Index Utilities Tips and Tricks

Paul Koufalis – White Star Software

James Palmer – Consultingwerk

Paul Koufalis

- Progress DBA and UNIX admin since 1994
- Expert OpenEdge technical consulting
- Wide range of experience
  - Small 10 person offices to 3500+ concurrent users
  - AIX, HPUX, Linux, Windows…if Progress runs on it, I've worked on it
- Father to these two monkeys

  - pk@wss.com

White Star software

James Palmer

- Senior Consultant / Database Administrator
- Working in the OpenEdge world since 2003
- Varied experience across a variety of industries and applications as a developer and more recently as a DBA
- Chairman of the UK & Ireland PUG, Director for EMEA PUG Challenge Ltd. and speaker on a variety of topics both at PUG Challenge events, and smaller conferences in the UK and USA

Agenda

- Introduction
- Online idxactivate
- Idxbuild
- Fix it!

Assumptions / Prerequisites

- OpenEdge 11.7 or 12
  - 90% will work in 10.2B08
- Type II Storage Areas
- Tables, indexes, lobs in distinct separate areas

White Star software

What is an index?

- Public Service Announcement for anyone too shy to ask
- Simplest form: A tree of all the rows in a table

Start at the root and follow a path of branches until you get to a leaf (i.e. a row in the DB)

Indexes are used for

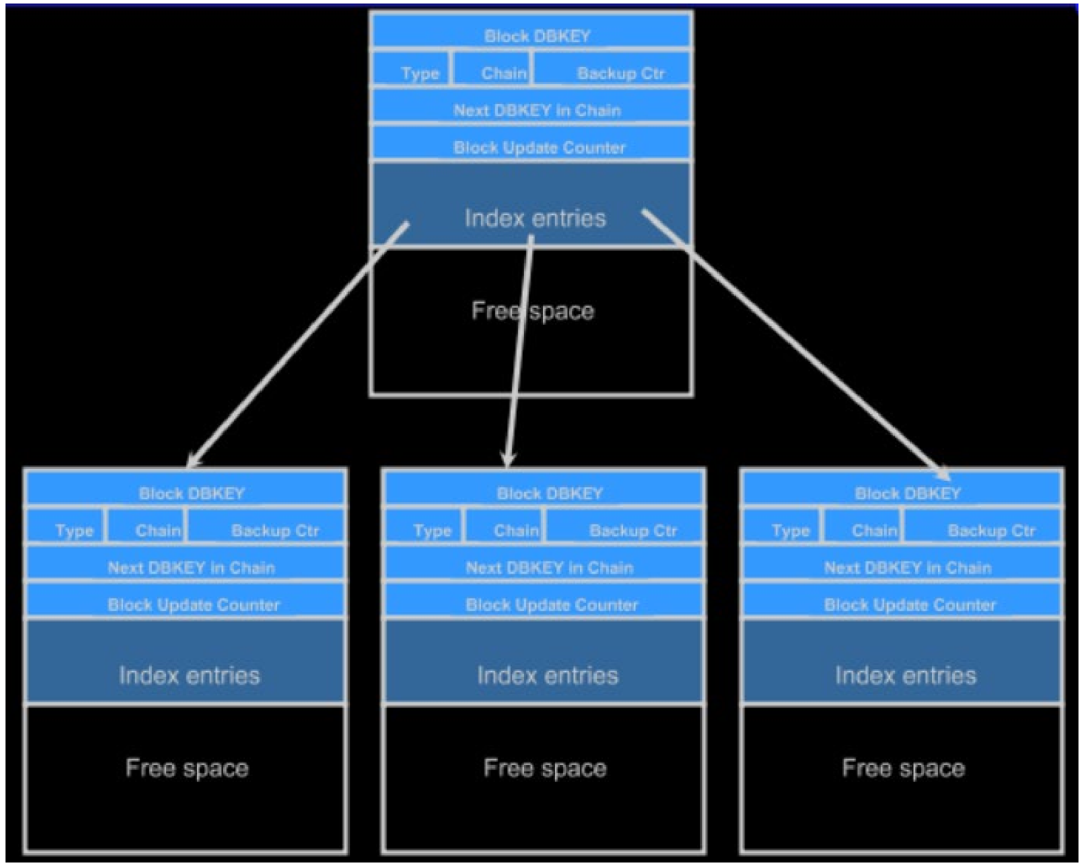
- Quick access to a row or set of rows
- To retrieve rows in a specific order
- To enforce uniqueness of columns
- To locate rows that contain a word or a phrase

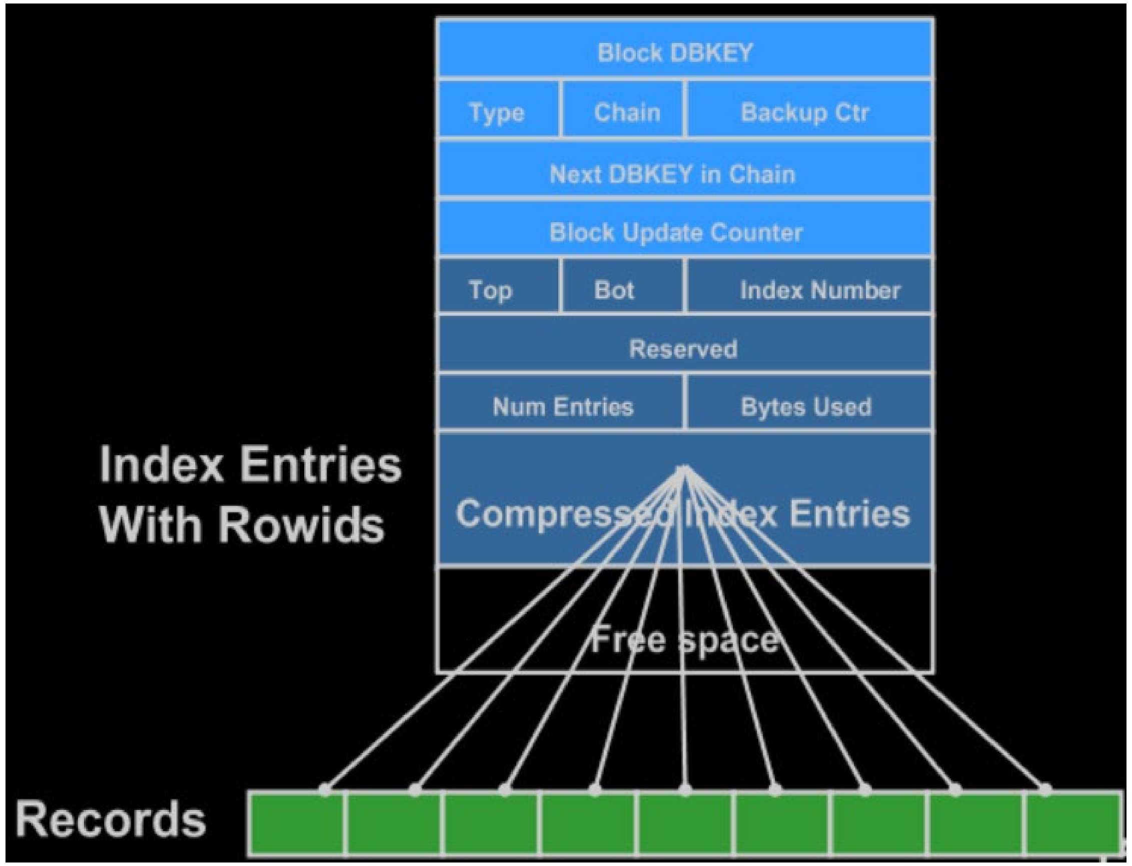
Records have a rowid

- Unique 64-bit identifier for a row in a table
  - PartitionId, Blocknumber in area, Row in Block
- Encoding of the “physical” address
  - Used to locate a record quickly
- Constant for life of record
  - Until you delete it or change partition key

- The “leaf” contains the ROWID

Index Structure – Simple B-Tree





White Star software

Indexes Need TLC

- Reading an index does no harm
- Writing to an index may lead to:
  - **Empty space in blocks**: most common issue
  - **Increased Levels**: The path from root to branch (…to branch to branch…) to leaf gets longer and longer
  - **Deleted Row Placeholders**
    - Can’t delete unique leaf until tx commits
  - **Corruption**
    - I know I know. Impossible.

What maintenance utilities are there?

- idxactivate
- idxanalys
- idxbuild
- idxcheck
- idxcompact
- idxdeactivate
- idxfix
- idxmove

White Star software

idxactivate

- Uses an existing active index to activate another
- Online or offline
- No need to deactivate AI and Replication !!

- More later

idxanalys

```
INDEX BLOCK SUMMARY FOR AREA “Package Index" : 10
-------------------------------------------------------------------------------
Table                      Index  Fields Levels  Blocks     Size  % Util Factor
PUB.PACKAGE
  I-ACCES                     15       6      4   75208   171.3M    58.7    1.8
  I-SERIES                    16       3      4  200752   505.5M    64.9    1.7
  I-PKG                       14       1      3   45383   173.1M    98.3    1.0
  I-RECEIVED                  19       3      3   34716    80.0M    59.4    1.8
  I-DELIVERED                 20       4      3   17440    36.3M    53.7    1.9
  I-WAREHOUSE                 21       5      4   54106   112.6M    53.7    1.9
  I-WHS-ZIP                   22       4      3   11129    21.2M    49.1    2.0
  I-ZAP                       23       3      3    9309    18.8M    52.0    2.0

                    -----------------------------------------------------------
Totals:                                          521342     1.3G    63.5    1.7
```

idxbuild

- Deletes indexes and rebuilds them from scratch
- Many reasons to do this

- “New” options and performance improvements since 10.2B08
    - People still don’t know about them

- More later

idxcheck

- Checks indexes for corruption
- Helps diagnose problems
- Online or offline
- Errors and warnings to screen and logfile

idxcompact

- Compacts indexes
- Used when space utilization (idxanalys) is low
- Reduces blocks in B-Tree, maybe even B-Tree levels
  - Less blocks and levels => faster queries
- Online or offline
- Clears deleted index placeholders
- Can be run in parallel for different partitions of the same index
- Users can work as normal
- No record or table locks
- No other administration tools on the index though

White Star software

idxfix

- Checks for index corruption
- Checks for missing or incorrect index for a record
- Repairs corrupted indexes
- Online or Offline
- Will enable verified indexes when run offline
- Has several sub options

idxfix

1. Scan records for missing index entries with index block validation.
2. Scan indexes for invalid index entries.
3. Both 1 and 2 above.
4. Cross-reference check of multiple indexes for a table.
5. Build indexes from existing indexes.
6. Delete one record and its index entries.
8. Scan records for missing index entries.

- SPOT THE INTENTIONAL “MISTAKE”
  - 7 Used to be quit, until option 8 came along

idxmove

- Move an index from one data area to another
- Online or offline
- Table is locked with a share lock while this happens
- Run at a quiet time in the system to avoid problems for users

dbrpr

- Undocumented and unsupported
- Could cause serious damage - caution
- Progress recommend contacting Tech Support before using the tool
- There are some reporting options that are safe so long as you
  - Turn off AI
  - Truncate BI
- Option 2 to test one or more indexes

Agenda

- Introduction
- **Online idxactivate**
- Idxbuild
- Fix it!

[Online] Index Activate

- "Officially" you can activate an index online and use it
- "Unofficially", it's a little more complicated …

- Who already uses this functionality?

[Online] Index Activate

- Uses an existing active index to activate another
  - Primary index by default
- Preferable to use a unique index otherwise will lead to extra locking
- Can specify transaction size (num records)
- Can be used on specific tenants or partitions if relevant

Benefits

- Can run multiple online idxactivate concurrently
- Do not need to shut down the database
- Do not need to deactivate after imaging
- Do not need to baseline OE Replication

- For customers with large databases and distant DR sites, the OE Repl benefit is HUGE

Inconveniences

- Sloooooooowwwwwwww
- Lots of AI/BI activity
- Users need to reload their schema cache
  - See –usernotifytime
- No CUD activity during idxactivate
- Things get wonky if there is no active primary index
- Some lightly documented GOTCHAs
  - That’s why you’re here, right?

Basic Requirements

- OpenEdge 10.2B08, 11.2.1, 11.3.0 or 12.x
  - Before that (10.1A+) there were significant bugs
- An active primary index
  - Again – there are scenarios where this is not officially required
- An inactive, non-primary index to activate

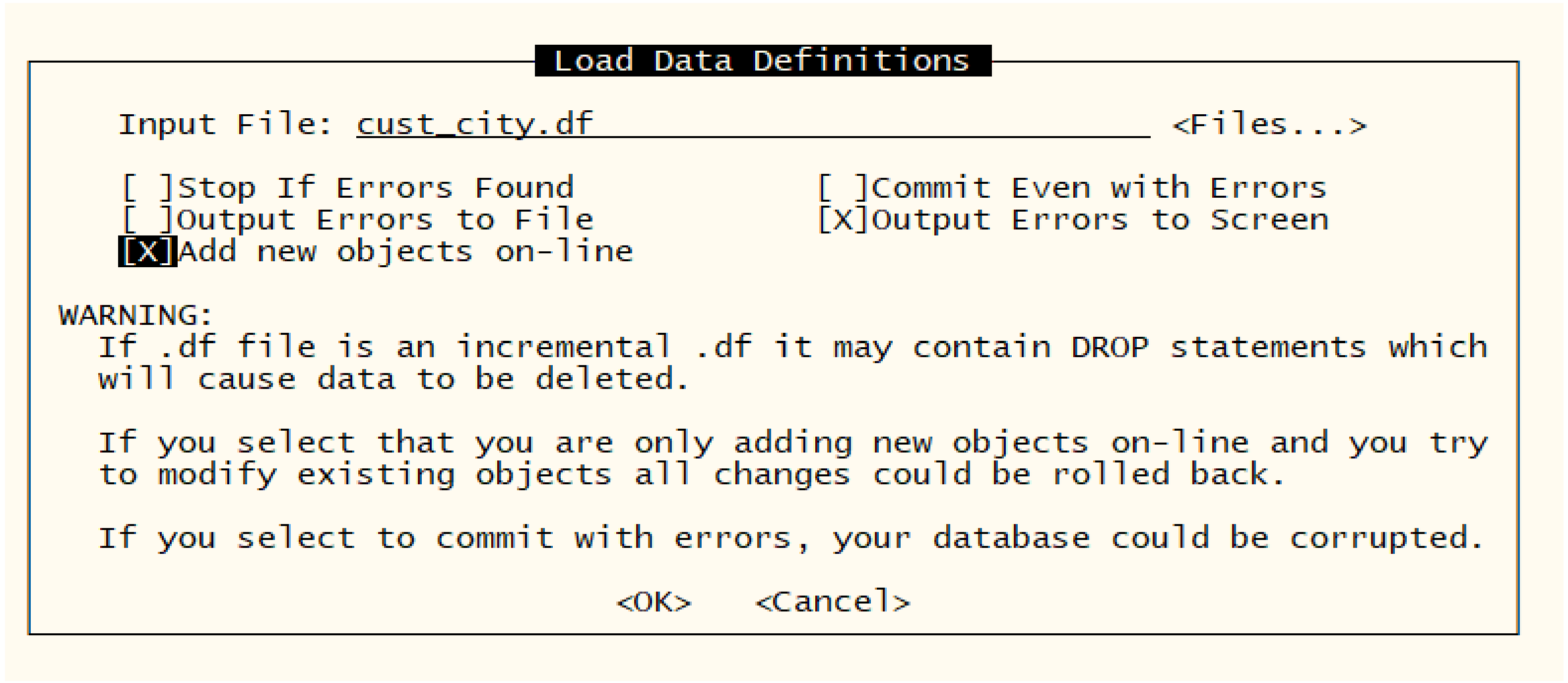
Step 1 – Create DF

- Create a DF with the new index
  - Must be INACTIVE

```
ADD INDEX "City" ON "Customer"
  AREA "Customer Index Area"
  INACTIVE
  INDEX-FIELD "City" ASCENDING ABBREVIATED
```

Step 2 – Load the DF

- Option 1: Data Dictionary
  - Optionally select “Add new objects on-line” if other users connected



```
                          Load Data Definitions

   Input File: cust_city.df                          <Files...>

   [ ]Stop If Errors Found              [ ]Commit Even with Errors
   [ ]Output Errors to File             [X]Output Errors to Screen
   [X]Add new objects on-line

 WARNING:
   If .df file is an incremental .df it may contain DROP statements which
   will cause data to be deleted.

   If you select that you are only adding new objects on-line and you try
   to modify existing objects all changes could be rolled back.

   If you select to commit with errors, your database could be corrupted.

                             <OK>    <Cancel>
```

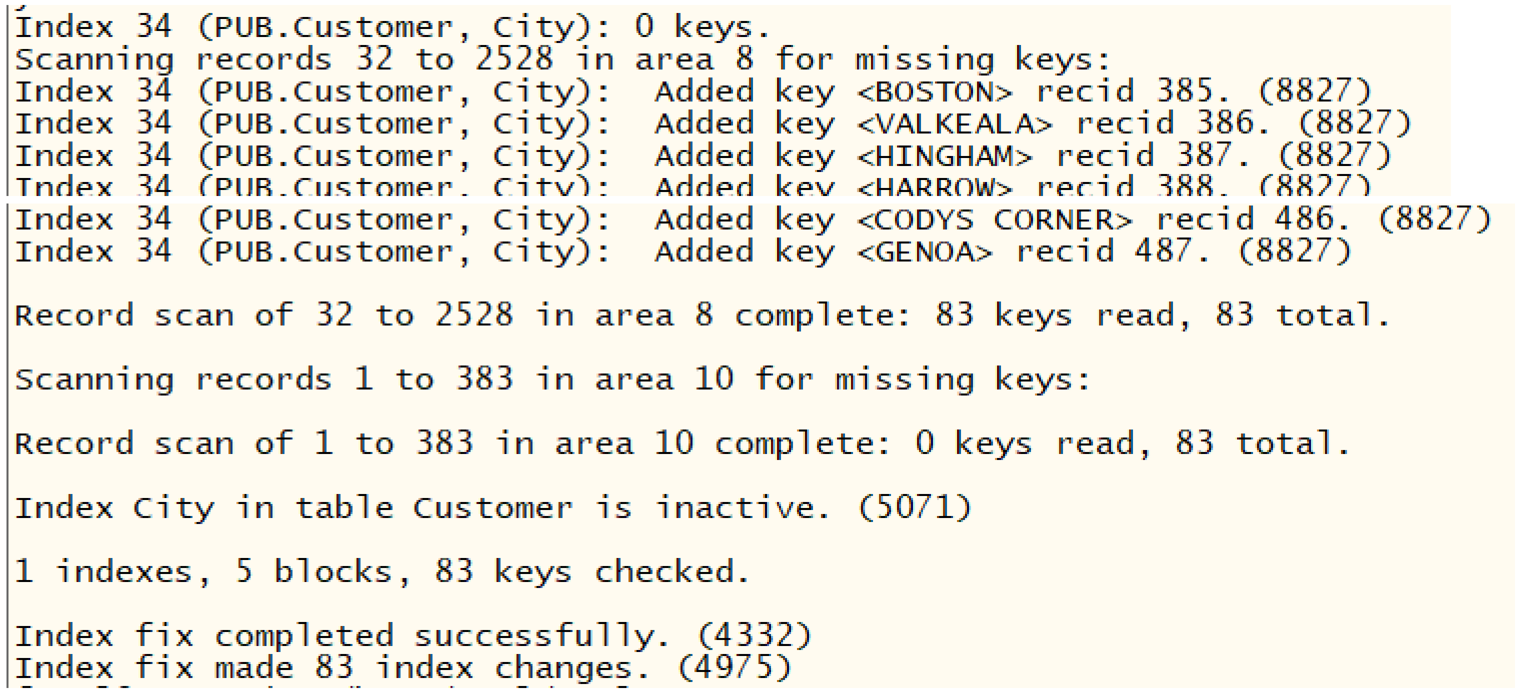
Step 2 – Alternate Option

- run prodict/load_df.p

```
/* CSV params are "DF file,force commit,OSC"
 * For Online Schema Change (OSC), entry must be
 * "" or "NEW OBJECTS"
*/
run prodict/load_df.p (dfFile + ",NO,NEW OBJECTS").
```

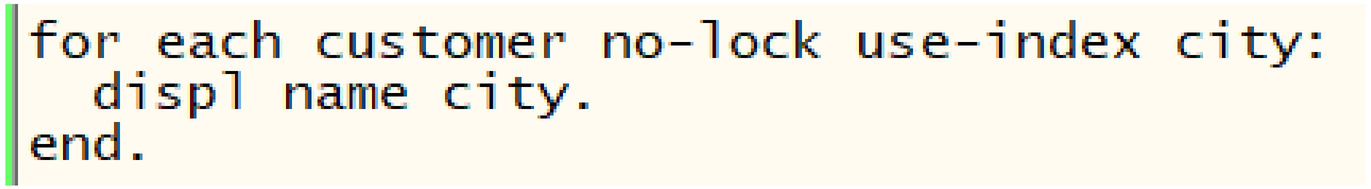
[Optional] Step 3 – Pre-build the index keys

- Use proutil idxfix – option 3 to pre-build the index keys

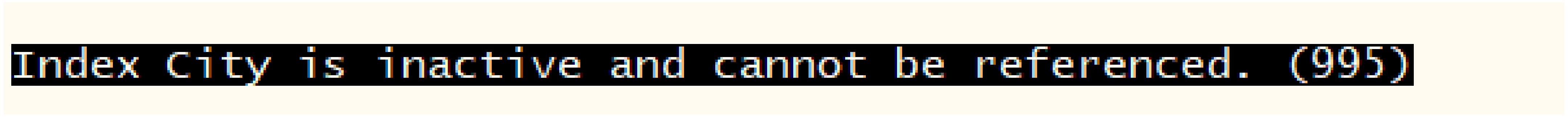


```
Index 34 (PUB.Customer, City): 0 keys.
Scanning records 32 to 2528 in area 8 for missing keys:
Index 34 (PUB.Customer, City):  Added key <BOSTON> recid 385. (8827)
Index 34 (PUB.Customer, City):  Added key <VALKEALA> recid 386. (8827)
Index 34 (PUB.Customer, City):  Added key <HINGHAM> recid 387. (8827)
Index 34 (PUB.Customer, City):  Added key <HARROW> recid 388. (8827)
Index 34 (PUB.Customer, City):  Added key <CODYS CORNER> recid 486. (8827)
Index 34 (PUB.Customer, City):  Added key <GENOA> recid 487. (8827)

Record scan of 32 to 2528 in area 8 complete: 83 keys read, 83 total.

Scanning records 1 to 383 in area 10 for missing keys:

Record scan of 1 to 383 in area 10 complete: 0 keys read, 83 total.

Index City in table Customer is inactive. (5071)

1 indexes, 5 blocks, 83 keys checked.

Index fix completed successfully. (4332)
Index fix made 83 index changes. (4975)
```

CAREFUL: Index is not yet active!



```
for each customer no-lock use-index city:
  displ name city.
end.
```



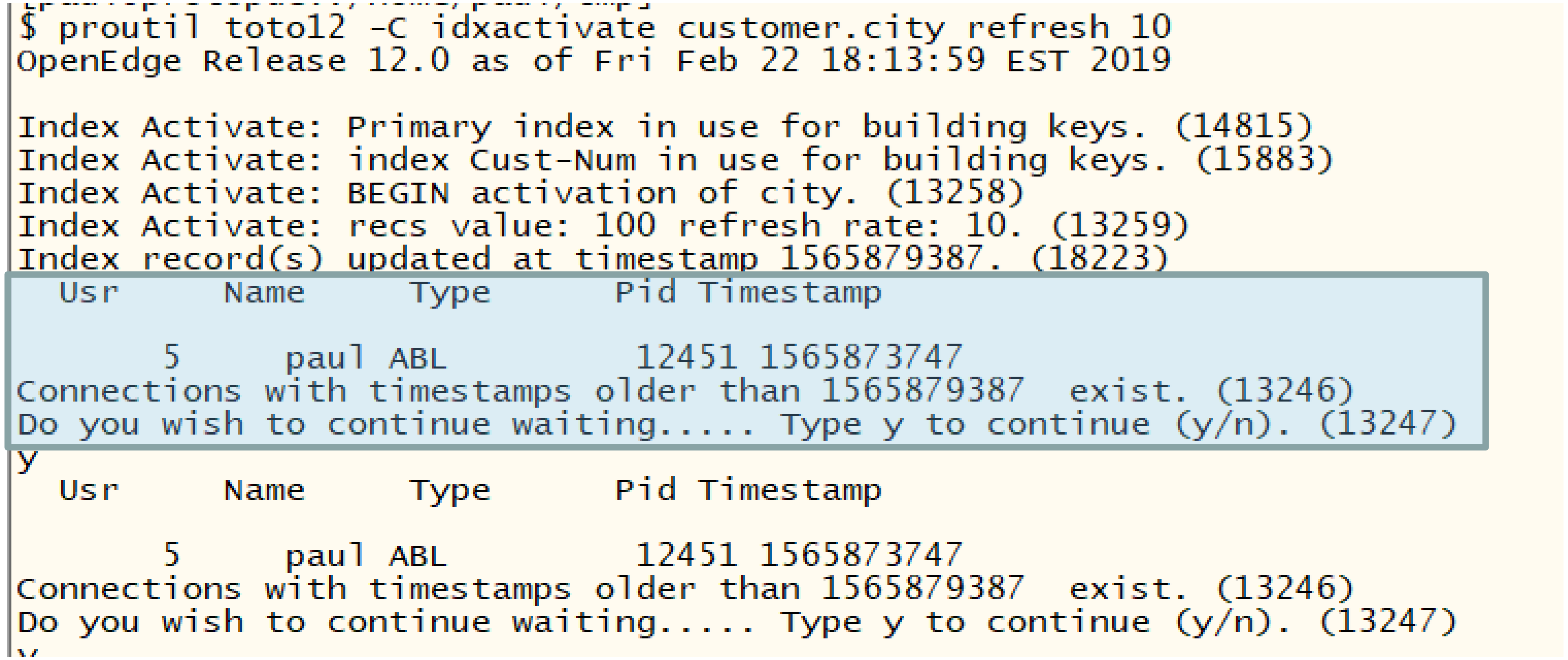
```
Index City is inactive and cannot be referenced. (995)
```

Step 4 – Activate the index

```
_proutil dbName –C idxactivate customer.city
```

- Optional Params:
    - recs n: number of records to process per tx. Defaut 100.
    - refresh t: frequency to update display of clients blocking activate
        - Have earlier schema (see –usernotifytime)
        - Default 60. Max 300.

GOTCHA #1: User schema time stamp



```
$ proutil toto12 -C idxactivate customer.city refresh 10
OpenEdge Release 12.0 as of Fri Feb 22 18:13:59 EST 2019

Index Activate: Primary index in use for building keys. (14815)
Index Activate: index Cust-Num in use for building keys. (15883)
Index Activate: BEGIN activation of city. (13258)
Index Activate: recs value: 100 refresh rate: 10. (13259)
Index record(s) updated at timestamp 1565879387. (18223)
  Usr      Name      Type      Pid Timestamp

       5      paul  ABL          12451 1565873747
Connections with timestamps older than 1565879387  exist. (13246)
Do you wish to continue waiting..... Type y to continue (y/n). (13247)
y
  Usr      Name      Type      Pid Timestamp

       5      paul  ABL          12451 1565873747
Connections with timestamps older than 1565879387  exist. (13246)
Do you wish to continue waiting..... Type y to continue (y/n). (13247)
```

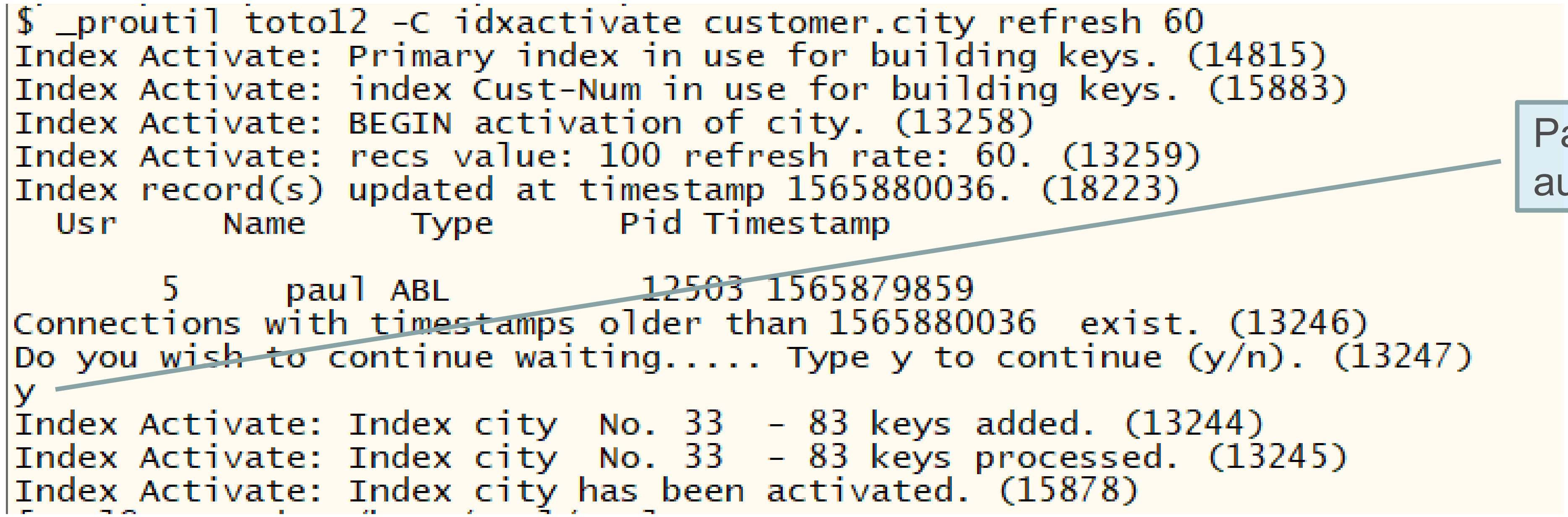
User Schema Time Stamp

- As of 11.7, use –usernotifytime DB startup parameter
  - How often client polls for notification of schema change
  - Default 0 (no polling) max 86,400 (24h)

- Try again with –usernotifytime 30

User Schema Time Stamp

- A few seconds after the initial “y” response, the index was activated



```
$ _proutil toto12 -C idxactivate customer.city refresh 60
Index Activate: Primary index in use for building keys. (14815)
Index Activate: index Cust-Num in use for building keys. (15883)
Index Activate: BEGIN activation of city. (13258)
Index Activate: recs value: 100 refresh rate: 60. (13259)
Index record(s) updated at timestamp 1565880036. (18223)
  Usr      Name      Type      Pid Timestamp

       5      paul ABL          12503 1565879859
Connections with timestamps older than 1565880036  exist. (13246)
Do you wish to continue waiting..... Type y to continue (y/n). (13247)
y
Index Activate: Index city  No. 33  - 83 keys added. (13244)
Index Activate: Index city  No. 33  - 83 keys processed. (13245)
Index Activate: Index city has been activated. (15878)
```

Pause here, then automatic activation

Still Not Perfect

- If you script this, need to detect “Do you wish to continue waiting…” and respond “y”
  - But watch out that you don’t loop indefinitely
- In 11.7.3, I have seen –usernotifytime not be effective
  - I.e. the “Do you wish to continue waiting…” looped much longer than the –usernotifytime
  - Must be a bug

GOTCHA #2: Detecting inactive indexes

- Do not use _index._active:
  - “Add new objects online” DF load sets this to yes
- Use _storageobject._object-state

```
for each _storageobject no-lock where _object-type = 2
                                  and _object-state = 1:

  find _index where _idx-num = _object-number no-lock no-error.
  if avail _index then find _file of _index no-lock no-error.

    /* do stuff */

end.
```

White Star software

GOTCHA #3: Primary indexes

- Cannot idxactivate a primary index
- Two options for primary indexes:

1. Load the DF without the "INACTIVE" property
   - Index will be built immediately
2. Use proutil db –C idxbuild
   - Requires disable AI and replication

GOTCHA #4: AI/BI Activity

- Be prepared for a potential increase in AI and BI activity
  - Disk space utilization
  - OE Replication vs. PICA buffers
  - System load

GOTCHA #5: New r-code

- Index is available for compile BEFORE idxactivate
  - By design
- Running source code may cause the still inactive index to be selected
  - Idem for building r-code
- Option: use –noinactiveindex client startup parameter
  - Compiler ignores inactive indexes

- When do you deploy new r-code?
  - After idxactivate completes

Online Index Activate Results

- My original idea was to use the same DB for Lab #1 and lab #2 (index rebuild)
- One table, 18.5M records
- Idxfix phase took 5+ hours to add keys
- Db.lg grew to 3.2 GB (one line per record!)

Agenda

- Introduction
- Online idxactivate
- Idxbuild
- Fix it!

Index Rebuild

- Offline only
- Destructive: it wipes and deactivates indexes before rebuilding them

- Block level operation not row level operation
    - => cannot use AI => deactivate AI and OE Replication
    - This is a big deal for large databases

Index Rebuild: three phases

1. Scan data areas
   - Hugely disk I/O bound with CPU for sort
   - Build keys and insert into temp buffers
   - Sort full buffers
2. Sort and merge
   - Hopefully all in memory: all CPU, no disk
3. Create index b-tree on disk
   - Read sorted list and insert keys into new index
   - Hugely disk I/O bound

White Star software

FAST Index Rebuild

- Make each phase faster!

- 10.2B08+. If you’re running anything older you have bigger problems
- MAXIMIZE use of system resources (CPU, mem, disk)
- MINIMIZE execution time

- There are no global optimal settings – depends on the available hardware
- These slides should give you a good idea

FAST Index Rebuild

- Example:

```
_proutil DB –C idxbuild all

     -B 512 -TB 64 -TM 32 –TMB 512 -TF 80 -SG 64
     -thread 1 -threadnum <# CPU>
     -mergethreads 4 -datascanthreads 8
     -z –rusage
```

FAST Index Rebuild

```
-B 512 -TB 64 -TM 32 –TMB 64
```

- Contrary to popular belief, bigger –B does not help idxbuild
- TB: sort block size. Use max value 64K
- TMB: Merge block size. Bigger is not necessarily better
- TM: # buffers to merge on each merge pass. Use max value 32

FAST Index Rebuild

```
-thread 1 -threadnum <#>
```

- On by default for Enterprise DB license
- Defaults to #CPU
  - Will be doing some benchmarking next week
  - Test if better to use high –threadnum and low –mergethreads or the opposite

FAST Index Rebuild

```
-datascanthreads <#>
```

- Important: Number of threads for disk I/O bound scan phase
- Increase until it doesn't get any faster
  - Try 1 to 2 X #CPU

1. Extract index keys from record
2. Insert key into sort block (-TB 64K)
3. Sort/merge full sort block into merge block (-TMB 512K)
4. Write merge block (hopefully to –TF memory)

FAST Index Rebuild

```
-SG 64
```

- Each index is assigned to a sort group
- See next slide on mergethreads

- Always use 64

White Star software

FAST Index Rebuild

```
-mergethreads <#>
```

- Scan process has sorted and merged sort blocks in TMB chunks
- Merge blocks are further merged –TM chunks at a time
  - Repeat until all the blocks are sorted into one long list

- -threadnum is # of threads to merge each sort group
- -mergethreads is the number of threads spawned by each –threadnum thread to merge TMB blocks

- Suggest total threads = threadnum X mergethreads < 2 X #CPU

FAST Index Rebuild

```
-TF 80
```

- This is the key parameter: the more memory you make available to idxbuild, the less disk IO
- Ideally ZERO disk IO

- Pre-10.2B08, use ramdisk for sort files to simulate -TF

FAST Index Rebuild

```
-z –rusage
```

- These parameters write detailed statistics info in the output
- IMPORTANT: look for zero writes to disk during sort/merge
- The only write to disk should be the actual creation of the b-tree

White Star software

Gotcha #1: idxbuild all

- Indexes deactivated at start and re-activated at end
- What happens if idxbuild fails in last area?

- All indexes remain inactive
- START OVER (NOOOOOOOOO !!!!!!!!)

- Solution: run idxbuild by area

Gotcha #2: wasted time

- Idxbuild will scan existing index areas
- Waste of time

- Solution: Manually truncate index areas before idxbuild

- CAREFUL: make sure there are no tables accidentally created in the index areas

Gotcha #3: Mixed tables and indexes in area

- Scan phase opens data files in read-only, allowing multiple threads
- If indexes to be built exist in the area, cannot open RO
  - Cannot scan multi-threaded

Index Rebuild Example Results

- Windows VM
- 4 CPU
- 16 GB RAM
- “Normal” gp2 disks

Windows VM

- Baseline pre-10.2B04:
  - 90 sec scan + 110 sec merge/sort/build
  - Significant temp file writes slow everything down
- Add multi-threading (-threadnum 8)
  - 90 sec scan + 63 sec merge/sort/build
- Add 8 datascanthreads (default –TB 8 –TM 8 –TMB 8)
  - 35 sec scan + 64 sec merge/sort/build
- Add –TB 64 –TM 32 –TMB 512 –B 512 –SG 64
  - 40 sec scan + 52 sec merge/sort/build
- Instead try –TB 8 –TM 32 –TMB 8
  - 35 sec scan + 57 sec merge/sort/build

White Star software

Index Rebuild Example Results

- Linux VM
- 8 CPU
- 16 GB RAM
- Fast disk with 4000 dedicated IOPS

Linux VM

- Scan time with 4 or 8 datascanthreads: 20 sec
- Merge/sort/build time varied from 32-35 sec

- With small –TB and –TMB: scan dropped to 14 sec and m/s/b to ~34 sec

General Comments

- Loading and index rebuilding into empty extent can be costly
    - Constantly growing the extent files
- Keep your DB structures from your test runs
    - They are already pre-grown to the correct size
- 2nd load and idxbuild into pre-built structure will be faster

Agenda

- Introduction
- Online idxactivate
- Idxbuild
- **Fix it!**

Fix It!

- What to do when things break?
- Last resort: backups and after image
    - We all have that enabled, right?
- But plenty you can do first
- We’re going to give you the chance to do this yourself

Fix It!

- First you need to determine a problem – tooling

- Users
  - often the first to notice a problem
- Protop dashboard
  - Logfile errors monitored
  - Paid service
- Idxanalys / dbanalys
  - dbanalys contains idxanalys along with data analysis
  - Careful - some corruption will cause database to stop - self preservation

Fix It!

- Tooling cont…
  - Logfiles
    - database
    - appserver
    - client logs
  - Idxcheck
    - Online or offline
    - Specify as narrow a field as possible
    - Area/Table/Specific index
    - Validation options customisable, but defaults usually suffice

Fix It!

- Then you need to know how to fix it

- Knowledgebase
  - Search by error number
- Good idea to backup before you try anything
  - Probkup if you can
  - OS Backup otherwise
  - If your attempts make things worse you have something to return to

Fix It!

- Which tool, when?

- idxcheck
  - When you want to check if index corruption is present
  - Reports problems, no option to fix
- idxfix
  - When you want to fix a corrupt index
  - Scan for records not in indexes and fix
  - Scan for indexes with missing records and fix
  - Delete records by recid (bypassing index)

Fix It!

- idxbuild
  - To rebuild an index entirely based on the existing records
  - Powerful, but at the cost of downtime
  - And AI/Replication must be off
- idxcompact
  - To improve the space utilization of an index

Fix It! - Specific Scenarios

- Let’s talk specifics

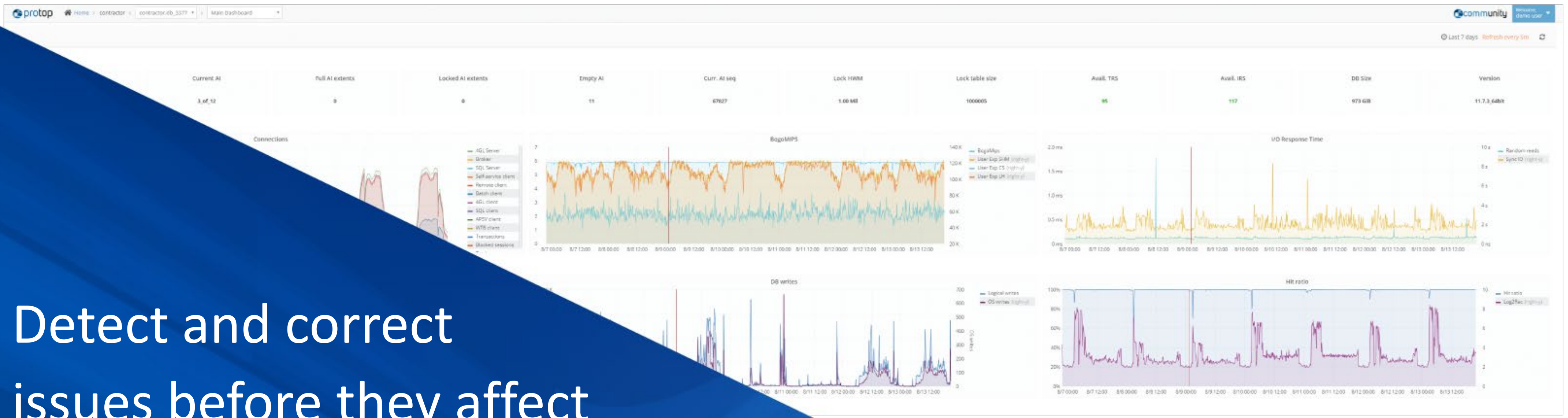
- Error 1422
  - system error: index <index name> in <table name > for recid <recid> could not be deleted. (1422)
- This usually means you’re trying to write/delete a record whose index entry is missing/corrupt
  - Sometimes a codepage issue
- If you need the data, find it using the recid and dump data then…
- idxfix option 6 to delete the record
- Reload as necessary

Fix It! - Specific Scenarios

- Index storage area growing very fast
- Most likely the index area has a low RPB and large block size
- So every block only contains very few, even one record
- Run dbanalys and check for non index data in the area
- Someone has put a table or a LOB into the index storage area
- Move it using table move for a table
- Move it using dump & load / buffer-copy for a LOB

protop

THE BEST OPENEDGE PERFORMANCE, MONITORING, AND ALERTING TOOL IN THE GALAXY! | WSS.COM/PROTOP



Current AI	Full AI extents	Locked AI extents	Empty AI	Curr. AI seq	Lock HWM	Lock table size	Avail. TRS	Avail. IRS	DB Size	Version
3_of_12	0	0	11	67827	1.00 Mil	1000005	95	117	973 GiB	11.7.3_64bit

Detect and correct issues before they affect your critical business processes

White Star software

Questions?

White Star
software