

# *Back to Basics: Dump and Load*

Scott M. Dulecki

BravePoint

**BRAVEPOINT**

# *Agenda*

- **Setting the stage**
- What is D&L
- Why D&L
- How to D&L
- Results
- Take-homes

# *Setting The Stage: Me*

- Scott M. Dulecki
- Board Member, Midwest MFG/PRO Users Group
- President, West Michigan Progress Users Group
- Past President, Michigan Progress Users Group
- PEG member 1998061901
- Author of:
  - *Back to Basics: Dump and Load*
  - *Safe Haven: Archiving in MFG/PRO*
  - *Safe Haven: MFG/PRO Basics*

# *Setting The Stage: Us*

- BravePoint Inc. ([www.BravePoint.com](http://www.BravePoint.com))
- 100+ Employees
- Progress Service Provider
- QAD Channel Sales Partner
- QAD Service Alliance Partner
- Three of us have used Progress since 1984

# *Setting The Stage: You*

- Official DBA?
- De facto DBA?
- Can spell DBA on a good day?

# Agenda

- Setting the stage
- **What is D&L**
- Why D&L
- How to D&L
- Results
- Take-homes

## *What is a D&L?*

- Recreate the database
- Dump out current data
  - Data, sequences, metaschema
- Create new database
  - New blocksize, structure, other settings
- Load dumped data into new database

# *Agenda*

- Setting the stage
- What is D&L
- **Why D&L**
- How to D&L
- Results
- Take-homes



# *Why Should You D&L?*



- Cowboy model
- Opportunist model
- Good Soldier model
- Twisted Arm model
- Intelligent model

# *Cowboy Model*

- “We do it once a year, whether we need to or not”

# *Opportunist Model*

- “Oh, joy! A three-day weekend!”
- “Happiness! A FOUR-day weekend!”
- “Happy, happy, joy, joy! It’s Christmas!”
  
- Often combined with a Cowboy

# *Good Soldier Model*

- “We’ve always done it this way”
- “It ain’t broke, so don’t try to fix it”
- “It’s been working fine so far”
  
- Someone once said to do it this way
- We’ve never questioned why or how...
- Someone else’s Opportunist Cowboy

# *Twisted Arm Model*

- We HAVE to D&L in order to...
  - Move to a new platform
  - Change the DB blocksize
  - Change Records per Block
  - Convert to Storage Areas (Type I, II)
  - Recover data from a corrupted DB

# *Intelligent Model...*

- Let the database tell you when it's time
  - Fragmentation
  - Scatter factor
  - Index rebuild
  - Improve performance
  - Space recovery

# Fragmentation

- Fragments are records
- Ideally, one fragment per record
- Run dbanalsys or tabanalsys to see

```
RECORD BLOCK SUMMARY
RECORD BLOCK SUMMARY FOR AREA "TRHIST" : 25
-----
Table                Records      Size      -Record Size (B)  ---Fragments--- Scatter
                   31028830    8.0G     Min  Max  Mean  Count Factor  Factor
PUB.tr_hist          31028830    8.0G     214  376  277   31028840    1.0    1.1
-----
Totals:              497122362   72.3G     6  2699  156   531383879    1.0    4.0
11138601 RM block(s) found in the database.
86.55% of the RM block space is used.
```

# *Scatter Factor*



- How close are the records?
  - Physically?
    - Less of an issue with dedicated storage areas
  - Logically?
    - By a particular (most common) index
    - Can still be an issue
- Run `dbanalyis` or `tabanalyis` to see



# *Scatter Factor Settings*



- For “real” tables:
  - 1.0 – Perfect... enjoy! (Green light)
  - 2.1 – Deteriorating... make plans (Yellow Light)
  - 3.1 – Performance problems ARE happening
  - 4.1 – Take action (Red Light)
- Note: Progress recommendations are very low

# Sample Scatter Factor

## RECORD BLOCK SUMMARY

RECORD BLOCK SUMMARY FOR AREA "TRHIST" : 25

| Table       | Records   | Size  | -Record Size (B)- |      |      | ---Fragments--- | Count | Factor | Scatter Factor |
|-------------|-----------|-------|-------------------|------|------|-----------------|-------|--------|----------------|
|             |           |       | Min               | Max  | Mean |                 |       |        |                |
| PUB.tr_hist | 31028830  | 8.0G  | 214               | 376  | 277  | 31028840        | 1.0   | 1.1    |                |
| Totals:     | 497122362 | 72.3G | 6                 | 2699 | 156  | 531383879       | 1.0   | 4.0    |                |

11138601 RM block(s) found in the database.

86.55% of the RM block space is used.

# *Index Rebuild*

- May improve performance without D&L
- Always happens with D&L
- Run `dbanalys` or `ixanalys` to see
  - Levels – how many reads?
  - Utilization – 60% cutoff

# Sample Index Status

INDEX BLOCK SUMMARY FOR AREA "TRHIST\_IDX" : 26

| Table        | Index | Fields | Levels | Blocks  | Size   | % Util | Factor |
|--------------|-------|--------|--------|---------|--------|--------|--------|
| PUB.tr_hist  |       |        |        |         |        |        |        |
| tr_addr_eff  | 3252  | 3      | 3      | 5513    | 11.7M  | 54.6   | 1.9    |
| tr_batch     | 3253  | 2      | 3      | 2540    | 7.2M   | 73.1   | 1.5    |
| tr_date_trn  | 3254  | 3      | 3      | 41524   | 99.4M  | 61.7   | 1.8    |
| tr_eff_trnbr | 3255  | 3      | 3      | 43772   | 99.5M  | 58.6   | 1.8    |
| tr_nbr_eff   | 3256  | 3      | 3      | 8084    | 18.9M  | 60.3   | 1.8    |
| tr_part_eff  | 3257  | 3      | 3      | 27388   | 61.1M  | 57.5   | 1.8    |
| tr_part_trn  | 3258  | 3      | 3      | 50163   | 107.6M | 55.3   | 1.9    |
| tr_serial    | 3259  | 2      | 3      | 5551    | 11.8M  | 54.9   | 1.9    |
| tr_trnbr     | 3260  | 2      | 3      | 34724   | 99.1M  | 73.5   | 1.5    |
| tr_type      | 3261  | 3      | 3      | 5804    | 13.1M  | 58.0   | 1.8    |
| tr_vend_lot  | 3262  | 2      | 3      | 2967    | 7.2M   | 62.6   | 1.7    |
| Totals:      |       |        |        | 3286834 | 16.6G  | 66.5   | 1.6    |

3286834 index block(s) found in the database.

66.46% of the index block space is used.

# *Improve Performance*

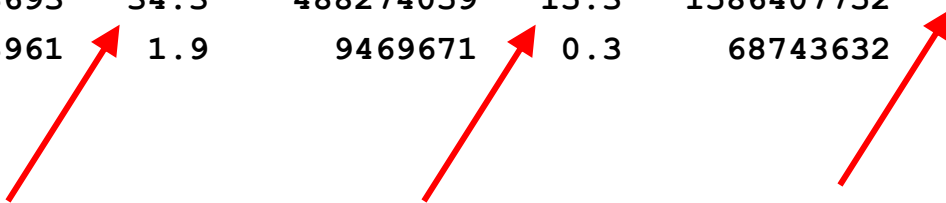


- Statement of the Obvious
  - The bigger the DB, the longer it takes to access
- Adjust parameters
  - See Twisted Arm model
- Make it smaller
  - Archive/delete data
  - May cause performance problems until D&L

# Database Summary

## DATABASE SUMMARY (2314)

| NAME      | Records    |       | Indexes   |       | Combined   |       |
|-----------|------------|-------|-----------|-------|------------|-------|
|           | Bytes      | Tot % | Bytes     | Tot % | Bytes      | Tot % |
| glr_mstr  | 0          | 0.0   | 3         | 0.0   | 3          | 0.0   |
| gltr_hist | 1098133693 | 34.3  | 488274059 | 15.3  | 1586407752 | 49.6  |
| gltw_wkfl | 59273961   | 1.9   | 9469671   | 0.3   | 68743632   | 2.1   |



# Database Summary Spreadsheet

## DATABASE SUMMARY (2314)

| NAME      | Records  | Bytes | Tot % | Indexes   | Bytes | Tot % | Combined   | Bytes | Tot % |
|-----------|----------|-------|-------|-----------|-------|-------|------------|-------|-------|
| glr_mstr  |          |       |       |           |       |       |            | 3     | 0.0   |
| gltr_hist | 10981336 |       | 34.3  | 488274059 | 15.3  |       | 1586407752 | 49.6  | 49.6  |
| gltw_wkfl | 592739   |       | 1.9   | 9469671   | 0.3   |       | 68743632   | 2.1   | 89    |

Load into spreadsheet and sort to find largest files, and largest payback.

## DATABASE SUMMARY (2314)

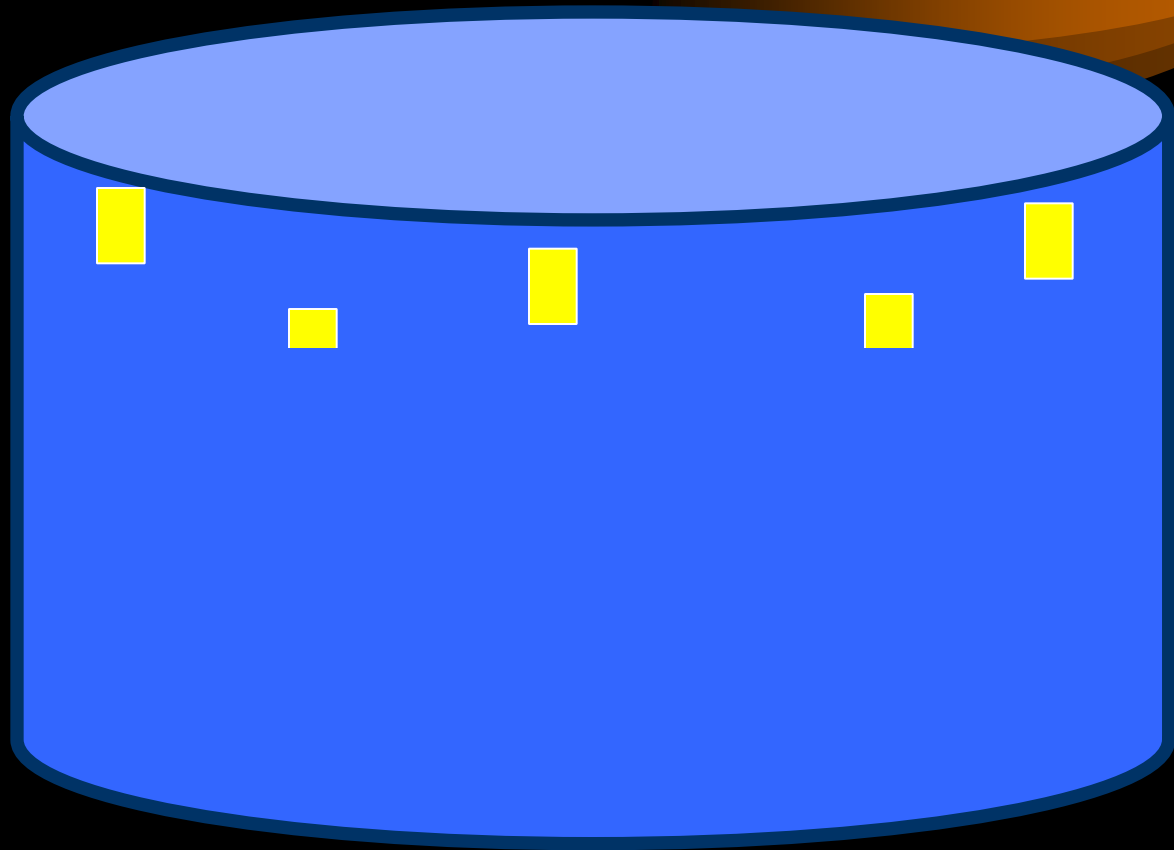
| NAME      | Records   | Bytes | Tot %     | Indexes | Bytes      | Tot % | Combined | Tot % | Cum Pct |
|-----------|-----------|-------|-----------|---------|------------|-------|----------|-------|---------|
| gltr_hist | 10981336  | 34.3  | 488274059 | 15.3    | 1586407752 | 49.6  | 49.6     | 49.6  |         |
| tr_hist   | 391692907 | 12.2  | 89583695  | 2.8     | 481276602  | 15    | 64.6     | 64.6  |         |
| trgl_det  | 221359788 | 6.9   | 153487999 | 4.8     | 374847787  | 11.7  | 76.3     | 76.3  |         |
| abs_mstr  | 204364202 | 6.4   | 52819555  | 1.7     | 257183757  | 8     | 84.3     | 84.3  |         |
| op_hist   | 66680262  | 2.1   | 17843096  | 0.6     | 84523358   | 2.6   | 86.9     | 86.9  |         |
| gltw_wkfl | 59273961  | 1.9   | 9469671   | 0.3     | 68743632   | 2.1   | 89       | 89    |         |
| schd_det  | 54449327  | 1.7   | 11986921  | 0.4     | 66436248   | 2.1   | 91.1     | 91.1  |         |

# *Space Recovery*

- When you archive/delete, you create space
  - Promon, 7 - Free blocks below HWM
- Progress never gives space back to the OS
- Force it with a D&L



*Let's Talk About Space...*



# Agenda

- Setting the stage
- What is D&L
- Why D&L
- **How to D&L**
- Results
- Take-homes

# *Step 1: Prepare!*

- Benchmark
  - Heavy reports
  - Heavy processes
  - Dbanalys
    - Before record counts to validate
    - May reveal corruption
      - “**SYSTEM ERROR: wrong dbkey in block. Found <dbkey>,should be <dbkey2> (1124)**”

## *Step 2: More Prep!*

- Disk space
  - 1-2 times DB size
  - Consider splitting mirrors
    - May be faster, but requires sync-up
  - Separate controllers, file systems if possible

## *Step 3: Choices*



- Dictionary D&L
- Bulk Loader
- Binary D&L
- Parallel D&L
- Automating the process

## *Step 4: BACK UP!!!*

- At relevant stages, make backups
  - Save early, save often
  - Save before
  - Save after

# *Dictionary Dump*

- Simplest approach and interface
- Usually slowest method
- Can be run non-interactively
  - `product/dump_d.p`
  - `product/load_d.p`
- Can't create files > 2GB
  - Do in stages
  - Code around it
  - Progress 10.1C allows larger files...

# *Binary Dump*

- Option on Proutil
- Fast... very fast...
- Must run only one per table
  - Multiple dictionary dumps may be faster
- Can run multiple tables concurrently
- Dump files are portable across OS
- Doesn't work if there are deleted fields
- No 2GB limit



# Binary Dump Tips

- Use `–RO` (read-only)
- Use a small `–B`
- Tool in Dan Foreman's *DBA Resource Kit*
  - Generate Binary D&L scripts
- OE 10
  - *Proutil dbname –C dump table –index 0*
- Can bring DB corruption with you...

# *General Dump Tips*

- Parallel dumps
  - Multiple CPUs, disks available
  - Finish faster
  - Biggest table will be bottleneck
- Don't forget
  - Sequences
  - `_user` table (no binary)
  - SQL92 privileges

# *Dictionary Load*

- Start loading once tables are dumped...
  - Finish even faster
- Slowest option (except in parallel)

# *Bulkload*



- Option on Proutil
- Load dictionary or .d files
- Fast... but not faster than binary
- Single-threaded only
- Requires index rebuild afterwards

# *Binary Load*

- Single or multi-threaded
- Use DB Broker
  - See speed (record creation)
  - Avoid crash recovery for each load
- Use `-i` (no integrity) for performance
- A number of bugs below 8.3C...

# *Parallel Load*



- Not in version 8
  - Slow, increases scatter
- V9 / OE10 – one load per storage area
- Remember...
  - No integrity
  - APWs, BIW
  - -bibufs
  - Big clustersize (16MB+)
  - -spin
  - Disable AI

# *Index Rebuild*



- Backup first... 😊
- Disk sort method
- Memory sort method

# *Disk Sort Method*



- Size: 1-2X data size
- Most compact index (95% utilization)
- Fastest
  - 20% faster on non-DB disk
- Single-threaded
- 2GB limit until 9.1D08 (no largefiles)
- Multi-volume sort files



# *Memory Sort Method*

- No disk space required
- Less compact index (70% utilization)
  - Use `idxcompact` afterwards – 9.1E+
- Much slower
- Use a larger `-B` (but not too large)

# *Index Rebuild Options*

- -TB 31 (64)
- -TMB
- -TF
- -TM 32
- Db.srt
- -t
- -B
- -SS
- -SG
- -rusage
- Disk sort (max)
- Temp Merge Block (10.2B04)
- Memory Usage Factor
- Disk sort (max)
- Multi-volume sort
- Unix disk sort
- Memory sort only
- 9.1B “Build indexes” option
- Sort Groups; use 64
- Statistics

# *Agenda*

- Setting the stage
- What is D&L
- Why D&L
- How to D&L
- **Results**
- Take-homes

# Benchmarks

- Dict Load/Idx Inactive/idxbuild: 27:07
- Dict Load/Idx Active (3 threads) 23:57
- Bulk Load/idxbuild: 16:10
- Serial Binary/idxbuild: 15:15
- Parallel Binary/idxbuild: 15:29
- Serial Binary with -SS: 19:56
- Parallel Binary with -SS: 38:04

# *Target Benchmark (1)*

- 80 GB MFG/PRO Database
- Progress 8.3E
- Single thread Binary dump time:6hrs
- Multi-thread Binary dump time: 4.5hrs
- Single thread Binary Load:10hrs
- Multi-thread Binary Load:4.5 hrs
- Index Rebuild: 12 hrs

## *Target Benchmark (2)*

- Progress 9.1E
- Single thread Binary dump time:3.5 hrs
- Multi-thread Binary dump time:00:50hrs
- Single thread Binary Load:6hrs
- Multi-thread Binary Load:2.5 hrs
- Index Rebuild: 8hrs

# *Real-World Impact*

- Deleting 50,000 Item Masters
- Initial for one part
  - 10 minutes, peaks of 15-20 minutes
- After 10,000 or so
  - 1 hour, peaks of 9 hours
- First weekend after D&L
  - 3,000 items... 100+/hour

# *Agenda*

- Setting the stage
- What is D&L
- Why D&L
- How to D&L
- Results
- **Take-homes**



# *Questions Before Wrapup*



# *Santa's List*

- Verify your record counts
- Check logs for errors
  - “fail”, “error”, “(1124)”
  - `ls -l /tmp/dl/*.e`
- Check logs for success
  - `grep " 0 errors" /tmp/dl/*.log | wc -l`
  - `grep " dumped" /tmp/dl/*.log`
  - `grep " loaded" /tmp/dl/*.log`
  - `ls -l /tmp/dl/*.d`
  - `ls -l /tmp/dl/*.bd*`
- Run Update Statistics

# *Tools and Helpers*

- PSDN: Tom Bascom, Massively Parallel D&L
  - Uses Buffer-Copy method
- Dan Foreman: DBA Resource Kit
  - Dump/Load checklist
  - Binary Dump and Load Script Generator
  - Dump/Load Verification Utility
  - Index Build By Area
- Dan Foreman: Pro D&L
- Scott Dulecki: *Back to Basics: Dump and Load*

# *For Further Information*

*For a copy of this presentation, leave me your  
business card with “Basic D&L” on it*

**Scott M. Dulecki**

**Bravepoint**

**616/957-3184**

**[sdulecki@bravepoint.com](mailto:sdulecki@bravepoint.com)**