Version 15, 02 oct 2024 07:41

Tales of the Secret Bunker 2024

research from the parmington foundation

Gus Bjorklund,

head groundskeeper

the parmington foundation

Mike Furgal,

assistant to the head groundskeeper

the parmington foundation

PUG CHALLENGE

Boston, USA

29 sep 2024

02 oct 2024

tpf

Abstract

The “Secret Bunker” has been used for many interesting OpenEdge RDBMS investigations in past years. Please join your intrepid explorers for another trip into the bowels of the (recently relocated) bunker.

As databases are used and modified over an extended period, database administrators will gain experience and knowledge of what they should have done. They will discover that various improvements in database organization could be made to improve day-to-day performance and maintenance operations. In this talk we will examine some techniques you might use for “online table dump and load” operations with the database.

OpenEdge Release 12 has a variety of new features and capabilities that can make such maintenance tasks more efficient and less disruptive to normal production operations. In the bunker we test some of these capabilities and now we report the results.

Part 0 of 3
About the Secret Bunker

Established in 2002 to investigate Progress RDBMS on Linux

The Original Benchmark Laboratory

Secret Bunker 2023 and 2024

BUNKERMUSEUM
JANSJE SCHONG

Why are we here today ?

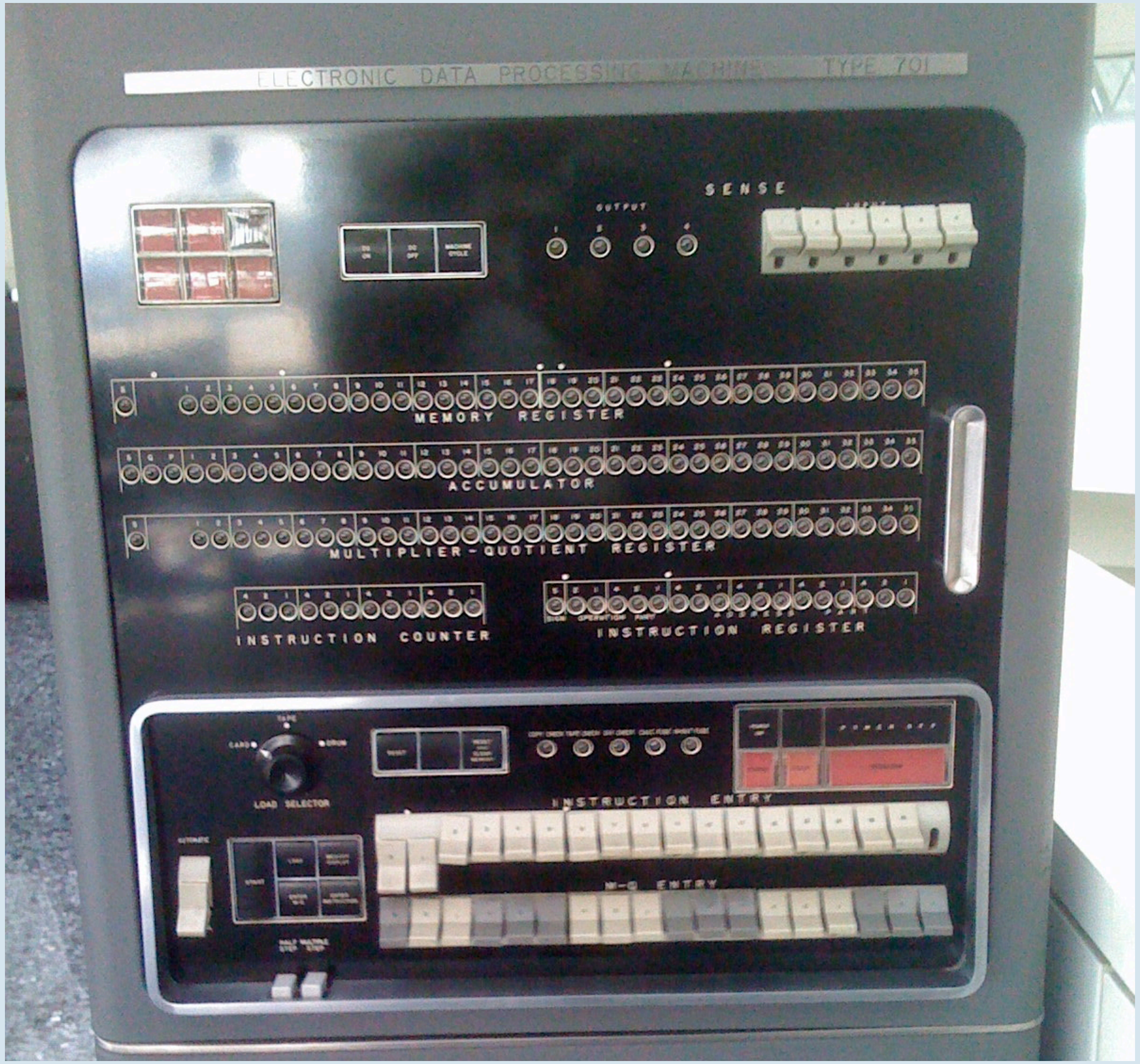
LAST Year’s Topic: Dump and Load Optimization

- Test out the new “proutil <dbname> -C tablereorg”
- How does it work ?
- Can it be run online effectively ?
- What can it do for you ?
- Benchmarks and results

This Year’s Topic:
Revisit Last Year’s Topic
+
Advanced Tablereorg

Part 1 of 3
Table Reorg Basics

Test Machine



https://en.wikipedia.org/wiki/Mainframe_computer#/media/File:IBM_701console.jpg

2023 and 2024 Test Machine

Dell PowerEdge 2050 iii (circa 2009)

- 2 XEON E5450 3 GHz 4-core processors
- 64 GB ECC RAM
- PERC/6 RAID controller
- 6 Hitachi 2 TB 7200 rpm SATA disk drives
  - /boot (xfs), /home (xfs) and swap on 4x2 TB sata drives in RAID 0.
  - /bi (ext3) on 2 tb drive
  - /ai (ext3) on 2 tb drive

“furgal test” time: 0.42 seconds. (228 mb/sec)

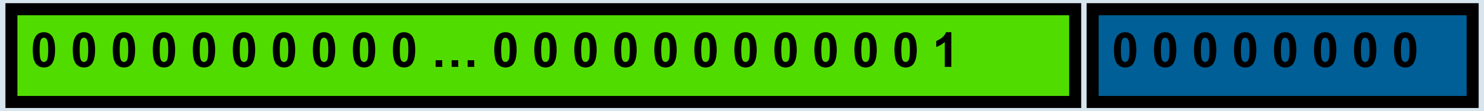
Why do we need the tablereorg utility?

0. make physical storage order match index order

1. reduce or eliminate record fragmentation

(dumping and loading could do both, but with downtime. Maybe a lot of downtime.)

What Is A Rowid ?



- Unique 64-bit identifier for a record in a table
  - Unique within a partition or area
- Encodes the “physiological” storage address
  - Used to locate record fast
- “Constant” for life of record
  - Until you delete it or
  - Change partition key value

Index Order Vs Storage Order -- NOT Optimal

Index	Rowid
BOLONIA	3331
BOLTON	5554
BOLTON	9001
BOLTON	9022
BONN	8001
BOSTON	1111
BOSTON	1118
BOSTON	7001
BOSTON	9002
BOSTON	8124
BOSTON	1003
BOSTON	2005
BOSTON	3332
BOSTON	9999
CARDIFF	1112

Index Order Vs Storage Order -- Optimal

Index	Rowid
BOLONIA	3331
BOLTON	3332
BOLTON	3333
BOLTON	3334
BONN	3335
BOSTON	3336
BOSTON	3337
BOSTON	3338
BOSTON	3339
BOSTON	3340
BOSTON	3341
BOSTON	3342
BOSTON	3343
BOSTON	3344
CARDIFF	3345

Index Order Vs Storage Order -- Optimal

Index	Rowid
BOLONIA	3331
BOLTON	3332
BOLTON	3333
BOLTON	3334
BONN	3335
BOSTON	3336
BOSTON	3337
BOSTON	3338
BOSTON	3339
BOSTON	3340
BOSTON	3341
BOSTON	3342
BOSTON	3343
BOSTON	3344
CARDIFF	3345

These 15 records can all fit into the same block.

15 * 132 = 1980 bytes

Room for many more in a 4 KB block.

Proutil db –C tablereorg arguments

```
proutil db-name -C tablereorg [owner-name.] table-name
    [ info ] |
    [ [ resume | resume-numrecs n | resume rowid n]
    [ nosmartscan ]
    [ restrict [ EQ value ] |
               [ LT | LE high-value ] |
               [ GT | GE low-value [ AND LT | LE high-value ] ] ]
    [ useindex index-name ] [ recs n ]
    [ searchdepth n ]
    [ reusepercent n ] ]
    [ tenant tenant-name | group group-name |
                 partition partition-name | composite initial ]
```

Why Am I Embarrased ?

Scenario 0: starting conditions -- last year

- Regular ATM database but small
  (only 10,000,000 account records)
- Unused after building so account numbers and rowids in same order
- All records a little over 100 bytes long
- Database is about 1.5 GB
  - 351,308 blocks (4k)
- No fragmented records
- 98% RM space utilization
- No free clusters
- 3 blocks on account RM chain

Scenario 0: starting conditions -- last year

- R al ATM database mall
  10,000,000 account re ls)
- Un after building
- All rec a little over 100 byte ng
- Database bout 1.5 GB
  - 371,020 bl (4k)
- No fragmented rds
- 94% RM space uti on
- No free clusters
- blocks on account RM

Record ordering was NOT optimal as I claimed.

Scenario 0: table reorg 1 -- last year

- proutil atm -C tablereorg account recs 1000
- Took 6 min 57 sec.
- Database grew from 1.5 GB to 3.1 GB
  - From 371,020 to 574,668 blocks (4k)
- No fragmented records
- 60% RM space utilization
- No free clusters
- 328,100 blocks on the RM chain

Scenario 0: table reorg 1 – last year

- m atm -C tablereorg unt recs 1000
- 6 min 57 sec.
- Dat e grew from 1.5 GB to GB
  - Fro 020 to 574,668 blocks (4
- No fragm d records
- 60% RM spa tilization
- No free clusters
- 328,100 blocks on M chain

Record ordering was NOT optimal as I claimed.

Test result is invalid.

Scenario 0: table reorg 1 – last year

- proutil atm -C tablereorg account recs 1000
- Took 6 min 57 sec.
- Database grew from 1.5 GB to 3.1 GB
  - From 371,020 to 574,668 blocks (4k)
- No fragmented records
- 60% RM space utilization
- No free clusters
- 328,100 blocks on the RM chain

Scenario 0: table reorg 1 – this year

- proutil atm -C tablereorg account recs 1000
- Took 6 min 57 sec.
- 0 records were processed
  - No changes were made

Scenario 0: table reorg 2 – last year

- reorg of just reorg’ed database
- proutil atm -C tablereorg account recs 1000
- Took a few seconds longer than first time (7 min +)
- Database grew slightly from 3.1 GB to 4.2 GB
  - From 574,668 to 593,676 blocks (4k)
- No fragmented records
- 62% RM space utilization !!!!
- No free clusters
- 531,596 blocks on the RM chain
- Records were moved but nothing much was accomplished

Scenario 0: table reorg 2 – this year

- reorg of just reorg’ed database
- proutil atm -C tablereorg account recs 1000
- Took a few seconds longer than first time (7 min +)
- 0 records were processed
- No changes were made

Scenario 1: Same as previous but bigger (100,000,000 accounts) – this year

- Database size is now 14 GB
- Again unused after building, but rows only partially ordered (same as last year)
- Account area is 3,332,495 blocks
- 92 % space utilization
- Zero fragmentation

Scenario 1: table reorg 1 – this year's result

- proutil atm -C tablereorg account recs 1000
- Took 65 min 37 sec.
- Database grew from 14 GB to 22.2 GB
- Account area grew by 2,049,856 blocks
  went from 3,332,495 to 5,382,351 blocks (4k)
- No fragmented records
- 60% RM space utilization
- No free clusters

Scenario 1: table reorg 2 – this year’s result

- reorg of just reorg’ed database
- proutil atm -C tablereorg account recs 1000
- Took 66 min 37 sec.
- 0 records processed

Scenario 2: Make account records vary in size -- this year

- 100,000,000 accounts
- Record size varies from ~100 to ~1500 bytes
- 1 in 20 records is ~100 to ~1500 bytes
- Database size is 9,900,428 blocks – 40 GB
- RM chain has 2,021,146 blocks
- There are 192.870,695 fragments
- 95 % RM space utilization

Scenario 2: table reorg 1 – this year

- proutil atm -C tablereorg account recs 10000
- Took 86 min 35 sec
- Database grew from 40 GB to 46 GB
- Account area grew by 2,122,944 blocks
- From 9,479,887 to 11,602,831 blocks
- Zero fragmentation
- 71.84% RM space utilization
- RM chain has gone from 1,760,347 to 10,699,287 blocks

Scenario 2: table reorg 2 – this year

- reorg of just reorg’ed database
- proutil atm -C tablereorg account recs 100000
- Took 1 min 52 sec
- 0 records processed

Part 2 of 3
Advanced Table Reorg

The Setup

Source Machine as described previously

Introduce a smaller Linux machine as Target

Source ⬄ Target network speed is gigabit ethernet

- 1 switch between nodes

Enable AI

Enable OE Replication

Target Test Machine

Lenovo ThinkCentre M82

- Intel Core i5-3470 Quad-Core CPU
- 32 GB RAM
- 1 TB PCIe M.2 SSD

“furgal test” time: 2.0 seconds. (48 MB/sec)

The Scenarios

1. ATM Standalone (modified to add readers)

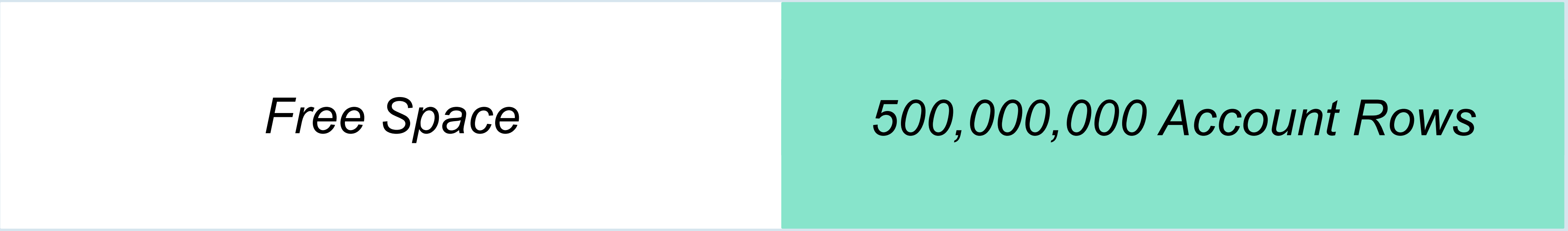
2. Table Reorg Standalone
3. Table Reorg with modified ATM

4. Table Partitioning Move Standalone
5. Table Partitioning Move with modified ATM
6. Tinkered with Table Partition Move with modified ATM

7. Dump/Load/Idxbuild/Rebaseline

Database Setup

- ATM database with 500,000,000 account records.
- Originally had 1,000,000,000 account records.
  - Original 500,000,000 removed
  - Index Rebuild to remove index place holders
- 120 GB Accounts Area



Modified ATM benchmark

- 15 Users
- 12 Users doing updates as normal ATM activity
  - Update Account, Teller, and Branch Balance
  - Create History Record
- 3 Users doing reading
  - Randomly Read 101 Account Record
  - Read a Teller and Branch Record

Modified ATM benchmark - RESULTS

- 420 Transactions Per Second
- Longest Response Time 0.5 seconds

- This is our baseline

proutil atm -C tablereorg account recs 1000 searchdepth 5

- Tablereorg is the only activity on the database and machine
  - Server is up
  - OE replication running

- 10 Hours 11 Minutes

- Last Bunker this took 8 Hours and 40 Minutes
  - No After Imaging
  - No OE Replication

Can you reset the test without rebaselining OE Replication?

Can you reset the test without rebaselining OE Replication?

- Have a backup on the source machine with –REPLTargetCreate
- Have that same backup on the target machine

- Restore them both
- Reenable replication

- Startup Target, Startup Source
  - dsrutil –C status on target 3105 - transition status
  - dsrutil –C status on source 1199 - inactive

Can you reset the test without rebaselining OE Replication?

Shutdown both and remove the atm.repl.recovery file on both target and source, startup and all is replicating as normal

Tablereorg and Modified ATM - RESULTS

- Tablereorg now took 24 Hours and 19 Minutes
  - Compared to 10 Hours 11 Minutes

ATM Results

- 240 Transactions Per Second
- Longest Response Time 51.6 seconds

Compared to Baseline ATM

- 420 Transactions Per Second
- Longest Response Time 0.5 seconds

Table Partitioning Move

- Enable Table Partitioning
- Add a field (myarea) to account with a non-unique index (iarea)
  - Partition setup when the value is 0 account resides in Account Area
  - Partition setup when the value is 1 account resides in Account1 Area

Table Partitioning Move

```
set schema 'pub';
alter table pub.accounts        <<--- table
    partition by list myarea   <<--- field

    using table area ”Accounts Area"
    (
       partition "Initial" values in (0) using table area ”Account Area"
    )
    using index ”iarea";
commit;
alter table pub.accounts
    add partition ”Account1" values in (1) using table area ”Account1
Area";
commit;
quit;
```

Table Partitioning Move

```
def var i as int no-undo.
etime(yes).
pause 0 before-hide.
for each accounts exclusive-lock by cust-num.
    if myarea = 1 then next.
    myarea = 1.
    i = i + 1.
    if i mod 50000 = 0 then display string(time, "HH:MM:SS") " " i.
end.
display etime format ">>>,>>9".
```

Table Partitioning Move Standalone - RESULT

Time to move all records to Account1 Area

- 17 Hours, 23 Minutes, 20 Seconds

Table Partitioning Move with ATM - RESULT

Time to move all records to Account1 Area

- 40 Hours, 45 Minutes

ATM Results

- 193 Transactions Per Second
- Longest Response Time 9.4 seconds
  - A small percentage of transactions had this wait time. But if I was the one waiting…

Compared to Baseline ATM

- 420 Transactions Per Second
- Longest Response Time 0.5 seconds

Table Partitioning Move alternative one

Lets add some sleep time

Every 200 records moved, sleep ½ second.

Time to move all records to Account1 Area

- 432 hours, or 18 days!!!!

ATM Results

- 339 Transactions Per Second
- Longest Response Time 4.2 seconds

Table Partitioning Move Rate

With no sleep time, reassigning to new area has a rate of about 8,000 records per second.

Adding in a ½ second pause every 200 records, reduced the rate to 240 records per second.

What about every 8,000 records, sleep ¼ second?

Table Partitioning Move alternative two

Every 8,000 records moved, sleep ¼ second.

Time to move all records to Account1 Area

- 51 hours

ATM Results

- 340 Transactions Per Second
- Longest Response Time 0.6 seconds

Time to move decreased dramatically, ATM results stayed the same

Results Summary

- ATM Alone
  - 420 Transactions Per Second (TPS)
  - Longest Response Time 0.5 seconds
- Tablereorg Alone
  - 10 hours 11 minutes
- ATM and Tablereorg together
  - Tablereorg – 24 hours and 19 minutes
  - ATM – 240 TPS

- Table Partition Move (TPM) Alone
  - 17 hours 23 minutes
- ATM and TPM
  - TPM– 40 hours 45 minutes
  - ATM 193 TPS
- ATM + TPM + 200 rows sleep ½ sec
  - TPM – 18 days!!
  - ATM - 339 TPS
- ATM + TPM + 8000 rows sleep ¼ sec
  - TPM – 51 hours
  - ATM - 340 TPS

Dump/Load/Idxbuild/Rebaseline

Dump – 27 Minutes

Load – 95 Minutes

Index Rebuild – 19 Minutes

Total 2 Hours, 22 Minutes

Dump/Load/Idxbuild/Rebaseline

Rebaseline LAN

Backup - 20 Minutes

Copy to Target - 30 Minutes

Restore – 40 Minutes

Total LAN – 90 Minutes

Grand Total LAN – 3 Hours 52 Minutes

During this time ATM TPS = 0

Rebaseline LAN

Backup - 20 Minutes

Copy to Target – 2 Hours 21 Minutes

Restore – 40 Minutes

Total WAN – 3 Hours 21 Minutes

Grand Total WAN – 5 Hours, 43 Minutes

During this time ATM TPS = 0

Now What ?

- Don’t believe Gus!!!!
- Tablereog is a fast and useful tool to remove fragmentation and scatter
  - The database may increase in size depending on how contiguous free space is organized
  - Can have significant effect on performance
  - Should be run at off hours when the user load is not high
- Table Partitioning Move is useful
  - Takes longer
  - You have control over the speed
  - Area size doubles
- If you can tolerate downtime, traditional D/L still wins
- TEST before using in production !!!

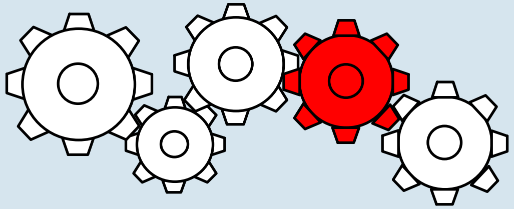
The Next Secret Bunker Location ?

What should we do for the next Secret Bunker talk?

Any ideas?

Questions

?



research from the parmington foundation