

Copy Smarter

Unload/Load, DSN1COPY and beyond

Copy Smarter - Unload/Load, DSN1COPY and beyond

Overview

What are we trying to achieve?

- Create repeatable and reliable copy processes for DB2 data
- Handle deviations in table structures
- Create a scheduler friendly process
- Bonus points for speed

- Spoiler alert: It's hard.

Building blocks of a copy process

- Copying the data is a small part. The full process looks like this:

- DDL for tablespaces, tables, indexes
- Also views, triggers, constraints, etc.
- Optionally rename objects
- Allocate target objects with sufficient space

Structures

- Invoke copy programs to bring data from A to B

Data

- Copy catalog statistics and RTS
- Rebuild indexes if required
- Adjust versioning information, row format, RBA format
- Take care of identity columns, sequences
- Rebind

Cleanup

A quick look at DDL

- Needs to be handled, regardless of data copy mechanism
- Db2 for z/OS does not come with a full-fledged DDL generator, but has ADMIN_INFO_SQL (used by Data Studio)
- Db2 for LUW has db2look, which can work with Db2 for z/OS, but its output is always LUW syntax
- Db2 Admin Tool has ADB2GEN
- Home-grown solutions: REXX and ISPF file tailoring
- Renaming objects is harder than it sounds due to views, triggers
- Many vendor solutions available

Requirement: Reliability

- Process should not break when objects are created, changed, or dropped
- Detect new page sets that were added
- Detect and reset restricted states
- Restart after failure

Requirement: Being scheduler-friendly

- Fixed set of jobs
- Number of jobs does not change
- Contents of jobs do not change
- Can be executed repeatedly

Requirement: Speed

- Programs that copy Db2 data:
 - Unload/Load
 - DSN1COPY
 - ADRDSSU / FlashCopy2
 - Vendor solutions

What are our options to copy data?

- Every Db2 shop has Unload/Load (either from IBM or vendor)
- Every Db2 shop has DSN1COPY
- ADRDSSU always available, can trigger FlashCopy2

	Ease of use	Automation	Flexibility	Speed	Aware of Db2
Unload/Load	?	?	?	?	?
DSN1COPY	?	?	?	?	?
ADRDSSU / FlashCopy2	?	?	?	?	?
Vendor solutions	?	?	?	?	?

Copying the data with Unload/Load

- Easy to use, is often the go-to solution
- Db2 manages space for you
- Use LISTDEF and TEMPLATE to process many objects at once
- Changing SYSPUNCH may be tedious
 - Change table names
 - Change RESUME YES to RESUME NO REPLACE
 - Add OVERRIDE (SYSTEMPERIOD, IDENTITY, TRANSID, NONDETERMINISTIC), add ENFORCE NO
- Slow

Copying the data with Unload/Load

- DOs and DON'Ts:
 - Use SPANNED YES for LOB and XML data
 - Use IDXDEFER ALL with partition level LOAD, then rebuild indexes
 - Identify and skip empty partitions (this can save you hours)
 - Use the cross loader if possible
 - Do not use FORMAT INTERNAL – unreliable
 - Do not use partlevel LOAD if number of partitions or limit keys differ

Identity columns and sequences

- Must be adjusted in target
- Use `MAXASSIGNEDVAL + INCREMENT` as new value
- Sequence objects: Use `ALTER SEQUENCE RESTART WITH`
- Identity columns: Use `ALTER TABLE ALTER COLUMN RESTART WITH`
- Implicit XML sequences: Query repeatedly to increase value
 - Cannot be altered directly
 - `SQLCODE = -20142, ERROR: SEQUENCE CANNOT BE USED AS SPECIFIED`

Why do Unload/Load based copies fail?

- During Unload/Load:
 - Missing authorization
 - Missing or incomplete target objects
 - Incompatible target object (e.g., insufficient column length, wrong code page)
 - Insufficient work data sets for sort
 - Objects in use by other utility

Why do Unload/Load based copies fail?

- Post Unload/Load:
 - Incorrect sequences and identity columns
 - Inaccessible due to restricted states
 - Inconsistencies if data was unloaded with SHRLEVEL CHANGE

Transfer Unload files from/to non-z/OS

- Either use `FORMAT DELIMITED`, then FTP as text
 - Does not work well for binary data, LOB, XML
- Or use standard `LOAD` format, then FTP as binary
 - Either use: `QUOTE SITE RDW`
Each record is prefixed by a 4 byte field, first 2 bytes = length
 - Or use: `QUOTE STRU R`
`X'FF01'` = end of record, `X'FF02'` = end of file, `X'FF'` becomes `X'FFFF'`
- Properly transfer VBS data sets with binary data
 - Use: `QUOTE MODE B`, then `QUOTE TYPE E`
 - Use: `SITE LRECL=X RECFM=VBS BLOCKSIZE=27998`
 - On a PC, you will need a separate program to split/merge record fragments

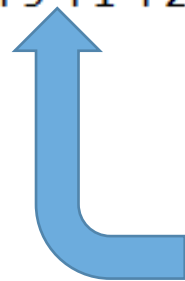
Transfer Unload files from/to non-z/OS

- Binary transfer with file structure: Record ends are lost

```
-----  
1.3  
F0F  
103  
-----  
1234...8  
FFFF00FF  
123412F8  
-----
```

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  
00000000 |F1 00 F3 F1 F2 F3 F4 01 02 FF F8
```

ñ.óñòóô..ÿø

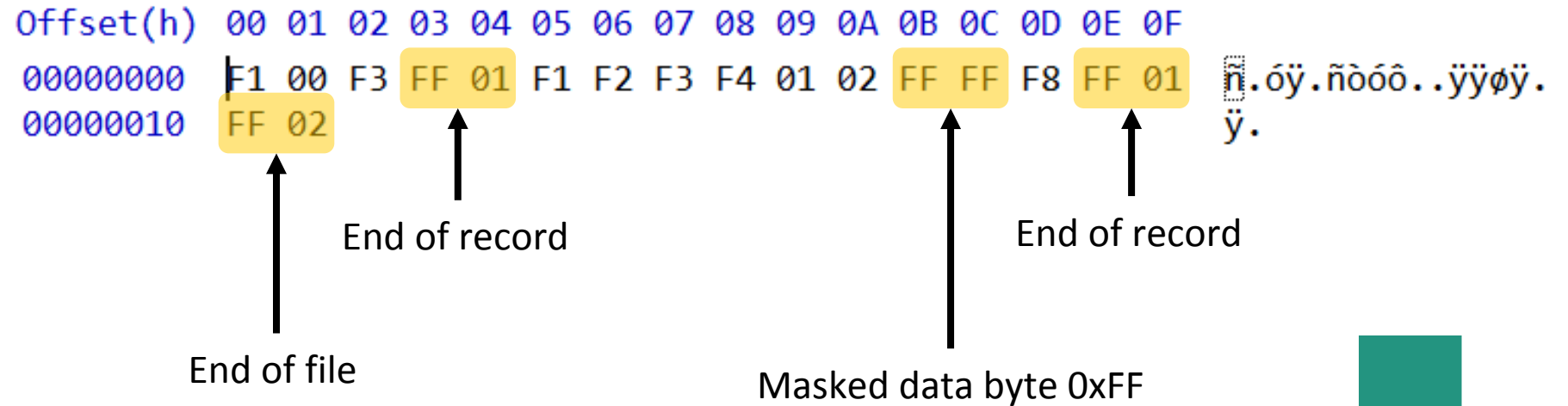


First record ends right here, but there is nothing that would indicate that

Transfer Unload files from/to non-z/OS

- Binary transfer with record structure

```
-----  
1.3  
F0F  
103  
-----  
1234...8  
FFFF00FF  
123412F8  
-----
```



What are our options to copy data?

- Every Db2 shop has Unload/Load (either from IBM or vendor)
- Every Db2 shop has DSN1COPY
- ADRDSSU always available, can trigger FlashCopy2

	Ease of use	Automation	Flexibility	Speed	Aware of Db2
Unload/Load	Good	Fair	Good	Bad	Yes
DSN1COPY	?	?	?	?	?
ADRDSSU / FlashCopy2	?	?	?	?	?
Vendor solutions	?	?	?	?	?

Copy the data with DSN1COPY

- Much faster than Unload/Load
- Works outside of Db2
- Need to allocate target VSAM clusters
- Need to check object compatibility **very thoroughly**
- Need to write / generate a lot of JCL with correct SYSXLAT members

Find all LOB tablespaces for a given table

```
SELECT
  STRIP (R.TBOWNER)      AS "BASE_TBcreator",
  STRIP (R.TBNAME)      AS "BASE_TBNAME",
  STRIP (R.COLNAME)     AS "BASE_COLNAME",
  R.PARTITION           AS "PARTITION",
  STRIP (S.DBNAME)      AS "LOB_DBNAME",
  STRIP (S.NAME)        AS "LOB_TSNAME",
  S.PGsize             AS "LOB_PGsize",
  S.DSSize             AS "LOB_DSSize",
  STRIP (R.AUXTBOWNER)  AS "AUX_TBcreator",
  STRIP (R.AUXTBNAME)   AS "AUX_TBNAME",
  STRIP (X.CREATOR)     AS "AUX_IXcreator",
  STRIP (X.NAME)        AS "AUX_IXNAME",
  X.PGsize             AS "AUX_IXPGsize",
  X.PIECESIZE          AS "AUX_IXPIECESIZE"
FROM
  SYSIBM.SYSAUXRELS R
INNER JOIN
  SYSIBM.SYSTABLES T
ON
  T.CREATOR = R.AUXTBOWNER AND
  T.NAME = R.AUXTBNAME
INNER JOIN
  SYSIBM.SYSTABLESPACE S
```

```
ON
  S.DBNAME = T.DBNAME AND
  S.NAME = T.TSNAME
INNER JOIN
  SYSIBM.SYSTABLEPART P
ON
  S.DBNAME = P.DBNAME AND
  S.NAME = P.TSNAME AND
  P.PARTITION IN (0 , 1)
INNER JOIN
  SYSIBM.SYSINDEXES X
ON
  X.TBcreator = R.AUXTBOWNER AND
  X.TBNAME = R.AUXTBNAME
INNER JOIN
  SYSIBM.SYSINDEXPART XP
ON
  XP.IXcreator = X.CREATOR AND
  XP.IXNAME = X.NAME AND
  XP.PARTITION IN (0 , 1)
WHERE
  R.TBOWNER = ? AND
  R.TBNAME = ?
FOR READ ONLY WITH UR
```

Find eligible image copy

```
SELECT
  DSNUM          AS IQDSNUM,
FROM
  SYSIBM.SYSCOPY
WHERE
  DBNAME = ? AND
  TSNAME = ? AND
  DSNUM IN (0, 1) AND
  ICTYPE = 'F' AND
  (
    STYPE IN (' ', 'R', 'S', 'W', 'X') OR
    (STYPE = 'T' AND DSNUM <> 0) OR
    (STYPE = 'T' AND DSNUM = 0
      AND LOWDSNUM = 1 AND HIGHDSNUM = 1)
  ) AND
  SHRLEVEL IN ('R', 'C')
ORDER BY
  TIMESTAMP DESC
FOR READ ONLY WITH UR
```

- IQDSNUM = 0: Tablespace level
- IQDSNUM > 0: Partition level
- Considers FlashCopy consistent image copies

Find eligible image copy

```
SELECT
  STRIP(C1.DSNAME) AS ICDSN,
  C1.TIMESTAMP AS ICTS,
  HEX(C1.START_RBA) AS ICRBA,
  C1.DEVTYPE AS ICDEVT,
  C1.DSVOLSER AS ICVOL,
  C1.ICUNIT AS ICUNIT,
  C1.FILESEQNO AS ICSEQNO,
  C1.NPAGESF AS ICNPAGES,
  C1.ICBACKUP AS ICBACKUP,
  C1.ICTYPE AS ICTYPE,
  C1.STYPE AS ICSTYPE
FROM
  SYSIBM.SYSCOPY C1
WHERE
  C1.DBNAME = ? AND
  C1.TSNAME = ? AND
  C1.ICTYPE = 'F' AND
  C1.DSNUM = ? AND
  C1.STYPE IN (' ', 'R', 'S', 'T', 'W', 'X') AND
  C1.SHRLEVEL IN ('R', ?)
AND (
  C1.DSNUM = 0
```

```
OR EXISTS (
  SELECT * FROM SYSIBM.SYSCOPY C2
  WHERE C2.START_RBA = C1.START_RBA
  AND C2.DSNUM = 1
)
)
AND (
  C1.DSNUM = 0
OR EXISTS (
  SELECT * FROM SYSIBM.SYSCOPY C3
  WHERE C3.START_RBA = C1.START_RBA
  AND C3.DSNUM = ?
)
)
)
ORDER BY
  C1.TIMESTAMP DESC
FOR READ ONLY
```

- Makes sure that partition level copies are only picked up if a copy was made for all partitions

Allocate target objects

- PBG tablespaces often problematic since Db2 can add partitions
 - Might create additional LOBs (and indexes), XML tablespaces (and indexes)
- Target PBG has fewer parts: Use ALTER TABLE ADD PARTITION
- Target PBG has too many parts:
 - Either: Drop and recreate
 - Or in V12: REORG with DROP_PART YES
 - Or in V11: REORG_DROP_PBG_PARTS = ENABLE
 - Neat trick: Empty all target partitions using LOAD REPLACE with empty SYSREC before the copy, then ignore extra partitions

Allocate target objects

- Non-partitioned tablespaces: Up to 32 VSAMs, use IDCAMS
- # of pieces based on TYPE, DSSIZE, PGSIZE, Numpagesf:

```
/* Get piece size (non-LOBs) or DSSIZE (LOBs) */
IF TYPE = "O" THEN DSSIZE_IN_KB = DSSIZE * 1024 * 1024
ELSE DSSIZE_IN_KB = 2 * 1024 * 1024
/* Correction for LOBs with DSSIZE 4 G */
IF DSSIZE_IN_KB = 4096 * 1024 THEN DSSIZE_IN_KB = 4095 * 1024
/* Calculate number of pieces */
SIZE_IN_KB = (NPAGESF * PGSIZE)
NUMPIECES = SIZE_IN_KB % DSSIZE_IN_KB
REMAINDER_IN_KB = SIZE_IN_KB // DSSIZE_IN_KB
IF REMAINDER_IN_KB > 0 THEN NUMPIECES = NUMPIECES + 1
```


Allocate target objects

- Popular choice `PRIQTY -1 SECQTY -1` causes problems with copy programs that work outside of Db2
- Inspect actual HI-U-RBA or use `SYSIBM.SYSCOPY.NPAGESF`
 - Non-partitioned TS or partition level copy of partitioned TS:
`NPAGESF * PGSIZE`
 - TS-level copy of range-partitioned TS (average size per partition):
`((NPAGESF * PGSIZE) + Numparts - 1) / Numparts`
 - Non-partitioned TS with n pieces or TS-level copy of PBG with n partitions:
Piece 1 to $(n-1)$: **`DSSIZE`**
Piece n : **`MOD (NPAGESF * PGSIZE, DSSIZE)`**

```
HI-A-RBA-----4294377472  
HI-U-RBA-----4293918720
```

Allocate target objects

Allocated: 4,294,377,472 Bytes = 4,095.4375 MB
Used: 4,293,918,720 Bytes = 4,095.0000 MB

- LOBs with DSSIZE 4 G only use 4095 MB
 - When non-EA: Final HI-A-RBA must be between 4095 and 4096 MB
 - Use MEGABYTES(94, 200) in your IDCAMS statement (YMMV)
- Pitfall: Partitioned objects with DSSIZE 4 G use 4096 MB
- **Best way:** Total size as PRIQTY, let SMS handle the details
 - This minimizes the number of extents (good performance)

```
DEFINE CLUSTER(CISZ(32768) REUSE LINEAR SHR(3 3) -  
  NAME(DSNC10.DSNDBC.BIGLOBDB.L1.I0001.A001)) -  
  DATA(NAME(DSNC10.DSNDBD.BIGLOBDB.L1.I0001.A001) -  
  KILOBYTES(4193280 419328))
```



SMS settings that make your life easier

```
Panel Utilities Scroll Help
DATA CLASS DISPLAY Page 1 of 5
Command ==> █

CDS Name . . . : ACTIVE
Data Class Name : MULVXDC

Description : DATA CLASS FOR EA-ENABLED MULTI VOLUME VSAM DATA SETS

Recfm . . . . . :
Lrecl . . . . . :
Override Space . . . . . : NO
Space Avgrec . . . . . :
  Avg Value . . . . . :
  Primary . . . . . :
  Secondary . . . . . :
  Directory . . . . . :
Retpd Or Expdt . . . . . :
Volume Count . . . . . : 1
Add'l Volume Amount . . . : SECONDARY

F1=Help   F2=Split  F3=End    F4=Return  F7=Up     F8=Down   F9=Swap
F10=Left  F11=Right F12=Cursor
```

Avoids unnecessary candidate volume entries in catalog

Simplifies space calculation



SMS settings that make your life easier

```
Panel Utilities Scroll Help
DATA CLASS DISPLAY Page 2 of 5
Command ==>

CDS Name . . . . . : ACTIVE
Data Class Name . . : MULVXDC

Data Set Name Type . . . . . : EXTENDED
  If Extended . . . . . : REQUIRED
  Extended Addressability . . : YES
  Record Access Bias . . . . . : USER
  RMODE31 . . . . . :

Space Constraint Relief . . . : YES
  Reduce Space Up To (%) . . . : 0
  Guaranteed Space Reduction . : NO
  Dynamic Volume Count . . . . : 10

Compaction . . . . . :
Spanned / Nonspanned . . . . . :
```

Increases chance of successful allocation

Avoids under-allocation

Allows SMS to add more volumes from the data set's storage group automatically

F1=Help F2=Split F3=End F4=Return F7=Up F8=Down F9=Swap
F10=Left F11=Right F12=Cursor



SMS settings that make your life easier

```
Panel Utilities Scroll Help
DATA CLASS DISPLAY Page 4 of 5
Command ==> █

CDS Name . . . . : ACTIVE
Data Class Name : MULVXDC

System Managed Buffer . . . . :
System Determined Blocksize : NO
Block Size Limit . . . . . :
EATTR . . . . . :
Recorg . . . . . : LS
Keylen . . . . . :
Keyoff . . . . . :
CIsze Data . . . . . :
% Freespace CI . . . . . :
                CA . . . . . :
Shareoptions Xregion . . . . :
                Xsystem . . . . :
```

F1=Help F2=Split F3=End F4=Return F7=Up F8=Down F9=Swap
F10=Left F11=Right F12=Cursor

Must be specified if Add'l
Volume Amount is set



SMS settings that make your life easier

```
Panel Utilities Scroll Help
DATA CLASS DISPLAY Page 5 of 5
Command ==> 
CDS Name . . . . : ACTIVE
Data Class Name : MULVXDC

Reuse . . . . . : NO
Initial Load . . . . . : RECOVERY
BW0 . . . . . :
Log . . . . . :
Logstream Id . . . . . :
FRlog . . . . . :
RLS CF Cache Value . . . . . : ALL
RLS Above the 2-GB Bar . . . . . : NO
Extent Constraint Removal . . . . . : YES
CA Reclaim . . . . . : YES
Log Replicate . . . . . : NO

F1=Help   F2=Split   F3=End     F4=Return  F7=Up      F8=Down    F9=Swap
F10=Left  F11=Right  F12=Cursor
```

Allows up to 7,257 extents per data set



SMS settings that make your life easier

```
Panel Utilities Scroll Help
STORAGE CLASS DISPLAY Page 1 of 2
Command ==>
CDS Name . . . . . : ACTIVE
Storage Class Name : DEFAULT
Description : DEFAULT STORAGE CLASS FOR SMS MANAGED DATA SETS
Performance Objectives
Direct Millisecond Response . . . . . :
Direct Bias . . . . . :
Sequential Millisecond Response . . . . . :
Sequential Bias . . . . . :
Initial Access Response Seconds . . . . . :
Sustained Data Rate (MB/sec) . . . . . :
OAM Sublevel . . . . . :
Availability . . . . . : NOPREF
Accessibility . . . . . : NOPREF
Backup . . . . . :
Versioning . . . . . :
F1=Help F2=Split F3=End F4=Return F7=Up F8=Down F9=Swap
F10=Left F11=Right F12=Cursor
```

Avoids multi stripe data sets
(multi stripe data sets cannot
use space constraint relief)



SMS settings that make your life easier

```
Panel Utilities Scroll Help
STORAGE CLASS DISPLAY Page 2 of 2
Command ==> 
CDS Name . . . . . : ACTIVE
Storage Class Name : DEFAULT
Guaranteed Space . . . . . : NO
Guaranteed Synchronous Write . . . : NO
Multi-Tiered SGs . . . . . :
Parallel Access Volume Capability : NOPREF
Cache Set Name . . . . . :
CF Direct Weight . . . . . :
CF Sequential Weight . . . . . :
Lock Set Name . . . . . :
Disconnect Sphere at CLOSE . . . . : NO

F1=Help   F2=Split   F3=End     F4=Return  F7=Up      F8=Down    F9=Swap
F10=Left  F11=Right  F12=Cursor
```

Avoids over-allocation (YES means primary space is allocated on all volumes)



Copying the data with DSN1COPY

- You need a script to generate required jobs (could be several thousand job steps)
- Not very scheduler friendly
 - Can be invoked from REXX in one single job step via
ADDRESS ATTCHMVS "DSN1COPY <parms>"
 - Requires dynamic allocation and error handling in REXX
- Read my rant at:

<http://ubs-hainer.com/solutions/bcv5/things-to-consider-when-using-dsn1copy-3>

Adjust version numbers, RBA / row format

- Extremely important after DSN1COPY based copy
- Failure to do so can lead to INCORROUT, ABEND S04E, S04F
- Db2 V10: Use REPAIR VERSIONS
- Db2 V11, V12: Use REPAIR CATALOG
- Problem:
 - Adjusting version numbers requires system pages
 - No system pages if tables have never been altered
 - This is changing (PI86880, UI51746)

Copying catalog statistics and RTS

- Catalog statistics are important for the Db2 optimizer
 - Dynamic SQL: Copy statistics
 - Static SQL: Copy statistics, rebind plan
- Rebind after updating catalog statistics
- Do not forget to rebind implicit trigger packages
 - Basic triggers: REBIND TRIGGER PACKAGE (*creator.name*)
 - Advanced triggers: Basic triggers: REBIND PACKAGE (*creator.name.**)
- RTS are important when UTSORTAL = YES

Rebuild indexes

- Use dynamic allocation of sort work data sets
 - Specify SORTDEVT, do not specify SORTNUM (or set IGNSORTN=YES)
 - Remove DFSORT related DDs from utility jobs
- Make sure to copy RTS for index first
- If RTS for index is unavailable:
 - Make sure you have good RTS for associated tablespace
 - REPAIR OBJECT SET INDEXSPACE (*dbname.spacename*) **RBDPEND**
 - Then rebuild index

Identity columns and sequences

- Must be adjusted in target
- Use `MAXASSIGNEDVAL + INCREMENT` value
- Sequence objects: Use `START WITH`
- Identity columns: Use `COLUMN RESTART`
- Implicit XML sequences: Query repeatedly to increase value
 - Cannot be altered directly
 - `SQLCODE = -20142, ERROR: SEQUENCE CANNOT BE USED AS SPECIFIED`

Same as for Unload/Load

Copying the data with DSN1COPY

- DOs and DON'Ts:
 - Pre-allocate all target VSAMs with the correct size
 - Check for restricted states in the source
 - Don't copy XML tablespaces into another Db2 subsystem
 - Don't copy from an object that has not been reorganized after the most recent ALTER TABLE or DROP TABLE
 - Don't copy partitioned tablespaces if partitions have been rotated, or if partitions have been inserted at any position other than the end
 - You'd think that „Relative Page Numbering“ helps, but it does not

Why do DSN1COPY based copies fail?

- During file system level copy:
 - Missing target page set
 - Cannot extend target page set or grow beyond 4 GB if non-EA
 - Remains of dropped tables in source causes OBID translation errors

Why do DSN1COPY based copies fail?

- Post file system level copy:
 - Did not include all source data sets
 - Incorrect sequences and identity columns
 - Did not take care of restricted states
 - Did not rebuild all target indexes
 - Incorrect OBID translation, log RBA, level ID
 - Did not do REORG before, REPAIR CATALOG after copy
 - Versioning problems / did not run REPAIR CATALOG in target
 - Copy was made despite structural incompatibilities

Fun with versions after APAR PI57004

- REPAIR CATALOG can have unexpected results:
 - Source tablespace and table are both version x, freshly reorged
 - Tablespaces and tables are 100% compatible
 - You run DSN1COPY, REPAIR CATALOG
 - You check the target catalog and it says version x+1
- Reason: **TIMESTAMP** or **DECIMAL** columns in the source table
 - **TIMESTAMP** columns problematic if source was created in V10 or older
 - **DECIMAL** columns problematic if source was created in V7 or older

My biggest problem with DSN1COPY

- There are situations where the copy process itself succeeds, and the target objects *look* OK, and *seem* to be accessible, but on occasion accessing the target tables will produce abend S04E with reason 00C90101 or similar reason codes.
- It's usually caused by missing REORGS / wrong SYSXLAT
- It can still be very hard to detect

What are our options to copy data?

- Every Db2 shop has Unload/Load (either from IBM or vendor)
- Every Db2 shop has DSN1COPY
- ADRDSSU always available, can trigger FlashCopy2

	Ease of use	Automation	Flexibility	Speed	Aware of Db2
Unload/Load	Good	Fair	Good	Bad	Yes
DSN1COPY	Bad	Bad	Bad	Good	Yes
ADRDSSU / FlashCopy2	?	?	?	?	?
Vendor solutions	?	?	?	?	?

What about ADRDSSU / FlashCopy2?

- NOT a good tool to copy one tablespace to another
- Does not translate DBID, PSID, OBIDs
 - Some people think they can use the REPAIR utility to fix DBID and PSID, but this is not always true.
- Does not reset log RBAs
- Does not set the PG1COPY flag bit, which is used by REPAIR CATALOG to trigger schema checking
- Read my other rant at:
<http://ubs-hainer.com/solutions/bcv5/copying-db2-objects-with-flashcopy>

After ADRDSSU, use REPAIR to fix Level ID, DBID, PSID, versions.

Table OBID is identical. We change the DBID from 012B to 012C, we change the PSID from 0004 to

```
DSNU050I      219 22:49:34.20 DSNUGUTC - REPAIR
DSNU650I -DBBG 219 22:49:34.21 DSNUCBLI - LEVELID TABLESPACE TVERSIDX.T1
DSNU683I -DBBG 219 22:49:34.57 DSNUCBRP - REPAIR LEVELID OPERATION SUCCESSFUL
DSNU050I      219 22:49:34.57 DSNUGUTC - REPAIR OBJECT
DSNU650I -DBBG 219 22:49:34.58 DSNUCBRL - LOCATE TABLESPACE TVERSIDX.T1 PAGE X'00'
DSNU650I -DBBG 219 22:49:34.86 DSNUCBRP - VERIFY OFFSET X'000C' DATA X'012B0004'
DSNU652I -DBBG 219 22:49:34.86 DSNUCBRR - VERIFY OPERATION SUCCESSFUL
DSNU650I -DBBG 219 22:49:34.86 DSNUCBRP - REPLACE OFFSET X'000C' DATA X'012C0002'
DSNU656I -DBBG 219 22:49:34.91 DSNUCBRR - REPLACE OPERATION SUCCESSFUL, DATA WAS X'012B0004'
DSNU050I      219 22:49:34.94 DSNUGUTC - REPAIR
DSNU650I -DBBG 219 22:49:34.95 DSNUCBVR - CATALOG TABLESPACE TVERSIDX.T1
DSNU675I -DBBG 219 22:49:35.27 DSNUCBVR - HIGH VERSION FOR DBID=X'012C' PSID=X'0002' IN THE
        Db2 CATALOG IS 0, BUT IN THE PAGE SET IS 1.
DSNU675I -DBBG 219 22:49:35.27 DSNUCBVR - LOW VERSION FOR DBID=X'012C' PSID=X'0002' IN THE
        Db2 CATALOG IS 0, BUT IN THE PAGE SET IS 1.
DSNU671I -DBBG 219 22:49:35.27 DSNUCBVR - DBID=X'012C' PSID=X'0002' OBID=X'07D0'
        TABLE VERSION IN THE CATALOG DOES NOT MATCH THE PAGE SET
DSNU695I -DBBG 219 22:49:35.29 DSNUCBVR - INFORMATION IN THE CATALOG WAS UPDATED
        TO MATCH THE PAGE SET
```

Then run REBUILD INDEX. It works, so the tablespace is OK, right?

```
LISTDEF L1 INCLUDE INDEXSPACES DATABASE TVERSIDX
LISTDEF STATEMENT PROCESSED SUCCESSFULLY
REBUILD INDEX LIST L1 SHRLEVEL REFERENCE
PROCESSING LIST ITEM: INDEXSPACE TVERSIDX.TVERSIDX
INDEXES WILL BE BUILT IN PARALLEL, NUMBER OF TASKS = 3
MAXIMUM INDEX PARALLELISM IS 3 BASED ON NUMBER OF INDEXES
DSNUCRUL - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS PROCESSED=100000
UNLOAD PHASE COMPLETE - ELAPSED TIME=00:00:00
SORT TASK SW01: 100000 RECORDS SORTED, ESTIMATED 0, VARIATION -1 PERCENT
SORT TASK SW01: USED DFSORT
SORT TASK SW01: MEMORY BELOW THE BAR: OPTIMAL 6 MB, USED 6 MB
DSNURBXC - SORTBLD PHASE STATISTICS - NUMBER OF KEYS=100000 FOR INDEX KAI.TVERSIDX_IX1
SORTBLD PHASE STATISTICS. NUMBER OF INDEXES = 1
SORTBLD PHASE COMPLETE, ELAPSED TIME = 00:00:00
MAXIMUM SORT AMOUNT ESTIMATION VARIATION WAS 0 PERCENT
TOTAL SORT MEMORY BELOW THE BAR: OPTIMAL 6 MB, USED 6 MB
UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Verify that the data is OK using SELECT *, all is good.

```
PAGE      1
***INPUT STATEMENT:
  SELECT * FROM KAI.SVERSIDX_TB1;
```

	COL01	COL02	COL03	COL04	COL05	COL06
1_	1	114921	624590	550815	939021	?
2_	2	724429	839504	684609	635107	181827
3_	3	552245	750931	245715	751530	724362
4_	4	612184	519980	182482	817834	464
5_	5	988725	654720	336550	924940	426401
6_	6	341380	?	118057	?	35152
7_	7	203656	182620	505645	698866	45088
8_	8	88569	214801	809647	318800	931594
9_	9	308779	77896	786255	919866	639709
10_	10	176410	847833	9513	749993	675077
11_	11	515483	139905	555669	683022	869535

[many more rows]

99997_	99997	857691	335066	729772	35849	22198
99998_	99998	46105	953504	617066	275336	766092
99999_	99999	361769	913569	?	819001	892942
100000_	100000	35631	989423	536785	986317	251987

SUCCESSFUL RETRIEVAL OF 100000 ROW(S)

Wrap things up by running RUNSTATS. Wait a second...

```
LISTDEF L1 INCLUDE TABLESPACES DATABASE TVERSIDX BASE
LISTDEF STATEMENT PROCESSED SUCCESSFULLY
LISTDEF L2 INCLUDE INDEXSPACES DATABASE TVERSIDX
LISTDEF STATEMENT PROCESSED SUCCESSFULLY
RUNSTATS TABLESPACE LIST L1 SHRLEVEL REFERENCE REPORT NO UPDATE ALL HISTORY ALL
TABLE (ALL)
PROCESSING LIST ITEM: TABLESPACE TVERSIDX.T1
UTILITY DATA BASE SERVICES MEMORY EXECUTION ABENDED, REASON=X'00C9021C'
```

00C9021C

While running a utility, the data manager detected an inconsistent data condition. A row was encountered that is not represented by a record OBD in the database descriptor (DBD). This abend may indicate an internal Db2® error, but most likely occurs due to a user error. Possible user errors may include:

- Data from a Db2 subsystem was copied to another Db2 subsystem incorrectly. This is the most common error.
- DSNDB01.DBD01 was regressed to a time prior to a table being created.



This Photo licensed under CC BY-SA

What happened?

```
Jobs Resources Devices Tools Filter View Options Help
DSN1PRNT JOB07063 < .DSN1PRNT.SYSPRINT>
Command ==>
Current Find Text:
-----1-----2-----3-----4-----5-----6-----7-----8-----9-----10-----11-----
PARTITION: # 0002
PAGE: # 00040000
HEADER PAGE: PGCOMB='10'X PGBIGRBA='00000000000000000000'X PGNUM='00040000'X PGFLAGS='38'X
HPGOBID='012B0004'X HPGHPREF='00042792'X HPGCATRL='00'X HPGREL='D7'X HPGZLD='D3'X
HPGCATV='00'X HPGTORBA='000000000000'X HPGTSTMP='20170721133537639837'X
HPGSSNM='DBBG' HPGFOID='0003'X HPGPGS7='4000'X HPGSGS7='0040'X HPGPARTN='000A'X
HPGZ3PNO='000000'X HPGZNUMP='10'X HPGTBLC='0001'X HPGROID='07D0'X
HPGZ4PNO='00040002'X HPGMAXL='0120'X HPGNUMCD='0009'X HPGFLGS='010C'X
HPGFLGS2='00'X HPGFLGS3='80'X HPGCONTM='20170721133806624835'X
HPGSGNAM='SYSDEFLT' HPGVCATN='DSNB10' HPGRBRBA='000000000000'X
HPGLEVEL='000000000000'X HPGPLEVL='000000000000'X HPGCLRSN='000000000000'X
HPGSCCSI='0417'X HPGDCCSI='0000'X HPGMCCSI='0000'X HPGPARTNUM='0000'X
HPGDSSZ='00400000'X HPGFLAG2='00'X HPGEPOCH='0001'X HPGRBLP='000000000000'X
HPGMASSELETETIMESTAMP='000000000000'X HPGDNUMB='01'X HPGDNUMC='0100'X
HPGDFSG='00000002'X HPGDLSG='00000002'X HPGSISP='00000000'X
HPGBIGTORBA='00000000000000000000'X HPGBIGRBRBA='00000000004CD235F77A'X
HPGBIGLEVEL='00000000004CD235F77A'X HPGBIGPLEVL='00000000004CD22A43A8'X
HPGBIGCLRSN='00000000004CD22B7BE7'X HPGBIGRBLP='00000000000000000000'X
HPGBIGMASSELETETIMESTAMP='00000000000000000000'X FOEND='52'X
DVI HASH BUCKET: HPGDBKT#='01'X HPG1BEYE='4E'X
F1=Help F3=Exit F5=Rfind F6=Info F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

DBID X'012B' PSID X'0004'
Table OBID X'07D0'



What happened?

```
Jobs Resources Devices Tools Filter View Options Help
DSN1PRNT JOB07063          <          .DSN1PRNT.SYSPRINT>
Command ==> █
Current Find Text:
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8-----+-----9-----+-----10-----+-----11-----
  HPGDCOLL#='0100'X  HPG10BID='07D0'X  HPG1V='01'X  HPG1RID='0004004201'X
SI HASH BUCKET:  HPGDBKT#='01'X  HPGS1BEYE='E2'X  HPGS10BI='07D0'X  HPGS1FSG='00000003'X
                HPGS1CSG='00000003'X  HPGS1LSG='0000009F'X

PAGE: # 00040001 -----
SEGMENTED SPACEMAP PAGE:  PGCOMB='00'X  PGBIGRBA='00000000000000000000'X  PGNUM='00040001'X
                        PGFLAGS='30'X  SEGNUM='015B'X  SEGFREE='00BC'X  SEGENT='00A0'X
                        SEGSIZE='0040'X  SEGLENT= 00002790'X  SEGBFMEM='00'X  FOEND='42'X

FIRST PART OF SEGMENTED SPACE MAP:
SEG 0001  00000000(0004)S  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
SEG 0002  0000000000001E0 F000000000000000000000000000000000000000000000000
SEG 0003  00000000407D0C0 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
SEG 0004  00000000507D080 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
SEG 0005  00000000607D080 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
SEG 0006  00000000707D080 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
SEG 0007  00000000807D080 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
SEG 0008  00000000907D080 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
SEG 0009  00000000A07D080 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
SEG 000A  00000000B07D080 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
F1=Help  F3=Exit  F5=Rfind  F6=Info  F7=Up  F8=Down  F9=Swap  F10=Left  F11=Right  F12=Cancel
```

The PSID appears on the space map page to identify the segment that contains the compression dictionary



Anything else?

- Space map pages and system pages contain PSID and OBIDs
- Number and locations of these pages varies (and so does the offset of the PSID/OBID fields on these pages)
- In the future, there will be system pages even if objects have never been altered
- Remember PI86880, UI51746?

What about ADRDSSU / FlashCopy2?

- DOs and DON'Ts:
 - Don't use ADRDSSU / FlashCopy2 to make a copy from one Db2 tablespace to another
 - No really, don't.

What are our options to copy data?

- Every Db2 shop has Unload/Load (either from IBM or vendor)
- Every Db2 shop has DSN1COPY
- ADRDSSU always available, can trigger FlashCopy2

	Ease of use	Automation	Flexibility	Speed	Aware of Db2
Unload/Load	Good	Fair	Good	Bad	Yes
DSN1COPY	Bad	Bad	Bad	Good	Yes
ADRDSSU / FlashCopy2	Bad	Bad	Bad	Good	No
Vendor solutions	?	?	?	?	?

Copy Smarter - Unload/Load, DSN1COPY and beyond

Conclusion



Conclusion

- Db2 itself does not provide a good mechanism to copy objects
- Problems mainly stem from:
 1. Missing tools for DDL generation
 2. Dependencies between Db2 catalog and contents of page sets
 3. Concept of version numbers after online schema changes
 4. Quirks of the native z/OS file system
- Unload/Load solves problems 2, 3, 4, but is too slow
- DSN1COPY lacks automation, is error prone

Conclusion

- Many Db2 shops simply use Unload/Load
- Some Db2 shops try to automate DSN1COPY
 - Works reasonably well for simple environments
 - Problems arise when newer Db2 features are exploited (table versioning, universal PBG tablespaces, partition rotation, clone tables, adding partitions in the middle of a tablespace, XML, etc.)
 - DSN1COPY may end with return code 0 even if the target is broken



Is there a better way?

- Vendor tools provide a degree of automation that is very hard to achieve manually
- UBS Hainer offers BCV5, which can do everything that was discussed today and more
- It combines unmatched flexibility with a very high copy speed
- BCV5 is easy to use, setting up a copy process takes mere minutes
- BCV5 is very scheduler friendly (fixed number of jobs, static JCL)
- BCV5 can also make consistent copies without stopping the source

What are our options to copy data?

- Every Db2 shop has Unload/Load (either from IBM or vendor)
- Every Db2 shop has DSN1COPY
- ADRDSSU always available, can trigger FlashCopy2

	Ease of use	Automation	Flexibility	Speed	Aware of Db2
Unload/Load	Good	Fair	Good	Bad	Yes
DSN1COPY	Bad	Bad	Bad	Good	Yes
ADRDSSU / FlashCopy2	Bad	Bad	Bad	Good	No
Vendor solutions	We certainly think so.				Yes

Questions or comments?



Thank you for your attention!

For more information visit
www.ubs-hainer.com
or send an email to
info@ubs-hainersoftware.com