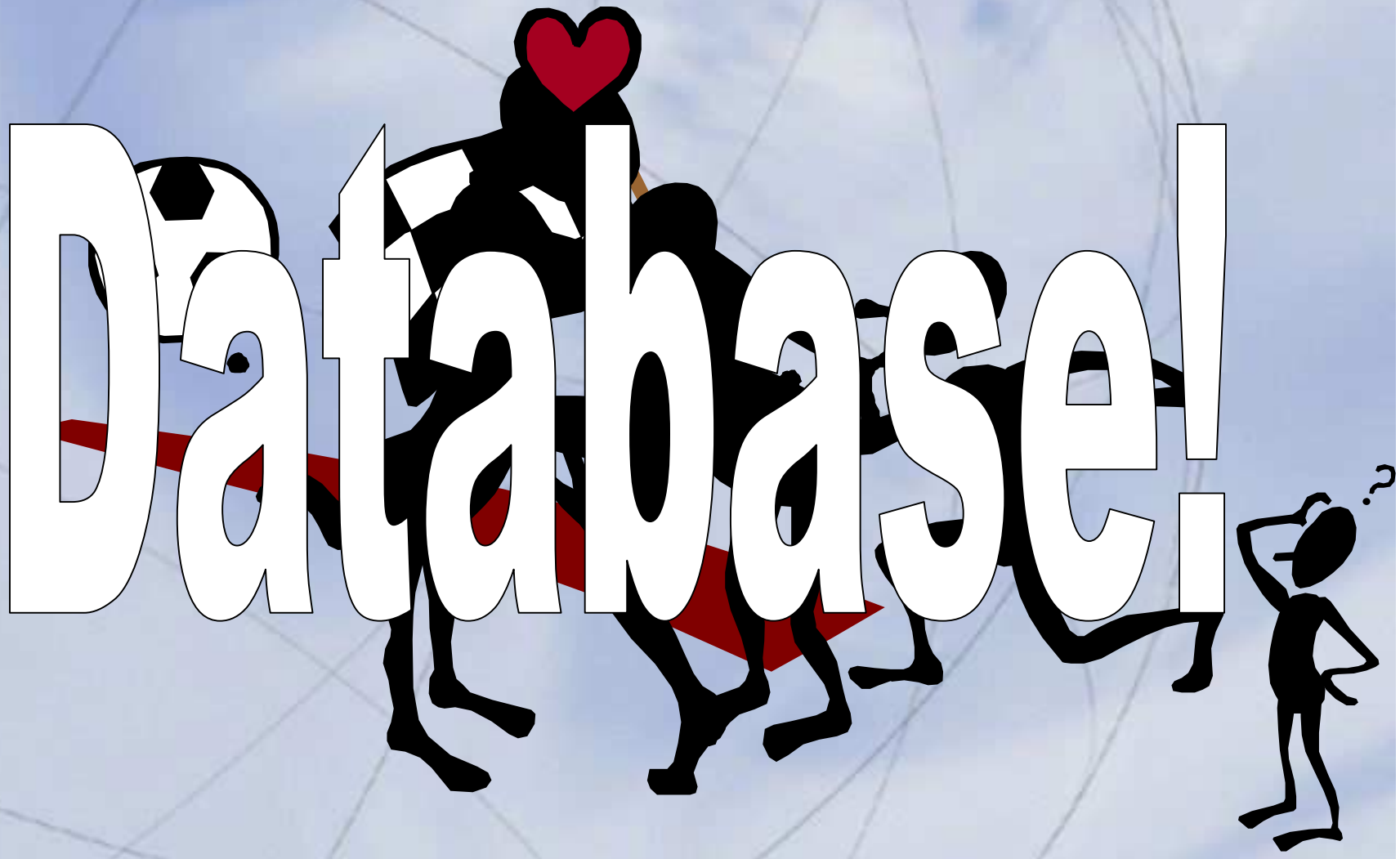
# Welcome to the


DELAWARE VALLEY
PROGRESS USERS GROUP

# Performance Thoughts



# 4GL Application Tuning

Carlos Beato
Systems Engineer
***ceduardo@progress.com***

# Agenda

- **Why and When to tune?**
- **Code & Index Analysis**
- **Profiler**
- ***Q&A***

# Why Tune?

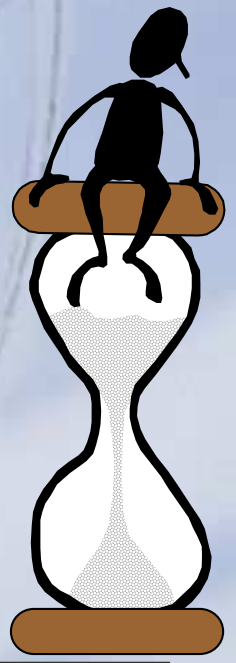
- **Modern systems are fast but not infinite**
- **Well-tuned database still takes time to search through current large table sizes**
- ***Largest single tuning win!***
  - ***Ex 1: 45 mins -> under 5 seconds***
  - ***Ex 2: Machine load reduced by 1/3rd***

# Why Would There Ever Be a Problem?

- **Progress 4GL will do what you ask it**
  - **“It just works!”**
  - **Doesn’t tell you that there’s a better way!**
- **Test environments typically tiny**
  - **Full table scan delay too small to notice**
- **Uninformed people writing code!**
  - **Managers & reporting tools**
  - **Novice programmers**

# When to tune?

- **Every code release!**
  - **Should be part of standard QA**
- **If there are reports of delays at a screen**



# Agenda

- Why/When to tune?
- Code & Index Analysis
- Profiler
- Q&A

# Code Analysis Methods

- “The Truth is in the Code”
- -zqil
- Cross-reference listing
- promon / VST’s / Fathom
- Eyes and Ears!
- proc.mon (-yx)

# Code Analysis Methods: The Truth is in the Code

- A thousand theories are not as valuable as one case of jumping in and finding out
- Always required after other analyses are complete anyway
  - code won't be fixed by running tools on it!
- Not practical over large amounts of code
  - Notoriously inaccurate
  - Time-consuming
- … so efforts must be focused.
  - Rest of these tools help you focus efforts

# Code Analysis Methods: The Undocumented –zqil Parameter

- **Shows *actual* index usage and index bracket depth for each query**
  - **Confirms or corrects programmer's *opinion/belief* of what index is being used, and to what degree**
  - ***Only –zqil & XREF* are independent of test environment size**
- **Requires self-service connection; outputs to database.lg file**
- **Related parameter –zqilv "verbose"**

# Code Analysis Methods: -zqil Output

- For each query shows:
  - Compiler index selection & bracket depth
  - Run-time selection & depth
  - Wall-clock time for complete execution
- Bracket depth shown for top and bottom of bracket

# Code Analysis Methods: -zqil Output

```
for each customer no-lock where
  customer.Name begins "A":
  display custnum.
end.
```

```
==Compiled Query Resolution Method: Query No. 7==
(6135)
01:14:58 INDEX 15 1 1 (6157)
01:14:58
==Server Query execution Method Query No. 7== (6136)
01:14:58 INDEX 15 (6141)
01:15:00
```

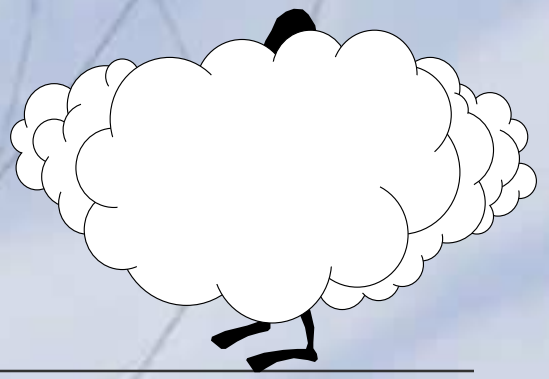
# Code Analysis Methods: How to Use -zqil Output

- Metaschema has index and table data
- Questions
  - Is bracket depth as expected?
  - Is index selection as expected?

```
Find first _Index where _Idx-num = 15 no-lock.
Find _File of _Index no-lock.
Display _File-Name _Index-Name.
For each _Index-Field of _Index no-lock,
  First _Field of _Index-Field:
  Display _Index-Seq _Field-Name.
End.
```

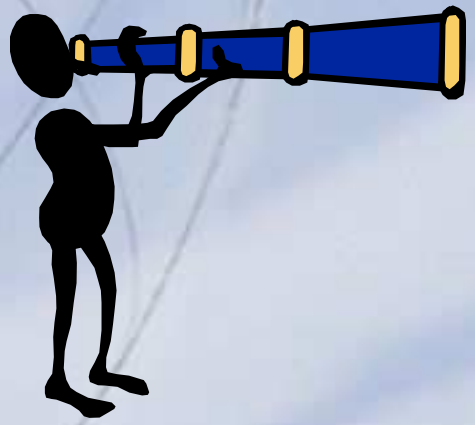
# Code Analysis Methods: -zqil Caveats

- Works only in self-service mode
- Undocumented and unsupported
- Only approximates query execution time
  - Includes display logic and other loop code, and varies based on system load
- Does not say “how much”, only “how long”
- Inconvenient to interpret



# Code Analysis Methods: Cross Reference Option

- COMPILE has XREF option
  - Unfortunately not very famous
- Very useful
- Voluminous output
- Good tool for picking “low-hanging fruit”



# Code Analysis Methods: Cross Reference Sample Output

```
.\test4.p .\test4.p 2 STRING "Order" 5 NONE UNTRANSLATABLE
.\test4.p .\test4.p 2 SEARCH sports2000.Order OrderNum WHOLE-INDEX
.\test4.p .\test4.p 3 ACCESS sports2000.Order Ordernum
.\test4.p .\test4.p 3 STRING "zzzzzzzzz9" 10 NONE TRANSLATABLE  FORMAT
.\test4.p .\test4.p 7 STRING "Customer" 8 NONE UNTRANSLATABLE
.\test4.p .\test4.p 7 SEARCH sports2000.Customer CustNum WHOLE-INDEX
.\test4.p .\test4.p 8 ACCESS sports2000.Customer CreditLimit
.\test4.p .\test4.p 8 ACCESS sports2000.Customer CreditLimit
.\test4.p .\test4.p 8 UPDATE sports2000.Customer CreditLimit
.\test4.p .\test4.p 12 ACCESS sports2000.Customer Name
.\test4.p .\test4.p 12 STRING "A" 1 NONE TRANSLATABLE
.\test4.p .\test4.p 12 SEARCH sports2000.Customer Name
.\test4.p .\test4.p 15 ACCESS sports2000.Customer CustNum
.\test4.p .\test4.p 15 STRING ">>>>9" 5 NONE TRANSLATABLE  FORMAT
.\test4.p .\test4.p 20 ACCESS sports2000.Customer CreditLimit
.\test4.p .\test4.p 20 SEARCH sports2000.Customer CustNum WHOLE-INDEX
.\test4.p .\test4.p 23 ACCESS sports2000.Customer CustNum
.\test4.p .\test4.p 23 ACCESS sports2000.Customer CreditLimit
.\test4.p .\test4.p 23 STRING ">>>>9" 5 NONE TRANSLATABLE  FORMAT
.\test4.p .\test4.p 23 STRING "->,>>>,>>9" 10 NONE TRANSLATABLE  FORMAT
.\test4.p .\test4.p 24 STRING "Order Num" 9 LEFT TRANSLATABLE
.\test4.p .\test4.p 24 STRING "Ordernum" 8 NONE UNTRANSLATABLE
.\test4.p .\test4.p 24 STRING "---------------" 16 NONE UNTRANSLATABLE
```

# Code Analysis Methods: How to Use Cross Reference

- COMPILE *file.p* XREF *xrefOutputFile*
- SEARCH entries show compiler index choices

```
.\test4.p .\test4.p 12 SEARCH
   sports2000.Customer Name
```

```
for each customer no-lock where
  customer.Name begins "A":
  display custnum.
end.
```

# Code Analysis Methods: How to Use Cross Reference

- **WHOLE-INDEX phrase shows table scans**
  - **Use *grep* to quickly reveal**

```
2 SEARCH sports2000.Order
     OrderNum WHOLE-INDEX
7 SEARCH sports2000.Customer
     CustNum WHOLE-INDEX
20 SEARCH sports2000.Customer
     CustNum WHOLE-INDEX
```

# Code Analysis Methods: Cross-Reference Caveats

- **Can be good indicator of "low-hanging fruit"**

```
for each customer no-lock where
    Customer.CreditLimit > 10000:
  20 SEARCH sports2000.Customer
    CustNum WHOLE-INDEX
```

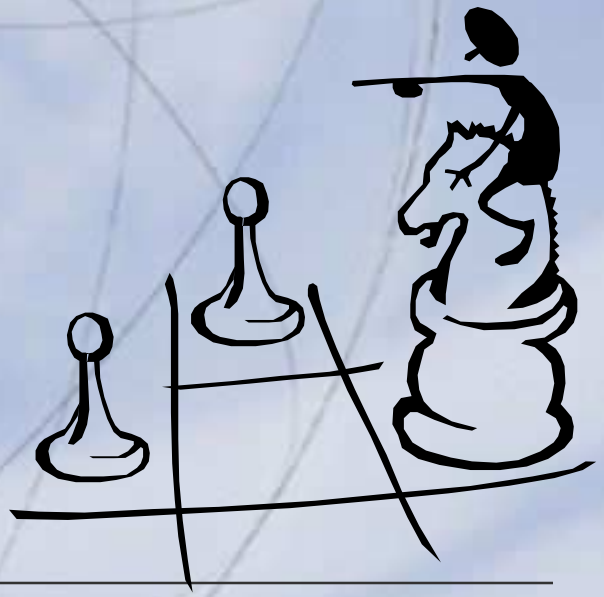
- **Not all WHOLE-INDEX scans are bad**

```
for each order no-lock:
```

- ***Not Acceptable as sole analysis tool!***

# Code Analysis Methods: Test Environment

- **Production Distribution:**
  - **Easiest: Copy Production to Test**
  - **Objection: Existing tests use defined data?**
  - **Solution: “Populate” procedure to add test data to each new copy of production**
- **Production Size!**
  - **As Large As Production**
  - **Larger in places: expand small tables**



# Code Analysis Methods: Test Environment

- **Theme: Reverse typical test/release performance characteristics:**
  - **Is: Fast in test, slower in production**
  - **Should be: Fast in test, Faster in production!**
  - **Testing = “Exposing problems”**
  - **Tune DB to *slow down* operations (small –B, small clusters)**

# Code Analysis Methods: Test Environment: Deployment

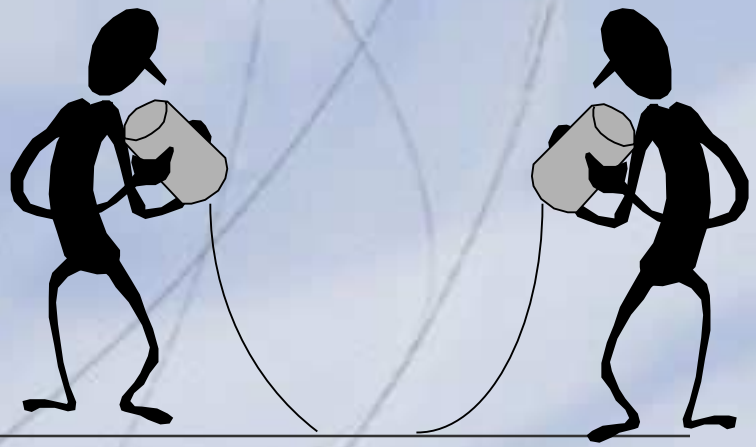
- **Always client/server, one server per client, prefer remote connection**
  - **Permits meaningful client/server statistic correlations**
  - **Better exposes real-world characteristics if real users are client/server**

# Code Analysis Methods: Promon

- **Record read:write ratio**
  - **(RecReads) vs (RecCreates) + (RecWrites) + (RecDeletes)**
  - **Goal: < 50:1**
  - **Over 100, 150:1 is a probable problem**
  - **Personal record: 8000:1**
    - **750,000 records read to print a label !!**
  - ***Running reports will skew number***
  - **Sample for large times, but watch for spikes in rate, esp. in small installations**

# Code Analysis Methods: Promon

- **Heaviest User information**
  - **Promon R&D / 3 / 2. I/O By Process**
  - **Same info in VST _UserIO**
  - **Fathom**
  - **Call them up and ask what they're running!**



# Code Analysis Methods: VST's

- _tablestat
  - -basetable/-tablerangesize in v9
  - -tablebase/-tablelimit in v8.3
- _indexstat
  - -baseindex/-indexrangesize in v9
  - -indexbase/-indexlimit in v8.3
- VST's must be enabled (8.3)
- Slight performance hit; not for production

# Code Analysis Methods: VST Monitor

```
define temp-table old-reads  (tableid, num-reads).
define temp-table curr-reads (tableid, num-reads).
define temp-table read-diff  (tableid, num-new-reads)
   index idx1 as primary num-new-reads descending.

Initialize old-reads with current _tablestat data.

Repeat:
    empty temp-tables curr-reads, read-diff.
    populate curr-reads from _tablestat.
    calculate read-diff.
    move curr-reads to old-reads.

    for each read-diff where num-new-reads > 0:
        display read-diff.tableid
                read-diff.num-new-reads.
    end.
    pause 10.  /* can also hit space bar */
End.
```

# Code Analysis Methods: VST's

- **Watch for**
  - **Very fast read rates**
  - **More records than appropriate**
- **Check suspicious code in isolation**
- **Use this info to separate data into storage areas**

# Code Analysis Methods: Eyes and Ears

- **What takes a long time to run?**
  - **Users will always be able to pinpoint problem areas**
    - ***"Users are Your Friends!"***
    - **Allows focus on tasks which cost users the most time/productivity**
    - **May be enduring something that's fixable**
  - **Pinpoint down to the keystroke level**
  - **Run record count monitor at that point**

# Code Analysis Tools: Eyes and Ears

- **Watch for heavy client disk activity**
  - **May be result of large sort**
  - **Might be able to tune or “index away”**
- **Know the rules!**
  - ***Progress Database Design Guide,*** **chapters 4-5.**

# Code Analysis Tools: Procedure Monitor – proc.mon

**Where's *your* busiest code??**

- **Client parameter –yx -> proc.mon**

# Code Use Analysis: sample proc.mon

| Caller | Callee | Load Size | Calls | Rd Bytes | Reread | Time |
|---|---|---|---|---|---|---|
| <top> | start1 | 929 | 1 | 929 | 0 | 1 |
| start1 | start2 | 17847 | 1 | 17847 | 0 | 9 |
| start2 | nightly | 75071 | 1 | 75071 | 0 | 13001 |
| | | | ... | | | |
| backlog | cd-deltrig | 1635 | 133 | 1635 | 0 | 161 |
| backlog | fprename | 1363 | 4 | 0 | 0 | 142 |
| backlog | f-str2itm | 13896 | 635 | 13896 | 0 | 57001 |
| backlog | notes | 3343 | 7 | 3343 | 0 | 35 |
| nightly | rbase-load | 38908 | 1 | 38908 | 0 | 2763721 |
| rbase | fprename | 1363 | 3 | 0 | 0 | 687 |
| rbase | notes | 3343 | 55820 | 0 | 0 | 155590 |
| nightly | xref-dmp | 2447 | 1 | 2447 | 0 | 398 |
| nightly | ntlyrpts | 49078 | 1 | 49078 | 0 | 372 |

# Code Analysis Tools: Code Use Analysis: proc.mon

**Watch for:**

- **large numbers of calls;** ***put these in internal procedures or (better) include files***
- **rereads;** ***could use more –mmax client memory***
- **long execution times;** ***concentrate rewrite efforts here***

# Code Analysis Tools: Code Use Analysis: proc.mon

**Collect associated code into procedure libraries**

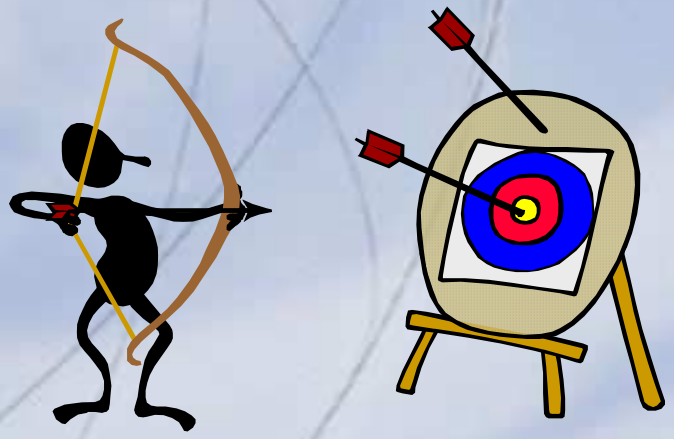
- **Faster to run, no fopen() step**
  - **EP:IP is 2:1 Unix, 5:1 Windows**
- **Use –pls if net-shared / found on fileserver**
- **V9 host-based? Shmem libraries!**
  - **Runs faster**
  - **Save memory for –B, –Bt, –mmax**

# Agenda

- Why/When to tune?
- Code & Index Analysis
- **Profiler**
- Q&A

# Code Analysis Tools: The Application Profiler Tool

- Shows how much time spent in each line
- Detailed look at execution cost of every line of code
- Installed by default (Windows)
  - %DLC%\src\samples\profiler



# Agenda

- Why/When to tune?
- Code & Index Analysis
- **Profiler**
- Q&A

# Questions?

**How do I....???**
**Can I...???**
**Is it better to..???**