Code performance workshop

The TL;DR version
• Intro
  • This is the short version of the Code Performance Workshop

• Compile-listing ain't enough

• Network effects

• Unnecessary work

• Conclusion
your hosts

- One named Peter
- One named Paul
one named Peter (Judge)

pjudge@progress.com
Software Architect*
@ Progress since 2003
Integration-y stuff – Authentication Gateway, HTTP-Out, Corticon et al
OE Best Practices / OERA / AutoEdge / CCS
4GL since 1996

* Aka programmer who knows PowerPoint
one named Paul (Koufalis)

pk@wss.com
Progress DBA and UNIX admin since 1994
Expert consulting related to technical aspects of Progress and OpenEdge
Wide range of experience
  Small 10 person offices to 2500+ concurrent users
  AIX, HPUX, Linux, Windows…if Progress runs on it, I’ve worked on it
  Single malts and American bourbons
The oldest and most respected independent DBA consulting firm in the world

Four of the world’s top OpenEdge DBAs

Author of ProTop, the #1 FREE OpenEdge Database Monitoring Tool
  • http://dashboard.dbappraise.com
Everybody knows that there's a problem: "It's slow". Now what?

1. Find the source of the problem
   1. Reproduce the issue or at least follow it live - **HOW?**
2. Identify the errant code – **HOW?**
3. Fix it. That's the easy part
"The system is slow" or better yet "it's slow". Sound familiar?

Three ways to identify problems

1. The user actually tells you what he was doing
   • Go buy a lottery ticket
2. Trend data
   • Conceptually easy (more later)
3. Sit in front of a screen 24/7 and watch the system
Finding the Problem - Trending data

- In a nutshell: there's a LOT of data

- Tools like ProTop and OE Management trend global data
  - For example: Index and table usage statistics

- Trending data at the user level is daunting
  - Ex.: 500 users, 500 tables, 1500 indexes => 1M data points per sample
    - Most of them zero
    - How often do you sample? Every minute? 5 minutes?
    - "Don't store the zeros" - sure – but there is a CPU cost to read the million samples and discard them
  - Same idea for CSC (Client Statement Cache)
Finding the Problem - Trending data

- Identify tables being hit
- In this case, 1M+ reads/sec in one table
- 750K+ reads/sec sustained for 2h – one table

Graphs from ProTop web dashboard
- Table names not shown but they're there
Finding the problem – from trending data to source code

- You have to play detective
- Repeating pattern tells me it's a nightly batch

- If necessary, login at 10:00pm and start monitoring in real time
- Processes consuming excessive CPU or disk I/O
  - Use nmon (Linux, AIX) or glance (HPUX) or Windows PerfMon
  - Processes doing excessive database reads/writes
    - Use VST data (more later)
Focus on 3 tools to help you identify problems:

- VSTs
- LOG-MANAGER
- –zqil

Profiler is a great tool but we don't have time
Assumption for Part 1: Problem is query-related
- Peter will talk later about unnecessary work in your code

_UserTableStat/_UserIndexStat VSTs
- Which process is hitting what tables/indexes
- Calculate volume but don’t ignore rate and relative size
  - 1K reads in 1M records may be normal
  - 10M reads in 10K records probably is not
  - 750K reads/sec is probably a tad high

CSC: What code is running
- Turn on Client Statement Cache to follow in real time
- Info in _Connect VST
Using VST Data

- Collect data before and after the problematic code and subtract
- Do a little math with etime() to calculate rates

```plaintext
find _connect where _connect-pid = 12345.

for each _UserTableStat where _UserTableStat-conn = _connect-usr:
    find _file no-lock where _file-number = _UserTableStat-id.
    displ _file-name _UserTableStat-read.
end.

for each _UserIndexStat where _UserIndexStat-conn = _connect-usr:
    find _index no-lock where _idx-num = _UserIndexStat-id.
    displ _index-name _UserIndexStat-read.
end.
```
## Using VST Data

ProTop Free does it all for you.

---

### Table Activity

<table>
<thead>
<tr>
<th>#</th>
<th>Area</th>
<th>Table Name</th>
<th>RM Chain</th>
<th>#Records</th>
<th>Turns</th>
<th>Create</th>
<th>Read v</th>
<th>Update</th>
<th>Delete</th>
<th>OS Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>780</td>
<td>20</td>
<td>s_crm-valid-queue</td>
<td>60</td>
<td>11939866</td>
<td>0.16</td>
<td>0</td>
<td>194453</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>782</td>
<td>20</td>
<td>s_crm-crm-field</td>
<td>60</td>
<td>48</td>
<td>0.52</td>
<td>0</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>787</td>
<td>20</td>
<td>s_crm-lat-field</td>
<td>60</td>
<td>48</td>
<td>0.36</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>849</td>
<td>22</td>
<td>wb_dept-user</td>
<td>60</td>
<td>7088</td>
<td>0.00</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>790</td>
<td>20</td>
<td>s_mazeam</td>
<td>59</td>
<td>1979</td>
<td>0.00</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>169</td>
<td>20</td>
<td>cost-factor</td>
<td>60</td>
<td>18</td>
<td>0.38</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>528</td>
<td>18</td>
<td>prod-exp</td>
<td>37</td>
<td>217216</td>
<td>0.00</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>557</td>
<td>18</td>
<td>product</td>
<td>1335</td>
<td>214159</td>
<td>0.00</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>654</td>
<td>142</td>
<td>so-pick-d</td>
<td>383</td>
<td>28455997</td>
<td>0.00</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td>alternate</td>
<td>88</td>
<td>488</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>am-list</td>
<td>68</td>
<td>2</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>am-list-d</td>
<td>4</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Index Activity

<table>
<thead>
<tr>
<th>#</th>
<th>Area</th>
<th>Index Name</th>
<th>Lvls</th>
<th>Blocks</th>
<th>Util</th>
<th>Idx</th>
<th>Root</th>
<th>Create</th>
<th>Read v</th>
<th>Split</th>
<th>Delete</th>
<th>BlkDl</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1748</td>
<td>21</td>
<td>s_crm-valid-queue.s_crm-changes</td>
<td>3</td>
<td>1,872</td>
<td>90%</td>
<td>38839</td>
<td>0</td>
<td>196,473</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1964</td>
<td>19</td>
<td>wm-send.WMS-ID</td>
<td>3</td>
<td>6,353</td>
<td>94%</td>
<td>13695</td>
<td>0</td>
<td>963</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>Field_Name</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>68552</td>
<td>0</td>
<td>621</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1953</td>
<td>21</td>
<td>wm-pick.WMS-ID</td>
<td>2</td>
<td>508</td>
<td>96%</td>
<td>41599</td>
<td>0</td>
<td>149</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1456</td>
<td>19</td>
<td>so-pick.ctrl-machine</td>
<td>3</td>
<td>969</td>
<td>90%</td>
<td>9535</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1463</td>
<td>19</td>
<td>so-pick.shipped</td>
<td>2</td>
<td>252</td>
<td>98%</td>
<td>9983</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>Field_Position</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>68516</td>
<td>0</td>
<td>51</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>Index_Field.Index/Number</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>68544</td>
<td>0</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1734</td>
<td>21</td>
<td>s_crm-crm-field.s_crm-crm-field</td>
<td>1</td>
<td>1</td>
<td>6%</td>
<td>35263</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1816</td>
<td>23</td>
<td>wb_dept-user.wb_user</td>
<td>2</td>
<td>36</td>
<td>50%</td>
<td>13359</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>Index_File/Field</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>68180</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>Field/File</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>68288</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1196</td>
<td>121</td>
<td>prod-exp-loc-d.prod-exp-loc-d</td>
<td>4</td>
<td>147,490</td>
<td>70%</td>
<td>127</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### User ID Activity

<table>
<thead>
<tr>
<th>User</th>
<th>Tenant Name</th>
<th>PID</th>
<th>Flags</th>
<th>Blk Ac</th>
<th>OS Rd</th>
<th>OS Wr</th>
<th>Hit4</th>
<th>Rec Lck</th>
<th>Rec Wts</th>
<th>Line</th>
<th>Program Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>773</td>
<td>0</td>
<td>48130</td>
<td>SX8</td>
<td>390386</td>
<td>0</td>
<td>0</td>
<td>100,004</td>
<td>194192</td>
<td>0</td>
<td>582</td>
<td>crmim/apply.p</td>
</tr>
<tr>
<td>779</td>
<td>0</td>
<td>13742</td>
<td>SX</td>
<td>1411</td>
<td>0</td>
<td>0</td>
<td>99,994</td>
<td>1</td>
<td>0</td>
<td>305</td>
<td>crmim/apply.p</td>
</tr>
<tr>
<td>772</td>
<td>0</td>
<td>21791</td>
<td>SXB</td>
<td>1291</td>
<td>0</td>
<td>0</td>
<td>100,004</td>
<td>169</td>
<td>0</td>
<td>-1</td>
<td>wm/senddata.p</td>
</tr>
<tr>
<td>783</td>
<td>0</td>
<td>6278</td>
<td>SXB</td>
<td>200</td>
<td>0</td>
<td>0</td>
<td>100,004</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>wm/senddata.p</td>
</tr>
</tbody>
</table>
LOG-MANAGER

- A terribly underused but awesomely amazing tool
- Allows you to leave debug messages in your code
  - No more /* Message here vValue. */
- Create some secret hotkey sequence to activate
  - I.e. you can turn it on in production for one user
- Writes detailed info to a log file
- As easy as ...

  assign log-manager:logfile-name = "c:\temp\wshop.log"
  log-manager:logging-level = 3
  log-manager:log-entry-types = "4GLTrace,4GLTrans,QryInfo".

  /* Writing your own messages is easy too ... */
  log-manager:write-message(
    string(LogLevelEnum:WARN) + ': ' + pcMessage, pcMessageGroup).
for each order where order-num > x:

Type: FOR Statement
Client Sort: N
Scrolling: N
Table: wshop.Order
  Indexes: Order-Num
Query Statistics:  Bad1 logmgr.p line 23
QueryId: 101299360
DB Blocks accessed:
  wshop : 15599
DB Reads:
  Table: wshop.Order : 4557
  Index: Order.Order-Num : UNAVAILABLE
wshop.Order Table:
  4GL Records : 3399
  Records from server: 3399
    Useful: 3399
    Failed: 0
  Select By Client: N

16K DB reads
4557 fragments
3399 records
for each order fields(order-num) where order-num > x:

Type: FOR Statement
Client Sort: N
Scrolling: N
Table: wshop.Order
  Indexes: Order-Num
Query Statistics: Bad1 logmgr.p line 23
QueryId: 101299360
DB Blocks accessed:
  wshop : 11333
DB Reads:
  Table: wshop.Order : 3400
  Index: Order.Order-Num : UNAVAILABLE
wshop.Order Table:
  4GL Records: 3399
  Records from server: 3399
  Useful: 3399
  Failed: 0
  Select By Client: N
for each order where terms = "net30" by terms:

Type: FOR Statement  
Client Sort: Y  
Scrolling: N  
Table: wshop.Order  
Indexes: Order-Num  
Query Statistics:  Bad3 logmgr.p line 45  
Entries in result list: 15526  
Time to build result list (ms): 169

DB Blocks accessed to build result list:
- wshop: 252527

DB Reads to build result list:
- Table: wshop.Order: 61300
- Index: Order.Order-Num: UNAVAILABLE

- Read all records (61K) PLUS another 20K fragments to return 15K records
- 320K total DB reads

wshop.Order Table:
Records from server: 15526  
Useful: 15526  
Failed: 0  
Select By Client: N  
Fields: Terms  
Query Statistics:  Bad3 logmgr.p

Entries in result list: 15526
QueryId: 35632288
DB Blocks accessed:
- wshop: 73703

DB Reads:
- Table: wshop.Order: 20571
- Index: Order.Order-Num: UNAVAILABLE

wshop.Order Table:
4GL Records: 15526
-zqil

- Unsupported and undocumented startup parameter
  - Aren't those the best!?!?

- Writes detailed run-time index usage information to db.lg - yes db.lg
  - Do NOT use in prod please

- Tells you which index is used and how many fields deep

- Format is INDEX # LOWER-BOUND UPPER-BOUND TYPE
for each order:                         INDEX 20 0 0 (pu order-num)
  ▪  ... where ship-date = today      INDEX 37 1 1 (Ship-date + carrier)
  ▪  ...   and carrier    = “pk”       INDEX 36 2 2 EQUALITY (Carrier + ship-date)
  ▪  ...   and sales-rep  = “pk”       INDEX 35 3 3 EQUALITY (sales-rep+carrier+ship-date)
  ▪  **Now let’s make it interesting**
  ▪  ... where ship-date=... and carrier=... and sales-rep GT “”
      •  INDEX 36 2 2 EQUALITY (Carrier + ship-date)
  ▪  ... where sales-rep=... and carrier=... and ship-date LT ...
      •  INDEX 35 2 3 (Sales-rep + Carrier + ship-date)
  ▪  ... where sales-rep =... and carrier NE... and ship-date LT ...
      •  INDEX 23 1 1 EQUALITY (sales-rep)
  ▪  ... where sales-rep =... and carrier GT “” and ship-date LT ...
      •  INDEX 35 2 1 (sales-rep + Carrier)
Network Effects

- Shared memory DB connections make programmers look like (geeky) rock stars
- Network connections are more like the morning after – not so pretty
Network Effects - Basics

- Field lists: fields of record that are sent to the client
- Prefetch: multiple records per message (no-lock)
- Message: Unit of measure for 4GL network data transfers
  - Size controlled by –Mm database start-up parameter
- MTU (maximum transmission unit): the largest packet size that can be transmitted over a network
  - This is a network parameter, not a Progress parameter

- Server Parameters
  - -Mi / -Ma / -Mpb: min, max users per server and number of servers per broker
  - 4GL servers are round-robin single-threaded
  - SQL servers are multi-threaded
Thank `<insert favourite deity>` Progress doesn’t send the LOB across the wire unless you ask for it.

- The LOB field is really a separate entity to the record
- The `real` record only contains a pointer to the LOB
- The LOB may be in another storage area (and should be)

- When you access the LOB, the client requests it from the server
Network Effects – Legacy Parameters

- **Mm**: message buffer size
  - Default 1024 is too small
  - Max 32600 doesn’t seem to be warranted
  - 8192 is a nice sweet spot

- **Mi / Ma**: min/max users per server
  - The more users per server, the less time the server can dedicate to any one user
  - -Mi 1 -Ma 5 is a good start
  - -Mi 1 -Ma 1 if you have heavy duty users
    - AppServer agents
Network Effects – New Prefetch Parameters

- No-lock queries
- Forward only or scrolling
- 10.2B06 + and 11.1+

- `prefetchPriority`: server defers poll for other requests while filling message
  - Current suggested value 100 records added before next poll
- `prefetchDelay`: Fills first message. By default first message contains one record
  - In theory this is better. In practice the ms difference is not significant
- `prefetchNumRecs`: How many records are stuffed in a message
  - 100 records is a good start (default is 16)
- `prefetchFactor`: How full (%-wise) to fill a message
  - 90-100%
Maximum Transmission Unit (MTU)

- A network parameter set at the NIC level
- Enable on the routing infrastructure
- Default is 1500
- “Jumbo Frames” is typically 9000 bytes

- The advantage lies in the relative size of network header data
  - 1500 byte MTU: 1460 byte payload / 1538 byte total = 95% efficient
  - 9000 byte MTU: 8960 byte payload / 9038 byte total = 99% efficient
Monitoring Server Messages

- **_ActServer VST**
  - Key: _server-ID
  - Interesting fields: _Server-ByteSent, _ServerMsgSent, _ServerRecSent
  - Calculate send size (Bytes sent / messages sent) and compare to –Mm

- **Bug:** When a record is larger than –Mm, only the first msg is counted
  - I.e. if you send a 4K record and –Mm is 1024, only 1 msg and 1024 bytes sent recorded
  - Blobs sent in 32,000 byte chunks – each chunk increments msgSent by 1 and byteSent by -Mm

- **ProTop Free:**

<table>
<thead>
<tr>
<th>Srv</th>
<th>Type</th>
<th>Port</th>
<th>Cnx</th>
<th>Max</th>
<th>QryRcvd</th>
<th>RecRcvd</th>
<th>MsgRcvd</th>
<th>rs/msg</th>
<th>RecSent</th>
<th>MsgSent</th>
<th>rs/msg</th>
<th>MB Sent</th>
<th>MB Rcvd</th>
<th>RcvdSz</th>
<th>SendSz</th>
</tr>
</thead>
<tbody>
<tr>
<td>9999</td>
<td>Total</td>
<td></td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.04</td>
<td>0.00</td>
<td>0</td>
<td>116</td>
</tr>
<tr>
<td>1</td>
<td>Auto</td>
<td>3000</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.04</td>
<td>0.00</td>
<td>0</td>
<td>116</td>
</tr>
<tr>
<td>4</td>
<td>Inact</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Inact</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Auto</td>
<td>3001</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
unnecessary work

1. Too many calls

2. Too much stuff per call

3. Too many copies of data
- Network typically major bottleneck
  - Makes other performance problems worse

- Number of roundtrips
  - Making a server connection has cost

- Data volume per roundtrip

- Network topography has impact …
  … but usually out of our control
optimise network roundtrips

```plaintext
loginWindow.w.run getLoginLanguages()
run getLoginCompanies()
run buildUI().
```
optimise network roundtrips

```java
run getLoginLanguages()
run getLoginCompanies()
run buildUI().
```
optimise network roundtrips

```
loginWindow.w
  run getLoginLanguages()

② run getLoginCompanies() ——> getLoginCompanies()

  run buildUI().
```
optimise network roundtrips

```java
// loginWindow.w
1 run getUIData()

run buildUI().

getUiData()
    run getLoginLanguages()
    run getLoginCompanies()
```
define temp-table ttData ...

run populateData (output table ttData).
run showData (input table ttData).
run getChanges (output table ttData).
run saveChanges (input table ttData).
define temp-table ttData ...

run populateData (output table ttData).
run showData (input table ttData).
run getChanges (output table ttData).
run saveChanges (input table ttData).

no deep copies
Yes, you define HANDLE variables to work with them

But you can pass / call them as datasets

- To a parameter defined as DATASET-HANDLE
- To a parameter defined as DATASET
- To a parameter defined as HANDLE

Nice thing with DATASET-HANDLE is that if you receive them you can use static code against them as if you got a 'real' dataset
procedure fetch_data:
  def output param poLotsaData as JsonObject.

  def var hDataset as handle.
  def var oDataObject as Object.

  oDataObject = new CustomerData().
  oDataObject:GetData(
    output dataset-handle hDataset).

  poLotsaData:Read(hDataset).
  /* now has a property called dsCustomer */

  oDataObject = new EmployeeData().
  oDataObject:GetData(
    output dataset-handle hDataset).

  poLotsaData:Read(hDataset).
  /* now has a property called dsCustomer
   AND one called dsEmployee */

end procedure.

class CustomerData:
  define dataset dsCustomer for ttCustomer, ...

  method public void GetData(
    output dataset dsCustomer).

end class.

class EmployeeData:
  define dataset dsEmployee for ttEmployee, ttDepartment, ...

  method public void GetData(
    output dataset dsEmployee).

end class.

dataset-handles are not really handles
Make sure you have a problem

- LOG-MANAGER:LOG-ENTRY-TYPES = 'Temp-tables'
- Data structure reference counts – datasets, buffers, objects, procedures, etc

```haskell
define variable hDS as handle no-undo.
define variable iLoop as integer no-undo.

hDS = session: first-dataset.
do while valid-handle(hDS):
    assign iCnt = iCnt + 1.
    hDS = hDS: next-sibling.
end.
```

```
TEMP-TABLE Created TEMP-TABLE Headers (ID: 4 NO-UNDO Indexes: 1) OpenEdge.Net
TEMP-TABLE Created TEMP-TABLE ConfigOption (ID: 5 NO-UNDO Indexes: 1) OpenEd
TEMP-TABLE Created TEMP-TABLE Registry (ID: 6 NO-UNDO Indexes: 1) OpenEdge.Net
TEMP-TABLE Created TEMP-TABLE HeaderParameter (ID: 7 NO-UNDO Indexes: 1) Open
```
- Make value passing of tables/dataset the exception
  - Look for ANY and ALL temp-table, table-handle, dataset, dataset-handle calls without BY-REFERENCE

- When you have to make a deep copy, clean up after yourself

- Desperate measures may be needed

```plaintext
run get_data.p (  
  output dataset-handle hDataset  
  by-reference).
```

```plaintext
run get_data.p (  
  output dataset-handle hDataset).  
finally:  
  delete object hDataset.  
end finally.
```

```plaintext
hDS = session:first-dataset.  
do while valid-handle(hDS):  
  delete object hDS.  
  hDS = session:first-dataset.  
end.
```
- Make sure you evaluate functions only once
  - `do i = 1 to udf()` vs. `do i = udf() to 1 by -1`
  - Functions are sadly not always idempotent
- Use arrays instead of delimited strings
- Stuff inside a loop
  - Record FINDs
  - RUN (INPUT-OUTPUT tt)
There are no few wizards. There is no barely any magic.

Measurement is the only sure way to KNOW
  - That you have a problem
  - That you fixed the problem
Unsupported tool introduced in V9

Now supported in PDSOE 11.6 (maybe 11.5 ?)

Example UI available in $DLC/src/samples
  • I just wrote some simple ChUI/text code to do it myself

Use the PROFILER handle
  • Ex.: PROFILER:ENABLED = TRUE.

Outputs time spent in each line of code, number of executions…

Sorry – no time to go into Profiler in detail
### Code Block #1

#### Calls To

<table>
<thead>
<tr>
<th>Code Block</th>
<th>Calls</th>
<th>Avg Time</th>
<th>Tot Time</th>
<th>% Session</th>
<th>Cum Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>report.p</td>
<td>1</td>
<td>11.715508</td>
<td>11.715508</td>
<td>93.911398</td>
<td>12.421169</td>
</tr>
<tr>
<td>cummulventes report.p</td>
<td>9,054</td>
<td>0.000065</td>
<td>0.584186</td>
<td>4.682829</td>
<td>0.584186</td>
</tr>
<tr>
<td>ListTop5 report.p</td>
<td>9</td>
<td>0.013497</td>
<td>0.121475</td>
<td>0.973742</td>
<td>0.121475</td>
</tr>
<tr>
<td>ProfileCode PROFILE/profile.w</td>
<td>2</td>
<td>0.026194</td>
<td>0.052388</td>
<td>0.419942</td>
<td>0.00000</td>
</tr>
</tbody>
</table>

#### Calling Code Block

<table>
<thead>
<tr>
<th>Calling Code Block</th>
<th>Calls From % Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProfileCode PROFILE/profile.w</td>
<td>2</td>
</tr>
</tbody>
</table>

#### Called Code Block

<table>
<thead>
<tr>
<th>Called Code Block</th>
<th>Calls To % Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>cummulventes report.p</td>
<td>9,054</td>
</tr>
<tr>
<td>ListTop5 report.p</td>
<td>9</td>
</tr>
</tbody>
</table>

#### Executables

<table>
<thead>
<tr>
<th>Line</th>
<th>Exec Count</th>
<th>Avg Exec</th>
<th>Tot Time</th>
<th>Cum Time</th>
<th>Score#1</th>
<th>Score#2</th>
<th>Database</th>
<th>Table</th>
<th>FileName</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>11,611</td>
<td>0.000116</td>
<td>1.345663</td>
<td>1.345663</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>25</td>
<td>1,118</td>
<td>0.000154</td>
<td>0.172549</td>
<td>0.172549</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.115074</td>
<td>0.115074</td>
<td>12.421169</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>33</td>
<td>9,054</td>
<td>0.000009</td>
<td>0.081806</td>
<td>0.665992</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>34</td>
<td>9,054</td>
<td>0.000002</td>
<td>0.017669</td>
<td>0.017669</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

27    IF FIRST-OF(country) THEN DO:
28      EMPTY TEMP-TABLE ttsales.
29    END.
31    FOR EACH order OF customer WHERE YEAR(order.orderdate) = cYear USE-INDEX orderdate:
32      FOR EACH orderline WHERE orderline.ordernum = order.ordernum:
33      RUN cummulventes.
This is not a "Learn how to use indexes 101" workshop

Go see *Proper Care and Feeding of an Index* by Mike Lonski

With that said, here is a simplified version of the rules

1. If there is a "CONTAINS" then use a word-index
2. If an index is unique and all of it's components are used in an equality match, use that index
3. Use the index with the most equality matches on SUCCESSIVE, LEADING INDEX components
4. Use the index with the most active range matches on SUCCESSIVE, LEADING INDEX components
5. Use the index with the most active sort matches
6. Use the first index alphabetically
7. Use the primary index
Sidebar – Multi-Index Use

- Where ... and ...
  - All components of each index involved in equality matches
  - No unique indexes

- Where ... or ...
  - Both sides of OR contain at least the lead component of an index
  - Equality or range matches
Sidebar - 4GL vs SQL = Rules vs Cost

- 4GL and SQL are two different animals
- The 4GL compiler uses rules to pick an index
- The SQL analyzer uses cost statistics to select the lowest cost path to the data
  - You need to run UPDATE STATISTICS

- CAVEAT: update statistics is buggy in earlier versions
  - Search the KB and the release notes
  - I have personally noticed issues up to 10.2B03

- TEST TEST TEST