# A very untraditional approach implementing schema changes

Steen Rasmussen, Technical Consulting

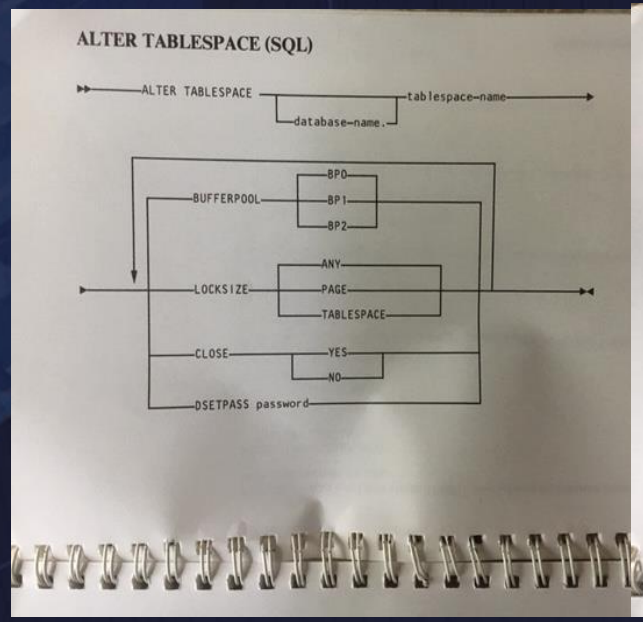
Broadcom Mainframe Software Division
Email Address: steen.rasmussen@broadcom.com or Db2steen@yahoo.com

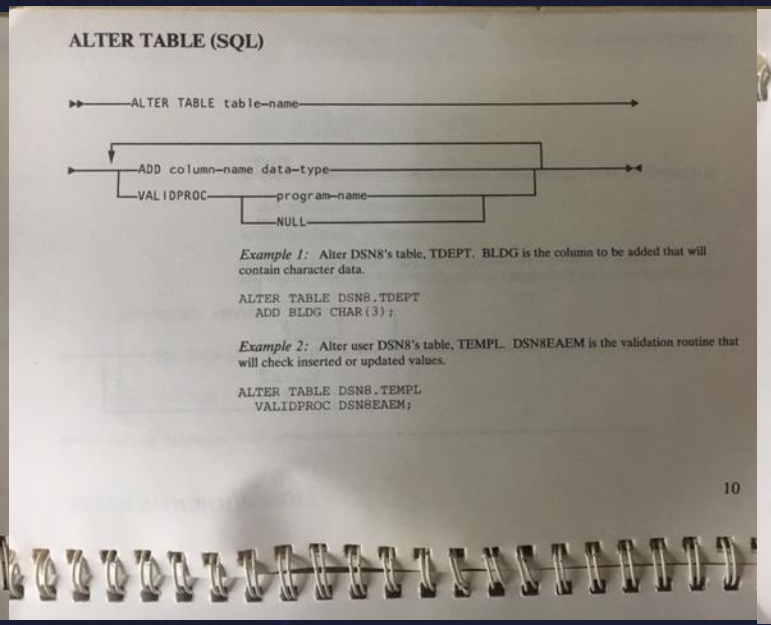
# Agenda

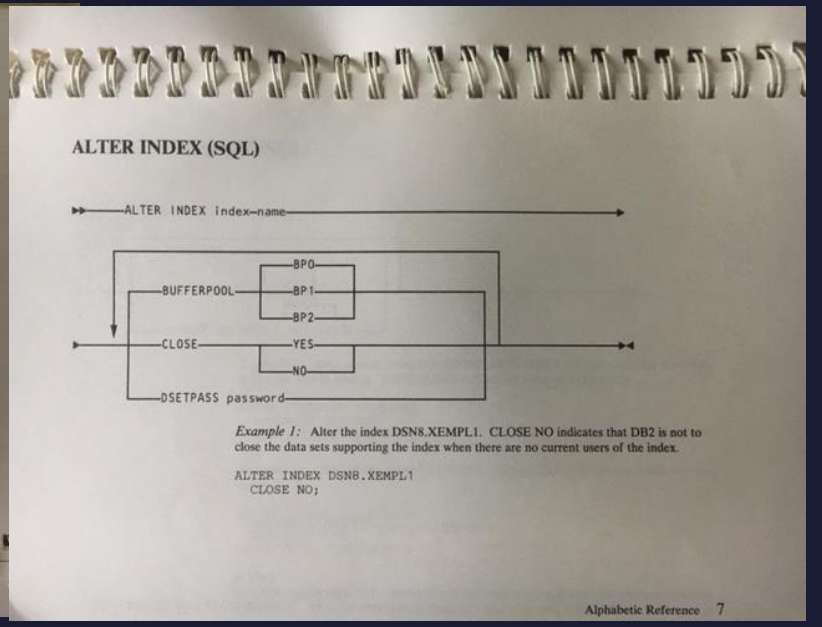
- *Traditional schema change implementation and evolvement*
- *Challenge-1 : Access Patch changes*
- *Challenge-2 : What if fallback is required – outage can be long and complex !!!*
- *Untraditional schema change approach – “cloning” production and how to make the outage almost disappear when fallback needed*
- *Data Refresh process from hours to minutes – kill two birds with one stone due to this “unique” environment*

# Traditional schema change implementation and evolvement

- Schema changes used to be “simple” but troublesome (*here DB2 V1R1M0)*
  - Almost all schema changes required the DBA nightmare of UDCL
  - Life has become much better and less intrusive and more DBA friendly
  - Application outages are being eliminated or reduced dramatically



# Schema Changes Evolvement Over Time

- From Db2 V1 through V7
  - Add Column at the end of a table
  - Add/Drop PK/FK
  - Change QTY's
  - Change PCTFREE / FREEPAGE
  - Change Tablespace Compression
- Db2 V8 really started to change the capabilities
  - Column Data Type and length increase
  - Add Partition to the end
  - Rotate Partition

Most changes needed reorg/load replace to make them effective

# Schema Changes Evolvement Over Time

- Db2 V9
  - Rename Column, Table, Index (If views etc. then these have to be dropped and re-created) – the scenario described later illustrates the complexity of RENAME
  - Add column to an index (and INCLUDE clause)
- Db2 10
  - UTS tablespace type introduced
  - Pending Changes (reorg needed to materialize – much better than drop/create)
  - PGSIZE, SEGSIZE, DSSIZE
  - MAXPARTITIONS for PBG
  - Column default value

# Schema Changes Evolvement Over Time

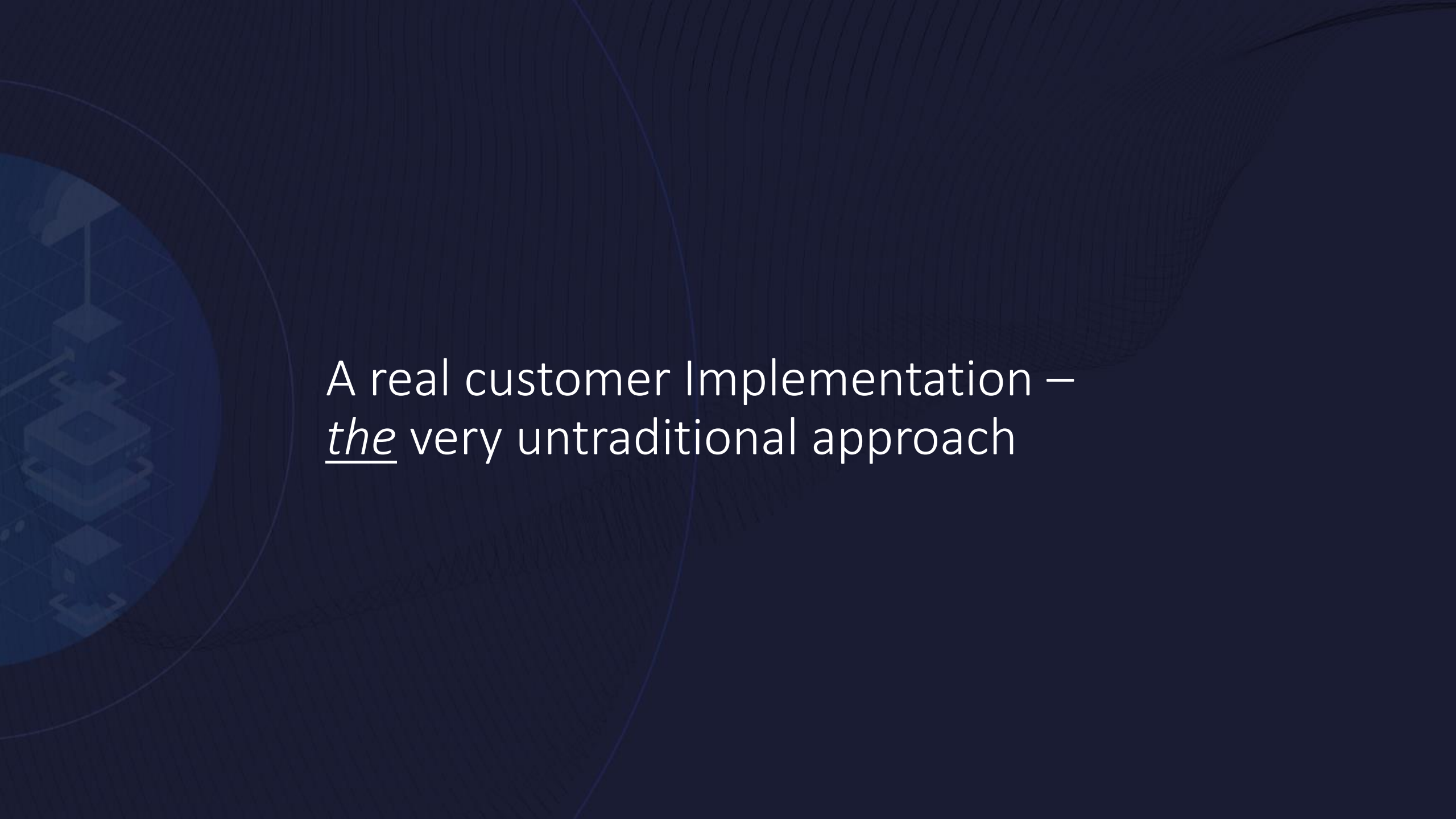
- Db2 11
  - Drop Column *(I love it but scared to death)*
    - *Great method to change data type, decrease size or other intrusive change: ADD + DROP)*
  - LIMITKEY as a pending change
  - BP size as a pending change
- Db2 12
  - Column length change immediate or pending *(controlled via ZPARM DDL_MATERIALIZATION)*
  - Relative byte addressing for UTS PBR
  - Add Partition in the middle
  - Conversion to PBR2 -> partition wise changes much easier
    - *Especially when partition is getting "full"*

# Schema Changes Evolvement Over Time

- From Db2 V8 – most schema changes Immediate or Deferred/Pending
  - Online Reorg needed to instantiate
  - Different types of Advisory Reorg : AREO* and AREOR
  - Much less restricted/intrusive compared to REORP ~ “dead in the water”
- The life of a DBA has become a lot better, but . . . .
  - Still need strong change control for various reasons
    - One being DROP COLUMN – if an Online Reorg executed prior to application changes implemented, someone might change your agenda !!

# *Traditional schema change implementation and evolvement – from a highlevel perspective*

- Execute Schema Changes as immediate or as a pending change – unless Unload-Drop-Create-Load is needed
- Potentially prepare backout script
- Execute Online Reorg to instantiate schema changes – or potentially Rebuild Index
- If Runstats is needed – potentially as inline stats *(improved in Db2 12)*
- Rebind Packages
  - Downside:
    - What about changed Access Path(s)
    - Extended Plan Management might not be an option due to schema changes


A real customer Implementation –
*the* very untraditional approach

# High Level Overview of Db2 implementation at …….

- Stand-alone subsystems : Test=6 and Production=3
- Test Data Sharing Groups:
  - DSNB – 6 active members
  - DSND – 4 active members
  - DSNE – 7 active members
  - DSNF – 4 active members
  - DSNH – 5 active members
- ***Production Data Sharing Groups:***
  - ***DSNA – 11 active members***
  - ***DSNG – 12 active members***
  - ***DSNI – 8 active members***

- "FLASH" process covered later:
- Production "cloned" into:
  - DSNA -> DSNL
  - DSNG -> DSNM
  - DSNI -> DSNN

# Untraditional schema change approach – “cloning” production and how to make the outage almost disappear.

- Most Db2 sites perform IMMEDIATE or PENDING Changes
  - *What if fall back*
  - *What about Access Path Changes*
- Highlevel overview of schema change process at ….. :
  1. Create “FLASH” ~ “Clone” production DDL and Data
     *(weekly to **test schema changes ahead of time in identical environment**)*
  2. Once ready to implement in REAL PROD : start production UT/RO
  3. Create new DDL
  4. Unload-Load data into new using USS Pipes
  5. Rename OLD to OLD_OLD and NEW to OLD
  6. Create views etc. REBIND and Start production in RW
     Old tablespaces/tables/indexes kept for a week before dropped
  7. In case of FALLBACK

     Drop views, Reverse RENAME’s and voila !!

# More details about the “clone” environment.

- Once the “FLASH” environment is established
  1. Remember all objects, all data including Packages, Explain tables etc. is available with identical names (only Db2 subsystem-id is different)
  2. Time to practice entire schema change flow in “clone” environment
  3. New tablespaces, tables, indexes etc. created
  4. Data pumped from original tables to NEW using USS-PIPES
  5. Either STATS migration or Runstats
  6. Perform Explain and compare before-after access paths – no surprises later in production
  7. Rename ORIGINAL TABLES to OLD
  8. Rename NEW tables to OLD
  9. Create views etc. REBIND and Start in RW
- Time to implement schema changes in production
  - Exact same procedure as in “FLASH” environment
  - In case of FALLBACK – no worries – will take a few minutes
  - Drop views, Reverse RENAME’s and voila !!
  - Old tablespaces and data available with almost no outage if fallback needed

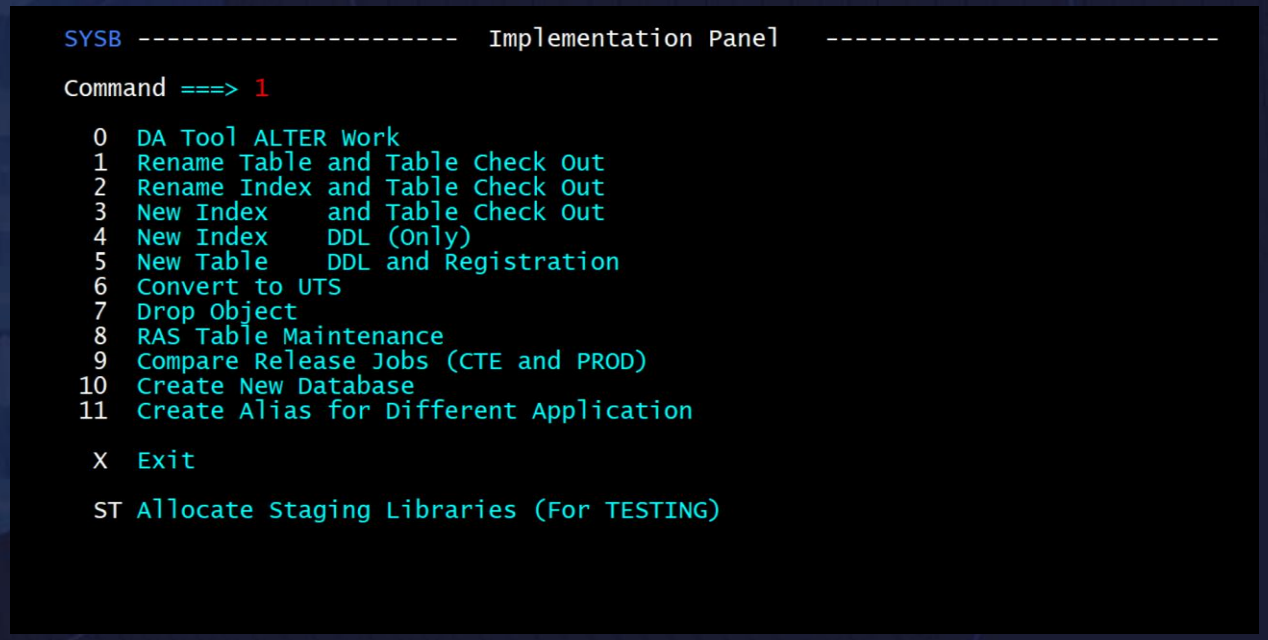
# High Level Description of “Cloning” (home-written)

- This is named “FLASH ENVIRONMENT”

- Every Sunday at a “quiet point” when no long-running batch:
  - SET LOG SUSPEND
  - All Db2 DASD with Db2 “flashed” with the NOCOPY option
  - SET LOG RESUME
  - VSAM RENAME for high-level qualifiers
  - Db2 changed to point to these
  - LPL issues resolved by Db2
  - One member active during backout
  - All object names are identical on both sides (makes catalog changes easier)

# Schema Change Process in Further Detail

- This is the process to generate jobs to perform a Table Rename. The production jobs will be run in weekly “dark window”.
  - ISPF panels and REXX’s all home written

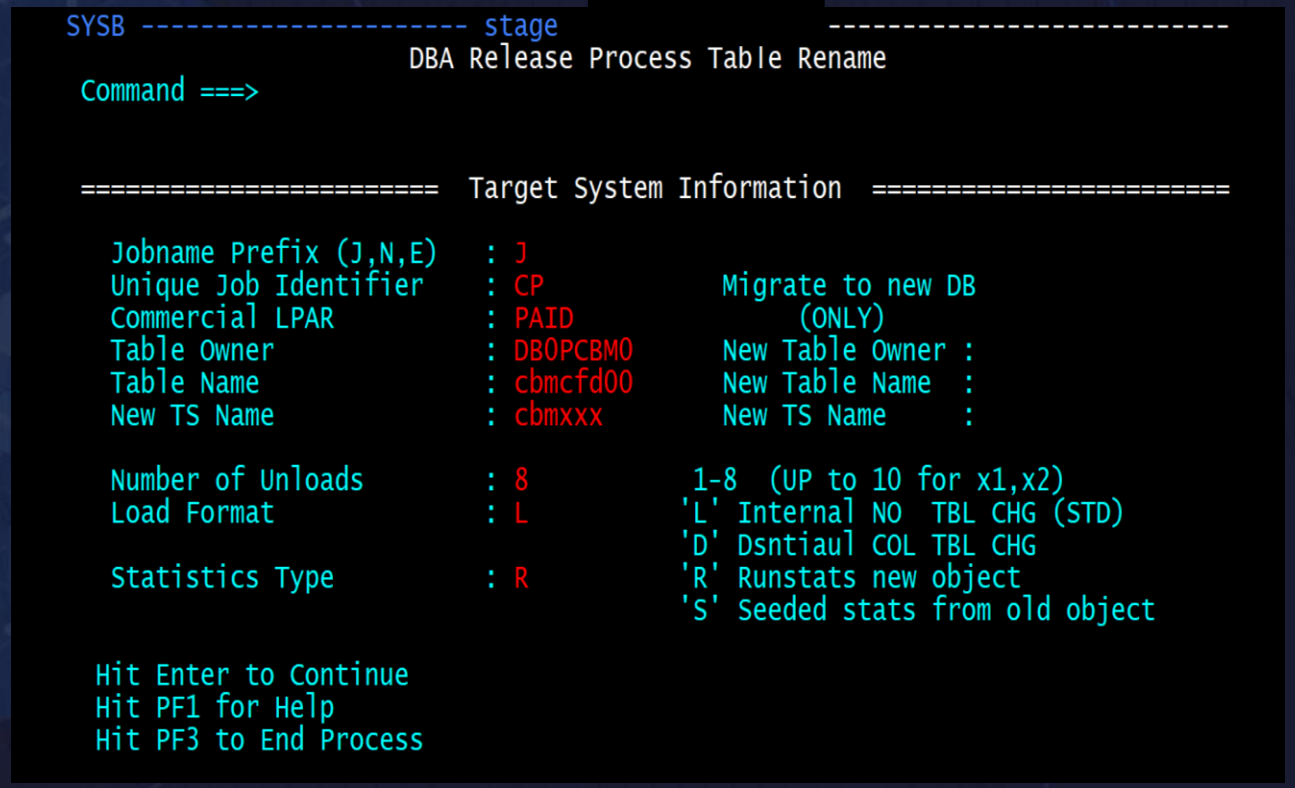



**STEP-1:**

**The “check out” process is simply to ensure that no more than one person will change the same table or dependent object.**

# Schema Change Process in Detail

## Job details entered and verified.

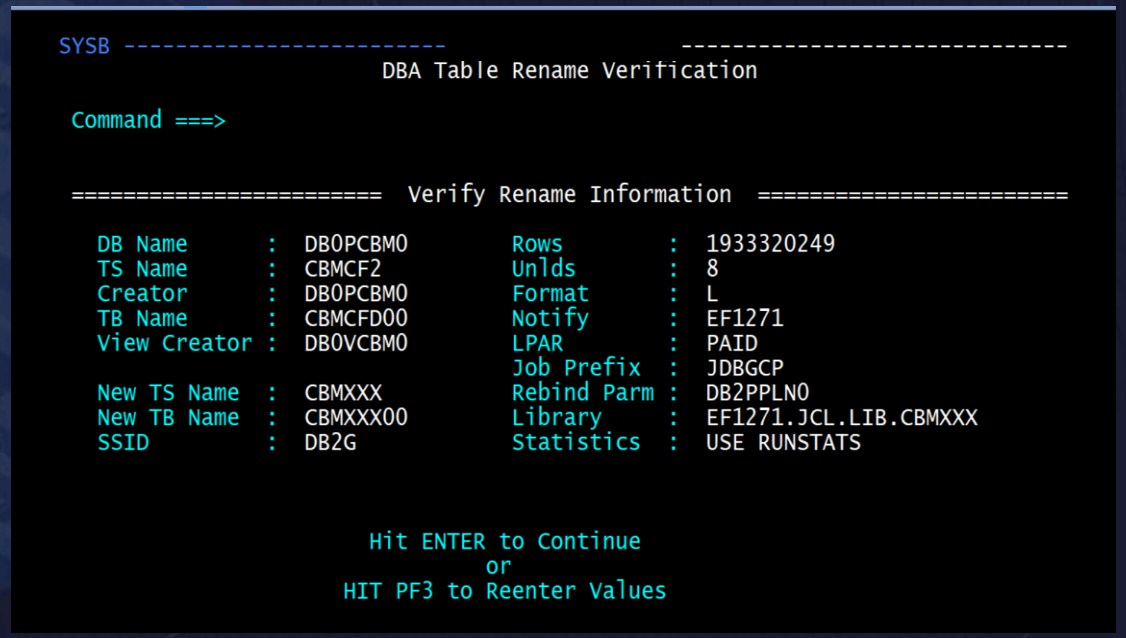



**STEP-2:**

**Details entered for jobs to be generated.**
**Table and tablespace entered to ensure not in use elsewhere.**
**Number of CA Fast Unload jobs to generate using USS Pipes as input to CA Fast Load (parallel unloads to limit window). More details later !!**
**Note that similar technology exists for both IBM and BMC utilities.**

# Schema Change Process in Detail

- Specify dataset to place generated jobs, production row-count, “new object names”, who to notify etc.

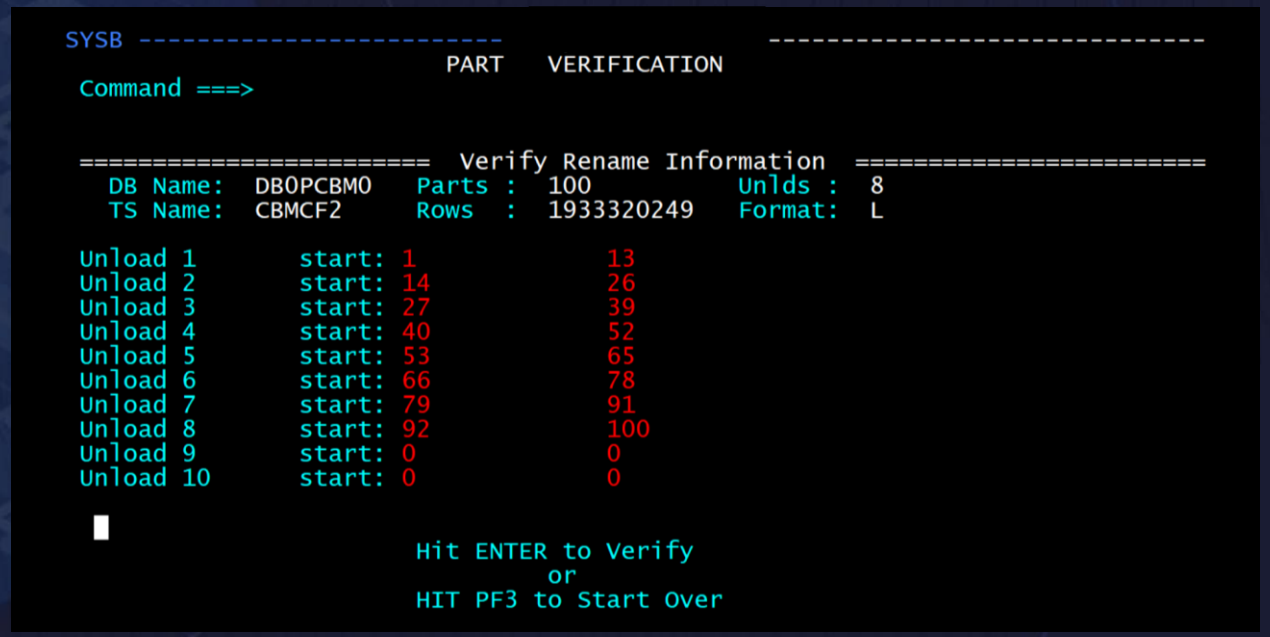



**STEP-3:**

**Besides the mentioned parameters, options to execute REBIND and RUNSTATS (another option is to copy statistics).**

# Schema Change Process in Detail

- Next step is to specify Unload job details.
  - The advantage of using pipes is that the load job will start as soon as the pipes are being filled (saving time)




**STEP-4:**

**Specify how many partitions to handle in each CA Fast Unload job. Will run in parallel using USS Pipes.**

**In this case only 8 unloads executed.**

**As noted: The “PIPE” technology isn’t new but advantage using UNIX System Services is it’s free.**

# Schema Change Process in Detail

- At this stage the dataset is built with the jobs needed to do a table rename – following jobname/member convention:
  - ENV = BDB2 for special production environment
    - Special production environment execution prior to production
    - Once finalized – moved to production and scheduled under CA7
  - ENV = HDBV for production
  - ENV = JDBG for production claim environment
    - JDBG*NP are for test (up to 25 environments)
    - Can be edited and moved to production and CA7
    - Other JDBG jobs for production and executed in “Dark Window” execution between 02 :00 - 05:30am Sunday
    - ***Once finalized moved to CA7***
  - $DBM jobs are executed in production cloned environment (FLASH)
    - Executed by DBA

# Schema Change Process in Detail

- Details of job-identifier suffix (*also explains what's executed*):
  - $1 : Verbal directions on the entire process and building of each job
  - BO : Back out member
  - CN : SELECT * for count member
  - CR : Create DDL member
  - C1 : Performs image copy
  - D1 and D2 : Internal jobs to build DDL
  - L1 : CA Load job
  - NA and NB : : Use of CA Plan Analyzer to capture access paths before and after the rename; this is an extra review of access path changes
  - NM : Object Rename member
  - RB : Rebind member
  - SC : Review object to verify current standards are met
  - SP : Control card builder if needed for LOAD process
  - U1-U8 : CA unload members

# Schema Change Process in Detail

- More job details and how process is tested in the “FLASH” environment:

$DBMCP$1 – overall instructions to guide you through the entire process
$DBMCPBO - job to do Back Out
$DBMCPCN – performs COUNT(*) on old object
$DBMCPCR – capture packages that use current table, capture table authorities, perform DDL create of new object using Batch Processor; perform RUNSTATS on empty new Tablespace; put new TS in UT status
$DBMCPC1 – image copy job
$DBMCPD1 – internal job to build DDL on flash only
$DBMCPD2 – internal job to manipulate updated DDL on flash only
$DBMCPL1 - CA Load job
$DBMCPNA – perform review of packages on table before starting change using Neon product
$DBMCPNB – perform review of packages on table after change using Neon product
$DBMCPNM – Runstats new table after it is populated; statistics will be extracted to a flat file to use to populate other environments, views and view grants on original table are captured and dropped; current table and current indexes are renamed to old; new table and new indexes are renamed to current
$DBMCPRB - recreate views on the newly renamed table; reestablish grants; perform rebinds of packages and review for changes in access path
$DBMCPSC – basic review done of object in FLASH environment to ensure all standards are met
$DBMCPSP – builds control cards for load process if using DSNTIAUL
$DBMCPU1 – CA Unload job 1 using PIPES
$DBMCPU2 - CA Unload job 2 using PIPES
$DBMCPU3 - CA Unload job 3 using PIPES
$DBMCPU4 - CA Unload job 4 using PIPES
$DBMCPU5 – CA Unload job 5 using PIPES
$DBMCPU6 - CA Unload job 6 using PIPES
$DBMCPU7 - CA Unload job 7 using PIPES
$DBMCPU8 – CA Unload job 8 using PIPES

# Example of Unload Jobs using USS Pipes

- Up to 10 created accessing separate ranges of partitions
- U1 is the only difference between each unload job – illustrates the USS Pipe in use

```
//*************************************************************
//*       UNLOAD - FIRST JOB
//*************************************************************
//JS03     EXEC  PGM=PTLDRIVM,TIME=1439,PARM='EP=UTLGLCTL/DB2M',
//         REGION=0M
//CAISLIB   INCLUDE MEMBER=CADB2M
//PTIMSG    DD  SYSOUT=*
//SYSIN     DD  *,SYMBOLS=EXECSYS
 FASTUNLOAD
   OUTPUT-FORMAT  LOAD
   SQL-ACCESS     EXTENSION
   SHRLEVEL       CHANGE
   EMPTY-RC       0
   ZIIP           YES
   LRECL-USER     YES
   SELECT * FROM DB0PCBM0.CBMCFD00
   PART  1:13
   NEWOBID 1;
//CA11NR   DD DUMMY
//SYSPUNCH DD DUMMY
//SYSREC01 DD  PATH='/pipes/EF1271U1_SYSREC01',FILEDATA=BINARY,
//         PATHOPTS=(OCREAT,OWRONLY),PATHDISP=KEEP,DSNTYPE=PIPE,
//         PATHMODE=(SIRUSR,SIWUSR),
//         RECFM=FB,LRECL=112
//*
```

# The Load job receiving the 10 unload pipe datasets

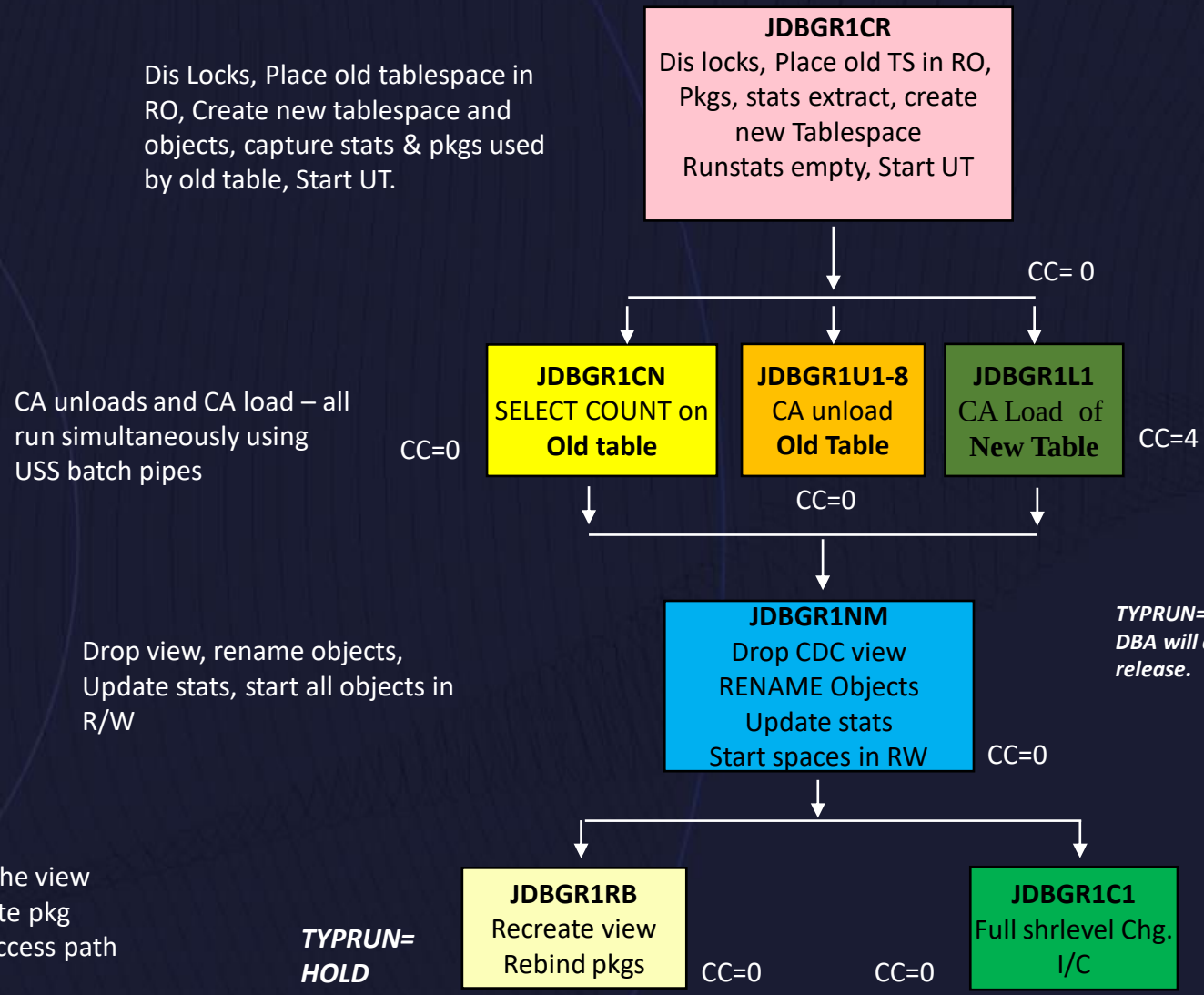
```
//*****************************************************************
//JS01   EXEC PGM=PTLDRIVM,PARM='EP=UTLGLCTL/DB2M',COND=(4,LT),
//       REGION=0M,TIME=1439
//CAISLIB  INCLUDE MEMBER=CADB2M
//$ORTPARM DD  DSN=PDBAPD.CNTL.LIB(SYNCSCAL),DISP=SHR
//SYSMAP   DD  DSN=DB2PASS.DB2M.L1.DB0PCBM0.CBMCF2.SYSMAP,
//       DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,
//       SPACE=(CYL,(001,150),RLSE)
//SYSERR   DD  DSN=DB2PASS.DB2M.L1.DB0PCBM0.CBMCF2.SYSERR,
//       DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,
//       SPACE=(CYL,(001,150),RLSE)
//SYSDISC  DD  DSN=DB2PASS.DB2M.L1.DB0PCBM0.CBMCF2.SYSDISC,
//       DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,
//       SPACE=(CYL,(001,150),RLSE)
//SYSUT1   DD  DSN=DB2PASS.DB2M.L1.DB0PCBM0.CBMCF2.SYSUT1,
//       DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,
//       SPACE=(CYL,(001,150),RLSE)
//SORTOUT  DD  DSN=DB2PASS.DB2M.L1.DB0PCBM0.CBMCF2.SORTOUT,
//       DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,
//       SPACE=(CYL,(001,150),RLSE)
//*    SYSMDUMP IS REQUIRED TO DIAGNOSE CA ISSUES.
//SYSMDUMP DD  DSN=DB2PASS.DB2M.L1.DB0PCBM0.CBMCF2.SYSMDUMP,
//       DISP=(NEW,DELETE,CATLG),UNIT=SYSDA,
//       SPACE=(CYL,(500,500),RLSE),
//       DCB=(RECFM=FBS,LRECL=4160,BLKSIZE=0)
//PTIMSG   DD   SYSOUT=*
//SYSUDUMP DD   DUMMY
//SYSABEND DD   DUMMY
//ABNLIGNR DD   DUMMY
```

```
//SYSIN    DD  *
FASTLOAD
 READER-BLOCKS   2400
 INPUT-FORMAT    UNLOAD       OUTPUT-CONTROL  BUILD
 ESTIMATED-INPUT 241665031    ENABLE-PFL20    YES
 NONLEAF-PCTFREE 0            IO-BUFFERS      250
 IXBUFFER-SIZE   1M           VSAM-BUFFERS    180
 REBUILD-INDEX   YES          STARTUP-ACCESS  FORCE
 MAXTASKS        AUTO         SORTNUM         AUTO
 SET-COPYPENDING NO
 SET-CHECKPENDING NO
 DISCARDS        100000
 DISPLAY-STATUS  5000000
 RESUME NO REPLACE KEEPDICTIONARY
 INTO TABLE
       DB0PCBM0.CBMXXX00
 OBID 1
//SYSULD   DD   PATH='/pipes/EF1271U1_SYSREC01',FILEDATA=BINARY,
//       PATHOPTS=(OCREAT,ORDONLY),PATHDISP=(KEEP),DSNTYPE=PIPE,
//       PATHMODE=(SIRUSR,SIWUSR),
//       RECFM=FB,LRECL=112
//SYSULD01 DD PATH='/pipes/EF1271U2_SYSREC01',FILEDATA=BINARY,
//       PATHOPTS=(OCREAT,ORDONLY),PATHDISP=(KEEP),DSNTYPE=PIPE,
//       PATHMODE=(SIRUSR,SIWUSR),
//       RECFM=FB,LRECL=112
//SYSULD02 DD PATH='/pipes/EF1271U3_SYSREC01',FILEDATA=BINARY,
//       PATHOPTS=(OCREAT,ORDONLY),PATHDISP=(KEEP),DSNTYPE=PIPE,
//       PATHMODE=(SIRUSR,SIWUSR),
//       RECFM=FB,LRECL=112
…… ETC ……..
//CAOESTOP DD DUMMY
```

**Ten SYSULD DD cards specifying the ten USS Pipes.**

**No CPU savings but significant elapsed savings due to early start of load.**

*Trigger from dummy job:* **JDBGR1ER**
*Job-dep:* **JDBGSWTW**
**( optional – add PAID release job)**
*Time:* ***HH:MM AM***






Version 2.0 (4/09/18)

*Data Refresh process from hours to minutes*

*Side effect / benefit from having the “flash / cloned” env.*

# Data Refresh Implementation

- Almost every Db2 site has the need – various methods used.
- Besides from data subsetting (different topic):

1. DSN1COPY to do IC-VSAM or VSAM-VSAM. Challenges when +2GB VSAM pagesets, unequal number of PBG's, extent processing etc.
2. Unload-Load consumes many resources, advantage is schema differences
3. Recover from production IC to target – with or without log-apply

- All three alternatives have the issue of data consistency
  - IC probably done as SHRLEVEL CHANGE and not the same RBA/LRSN
  - What about RI constraints

# Data Refresh Implementation

- Requirement to “clone” a few tables every night with up to 8 targets
  - Biggest table 100 partitions, 3 NPI’s and ~4bn rows
  - Used for batch, CICS and web-services (mostly claims processing)
- Old implementation
  - Unload and load took almost 3 hours elapsed and hours of CPU
  - Impossible to get all 8 environments available every night
  - Unavailability of data causing application outages
- Due to the availability of the “flash” environment created weekly this provides a nice “golden copy” to clone from

# Data Refresh implementation

- New implementation investigated using CA RC/Merger
  - Page-level data copy <> row level
  - Initial “dirty data cloning” illustrated huge savings
  - Data and Indexes cloned in minutes
  - Source Db2 VSAM pagesets “moved” – including:
    - PAGE RBA RESET, OBID translation, VERSION REPAIR etc.
- Currently production unload and target load
  - Used as “golden copy” for remaining environments using RC/Merger
- Next step being looked at (requires shared DASD between the environments) :
  - IBM COPY CONSISTENT YES FLASHCOPY YES
  - Special entry in SYSCOPY and IN-FLIGHT’s backed out by COPY utility
  - No dirty data, no need to rebuild indexes to avoid inconsistences

# Data Refresh implementation

- An even more optimized method does exist using RC/Merger
- In this case, data refresh into 8 environments and all in the same Db2 system
  - More savings by having TARGET ID's identical to SOURCE
    - Patented solution to RESERVE OBID's *(unless multiple targets in same SSID)*
    - Eliminates need to translate every row – just page-header needed
- Yet another even quicker solution considered was using MOVE as opposed to COPY
    - Source Pageset RENAMED to Target VSAM pageset (no need to copy data) – reminder that the Source data no longer exists.
    - ID translation might be needed as well as Repair

*Hopefully you enjoyed the content and maybe even some ideas*