Set up a copy process in less time than it takes to make a cup of coffee

Kai Stroh, UBS Hainer GmbH
Set up a copy process in less time than it takes to make a cup of coffee

Overview
Set up a copy process in less time than it takes to make a cup of coffee

What are we trying to achieve?

• Create a process to copy Db2 tables that is:
  • Flexible
  • Reliable
  • Comprehensive
  • Scheduler-friendly
  • Fast

• Spoiler alert: It’s hard.
Is it possible to -

Set up a copy process in less time than it takes to make a cup of coffee???

Kai Stroh, UBS Hainer GmbH
Set up a copy process in less time than it takes to make a cup of coffee

Building blocks of a copy process

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

<table>
<thead>
<tr>
<th>Structures</th>
<th>Data</th>
<th>Cleanup</th>
</tr>
</thead>
<tbody>
<tr>
<td>• DDL for tablespaces, tables, indexes</td>
<td>• Copy catalog statistics and RTS</td>
<td>• Adjust versioning information, row format, RBA format</td>
</tr>
<tr>
<td>• Also views, triggers, constraints, etc.</td>
<td>• Rebuild indexes if required</td>
<td>• Take care of identity columns, sequences</td>
</tr>
<tr>
<td>• Optionally rename objects</td>
<td>• Allocate target objects with sufficient space</td>
<td>• Rebind</td>
</tr>
<tr>
<td>• Allocate target objects with sufficient space</td>
<td>• Invoke copy programs to bring data from A to B</td>
<td></td>
</tr>
</tbody>
</table>
Set up a copy process in less time than it takes to make a cup of coffee

Flexible

• Copy within one Db2 subsystem
• Copy into another Db2 subsystem
• Copy between data sharing and standalone systems
• Copy between different Db2 versions
• Copy to a remote LPAR
• Copy from real page sets and image copies
Set up a copy process in less time than it takes to make a cup of coffee

Reliable

• Process should not break when objects are created, changed, or dropped
• Detect new page sets that were added
• Detect and reset restricted states
• Correctly allocate target VSAMs
  • Sometimes tricky due to 4 GB limit for non-EA page sets
• Restart after failure
Set up a copy process in less time than it takes to make a cup of coffee

Comprehensive

- Create missing target objects, work with existing target objects
- Handle identity columns
- Handle sequence objects (including implicit sequences)
- Possibility to rename objects
- Copy RUNSTATS information, RTS, SYSINDEXCLEANUP entries
- Rebind implicit packages after copying statistics
- Trigger utilities such as REPAIR CATALOG
Set up a copy process in less time than it takes to make a cup of coffee

Scheduler-friendly

• Fixed set of jobs
• Number of jobs does not change
• Contents of jobs do not change
• Can be executed repeatedly
Set up a copy process in less time than it takes to make a cup of coffee

**Fast**

- Programs that copy Db2 data:
  - Unload/Load
  - DSN1COPY
  - ADRDSSU / FlashCopy2
  - Vendor solutions
Set up a copy process in less time than it takes to make a cup of coffee

What are our options?

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

<table>
<thead>
<tr>
<th></th>
<th>Ease of use</th>
<th>Automation</th>
<th>Flexibility</th>
<th>Speed</th>
<th>Aware of Db2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unload/Load</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>DSN1COPY</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>ADRDSSU / FlashCopy2</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Set up a copy process in less time than it takes to make a cup of coffee

Why do copy processes 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
  • Source in use by other utility
Why do copy processes fail?

• During file system level copy:
  • Missing target page set
  • Cannot extend target page set or grow beyond 4 GB if non-EA
  • Incorrect OBID translation
Set up a copy process in less time than it takes to make a cup of coffee

Why do copy processes fail?

• Post Unload/Load:
  • Incorrect sequences and identity columns
  • Did not take care of restricted states

• Post file system level copy:
  • Incorrect OBID translation, log RBA, level ID
  • Did not do REORG before, REPAIR CATALOG after copy
  • Copy was made despite structural incompatibilities
Set up a copy process in less time than it takes to make a cup of coffee

How do we prevent failure?

• Automation.
Set up a copy process in less time than it takes to make a cup of coffee

Common challenges & how to tackle them
Set up a copy process in less time than it takes to make a cup of coffee

Generate DDL

• Db2 for z/OS does not come with a DDL generator
• 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
• Vendor solutions
• Renaming objects is harder than it sounds due to views, triggers
• Compatibility check for existing target objects?
Set up a copy process in less time than it takes to make a cup of coffee

Automating DDL generation

• DDL generation, execution should be part of the process
• Be flexible when online schema changes occur
• Option 1: Always drop and recreate target
• Option 2: Never drop and recreate target
• Option 3: Drop and recreate intelligently
Allocate target objects

• Partitioned, universal PBR, universal PBR2 are easy
• PBG tablespaces: Up to 4096 parts, use ALTER TABLE ADD PARTITION
  • Might create additional LOBs (and indexes), XML tablespaces (and indexes)
• Target PBG has too many partitions:
  • Drop and recreate
  • V12: REORG with DROP_PART YES
  • V11: REORG_DROP_PBG_PARTS = ENABLE
  • All versions: LOAD REPLACE with empty SYSIN, ignore extra partitions
Find all LOB tablespaces for a given table

```sql
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

- IQDSNUM = 0: Tablespace level
- IQDSNUM > 0: Partition level
- Considers FlashCopy consistent image copies
Set up a copy process in less time than it takes to make a cup of coffee

Find eligible image copy

- Avoids full copies of single partitions to be picked up
Allocate target objects

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

```c
/* 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
```
Set up a copy process in less time than it takes to make a cup of coffee

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
  - Partlevel copy of partitioned TS: 
    NPAGESF * PGSIZE
  - TS-level copy of partitioned TS: 
    \[((NPAGESF * PGSIZE) + NUMPARTS – 1) / NUMPARTS\]
  - Non-partitioned TS with \(n\) pieces:
    Piece 1 to \((n-1)\): DSSIZE
    Piece \(n\): MOD(NPAGESF * PGSIZE, DSSIZE)
Allocate target objects

- 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)

```sql
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))
```
Set up a copy process in less time than it takes to make a cup of coffee

Automate the data set allocation

• Stop all target objects
• Redefine all target VSAM clusters
• Use ISPF skeleton to generate SYSIN for IDCAMS
• Make sure SMS is configured appropriately

© UBS Hainer GmbH, the IBM Business Partner
Allocate target objects

Set up a copy process in less time than it takes to make a cup of coffee

Avoids unnecessary candidate volume entries in catalog
Simplifies space calculation
Allocate target objects

Set up a copy process in less time than it takes to make a cup of coffee

- Increases chance of successful allocation
- Avoids under-allocation
- Allows SMS to add more volumes from the data set's storage group automatically
Set up a copy process in less time than it takes to make a cup of coffee

Allocate target objects

<table>
<thead>
<tr>
<th>Command</th>
<th>CDS Name</th>
<th>Data Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>...</td>
<td>MULVXDC</td>
</tr>
</tbody>
</table>

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

**Must be specified if Add’l Volume Amount is set**
Allocate target objects

Command ==>  

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

Reuse . . . . . . : NO  
Initial Load . . . : RECOVERY  
BWO . . . . . . :  
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
Allocate target objects

- Set up a copy process in less time than it takes to make a cup of coffee

- Avoids multi stripe data sets
Allocate target objects

Avoids over-allocation
Set up a copy process in less time than it takes to make a cup of coffee

Copy the data with Unload/Load

• Easy to use
• 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)
• Slow

© UBS Hainer GmbH, the IBM Business Partner
Set up a copy process in less time than it takes to make a cup of coffee

Copying the data with Unload/Load

• DOs and DON’Ts:
  • Use SPANNED YES for LOB and XML data
  • Use dynamic allocation of sort work data sets for LOAD
  • 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
Set up a copy process in less time than it takes to make a cup of coffee

Copying the data with DSN1COPY

- Much faster than Unload/Load
- You MUST check compatibility of involved objects
- 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
  - Requires dynamic allocation and error handling in REXX
- Read my rant at:
Set up a copy process in less time than it takes to make a cup of coffee

Copying the data with DSN1COPY

• DOs and DON’Ts:
  • Do pre-allocate all target VSAMs with the correct size
  • Always run DSN1COPY with RESET, OBIDXLAT
  • 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

© UBS Hainer GmbH, the IBM Business Partner
Set up a copy process in less time than it takes to make a cup of coffee

What about ADRDSSU / FlashCopy2?

• NOT a good tool to copy Db2 data
• Does not translate DBID, PSID, OBIDs
  • If OBIDs match, REPAIR utility can be used to set DBID/PSID in header page
  • BUT: Segmented space map pages contain PSID as well
• 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:
Set up a copy process in less time than it takes to make a cup of coffee

After ADRDSSU, use REPAIR to fix Level ID, DBID, PSID, versions.
Then run REBUILD INDEX. It works, so the tablespace is OK, right?
Set up a copy process in less time than it takes to make a cup of coffee

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

---

### **INPUT STATEMENT:**

```
SELECT * FROM KAI.SVERSIDX_TB1;
```

---

<table>
<thead>
<tr>
<th>COL01</th>
<th>COL02</th>
<th>COL03</th>
<th>COL04</th>
<th>COL05</th>
<th>COL06</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>114921</td>
<td>624590</td>
<td>550815</td>
<td>939021</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>724429</td>
<td>839504</td>
<td>684609</td>
<td>635107</td>
<td>181827</td>
</tr>
<tr>
<td>3</td>
<td>552245</td>
<td>750931</td>
<td>245715</td>
<td>751530</td>
<td>724362</td>
</tr>
<tr>
<td>4</td>
<td>612184</td>
<td>519980</td>
<td>182482</td>
<td>817834</td>
<td>464</td>
</tr>
<tr>
<td>5</td>
<td>988725</td>
<td>654720</td>
<td>336550</td>
<td>924940</td>
<td>426401</td>
</tr>
<tr>
<td>6</td>
<td>341380</td>
<td>?</td>
<td>118057</td>
<td>?</td>
<td>35152</td>
</tr>
<tr>
<td>7</td>
<td>203656</td>
<td>182620</td>
<td>505645</td>
<td>698866</td>
<td>45088</td>
</tr>
<tr>
<td>8</td>
<td>88569</td>
<td>214801</td>
<td>809647</td>
<td>318800</td>
<td>931594</td>
</tr>
<tr>
<td>9</td>
<td>308779</td>
<td>77896</td>
<td>786255</td>
<td>919866</td>
<td>639709</td>
</tr>
<tr>
<td>10</td>
<td>176410</td>
<td>847833</td>
<td>9513</td>
<td>749993</td>
<td>675077</td>
</tr>
<tr>
<td>11</td>
<td>515483</td>
<td>139905</td>
<td>555699</td>
<td>683022</td>
<td>869535</td>
</tr>
</tbody>
</table>

---

SUCCESSFUL RETRIEVAL OF 100000 ROW(S)
SETUP A COPY PROCESS IN LESS TIME THAN IT TAKES TO MAKE A CUP OF COFFEE

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.
Set up a copy process in less time than it takes to make a cup of coffee

What happened?

© UBS Hainer GmbH, the IBM Business Partner
Set up a copy process in less time than it takes to make a cup of coffee

What happened?

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Resources</th>
<th>Devices</th>
<th>Tools</th>
<th>Filter</th>
<th>View</th>
<th>Options</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSN1PRNT JOB07063</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>=&gt;</td>
<td>.DSN1PRNT.SYSPRINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current Find Text:</td>
<td></td>
<td>1-2-3-4-5-6-7-8-9-10-11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APGDCOLLM='0100'X APG10D10='0700'X APG11='01'X APG1R1D='00000004201'X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SI HASH BUCKET: APGDBKTH='01'X APGS1BEYE='E2'X APGS1DBI='0700'X APGS1FSG='00000003'X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APGS1C5G='00000003'X APGS1LSG='00000009F'X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PAGE: # 00040001
SEGMENTED SPACEMAP PAGE: PGCOMB='00'X PBG1CRBA='00000000000000000000000000000000'X PGNUM='00000001'X PGFLAGS='30'X SEGNUM='0150'X SEGFREE='0000'C SEGLEN='0000'X SEGSIZE='0040'C SEGLEN='0000750'X SEGDFMEM='00'C FOEND='42'X

The PSID appears on the space map page to identify the segment that contains the compression dictionary. (It also appears in system pages.)
Set up a copy process in less time than it takes to make a cup of coffee

What about ADRDSSU / FlashCopy2?

• DOs and DON’Ts:
  • Don’t use ADRDSSU / FlashCopy2 to copy Db2 objects
  • No really, don’t.
Set up a copy process in less time than it takes to make a cup of coffee

What are our options?

- Every Db2 shop has Unload/Load (either from IBM or vendor)
- EveryDb2 shop has DSN1COPY
- ADRDSSU always available, can trigger FlashCopy2

<table>
<thead>
<tr>
<th></th>
<th>Ease of use</th>
<th>Automation</th>
<th>Flexibility</th>
<th>Speed</th>
<th>Aware of Db2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unload/Load</td>
<td>Good</td>
<td>Fair</td>
<td>Good</td>
<td>Bad</td>
<td>Yes</td>
</tr>
<tr>
<td>DSN1COPY</td>
<td>Bad</td>
<td>Bad</td>
<td>Bad</td>
<td>Good</td>
<td>Yes</td>
</tr>
<tr>
<td>ADRDSSU / FlashCopy2</td>
<td>Bad</td>
<td>Bad</td>
<td>Bad</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>

© UBS Hainer GmbH, the IBM Business Partner
Set up a copy process in less time than it takes to make a cup of coffee

Copy 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

© UBS Hainer GmbH, the IBM Business Partner
Set up a copy process in less time than it takes to make a cup of coffee

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.spacenam) RBDPEND
  • Then rebuild index
Set up a copy process in less time than it takes to make a cup of coffee

Adjust version numbers, RBA / row format

• Extremely important after data set level 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 going to change (look for the following APARs: PI76461, PI76462, PI75145, PI76179, PI78780, PI80006, PI81005)
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
Set up a copy process in less time than it takes to make a cup of coffee

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
Set up a copy process in less time than it takes to make a cup of coffee

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
Set up a copy process in less time than it takes to make a cup of coffee

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
Set up a copy process in less time than it takes to make a cup of coffee

Questions or comments?
Set up a copy process in less time than it takes to make a cup of coffee

Thank you for your attention!

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

© UBS Hainer GmbH, the IBM Business Partner