Loading Large Tables

Rob Gould
September 2026

Db2 – Loading Large Tables

- **Background**
- 1st Tests – NPIs or no NPIs
- Splitting the load file
- 1st Prod Run
- Bang - Splitting DfSort
- Inline Stats & Copies
- Adding / Amending Columns
- Adding Indexes

Background

- As companies move to be more ‘Cloudy’ extracting data from Db2 z/OS to be moved to the cloud becomes more of a requirement.
- We have used this approach on a number of Migrations to Cloud
- In this instance the target was Db2 on z/OS.
- The same principals apply to cloud based databases and to event streaming platforms, Kafka etc.

Background

- New z/OS Db2 table data built from extracts of 12 z/OS Db2 tables across Db2 Data sharing Groups.
- Load file created using batch suite of unload & DFSORT jobs, 70+ jobs in total.
- Would recommend using z/OS Job groups
- Initially 1.5Bn Rows now 1.7Bn
- 43 Columns
- 256 Partition – Partition by Range Tablespace
- 1 Partitioned index, 4 NPIs initially
- 6 Hour Change slot
- 2 Hrs outage for Table load

Background

- Not allowed to lock production tables so data taken straight from the underlying Db2 datasets.
- This gives us the usual problems with unloading a moving target.
- CDC driven process to apply changes after Load complete to get data to reflect tables at time of unload.
- CDC in doubt period to cover time taken for unload & load.
- CDC process then used to keep data current.

z/OS Jobgroups

- Enables you to create Batch applications outside of TWS
- Very useful

```
//ROBJOBG1 JOBGROUP
//ROB400J0 GJOB
//ROB401J0 GJOB
//       AFTER NAME=ROB400J0,
//        WHEN=(RC=0)
//ROB402J0 GJOB
//       AFTER NAME=ROB401J0,
//        WHEN=(RC=0)
//ROB403J0 GJOB
//       AFTER NAME=ROB402J0,
//        WHEN=(RC=0)
//ROB404J0 GJOB
//       AFTER NAME=ROB403J0,
//        WHEN=(RC=0)
//ROBJOBG1 ENDGROUP
//*
//*
//ROB400P0 JOB 'SOME JOB OR OTHER',REGION=256M,
//          CLASS=C,MSGCLASS=M,NOTIFY=&SYSUID
//*
//  SCHEDULE JOBGROUP=ROBJOBG1
```

Initial Plan

Hr	
1	Create Data files
2	
3	Load Table
4	
5	Contingency
6	

Db2 – Loading Large Tables

- Background
- **1st Tests – NPIs or no NPIs**
- Splitting the load file
- 1st Prod Run
- Bang - Splitting DfSort
- Inline Stats & Copies
- Adding Columns
- Adding Indexes

1st Tests

- Straight ‘Load replace Log No’
- Using IBM Load
- One 1.5Bn Rows load file
- 1 Partitioned Index, 4 NPIs
- Ran for 9 hours and then we gave up

- Dropped the NPIs
- Ran for 2 ¾ hours – better but not great as still have to rebuild the NPIs, Take IC, Runstats etc.
- Very obvious without NPIs was the way to go.

- Test systems problematic, either really busy or completely empty so difficult to guess what would happen on a loaded Prod system.

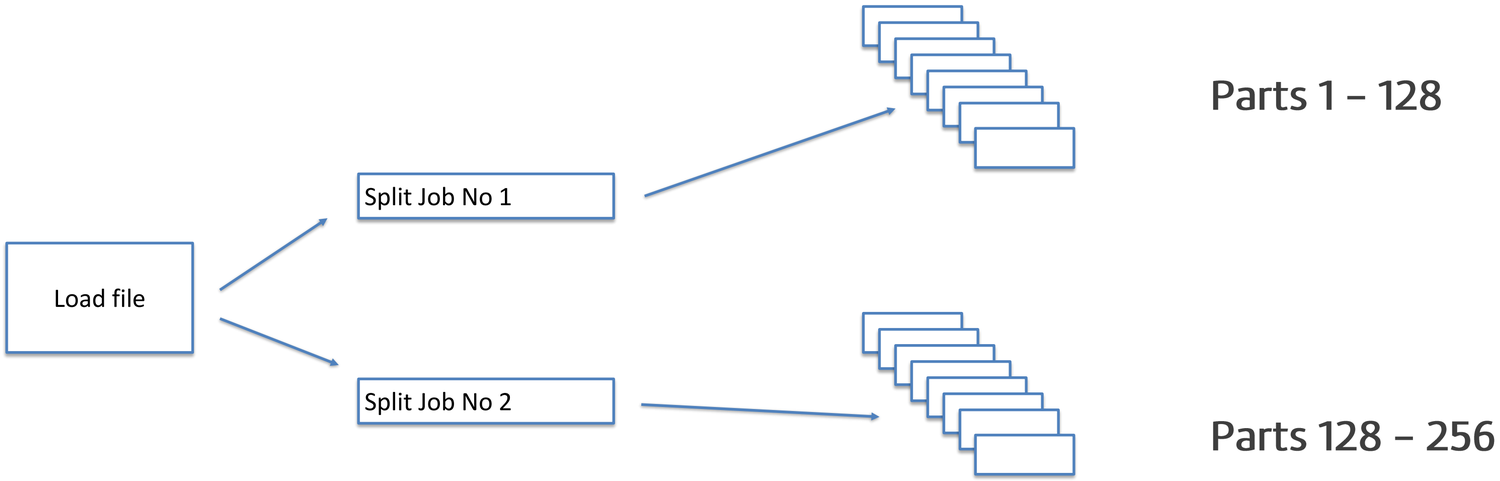
Db2 – Loading Large Tables

- Background
- 1st Tests – NPIs or no NPIs
- **Splitting the load file**
- 1st Prod Run
- Bang - Splitting DfSort
- Inline Stats & Copies
- Adding / Adding Columns
- Adding Indexes

Splitting the Load file –

Only viable solution was loading partitions in parallel.

- Had to add two DFSORT Jobs each extracting 128 Partitions into separate files.
- Running in Parallel – just splitting on Partition Key
- Jobs Run for an hour so added an hour to load file preparation.



Splitting the Load File

```
//*-------------------------------------------------------
------
//* DFSORT – ICEMAN Bin Sift ...
//*-------------------------------------------------------
------
//BINSIFT EXEC PGM=ICEMAN
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DISP=SHR,DCB=(BUFNO=60),DSN=&INNPUT
//OUT001   DD
DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//       DCB=(DSORG=PS,RECFM=FB,LRECL=&RECL,BLKSIZE=0,BUFNO=60
),
//       SPACE=(CYL,(100,50),RLSE),DSN=&DSNPFX..P001
//OUT002   DD
DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//       DCB=(DSORG=PS,RECFM=FB,LRECL=&RECL,BLKSIZE=0,BUFNO=60
),
//       SPACE=(CYL,(100,50),RLSE),DSN=&DSNPFX..P002
//OUT003   DD
DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//       DCB=(DSORG=PS,RECFM=FB,LRECL=&RECL,BLKSIZE=0,BUFNO=60
),
//       SPACE=(CYL,(100,50),RLSE),DSN=&DSNPFX..P003
//OUT004   DD
DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//       DCB=(DSORG=PS,RECFM=FB,LRECL=&RECL,BLKSIZE=0,BUFNO=60
),
//       SPACE=(CYL,(100,50),RLSE),DSN=&DSNPFX..P004
//OUT005   DD
```

```
* Split input by key
  SORT FIELDS=COPY
*
  OUTFIL FNAMES=(OUT001),
    INCLUDE=(127,10,CH,LE,C'AABB1111  ')
  OUTFIL FNAMES=(OUT002),
   INCLUDE=(127,10,CH,GT,C'AABB1111  ',AND,127,10,CH,LE,C'ABFF1111  ')
  OUTFIL FNAMES=(OUT003),
   INCLUDE=(127,10,CH,GT,C'ABFF1111  ',AND,127,10,CH,LE,C'AFDD1111  ')
  OUTFIL FNAMES=(OUT004),
   INCLUDE=(127,10,CH,GT,C'AFDD1111  ',AND,127,10,CH,LE,C'AKBB1111  ')
  OUTFIL FNAMES=(OUT005),
   INCLUDE=(127,10,CH,GT,C'AKBB1111  ',AND,127,10,CH,LE,C'APNN1111  ')
  OUTFIL FNAMES=(OUT006),
   INCLUDE=(127,10,CH,GT,C'APNN1111  ',AND,127,10,CH,LE,C'ARLL1111  ')
  OUTFIL FNAMES=(OUT007),
   INCLUDE=(127,10,CH,GT,C'ARLL1111  ',AND,127,10,CH,LE,C'AUBB1111  ')
```

Db2 – Loading Large Tables

- Background
- 1st Tests – NPIs or no NPIs
- Splitting the load file
- **1st Prod Run**
- Bang - Splitting DfSort
- Inline Stats & Copies
- Adding Columns
- Adding Indexes

Plan with Splits

Hr	
1	Create & Split Data files
2	
3	
4	Load Table Reorg Partitions, backup & runstats
5	Recreate indexes Rebuild indexes, backup & runstats
6	Contingency – Not a lot

Load Job

```
//*************************************************************
//* LOAD RESUME PART n REPLACE Large table
//*************************************************************
//SLSPNLR EXEC PGM=DSNUTILB,
//         PARM='&SSID,&UTXID,&UTPROC'
// INCLUDE MEMBER=DB2STUFF
//SYSPRINT DD  SYSOUT=*,OUTLIM=0
//SYSUDUMP DD  DUMMY
//UTPRINT  DD  SYSOUT=*                    Important
//*
//INDD001  DD  DISP=SHR,DSN=ROBSHLQ.DB2.ROBDB2J1.LOADDATA.P0
11
//INDD002  DD  DISP=SHR,DSN=ROBSHLQ.DB2.ROBDB2J1.LOADDATA.P
012
//INDD003  DD  DISP=SHR,DSN=ROBSHLQ.DB2.ROBDB2J1.LOADDATA.P
013
//INDD004  DD  DISP=SHR,DSN=ROBSHLQ.DB2.ROBDB2J1.LOADDATA.P
014
//INDD005  DD  DISP=SHR,DSN=ROBSHLQ.DB2.ROBDB2J1.LOADDATA.P
015
//INDD006  DD  DISP=SHR,DSN=ROBSHLQ.DB2.ROBDB2J1.LOADDATA.P
016
//INDD007  DD  DISP=SHR,DSN=ROBSHLQ.DB2.ROBDB2J1.LOADDATA.P0
17
//INDD008  DD  DISP=SHR,DSN=ROBSHLQ.DB2.ROBDB2J1.LOADDATA.P
018
//INDD009  DD  DISP=SHR,DSN=ROBSHLQ.DB2.ROBDB2J1.LOADDATA.P
019
//INDD010  DD  DISP=SHR,DSN=ROBSHLQ.DB2.ROBDB2J1.LOADDATA.P0
20
//*
//*
//DFSPARM  DD *
//*  OPTION
DYNALLOC(3390,32),MAINSIZE=MAX,RESINV=16K,DYNAPCT=25
//*
/*
//*
//SYSUT1   DD  UNIT=3390,STORCLAS=LARGE,DATACLAS=GURTBIG,
//         SPACE=(CYL,(500,500),RLSE)
//SORTOUT  DD  UNIT=3390,STORCLAS=LARGE,DATACLAS=GURTBIG,
//         SPACE=(CYL,(500,500),RLSE)
//SYSMAP   DD  UNIT=3390,STORCLAS=LARGE,DATACLAS=GURTBIG,
//         SPACE=(CYL,(500,500),RLSE)
//SYSERR   DD  UNIT=3390,STORCLAS=LARGE,DATACLAS=GURTBIG,
//         SPACE=(CYL,(500,500),RLSE)
//*
```

```
//SYSIN    DD  *,SYMBOLS=EXECSYS
--
  OPTIONS PREVIEW
--
   TEMPLATE WORK
     DSN(ROBSHLQ.WK.&DB..&TS..P&PART..V0)
     DISP(NEW,CATLG,DELETE)
     UNIT=3390
     DATACLAS GURTBIG
     BUFNO=60
   TEMPLATE ICOPY1
     DSN 'ROBSHLQ.&DB..&TS..P&PART..IC(+1)'
     DISP(NEW,CATLG,DELETE)
     UNIT=3390
     DATACLAS GURTBIG
     BUFNO=60
   TEMPLATE SREC
     DSN(ROBSHLQ.SR.&DB..&TS..P&PART..V0)
     DISP(NEW,CATLG,DELETE)
     UNIT=3390
     DATACLAS GURTBIG
     BUFNO=60
  TEMPLATE SXSDISC
    DSN=('ROBSHLQ.DB2.T0S3083P.SYSDIS2.P&PART.')
    DISP(MOD,CATLG,CATLG)
    UNIT=3390
    SPACE (160,50) CYL
    DATACLAS GURTBIG
--
  OPTIONS EVENT(ITEMERROR,SKIP)
--
```

```
LOAD DATA          SHRLEVEL NONE LOG NO NOCOPYPEND
          PARALLEL(0) INDEXDEFER NPI SORTKEYS SORTDEVT 3390
     INTO TABLE STC6027.T0S3083P  IGNOREFIELDS YES
      PART 011 REPLACE DISCARDDN SXSDISC INDDN(INDD001) REUSE
//       DD  DISP=SHR,DSN=ROBSHLQ.DB2.DATAC(LOADCARD)
//       DD  *,SYMBOLS=EXECSYS
    INTO TABLE STC6027.T0S3083P  IGNOREFIELDS YES
      PART 012 REPLACE DISCARDDN SXSDISC INDDN(INDD002) REUSE
//       DD  DISP=SHR,DSN=ROBSHLQ.DB2.DATAC(LOADCARD)
//       DD  *,SYMBOLS=EXECSYS
    INTO TABLE STC6027.T0S3083P  IGNOREFIELDS YES
      PART 013 REPLACE DISCARDDN SXSDISC INDDN(INDD003) REUSE
//       DD  DISP=SHR,DSN=ROBSHLQ.DB2.DATAC(LOADCARD)
//       DD  *,SYMBOLS=EXECSYS
    INTO TABLE STC6027.T0S3083P  IGNOREFIELDS YES
      PART 014 REPLACE DISCARDDN SXSDISC INDDN(INDD004) REUSE
//       DD  DISP=SHR,DSN=ROBSHLQ.DB2.DATAC(LOADCARD)
//       DD  *,SYMBOLS=EXECSYS
    INTO TABLE STC6027.T0S3083P  IGNOREFIELDS YES
      PART 015 REPLACE DISCARDDN SXSDISC INDDN(INDD005) REUSE
//       DD  DISP=SHR,DSN=ROBSHLQ.DB2.DATAC(LOADCARD)
//       DD  *,SYMBOLS=EXECSYS
    INTO TABLE STC6027.T0S3083P  IGNOREFIELDS YES
      PART 016 REPLACE DISCARDDN SXSDISC INDDN(INDD006) REUSE
//       DD  DISP=SHR,DSN=ROBSHLQ.DB2.DATAC(LOADCARD)
//       DD  *,SYMBOLS=EXECSYS
    INTO TABLE STC6027.T0S3083P  IGNOREFIELDS YES
      PART 017 REPLACE DISCARDDN SXSDISC INDDN(INDD007) REUSE
//       DD  DISP=SHR,DSN=ROBSHLQ.DB2.DATAC(LOADCARD)
//       DD  *,SYMBOLS=EXECSYS
    INTO TABLE STC6027.T0S3083P  IGNOREFIELDS YES
      PART 018 REPLACE DISCARDDN SXSDISC INDDN(INDD008) REUSE
//       DD  DISP=SHR,DSN=ROBSHLQ.DB2.DATAC(LOADCARD)
//       DD  *,SYMBOLS=EXECSYS
    INTO TABLE STC6027.T0S3083P  IGNOREFIELDS YES
      PART 019 REPLACE DISCARDDN SXSDISC INDDN(INDD009) REUSE
//       DD  DISP=SHR,DSN=ROBSHLQ.DB2.DATAC(LOADCARD)
//       DD  *,SYMBOLS=EXECSYS
    INTO TABLE STC6027.T0S3083P  IGNOREFIELDS YES
      PART 020 REPLACE DISCARDDN SXSDISC INDDN(INDD010) REUSE
//       DD  DISP=SHR,DSN=ROBSHLQ.DB2.DATAC(LOADCARD)
//*
```

Db2 – Loading Large Tables

Step	Description	Run time
• Drop & recreate Objects	• Wanted to remove -1 default for all PRIQTYs so auto space sizing wouldn't be needed. • Dropped all NPIs	1 Min
• Load 1.5Bn rows	• 26 Load jobs, each loading 10 Partitions. • We ran 5 in parallel on one Member of the datasharing group.	13 ½ Mins
• Reorg Tablespace, • Runstats table and indexes • Image Copy both.	• 10 Jobs each doing 25 Partitions. • Ran 3 at a time.	35 Mins
• Rebuild & IC 4 NPI Indexes	• One for each NPI running in parallel	13 Mins
	Total Time	**62 Mins**

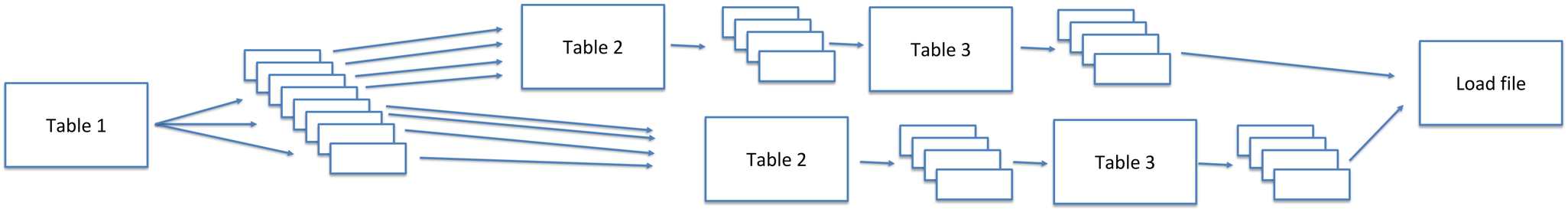
Db2 – Loading Large Tables

- Background
- 1st Tests – NPIs or no NPIs
- Splitting the load file
- 1st Prod Run
- **Splitting DfSort**
- Inline Stats & Copies
- Adding Columns
- Adding Indexes

Bang – Splitting DFSort

- 1 Job matching and merging two 1 Billion row plus files on a many to many merge
  - Blew the DFSort limit of 4Bn records – who knew!
  - Only time we failed to get the table loaded
- Stuck split jobs in to put Table unload into 10 Files split by key range.
- Then ran 5 jobs at a time
- Reduced end to end of this job from over an hour to 40 Minutes.

Splitting Unload files



- Merge back into one file enabled us to sort file into Primary Key order thereby eliminating the need for the Reorgs

Splitting Unload files

```
//SORTIN   DD  DISP=SHR,DSN=&BIGFILE
//SORTOUT  DD  DISP=(NEW,CATLG,CATLG),UNIT=3390,
//           DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=0),
//           SPACE=(TRK,(1,1),RLSE),DSN=&CMDFIL
//SYSIN    DD  *,SYMBOLS=EXECSYS
  SORT FIELDS=COPY
  OUTREC IFTHEN=(WHEN=INIT,
              BUILD=(C'SPLIT1R=',2,15,ZD,DIV,&PARTS,80:X)),
         IFTHEN=(WHEN=INIT,
              BUILD=(2X,1,78,SQZ=(SHIFT=LEFT)))
/*
```

OUTPUT is ‘SPLIT1R=238186763’

Splitting Unload files

```
//*****************************************************************
//* COPY01 – DFSort ICEMAN COPY with OUTFIL BUILD SPLIT1R x 10
//*****************************************************************
//SCOPY01 EXEC PGM=ICEMAN
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SORTIN   DD  DISP=SHR,DSN=&F2INN
//*SORTOUT  DD  SYSOUT=*          Default SYSREC output
//*
//* OUTFIL Split–Files :
//* ====================
//SF01    DD  DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//          DCB=(DSORG=PS,RECFM=FB,LRECL=&F3REC,BLKSIZE=0),
//          SPACE=(CYL,(500,500),RLSE),DSN=&F3OUT..P01
//SF02    DD  DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//          DCB=(DSORG=PS,RECFM=FB,LRECL=&F3REC,BLKSIZE=0),
//          SPACE=(CYL,(500,500),RLSE),DSN=&F3OUT..P02
//SF03    DD  DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//          DCB=(DSORG=PS,RECFM=FB,LRECL=&F3REC,BLKSIZE=0),
//          SPACE=(CYL,(500,500),RLSE),DSN=&F3OUT..P03
//SF04    DD  DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//          DCB=(DSORG=PS,RECFM=FB,LRECL=&F3REC,BLKSIZE=0),
//          SPACE=(CYL,(500,500),RLSE),DSN=&F3OUT..P04
//SF05    DD  DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//          DCB=(DSORG=PS,RECFM=FB,LRECL=&F3REC,BLKSIZE=0),
//          SPACE=(CYL,(500,500),RLSE),DSN=&F3OUT..P05
//SF06    DD  DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//          DCB=(DSORG=PS,RECFM=FB,LRECL=&F3REC,BLKSIZE=0),
//          SPACE=(CYL,(500,500),RLSE),DSN=&F3OUT..P06
//SF07    DD  DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//          DCB=(DSORG=PS,RECFM=FB,LRECL=&F3REC,BLKSIZE=0),
//          SPACE=(CYL,(500,500),RLSE),DSN=&F3OUT..P07
//SF08    DD  DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//          DCB=(DSORG=PS,RECFM=FB,LRECL=&F3REC,BLKSIZE=0),
//          SPACE=(CYL,(500,500),RLSE),DSN=&F3OUT..P08
//SF09    DD  DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//          DCB=(DSORG=PS,RECFM=FB,LRECL=&F3REC,BLKSIZE=0),
//          SPACE=(CYL,(500,500),RLSE),DSN=&F3OUT..P09
//SF10    DD  DISP=(NEW,CATLG,CATLG),UNIT=3390,DATACLAS=GURTBIG,
//          DCB=(DSORG=PS,RECFM=FB,LRECL=&F3REC,BLKSIZE=0),
//          SPACE=(CYL,(500,500),RLSE),DSN=&F3OUT..P10
//*
//SYSIN    DD  *,SYMBOLS=EXECSYS
* Control statements for the MAIN task ...
 SORT FIELDS=COPY
*
* OUTFIL BUILD with SPLIT1R parameter from IPX115PA job ...
* Cannot use HEADER1 with SPLIT1R ...
* Record is External FB
 OUTFIL REMOVECC,
      FNAMES=(SF01,SF02,SF03,SF04,SF05,
           SF06,SF07,SF08,SF09,SF10),
      BUILD=(1,214),
//       DD  DISP=SHR,DSN=&SPLTFIL
//*
//
```

Db2 – Loading Large Tables

Step	Description	Run time
• Stop Application		
• Backup Tablespace & Indexes		3 Mins
• Drop NPIs	Dropped all NPIs	1 Min
• Load in Primary Key Order • 1.6Bn rows	• 26 Load jobs, each loading 10 Partitions. • We ran 5 in parallel on one Member of the datasharing group.	15 ½ Mins
• Runstats table and indexes • Image Copy both.	• 10 Jobs each doing 25 Partitions. • Ran 3 at a time.	21 ½ Mins
• Rebuild & IC 4 NPI Indexes	• One for each NPI running in parallel	16 Mins
	Total Time	**57 Mins**
Restart Application		

Db2 – Loading Large Tables

- Background
- 1st Tests – NPIs or no NPIs
- Splitting the load file
- 1st Prod Run
- Bang - Splitting DfSort
- **Inline Stats & Copies**
- Adding Columns
- Adding Indexes

Inline Stats & Image Copies

- Occurred to me – eventually – that if we didn't need the Reorgs we could use Inline Stats & Image copies on the Load.
- We couldn't image copy the Indexes at the same time but as it was only an informational message we could turn table back online before the index copies have finished.
- New Load card

```
LOAD DATA          SHRLEVEL NONE LOG NO NOCOPYPEND
     PARALLEL(0) INDEXDEFER NPI SORTKEYS SORTDEVT 3390

COPYDDN(ICOPY1)

STATISTICS
  TABLE ALL SAMPLE 20
  INDEX (ALL FREQVAL NUMCOLS 6 COUNT 15)
```

Db2 – Loading Large Tables

Step	Description	Run time
• Stop Application		
• Backup Tablespace & Indexes		4 Mins
• Drop NPIs	Dropped all NPIs	1 Min
• Load in Primary Key Order • 1.7Bn Rows • Inline Stats & Backups	• 26 Load jobs, each loading 10 Partitions. • We ran 5 in parallel on one Member of the datasharing group.	22 ½ Mins
• Rebuild & IC 7 NPI Indexes	• One for each NPI running in parallel	20 Mins
	Total Outage Time	**47.5 Mins**
Turn Application back on		
Backup Partitioned Index		4 Mins

Db2 – Loading Large Tables

- Background
- 1st Tests – NPIs or no NPIs
- Splitting the load file
- 1st Prod Run
- Bang - Splitting DfSort
- Inline Stats & Copies
- **Adding Columns**
- Adding Indexes

Adding / Amending Columns

- Only tried at reload time.
- Not sure if an online Reorg switch will complete without turning off the application.
- When I try it I will let you know.

Db2 – Loading Large Tables

- Background
- 1st Tests – NPIs or no NPIs
- Splitting the load file
- 1st Prod Run
- Bang - Splitting DfSort
- Inline Stats & Copies
- Adding Columns
- **Adding Indexes**

Adding Indexes

- Mostly added when we have been reloading
- Did add one when we turned the Application off for 20 Minutes to let the rebuild complete without any work running.