Version 24, 14 nov 2023 12:44 **Tales of the Secret Bunker** 2023

research from the parmington foundation

Gus Bjorklund,

head groundskeeper

the parmington foundation

Mike Furgal,

assistant to the head groundskeeper the parmington foundation

Tom Bascom,

curmudgeon



PUG **CHALLENGE AMERICAS**

Burlington, MA, USA 12 – 15 nov 2023

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.



About the Secret Bunker



Established in 2002 to investigate Progress on Linux



The Founders

- Foreman
- Bjorklund
- Harlow









2002 Secret Bunker was at this secret location





The Benchmark Laboratory





Secret Bunker 2023



This Year's Secret Bunker Location







This 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

2023 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 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

[D]

Item	Qty	Cost
Computer	1	\$189.00
64 GB ECC RAM	1	\$50.00
Disk drives	6	\$300.00 (@ \$50 ea)
Drive mounting screws	24	\$6.32 for 50
SAS Drive carriers	6	\$29.94 (@ \$4.99 ea)
SAS to SATA adapters	6	\$30.00 (@\$5.00 ea)
Operating system	1	\$0.00
Total	44	\$605.26

"bigrow test" time: 1.1 seconds. (82.27 mb/sec)



About Scatter and Fragmentation

Database Scatter

- Comes in 3 forms
 - Record Fragmentation
 - Physical Scatter
 - Logical Scatter
- Record Fragmentation





Physical Scatter



each color represent a different table

tpf

Logical Scatter

- How ordered are the records by some index?
- Each index gives a different logical ordering
- Are rowids in the same order as the keys? (there can be only one physical ordering)

| :50 | 167 | 40 | 407 | 424 | 177 | 124 | 456 | 321 | 286 | 46 | 37 | 390 | 244 | 259

 | 202 | 135 | 336 | 359 | 480 | 108 | 83 | 274 | 462

 | 394 | 224 | 30 | 341
 | 441 | 451
 | 95 | 51 | 316
 | 172 | 104 | 32 | 374 | 150
 | 323 |
|-----|--|--|--|---|--|---|---|---|---|---|---|--|---
--
--
---|---|---|---|---|---|--|--|--
--
--|---
---|--|--|--
--
--|--|--
---|--
--|---|--|---|---|
| 282 | 436 | 23 | 62 | 400 | 75 | 427 | 473 | 458 | 67 | 41 | 125 | 326 | 198 | 309

 | 152 | 213 | 284 | 255 | | 65 | 385 | 267 | 248

 | 337 | 439 | 147 | 383
 | 460 | 364
 | 474 | 419 | 39
 | 170 | 465 | 405 | 154 | 252
 | 80 |
| 191 | 268 | 197 | 144 | 411 | 164 | 84 | 130 | 60 | 123 | 461 | 377 | 307 | 272 | 11

 | 221 | 450 | 76 | 351 | 5 | 109 | 290 | 128 | 380

 | 118 | 429 | 74 | 345
 | 116 | 63
 | 121 | 18 | 357
 | 468 | 106 | 420 | 434 | 295
 | 155 |
| 301 | 141 | 22 | 397 | 437 | 183 | 300 | 449 | 218 | 233 | 347 | 261 | 187 | 190 | 9

 | 143 | 146 | 20 | 17 | 24 | 331 | 243 | 303 | 45

 | 452 | 192 | 269 | 79
 | 256 | 422
 | 239 | 199 | 7
 | 367 | 356 | 153 | 432 | 446
 | 56 |
| 131 | 194 | 312 | 457 | 242 | 349 | 102 | 348 | 190 | 49 | 260 | 201 | 245 | 291 | 158

 | 85 | 204 | 388 | 416 | 319 | 174 | 181 | 279 | 391

 | 6 | 220 | 113 | 280
 | 225 | 58
 | 138 | 229 | 477
 | 313 | 373 | 200 | 103 | 69
 | 217 |
| 230 | 15 | 289 | 276 | 72 | 188 | 264 | 50 | 298 | 61 | 402 | 184 | 318 | 470 | 71

 | 270 | 241 | 119 | 177 | 163 | 148 | 360 | 262 | 297

 | 142 | 251 | 165 | 314
 | 384 | 173
 | 294 | 305 | 315
 | 342 | 13 | 471 | 363 | 186
 | 472 |
| 410 | 403 | 404 | 296 | 479 | 110 | 333 | 185 | 476 | 382 | 33 | 401 | 317 | 205 | 379

 | 418 | 93 | 271 | 240 | 73 | 285 | 171 | 136 | 425

 | 57 | 226 | 206 | 393
 | 166 | 232
 | 114 | 408 | 343
 | 339 | 223 | 160 | 370 | 31
 | 381 |
| 389 | 327 | 392 | 21 | 212 | 409 | 358 | 145 | 53 | 406 | 299 | 426 | 365 | 362 | 78

 | 448 | 350 | 98 | 92 | 322 | 175 | 54 | 90 | 96

 | 354 | 70 | 105 | 431
 | 126 | 249
 | 273 | 43 | 34
 | 330 | 36 | 328 | 338 | 447
 | 361 |
| 454 | 100 | 2 | 320 | 423 | 26 | 210 | 387 | 263 | 162 | 208 | 139 | 292 | 19 | 428

 | 247 | 87 | 52 | 463 | 209 | 133 | 430 | 455 | 180

 | 195 | 398 | 266 | 111
 | 134 | 467
 | 435 | 372 | 308
 | 156 | 310 | 168 | 469 | 169
 | 369 |
| 253 | 81 | 4 | 48 | 258 | 215 | 352 | 44 | 4 | 288 | 86 | 120 | 466 | 47 | 12

 | 478 | 443 | 395 | 216 | 324 | 55 | 340 | 203 | 334

 | 287 | 332 | 417 | 355
 | 413 | 386
 | 129 | 64 | 91
 | 459 | 438 | 16 | 265 | 464
 | 97 |
| 42 | 346 | 325 | 421 | 44 | 306 | 182 | 277 | 238 | 302 | 179 | 371 | 376 | 214 | 82

 | 77 | 8 | 353 | 28 | 246 | 396 | 207 | 211 | 433

 | 378 | 14 | 271 | 25
 | 140 | 335
 | 68 | 445 | 151
 | 375 | 59 | 161 | 283 | 88
 | 329 |
| 122 | 38 | 293 | 415 | 189 | 99 | 159 | 137 | 193 | 117 | 27 | 237 | 115 | 94 | 236

 | 222 | 304 | 178 | 89 | 176 | 281 | 440 | 29 | 278

 | 254 | 275 | 10 | 107
 | 66 | 368
 | 344 | 442 | 366
 | 149 | 311 | 257 | 399 | 132
 | 228 |
| | 282
191
301
131
230
410
389
454
253
42
122 | 50 167 282 436 191 268 301 141 131 194 230 15 410 403 389 327 454 100 253 81 42 346 122 38 | 50 167 40 282 436 23 191 268 197 301 141 22 131 194 312 131 194 312 131 194 312 131 194 312 132 153 289 410 403 404 389 327 392 454 100 22 253 81 22 42 346 325 122 38 293 | 150 167 400 407 282 436 233 62 191 268 197 144 301 141 222 397 131 194 312 457 230 15 289 276 410 403 404 296 389 327 392 211 454 100 22 48 253 81 325 421 42 346 325 421 | 50 167 40 407 424 282 436 23 62 400 191 268 197 144 411 301 141 22 397 437 131 194 312 457 242 230 15 289 276 72 410 403 404 296 479 389 327 392 211 212 454 100 2 48 258 425 81 253 416 325 421 422 346 325 421 441 425 346 325 421 441 | $ \begin{array}{c c c c c c } \hline 167 & 40 & 407 & 424 & 177 \\ \hline 282 & 436 & 23 & 62 & 400 & 75 \\ \hline 191 & 268 & 197 & 144 & 411 & 164 \\ \hline 101 & 268 & 397 & 437 & 183 \\ \hline 134 & 194 & 212 & 397 & 437 & 183 \\ \hline 134 & 194 & 212 & 242 & 349 \\ \hline 134 & 194 & 212 & 242 & 243 & 243 \\ \hline 134 & 154 & 289 & 276 & 72 & 188 \\ \hline 410 & 403 & 269 & 276 & 72 & 188 \\ \hline 410 & 403 & 269 & 276 & 212 & 243 & 263 \\ \hline 454 & 100 & 22 & 214 & 215 & 215 \\ \hline 42 & 346 & 225 & 421 & 44 & 206 \\ \hline 42 & 346 & 255 & 421 & 44 & 306 \\ \hline 42 & 238 & 238 & 415 & 189 & 99 \\ \end{array}$ | $ \begin{array}{c c c c c c } \hline $167 & 40 & 407 & 424 & 177 & 124 \\ \hline $282 & 436 & 23 & 62 & 400 & 75 & 427 \\ \hline $191 & 268 & 197 & 144 & 411 & 164 & 84 \\ \hline $101 & 268 & 197 & 144 & 411 & 164 & 84 \\ \hline $301 & 141 & 22 & 397 & 437 & 183 & 300 \\ \hline $134 & 194 & 22 & 397 & 437 & 183 & 300 \\ \hline $134 & 194 & 312 & 457 & 242 & 349 & 102 \\ \hline $135 & 145 & 289 & 276 & 72 & 188 & 264 \\ \hline $410 & 403 & 404 & 296 & 479 & 110 & 333 \\ \hline $389 & 327 & 392 & 21 & 212 & 409 & 358 \\ \hline $454 & 100 & 2 & 2 & 216 & 216 & 216 \\ \hline $455 & 811 & 2 & 2 & 421 & 424 & 306 & 182 \\ \hline $422 & 388 & 293 & 415 & 189 & 99 & 159 \\ \hline \ $456 & 538 & 538 & 538 & 538 \\ \hline \ $457 & 538 & 538 & 538 & 538 \\ \hline \ $458 & 538 & 538 & 538 & 538 \\ \hline \ \ $458 & 538 & 538 & 538 & 538 \\ \hline \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$ | $ \begin{array}{c c c c c c c c c } \hline $167 & 40 & 407 & 424 & 177 & 124 & 456 \\ \hline $282 & 436 & 23 & 62 & 400 & 75 & 427 & 473 \\ \hline $191 & 268 & 197 & 144 & 411 & 164 & 84 & 130 \\ \hline $101 & 141 & 22 & 397 & 437 & 183 & 300 & 449 \\ \hline $301 & 141 & 22 & 397 & 437 & 183 & 300 & 449 \\ \hline $134 & 144 & 22 & 397 & 242 & 349 & 102 & 348 \\ \hline $135 & 144 & 256 & 72 & 188 & 264 & 50 \\ \hline $136 & 403 & 404 & 296 & 479 & 110 & 333 & 185 \\ \hline $389 & 327 & 392 & 21 & 212 & 409 & 358 & 145 \\ \hline $389 & 327 & 392 & 21 & 212 & 409 & 358 & 145 \\ \hline $430 & 404 & 296 & 479 & 100 & 358 & 145 \\ \hline $454 & 100 & 2 & 2 & 48 & 258 & 216 & 310 & 377 \\ \hline $454 & 100 & 2 & 2 & 421 & 44 & 306 & 182 & 277 \\ \hline $422 & 388 & 293 & 415 & 189 & 99 & 159 & 137 \\ \hline $ | $ \begin{array}{ c c c c c } \hline $ 167 & 40 & 407 & 424 & 177 & 124 & 456 & 321 \\ \hline $ 282 & 436 & 23 & 62 & 400 & 75 & 427 & 473 & 458 \\ \hline $ 191 & 268 & 197 & 144 & 411 & 164 & 84 & 130 & 600 \\ \hline $ 101 & 202 & 397 & 437 & 183 & 300 & 449 & 218 \\ \hline $ 301 & 141 & 22 & 397 & 437 & 183 & 300 & 449 & 218 \\ \hline $ 134 & 214 & 225 & 242 & 349 & 102 & 348 & 206 \\ \hline $ 134 & 214 & 215 & 242 & 349 & 102 & 348 & 206 \\ \hline $ 135 & 248 & 276 & 242 & 349 & 264 & 50 & 218 \\ \hline $ 140 & 249 & 276 & 72 & 188 & 264 & 50 & 218 \\ \hline $ 140 & 249 & 276 & 72 & 188 & 264 & 50 & 218 \\ \hline $ 140 & 249 & 276 & 210 & 132 & 185 & 216 \\ \hline $ 140 & 249 & 276 & 210 & 210 & 310 & 120 & 217 \\ \hline $ 141 & 249 & 248 & 258 & 215 & 326 & 247 & 248 \\ \hline $ 142 & 346 & 325 & 421 & 448 & 306 & 182 & 277 & 238 \\ \hline $ 142 & 348 & 253 & 415 & 189 & 99 & 159 & 137 & 193 \\ \hline $ 142 & 38 & 293 & 415 & 189 & 99 & 159 & 137 & 193 \\ \hline $ 142 & 38 & 293 & 415 & 189 & 99 & 159 & 137 & 193 \\ \hline $ 142 & 38 & 293 & 415 & 189 & 99 & 159 & 137 & 193 \\ \hline $ 142 & 38 & 293 & 215 & 189 & 293 & 215 & 216 & 217 & 218 \\ \hline $ 142 & 38 & 293 & 215 & 189 & 290 & 159 & 137 & 218 \\ \hline $ 142 & 38 & 293 & 215 & 189 & 290 & 159 & 137 & 218 \\ \hline $ 142 & 38 & 293 & 215 & 189 & 290 & 159 & 137 & 218 \\ \hline $ 142 & 38 & 293 & 215 & 189 & 290 & 215 & 217 & 218 \\ \hline $ 142 & 38 & 293 & 215 & 189 & 290 & 215 & 217 & 218 \\ \hline $ 142 & 38 & 293 & 215 & 189 & 290 & 215 & 217 & 218 \\ \hline $ 142 & 38 & 216 & 216 & 216 & 216 & 216 & 216 \\ \hline $ 142 & 216 & 216 & 216 & 216 & 216 & 216 & 216 \\ \hline $ 142 & 216 $ | $ \begin{array}{ c c c c c } \hline $ 167 & 40 & 407 & 424 & 177 & 124 & 456 & 321 & 286 \\ \hline $ 282 & 436 & 23 & 62 & 400 & 75 & 427 & 473 & 458 & 67 \\ \hline $ 191 & 268 & 197 & 144 & 411 & 164 & 84 & 130 & 600 & 123 \\ \hline $ 101 & 22 & 397 & 437 & 183 & 300 & 449 & 218 & 233 \\ \hline $ 134 & 22 & 397 & 437 & 183 & 300 & 449 & 218 & 233 \\ \hline $ 141 & 22 & 397 & 242 & 349 & 102 & 348 & 190 & 491 \\ \hline $ 135 & 249 & 276 & 72 & 188 & 264 & 50 & 298 & 61 \\ \hline $ 136 & 249 & 276 & 72 & 188 & 264 & 50 & 298 & 61 \\ \hline $ 141 & 403 & 404 & 296 & 479 & 110 & 333 & 185 & 476 & 382 \\ \hline $ 389 & 327 & 392 & 21 & 212 & 409 & 358 & 145 & 53 & 406 \\ \hline $ 145 & 301 & 22 & 21 & 212 & 409 & 358 & 145 & 53 & 406 \\ \hline $ 145 & 310 & 22 & 21 & 212 & 409 & 510 & 387 & 263 & 162 \\ \hline $ 145 & 310 & 22 & 21 & 42 & 216 & 316 & 317 & 238 & 302 \\ \hline $ 142 & 346 & 325 & 421 & 44 & 306 & 182 & 277 & 238 & 302 \\ \hline $ 142 & 38 & 293 & 415 & 189 & 99 & 159 & 137 & 193 & 117 \\ \hline $ 142 & 38 & 293 & 415 & 189 & 99 & 159 & 137 & 193 & 117 \\ \hline $ 142 & 38 & 293 & 415 & 189 & 99 & 159 & 137 & 193 & 117 \\ \hline $ 142 & 38 & 293 & 415 & 189 & 99 & 159 & 137 & 193 & 117 \\ \hline $ 142 & 38 & 293 & 415 & 189 & 99 & 159 & 137 & 193 & 117 \\ \hline $ 142 & 38 & 293 & 115 & 189 & 99 & 159 & 137 & 193 & 117 \\ \hline $ 142 & 38 & 293 & 115 & 189 & 99 & 159 & 137 & 193 & 117 \\ \hline $ 142 & 38 & 293 & 115 & 189 & 99 & 159 & 137 & 193 & 117 \\ \hline $ 142 & 38 & 214 & 315 & 316 & 317 & 318 & 317 \\ \hline $ 142 & 38 & 214 & 315 & 318 & 318 & 318 & 317 \\ \hline $ 142 & 38 & 214 & 315 & 318 & 318 & 318 & 318 \\ \hline $ 142 & 318 & 318 & 318 & 318 & 318 & 318 & 318 \\ \hline $ 142 & 318 & 318 & 318 & 318 & 318 & 318 & 318 & 318 & 318 \\ \hline $ 142 & 318 & 318 & 318 & 318 & 318 & 318 & 318 & 318 & 318 \\ \hline $ 142 & 318 & 3$ | 5016740407424177124456321286461282436236240075427473458674119126819714441116484130601234613011412239743718330044921823334713319431245724234910234819049260134194312457242349102348190492601351943124572423491023481904926013015289276721882645002986140214040340429647911033318547638237389327392212124093581455340620845440029423263265216387263362208455381294232632153524449336236245234625542144306182277238302179452388293415189299159137193110277 | 50167404074241771244563212864637128243623624007542747345867411251912681971444111648413060123461377301141223974371833004492182333472611341943124572423491023481904926020113519431245724234910234819049260201135194312457242349102348190402062011404034042667218826450298614021843164033022161103331854763823040431640340420611033318547638230403183273922112340326514538746512646636745440620621621621621621621621621621645440621621621621621621621621621621645440621621621621621 | $ \begin{array}{ c c c c c c c c c c c c c c c c c c c$ | 3
3 <th>2167404074241771244563212864637390244259282436236240075427473458674112532619830919126819714441116484130601236113773072721113011412239743718330044921823334726118720211130314122397437183300449218233347261187202111304141223974371833004492182334412452451113041412239743718330044921823334726118720211330414423437183160348190243244245145145304145247184246160348190245245245245245245314430246246246246246246246246246246246246314430449246246246246246246246246246246246246246345346<</th> <th>1144411144511113311</th> <th>114441144522111</th> <th>1144111145111</th> <th>1144111</th> <th>167 40 407 424 177 124 456 221 286 46 370 390 244 250 202 135 336 350 436 282 436 23 62 400 75 427 473 456 67 41 125 326 198 309 152 213 284 255 233 191 268 197 144 411 164 84 130 60 123 461 377 307 272 11 212 450 460 17 24 101 141 22 397 437 164 449 140 140 141 140 141 140 141 140 140 141 140 141 140 140 141 140 141 140 140 141 140 141 141 141 141 141 141 141 141 141 141 141 141 141 141 141 141 141</th> <th>100 100 400 400 424 170 124 456 321 286 46 37 300 244 250 102 135 136 350 480 480 282 436 23 62 400 75 427 473 458 67 41 125 326 198 102 120</th> <th>100 100 400 420 170 120 450 280 460 370 280 240 250 200 135 330 350 480 100 380 282 430 23 62 400 75 427 473 60 61 107 300 120 100 120</th> <th>100 100 400 420 170 124 456 321 280 46 370 390 244 250 130 336 350 480 380<th>100 100 400 420 1</th><th>100 100 400 420 1</th><th>50 167 40 407 424 177 124 456 21 286 47 390 244 250 202 135 336 350 480 380 274 462 394 393 282 436 23 62 400 75 427 435 61 41 26 307 272 410 210 260 26</th><th>5 167 40 407 424 177 124 456 321 286 6 37 30 24 250 135 336 359 400 108 83 274 426 337 300 301
301 301 <</th><th>50 167 40 407 424 177 124 456 321 286 37 390 244 250 213 386 350 480 108 83 274 462 394 303 314 282 436 23 62 400 75 427 436 67 41 125 326 197 121 213 243 255 351 453 267 248 337 439 430 430 430 <</th><th>5 167 40 407 424 177 124 456 321 286 37 390 244 250 135 336 350 480 108 83 274 462 394 240 301 341 341 282 436 23 62 400 75 473 480 41 420 41 420 41 410 <td< th=""><th>b 167 40 407 424 177 124 456 21 28 40 23 23 62 400 <</th><th>5 167 40 407 424 177 124 456 121 280 281 290 201 330</th><th>5 107 407 424 177 124 456 321 26 37 20 23 23 30 <th>50 60 40 407 424 177 124 456 321 64 37 30 24 250 30<</th><th>10 40 40 42 17 42 45 32 40 30 40 40 40 40 42 17 42 45 32 40 30 40</th><th>50 67 80 67 80 80 80 80 <</th><th>10 40 40 424 477 124 456 321 46 37 390 244 250 320
 320 <th< th=""><th>10 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 <</th><th>10 40 40 40 40 <</th></th<></th></th></td<></th></th> | 2167404074241771244563212864637390244259282436236240075427473458674112532619830919126819714441116484130601236113773072721113011412239743718330044921823334726118720211130314122397437183300449218233347261187202111304141223974371833004492182334412452451113041412239743718330044921823334726118720211330414423437183160348190243244245145145304145247184246160348190245245245245245245314430246246246246246246246246246246246246314430449246246246246246246246246246246246246246345346< | 1144411144511113311 | 114441144522111 | 1144111145111 | 1144111 | 167 40 407 424 177 124 456 221 286 46 370 390 244 250 202 135 336 350 436 282 436 23 62 400 75 427 473 456 67 41 125 326 198 309 152 213 284 255 233 191 268 197 144 411 164 84 130 60 123 461 377 307 272 11 212 450 460 17 24 101 141 22 397 437 164 449 140 140 141 140 141 140 141 140 140 141 140 141 140 140 141 140 141 140 140 141 140 141 141 141 141 141 141 141 141 141 141 141 141 141 141 141 141 141 | 100 100 400 400 424 170 124 456 321 286 46 37 300 244 250 102 135 136 350 480 480 282 436 23 62 400 75 427 473 458 67 41 125 326 198 102 120 | 100 100 400 420 170 120 450 280 460 370 280 240 250 200 135 330 350 480 100 380 282 430 23 62 400 75 427 473 60 61 107 300 120 100 120 | 100 100 400 420 170 124 456 321 280 46 370 390 244 250 130 336 350 480 380 <th>100 100 400 420 1</th> <th>100 100 400 420 120
 120 1</th> <th>50 167 40 407 424 177 124 456 21 286 47 390 244 250 202 135 336 350 480 380 274 462 394 393 282 436 23 62 400 75 427 435 61 41 26 307 272 410 210 260 26</th> <th>5 167 40 407 424 177 124 456 321 286 6 37 30 24 250 135 336 359 400 108 83 274 426 337 300 301 <</th> <th>50 167 40 407 424 177 124 456 321 286 37 390 244 250 213 386 350 480 108 83 274 462 394 303 314 282 436 23 62 400 75 427 436 67 41 125 326 197 121 213 243 255 351 453 267 248 337 439 430 430 430 <</th> <th>5 167 40 407 424 177 124 456 321 286 37 390 244 250 135 336 350 480 108 83 274 462 394 240 301 341 341 282 436 23 62 400 75 473 480 41 420 41 420 41 410 <td< th=""><th>b 167 40 407 424 177 124 456 21 28 40 23 23 62 400 <</th><th>5 167 40 407 424 177 124 456 121 280 281 290 201 330</th><th>5 107 407 424 177 124 456 321 26 37 20 23 23 30 <th>50 60 40 407 424 177 124 456 321 64 37 30 24 250 30<</th><th>10 40 40
42 17 42 45 32 40 30 40 40 40 40 42 17 42 45 32 40 30 40</th><th>50 67 80 67 80 80 80 80 <</th><th>10 40 40 424 477 124 456 321 46 37 390 244 250 320 <th< th=""><th>10 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 <</th><th>10 40 40 40 40 <</th></th<></th></th></td<></th> | 100 100 400 420 1 | 100 100 400 420 1 | 50 167 40 407 424 177 124 456 21 286 47 390 244 250 202 135 336 350 480 380 274 462 394 393 282 436 23 62 400 75 427 435 61 41 26 307 272 410 210 260 26 | 5 167 40 407 424 177 124 456 321 286 6 37 30 24 250 135 336 359 400 108 83 274 426 337 300 301
 301 301 301 301 301 301 301 < | 50 167 40 407 424 177 124 456 321 286 37 390 244 250 213 386 350 480 108 83 274 462 394 303 314 282 436 23 62 400 75 427 436 67 41 125 326 197 121 213 243 255 351 453 267 248 337 439 430 430 430 < | 5 167 40 407 424 177 124 456 321 286 37 390 244 250 135 336 350 480 108 83 274 462 394 240 301 341 341 282 436 23 62 400 75 473 480 41 420 41 420 41 410 <td< th=""><th>b 167 40 407 424 177 124 456 21 28 40 23 23 62 400 <</th><th>5 167 40 407 424 177 124 456 121 280 281 290 201 330</th><th>5 107 407 424 177 124 456 321 26 37 20 23 23 30 <th>50 60 40 407 424 177 124 456 321 64 37 30 24 250 30<</th><th>10 40 40 42 17 42 45 32 40 30 40 40 40 40 42 17 42 45 32 40 30 40</th><th>50 67 80 67 80 80 80 80 <</th><th>10 40 40 424 477 124 456 321 46 37 390 244 250 320
 320 <th< th=""><th>10 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 <</th><th>10 40 40 40 40 <</th></th<></th></th></td<> | b 167 40 407 424 177 124 456 21 28 40 23 23 62 400 < | 5 167 40 407 424 177 124 456 121 280 281 290 201 330 | 5 107 407 424 177 124 456 321 26 37 20 23 23 30 <th>50 60 40 407 424 177 124 456 321 64 37 30 24 250 30<</th> <th>10 40 40 42 17 42 45 32 40 30 40 40 40 40 42 17 42 45 32 40 30 40</th> <th>50 67 80 67 80 80 80 80 <</th> <th>10 40 40 424 477 124 456 321 46 37 390 244 250 320
 320 <th< th=""><th>10 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 <</th><th>10 40 40 40 40 <</th></th<></th> | 50 60 40 407 424 177 124 456 321 64 37 30 24 250 30< | 10 40 40 42 17 42 45 32 40 30 40 40 40 40 42 17 42 45 32 40 30 40 | 50 67 80 67 80 80 80 80 < | 10 40 40 424 477 124 456 321 46 37 390 244 250 320 <th< th=""><th>10 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 <</th><th>10 40 40 40 40 <</th></th<> | 10 40
40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 < | 10 40 40 40 40 < |

17 Tales of The Secret Bunker 2023 – Furgal, Bascom, Bjorklund

- An online utility that puts rows into a physical order that matches order of one index
- Operates on a table (or part of table) at a time
 - Removes Record Fragmentation (where possible)
 - Removes Physical Scatter
 - Removes Logical Scatter





Holes after deleting some records



RM chain is a list of blocks that have usable space for additional records

								5	22																		
												12						17									
								23													16						
7	25	39	14	30	47	35						33					2	28									
	24	38		40	36	4			10	41									6								
13	42	8	3	46		34			32	15						21											
37	43	20	45	19	44	31													9	29	1						
											26	11	27	18											46	47	48

Numbers represent the order of the RM chain



21 Tales of The Secret Bunker 2023 – Furgal, Bascom, Bjorklund

7	25	39	14	30	47	35
	24	38		40	36	4
13	42	8	3	46		34
37	43	20	45	19	44	31

			_	_	_	_
5	22					
				12		
23						
				33		
	10	41				
	32	15				
			26	11	27	18

		17				
					16	
	2	28				
			6			
21						
			9	29	1	

		46	47	48



tablereorg moves records to establish a NEW physical ordering to match order of index you specify

Record one, two, etc move to a new cluster





New cluster fills, old cluster available for reuse





Available clusters are reused





When completed there are no "holes"

									ľ	f														
											[



Now let's try it out



27 Tales of The Secret Bunker 2023 – Furgal, Bascom, Bjorklund

Created 6 scenarios with

- ATM Database
 - Accounts Table
 - 10,000,000 records
 - 1,000,000,000 records
 - 1,000,000,000 records varying size from 100 to 5000 bytes
 - 500,000,000 records
- Demo database
 - Customer table
 - 33 records



Scenario 0: starting conditions

- Regular ATM database but small (only 10,000,000 account records)
- Unused after building
- All records a little over 100 bytes long
- Database is about 1.5 GB
 - 371,020 blocks (4k)
- No fragmented records
- 94% RM space utilization
- No free clusters
- 3 blocks on account RM chain



Scenario 0: table reorg 1

- 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 2

- 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

- Database size is now 14 GB
- Again unused after building
- There are 3,752,908 blocks
- No free clusters
- 2 blocks on the RM chain
- Zero fragmentation

Scenario 1: table reorg 1

- proutil atm -C tablereorg account recs 1000
- Took 72 min 54 sec.
- Database grew from 14 GB to 22.2 GB
 - From 3,752,908 to 5,815,500 blocks (4k)
- No fragmented records
- 60% RM space utilization
- No free clusters
- 3,332,660 blocks on the RM chain



Scenario 1: table reorg 2

- proutil atm -C tablereorg account recs 1000
- Took 67 min 12 sec.
- Database grew from 22.2 GB to 22.9 GB
 - From 5,815,500 to 5,998,028 blocks (4k)
- No fragmented records
- 61.2% RM space utilization
- No free clusters
- 5,395,252 blocks on the RM chain



Scenario 1: table reorg 3

- No change in database size
- 16 additional blocks on RM chain
- Nothing much happened
 - records moved but otherwise result was the same

Scenario 2: Make account records vary in size

- 100,000,000 accounts
- Record size varies from ~100 to ~500 bytes
- 1 in 20 records is ~100 to ~1500 bytes
- Database size is 9,916,940 blocks 37 GB
- RM chain has 2,021,146 blocks
- There are 190,872,537 fragments
- 95.3 % RM space utilization

Scenario 2: table reorg 1

- Took 86 min 35 sec
- Database grew from 37 GB to 53 GB
- From 9,916,940 to 13,003,212 blocks
- Zero fragmentation
- 71.84% RM space utilization
- RM chain has gone from 2,021,146 to 11,262,187 blocks



- Create an Account record, then create another account record with the account.id x -1.
 - Total 1,000,000,000 records
 - 1, -1, 2, -2, 3, -3, 4, -4, etc
- Delete all negative accounts
 - 1, empty, 2 empty, 3, empty, 4, empty, etc
 - Total 500,000,000 records

Scenario 4

- Create an Account record, assign the account.id x -1
- Once complete, create Account records as normal
- Result
 - Total 1,000,000,000 records
 - -1,-2,-3, -499,999,999, -500,000,000,1,2,3,4, etc
- Delete all negative accounts
 - Empty, empty,, 1,2,3,4, etc
 - Total 500,000,000 records



Warning: Rathole ahead

How can you delete 500 MM records quickly?



Simplest way to delete 500 MM records:

for each account: delete account. end.

Google leads us to a better way



Progress community article P36834 says to do this:

```
define var recNum as int no-undo.
define var tranSize as int no-undo initial 10000.
```

```
recNum = 0.
outer:
do while true transaction:
    for each account exclusive-lock:
        recNum = recNum + 1.
        delete account.
        if ((recNum modulo tranSize) eq 0) then next outer.
    end.
    leave.
end.
```

Instead of 500 MM transactions 50,000 transactions.

This worked great --- until it didn't. UNIQUE index on account.id caused deleted index entry place-holders.

Eventually minutes went by after every 10,000 deletes to skip over these place-holders.



FOR EACH account: DELETE account. END.

500 MM transactions. This worked best.



FOR EACH account: DELETE account. END.

500 MM transactions. This worked best.

Here, 1 FOR EACH. The 10,000 records / tx code does 50,000 FOR EACH.



Rathole concluded



Scenario 3 – Interleaved deleted rows

Tablereorg with all defaults

- proutil atm –C tablereorg account
- Took 10:52:03 to complete

- Tablereorg with 1,000 records per transaction
 - proutil atm –C tablereorg account recs 1000
 - Took 10:40:18 to complete

- The database size grew from 126 GB to 171 GB
- Why? Let's investigate
 - Accounts Area
 - Records Per Block: 64
 - Data Cluster Size: 512
 - Records: 500,000,000
 - Average size: 124 bytes
 - Database Blocksize: 8192
 - How many records fit in a block?
 - 8192 (128ish) = 8000ish / 124 bytes = 64 records should fit in a block



- A freshly loaded database has an accounts area of 65,007,517,696 bytes
 - 512 8K blocks per cluster = 15,499 clusters.
- The database prior to the table reorg has an accounts area of 130,036,006,912 bytes
 - 512 8K blocks per cluster = 31,003 clusters.
- The tablereorg database has an accounts area of 178,115,313,664 bytes
 - 512 8K blocks per cluster = 42,466 clusters.
- Tablereorg added 11,463 Data Clusters



QUIZ !

Why are there

11,463 additional clusters

after the reorg?



52 Tales of The Secret Bunker 2023 – Furgal, Bascom, Bjorklund

2 Issues Found: Area Size Growth 2 Hour Startup Time

Let's Try Scenario 4

A large chunk of empty space followed by contiguous records



54 Tales of The Secret Bunker 2023 – Furgal, Bascom, Bjorklund

Scenario 4 – Large chunk of free space

- Time to complete with 1,000 rows per transaction
 - 7:47:44
- Database Size before Reorg:
 - 126 GB
- Database size after Reorg:
 - 126 GB
- Account Area 130,036,006,912 bytes
 - 512 8K blocks per cluster = 31,003 clusters.
- No additional Data Clusters added this time



What about the startup time?



```
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 ]
```



searchdepth

Specifies the percentage of the record free chain to search for free blocks to be used during the table reorganization. The minimum is 0, the maximum is 100, and the default is 100.

- proutil atm -C tablereorg account recs 1000 searchdepth 5
 - Since the record free chain (RM chain) was large it spent all the startup time scanning
 - Before table reorg, there were 31,003 Data Clusters.
 - A fresh Dump/Load database has 15,499 Data Clusters.
 - This leaves 15,504 empty clusters, or 7,938,048 blocks to scan



Scenario 4: What about Performance Impact?



Modified ATM benchmark

- 150 Users
- 7 Users doing updates as normal ATM activity
 - Update Account, Teller, and Branch Balance
 - Create History Record
- 143 Users doing reading
 - Randomly Read 101 Account Record
 - Read a Teller and Branch Record

ATM Results

	Baseline wi	ith 150 users	
Trans Per	Records	Records	Longest
Sec	Read	Updated	Wait
258.00	68,878.00	8,381.00	4.30
184.60	45,849.00	9,383.00	4.20
176.10	43,032.00	9,657.00	4.10
174.60	43,240.00	8,988.00	3.90

During Tablereorg

Trans Per	Records	Records	Longest
Sec	Read	Updated	Wait
141.90	38,072.00	4,341.00	146.60
135.30	37,667.00	2,786.00	27.30
135.70	38,483.00	2,081.00	72.30
139.40	39,106.00	2,555.00	103.00



Final Test – Compared to a traditional Dump/Load

- proutil atm -C tablereorg account recs 1000 searchdepth 5
 - 8 hours, 40 minutes to complete
- Binary Dump/Load
 - Dump: 43 minutes
 - Load 24 minutes
 - Index Rebuild: 8 minutes
 - Total Time: 1 hour, 15 minutes
- Binary Dump/Load
 - Must disable AI, then re-enable
 - Must disable OE Replication, then re-enable and rebaseline
- Tablereorg allows AI and OE Replication to stay intact



500,033 customer records from the demo database

	Time	Al Size	AI Notes	Avg Note Size
Partitioning	2:07	2,263,482,368	10,543,724	214.68
Table Reorg	1:25	2,063,204,352	7,643,092	269.94
Buffer copy	2:00	2,123,497,472	9,043,655	234.81
Dump/Load	0:46	N/A	N/A	N/A



Lastly, something a bit different!

Try this at home:

```
mkdir homework
cd homework
prodb foo isports
echo a foo.al >addai.st
prostrct add foo addai.st
probkup foo /dev/null
rfutil foo -C aimage begin
proutil foo -C dbanalys >foo.ana1
proutil foo -C tablereorg customer recs 1000
proutil foo -C dbanalys >foo.ana2
diff -Bw foo.ana1 foo.ana2
rfutil foo -a foo.a1 -C aimage scan verbose >aiscan.txt
more aiscan.txt
```



So what can tablereorg do for you?



Tablereorg will attempt to

- Minimize record fragmentation
- Make physical order align with index key order
 - For one index you choose
 - Rowid order and key order will be the same (as much as feasible)
 - Choose wisely (default is primary key)
- Performance will improve for some portions of the application
- Operates on one table at a time

Tablereorg will also

- Move all records
 - ROWID's will change
 - All indexes for the table will be updated
- Table's storage area will expand (if not enough free clusters)
 - Previous record locations become free RM space
 - Might fail if data area fills or reaches max size
- May expand RM chain
 - Depending on
 - Available free clusters
 - Existing RM chain
 - Searchdepth setting



- Tablereog is a useful tool to remove fragmentation and scatter
- Appears to be pretty fast
- The database may increase in size depending on how contiguous free space is organized
- Should be run at off hours when the user load is not high
- TEST before using in production !!!
- Can be restarted if something goes wrong
- Be careful of the defaults to limit startup time
- As always YMMV
 - Transportation, meals, and accommodation not included



Questions





research from the parmington foundation