USE PACKAGED PROCEDURES TO ACHIEVE SUPERIOR PERFORMANCE

Mark W. Farnham, Rightsizing, Inc.

Summary:

Each individual sql statement submitted from a client (even sqlplus running directly on the database server) generates a pair of sqlnet waits. That is no big deal if you are submitting a single large set operation transaction and the sqlnet waits are a tiny fraction of the time of executing the statement. But if you are submitting several transactions and especially several small transactions such as the bits that make up an On-Line Transaction Processing (OLTP) transaction, it adds up, holding locks longer than needed and thus interfering with scaling against many transactors.

Abstract:

Many client and middle tier programs create transaction SQL on the fly. Except for value substitutions, constructed SQL is often repetitive and predictable. Replacing such constructions with calls to packaged procedures stored in the database can dramatically reduce the inbound traffic from the sql text to just a call name and any required parameters. For multi-statement transactions this also eliminates the handshakes between steps of the transaction and can greatly improve scalability by reducing the time locks are held. Further, opportunities for sql injection attacks are greatly reduced. This method is amenable to retrofitting existing applications as well as being an excellent design paradigm, regardless of your client language choice.

Remote submission of complete logical units of work and ensured multistage completion of transactions with business rules requiring an order of update different from the table and row order are additional techniques to improve concurrency, reduce locking delays, and further reduce transaction work that must be resubmitted after a deadlock is resolved. Employing these methods will reduce the possibility of application deadlocks to a few scenarios involving things like partially overlapping sets of rows or deferred constraints in support of a questionable schema design, while failing to employ a design method to avoid deadlocks can make the generation of deadlocks common and chronic. Because deadlocks do not routinely appear in unit testing, this principle is important to avoid surprises in production.

Beyond merely avoiding deadlocks, adding the discipline of describing an order of update across the enterprise will often have the side benefit of revealing duplications and previously unrecognized relationships amongst the applications and systems across an enterprise. Once put in place, a comprehensive corporate data dictionary framework used to keep track of the order of update can also be used to improve agility in deploying new systems. It can also be invaluable to analysis and business intelligence efforts.

Most Important for Small Transactions

As alluded to in the summary, the fraction of extra overhead due to sqlnet handshakes for large set transactions tends to be very small and likely not worth engineering out for purposes of improved elapsed time and concurrency. It is also quite unlikely to degrade elapsed time or concurrency, and any advantages gained by using server stored procedures with respect to security remain.

So while it may not be cost effective to re-engineer existing applications except where elapsed time and concurrency have become an issue or the chances of sql injection attach are high, I consider it a leading development method to adopt in general.

Still, since demonstrating that achieving superior performance may be gained via use of packaged procedures is the goal, the focus here will be on small, frequent transactions, including multiple statement logical units of work (LUWs).

PL/SQL has an excellent framework for instrumenting your Oracle code

As luck would have it Bob Rhubart has published "Top Ten 2 Minute Tech Tips by KScope 2015 Speakers" at https://blogs.oracle.com/archbeat/entry/top_ten_2_minute_tech in time for me to include a reference to my friend Cary Millsap entitled "Trace Your Code – All of It."

PL/SQL provides an excellent framework for instrumenting your Oracle code, either with the Oracle supplied packages or even better with the (all free) Instrumentation Library for Oracle (ILO) that can be found at sourceforge.net with a simple search. All the examples and hints that you're likely to need about slapping the calls for instrumentation into PL/SQL can be found in a slide deck authored by Karen Morton on 19 December 2008 for Method-r entitled "Performance Instrumentation for PL/SQL: When, Why, How(PPT)." This is also available for free download at method-r.com/papers. I highly recommend this for all your production code. I am not going to inflate my paper by re-wording any of this excellent advice. Listen to the 2 minutes tech tip interview and read the paper Karen wrote for Method-R if you have any questions. The last update I've seen to the Karen Morton paper I have seen is from 2008. At this writing I believe it remains state of the art. Even if technology drifts and some of the precise calls change, the principles of her advice are likely to remain valid.

Instrument you code so that getting proper scoping of collected metrics from trace files becomes a matter of fact instead of a matter of heuristic art.

"Trace your code – All of It." Good luck doing that from code generated on a client and passed in. It can be done, but again, good luck. If instead you pre-build the bulk of the assembled client side choices as procedures and then use the flexibility of the client user interface program to call the correct procedure with parameters having the instrumentation can be quite routine.

Even in the case of large existing code bases you can cherry pick the likely expensive things to package and instrument. If you have to diagnose something *even once* for performance in an existing largely un-instrumented code base put it back to bed in an instrumented form. You'll be very glad indeed the next time diagnosis is required. And very likely there will be a next time to examine once some piece of code has been indicted during a performance investigation. Even if you have repaired the root cause, the code will be on your radar. If it is now instrumented you can cheaply determine whether or not it is part of the problem or just a distraction of past ghosts.

So with regard to existing code that you have decided you cannot justify the expense and possible editing errors to completely instrument, I will offer a slight tweak: Instrument your code. All your new code, and any existing code the next time you have to change or investigate it.

Ironically I want to focus on a few simple procedures in my examples and I'm tracing them individually from end to end so I've judged that even the few lines of ILO would be a distraction from the points I'm making as well as an impediment to displaying the relevant code fragments and trace fragments on a presentation screen.

Notice in particular that the omission of ILO calls is a matter of scoping, not lack of tracing: I have determined in advance that the proper scope for tracing for my examples is the entire transaction. Even if less than the entire trace file is of interest it is a simple matter to snip the trace file to the scope of interest because I know in advance which trace file is of interest and the total bulk is manageable.

If I were to implement any of these things for any sort of production activity, the instrumentation would be there because I would not know (and likely *could not know*) in advance what the proper scoping would be when a performance issue eventuates.

The only excuse I know of for NOT instrumenting your PL/SQL code is so you can display the demo code on a slide in a font large enough to read.

Trivial comparison: Update and Commit One Row

First executed from SQL*Plus on the same host as the database server, second as a procedure call.

SQL*Plus script:

```
update inventory set onhand = onhand - &purchase_qty
where skucs = &skuc_s;
commit;
```

and the resulting trace output with the purchase_qty set to 100 and the skucs set to 1:

```
*** 2015-06-06 11:59:30.758
WAIT #3: nam='SQL*Net message from client' ela= 7858361 driver id=1111838976 #bytes=1 p3=0 obj#=116
tim=2049435525067
CLOSE #3:c=0,e=99,dep=0,type=0,tim=2049435525303
=====================
PARSING IN CURSOR #5 len=58 dep=0 uid=91 oct=6 lid=91 tim=2049435526367 hv=1676997574
ad='7ffbe300910' sqlid='1h56wm1jz9wy6'
update inventory set onhand = onhand - 100
where skucs = 1
END OF STMT
PARSE #5:c=0,e=1009,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=2762941418,tim=2049435526366
EXEC #5:c=0,e=146,p=0,cr=1,cu=3,mis=0,r=1,dep=0,og=1,plh=2762941418,tim=2049435526606
STAT #5 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #5 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0
us cost=0 size=6 card=1)'
WAIT #5: nam='SQL*Net message to client' ela= 4 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2049435526718
WAIT #5: nam='SQL*Net message from client' ela= 2388 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2049435529133
CLOSE #5:c=0,e=16,dep=0,type=0,tim=2049435529282
=====================
PARSING IN CURSOR #4 len=6 dep=0 uid=91 oct=44 lid=91 tim=2049435529331 hv=3480936638 ad='0'
sqlid='23wm3kz7rps5y'
commit
END OF STMT
PARSE #4:c=0,e=8,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,plh=0,tim=2049435529331
XCTEND rlbk=0, rd_only=0, tim=2049435529436
EXEC #4:c=0,e=2142,p=0,cr=0,cu=1,mis=0,r=0,dep=0,og=0,plh=0,tim=2049435531562
WAIT #4: nam='log file sync' ela= 206030 buffer#=12054 sync scn=105999008 p3=0 obj#=-1
tim=2049435737656
```

Notice that substitution variables arrive as literals. The text in yellow is what was transmitted. The SQL*Net message pair in the middle of the transaction are marked in red.

We could have defined variables and used a PL/SQL block to avoid this and the line turn-around. For example if the following is submitted from SQL*Plus:

```
variable purchase_qty number
variable skucs number
begin
:purchase_qty := 100;
:skucs := 1;
update inventory set onhand = onhand - :purchase_qty
   where skucs = :skucs;
commit;
end;
/
```

Then the trace file looks like this:

```
*** 2015-06-06 13:05:14.025
WAIT #3: nam='SQL*Net message from client' ela= 13981489 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2053378728408
CLOSE #3:c=0,e=13,dep=0,type=1,tim=2053378728646
=====================
```

```
PARSING IN CURSOR #2 len=128 dep=0 uid=91 oct=47 lid=91 tim=2053378728848 hv=4238991900 ad='7ffb5b6ed40'
sqlid='7t8134myamshw'
begin
:purchase_qty := 100;
:skucs := 1;
update inventory set onhand = onhand - :purchase_qty
where skucs = :skucs;
commit;
end;
END OF STMT
PARSE #2:c=0,e=148,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=2053378728847
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=1c2f2430  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1c2f2448  bln=22  avl=00  flg=01
=====================
PARSING IN CURSOR #4 len=60 dep=1 uid=91 oct=6 lid=91 tim=2053378729409 hv=3209938921 ad='7ffbee1d3e0'
sqlid='2zy5fhkzp7jz9'
UPDATE INVENTORY SET ONHAND = ONHAND - :B1 WHERE SKUCS = :B2
END OF STMT
PARSE #4:c=0,e=257,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=1,plh=0,tim=2053378729408
BINDS #4:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=202001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1c2f2430  bln=22  avl=02  flg=09
  value=100
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=202001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1c2f2448  bln=22  avl=02  flg=09
  value=1
EXEC #4:c=0,e=1225,p=0,cr=1,cu=3,mis=1,r=1,dep=1,og=1,plh=2762941418,tim=2053378730701
STAT #4 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #4 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0 us
cost=0 size=6 card=1)'
CLOSE #4:c=0,e=2,dep=1,type=3,tim=2053378730824
=====================
PARSING IN CURSOR #3 len=6 dep=1 uid=91 oct=44 lid=91 tim=2053378730860 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
PARSE #3:c=0,e=4,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=0,plh=0,tim=2053378730860
XCTEND rlbk=0, rd_only=0, tim=2053378730928
EXEC #3:c=0,e=138,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=2053378731051
CLOSE #3:c=0,e=2,dep=1,type=3,tim=2053378731107
WAIT #2: nam='SQL*Net message to client' ela= 4 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2053378731151
EXEC #2:c=0,e=2216,p=0,cr=1,cu=4,mis=0,r=1,dep=0,og=1,plh=0,tim=2053378731175
WAIT #2: nam='log file sync' ela= 79 buffer#=3519 sync scn=106002160 p3=0 obj#=-1 tim=2053378731280
```

The text in yellow is what was transmitted. Notice there is no SQL*Net message pair between the update and the commit, which I have marked in green. The purple calls out the parse of the update as well as the execution plan hash value (plh) when it is executed.

The first bit of purple shows the shorthand identification bits of the sql text: hash value, address, and sqlid. In the next bit of purple, the stats of the actual parse are presented. Notably e=257, mis=1, and plh=0, meaning respectively that the Elapsed time was about 257 microseconds, we MISsed once looking for this code in the existing library cache, and Oracle isn't telling us the plh yet. Notice that the plh does appear in purple in the exec line.

If the same block is passed up again, it likely finds an exact match in the library cache (mis=0) and so can re-use the plan and find it with the aid of the plh.

In fact I did toss the same code back up shortly later in a different session:

```
*** 2015-06-06 13:47:41.228
WAIT #2: nam='SQL*Net message from client' ela= 18199211 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2055925899987
CLOSE #2:c=0,e=12,dep=0,type=1,tim=2055925900131
=====================
PARSING IN CURSOR #3 len=128 dep=0 uid=91 oct=47 lid=91 tim=2055925900233 hv=4238991900 ad='7ffb5b6ed40'
sqlid='7t8134myamshw'
begin
:purchase_qty := 100;
:skucs := 1;
update inventory set onhand = onhand - :purchase_qty
where skucs = :skucs;
commit;
end;
END OF STMT
PARSE #3:c=0,e=61,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=2055925900231
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=1adb06a0  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1adb06b8  bln=22  avl=00  flg=01
=====================
PARSING IN CURSOR #2 len=60 dep=1 uid=91 oct=6 lid=91 tim=2055925900501 hv=3209938921 ad='7ffbee1d3e0'
sqlid='2zy5fhkzp7jz9'
UPDATE INVENTORY SET ONHAND = ONHAND - :B1 WHERE SKUCS = :B2
END OF STMT
PARSE #2:c=0,e=42,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=2762941418,tim=2055925900501
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=202001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1adb06a0  bln=22  avl=02  flg=09
  value=100
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=202001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1adb06b8  bln=22  avl=02  flg=09
  value=1
EXEC #2:c=15600,e=423,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=2762941418,tim=2055925900969
STAT #2 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #2 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0 us
cost=0 size=6 card=1)'
CLOSE #2:c=0,e=1,dep=1,type=3,tim=2055925901079
=====================
PARSING IN CURSOR #2 len=6 dep=1 uid=91 oct=44 lid=91 tim=2055925901105 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
PARSE #2:c=0,e=5,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=0,plh=0,tim=2055925901105
XCTEND rlbk=0, rd_only=0, tim=2055925901151
EXEC #2:c=0,e=113,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=2055925901249
CLOSE #2:c=0,e=1,dep=1,type=3,tim=2055925901293
```

So you can see the sql and plan are found and things happen more quickly because some of the work already leveraged is utilized. Still, the entire PL/SQL block has to be transmitted.

We can do better with a packaged procedure stored in the database. Of course with PL/SQL stored in the database we don't have to pay to ship the code for every call; this simple point is often ignored or discounted in a world of high bandwidth, low latency networks. This code is also intentionally bare bones to minimize the amount of code sent between the client and the database. For one off interactive experiments from SQL*Plus that is marginally okay, but if you're building production systems you will profit from error control, tracing, and the opportunity to improve scalability when there is no penalty for shipping the code and only a very marginal cost for increasing the code footprint.

We'll get to that in a bit, but first, what if we just package up the same bare bones code? Included in a package called *sell_pkg* is the procedure *minus_inventory_p1*. The relevant code snippet and the trace from sending exec sell_pkg.minus_inventory_p1 (1,1); from SQL*Plus follow:

```
procedure minus_inventory_p1(p_skucs number, p_purchase_qty number) is
   --
   begin
      update inventory i set i.onhand = i.onhand - p_purchase_qty
         where i.skucs = p_skucs;
      commit;
   --
end minus_inventory_p1;
```

```
*** 2015-06-10 02:01:12.187
WAIT #5: nam='SQL*Net message from client' ela= 467134148 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2359132558708
CLOSE #5:c=0,e=29,dep=0,type=1,tim=2359132558892
=====================
PARSING IN CURSOR #1 len=46 dep=0 uid=91 oct=47 lid=91 tim=2359132560458 hv=1859270037 ad='7ffbe698cd8'
sqlid='6585ph9rd4dcp'
BEGIN sell_pkg.minus_inventory_p1(1,1); END;
END OF STMT
PARSE #1:c=0,e=1522,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=2359132560457
=====================
PARSING IN CURSOR #4 len=69 dep=1 uid=91 oct=6 lid=91 tim=2359132560802 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
PARSE #4:c=0,e=157,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=1,plh=0,tim=2359132560802
BINDS #4:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1bf43a68  bln=22  avl=02  flg=09
  value=1
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1bf43a68  bln=22  avl=02  flg=09
  value=1
EXEC #4:c=0,e=933,p=0,cr=1,cu=3,mis=1,r=1,dep=1,og=1,plh=2762941418,tim=2359132561791
STAT #4 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #4 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0 us
cost=0 size=7 card=1)'
CLOSE #4:c=0,e=2,dep=1,type=3,tim=2359132561886
=====================
PARSING IN CURSOR #2 len=6 dep=1 uid=91 oct=44 lid=91 tim=2359132561925 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
PARSE #2:c=0,e=4,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=0,plh=0,tim=2359132561924
XCTEND rlbk=0, rd_only=0, tim=2359132561976
EXEC #2:c=0,e=137,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=2359132562101
CLOSE #2:c=0,e=0,dep=1,type=3,tim=2359132562149
EXEC #1:c=0,e=1655,p=0,cr=1,cu=4,mis=0,r=1,dep=0,og=1,plh=0,tim=2359132562178
WAIT #1: nam='log file sync' ela= 252 buffer#=10853 sync scn=106267234 p3=0 obj#=-1 tim=2359132562456
WAIT #1: nam='SQL*Net message to client' ela= 2 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2359132562768
```

And once again, to see the relevant times for a non-first execution:

```
   *** 2015-06-10 03:54:54.325
```

```
WAIT #2: nam='SQL*Net message from client' ela= 130628915 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2365954593931
CLOSE #2:c=0,e=11,dep=0,type=1,tim=2365954594111
=====================
PARSING IN CURSOR #3 len=47 dep=0 uid=91 oct=47 lid=91 tim=2365954595551 hv=3620327072
ad='7ffbea086b8' sqlid='3uut1f7bwmmp0'
BEGIN sell_pkg.minus_inventory_p1 (1,1); END;
END OF STMT
PARSE #3:c=0,e=1398,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=2365954595550
=====================
PARSING IN CURSOR #2 len=69 dep=1 uid=91 oct=6 lid=91 tim=2365954595782 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
PARSE #2:c=0,e=40,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=2762941418,tim=2365954595781
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da23a68  bln=22  avl=02  flg=09
  value=1
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da23a68  bln=22  avl=02  flg=09
  value=1
EXEC #2:c=0,e=222,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=2762941418,tim=2365954596059
STAT #2 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #2 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0
us cost=0 size=7 card=1)'
CLOSE #2:c=0,e=2,dep=1,type=3,tim=2365954596153
=====================
PARSING IN CURSOR #2 len=6 dep=1 uid=91 oct=44 lid=91 tim=2365954596180 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
PARSE #2:c=0,e=4,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=0,plh=0,tim=2365954596179
XCTEND rlbk=0, rd_only=0, tim=2365954596229
EXEC #2:c=0,e=119,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=2365954596336
CLOSE #2:c=0,e=0,dep=1,type=3,tim=2365954596383
EXEC #3:c=0,e=796,p=0,cr=1,cu=4,mis=0,r=1,dep=0,og=1,plh=0,tim=2365954596403
WAIT #3: nam='log file sync' ela= 404 buffer#=1868 sync scn=106271903 p3=0 obj#=-1 tim=2365954596832
```

Oops. I added a space after the procedure name before the open parentheses this time. Even so the statements *within* the PL/SQL routine are already parsed and planned, so the parse time is reduced dramatically.

Let's do that one more time:

```
T *** 2015-06-10 04:23:40.411
WAIT #2: nam='SQL*Net message from client' ela= 59946674 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2367680653722
CLOSE #2:c=0,e=12,dep=0,type=1,tim=2367680653925
=====================
PARSING IN CURSOR #3 len=47 dep=0 uid=91 oct=47 lid=91 tim=2367680654095 hv=3620327072
ad='7ffbea086b8' sqlid='3uut1f7bwmmp0'
BEGIN sell_pkg.minus_inventory_p1 (1,1); END;
END OF STMT
PARSE #3:c=0,e=126,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=2367680654094
=====================
PARSING IN CURSOR #2 len=69 dep=1 uid=91 oct=6 lid=91 tim=2367680654342 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
PARSE #2:c=0,e=23,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=2762941418,tim=2367680654342
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=0d393a68  bln=22  avl=02  flg=09
  value=1
```

```
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=0d393a68  bln=22  avl=02  flg=09
  value=1
EXEC #2:c=0,e=268,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=2762941418,tim=2367680654675
STAT #2 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #2 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0
us cost=0 size=7 card=1)'
CLOSE #2:c=0,e=1,dep=1,type=3,tim=2367680654783
=====================
PARSING IN CURSOR #2 len=6 dep=1 uid=91 oct=44 lid=91 tim=2367680654818 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
PARSE #2:c=0,e=10,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=0,plh=0,tim=2367680654818
XCTEND rlbk=0, rd_only=0, tim=2367680654877
EXEC #2:c=0,e=119,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=2367680654982
CLOSE #2:c=0,e=1,dep=1,type=3,tim=2367680655035
EXEC #3:c=0,e=904,p=0,cr=1,cu=4,mis=0,r=1,dep=0,og=1,plh=0,tim=2367680655058
WAIT #3: nam='log file sync' ela= 264 buffer#=6087 sync scn=106273429 p3=0 obj#=-1 tim=2367680655350
```

Now that call had *exactly* the same literal arguments as parameters. What if the exec statement has different arguments? This time, again from SQL*Plus, we'll send in exec sell_pkg.minus_inventory_p1 (1,2); and we get:

```
*** 2015-06-10 04:35:52.152
WAIT #2: nam='SQL*Net message from client' ela= 66882816 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2368412379136
CLOSE #2:c=0,e=12,dep=0,type=1,tim=2368412379289
=====================
PARSING IN CURSOR #3 len=47 dep=0 uid=91 oct=47 lid=91 tim=2368412380713 hv=1282889496
ad='7ffbeac6618' sqlid='2tgtg3j67fpss'
BEGIN sell_pkg.minus_inventory_p1 (1,2); END;
END OF STMT
PARSE #3:c=0,e=1385,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=2368412380712
=====================
PARSING IN CURSOR #2 len=69 dep=1 uid=91 oct=6 lid=91 tim=2368412380996 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
PARSE #2:c=0,e=27,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=2762941418,tim=2368412380995
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=b6a769d8  bln=22  avl=02  flg=09
  value=2
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=b6a769c0  bln=22  avl=02  flg=09
  value=1
EXEC #2:c=0,e=220,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=2762941418,tim=2368412381266
STAT #2 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #2 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0
us cost=0 size=7 card=1)'
CLOSE #2:c=0,e=2,dep=1,type=3,tim=2368412381357
=====================
PARSING IN CURSOR #2 len=6 dep=1 uid=91 oct=44 lid=91 tim=2368412381381 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
PARSE #2:c=0,e=4,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=0,plh=0,tim=2368412381380
XCTEND rlbk=0, rd_only=0, tim=2368412381426
EXEC #2:c=0,e=143,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=2368412381555
CLOSE #2:c=0,e=1,dep=1,type=3,tim=2368412381600
EXEC #3:c=0,e=846,p=0,cr=1,cu=4,mis=0,r=1,dep=0,og=1,plh=0,tim=2368412381618
WAIT #3: nam='log file sync' ela= 264 buffer#=6594 sync scn=106273883 p3=0 obj#=-1 tim=2368412381905
```

So that's no fun. We're still polluting the shared pool with different sql text to parse from the literal arguments, even though we do have the improvement that the bulk of the call from the body of the procedure uses binds. Let's see if we can fix that up as well:

```
variable purchase_qty number
variable skucs number

begin
:purchase_qty := &qty;
:skucs := &skucs;
sell_pkg.minus_inventory_p1(:skucs,:purchase_qty);
end;
```

Executed twice from SQL*Plus, once each with values of 1 and 3 and once with values of 1 and 5 yields:

```
*** 2015-06-10 05:11:48.731
WAIT #6: nam='SQL*Net message from client' ela= 51107927 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2370568926269
CLOSE #6:c=0,e=26,dep=0,type=0,tim=2370568926457
=====================
PARSING IN CURSOR #4 len=94 dep=0 uid=91 oct=47 lid=91 tim=2370568927053 hv=1280339793
ad='7ffbe776478' sqlid='5ctqpcj650vuj'
begin
:purchase_qty := 3;
:skucs := 1;
sell_pkg.minus_inventory_p1(:skucs,:purchase_qty);
end;
END OF STMT
PARSE #4:c=0,e=544,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=2370568927053
BINDS #4:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=0d2756e8  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=0d275700  bln=22  avl=00  flg=01
=====================
PARSING IN CURSOR #7 len=69 dep=1 uid=91 oct=6 lid=91 tim=2370568928461 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
PARSE #7:c=0,e=27,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=2762941418,tim=2370568928460
BINDS #7:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da23a68  bln=22  avl=02  flg=09
  value=3
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da23a98  bln=22  avl=02  flg=09
  value=1
EXEC #7:c=0,e=290,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=2762941418,tim=2370568928839
STAT #7 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #7 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0
us cost=0 size=7 card=1)'
CLOSE #7:c=0,e=1,dep=1,type=3,tim=2370568928956
=====================
PARSING IN CURSOR #5 len=6 dep=1 uid=91 oct=44 lid=91 tim=2370568928992 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
PARSE #5:c=0,e=4,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=0,plh=0,tim=2370568928992
XCTEND rlbk=0, rd_only=0, tim=2370568929078
```

```
EXEC #5:c=0,e=123,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=2370568929188
CLOSE #5:c=0,e=1,dep=1,type=3,tim=2370568929243
WAIT #4: nam='SQL*Net message to client' ela= 3 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2370568929274
EXEC #4:c=0,e=2107,p=0,cr=1,cu=4,mis=1,r=1,dep=0,og=1,plh=0,tim=2370568929300
WAIT #4: nam='log file sync' ela= 203 buffer#=10821 sync scn=106275571 p3=0 obj#=-1 tim=2370568929574

*** 2015-06-10 05:12:29.691
WAIT #4: nam='SQL*Net message from client' ela= 40958658 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2370609888395
CLOSE #4:c=0,e=48,dep=0,type=0,tim=2370609888610
=====================
PARSING IN CURSOR #2 len=94 dep=0 uid=91 oct=47 lid=91 tim=2370609889406 hv=1406895200
ad='7ffbe7044a0' sqlid='3dnrvvj9xr130'
begin
:purchase_qty := 5;
:skucs := 2;
sell_pkg.minus_inventory_p1(:skucs,:purchase_qty);
end;
END OF STMT
PARSE #2:c=0,e=747,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=2370609889405
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=0d2756e8  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=0d275700  bln=22  avl=00  flg=01
=====================
PARSING IN CURSOR #3 len=69 dep=1 uid=91 oct=6 lid=91 tim=2370609890626 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da23a68  bln=22  avl=02  flg=09
  value=5
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da23a98  bln=22  avl=02  flg=09
  value=2
EXEC #3:c=0,e=296,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=2762941418,tim=2370609890899
CLOSE #3:c=0,e=1,dep=1,type=3,tim=2370609890954
XCTEND rlbk=0, rd_only=0, tim=2370609890992
=====================
PARSING IN CURSOR #6 len=6 dep=1 uid=91 oct=44 lid=91 tim=2370609891139 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
EXEC #6:c=0,e=156,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=2370609891135
CLOSE #6:c=0,e=2,dep=1,type=3,tim=2370609891247
WAIT #2: nam='SQL*Net message to client' ela= 2 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2370609891281
EXEC #2:c=0,e=1786,p=0,cr=1,cu=4,mis=1,r=1,dep=0,og=1,plh=0,tim=2370609891305
WAIT #2: nam='log file sync' ela= 221 buffer#=10824 sync scn=106275588 p3=0 obj#=-1 tim=2370609891560
```

So while the sql statements within the procedure are in cache, even a change so small as different arguments to bind cause a parsing of the initial block.

As far as I know, if you ship the code from SQL*Plus, even calls to procedures with bind variables, the initial block will get a miss if either the punctuation or the argument values themselves to be bound have changed. There are two more methods to *attempt to* handle this from SQL*Plus (each with its own drawbacks).

The first method is to submit separate blocks to the server (the first with just the bind variable assignments, the second with just the procedure or function call (using the bind variables that have been assigned.) The advantage of this is that the block with the procedure call will be recognized and won't have to be re-parsed. The disadvantage is that the bind variable assignments will have to be reparsed each time they are not identical to something seen *recently enough before* to remain in the library.

The second method is to just go ahead and use SQL*Plus substitutions for the function call and rely on setting cursor_sharing. The advantage is with cursor_sharing set you will likely re-use the cursor and not require separate parsing of the bind values. The disadvantage is having cursor sharing on and losing the likely more understandable error reporting if the variable assignments are explicitly parsed. (As well as the commonly known disadvantage of cursor sharing that it may suppress the generation of *desired* alternate plans for different literal arguments.)

Here is an example of the first method:

```
From SQL*Plus with event 10046 level 28:

begin
:purchase_qty := 42;
:skucs := 1;
end;
/

exec sell_pkg.minus_inventory_p1(:skucs,:purchase_qty);

begin
:purchase_qty := 54;
:skucs := 1;
end;
/

exec sell_pkg.minus_inventory_p1(:skucs,:purchase_qty);

Yields this trace (snipped to relevant scope):

*** 2015-06-12 15:21:34.410
WAIT #2: nam='SQL*Net message from client' ela= 30577870 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2579940373975
CLOSE #2:c=0,e=12,dep=0,type=1,tim=2579940374157
=====================
PARSING IN CURSOR #3 len=44 dep=0 uid=91 oct=47 lid=91 tim=2579940374686 hv=118800706
ad='7ffbe21f320' sqlid='6w6tg383j9ha2'
begin
:purchase_qty := 42;
:skucs := 1;
end;
END OF STMT
PARSE #3:c=0,e=487,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=2579940374685
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=1ef905e0  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1ef905f8  bln=22  avl=00  flg=01
WAIT #3: nam='SQL*Net message to client' ela= 2 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2579940375714
EXEC #3:c=0,e=979,p=0,cr=0,cu=0,mis=1,r=1,dep=0,og=1,plh=0,tim=2579940375735

*** 2015-06-12 15:21:55.310
WAIT #3: nam='SQL*Net message from client' ela= 20895793 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2579961271586
CLOSE #3:c=0,e=52,dep=0,type=0,tim=2579961271817
=====================
```

```
PARSING IN CURSOR #2 len=63 dep=0 uid=91 oct=47 lid=91 tim=2579961272175 hv=3736181483
ad='7ffbeea9bd0' sqlid='fjx4svggb36rb'
BEGIN sell_pkg.minus_inventory_p1(:skucs,:purchase_qty); END;
END OF STMT
PARSE #2:c=0,e=303,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=2579961272174
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=1ef905e0  bln=22  avl=02  flg=05
  value=1
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1ef905f8  bln=22  avl=02  flg=01
  value=42
=====================
PARSING IN CURSOR #3 len=69 dep=1 uid=91 oct=6 lid=91 tim=2579961272597 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
PARSE #3:c=0,e=26,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=2762941418,tim=2579961272596
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da63a98  bln=22  avl=02  flg=09
  value=42
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da63a68  bln=22  avl=02  flg=09
  value=1
EXEC #3:c=0,e=320,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=2762941418,tim=2579961273002
STAT #3 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #3 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0
us cost=0 size=7 card=1)'
CLOSE #3:c=0,e=2,dep=1,type=3,tim=2579961273137
=====================
PARSING IN CURSOR #3 len=6 dep=1 uid=91 oct=44 lid=91 tim=2579961273180 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
PARSE #3:c=0,e=7,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=0,plh=0,tim=2579961273180
XCTEND rlbk=0, rd_only=0, tim=2579961273260
EXEC #3:c=0,e=194,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=2579961273438
CLOSE #3:c=0,e=1,dep=1,type=3,tim=2579961273510
WAIT #2: nam='SQL*Net message to client' ela= 24 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2579961273564
EXEC #2:c=0,e=1335,p=0,cr=1,cu=4,mis=0,r=1,dep=0,og=1,plh=0,tim=2579961273602
WAIT #2: nam='log file sync' ela= 128 buffer#=3344 sync scn=106443797 p3=0 obj#=-1 tim=2579961273782

*** 2015-06-12 15:26:23.700
WAIT #2: nam='SQL*Net message from client' ela= 268382597 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2580229656490
CLOSE #2:c=0,e=49,dep=0,type=0,tim=2580229656725
=====================
PARSING IN CURSOR #3 len=44 dep=0 uid=91 oct=47 lid=91 tim=2580229657325 hv=1563600720
ad='7ffb5f62280' sqlid='8dbnw9jfm59uh'
begin
:purchase_qty := 54;
:skucs := 1;
end;
END OF STMT
PARSE #3:c=0,e=542,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=2580229657324
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=1ef909b0  bln=22  avl=00  flg=05
 Bind#1
```

```
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1ef909c8  bln=22  avl=00  flg=01
WAIT #3: nam='SQL*Net message to client' ela= 4 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2580229658836
EXEC #3:c=0,e=1421,p=0,cr=0,cu=0,mis=1,r=1,dep=0,og=1,plh=0,tim=2580229658870

*** 2015-06-12 15:26:36.070
WAIT #3: nam='SQL*Net message from client' ela= 12367948 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2580242026902
CLOSE #3:c=0,e=47,dep=0,type=0,tim=2580242027089
=====================
PARSING IN CURSOR #2 len=63 dep=0 uid=91 oct=47 lid=91 tim=2580242027239 hv=3736181483
ad='7ffbeea9bd0' sqlid='fjx4svggb36rb'
BEGIN sell_pkg.minus_inventory_p1(:skucs,:purchase_qty); END;
END OF STMT
PARSE #2:c=0,e=68,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=2580242027237
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=48 off=0
  kxsbbbfp=1ef909b0  bln=22  avl=02  flg=05
  value=1
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1ef909c8  bln=22  avl=02  flg=01
  value=54
=====================
PARSING IN CURSOR #3 len=69 dep=1 uid=91 oct=6 lid=91 tim=2580242027582 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da63a98  bln=22  avl=02  flg=09
  value=54
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1da63a68  bln=22  avl=02  flg=09
  value=1
EXEC #3:c=0,e=396,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=2762941418,tim=2580242027947
CLOSE #3:c=0,e=2,dep=1,type=3,tim=2580242028018
XCTEND rlbk=0, rd_only=0, tim=2580242028058
=====================
PARSING IN CURSOR #3 len=6 dep=1 uid=91 oct=44 lid=91 tim=2580242028171 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
EXEC #3:c=0,e=126,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=2580242028170
CLOSE #3:c=0,e=1,dep=1,type=3,tim=2580242028259
WAIT #2: nam='SQL*Net message to client' ela= 2 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=2580242028282
EXEC #2:c=0,e=985,p=0,cr=1,cu=4,mis=0,r=1,dep=0,og=1,plh=0,tim=2580242028300
WAIT #2: nam='log file sync' ela= 195 buffer#=3507 sync scn=106443975 p3=0 obj#=-1 tim=2580242028525
```

If the routines have not been run before in this instance or have cycled out, the first parses of this pair might be more expensive, but the point is that except for the bind substitutions and getting a cache hit the parsing and plan creation are already done. The SQL*net wait pairs have returned, but only once between the value setting block and the procedure call, *not* between each sql statement. This may become important when there are several statements in an LUW.

Here is an *attempt* of the second method:

```
accept skucs prompt 'What stock number color size?'
accept purchase_qty prompt 'How many do you want?'

exec sell_pkg.minus_inventory_p1(&skucs,&purchase_qty);
```

will be issued twice, with different arguments after

```
alter session set cursor_sharing=force
```

has been set for the session.

```
*** 2015-06-19 07:47:36.240
WAIT #3: nam='SQL*Net message from client' ela= 15760129 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=3157482673155
CLOSE #3:c=0,e=26,dep=0,type=1,tim=3157482673331
=====================
PARSING IN CURSOR #2 len=47 dep=0 uid=91 oct=47 lid=91 tim=3157482674712 hv=335044358
ad='7ffb5fdbff8' sqlid='b9rmr7n9zhrs6'
BEGIN sell_pkg.minus_inventory_p1(1,42); END;
END OF STMT
PARSE #2:c=15600,e=1340,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=3157482674710
=====================
PARSING IN CURSOR #4 len=69 dep=1 uid=91 oct=6 lid=91 tim=3157482676191 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
PARSE #4:c=0,e=176,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=1,plh=0,tim=3157482676191
=====================
BINDS #4:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=b6d85f88  bln=22  avl=02  flg=09
  value=42
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=b6d85f70  bln=22  avl=02  flg=09
  value=1
EXEC #4:c=15600,e=33319,p=0,cr=195,cu=3,mis=1,r=1,dep=1,og=1,plh=2762941418,tim=3157482709560
STAT #4 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0 time=0 us)'
STAT #4 id=2 cnt=1 pid=1 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK (cr=1 pr=0 pw=0 time=0
us cost=0 size=8 card=1)'
CLOSE #4:c=0,e=0,dep=1,type=3,tim=3157482709679
=====================
PARSING IN CURSOR #3 len=6 dep=1 uid=91 oct=44 lid=91 tim=3157482709734 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
PARSE #3:c=0,e=6,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=0,plh=0,tim=3157482709733
XCTEND rlbk=0, rd_only=0, tim=3157482709776
EXEC #3:c=0,e=157,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=3157482709922
CLOSE #3:c=0,e=0,dep=1,type=3,tim=3157482709962
EXEC #2:c=15600,e=35196,p=0,cr=195,cu=4,mis=0,r=1,dep=0,og=1,plh=0,tim=3157482710000
WAIT #2: nam='log file sync' ela= 233 buffer#=5893 sync scn=106952662 p3=0 obj#=-1 tim=3157482710262
WAIT #2: nam='SQL*Net message to client' ela= 2 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=3157482710378

*** 2015-06-19 07:47:45.930
WAIT #2: nam='SQL*Net message from client' ela= 9648298 driver id=1111838976 #bytes=1 p3=0 obj#=-1
tim=3157492358704
CLOSE #2:c=0,e=53,dep=0,type=0,tim=3157492358919
=====================
PARSING IN CURSOR #5 len=47 dep=0 uid=91 oct=47 lid=91 tim=3157492361003 hv=774554029
ad='7ffc9240ba0' sqlid='ayx304hr2phdd'
BEGIN sell_pkg.minus_inventory_p1(1,54); END;
END OF STMT
```

```
PARSE #5:c=0,e=2022,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1,plh=0,tim=3157492361001
=====================
PARSING IN CURSOR #6 len=69 dep=1 uid=91 oct=6 lid=91 tim=3157492361321 hv=391566703 ad='7ffbe533c20'
sqlid='52cqs6wbpdpbg'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.SKUCS = :B1
END OF STMT
BINDS #6:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=c0f5f398  bln=22  avl=02  flg=09
  value=54
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=c0f5f380  bln=22  avl=02  flg=09
  value=1
EXEC #6:c=0,e=355,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=2762941418,tim=3157492361646
CLOSE #6:c=0,e=2,dep=1,type=3,tim=3157492361726
XCTEND rlbk=0, rd_only=0, tim=3157492361774
=====================
PARSING IN CURSOR #4 len=6 dep=1 uid=91 oct=44 lid=91 tim=3157492361910 hv=255718823 ad='0'
sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
EXEC #4:c=0,e=150,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=3157492361909
CLOSE #4:c=0,e=1,dep=1,type=3,tim=3157492362030
EXEC #5:c=0,e=886,p=0,cr=1,cu=4,mis=0,r=1,dep=0,og=1,plh=0,tim=3157492362056
WAIT #5: nam='log file sync' ela= 213 buffer#=5895 sync scn=106952666 p3=0 obj#=-1 tim=3157492362308
```

So we see that cursor sharing does *not* apply to parsing block execution directives. Indeed if we look soon enough we'll find these strings in the v$sqlarea:

SQL_TEXT	SQL_ID	PARSE_CALLS
BEGIN sell_pkg.minus_inventory_p1(1,42); END;	b9rmr7n9zhrs6	1
BEGIN sell_pkg.minus_inventory_p1(1,54); END;	ayx304hr2phdd	1

So neither of the proposed methods completely eliminates parsing of transactions submitted from SQL*Plus. This is symptomatic of the fact that SQL*Plus is *not* primarily designed as an efficient transaction processing engine, but rather as a tool for convenience. Please do not lose sight of the fact that the first method, by de-coupling the variable binding from the procedure call parsing, both minimizes the complications of parsing when submitted from SQL*Plus *and* models better to the creation of processing services which may natively assign values to be bound while perhaps holding cursors open for the various bits of a LUW. Again, looking into the v$sqlarea, we then find

SQL_TEXT	SQL_ID	PARSE_CALLS
begin :purchase_qty := 43; :skucs := 1; end;	1z551zdtwc463	1
begin :purchase_qty := 55; :skucs := 1; end;	4fguk7453yfap	1
BEGIN sell_pkg.minus_inventory_p1(:skucs,:purchase_qty); END;	fjx4svggb36rb	1

where we can see the bind variable assignments sent from SQL*Plus each time, but our packaged procedure invocation exactly once with a single parse for multiple calls.

So even though the first method does not eliminate all parsing from SQL*Plus and does involve a single avoidable pair of SQL*Net waits in submitting the work (although notably *not* during the window when the transaction is in flight), method one is my recommendation for prototyping transactions from SQL*Plus. For full LUW examinations that is all I will show.

Less Trivial comparison: Several Steps in a LUW

Separate files u_inventory_04_pkg.sql and u_inventory_04_pkg_v01.spo, which show the definition and compilation of the package used for both the simple inventory update and the logical unit of work to sell something. In line here are the sqlplus execution and the tracefile so you can read the details of just how efficient procedures and functions stored in the database can be. I suggest that a reader less than expert at reading trace files opens this file in an editor and makes color highlighting and commentary similar to that above as a learning exercise. One thing you may notice is that I use little scripts for just about everything of any substantial length. I won't be typing "ALTER session SET EVENTS '10046 trace name context forever, level 12';" when I can type "@t10046-12" and I suggest you think about this as a standard naming convention for any traces you'll ever plan to use more than once. Even with a lot of variety you don't need a commented catalog to know what t10046-28 or t10046-0 will do.

Here is the execution:

```
SQL*Plus: Release 11.2.0.1.0 Production on Thu Nov 12 23:23:00 2015
Copyright (c) 1982, 2010, Oracle.  All rights reserved.
Enter user-name: mwf/redacted
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> set echo on;
SQL> @q_my_tracefile
SQL> select s.inst_id,s.sid,s.serial#,p.pid, p.tracefile
  2  from gv$process p,
  3  (select inst_id, sid, serial#, paddr
  4     from gv$session
  5     where sid in (select sid from v$mystat where rownum < 2)
  6  ) s
  7  where p.addr = s.paddr
  8  /
   INST_ID       SID    SERIAL#       PID
---------- ---------- ---------- ----------
TRACEFILE
--------------------------------------------------------------------------------------------------------------------------------
         1        130      11140         22
c:\ora\diag\rdbms\rsiz\rsiz\trace\rsiz_ora_7428.trc

SQL> select * from inventory;
     SKUCS     ONHAND
---------- ----------
         1     998046
         2     999887

SQL> select * from shipping;
   SHIP_ID      SKUCS   QUANTITY NAME
---------- ---------- ---------- --------------------------------------------------------------------------------------------------
SHIPTO_ADDR1
--------------------------------------------------------------------------------
SHIPTO_ADDR2
--------------------------------------------------------------------------------
SHIPTO_CITY
--------------------------------------------------------------------------------
SHIPTO_STATE
--------------------------------------------------------------------------------
SHIPTO_POSTAL
```

```
-------------------------------------------------------------------------------------------------
    10001          2         54 Mark W. Farnham
4213 Mockingbird Lane
near the graveyard
Lebanon
NH
03766-1239


SQL> @t10046-12
SQL> ALTER session SET EVENTS '10046 trace name context forever, level 12';

Session altered.

SQL> @u_inventory_04_exec_02
SQL> variable purchase_qty number
SQL> variable skucs number
SQL> variable custid number
SQL> variable status number
SQL>
SQL> begin
  2  :purchase_qty := &qty;
  3  :skucs := &skucs;
  4  :custid := &custid;
  5  end;
  6  /
Enter value for qty: 54
old   2: :purchase_qty := &qty;
new   2: :purchase_qty := 54;
Enter value for skucs: 2
old   3: :skucs := &skucs;
new   3: :skucs := 2;
Enter value for custid: 1
old   4: :custid := &custid;
new   4: :custid := 1;

PL/SQL procedure successfully completed.

SQL>
SQL> exec :status := sell_pkg.sell_and_ship2(:custid,:skucs,:purchase_qty);

PL/SQL procedure successfully completed.

SQL>
SQL> @t10046-0
SQL> alter session set events '10046 trace name context forever, level 0';

Session altered.

SQL> print status

   STATUS
----------
        0

SQL> select * from inventory;
```

```
    SKUCS     ONHAND
---------- ----------
         1     998046
         2     999833

SQL> select * from shipping;

  SHIP_ID     SKUCS   QUANTITY NAME
---------- ---------- ---------- ----------------------------------------------------------------------------------------------------
SHIPTO_ADDR1
--------------------------------------------------------------------------------------------------------------------------------
SHIPTO_ADDR2
--------------------------------------------------------------------------------------------------------------------------------
SHIPTO_CITY
--------------------------------------------------------------------------------------------------------------------------------
SHIPTO_STATE
--------------------------------------------------------------------------------------------------------------------------------
SHIPTO_POSTAL
--------------------------------------------------------------------------------------------------------------------------------
    10001         2         54 Mark W. Farnham
4213 Mockingbird Lane
near the graveyard
Lebanon
NH
03766-1239

    10002         2         54 Mark W. Farnham
4213 Mockingbird Lane
near the graveyard
Lebanon
NH
03766-1239


SQL>
```

And here is the tracefile: c:\ora\diag\rdbms\rsiz\rsiz\trace\rsiz_ora_7428.trc

```
Trace file c:\ora\diag\rdbms\rsiz\rsiz\trace\rsiz_ora_7428.trc
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Windows NT Version V6.1 Service Pack 1
CPU                 : 4 - type 8664, 2 Physical Cores
Process Affinity    : 0x0x0000000000000000
Memory (Avail/Total): Ph:1620M/8181M, Ph+PgF:17036M/24564M
Instance name: rsiz
Redo thread mounted by this instance: 1
Oracle process number: 22
Windows thread id: 7428, image: ORACLE.EXE (SHAD)


*** 2015-11-12 23:26:27.538
*** SESSION ID:(130.11140) 2015-11-12 23:26:27.538
```

```
*** CLIENT ID:() 2015-11-12 23:26:27.538
*** SERVICE NAME:(SYS$USERS) 2015-11-12 23:26:27.538
*** MODULE NAME:(SQL*Plus) 2015-11-12 23:26:27.538
*** ACTION NAME:() 2015-11-12 23:26:27.538

WAIT #2: nam='SQL*Net message to client' ela= 4 driver id=1111838976 #bytes=1
p3=0 obj#=-1 tim=810733007172

*** 2015-11-12 23:27:18.559
WAIT #2: nam='SQL*Net message from client' ela= 51018930 driver id=1111838976
#bytes=1 p3=0 obj#=-1 tim=810784027698
CLOSE #2:c=0,e=13,dep=0,type=1,tim=810784027897
=====================
PARSING IN CURSOR #3 len=58 dep=0 uid=91 oct=47 lid=91 tim=810784028111
hv=356505772 ad='7ffc98f35a0' sqlid='953w0nsamzq5c'
begin
:purchase_qty := 54;
:skucs := 2;
:custid := 1;
end;
END OF STMT
PARSE #3:c=0,e=160,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=810784028109
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=72 off=0
  kxsbbbfp=1c93f840  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1c93f858  bln=22  avl=00  flg=01
 Bind#2
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=48
  kxsbbbfp=1c93f870  bln=22  avl=00  flg=01
WAIT #3: nam='SQL*Net message to client' ela= 5 driver id=1111838976 #bytes=1
p3=0 obj#=-1 tim=810784028436
EXEC #3:c=0,e=235,p=0,cr=0,cu=0,mis=0,r=1,dep=0,og=1,plh=0,tim=810784028465
WAIT #3: nam='SQL*Net message from client' ela= 4839 driver id=1111838976
#bytes=1 p3=0 obj#=-1 tim=810784033374
CLOSE #3:c=0,e=33,dep=0,type=0,tim=810784033516
=====================
PARSING IN CURSOR #2 len=78 dep=0 uid=91 oct=47 lid=91 tim=810784033678
hv=230108677 ad='7ffb3e5eaf8' sqlid='782bwm86vfbh5'
BEGIN :status := sell_pkg.sell_and_ship2(:custid,:skucs,:purchase_qty); END;
END OF STMT
PARSE #2:c=0,e=117,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=0,tim=810784033677
BINDS #2:
 Bind#0
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=96 off=0
  kxsbbbfp=1c93f828  bln=22  avl=00  flg=05
 Bind#1
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
```

```
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=24
  kxsbbbfp=1c93f840  bln=22  avl=02  flg=01
  value=1
 Bind#2
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=48
  kxsbbbfp=1c93f858  bln=22  avl=02  flg=01
  value=2
 Bind#3
  oacdty=02 mxl=22(22) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1000000 frm=00 csi=00 siz=0 off=72
  kxsbbbfp=1c93f870  bln=22  avl=02  flg=01
  value=54
=====================
PARSING IN CURSOR #3 len=52 dep=1 uid=91 oct=3 lid=91 tim=810784034219
hv=1354294563 ad='7ffbed312d8' sqlid='dmf0kkj8bjt93'
SELECT I.ROWID FROM INVENTORY I WHERE I.SKUCS = :B1
END OF STMT
PARSE
#3:c=0,e=39,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=1415135660,tim=810784034219
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1c938460  bln=22  avl=02  flg=05
  value=2
EXEC
#3:c=0,e=102,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=1415135660,tim=810784034398
FETCH
#3:c=0,e=47,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=1,plh=1415135660,tim=810784034485
STAT #3 id=1 cnt=1 pid=0 pos=1 obj=178147 op='INDEX UNIQUE SCAN INVENTORY_PK
(cr=1 pr=0 pw=0 time=0 us cost=0 size=8 card=1)'
CLOSE #3:c=0,e=2,dep=1,type=3,tim=810784034586
=====================
PARSING IN CURSOR #3 len=53 dep=1 uid=91 oct=3 lid=91 tim=810784034708
hv=759153117 ad='7ffb3f62268' sqlid='gywyzunqmzhfx'
SELECT I.ONHAND FROM INVENTORY I WHERE I.ROWID = :B1
END OF STMT
PARSE
#3:c=0,e=21,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=4009304371,tim=810784034707
BINDS #3:
 Bind#0
  oacdty=01 mxl=32(18) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=01 csi=873 siz=32 off=0
  kxsbbbfp=1c8e3ce0  bln=32  avl=18  flg=09
  value="AAArfiAAGAAAAkBAAB"
EXEC
#3:c=0,e=85,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=4009304371,tim=810784034948
FETCH
#3:c=0,e=23,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=1,plh=4009304371,tim=810784035004
STAT #3 id=1 cnt=1 pid=0 pos=1 obj=178146 op='TABLE ACCESS BY USER ROWID
INVENTORY (cr=1 pr=0 pw=0 time=0 us cost=1 size=8 card=1)'
CLOSE #3:c=0,e=1,dep=1,type=3,tim=810784035078
=====================
```

```
PARSING IN CURSOR #3 len=132 dep=1 uid=91 oct=3 lid=91 tim=810784035152
hv=3497336310 ad='7ffbef25630' sqlid='amv9xsb87a7gq'
SELECT C.NAME, C.SHIPTO_ADDR1, C.SHIPTO_ADDR2, C.SHIPTO_CITY, C.SHIPTO_STATE,
C.SHIPTO_POSTAL FROM CUSTOMER C WHERE C.CUST_ID = :B1
END OF STMT
PARSE
#3:c=0,e=37,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=3389176309,tim=810784035151
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1c936800  bln=22  avl=02  flg=05
  value=1
EXEC
#3:c=0,e=86,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=3389176309,tim=810784035310
FETCH
#3:c=0,e=23,p=0,cr=2,cu=0,mis=0,r=1,dep=1,og=1,plh=3389176309,tim=810784035370
STAT #3 id=1 cnt=1 pid=0 pos=1 obj=178148 op='TABLE ACCESS BY INDEX ROWID
CUSTOMER (cr=2 pr=0 pw=0 time=0 us cost=1 size=82 card=1)'
STAT #3 id=2 cnt=1 pid=1 pos=1 obj=178149 op='INDEX UNIQUE SCAN CUSTOMER_PK
(cr=1 pr=0 pw=0 time=0 us cost=0 size=0 card=1)'
CLOSE #3:c=0,e=1,dep=1,type=3,tim=810784035463
=====================
PARSING IN CURSOR #3 len=36 dep=1 uid=91 oct=3 lid=91 tim=810784035546
hv=7117464 ad='7ffbef25420' sqlid='8cums1w06t6ns'
Select SHIPPING_ID.NEXTVAL from dual
END OF STMT
PARSE
#3:c=0,e=48,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=3236968432,tim=810784035545
EXEC
#3:c=0,e=31,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=3236968432,tim=810784035641
FETCH
#3:c=0,e=28,p=0,cr=0,cu=0,mis=0,r=1,dep=1,og=1,plh=3236968432,tim=810784035704
STAT #3 id=1 cnt=1 pid=0 pos=1 obj=178154 op='SEQUENCE  SHIPPING_ID (cr=0 pr=0
pw=0 time=0 us)'
STAT #3 id=2 cnt=1 pid=1 pos=1 obj=0 op='FAST DUAL  (cr=0 pr=0 pw=0 time=0 us
cost=2 size=0 card=1)'
CLOSE #3:c=0,e=1,dep=1,type=3,tim=810784035788
=====================
PARSING IN CURSOR #3 len=53 dep=1 uid=91 oct=3 lid=91 tim=810784035845
hv=759153117 ad='7ffb3f62268' sqlid='gywyzunqmzhfx'
SELECT I.ONHAND FROM INVENTORY I WHERE I.ROWID = :B1
END OF STMT
PARSE
#3:c=0,e=17,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=4009304371,tim=810784035844
BINDS #3:
 Bind#0
  oacdty=01 mxl=32(18) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=01 csi=873 siz=32 off=0
  kxsbbbfp=1c8e3ce0  bln=32  avl=18  flg=09
  value="AAArfiAAGAAAAkBAAB"
EXEC
#3:c=0,e=100,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=4009304371,tim=810784036051
```

```
FETCH
#3:c=0,e=15,p=0,cr=1,cu=0,mis=0,r=1,dep=1,og=1,plh=4009304371,tim=810784036102
STAT #3 id=1 cnt=1 pid=0 pos=1 obj=178146 op='TABLE ACCESS BY USER ROWID
INVENTORY (cr=1 pr=0 pw=0 time=0 us cost=1 size=8 card=1)'
CLOSE #3:c=0,e=2,dep=1,type=3,tim=810784036189
=====================
PARSING IN CURSOR #3 len=69 dep=1 uid=91 oct=6 lid=91 tim=810784036300
hv=732238760 ad='7ffbe7f02f8' sqlid='fu3n8rspua4x8'
UPDATE INVENTORY I SET I.ONHAND = I.ONHAND - :B2 WHERE I.ROWID = :B1
END OF STMT
PARSE
#3:c=0,e=41,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=1226990995,tim=810784036299
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1c8e3b38  bln=22  avl=02  flg=09
  value=54
 Bind#1
  oacdty=01 mxl=32(18) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=01 csi=873 siz=32 off=0
  kxsbbbfp=1c8e3ce0  bln=32  avl=18  flg=09
  value="AAArfiAAGAAAAkBAAB"
EXEC
#3:c=0,e=344,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=1226990995,tim=810784036722
STAT #3 id=1 cnt=0 pid=0 pos=1 obj=0 op='UPDATE  INVENTORY (cr=1 pr=0 pw=0
time=0 us)'
STAT #3 id=2 cnt=1 pid=1 pos=1 obj=178146 op='TABLE ACCESS BY USER ROWID
INVENTORY (cr=1 pr=0 pw=0 time=0 us cost=1 size=8 card=1)'
CLOSE #3:c=0,e=1,dep=1,type=3,tim=810784036841
=====================
PARSING IN CURSOR #3 len=205 dep=1 uid=91 oct=2 lid=91 tim=810784036915
hv=4050659732 ad='7ffc982e3e0' sqlid='dznwb0zsr0acn'
INSERT INTO SHIPPING S (S.SHIP_ID, S.SKUCS, S.QUANTITY, S.NAME, S.SHIPTO_ADDR1,
S.SHIPTO_ADDR2, S.SHIPTO_CITY, S.SHIPTO_STATE, S.SHIPTO_POSTAL) VALUES (:B9 ,
:B8 , :B7 , :B6 , :B5 , :B4 , :B3 , :B2 , :B1 )
END OF STMT
PARSE #3:c=0,e=37,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=0,tim=810784036914
BINDS #3:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1c8e3ca0  bln=22  avl=04  flg=09
  value=10002
 Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1c8e3b08  bln=22  avl=02  flg=09
  value=2
 Bind#2
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=1c8e3b38  bln=22  avl=02  flg=09
  value=54
```

```
 Bind#3
  oacdty=01 mxl=128(100) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=01 csi=873 siz=128 off=0
  kxsbbbfp=1c8e3d20  bln=128  avl=15  flg=09
  value="Mark W. Farnham"
 Bind#4
  oacdty=01 mxl=128(100) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=01 csi=873 siz=128 off=0
  kxsbbbfp=1c8e3db0  bln=128  avl=21  flg=09
  value="4213 Mockingbird Lane"
 Bind#5
  oacdty=01 mxl=128(100) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=01 csi=873 siz=128 off=0
  kxsbbbfp=1c8e3e40  bln=128  avl=18  flg=09
  value="near the graveyard"
 Bind#6
  oacdty=01 mxl=128(100) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=01 csi=873 siz=128 off=0
  kxsbbbfp=1c8e3ed0  bln=128  avl=07  flg=09
  value="Lebanon"
 Bind#7
  oacdty=01 mxl=128(100) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=01 csi=873 siz=128 off=0
  kxsbbbfp=1c8e3f60  bln=128  avl=02  flg=09
  value="NH"
 Bind#8
  oacdty=01 mxl=128(100) mxlc=00 mal=00 scl=00 pre=00
  oacflg=13 fl2=206001 frm=01 csi=873 siz=128 off=0
  kxsbbbfp=1c8e3ff0  bln=128  avl=10  flg=09
  value="03766-1239"
EXEC #3:c=0,e=586,p=0,cr=1,cu=3,mis=0,r=1,dep=1,og=1,plh=0,tim=810784037579
STAT #3 id=1 cnt=0 pid=0 pos=1 obj=0 op='LOAD TABLE CONVENTIONAL  (cr=1 pr=0
pw=0 time=0 us)'
CLOSE #3:c=0,e=1,dep=1,type=3,tim=810784037674
=====================
PARSING IN CURSOR #3 len=6 dep=1 uid=91 oct=44 lid=91 tim=810784037713
hv=255718823 ad='0' sqlid='8ggw94h7mvxd7'
COMMIT
END OF STMT
PARSE #3:c=0,e=5,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=0,plh=0,tim=810784037712
XCTEND rlbk=0, rd_only=0, tim=810784037787
EXEC #3:c=0,e=143,p=0,cr=0,cu=1,mis=0,r=0,dep=1,og=0,plh=0,tim=810784037915
CLOSE #3:c=0,e=1,dep=1,type=3,tim=810784037983
WAIT #2: nam='SQL*Net message to client' ela= 5 driver id=1111838976 #bytes=1
p3=0 obj#=-1 tim=810784038018
EXEC #2:c=0,e=4284,p=0,cr=7,cu=7,mis=0,r=1,dep=0,og=1,plh=0,tim=810784038043
WAIT #2: nam='log file sync' ela= 37 buffer#=14412 sync scn=116487761 p3=0
obj#=-1 tim=810784038226

*** 2015-11-12 23:27:44.549
WAIT #2: nam='SQL*Net message from client' ela= 25978934 driver id=1111838976
#bytes=1 p3=0 obj#=-1 tim=810810017267
CLOSE #2:c=0,e=45,dep=0,type=0,tim=810810017472
=====================
```

```
PARSING IN CURSOR #3 len=68 dep=0 uid=91 oct=42 lid=91 tim=810810017547
hv=222310797 ad='0' sqlid='bb055g46n0ccd'
alter session set events '10046 trace name context forever, level 0'
END OF STMT
PARSE #3:c=0,e=27,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,plh=0,tim=810810017547
EXEC #3:c=0,e=908,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,plh=0,tim=810810018738

*** 2015-11-12 23:27:51.804
CLOSE #3:c=0,e=14,dep=0,type=1,tim=810817271186
=====================
PARSING IN CURSOR #2 len=32 dep=0 uid=91 oct=3 lid=91 tim=810817271465
hv=415332577 ad='7ffbe985f88' sqlid='1k2snmccc2y71'
SELECT :status status FROM DUAL
END OF STMT
PARSE
#2:c=0,e=107,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=1388734953,tim=810817271465
EXEC
#2:c=0,e=46,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=1388734953,tim=810817271619
FETCH
#2:c=0,e=13,p=0,cr=0,cu=0,mis=0,r=1,dep=0,og=1,plh=1388734953,tim=810817271729
STAT #2 id=1 cnt=1 pid=0 pos=1 obj=0 op='FAST DUAL  (cr=0 pr=0 pw=0 time=0 us
cost=2 size=0 card=1)'
FETCH
#2:c=0,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,plh=1388734953,tim=810817272127

*** 2015-11-12 23:27:58.523
CLOSE #2:c=0,e=27,dep=0,type=0,tim=810823990649
=====================
PARSING IN CURSOR #3 len=23 dep=0 uid=91 oct=3 lid=91 tim=810823990936
hv=2348030290 ad='7ffc9884340' sqlid='awttf6q5z86ak'
select * from inventory
END OF STMT
PARSE
#3:c=0,e=109,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=947045911,tim=810823990935
EXEC
#3:c=0,e=38,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=947045911,tim=810823991062
FETCH
#3:c=0,e=101,p=0,cr=3,cu=0,mis=0,r=1,dep=0,og=1,plh=947045911,tim=810823991261
FETCH
#3:c=0,e=22,p=0,cr=1,cu=0,mis=0,r=1,dep=0,og=1,plh=947045911,tim=810823991555
STAT #3 id=1 cnt=2 pid=0 pos=1 obj=178146 op='TABLE ACCESS FULL INVENTORY (cr=4
pr=0 pw=0 time=0 us cost=2 size=16 card=2)'

*** 2015-11-12 23:28:05.026
CLOSE #3:c=0,e=21,dep=0,type=0,tim=810830492741
=====================
PARSING IN CURSOR #2 len=22 dep=0 uid=91 oct=3 lid=91 tim=810830492966
hv=1161697224 ad='7ffbe2341c8' sqlid='146zmb12mw5y8'
select * from shipping
END OF STMT
PARSE
#2:c=0,e=86,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=3520420,tim=810830492965
EXEC
#2:c=0,e=34,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=3520420,tim=810830493060
```

```
FETCH
#2:c=0,e=92,p=0,cr=5,cu=0,mis=0,r=1,dep=0,og=1,plh=3520420,tim=810830493239
FETCH
#2:c=0,e=15,p=0,cr=1,cu=0,mis=0,r=1,dep=0,og=1,plh=3520420,tim=810830493522
STAT #2 id=1 cnt=2 pid=0 pos=1 obj=179719 op='TABLE ACCESS FULL SHIPPING (cr=6
pr=0 pw=0 time=0 us cost=2 size=176 card=2)'

*** 2015-11-12 23:36:43.371
CLOSE #2:c=0,e=22,dep=0,type=0,tim=811348823353
XCTEND rlbk=1, rd_only=1, tim=811348823599
XCTEND rlbk=0, rd_only=1, tim=811348823970
====================
PARSING IN CURSOR #3 len=447 dep=1 uid=0 oct=2 lid=0 tim=811348824186
hv=1097020010 ad='7ffbe79d310' sqlid='f711myt0q6cma'
insert into sys.aud$( sessionid,entryid,statement,ntimestamp#,
userid,userhost,terminal,action#,returncode,
logoff$lread,logoff$pread,logoff$lwrite,logoff$dead,
logoff$time,comment$text,spare1,clientid,sessioncpu,proxy$sid,user$guid,
instance#,process#,auditid,dbid) values(:1,:2,:3,SYS_EXTRACT_UTC(SYSTIMESTAMP),
:4,:5,:6,:7,:8,     :9,:10,:11,:12,     cast(SYS_EXTRACT_UTC(systimestamp) as
date),:13,:14,:15,:16,:17,:18,     :19,:20,:21,:22)
END OF STMT
PARSE #3:c=0,e=86,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,plh=0,tim=811348824185
EXEC #3:c=0,e=224,p=0,cr=1,cu=2,mis=0,r=1,dep=1,og=4,plh=0,tim=811348824632
STAT #3 id=1 cnt=0 pid=0 pos=1 obj=0 op='LOAD TABLE CONVENTIONAL  (cr=1 pr=0
pw=0 time=0 us)'
CLOSE #3:c=0,e=7,dep=1,type=0,tim=811348824729
CLOSE #1:c=0,e=16,dep=0,type=0,tim=811348825279
```