The SQL Statement Cache

Hedwig Fuchs
hedwig.fuchs@hcl-software.com

Agenda

- Introduction
- onconfig Parameter
- Monitoring
- syssscelem
- bad query plan
- save the statement cache
- STMT_CACHE 1

SQL Statement Cache - Introduction

Running a SQL statement has two steps:

- optimizing
- executing

The SQL statement cache (SSC) is used to store optimized SQL statements.

When a statement is stored in the SSC and shall be executed, the optimizing step is skipped.

-> **Performance improvement**

SQL Statement Cache - Introduction

But maybe it does not make sense to have ALL statements in the statement cache

- query plan for a statement is changing often
- statements which are executed rarely

SQL Statement Cache - Onconfig Parameter

STMT_CACHE turns on, enables or disables the SSC

STMT_CACHE_HITS specifies the number of references to statement before it is fully inserted into the SSC

STMT_CACHE_SIZE defines the size

STMT_CACHE_NUMPOOL defines number of memory pools

STMT_CACHE_NOLIMIT controls whether SSC can grow or not

STMT_QUERY_PLAN defines if query plan is hold in the SSC

SQL Statement Cache - Onconfig Parameter

STMT_CACHE	STMT_CACHE_HITS	STMT_CACHE_NOLIMIT	STMT_CACHE_QUERY_PLAN
onmode -wf/wm	onmode -wf/wm	onmode -wf/wm	onmode -wf/wm
	onmode -W	onmode -W	
onmode -e			
SQL Admin API	SQL Admin API	SQL Admin API	

STMT_CACHE_NUMPOOL, STMT_CACHE_SIZE need a restart

SQL Statement Cache – other settings

environment variable STMT_CACHE

SQL statement SET STATEMENT CACHE ON/OFF

allow control on a session level/statement level.

SQL Statement Cache - Monitoring

onstat –g ssc is the command to monitor the SSC

```
$> onstat -g ssc

Statement Cache Summary:
#lrus   currsize   maxsize   Poolsize    #hits   nolimit
8       13056      524288    32768       4       0

Statement Cache Entries:

uniqid lru hash ref_cnt hits flag heap_ptr   database       user
------------------------------------------------------------------
     1   0 3367        0    7  -F- 46640040   db1            myuser
  select customer_num from customer

    Total number of entries: 1
```

SQL Statement Cache - Monitoring

```
Statement Cache Summary:

#lrus   currsize    maxsize     Poolsize   #hits    nolimit
8       0           524288      16384      4        1
```

LRU = last recently used

Number of LRUs is defined by VPCLASS cpu

SQL Statement Cache - Monitoring

```
Statement Cache Entries:
uniqid lru hash ref_cnt hits flag heap_ptr  database   user
-------------------------------------------------------------
    14   5 3367        1    4  -F- 4e1a6040  db1        myuser
  select customer_num from customer
```

ref_cnt = 1: is currently used by 1 session

hits = 4: is hit 4 times after 1st execution

flag = F: statement is fully cached

SQL Statement Cache - Monitoring

Statements are always in the statement cache even when not yet eligible for reuse.

SSC is empty:

```
$> onstat -g ssc
Statement Cache Summary:
#lrus    currsize     maxsize
8        0
Stmt Cache is empty.
```

SSC has statements which are not (yet) eligible for reuse:

```
$> onstat -g ssc
Statement Cache Summary:
#lrus    currsize     maxsize
8        3072
Total number of entries: 0
```

SQL Statement Cache - Monitoring

onstat –g ssc all shows them

```
$> onstat -g ssc all
Statement Cache Summary:
#lrus   currsize   maxsize  Poolsize        #hits   nolimit
8       3072       524288   20480           4       0

Statement Cache Entries:

uniqid lru hash ref_cnt hits flag heap_ptr  database  user
------------------------------------------------------------
     1   0 3367        0    3  --- 46640040  db1       myuser
  select customer_num from customer

     Total number of entries: 1
```

SQL Statement Cache - Monitoring

onstat –g ssc pool for the memory pools

```
$> onstat -g ssc pool
Pool Summary:
name         class addr     totalsize freesize   #allocfrag #freefrag
sscpool0     V     46428040 12288     1376       2          2
Blkpool Summary:
name                 class addr             size              #blks
Pool Summary:
name         class addr     totalsize freesize   #allocfrag #freefrag
sscpool1     V     46429040 8192      3792       2          2
Blkpool Summary:
name                 class addr             size              #blks

$> onstat -g mem
Pool Summary:
name         class addr     totalsize freesize   #allocfrag #freefrag
sscpool0     V     46428040 12288     1376       2          2
sscpool1     V     46429040 8192      3792       2          2
```

SQL Statement Cache - Monitoring

From a case where I had to investigate shared memory usage

```
$> onstat -g ssc

Statement Cache Summary:
#lrus   currsize   maxsize   Poolsize     #hits   nolimit
8       187317664  524288    189112320    4       0
    Total number of entries: 0
```

SQL Statement Cache - syssscelem

The **syssscelem** table provides information about Statement Cache Entries (similar to onstat -g ssc)

```
uniqid      1
lru         0
hash        5442
ref_cnt     0
hits        7
flag        2
heap_ptr    1180958784
database    db1
user        myuser
stmtstring  select customer_num from customer where customer_num=20

uniqid lru hash ref_cnt hits flag heap_ptr       database            user
------------------------------------------------------------------------
-
     1   0 5442        0    7  -F- 46640040       db1                 myuser
  select customer_num from customer where customer_num=20
```

SQL Statement Cache - syssscelem

Columns in the table which are not in onstat -g ssc:

- queryplan
- valid
- locked

SQL Statement Cache - syssscelem

When STMT_CACHE_QUERY_PLAN is set to 1, syssscelem.**queryplan** has the query plan of the cached statement.

```
$> echo "select stmtstring, queryplan from syssscelem" …

stmtstring  select customer_num from customer where customer_num=20
queryplan
     QUERY: (OPTIMIZATION TIMESTAMP: 09-08-2026 08:55:11)
      Estimated Cost: 1
      Estimated # of Rows Returned: 1

        1) hedwigb.customer: INDEX PATH
            (1) Index Name: myuser. 100_1
               Index Keys: customer_num   (Key-Only)   (Serial, fragments: ALL)
               Lower Index Filter: myuser.customer.customer_num = 20
```

SQL Statement Cache - syssscelem

- queryplan **UNAVAILABLE**
  Means either STMT_CACHE_QUERY_PLAN not set or statement not yet fully cached

- queryplan is defined as char(16000)
  Maybe query plan is not complete

- No query statistics, only the plan

SQL Statement Cache - bad query plan

Problem

Query suddenly taking much more time

Poss. Cause

bad query plan in the SSC

First steps are to

- confirm that statement is in the SSC
- check the queryplan (when exists)

SQL Statement Cache - bad query plan

What can be done

1. restart
2. onmode -e flush
   Flushes the statements that are not in use (ref_cnt = 0) from the SQL statement cache.

SQL Statement Cache - bad query plan

3. Invalidate the entry in the SSC via column **valid** in syssscelem

```
$> echo "update syssscelem set valid=0 where uniqid=5" | dbaccess

Statement Cache Entries:

uniqid lru hash ref_cnt hits flag heap_ptr  database        user
------------------------------------------------------------------
     2   1 3367        0   12  DF- 4e4de040  db1             myuser
  select customer_num from customer
```

A statement can be invalidated even when "in use".
flag shows D for dropped

SQL Statement Cache - bad query plan

When statement is again eligible for the SSC, a new entry will be added

```
Statement Cache Entries:

uniqid lru hash ref_cnt hits flag heap_ptr  database   user
-------------------------------------------------------------
     2   1 3367        0   12  DF- 4e4de040  db1        myuser
  select customer_num from customer

     5   3 3367        0   11  -F- 46640840  db1        myuser
  select customer_num from customer
```

SQL Statement Cache - bad query plan

4. Consider to lock a statement in the SSC via column **locked** in syssscelem

```
$> echo "update syssscelem set locked=1 where uniqid=1" | dbaccess
$> onstat -g ssc

Statement Cache Entries:
uniqid lru hash ref_cnt hits flag heap_ptr     database       user
------------------------------------------------------------------
     1   0 3367        0   15  -FL 46640040     db1            mysuser
  select customer_num from customer
```

flag shows L for locked

Now not even an update statistics for the table will drop the statement.

SQL Statement Cache - syssscelem

When a locked statement is invalidate, it is not locked any longer

```
$> echo "update syssscelem set locked=1 where uniqid=1" | dbaccess
$> echo "update syssscelem set valid=0 where uniqid=1" | dbaccess
Statement Cache Entries:

uniqid lru hash ref_cnt hits flag heap_ptr     database    user
------------------------------------------------------------------
     1   0 3367        0   15  DF- 46640040     db1         myuser
  select customer_num from customer
```

When a statement is invalid, it can't be made valid again

```
$> echo "update syssscelem set valid=1 where uniqid=1" | dbaccess

  346: Could not update a row in the table.
12238: Cannot update Statement Cache entry that is marked as invalid.
```

SQL Statement Cache - save the statement cache

The SSC can be saved. To restore it after a restart of the instance.

When a cache entry is locked, then the query plan will be saved too.

```
$> echo "execute function task('statement cache save') " | dbaccess
(expression)  Statement Cache saved to sysadmin:statement_cache

$> echo "select * from statement_cache" | dbaccess

uniqid   1
type     1
data     select customer_num from customer

uniqid   1
type     2
data     AAABAAAAAgABAAIAAAADIgAAAQAAAAAAAQA
```

SQL Statement Cache - save the statement cache

```
$> echo "execute function task('statement cache restore')" | dbaccess
 (expression)  Statement Cache restored from sysadmin:statement_cache

$> echo "select * from syssscelem" |dbaccess

stmtstring  select customer_num from customer
queryplan
        QUERY: (OPTIMIZATION TIMESTAMP: 03-09-2026 08:59:41)

stmtstring  select * from items
queryplan
        QUERY: (OPTIMIZATION TIMESTAMP: 09-09-2026 10:38:23)
```

SQL Statement Cache - STMT_CACHE 1

Example from a case:

```
Statement Cache Entries:
uniqid lru hash ref_cnt hits flag heap_ptr   database            user
------------------------------------------------------------------------
     1   0 5441        0    6  -F- 4663e040   db1                 myuser
  select customer_num from customer where customer_num=10
     9   0 5449        0    3  -F- 4e29d040   db1                 myuser
  select customer_num from customer where customer_num=90
    17   0 5450        0    7  -F- 4e2bd040   db1                 myuser
  select customer_num from customer where customer_num=28
     2   1 5442        0    3  -F- 4e281040   db1                 myuser
  select customer_num from customer where customer_num=20
    10   1 5443        0    4  -F- 4e2a1040   db1                 myuser
  select customer_num from customer where customer_num=21
    18   1 5451        0    5  -F- 4e2c1040   db1                 myuser
  select customer_num from customer where customer_num=29
     3   2 5443        0    5  -F- 4e285040   db1                 myuser
  select customer_num from customer where customer_num=30
    11   2 5444        0    3  -F- 4e2a5040   db1                 myuser
  select customer_num from customer where customer_num=22
```

SQL Statement Cache - STMT_CACHE 1

STMT_CACHE 1 means enable the SSC but it is not used per default.

Using the environment variable STMT_CACHE then allows to use the SSC for this application.

The SQL statement SET STATEMENT CACHE ON/OFF then allows to use the SSC for specific statements.

Questions?

Speaker
Hedwig Fuchs
hedwig.fuchs@hcl-software.com

SQL Statement Cache - DEBUG

Environment variable

STMT_CACHE_DEBUG=<1 or 2>:<file_name>

```
$> export STMT_CACHE_DEBUG=2:mySSCtrace
$> echo "select * from systables" |dbaccess
$> ls -l
-rw-rw-rw- 1 myuser rd 546 Sep 09 07:47 mySSCtrace1172

SSC hasn't reached cache limit. pool_usage:1952 cache_usage:0 cache_size:524288
SSC identity (stmt not found):  select * from systables
SSC qualify (OK): select * from systables
SSC key insert (OK):  select * from systables
SSC full insert (OK):  select * from systables
```