

# PROGRESS EXCHANGE 2013

DISCOVER. DEVELOP. DELIVER.

## OpenEdge Database Performance Tuning

## Contents

Lab 1 Performance Monitoring .....	10
Part (1a) Promon and the Gather Script .....	10
Part (1b) prostack (Unix) and Progetstack .....	16
Part (1c) Operating System Tools.....	21
Part (1d) db request statement cache .....	27
Lab (2) New performance features.....	35
Part (2a) Demonstration of -lruskips (-lru2skips works the same) .....	35
Part (2b) Demonstration of Things that can effect LRU.....	38
Part (2c) Demonstration of -B2 .....	41
Lab 3 Demonstration of -Nmsg, -prefetchFactor, -prefetchDelay, -prefetchNumRecs.....	45
Lab 4 Use of the proutil increaseto feature .....	49
Part (4a) Demonstration of increaseto -L .....	49
Part (4b) Demonstration of increaseto -B.....	51
Caveat to increaseto due to current logged in users.....	54
Part (4c) Demonstration of increaseto -B2 (Disabling LRU2 Policy) .....	56

## Purpose

This document accompanies the Progress Exchange 2013 Performance Tuning Workshop. It provides step-by-step instructions for the hands-on portions of the Workshop.

## Disclaimer

This document is not a manual. It provides examples of OpenEdge features and methods for monitoring database performance. Complete documentation for using the OpenEdge can be found online here <http://communities.progress.com/pcom/docs/DOC-16074>. Progress Software cannot be held responsible for the content of this document nor for any damage that may occur to your environment.

## Overview

In this workshop, you provide hands-on experience using common tools to analyze performance with an OpenEdge database. This workshop will also provide demonstrations and ways to test using some new performance tuning options for the OpenEdge database.

# LAB 0 Setting up Putty for Connection to the Progress Cloud machines.

## Objective

Configure putty to connect to the Amazon cloud machines using an authentication key file.

## Duration

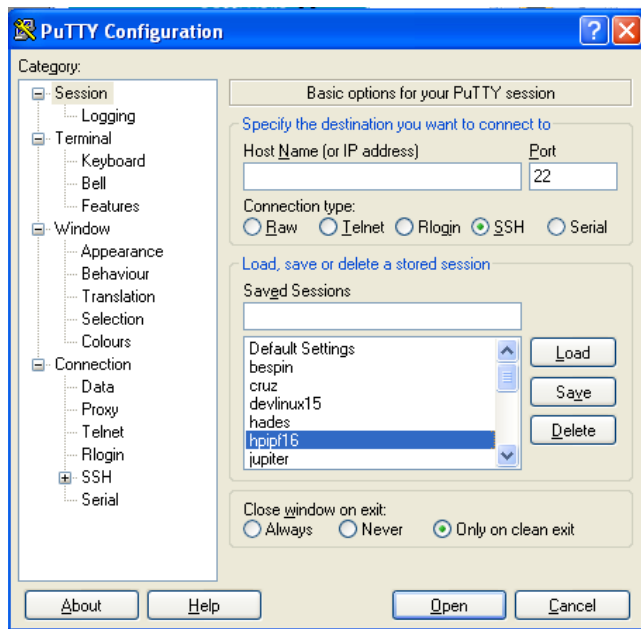
5 Minutes

## Goals

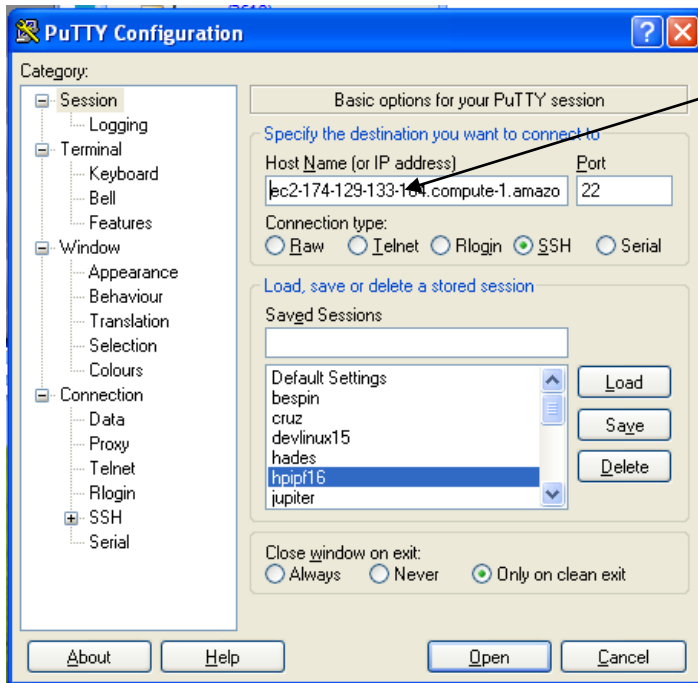
Configure putty to connect to the Amazon cloud machines using an authentication key file.

## Instructions

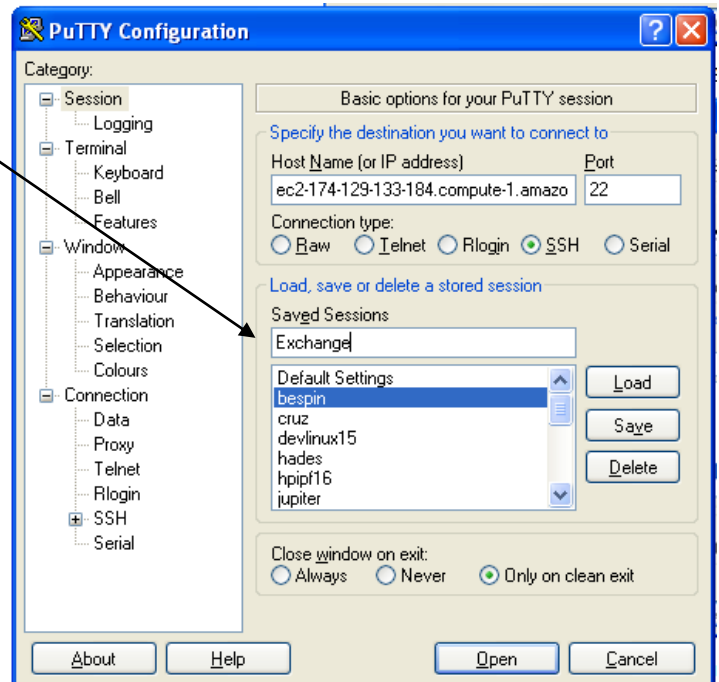
- 1) A putty.ppk file should have been sent to you by email.
- 2) Save this file to a directory on disk that you can easily find afterwards.
- 3) Open putty.exe
- 4) The initial configuration screen for putty should look like this:



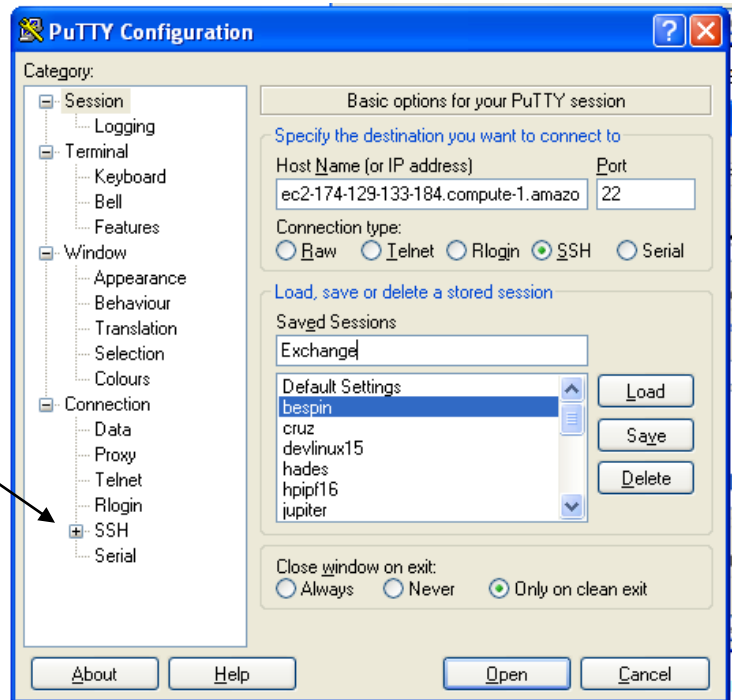
5) Enter the hostname or IP address for the Amazon Cloud system in the Host (or IP address) field



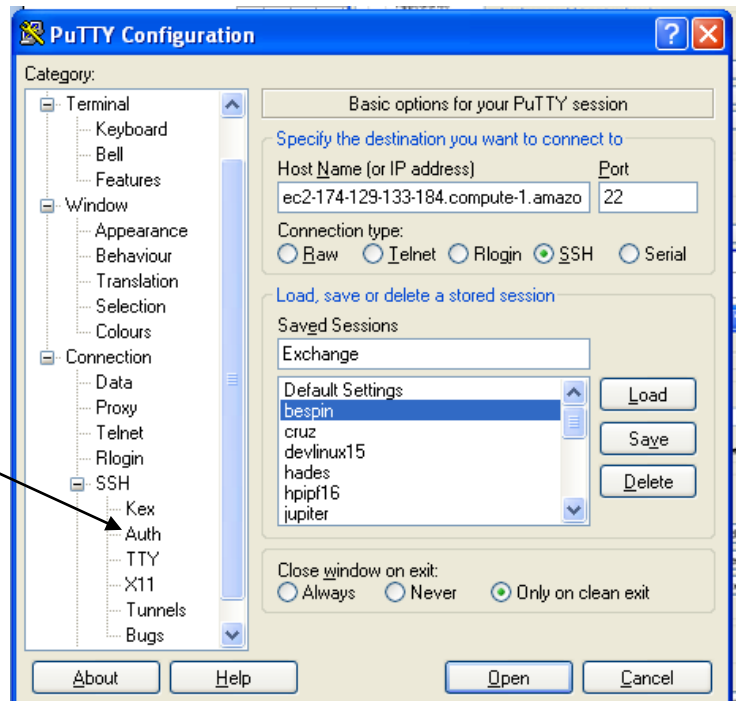
6) After entering the host you can define a saved session name



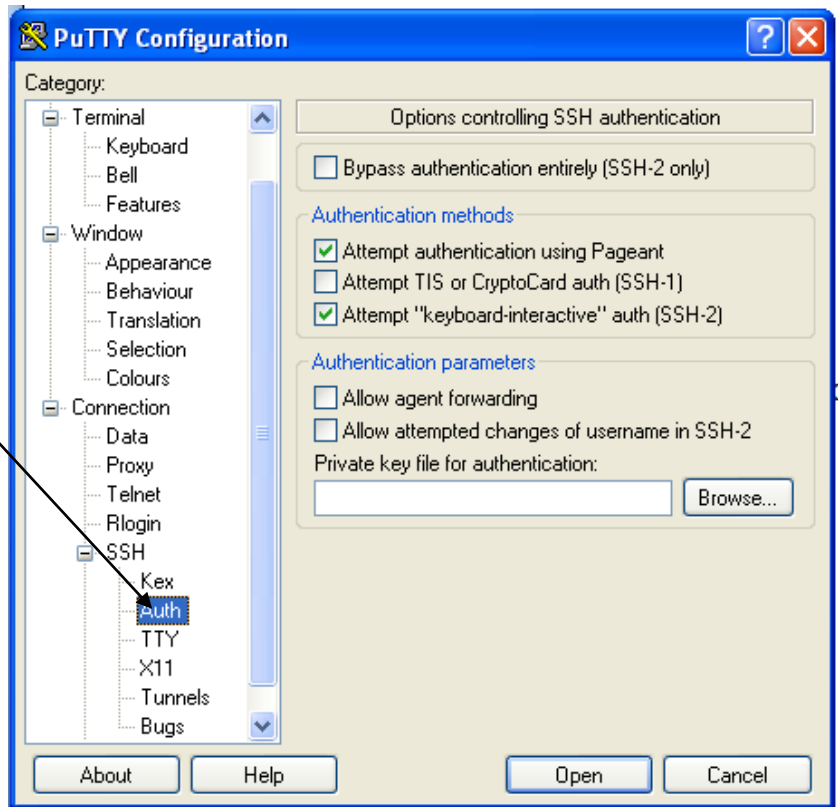
7) Then expand the SSH option in the left pane by clicking on the expand toggle



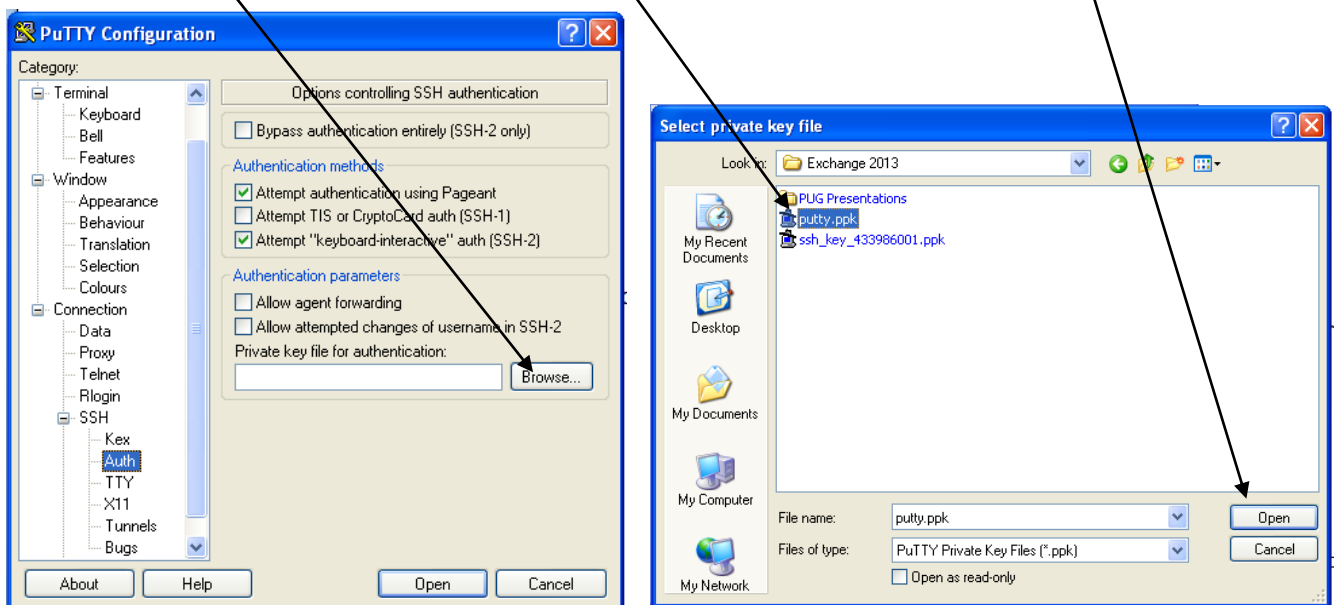
8) The Auth portion of the SSH configuration will now be visible



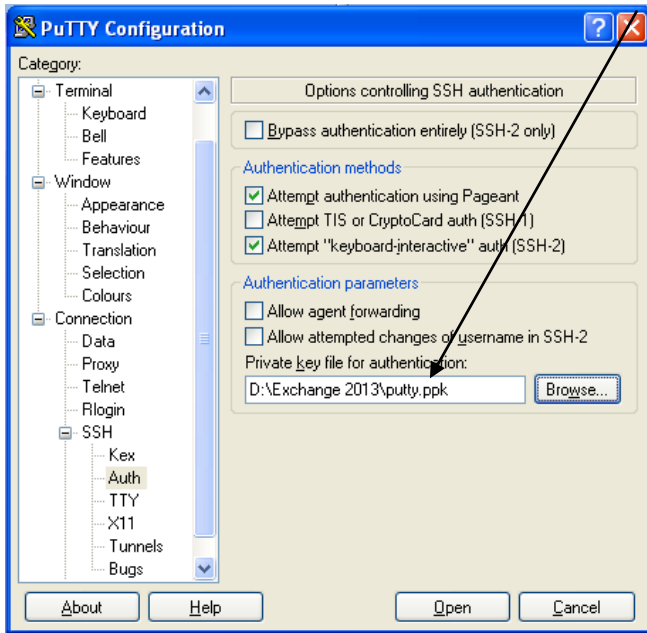
9) Click on the Auth option



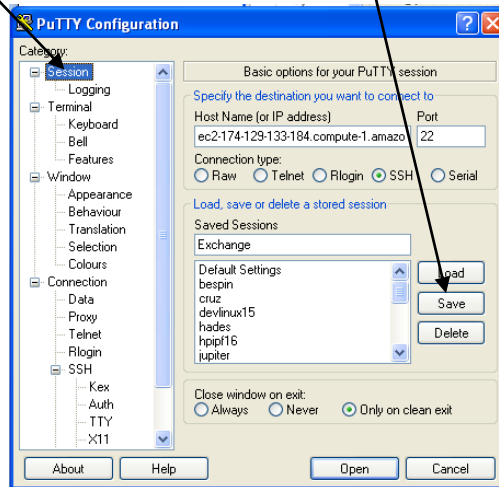
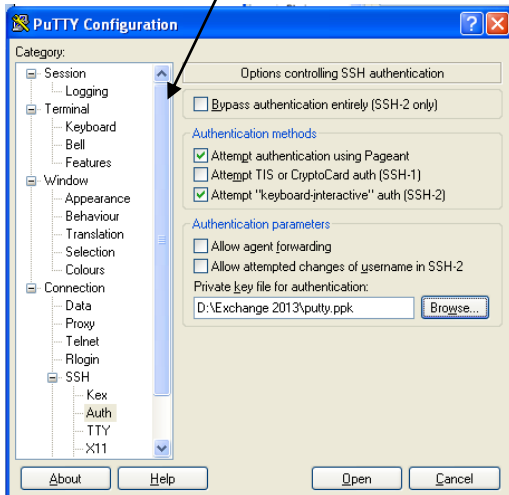
10) Click on the browse button and locate the putty.ppk that was saved in step 2 then click open



11) The private key file for authentication should now be selected

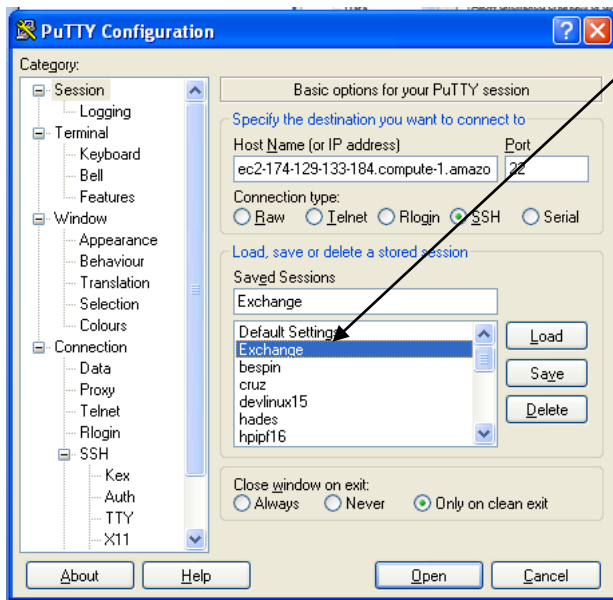


12) To save this configuration you must now scroll up in the left pane of the putty configuration window until Session is visible then click on Session then click on the save button

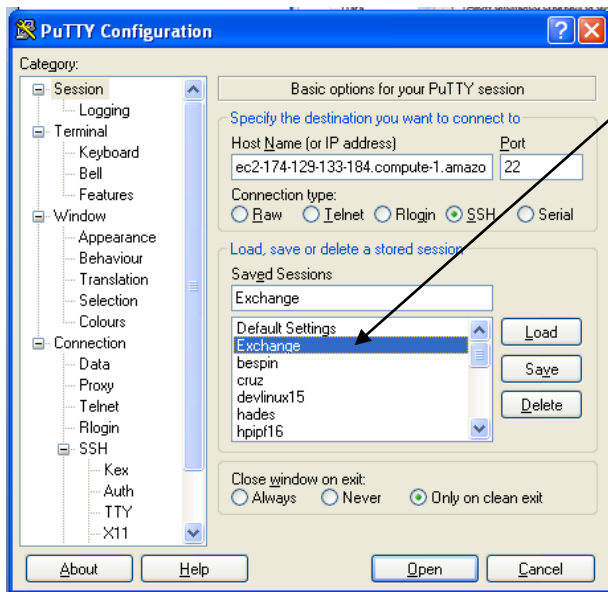




13) This will store the configuration under the name chosen in step 6



14) Throughout the rest of this workshop whenever you are asked to start a new putty session open putty, highlight the named session you saved, then click on open



15) For all putty sessions the root login will be used. No password is necessary.

## **Lab 1 Performance Monitoring**

### **Objectives**

In parts a, b, and c of this lab you will learn how to monitor the OpenEdge database with various tools.

- You will learn to monitor the database with the promon gather script:
- You will learn what the gather script does and does not do for the database administrator?
- Learn common operating system utilities and their role in identifying performance problems.

### **Part (1a) Promon and the Gather Script**

#### **Duration**

20 minutes

#### **Goals**

- In Lab1a you will learn how to monitor the progress database with promon and the gather script which comprehensively collects data to analyze performance and help detect problems.

#### **Instructions**

**Open two or more shell sessions for this exercise.**

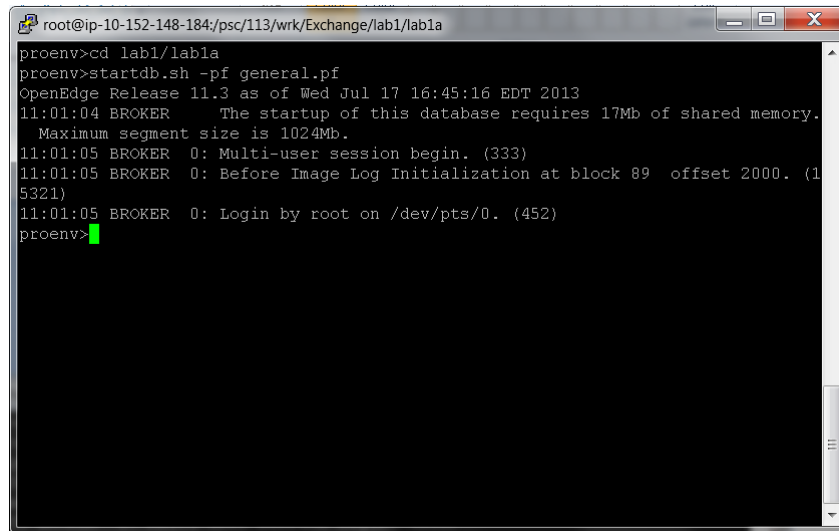
1) Open three putty sessions and type the following commands in both sessions:

```
./proenv
```

```
cd lab1/lab1a
```

2) In the first session type this command to start the database with some general parameters:

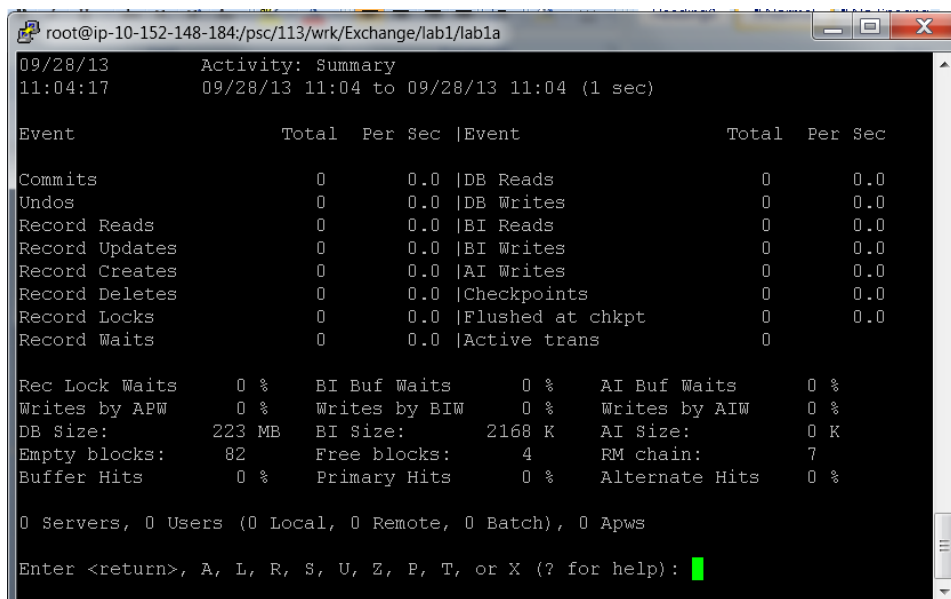
```
startdb.sh -pf general.pf
```



```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1a
proenv>cd lab1/lab1a
proenv>startdb.sh -pf general.pf
OpenEdge Release 11.3 as of Wed Jul 17 16:45:16 EDT 2013
11:01:04 BROKER      The startup of this database requires 17Mb of shared memory.
      Maximum segment size is 1024Mb.
11:01:05 BROKER  0: Multi-user session begin. (333)
11:01:05 BROKER  0: Before Image Log Initialization at block 89  offset 2000. (1
5321)
11:01:05 BROKER  0: Login by root on /dev/pts/0. (452)
proenv>
```

3) in the second session type these command to run Promon against the database and look at the database activity:

```
promon.sh
Enter R&D      (R&D. Advanced options)
Enter 2       (2. Activity Displays ...)
Enter 1       (1. Summary)
Enter Z to zero out the counters
```



```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1a
09/28/13      Activity: Summary
11:04:17      09/28/13 11:04 to 09/28/13 11:04 (1 sec)

Event                Total  Per Sec |Event                Total  Per Sec
-----
Commits              0      0.0 |DB Reads              0      0.0
Undos                0      0.0 |DB Writes             0      0.0
Record Reads         0      0.0 |BI Reads              0      0.0
Record Updates       0      0.0 |BI Writes             0      0.0
Record Creates       0      0.0 |AI Writes             0      0.0
Record Deletes       0      0.0 |Checkpoints           0      0.0
Record Locks         0      0.0 |Flushed at chkpt     0      0.0
Record Waits         0      0.0 |Active trans          0

Rec Lock Waits      0 %   BI Buf Waits        0 %   AI Buf Waits        0 %
Writes by APW       0 %   Writes by BIW       0 %   Writes by AIW       0 %
DB Size:            223 MB  BI Size:             2168 K  AI Size:             0 K
Empty blocks:       82      Free blocks:         4      RM chain:           7
Buffer Hits         0 %   Primary Hits        0 %   Alternate Hits      0 %

0 Servers, 0 Users (0 Local, 0 Remote, 0 Batch), 0 Apws
Enter <return>, A, L, R, S, U, Z, P, T, or X (? for help):
```

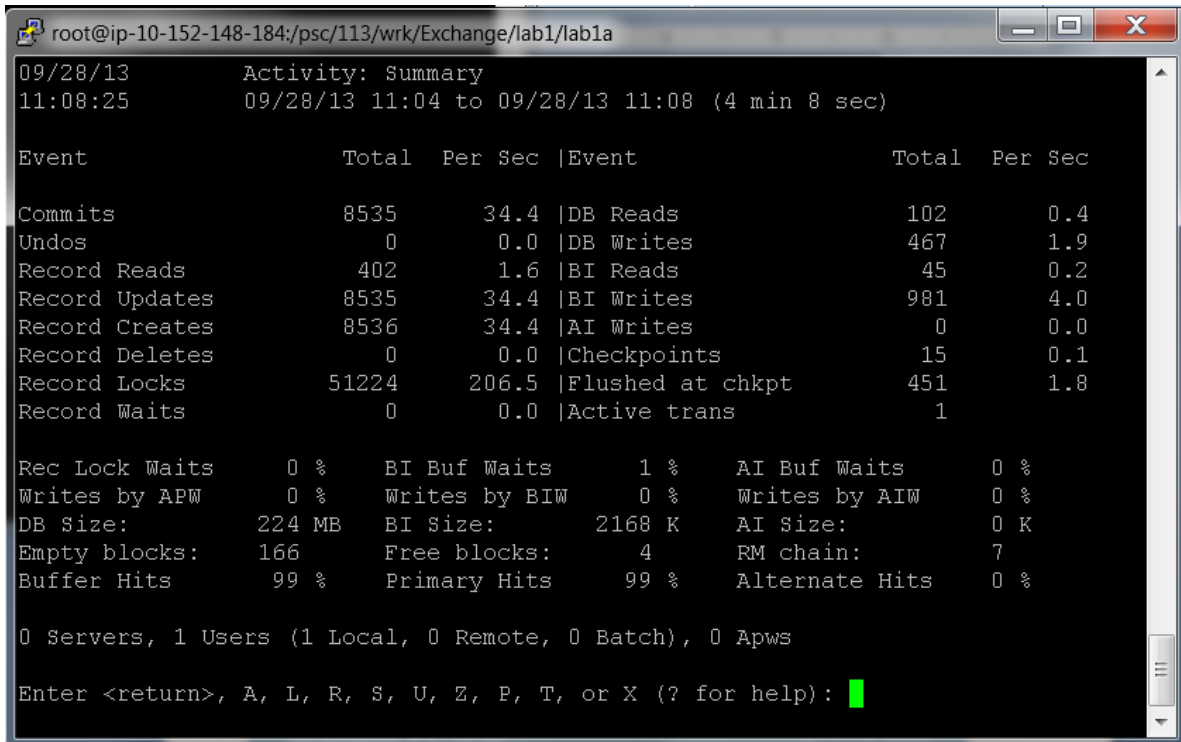
9) In the first session type this command to create some activity against the database:

multiuser.sh -p busywork-no-pause1.p

10) When the program finishes it will say "Press space bar to continue." which will finish the program.

11) At this point go to the second putty session which is running promon

12) press U for update



```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1a
09/28/13 Activity: Summary
11:08:25 09/28/13 11:04 to 09/28/13 11:08 (4 min 8 sec)

Event                Total   Per Sec |Event                Total   Per Sec
-----
Commits              8535    34.4 |DB Reads              102     0.4
Undos                 0        0.0 |DB Writes             467     1.9
Record Reads         402     1.6 |BI Reads              45      0.2
Record Updates       8535    34.4 |BI Writes             981     4.0
Record Creates       8536    34.4 |AI Writes             0        0.0
Record Deletes        0        0.0 |Checkpoints          15      0.1
Record Locks         51224   206.5 |Flushed at chkpt     451     1.8
Record Waits          0        0.0 |Active trans         1

Rec Lock Waits      0 %    BI Buf Waits      1 %    AI Buf Waits      0 %
Writes by APW       0 %    Writes by BIW     0 %    Writes by AIW     0 %
DB Size:            224 MB  BI Size:          2168 K  AI Size:           0 K
Empty blocks:       166    Free blocks:      4      RM chain:         7
Buffer Hits         99 %    Primary Hits     99 %    Alternate Hits    0 %

0 Servers, 1 Users (1 Local, 0 Remote, 0 Batch), 0 Apws
Enter <return>, A, L, R, S, U, Z, P, T, or X (? for help): █
```

13) In the third putty session and issue the following commands:

multiuser.sh -p busywork-with-pause.p

14) press space bar to begin the first phase of work in the third putty session

15) go back to promon and press U for update

```
root@ip-10-152-148-184:/pvc/113/wrk/Exchange/lab1/lab1a
09/28/13 Activity: Summary
11:10:52 09/28/13 11:04 to 09/28/13 11:10 (6 min 35 sec)

Event                Total  Per Sec |Event                Total  Per Sec
-----
Commits              11100   28.1 |DB Reads              102    0.3
Undos                 0        0.0 |DB Writes             601    1.5
Record Reads         926     2.3 |BI Reads              57     0.1
Record Updates      11000   27.8 |BI Writes            1298   3.3
Record Creates      11000   27.8 |AI Writes             0     0.0
Record Deletes       100     0.3 |Checkpoints          19     0.0
Record Locks        66828  169.2 |Flushed at chkpt     581    1.5
Record Waits         0        0.0 |Active trans         0

Rec Lock Waits      0 %    BI Buf Waits      1 %    AI Buf Waits      0 %
Writes by APW       0 %    Writes by BIW     0 %    Writes by AIW     0 %
DB Size:            225 MB  BI Size:          2168 K  AI Size:           0 K
Empty blocks:       147     Free blocks:      4        RM chain:         11
Buffer Hits         99 %    Primary Hits     99 %    Alternate Hits    0 %

0 Servers, 1 Users (1 Local, 0 Remote, 0 Batch), 0 Apws
Enter <return>, A, L, R, S, U, Z, P, T, or X (? for help): █
```

16) Disconnect both client sessions

press space bar in each of the client sessions until you return to the proenv prompt

In some cases because the client is in a tight loop it may be necessary to kill the session and start a new client session.

17) Connect both client sessions again and run busywork-no-pause1.p

multiuser.sh -p busywork-no-pause1.p

18) Use the same promon screen to monitor the data

Press u several times note how quickly the information is changing which makes analysis more difficult.

Note how quickly the information is changing which makes analysis more difficult.

When taken in isolation, promon can easily see everything any one process does and that is clearly visible to you the DBA.

19) Press X to exit the promon session

When more than one client is performing work it can cloud the stream of data and make it more challenging to identify what is happening at a minute, granular level.

20) Use the gather.sh (gather.sh on Unix) script against the database. Issue the following command in the putty session where promon was running:

```
gather.sh perf
```

The perf option is available for the Unix version of the gather script.

This limits the gather to collect performance monitoring data only.

The script for this workshop has been customized to embed a specific database name within the script.

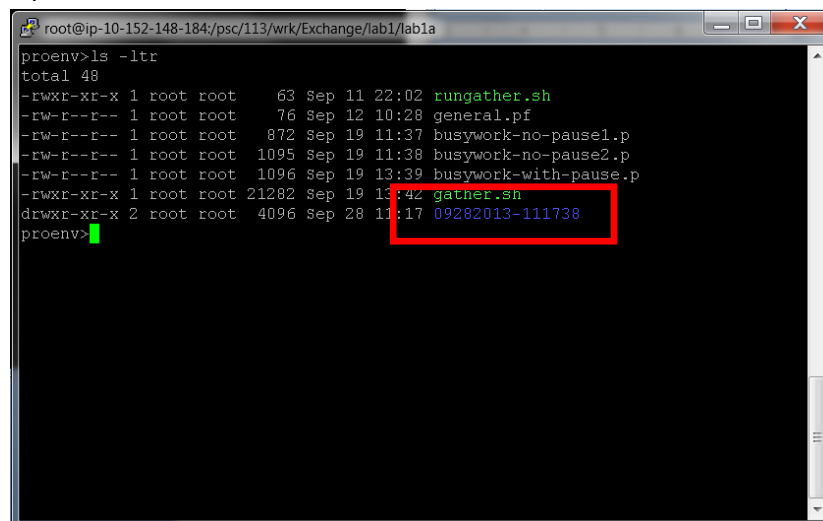
Issue this command from the proenv prompt:

21) Hit the Enter key at least once after the gather script has been started as it will typically not signify it is completed.

22) When the command has returned to the proenv prompt issue the following command:

```
ls -ltr
```

You should see a directory which will be the date and timestamp when the gather script command was run similar to what is pictured here:



```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1a
proenv>ls -ltr
total 48
-rwxr-xr-x 1 root root 63 Sep 11 22:02 rungather.sh
-rw-r--r-- 1 root root 76 Sep 12 10:28 general.pf
-rw-r--r-- 1 root root 872 Sep 19 11:37 busywork-no-pause1.p
-rw-r--r-- 1 root root 1095 Sep 19 11:38 busywork-no-pause2.p
-rw-r--r-- 1 root root 1096 Sep 19 13:39 busywork-with-pause.p
-rwxr-xr-x 1 root root 21282 Sep 19 13:42 gather.sh
drwxr-xr-x 2 root root 4096 Sep 28 11:17 09282013-111738
proenv>
```

23) Use the cd to switch directories into the directory created by the gather script

24) Issue the ls command to view the list of files created by the gather script.

The screen shot below is just an example of the files that gather creates.

```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1a/09282013-111738
proenv>ls -ltr
total 48
-rwxr-xr-x 1 root root 63 Sep 11 22:02 rungather.sh
-rw-r--r-- 1 root root 76 Sep 12 10:28 general.pf
-rw-r--r-- 1 root root 872 Sep 19 11:37 busywork-no-pause1.p
-rw-r--r-- 1 root root 1095 Sep 19 11:38 busywork-no-pause2.p
-rw-r--r-- 1 root root 1096 Sep 19 13:39 busywork-with-pause.p
-rwxr-xr-x 1 root root 21282 Sep 19 13:42 gather.sh
drwxr-xr-x 2 root root 4096 Sep 28 11:17 09282013-111738
proenv>cd 09282013-111738/
proenv>ls
gather.out gatherin.txt iostat-dktx.out sar-q.out vmstat.out
proenv>
```

25) Close all putty sessions.

Without the perf option the gather script on Unix will collect a list of Progress processes running on the system and send non-destructive signals to each process to get stack trace information.

Why is gather so long? Because it is better to throw the kitchen sink at it than hunt and peck for answers when an emergency is occurring. All the files gather created in one simple script instead of starting them manually.

For targeted situations when no critical time crunch is present, hunting and pecking in promon might be better however for emergencies, the gather script collects most of what is needed for problem analysis by Technical Support and Development or general performance tuning investigations.

## Part (1b) prostack (Unix) and Progetstack

(Windows progetstack is not part of this workshop demonstration)

### Duration

5 minutes

### Goals

- In Lab1b you will learn how to trigger OpenEdge executables to create a stack trace which can help in the isolation of performance problems and process hangs.

### Instructions

Prostack and progetstack are Progress tools introduced in 10.1C and written to connect to a running process and trigger the process, if it is responsive, to drop a stack trace including both 4GL and C code stack.

#### prostack syntax on Unix is:

```
prostack { -r | -a Pid } ImageFile [ CoreFile ]
```

Example against \_progres if the PID was 2849:

```
prostack -a 2849 /psc/113/dlc/bin/_progres
```

#### progetstack syntax on Windows is:

```
progetstack {PID}
```

### Sending a Signal to the Process

On Unix other command line tools can be used to trigger the generation of a stack:

1) Start a putty session and type the following commands:

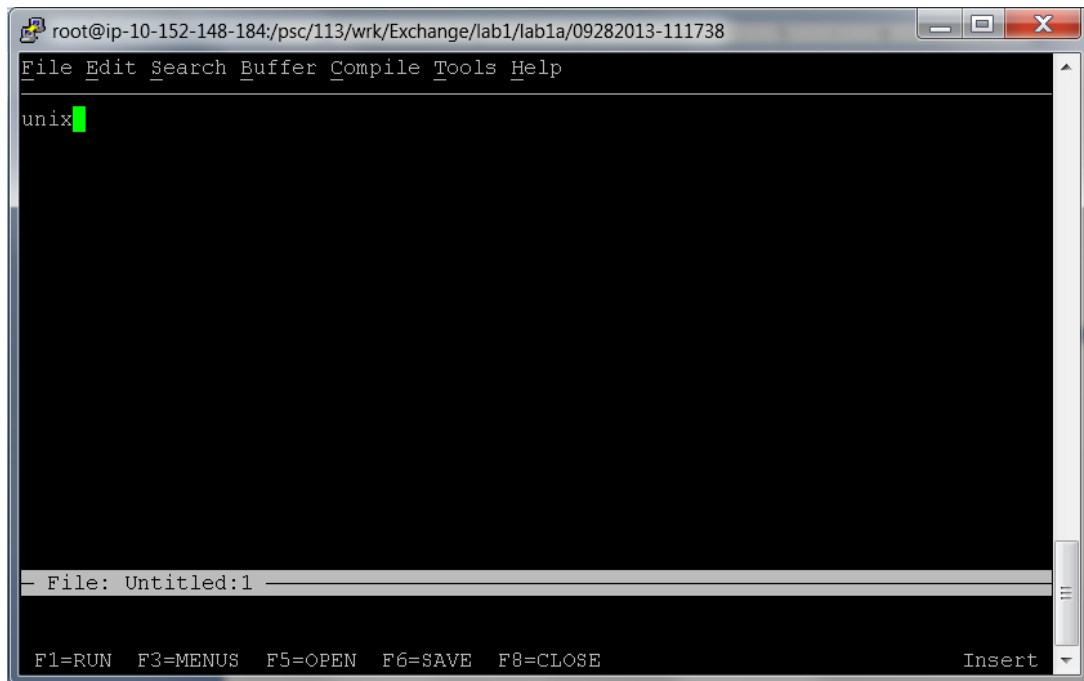
```
./proenv
```

```
cd lab1/lab1b
```

```
pro
```



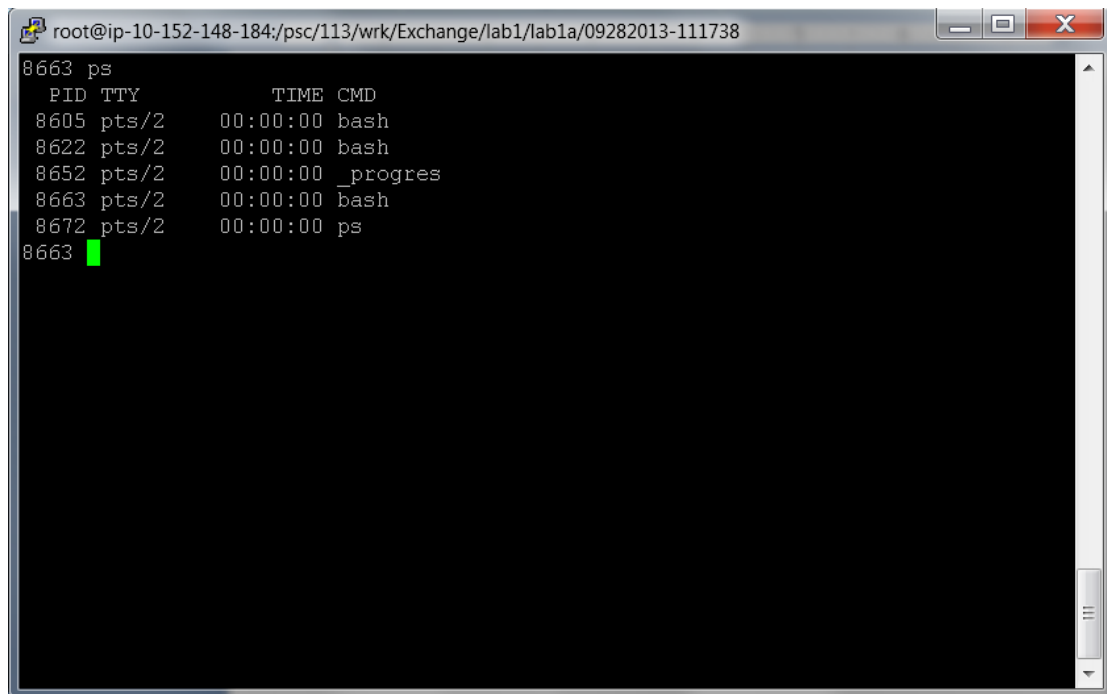
2) Type unix



A screenshot of a terminal window. The title bar shows the path: root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1a/09282013-111738. The menu bar includes File, Edit, Search, Buffer, Compile, Tools, and Help. The main area shows the text 'unix' followed by a green cursor. At the bottom, there is a status bar with 'File: Untitled:1' and function key definitions: F1=RUN, F3=MENUS, F5=OPEN, F6=SAVE, F8=CLOSE, and an Insert key.

3) Type control + x to enter a subshell within the Progress session

4) Type ps to get a list of jobs

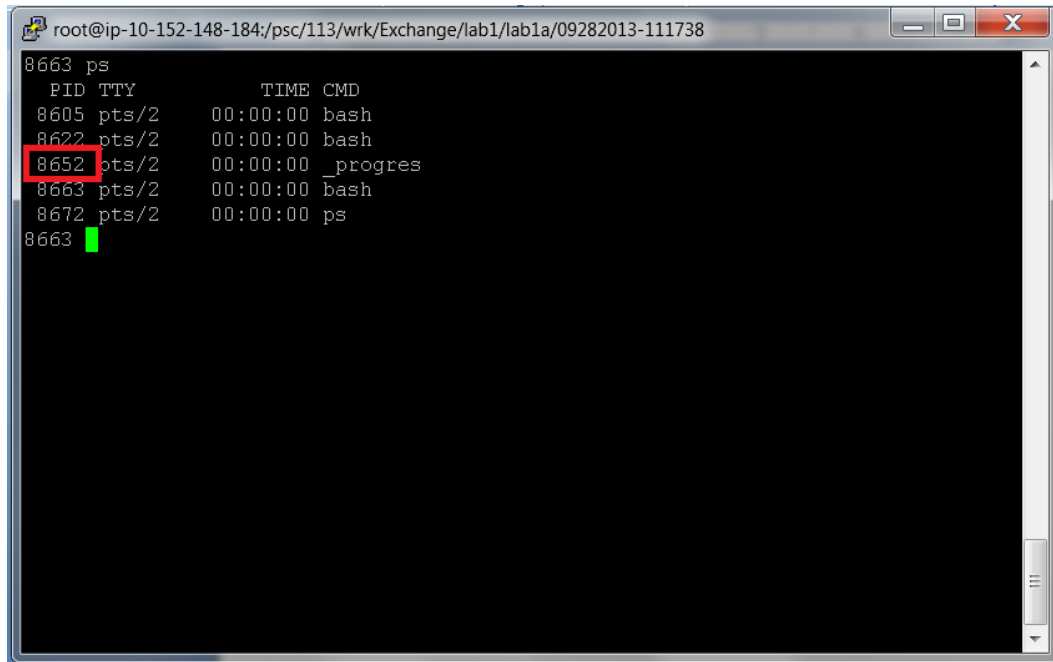


A screenshot of a terminal window showing the output of the 'ps' command. The title bar is the same as in the previous screenshot. The output is as follows:

```
8663 ps
  PID TTY          TIME CMD
 8605 pts/2        00:00:00 bash
 8622 pts/2        00:00:00 bash
 8652 pts/2        00:00:00 _progres
 8663 pts/2        00:00:00 bash
 8672 pts/2        00:00:00 ps
8663
```

5) Identify the PID for the \_progres session which is found in the first column.

Example:



```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1a/09282013-111738
8663 ps
  PID TTY          TIME CMD
 8605 pts/2        00:00:00 bash
 8622 pts/2        00:00:00 bash
 8652 pts/2        00:00:00 _progres
 8663 pts/2        00:00:00 bash
 8672 pts/2        00:00:00 ps
8663 █
```

6) Type `prostack -a <PID of the _progres session> /psc/113/dlc/bin/_progres`

Example:

```
prostack -a 8652 /psc/113/dlc/bin/_progres
```

7) This will bring up a menu system for asking you to choose the operating system you are running on.

```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1a/09282013-111738
8663 pts/2    00:00:00 bash
8672 pts/2    00:00:00 ps
8663 prostack -a 8652 /psc/113/dlc/bin/_progres

*****

'uname -a' for your machine indicates:  Linux ip-10-152-148-184 2.6.32-2
79.14.1.el6.centos.plus.x86_64 #1 SMP Wed Nov 7 00:40:45 UTC 2012 x86_64 x86_64
x86_64 GNU/Linux

Select the option which best describes your UNIX machine:

1. Tru64:      Compaq Tru64
2. AIX:        IBM RS6000
3. HPUX:       Hewlett Packard
4. Sun:        Solaris Intel
5. SunOS:      Sparc Solaris
6. UnixV4:     UnixWare, SCO OS5, or DG/UX Intel
7. Linux:      Any Linux

99. exit

Option: 7
You selected 'Linux' to describe your system. Is this correct (y/n)? █
```

Choose 7 (for Linux which is used for the Exchange labs)

Then enter Y to signify that this choice is correct.

8) For the Linux operating system the GDB debugger will normally be used and this output will be generated:

```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1a/09282013-111738
*****

The following debuggers are available through
Linux:
    /usr/bin/gdb

*****

Using the gdb debugger to produce a stack trace from
image file:
    /psc/113/dlc/bin/_progres
and core file:
    core
Appending output to file gdb.log...

*****

*****

8663 █
```

9) Type exit to exit the subshell and return to the Progress client session and press the space bar to end the procedure.

10) Press the escape key and the M key to enter the menu

ESC + M

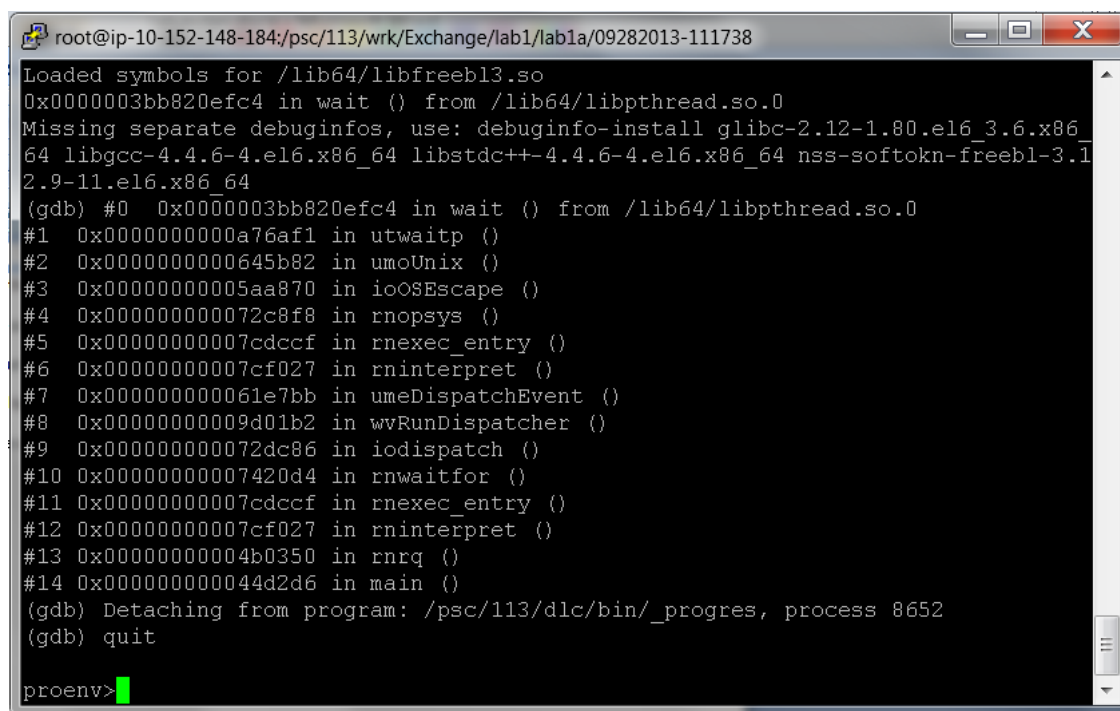
11) Hit the down arrow button on your keyboard and select X for Exit

12) Press the N key to indicate that no code should be saved.

13) Issue the following commands:

```
cat gdb.log
```

Example excerpt of gdb.log output:



```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1a/09282013-111738
Loaded symbols for /lib64/libfreebl3.so
0x0000003bb820efc4 in wait () from /lib64/libpthread.so.0
Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.80.el6_3.6.x86_
64 libgcc-4.4.6-4.el6.x86_64 libstdc++-4.4.6-4.el6.x86_64 nss-softokn-freebl-3.1
2.9-11.el6.x86_64
(gdb) #0  0x0000003bb820efc4 in wait () from /lib64/libpthread.so.0
#1  0x0000000000a76af1 in utwaitp ()
#2  0x0000000000645b82 in umoUnix ()
#3  0x00000000005aa870 in ioOSEscape ()
#4  0x000000000072c8f8 in rnopsys ()
#5  0x00000000007cdccf in rnexec_entry ()
#6  0x00000000007cf027 in rninterpret ()
#7  0x000000000061e7bb in umeDispatchEvent ()
#8  0x00000000009d01b2 in wwRunDispatcher ()
#9  0x000000000072dc86 in iodispatch ()
#10 0x00000000007420d4 in rnwaitfor ()
#11 0x00000000007cdccf in rnexec_entry ()
#12 0x00000000007cf027 in rninterpret ()
#13 0x00000000004b0350 in rnrq ()
#14 0x000000000044d2d6 in main ()
(gdb) Detaching from program: /psc/113/dlc/bin/_progres, process 8652
(gdb) quit
proenv>
```

## Part (1c) Operating System Tools

### Duration

10 minutes

### Goals

In Lab1c you will learn how to monitor the system the Progress database is running on with various operating system tools

### Instructions

- 1) open four putty sessions to the server
- 2) arrange the sessions so that each can be seen
- 3) type the following in each session

```
./proenv  
cd lab1/lab1c
```

- 4) In the first of the four putty sessions give this command  
sar -q 5 100
- 6) In the second of the four putty sessions give this command  
iostat -dktx 5 100
- 7) In the third of the four putty sessions give this command  
vmstat 5 100
- 8) In the fourth of the four putty sessions give this command  
./busywork.sh

**NOTE:** It may be beneficial to resize each of the sessions to prevent line wrap for each of the tools.

9) In the session with sar -q notice the high runq-sz

```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1d
12:11:18      10      121      11.82      4.33      1.88
12:11:23      13      121      12.16      4.53      1.95
12:11:28      13      121      12.54      4.73      2.03
12:11:33      13      121      12.50      4.86      2.09
12:11:38      13      121      12.78      5.04      2.16
12:11:43      14      121      13.04      5.22      2.24
12:11:48      13      121      13.27      5.40      2.31

12:11:48      runq-sz  plist-sz  ldavg-1   ldavg-5   ldavg-15
12:11:53          2      121      13.49      5.58      2.38
12:11:58          14      120      13.69      5.75      2.46
12:12:03          13      119      13.16      5.77      2.48
12:12:08          5      119      13.06      5.87      2.53
12:12:13          3      119      12.26      5.82      2.53
12:12:18          13      119      12.32      5.94      2.59
12:12:23          13      119      12.37      6.06      2.65
12:12:28          13      119      12.42      6.17      2.70
12:12:33          13      119      12.47      6.29      2.76
12:12:38          5      119      12.51      6.40      2.81
12:12:43          13      119      12.55      6.51      2.87
12:12:48          13      119      12.02      6.50      2.88
12:12:53          11      119      12.10      6.61      2.94
12:12:58          13      119      12.17      6.71      2.99
```

10) In the session with the iostat observe the %util

```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1d
Device:            rrqm/s    wrqm/s     r/s     w/s    kB/s    kB/s  avgrq-sz  avgqu-sz   await  svctm   %util
xvdap1             0.00    6384.20    2.00    627.80   61.60  27336.00   87.00   147.81  220.75   1.59  100.00
xvdb               0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00

09/28/13 12:11:43
Device:            rrqm/s    wrqm/s     r/s     w/s    kB/s    kB/s  avgrq-sz  avgqu-sz   await  svctm   %util
xvdap1             0.00    6758.00    0.20    717.80   0.80  30591.20   85.21   180.06  262.97   1.39  100.00
xvdb               0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00

09/28/13 12:11:48
Device:            rrqm/s    wrqm/s     r/s     w/s    kB/s    kB/s  avgrq-sz  avgqu-sz   await  svctm   %util
xvdap1             0.00    6158.00    4.60    619.20   29.60  26980.80   86.60   143.17  229.14   1.60  100.00
xvdb               0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00

09/28/13 12:11:53
Device:            rrqm/s    wrqm/s     r/s     w/s    kB/s    kB/s  avgrq-sz  avgqu-sz   await  svctm   %util
xvdap1             0.00    6658.60    1.00    691.60    4.00  29456.80   85.07   184.47  267.35   1.44  100.00
xvdb               0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00

09/28/13 12:11:58
Device:            rrqm/s    wrqm/s     r/s     w/s    kB/s    kB/s  avgrq-sz  avgqu-sz   await  svctm   %util
xvdap1             0.00    4278.00   11.00   510.00  128.00  20344.00   78.59   98.97  195.15   1.45   75.72
xvdb               0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00

09/28/13 12:12:03
Device:            rrqm/s    wrqm/s     r/s     w/s    kB/s    kB/s  avgrq-sz  avgqu-sz   await  svctm   %util
xvdap1             0.00     8.00     0.00     7.60     0.00     62.40   16.42     0.09  11.84   0.76   0.58
xvdb               0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00

09/28/13 12:12:08
Device:            rrqm/s    wrqm/s     r/s     w/s    kB/s    kB/s  avgrq-sz  avgqu-sz   await  svctm   %util
xvdap1             0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
xvdb               0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
```

11) In the session with the vmstat watch the following columns:

r: The number of processes waiting for run time.

b: The number of processes in uninterruptible sleep.

free: the amount of idle memory (kB).

pi aka bi: Blocks sent to a block device (blocks/s).

po aka bo: Blocks received from a block device (blocks/s).

us user time

sy system time

wa: Time spent waiting for IO.

```

root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1d
13 0 0 1733136 36944 117680 0 0 0 0 1996 783 94 1 5 0 0
 1 0 0 1733136 36944 117680 0 0 0 0 2011 786 96 1 4 0 0
13 0 0 1733136 36952 117680 0 0 0 2 2029 802 96 1 4 0 0
13 0 0 1733136 36952 117680 0 0 0 0 2012 802 95 1 5 0 0
 3 0 0 1733384 36952 117680 0 0 0 0 2032 803 95 1 4 0 0
13 0 0 1733384 36952 117680 0 0 0 0 2007 776 95 1 4 0 0
13 0 0 1733384 36952 117680 0 0 0 0 2011 793 95 1 5 0 0
13 0 0 1733384 36960 117680 0 0 0 2 2020 785 95 1 4 0 0
13 0 0 1733384 36960 117680 0 0 0 1 2009 783 95 1 4 0 0
13 0 0 1733384 36960 117680 0 0 0 2 2000 803 94 1 5 0 0
13 0 0 1733384 36960 117680 0 0 0 0 2015 780 96 1 3 0 0
13 0 0 1733384 36960 117680 0 0 0 0 2026 805 95 1 4 0 0
13 0 0 1733384 36960 117680 0 0 0 0 2030 795 96 1 4 0 0
 1 0 0 1733384 36960 117680 0 0 0 0 1985 792 94 1 6 0 0
13 0 0 1733384 36960 117680 0 0 0 0 2017 779 96 0 3 0 0
procs -----memory-----swap-----io-----system-----cpu-----
 r b swpd free buff cache si so bi bo in cs us sy id wa st
13 0 0 1733384 36960 117680 0 0 0 0 2014 790 95 1 4 0 0
13 0 0 1733260 36960 117680 0 0 0 0 2016 782 96 1 4 0 0
 0 0 0 7333520 36960 118180 0 0 182 0 2016 752 75 21 3 1 0
 0 0 0 7338544 36968 114612 0 0 71 25 44 36 0 0 99 1 0
14 0 0 6641820 37056 170300 0 0 172 31 468 287 6 13 80 1 0
 0 4 0 1304364 37404 529548 0 0 74 10326 2077 821 26 67 5 2 0
13 2 0 1149900 37548 675324 0 0 51 29537 2569 1630 87 7 3 4 0
 1 3 0 998404 37704 825164 0 0 0 30212 2599 1579 91 7 1 1 0
12 3 0 850844 37840 964772 0 0 0 28936 2873 1725 91 7 1 1 0
13 3 0 714568 37980 1102208 0 0 0 28607 2687 1756 90 7 1 1 0
14 3 0 557832 38128 1252028 0 0 0 30346 2582 1511 91 7 1 1 0
14 2 0 411636 38260 1390008 0 0 0 27659 2685 1760 91 7 1 2 0
14 2 0 269656 38412 1534512 0 0 0 29441 2587 1533 90 7 1 2 0

```

### Windows perfmon (not part of Workshop--intended for information only)

Windows perfmon can be controlled through its graphic interface or from command line.

perfmon launches the graphic interface where counters can be selected for the OS to monitor.

Physical Disk

Logical Disk



Memory

CPU (processors)

Processes

Can all be monitored. For some of these things it is beneficial to monitor all instances.

Some like Physical and Logical disks it is important to monitor each instance (disk) separately for proper performance analysis.

**Windows logman (not part of Workshop--intended for information only)**

The Windows command line tool logman can be used to control the starting and stopping of Perfmon data collection in Windows.

## **UNIX sar, iostat, vmstat: The Unix trilogy of tools.**

Sar -u collects data regarding CPU usage

Sar -q collects queue depth for processors

Sar -d collects disk usage information

iostat (does not exist on all platforms) can also collect data on disk / device usage.

Vmstat collects data on virtual memory usage but also has some useful information on CPU utilization and blocked process counts.

## Part (1d) db request statement cache

### Duration

10 minutes

### Goals

- In Lab1d you will learn how to identify performance problems caused by code which might lock excessive records; read more records than expected; or similar performance degrading behavior.
- Learn how to identify the user and ultimately the procedure and perhaps even the line of code responsible.

### Instructions

- 1) start two putty sessions with the following commands

```
./proenv  
cd lab1/lab1d
```

- 2) In putty session 1 start the database and have it listen on a port

```
./dbstart.sh
```

- 3) Start a promon session against the database on your machine

```
promon.sh
```

4) Enter the following commands in promon

Enter R&D (R&D. Advanced options)

Enter 2 (2. Activity Displays ...)

Enter 1 (1. Summary)

Enter Z to zero the counters

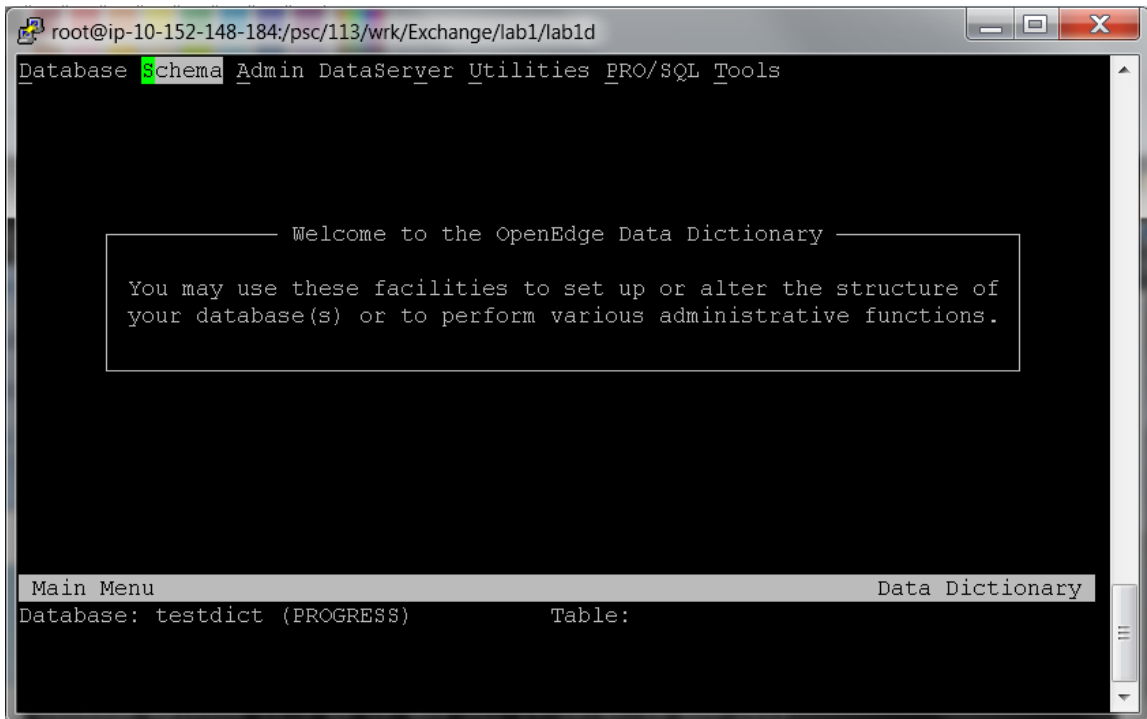
Enter A to turn on automatic iterations

```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1d
18:42:03 09/28/13 18:41 to 09/28/13 18:41 (37 sec)
Event          Total  Per Sec |Event          Total  Per Sec
Commits         0    0.0 |DB Reads       40    1.1
Undos           0    0.0 |DB Writes      3     0.1
Record Reads    1    0.0 |BI Reads       39    1.1
Record Updates  0    0.0 |BI Writes      1     0.0
Record Creates  0    0.0 |AI Writes      0     0.0
Record Deletes  0    0.0 |Checkpoints    0     0.0
Record Locks    0    0.0 |Flushed at chkpt 0     0.0
Record Waits    0    0.0 |Active trans   0
Rec Lock Waits  0 %  BI Buf Waits  0 %  AI Buf Waits  0 %
Writes by APW  0 %  Writes by BIW  0 %  Writes by AIW  0 %
DB Size:      225 MB  BI Size:      2168 K  AI Size:      0 K
Empty blocks: 150    Free blocks:  4    RM chain:    10
Buffer Hits   94 %  Primary Hits  94 %  Alternate Hits 0 %
0 Servers, 0 Users (0 Local, 0 Remote, 0 Batch), 0 Apws
Enter <return>, A, L, R, S, U, Z, P, T, or X (? for help): a
Iteration 1 of 9999, pause for 10 seconds ...
```

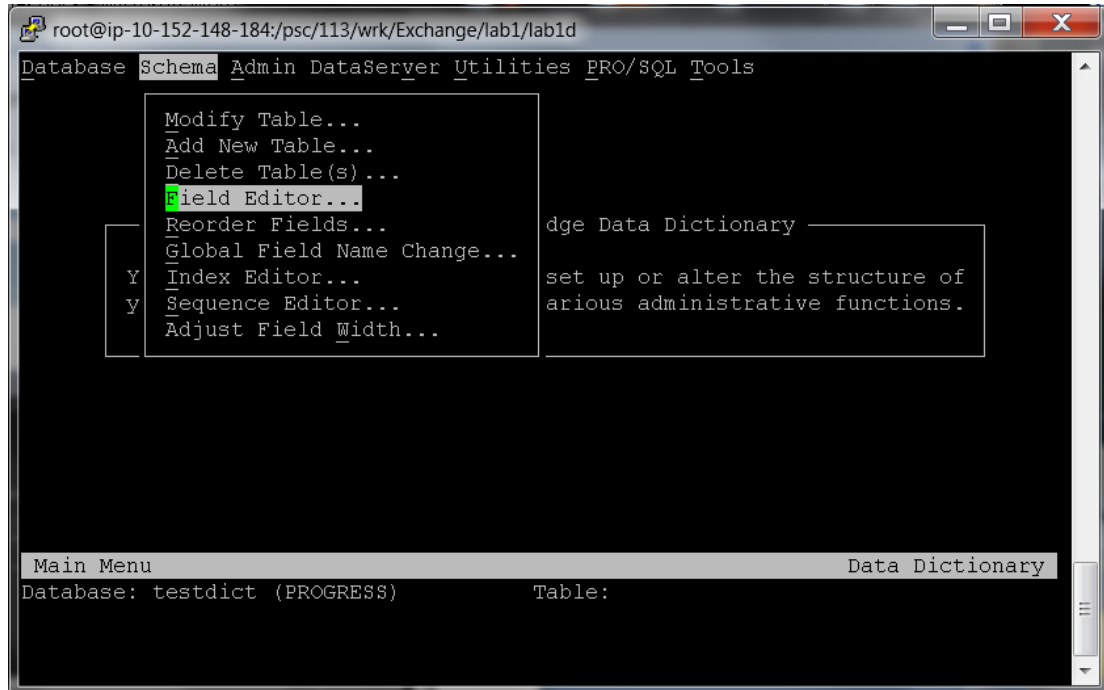
5) In the second putty session start a client session against the database and run the Data Dictionary with this command:

```
./clientstart.sh
```

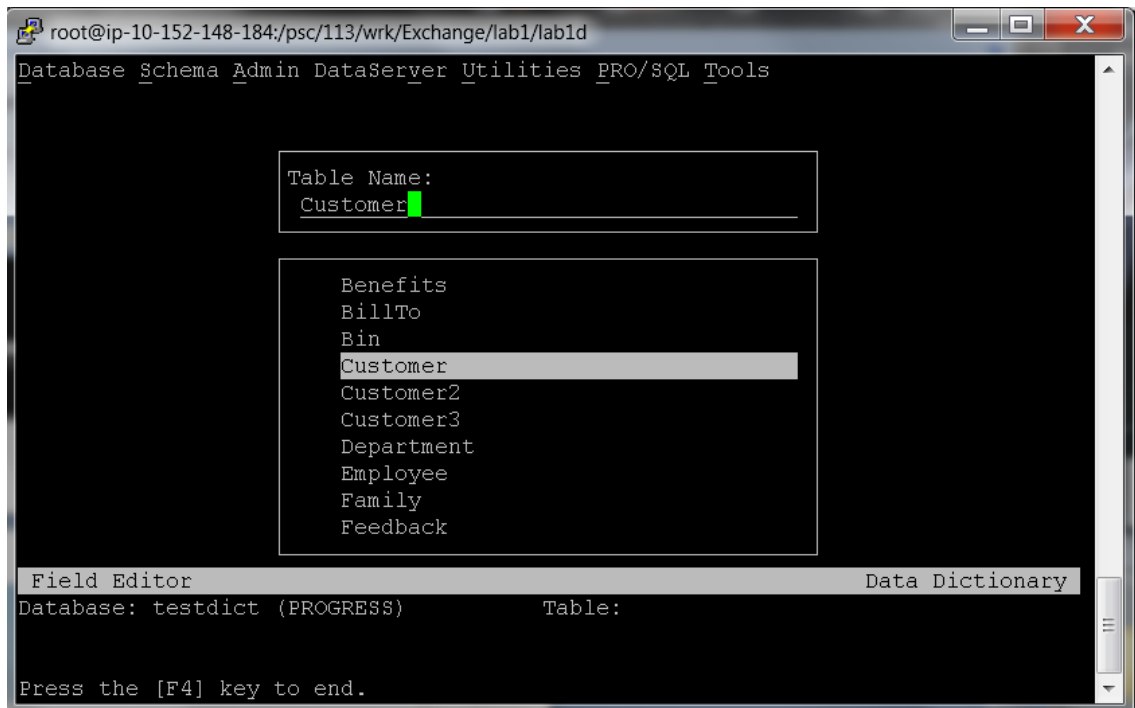
## 6) Select the Schema Menu



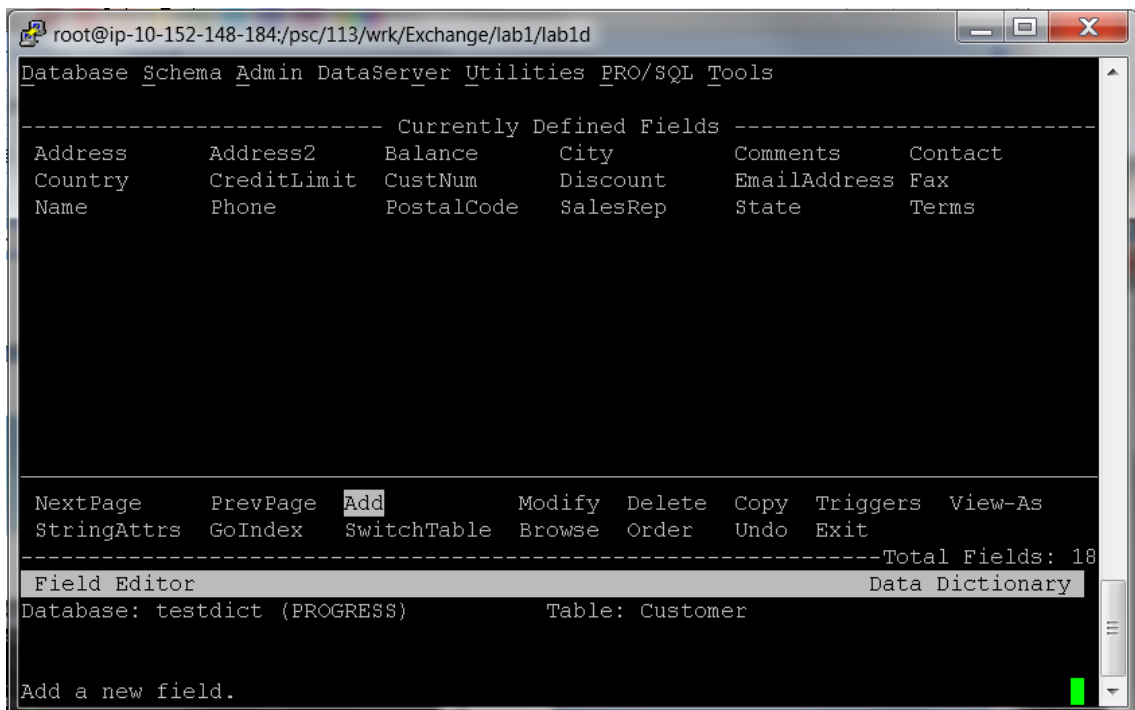
## 7) Hit the down arrow on the keyboard and select Field Editor



8) Select the Customer table



9) Select Add to add a Field



10) Specify testchar as the field name and hit enter

11) Enter character as the field type and hit enter

```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1d
Database Schema Admin DataServer Utilities PRO/SQL Tools
-----
Field-Name: testfield          Data-Type: char
Format:                        Extent:
Label:                          Decimals:
Column-label:                   Order:
Initial:                        Mandatory: (Not Null)
Component of-> View: no   Index: no   Position:      Case-sensitive:
Valexp:
Valmsg:
Help:
Desc:
-----
NextPage   PrevPage   Add      Modify   Delete   Copy   Triggers   View-As
StringAttr GoIndex   SwitchTable Browse   Order   Undo   Exit
-----
-----Total Fields: 18
Field Editor                                     Data Dictionary
Database: testdict (PROGRESS)          Table: Customer
Enter data or press F4 to end.
```

12) At this point the Data Dictionary should hang.

13) Return to the promon session started in step 3

Notice the tremendous activity occurring. Imagine this were a program used within your company where some new program isn't behaving as expected. Imagining more than one program was recently added and now you don't know what program is the culprit.

```

root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1d
09/28/13      Activity: Summary
18:53:04      09/28/13 18:52 to 09/28/13 18:53 (10 sec)

Event          Total  Per Sec |Event          Total  Per Sec
-----
Commits         0      0.0 |DB Reads       979    97.9
Undos           0      0.0 |DB Writes      915    91.5
Record Reads   29243  2924.3 |BI Reads       48     4.8
Record Updates 29244  2924.4 |BI Writes      680    68.0
Record Creates  0      0.0 |AI Writes      0      0.0
Record Deletes  0      0.0 |Checkpoints    8      0.8
Record Locks   58487  5848.7 |Flushed at chkpt 907    90.7
Record Waits    0      0.0 |Active trans   1      1

Rec Lock Waits  0 %   BI Buf Waits  1 %   AI Buf Waits  0 %
Writes by APW  0 %   Writes by BIW  0 %   Writes by AIW  0 %
DB Size:      225 MB  BI Size:      9336 K  AI Size:      0 K
Empty blocks: 150   Free blocks:  4   RM chain:    10
Buffer Hits   98 %   Primary Hits  98 %   Alternate Hits 100 %

1 Servers, 1 Users (0 Local, 1 Remote, 0 Batch), 0 Apws
Iteration 67 of 9999, pause for 10 seconds ...

```

14) Enter control-C to stop the automatic iterations

15) Enter T for Top

Since there were many locks being created by the user let's focus on who is performing the most locks.

Enter 3 (3. Other Displays ...)

Enter 3 (3. Lock Requests By User)



Identify the user number with the most locks

```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab1/lab1d
09/28/13      Lock Requests By User
18:53:31

  Usr:Ten   User      Domain      --- Record ---      ---- Trans ---      --- Sche
ma ---
           Name              Locks   Waits      Locks   Waits      Locks
Waits
    0       root        0         0         0       0         0
    0
    1       root        0         0         0       0         0
    0
    5       root        0         0         0       0         0
    0
    24      root        -4        274524    0       0         1
    0

Enter <return>, R, U, P, T, or X (? for help): █
```

- a) Enter T for Top
- b) Enter 1 (1. Status Displays ...)
- c) Enter 18 (18. Client Database-Request Statement Cache ...)
- d) Enter 2 (2. Activate For All Users)
- e) Enter 1 (1-Single)
- f) Select 7 (7. View Database-Request Statement Cache)

Select the user (there will likely be only one unless you have strayed from the script)

The code which is creating the excessive volume of work will be listed.

If you choose to repeat this step for the full stack steps 3 to 7 above can be repeated but a new field name must be chosen each time.

16) In the promon session type the following commands:

- a) Enter T for Top
- b) Enter 1 (1. Status Displays ...)
- c) Enter 18 (18. Client Database-Request Statement Cache ...)
- d) Enter 2 (2. Activate For All Users)
- e) Enter 2 (2-Stack)
- f) Select 7 (7. View Database-Request Statement Cache)

17) To end the client session, when the message "can you find my code name?" is on the screen follow these steps:

- a) Press the F4 key on the keyboard  
It may take a few seconds for it to backout the temporary work.
- b) The Schema Menu should be highlighted.
- c) Press the D key to select the Database menu.
- d) Press X to exit.

## Lab (2) New performance features

### Objective

In lab 2 you will learn the benefits to performance using the new -lruskips and lru2skips parameters. Database administrators will be advised on best practices to review metrics that can identify if lruskips or lruskips2 should be used.

### Part (2a) Demonstration of -lruskips (-lru2skips works the same)

#### Duration

8 minutes

#### Goals

Part (a) will provide hands-on demonstration of the -lruskips (and by extension the -lru2skips) parameter and how it may benefit the performance of your database.

#### Instructions

1) open three putty sessions and issue the following commands in each

```
./proenv
```

```
cd lab2/lab2a
```

2) In the first putty session issue this command

```
./busywork.sh
```

3) In the second putty session issue this command

```
promon.sh < ./promoninput
```

Note the High values for LRU (if you don't see it immediately just wait about 10 seconds).

Owner	----- Total	Locks ----- /Sec	----- /Sec	Busy ----- Pct	Naps /Sec	----- /Sec	Spins ----- /Lock	----- /Busy	----- Total	Nap Max ----- HWM
MTX --	0	0	0	0.0	0	0	0	0	0	0
USR --	0	0	0	0.0	0	0	0	0	0	0
OM --	0	0	0	0.0	0	0	0	0	0	0
BIB --	0	0	0	0.0	0	0	0	0	0	0
SCH --	0	0	0	0.0	0	0	0	0	0	0
LKP --	0	0	0	0.0	0	0	0	0	0	0
GST --	0	0	0	0.0	0	0	0	0	0	0
TXT --	0	0	0	0.0	0	0	0	0	0	0
LKT --	0	0	0	0.0	0	0	0	0	0	0
LKT --	0	0	0	0.0	0	0	0	0	0	0
LKT --	0	0	0	0.0	0	0	0	0	0	0
LKT --	0	0	0	0.0	0	0	0	0	0	0
SEQ --	0	0	0	0.0	0	0	0	0	0	0
AIB --	0	0	0	0.0	0	0	0	0	0	0
TXQ --	2	0	0	0.0	0	0	0	0	0	0
EC --	0	0	0	0.0	0	0	0	0	0	0
LKF --	0	0	0	0.0	0	0	0	0	0	0
BFP --	0	0	0	0.0	0	0	0	0	0	0
BHT --	767302	127883	25	0.0	61	619816	4	24306	0	80
EWQ --	0	0	0	0.0	0	0	0	0	0	0
CFQ --	0	0	0	0.0	0	0	0	0	0	0
LRU --	732243	122040	55	0.0	95	1165379	9	21124	4	40
LRU --	0	0	0	0.0	0	0	0	0	0	0
BUF --	395624	65937	0	0.0	2	25258	0	151550	0	40
BUF --	342776	57129	0	0.0	4	40437	0	80875	0	40
BUF --	397355	66225	0	0.0	3	30585	0	45878	0	40
BUF --	346516	57752	0	0.0	2	23681	0	71044	0	40
INC --	0	0	0	0.0	0	0	0	0	0	0
L29 --	0	0	0	0.0	0	0	0	0	0	0
L30 --	0	0	0	0.0	0	0	0	0	0	0
L31 --	0	0	0	0.0	0	0	0	0	0	0

4) In the third putty session issue this command

- promon.sh
- Enter R&D (R&D. Advanced options)
- Enter 4 (4. Administrative Functions ...)
- Enter 4 (4. Adjust Latch Options)
- Enter 4 (4. Adjust LRU force skips: 0)
- Enter 20 (new value for LRU force skips)

Note the decrease in the LRU latching amounts in the second promon screen..

Owner	Total	Locks /Sec	Busy /Sec	Pct	Naps /Sec	Spins /Lock	Nap Max
MTX	0	0	0	0.0	0	0	0
USR	0	0	0	0.0	0	0	0
OM	0	0	0	0.0	0	0	0
BIB	0	0	0	0.0	0	0	0
SCH	0	0	0	0.0	0	0	0
LKP	0	0	0	0.0	0	0	0
GST	0	0	0	0.0	0	0	0
TXT	0	0	0	0.0	0	0	0
LKT	0	0	0	0.0	0	0	0
LKT	0	0	0	0.0	0	0	0
LKT	0	0	0	0.0	0	0	0
LKT	0	0	0	0.0	0	0	0
SEQ	0	0	0	0.0	0	0	0
AIB	0	0	0	0.0	0	0	0
TXQ	2	0	0	0.0	0	0	0
EC	0	0	0	0.0	0	0	0
LKF	0	0	0	0.0	0	0	0
BFP	0	0	0	0.0	0	0	0
BHT	810657	135109	30	0.0	72	733509	80
FWQ	0	0	0	0.0	0	0	0
CRQ	0	0	0	0.0	0	0	0
LRU	47621	7936	0	0.0	37	378048	40
LRU	0	0	0	0.0	0	0	0
BUF	418040	69673	0	0.0	3	31759	40
BUF	368070	61345	0	0.0	3	32230	40
BUF	415635	69272	0	0.0	5	50096	40
BUF	362496	60416	0	0.0	4	43813	40
INC	0	0	0	0.0	0	0	0
L29	0	0	0	0.0	0	0	0
L30	0	0	0	0.0	0	0	0
L31	0	0	0	0.0	0	0	0

As LRUSKIPS is increased from the default the contention on LRU will shift to the BUF buffer latches which have gone up as the LRU latching has gone down.

10) Play with varying values for LRU force skips in the third putty session and observe the behavior of LRU and BUF.

## Part (2b) Demonstration of Things that can effect LRU

### Duration

5 Minutes

### Goals

Show negative impact of insufficient value for -omsize.

### Instruction

1) open two putty sessions and issue the following commands in each

```
./proenv  
cd lab2/lab2b
```

2) In the first putty session issue this command

```
startdb.sh -omsize 10
```

3) In the second putty session issue this command

```
multiuser.sh
```

In the procedure editor type this command but don't run the command yet.

```
for each _file no-lock: end.
```

4) In the first putty session issue this command

```
promon.sh < ./promonauto
```

**NOTE:** You may want to resize the screen so that the columns don't wrap around.

5) In the putty session with mpro issue the GO command

```
F1 or CTRL-X
```

6) In the putty session with promon notice the large number of locks for the OM latch.

7) End the client session with the following commands:

Press the ESC + M key to enter the menu  
Press F for File  
Press X to exit

8) Let's demonstrate reading from a larger table with an omsize value that is too small  
`multiuser.sh -p measuring-transmission-time.p`

Note the values for OM and LRU in the promon session.

Note the etime duration.

9) Let's start the database back up again in the first putty session issue this command

`startdb.sh -omsizes 2048`

10) In the first putty session issue this command

`promon.sh < ./promonauto`

**NOTE:** You may want to resize the screen so that the columns don't wrap around.

11) In the second putty session issue this command

`multiuser.sh -p measuring-transmission-time.p`

Observe the promon session.

What changes did you observe for the OM latch?

What changes did you see for the LRU latch?

How different was the etime duration?

12) Let's restart the database and alter the value for the client private buffer pool limit with the following command:

`startdb.sh -B 100000 -Bpmax 25000`

13) Let's run the etime code again with this command:

`multiuser.sh -p measuring-transmission-time.p -Bp 1000`

While the difference isn't significant for this small test the greater benefit will be for large groups of users. Performing concurrent operations against a very busy buffer pool.



## Part (2c) Demonstration of -B2

### Duration

15 Minutes

### Goals

Show performance benefits of the alternate buffer pool; the method to enable areas or specific tables and indices to use the alternate buffer pool. How to chart benefits from use of the alternate buffer pool. Learn what things can enable the LRU2 latching for the alternate buffer pool and how to disable the LRU2 latch if it has accidentally been enabled.

### Notes

This database has three copies of the customer table with 500k+ records each  
The first customer table is in the cust\_data area which is type I  
The second customer table is in the cust\_data2 area with is Type II  
The third customer table is in the cust\_data3 area which is Type II and enabled for B2 usage  
Each of these tables has its own index area in a corresponding area type (type I index area for type I data area and type II index area for type II data area)

### Instruction

1) Start a putty session and issue the following commands

```
./proenv
```

```
cd lab2/lab2c
```

2) Start the database with a reasonable amount of buffers for the primary buffer pool:

```
startdb.sh -B 100000
```

3) Start a client session and read all the customer records:

```
multiuser.sh -p customer.p
```

NOTE the elapsed time to read all the customer records from the Type I area.

```
Message
-----
The average for each for customer table took 3159 milliseconds.
-----
<OK>
```

Press Enter to clear the message.

4) Restart the database and allocate a reasonable amount of buffers for the primary and alternate buffer pools:

```
startdb.sh -B 100000 -B2 100000
```

5) Start a client session and read all the customer2 table records in a Type II area:

```
multiuser.sh -p customer2.p
```

NOTE the elapsed time to read all the customer records from the Type II area.

```
Message
-----
The average for each for customer2 table took 3138 milliseconds.
-----
<OK>
```

6) Start Restart the database and allocate a reasonable amount of buffers for the primary and alternate buffer pools:

```
startdb.sh -B 100000 -B2 100000
```

7) Start a client session and read all the customer3 table records in a Type II area which has been enabled to use the alternate buffer pool:

```
multiuser.sh -p customer3.p
```

Note the decrease in elapsed time when using the alternate buffer pool.

```
Message
-----
The average for each for customer3 table took 3011.75 milliseconds.
-----
<OK>
```

8) Let's check to see if the LRU2 Replacement policy was enabled by checking promon:

promon <dbname>

Enter R&D (R&D. Advanced options)

Enter 2 (2. Activity Displays ...)

Enter 3 (3. Buffer Cache)

```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab2/lab2c
09/29/13 Activity: Buffer Cache
19:10:27 09/29/13 19:08 to 09/29/13 19:10 (1 min 29 sec)

          Total          Per Min          Per Sec          Per Tx
Database Buffer Pool
Logical reads      4028688      2715969      45266.16      0.00
Logical writes           0           0           0.00      0.00
O/S reads          16911         11401         190.01      0.00
O/S writes           3           2           0.03      0.00
Checkpoints           0           0           0.00      0.00
Marked to checkpoint 0           0           0.00      0.00
Flushed at checkpoint 0           0           0.00      0.00
Writes deferred      0           0           0.00      0.00
LRU skips            0           0           0.00      0.00
LRU writes           0           0           0.00      0.00
APW enqueues         0           0           0.00      0.00
Database buffer pool hit ratio: 99 %

Primary Buffer Pool
Logical reads       1542         1040         17.33      0.00
Logical writes           0           0           0.00      0.00
O/S reads           104           70           1.17      0.00
O/S writes           3           2           0.03      0.00
Marked to checkpoint 0           0           0.00      0.00
Flushed at checkpoint 0           0           0.00      0.00
Writes deferred      0           0           0.00      0.00
LRU skips            0           0           0.00      0.00
LRU writes           0           0           0.00      0.00
APW enqueues         0           0           0.00      0.00
Primary buffer pool hit ratio: 93 %

Alternate Buffer Pool
Logical reads      4027146      2714930      45248.83      0.00
Logical writes           0           0           0.00      0.00
O/S reads          16807         11331         188.84      0.00
O/S writes           0           0           0.00      0.00
Marked to checkpoint 0           0           0.00      0.00
Flushed at checkpoint 0           0           0.00      0.00
Writes deferred      0           0           0.00      0.00
LRU2 skips           0           0           0.00      0.00
LRU2 writes           0           0           0.00      0.00
APW enqueues         0           0           0.00      0.00
Alternate buffer pool hit ratio: 99 %
LRU2 replacement policy disabled.
```

If the LRU2 Replacement policy has been enabled then the alternate buffer pool has been exhausted which will negatively affect performance.

Consider using proutil <dbname> -C increaseto -B2 <larger B2 size>

This will be demonstrated later in the workshop.

NOTE: In the past probkup would overwhelm the alternate buffer pool due to a bug.

This has been fixed in 10.2B05 and 11.0.

## Lab 3 Demonstration of -Nmsg, -prefetchFactor, -prefetchDelay, -prefetchNumRecs.

### Objective

Demonstrate methods to improve communication time between remote clients and servers when the application code uses prefetch, no-lock, or scrolling queries.

Considering which options are better for individual client connections and multiple concurrent client connections.

Observing the trade-offs (more / less packets versus delay in time) using the new network parameters.

### Duration

15 Minutes

### Instruction

1) Open two putty sessions and issue the following commands in each

```
./proenv  
cd lab3
```

2) In the first putty session issue this command to start the database listening on port 7077 with a small buffer pool of 10000:

```
startdb.sh -S 7077 -B 10000
```

3) In the first putty session start a promon session against the database and issue the follow commands in promon:

```
promon.sh
```

```
Enter R&D          (R&D. Advanced options)
```

```
Enter 2          (2. Activity Displays ...)
```

```
Enter 2          (2. Servers)
```

```
Enter Z to zero the counters
```

4) In the second putty session run this command to connect to the database via TCP and query the customer table and display the elapsed time:

```
multiuser.sh -S 7077 -H localhost -p measuring-transmission-time.p
```

This will serve as a baseline for comparison versus the new parameters added later in this lab.

5) A message will be displayed in the client session:

```
" Just finished populating the database buffer pool."
```

At this point return to the putty session that contains promon

```
Enter Z to zero the counters
```

6) Press the Enter key in the Progress client session to run the code to collect the elapsed time.

```
Note the etime value now.
```

7) Hit U in the promon session to collect a sample

```
Note the messages sent and received
```

8) Hit X to exit the promon session.

9) Restart the database using the additional parameter -Mm 8192

```
startdb.sh -S 7077 -B 10000 -Mm 8192
```

10) In the first putty session start a promon session against the database and issue the follow commands in promon:

```
promon.sh
```

```
Enter R&D          (R&D. Advanced options)
```

```
Enter 2           (2. Activity Displays ...)
```

```
Enter 2           (2. Servers)
```

```
Enter Z to zero the counters
```

11) Run the etime code with the additional parameter -Mm

```
multiuser.sh -S 7077 -p measuring-transmission-time.p -Mm 8192
```

12) A message will be displayed in the client session:

```
" Just finished populating the database buffer pool."
```

At this point return to the putty session that contains promon

```
Enter Z to zero the counters
```

13) Press the Enter key in the Progress client session to run the code to collect the elapsed time.

```
Note the etime value now.
```

14) Hit U in the promon session to collect a sample

```
Note the messages sent and received
```

15) Hit X to exit the promon session.

16) Restart the database adding the additional parameter -prefetchDelay

```
startdb.sh -S 7077 -B 10000 -prefetchDelay -Mm 8192
```

17) In the first putty session start a promon session against the database and issue the follow commands in promon:

```
promon.sh
Enter R&D          (R&D. Advanced options)
Enter 2           (2. Activity Displays ...)
Enter 2           (2. Servers)
Enter Z to zero the counters
```

18) Run the etime code with the additional parameter -Mm

```
multiuser.sh -S 7077 -p measuring-transmission-time.p -Mm 8192
```

19) A message will be displayed in the client session:

```
" Just finished populating the database buffer pool."
```

At this point return to the putty session that contains promon

```
Enter Z to zero the counters
```

20) Press the Enter key in the Progress client session to run the code to collect the elapsed time.

Note the etime value now.

21) Hit U in the promon session to collect a sample

Note the messages sent and received

22) Hit X to exit the promon session.

Which went down or up? Is that what you expected.

23) Restart the database again and add -prefetchFactor

```
startdb.sh -S 7077 -B 10000 -prefetchDelay -prefetchFactor 90 -Mm 8192
```

24) Run the etime code with the additional parameter -Mm

```
multiuser.sh -S 7077 -p measuring-transmission-time.p -Mm 8192
```

25) A message will be displayed in the client session:

```
" Just finished populating the database buffer pool."
```

At this point return to the putty session that contains promon

Enter Z to zero the counters

26) Press the Enter key in the Progress client session to run the code to collect the elapsed time.

Note the etime value now.

27) Hit U in the promon session to collect a sample

Note the messages sent and received

28) Hit X to exit the promon session.

Which went down or up? Is that what you expected.



## Lab 4 Use of the proutil increaseto feature.

### Objective:

- Lab 4 will simulate common problems related to the database startup parameters (-B, -B2, -L, -aibufs, -bibufs, and -Mxs).
- Some caveats to the proutil increaseto function will be shown.

## Part (4a) Demonstration of increaseto -L

### Duration

5 Minutes

### Instruction

1) Start a putty session

2) Enter the following commands:

```
./proenv  
cd lab4/lab4a
```

3) Start the database with a small value for -L  
startdb.sh -L 500

4) Start a client session and run a piece of code to fill the -L:  
multiuser.sh -p lock-lots-of-records.p

The session will have died because the Lock Table has been exhausted.

Lock table overflow, increase -L on server (915)

5) Issue these commands within the client session:

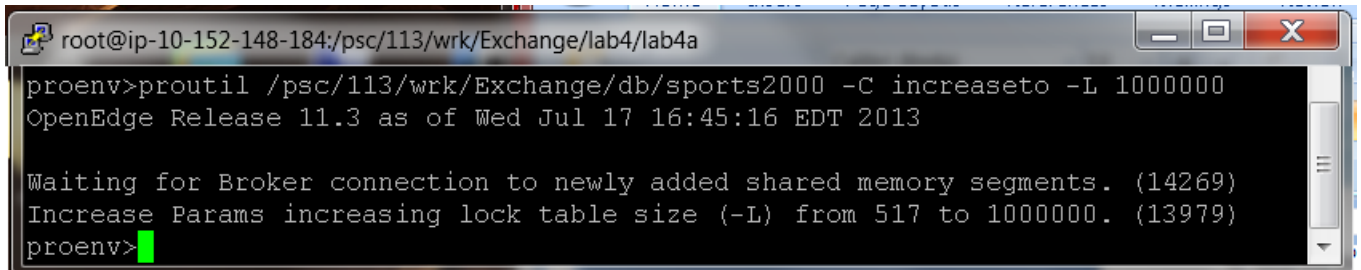
Press the F4 key within the client session.  
Enter the following in the procedure editor:  
quit

Type the following control key sequence:  
CTRL + X

6) Issue the following command:

```
proutil /psc/113/wrk/Exchange/db/sports2000 -C increaseto -L 1000000
```

This will be the screen results:

A terminal window with a title bar showing the path 'root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab4/lab4a'. The terminal content shows a 'proenv' prompt followed by the command 'proutil /psc/113/wrk/Exchange/db/sports2000 -C increaseto -L 1000000'. The output includes 'OpenEdge Release 11.3 as of Wed Jul 17 16:45:16 EDT 2013', a message 'Waiting for Broker connection to newly added shared memory segments. (14269)', and a message 'Increase Params increasing lock table size (-L) from 517 to 1000000. (13979)'. The prompt returns to 'proenv>' with a green cursor.

```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab4/lab4a
proenv>proutil /psc/113/wrk/Exchange/db/sports2000 -C increaseto -L 1000000
OpenEdge Release 11.3 as of Wed Jul 17 16:45:16 EDT 2013
Waiting for Broker connection to newly added shared memory segments. (14269)
Increase Params increasing lock table size (-L) from 517 to 1000000. (13979)
proenv>
```

7) Start a client session and run the same piece of code:

```
multiuser.sh -p lock-lots-of-records.p
```

Note the client did not die because the lock table was not exhausted.

## Part (4b) Demonstration of increaseto -B

### Duration

8 Minutes

### Instruction

1) Open two putty sessions and issue the following commands:

```
./proenv
```

```
cd lab4/lab4b
```

2) In the first putty session issue the following commands to start the database with a small buffer pool:

```
startdb.sh -B 1000 -L 1000000
```

3) Run promon against the database:

```
promon.sh <./promonauto
```

NOTE: You may want to resize the putty session to prevent the column wrapping.

4) In the second putty session run this command:

```
./twosessions.sh
```

5) In the putty session running promon:

Notice the high contention

Owner	Total	Locks /sec	Busy /sec	Busy Pct	Naps /sec	Spins /Sec	Spins /Lock	Spins /Busy	Total	Max HWM
MTX	0	0	0	0.0	0	0	0	0	0	0
USR	4	0	0	0.0	0	0	0	0	0	0
OM	0	0	0	0.0	0	0	0	0	0	0
BIB	0	0	0	0.0	0	0	0	0	0	0
SCH	2	0	0	0.0	0	0	0	0	0	0
LKP	2	0	0	0.0	0	0	0	0	0	0
GST	0	0	0	0.0	0	0	0	0	0	0
TXT	0	0	0	0.0	0	0	0	0	0	0
LKT	501099	83516	76	0.0	0	12236	0	159	0	0
LKT	501084	83514	81	0.0	0	13492	0	164	1	10
LKT	501144	83524	78	0.0	0	14107	0	178	0	0
LKT	501088	83514	106	0.1	0	19785	0	185	0	0
SEQ	0	0	0	0.0	0	0	0	0	0	0
AIB	0	0	0	0.0	0	0	0	0	0	0
TXQ	2	0	0	0.0	0	0	0	0	0	0
EC	0	0	0	0.0	0	0	0	0	0	0
LKF	2004413	334068	779	0.2	0	1269290	3	1627	3	10
BFP	0	0	0	0.0	0	0	0	0	0	0
BHT	2071622	345270	11	0.0	0	10092	0	865	0	0
FMQ	0	0	0	0.0	0	0	0	0	0	0
CPQ	0	0	0	0.0	0	0	0	0	0	0
LRU	2013455	335575	470	0.1	0	2310488	6	4912	2	10
LRU	0	0	0	0.0	0	0	0	0	0	0
BUF	1007874	167979	0	0.0	0	0	0	0	0	0
BUF	1017632	169605	0	0.0	0	0	0	0	0	0
BUF	1020203	170033	0	0.0	0	0	0	0	0	0
BUF	1007654	167942	0	0.0	0	0	0	0	0	0
INC	0	0	0	0.0	0	0	0	0	0	0
L29	0	0	0	0.0	0	0	0	0	0	0
L30	0	0	0	0.0	0	0	0	0	0	0
L31	0	0	0	0.0	0	0	0	0	0	0

6) In the putty session where the mpro was run exit the mpro session hit Enter to return to the proenv command prompt.

7) In the putty session running promon let's end the promon session even if it isn't finished with its iterations by issuing this control sequence:

CTRL + C

8) In the first putty session issue the following command:

proutil /psc/113/wrk/Exchange/db/sports2000 -C increaseto -B 100000

9) In the first putty session start promon against the database

promon.sh <./promonauto

10) In the second putty session run this command:

```
./twosessions.sh
```

11) In the putty session where promon is running notice the reduced contention

```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab4/lab4b
09/29/13 Activity: Latch Counts
20:20:46 09/29/13 20:20 to 09/29/13 20:20 (6 sec)

Owner      Total      Locks      /Sec      /Sec      Busy      Pct      Naps      /Sec      Spins      /Lock      /Busy      Total      Nap      Max      HMM
-----
MTX --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
USR --      4           0           0           0           0.0       0           0           0           0           0           0           0           0
CM  --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
BIB --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
SCH --      2           0           0           0           0.0       0           0           0           0           0           0           0           0
LKP --      2           0           0           0           0.0       0           0           0           0           0           0           0           0
GST --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
TXT --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
LKT --      250276      41712      0           0           0.0       0           0           0           0           0           0           0           0
LKT --      250277      41712      0           0           0.0       0           0           0           0           0           0           0           10
LKT --      250276      41712      0           0           0.0       0           0           0           0           0           0           0           0
LKT --      250268      41711      0           0           0.0       0           0           0           0           0           0           0           0
SEQ --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
AIB --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
TXQ --      2           0           0           0           0.0       0           0           0           0           0           0           0           0
EC  --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
LKF --      1001097     166849     0           0           0.0       0           0           0           0           0           0           0           10
BFP --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
BHT --      1034731     172455     0           0           0.0       0           0           0           0           0           0           0           0
FMQ --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
CPQ --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
LRU --      1005627     167604     0           0           0.0       2           0           0           0           0           0           0           10
LRU --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
BUF --      521139      86856      0           0           0.0       0           0           0           0           0           0           0           0
BUF --      485336      80889      0           0           0.0       0           0           0           0           0           0           0           0
BUF --      522817      87136      0           0           0.0       0           0           0           0           0           0           0           0
BUF --      495136      82522      0           0           0.0       0           0           0           0           0           0           0           0
INC --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
L29 --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
L30 --      0           0           0           0           0.0       0           0           0           0           0           0           0           0
L31 --      0           0           0           0           0.0       0           0           0           0           0           0           0           0

Iteration 3 of 20, pause for 5 seconds ...
```

## Caveat to increase to due to current logged in users.

### Duration

5 Minutes

### Instruction

- 1) Open two putty sessions and issue the following commands

```
./proenv
```

```
cd lab4/lab4c
```

- 2) In the first putty session start the database with this command to intentionally give very few database buffers

```
startdb.sh
```

- 3) Start promon against the database

```
promon.sh
```

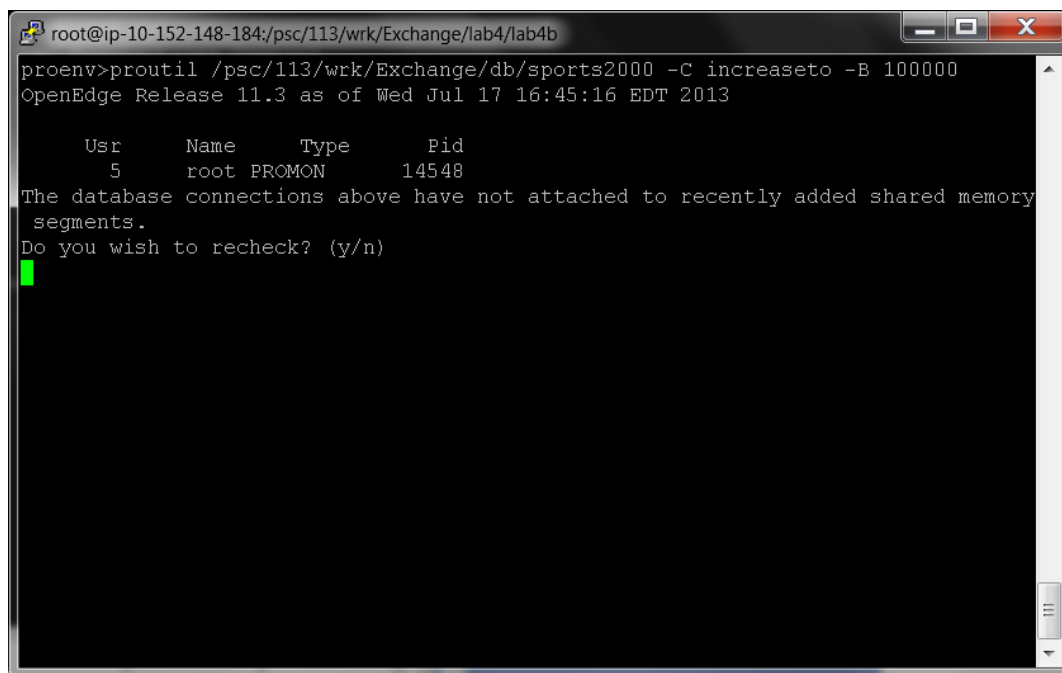
- 4) In the second putty session issue the following commands:

```
proutil /psc/113/wrk/Exchange/db/sports2000 -C increase2 -B 1000
```

Note the message:

Users already connect may not be able to immediately connect to the newly allocated shared memory.

If not, the proutil function will display those users that must be disconnected and reconnected:



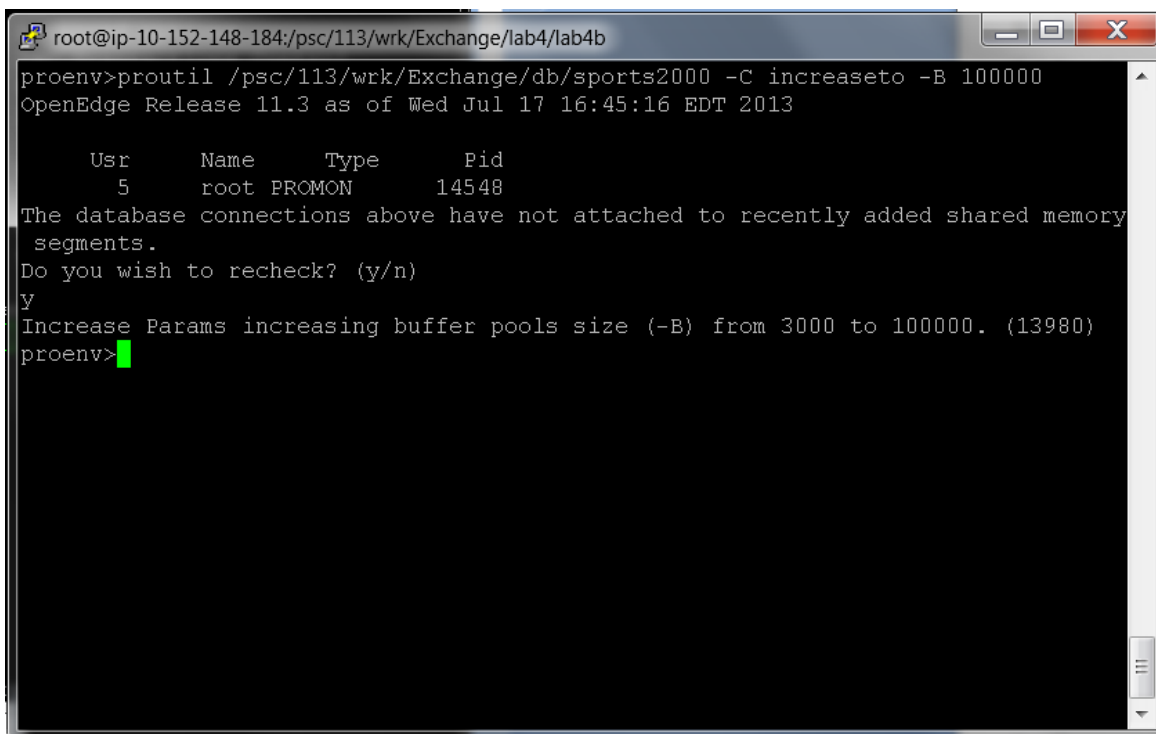
```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab4/lab4b
proenv>proutil /psc/113/wrk/Exchange/db/sports2000 -C increaseto -B 100000
OpenEdge Release 11.3 as of Wed Jul 17 16:45:16 EDT 2013

  Usr   Name   Type   Pid
   5    root  PROMON 14548

The database connections above have not attached to recently added shared memory
segments.
Do you wish to recheck? (y/n)
█
```

5) In the putty session running promon Enter X to exit promon.

6) In the putty session which is waiting to increase the -B enter Y to recheck.



```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab4/lab4b
proenv>proutil /psc/113/wrk/Exchange/db/sports2000 -C increaseto -B 100000
OpenEdge Release 11.3 as of Wed Jul 17 16:45:16 EDT 2013

  Usr   Name   Type   Pid
   5    root  PROMON 14548

The database connections above have not attached to recently added shared memory
segments.
Do you wish to recheck? (y/n)
y
Increase Params increasing buffer pools size (-B) from 3000 to 100000. (13980)
proenv>█
```

## Part (4c) Demonstration of increaseto -B2 (Disabling LRU2 Policy)

### Duration

5 Minutes

### Instruction

- 1) Open two putty sessions and issue the following commands

```
./proenv
```

```
cd lab4/lab4c
```

- 2) Issue the following command in the first putty session

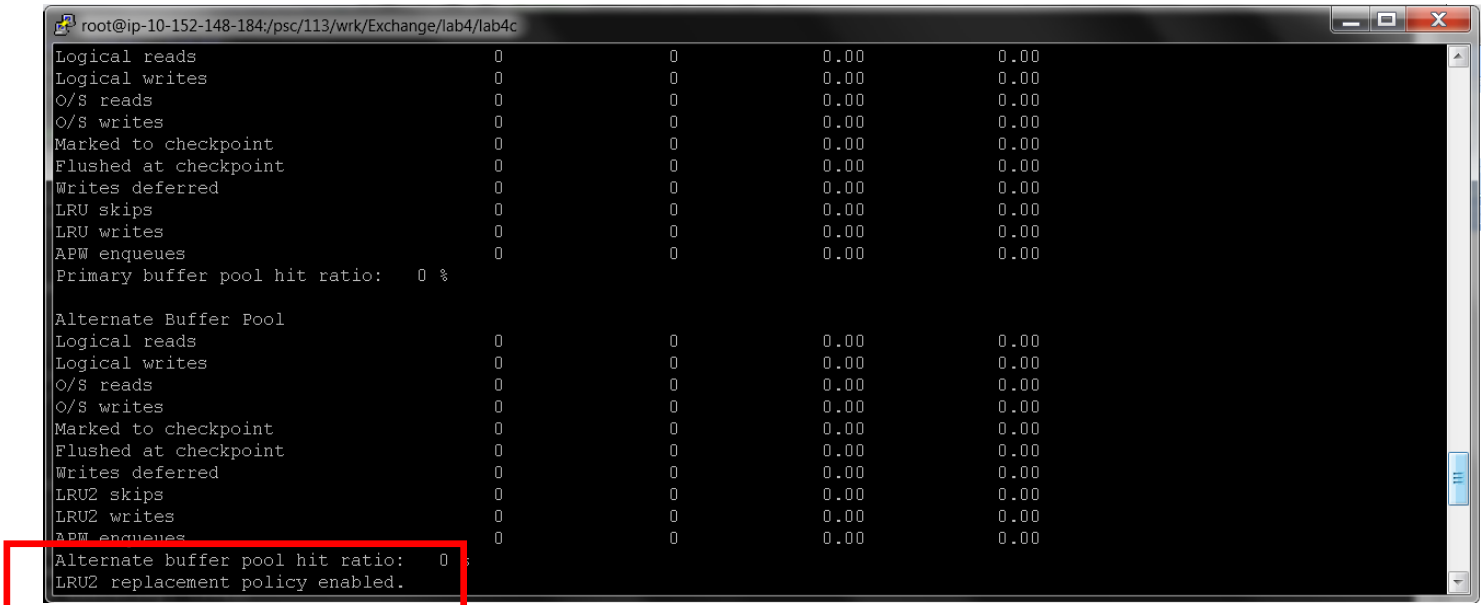
```
demonstrate-lru2-enabled.sh
```

Hit the return key once more so the program will return to the prompt when it is done.

- 3) In the second putty session issue this the following commands:

```
gatherlru.sh
```

Notice the LRU2 Replacement Policy is Enabled



```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab4/lab4c
Logical reads          0          0          0.00          0.00
Logical writes        0          0          0.00          0.00
O/S reads             0          0          0.00          0.00
O/S writes            0          0          0.00          0.00
Marked to checkpoint  0          0          0.00          0.00
Flushed at checkpoint 0          0          0.00          0.00
Writes deferred       0          0          0.00          0.00
LRU skips             0          0          0.00          0.00
LRU writes            0          0          0.00          0.00
APW enqueues         0          0          0.00          0.00
Primary buffer pool hit ratio:  0 %

Alternate Buffer Pool
Logical reads          0          0          0.00          0.00
Logical writes        0          0          0.00          0.00
O/S reads             0          0          0.00          0.00
O/S writes            0          0          0.00          0.00
Marked to checkpoint  0          0          0.00          0.00
Flushed at checkpoint 0          0          0.00          0.00
Writes deferred       0          0          0.00          0.00
LRU2 skips            0          0          0.00          0.00
LRU2 writes           0          0          0.00          0.00
APW enqueues         0          0          0.00          0.00
Alternate buffer pool hit ratio: 0 %
LRU2 replacement policy enabled.
```



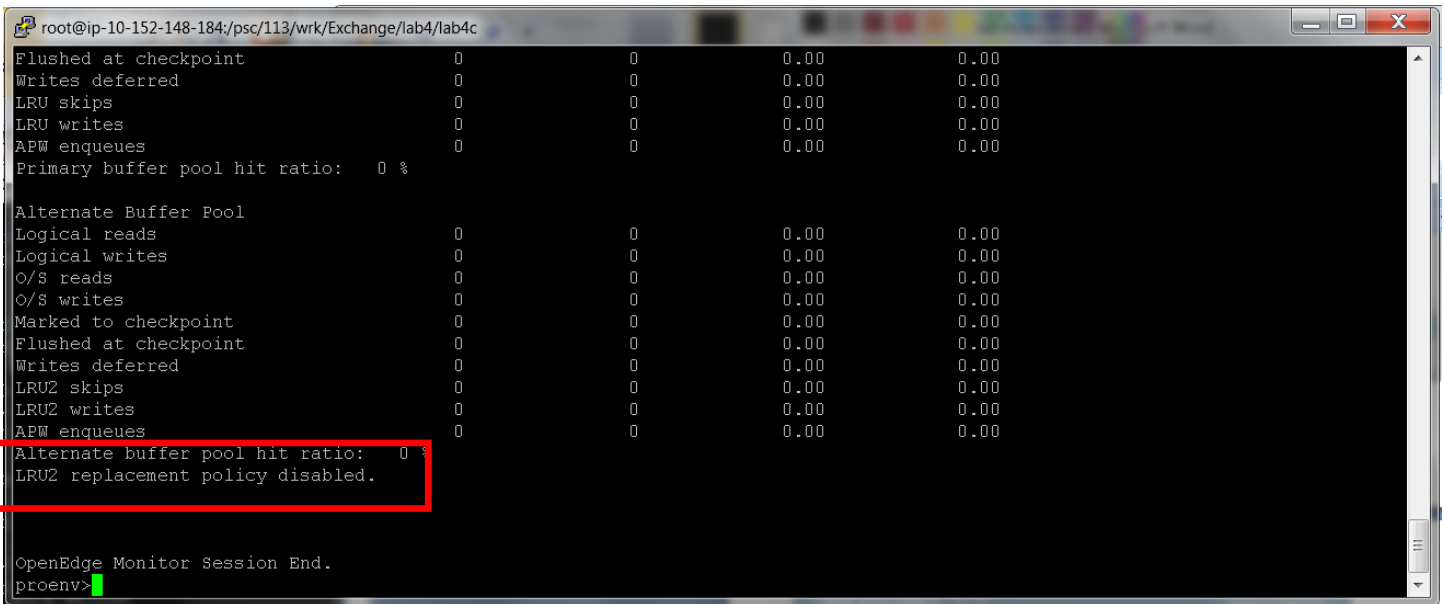
4) From the command prompt in the second putty session issue the command to increase the -B2

```
proutil /psc/113/wrk/Exchange/db/sports2000 -C increaseto -B2 100000
```

5) In the first putty session run this command again:

```
gatherlru.sh
```

Notice the LRU2 Replacement Policy is now Disabled

A terminal window screenshot showing the output of the 'gatherlru.sh' command. The window title is 'root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab4/lab4c'. The output displays various performance metrics for the primary and alternate buffer pools. The 'LRU2 replacement policy disabled.' line is highlighted with a red box. At the bottom, it says 'OpenEdge Monitor Session End.' and 'proenv>'.

```
root@ip-10-152-148-184:/psc/113/wrk/Exchange/lab4/lab4c
Flushed at checkpoint      0          0          0.00      0.00
Writes deferred           0          0          0.00      0.00
LRU skips                 0          0          0.00      0.00
LRU writes                0          0          0.00      0.00
APW enqueues              0          0          0.00      0.00
Primary buffer pool hit ratio:  0 %

Alternate Buffer Pool
Logical reads             0          0          0.00      0.00
Logical writes            0          0          0.00      0.00
O/S reads                 0          0          0.00      0.00
O/S writes                0          0          0.00      0.00
Marked to checkpoint      0          0          0.00      0.00
Flushed at checkpoint      0          0          0.00      0.00
Writes deferred           0          0          0.00      0.00
LRU2 skips                0          0          0.00      0.00
LRU2 writes               0          0          0.00      0.00
APW enqueues              0          0          0.00      0.00
Alternate buffer pool hit ratio: 0 %
LRU2 replacement policy disabled.

OpenEdge Monitor Session End.
proenv>
```