k.a. mass) deletes in a segmented TS where there is no referential integrity
  - If DATA CAPTURE CHANGES you get an UNDO record containing the whole row
  - If no DATA CAPTURE CHANGES, DB2 only logs the changes to the Spacemap page - no row information gets logged
- All other deletes get an UNDO record containing the whole row

# Index Records

- For all index changes, DB2 produces a record containing the following
  - The offset in the index page
  - The old or new key value depending on type
  - The RID of the data row

# Compensation Records

- DB2 also logs rollbacks using UNDO/REDO records called compensation records
- This allows a warmstart or rollforward to take account of rollbacks
- Note that if you have  $n$  updates, and you cancel a rollback near to completion, you will have  $2n$  updates to rollback next time, and so on...
  - This is especially important during warmstart after a DB2 failure
  - May be a case for deferred restart
- Only cancel a DB2 job once!

# Checkpoint Records

- To reduce restart time, DB2 takes periodic checkpoints during normal operation
- Begin and End checkpoint
- Unit of recovery summary
  - Information on in-flight UR's
- Image copies of special tablespaces
  - SYSUTILX, DBD01 and SYSCOPY
- Pageset summary
  - Log RBA of earliest checkpoint interval containing log records which have not been applied to DASD
  - One record per open pageset
- Pageset exception summary
  - One record per pageset in an exception state

# Pageset Control Records

- Control pageset allocation, Open and Close of every pageset
  - Also held in SYSLGRNX... but also required in log so it's available at restart
- Exception Status - for example
  - Stopped
  - Started UT or RO
  - Write Error range
  - Deferred Restart
  - Indoubt
  - COPYP, CHKP, RECP status

# So... DB2 Logging Fundamentals

## — Activity involving changes to DB2 is documented in the LOG

- Any change to DATA pages (REDO and UNDO information)
  - Insert, Delete, Update
  - Any activity performed by DB2
- Any change to INDEX pages (REDO and UNDO information)
  - ICOPY doesn't matter
  - Any activity performed by DB2
- UNDO information
  - If ROLLBACK issued
  - If abend, SQL error etc.
  - Once COMMIT executed – this information is “no longer needed”
- GRANT, REVOKE, BIND, FREE, DROP, CREATE, ALTER,....
  - Any catalog change is logged as well
  - In reality these commands are INSERTS, DELETES and UPDATES
- Some utility records, imagecopy info for system objects, .....

# DB2 Logging Fundamentals

## — What is logged when logging occurs ?

- Connection-type : TSO, BATCH, UTILITY, CICS, IMS
- Connection-id : established by attachment facility  
CAF : TSO/DB2CALL TSO : TSO/BATCH
- Correlation-id : associated with DB2 thread
- Auth-id : who is executing this transaction
- Plan name : The PLAN under which this operation executes
- RBA / LRSN : Timestamp of operation
- URID : Unit-of-Recovery id
- DBID, PSID and OBID : Internal identifiers for the object updated
- Operation type : Insert, Update, Delete, Utility type,.....
- Data changed : (much more about this later)

*Extracted From the IBM manual Example of a log record sequence for an INSERT of one row using TSO*

Type of record	Information recorded
1. Begin_UR	Beginning of the unit of recovery. Includes the connection name, correlation name, authorization ID, plan name, and LUWID.
2. Undo/Redo for data	Insertion of data. Includes the database ID (DBID), page set ID, page number, internal record identifier (RID), and the data inserted.
3. Undo/Redo for Index	Insertion of index entry. Includes the DBID, index space object ID, page number, and index entry to be added.
4. Begin Commit 1	The beginning of the commit process. The application has requested a commit either explicitly (EXEC SQL COMMIT) or implicitly (for example, by ending the program).
5. Phase 1-2 Transition	The agreement to commit in TSO. In CICS and IMS, an End Phase 1 record notes that DB2 agrees to commit. If both parties agree, a Begin Phase 2 record is written; otherwise, a Begin Abort record is written, noting that the unit of recovery is to be rolled back.
6. End Phase 2	Completion of all work required for commit.

# DB2 Logging Fundamentals

## — What is NOT logged ?

- SELECT statements
  - Auth-id switching
  - SQLID assignments
  - Access denied
  - DB2 Commands
  - STOP and START operations
- 
- ~~PACKAGE name (also TRIGGER package)~~
  - If Update, Delete, Insert derived from a TRIGGER (V8 solves this) This can be a huge challenge – we will discuss TRIGGER issues later



SMF reporting can be used to track these Events.

## -START TRACE (AUDIT) CLASS (x,y) DEST (GTF) LOCATION (\*)

Audit class #	Events description
1	Access denied because of inadequate authorization. (default)
2	GRANT and REVOKE statements
3	CREATE, ALTER, and DROP statements that affect audited tables
4	<p>Changes (Only the first attempt to change a table, within a unit of recovery)                      If the data is not changed then the attempt to make a change is recorded                      If the change is not successful and is rolled back, the audit record remains; it is not deleted. (includes access by the LOAD utility)</p> <p>Accesses to a dependent table that are caused by attempted deletions from a parent table are also audited. The audit record is written even if the delete rule is RESTRICT, which prevents the deletion from the parent table. The audit record is also written when the rule is CASCADE or SET NULL, which can result in deletions that cascade to the dependent table.</p>
5	All read accesses.
6	The bind of static and dynamic SQL statements
7	Assignment or change of an authorization
8	The start of a utility job, and the end of each phase of the utility.
9	Various types of records that are written to IFCID 0146 by the IFI WRITE function.

# Data Capture Changes – Truths and Myths

- Truths
  - INSERT - entire row is always logged
  - UPDATE in general - from first changed byte to last changed
    - If VARCHAR present - from first changed byte to end of row
    - V7 changed this : if LENGTH not changed, logging as if no VARCHAR (unless compression or EDITPROC)
    - If DCC enabled – entire before and after image is logged
  - DELETE
    - **Qualified** (DELETE from tb WHERE .....)
      - Every row deleted is always logged in its entirety
    - **Unqualified** (aka. MASS delete : DELETE FROM tb) and NO RI
      - Without DCC - only logging of changes to spacemap pages.
      - With DCC – every row is deleted and logged individually. Might consider to disable/enable DCC in programs doing mass deletes.
  - If tablespace compressed – log-records are compressed
  - The benefit of having Data Capture Changes enabled on tables will be covered later

```
ALTER TABLE tbc.tbnm  
DATA CAPTURE CHANGES;
```

```
ALTER TABLE tbc.tbnm  
DATA CAPTURE NONE;
```

# Data Capture Changes – Truths and Myths

## – Myths

- Logging will increase too much
- CPU overhead will increase too much

} *Maybe NOT – it DEPENDS*

## – Consider the following

- Ratio between Inserts, Deletes and Updates ?
- Are updated columns placed next to each other ?
- Is compression used ?
- Any varchar columns updated ?

## – Also – only a smaller portion of the log comes from DML

- Checkpoint records
- Pageset allocation and summary records
- Exception statuses
- Backout information
- Index updates
- Etc. etc



Benchmarks show  
DML log records take  
up about 25-30%

- Having the entire BEFORE and AFTER image (Data Capture Changes turned on) can be a good insurance – just like imagecopy is  
(measure one week without and one week with DCC enabled)

# Exploiting the DB2 Log - scenarios

- Logging fundamentals covered
  - What is stored in the log
  - What is not stored in the log
  - When is logging done
  - What is Data Capture Changes
  - Data Capture Changes impact on DELETE and UPDATE
  - Logging impact based on physical attributes (compression)
  
- Real life scenarios
  - What is the “DB2 World” doing out there .....
  - How can the DB2 log do for you
  - Time for alternative ways of thinking

# Audit Control

- Besides DB2 Recovery – Audit control is probably the theme using the DB2 Log the most
- Sarbanes Oxley and other regulations will not make this issue less important and the Log less used
  - Which user-id changed WHAT and WHEN
  - Tracking of any changes to sensitive data
  - SYSADM is very powerful – many auditors get less paranoid when reporting of any activity implemented
  - Futures :
    - Pattern analysis, excessive logging against certain objects, changed behavior, anomalies etc.
    - Fraud and “Identity Theft” will be a key driver of log analysis

# Audit Control

- Another issue related to Sarbanes Oxley – which objects are created, altered, dropped ?
  - DB2 catalog shows when object was created - LAST TIME
  - DB2 catalog shows when object was altered - LAST TIME
  - DB2 catalog does NOT show which objects have been dropped
- Everything is in the Log in order to report
- Traverse through the log and find changes to the DB2 catalog in order to have a complete log for Object Change Management

# Change Propagation

- Many sophisticated solutions available - many require Log Capture Exit to be active
- The Log itself can be used as “poor man’s change propagator”
  - When no massaging of data
  - When no synchronization with IMS, VSAM and other files
  - Do not forget:
    - Check if URID looked at has been committed
    - Even though a table has been dropped – the log-records are still in the log
    - MASS-deletes might be a challenge if DCC isn’t enabled

# Change Propagation

- Get a list of OBID's from the catalog for the tables in interest (remember a table might have been dropped and OBID re-used)
- Traverse through the Log starting where you last stopped
- Get the REDO log-records
  - INSERTs and DELETes : the entire row image is ready
  - UPDATES when DCC : BEFORE image used for WHERE clause and AFTER IMAGE used for UPDATE SET clause.
  - UPDATES without DCC? Two options:
    - Use RID from log-record and look at active VSAM dataset for table. Log must be read from current-point-in-time and back to URID to check if RID has been updated after the current log-record.
    - An alternative is to read the imagecopy and then read the log from IC-RBA and forward to URID currently under investigation

*(see why Data Capture Changes can be a good choice ? )*

- Map column names to log-record content.
- Create the format you like – if SQL DML, these can be applied on any RDBMS

# Application Quality Assurance

- Two different scenarios observed where Log processing can optimize daily tasks when developing programs.
- ☑ Ensure SQL statements executed in correct sequence and the proper tables and columns are manipulated.
  - Develop a panel where the developer can type PLAN name and optional date/time
  - Locate the PLAN name and the USER-ID
  - Retrieve the REDO log-records matching the criteria and produce a report and/or SQL statements to be verified by the developer or project team
  - Table names as input (beside PLAN and USER-id) is not recommended in order NOT to forget any table updates due to Referential Integrity

# Application Quality Assurance

- ☑ For iterative test cases – the tables updated must be backed out in order to conduct the same test after program changes completed:
  - **LOAD REPLACE or RECOVER TORBA is a solution ????**
  - If many tables or loads of data – this approach is time consuming AND other applications cannot access the tables
  - Develop a panel where the developer can type PLAN name and date/time
  - Locate the PLAN name and the USER-ID.
  - Retrieve the log-records matching the criteria and produce the SQL statements to be executed in order to re-establish the data to the PIT immediately prior to the start of the PLAN specified.
  - Since SQL statements are created to “backout” the PLAN, these can be executed concurrently with other activity.
  - Table names as input (beside PLAN and USER-id) is not recommended in order NOT to forget any table updates due to Referential Integrity



Time to look at some

# HORROR

stories



# Application Recovery – Intelligent Recover ?

- Have you ever faced a situation like this:
  - A batch job is executed with “wrong” SQL statements. Many hours later someone finds out data is missing in a table. It appears that the job by accident deleted 80K rows. Only application RI involved. Recovery not feasible due to online activity (no desire to re-enter transactions added after “the mess”).
  - All the DELETES are on the log. We know the name of the plan used in the batch job, and we even know the table name where the deletes happened.
  - Like the previous example – we can traverse through the log and create INSERT statements based on the log-records which DB2 produced when these famous deletes were executed. This will allow us to do an “Online Recovery” with no impact.

## WARNING :

Application logic based on existence of a row ?

# Application Recovery – Intelligent Recover

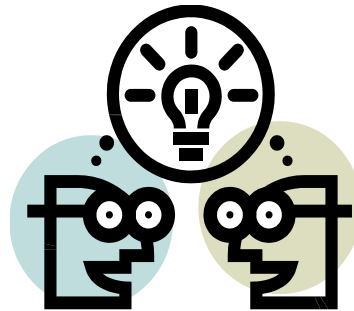
- An even worse scenario:
  - A logic error caused a few rows to be deleted from one table.
  - Not a big deal ..... well
  - This table was involved in a +100 table RI structure
  - Not a big deal .....well
  - This table was THE parent table with DELETE CASCADE
  - Again – this was observed hours after the “disaster” and subsequent batch jobs and online applications had done 1000’s of updates.
  - This scenario was evaluated and a choice between two evils had to be done. The least evil approach was picked and deleted rows retrieved from the log and re-applied
  - Necessary to investigate application logic before this attempt is made – but a complete outage for hours was avoided

# Disaster Recovery Scenarios

- Prepare for new application “Go Live” on Monday
  - About 20 new tables and 100MB table data
  - Application integrated with existing applications
  - A total of 150 tables and 1 TB data
  - IMS DB involved
- Sunday chosen to test new application
  - 20 selected people to test online system with real transactions and real data for a couple of hours
  - Only a few thousands DML statements estimated to be executed – but 150 tables and IMS DB involved
  - Sunday afternoon GO or NOGO based on the test
    - In case of a NOGO? Make the “test” invisible
- The test turned out to be a GO, but:
  - We simulated a NOGO, and extracted the activity done by those selected users in this isolated environment - which took about 5 minutes - Then, we re-executed all this data in a test environment
  - Thus, we would have been able to do the entire "recovery" in less than 10 minutes
- Again, once you know what's in the log, think about some alternative methods to speed up your recovery, if needed.

# Issues of concern

- We have covered some isolated scenarios where alternate methods of recovery have been selected
- We have seen how the DB2 log can be used for many other purposes than just recovery
- Now it's time to sit down and think carefully about the consequences



# Trigger Issues

- Many issues when triggers exist in the environment where the described procedures are used to either re-execute or backout SQL statements using SQL statements
- DB2 V7 does NOT log if DML is from a trigger – DB2 V8 does set a flag, but this only solves SOME issues
- Wish list: A parameter to specify if triggers should be executed or not
- Utility approach is different – except for ONLINE LOAD, all utilities skip trigger processing
- Important to understand what is happening when triggers are involved when the described processes are used
- Let us look at the challenges and how these can be addressed...



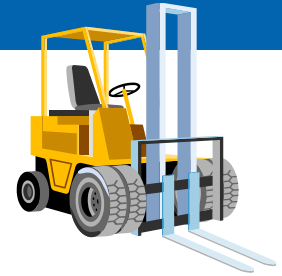
# Trigger Issues when triggers exist

- Procedure when executing log generated SQL:
  - If target environment has NO triggers, the only issue is to consider IF SQL extracted from the log should hold SQL residing from TRIGGERED statements or only the REAL EXECUTED statements (remember this is a V8 only possibility). When reading the log, the statements originating from triggers can be bypassed.
  - If target environment has triggers OR it's a pre-V8 environment:
    - The challenge is – the statements generated from the log hold both the original executed SQL statements AND the triggered SQL statements
    - If nothing is done – the triggers will be fired again when executing the SQL generated from the log (double transactions)
    - Save the target environment triggers (in the sequence they were created), and drop them prior to executing the log generated statements.
    - Upon completion : create triggers again
    - Outage might NOT be acceptable

# Identity Columns and Sequences

- Like triggers – Identity columns and usage of Sequences can complicate the scenarios of using the DB2 log as an alternate recovery method
  - DELETE statements might not be a problem.
  - UPDATE statements neither
  - INSERT statements probably the biggest challenge
    - When the target environment is not 100% identical in terms of data
    - Especially if generated ALWAYS is used
    - When target is ahead of “next available” number
    - When RI is involved using Identity columns / Sequences

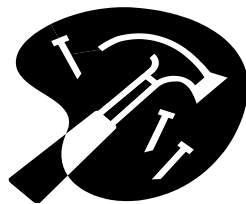
Some



# Miscellaneous



scenarios



# Track Violations to YOUR Standards

- Most DB2 sites have either standards, guidelines and/or ROT like :
  - Application commit frequency (plan level)
  - Updates per table between commit points (plan level and lock escalation)
  - Number of updates per tablespace between imagecopies (extended recovery time)
- The log holds:
  - Every DML statement
  - Commit points
  - PLAN name
  - OBID for tables
- Traversing the log can simply have these counters in order to find the offenders so outage can be minimized for:
  - Application abends / rollbacks
  - Recovery time “when” needed

# How to Read Log records

- Instrumentation facility (performance trace)
  - Requires DB2 to be up
  - Performance traces provide overhead (for some unacceptable)
  - Volume of output can be huge
- IBM supplied macro DSNJSLR
  - Can be used while DB2 is inactive
  - Reads log records with OPEN, GET and CLOSE by Log RBA
- DB2 Log Capture Exit
  - Executes in real time while DB2 is up
  - Very critical for performance
- Highlvl.SDSNSAMP(DSNDQJ00)
  - Describes Log records
- The vendor method
  - Log Analysis tools for DB2 for z/OS

# DB2 v8 z/OS new Log stuff

- Active and archive logs are increased !
  - Maximum size of active or archive log data set = 4 GB (APAR for v6&7)
  - Maximum number of archive log volumes increases from 1000 to 10,000 per log copy
    - DB2 system can remain recoverable without having to take frequent image copies
  - Maximum number of active log data sets increases from 31 to 93 per log copy
    - more active logs means less chances for DB2 to require the archive log data sets for an extended rollback or media recovery
    - faster recovery as active log read is generally faster than archive log read
  - Changes are required in the BSDS to allow the new maximum values
- New messages to monitor system checkpoints and log offload activity, to identify problems faster
- During restart processing, new progress message during the backout of postponed URs (issued at two minute intervals)

# DB2 9 z/OS new Log stuff

- Logging performance is improved substantially
  - data sharing improvements and archiving
- Optional logging of INSERTs (to improve performance)
- Log contention relief for data sharing, archive log striping
- Allow log encryption
  - both active and archive logs
- Not log tables
- Utilities can truncate log based on timestamp

# Wrap up

- A special thanks to Steen Rasmussen, Phil Grainger and Greg Parrish for their contributions to this presentation.
- The “horror stories” are true. The names have been omitted to protect the innocent.
- Hopefully this can be of some benefit to avoid horror stories of your own!



thank you