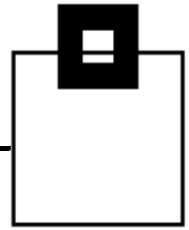
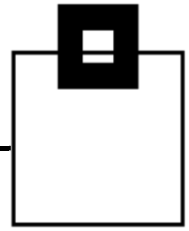


Workload Insights Without a Trace
-
Introducing DB2 z/OS SQL tracking

Agenda

- What's new in DB2 10
- What's of interest for geeks in DB2 10
- What's of interest for (performance) auditors
- Questions

DSC at a glance



Characteristics:

The DSC is where dynamic SQL statements, and *only* dynamic SQL statements, reside once they have been PREPAREd if certain ZPARM and/or BIND options are in use.

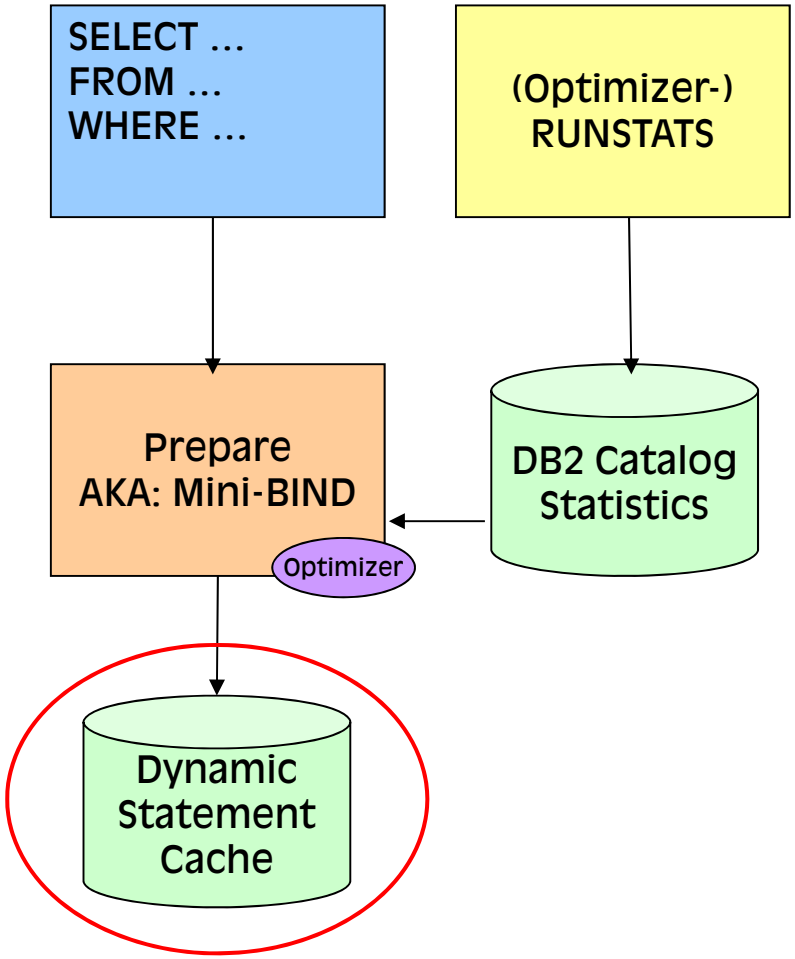
The next time the exact same statement is to be PREPAREd the cache is searched and, if all is valid and certain ZPARM and/or BIND options are in use, then the PREPARE is avoided

→ Thus saving **lots** of CPU time.

Ideally an SQL statement should stay in the cache forever, but the real world shows that two days of residency or latency is typical.

DSC at a glance

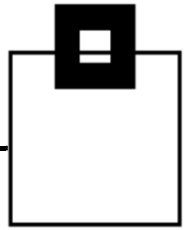
Characteristics:



Access Paths for dynamic SQL are determined on the fly and stored in the DSC.

LRU, RUNSTATS, ALTER, DROP, REVOKE, DB2 RESTART invalidates and flushes the DSC for an object.

DSC at a glance

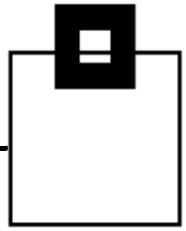


Cache behavior overview:

CACHEDYN->		NO	YES
KEEPDYNAMIC	NO	• No skeletons cached in EDMP • Only full prepares • No prepared statements kept across commits (note1) • No statement strings kept across commits NONE	• Skeletons cached in EDMP • 1st prepare full; others short (note 2) • No prepared statements kept across commits (note 1) • No statement strings kept across commits Global
	YES	• No skeletons cached in EDMP • Only full prepares • No prepared statements kept across commits (note 1) • Stmt strings kept across commits – implicit prepares LOCAL	• Skeletons cached in EDMP • 1st prepare full; others short (note 2) • Prepared stmts across commits – avoids prepares (note 3) • Stmt strings kept across commits – implicit prepares FULL AKA: Prepare Avoidance

Note 1: unless a cursor WITH HOLD is open
Note 2: unless invalidated or flushed out due to LRU
Note 3: assuming MAXKEEPD > 0

DSC at a glance



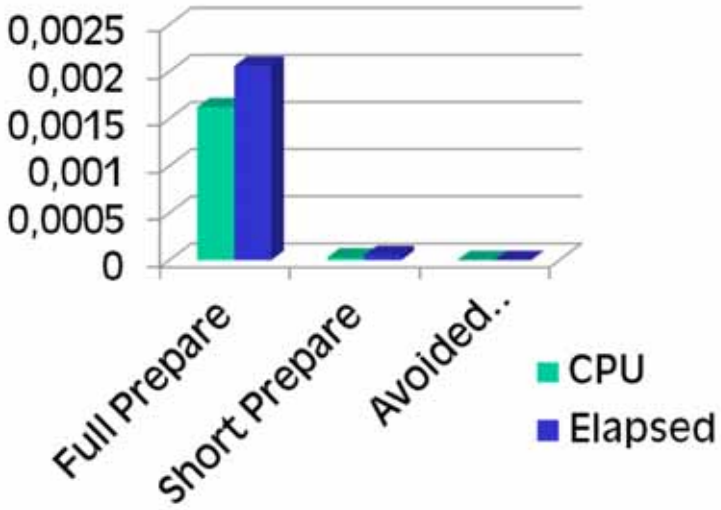
The many flavors of PREPARE:

- Full
  Skeleton copy of the SQL is not in the cache or the cache is not active. Caused by a PREPARE or EXECUTE IMMEDIATE statement.
- Short
  A skeleton copy of the PREPAREd SQL statement is copied to local storage.
- Avoided
  PREPARE avoided by using full caching. PREPAREd statement information is still in thread's local storage.
- Implicit
  Due to limits, such as MAXKEEPD, a PREPARE cannot be avoided and DB2 will issue the PREPARE on behalf of the application.

DSC at a glance

DB2 Setup and Support:

- The right setup saves a lot of money
  - Exploit the full flavor of caching
  - MAXKEEPD>0
  - CACHEDYN=YES
  - KEEPDYNAMIC(YES)



DSC at a glance

So far, so good …

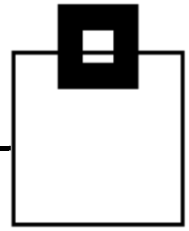
The DSC opened up great opportunities for dynamic SQL

- Packaged applications like SAP
- Less-mainframe-skilled developers
- Interactive multi-platform solutions
- The mainframe competing with the distributed environment
- Cost efficient and well performing applications

→ The key is the Dynamic Statement Cache …

… USE IT, or IBM will make you use it…

DSC ZPARMs through the ages



DSC relevant parameters and what they have been set to or defaulted to over the releases of DB2

Version	MAXKEEPD 0 - 65535	CACHEDYN (Yes/No)	Notes
5	5000	NO	EDMPOOL was the CACHE
6	5000	NO	If CACHEDYN „YES“ then new EDMDSPAC With valid range 1K – 2,097,152K
7	5000	NO	New EDMDSMAX with valid range 0 – 2,097,152K and default 1,048,576K
8	5000	YES	EDMDSPAC and EDMDSMAX removed. New EDMSTMTC with valid range 5,000K – 1,048,576K and new „opaque“ ZPARM CACHEDYN_FREELOCAL valid values 0 or 1 with default 0 (off)
9	5000	YES	CACHEDYN_FREELOCAL default changed to 1 (on) and EDMSTMTC default changed to 56,693K
10	5000	YES	EDMSTMTC default changed to 113,386K

DSC Tracing

Besides the caching, the DSC gives you insight:

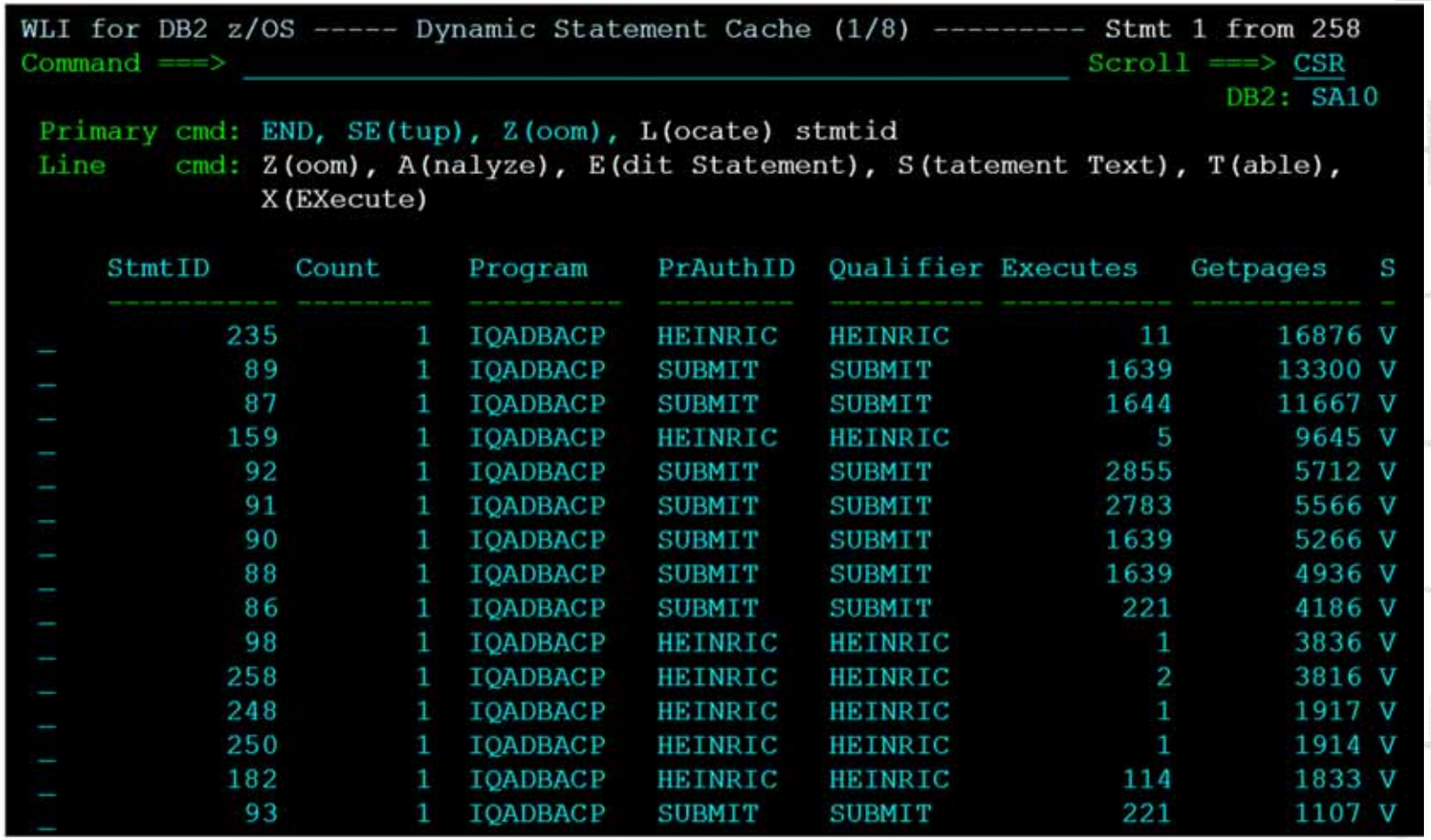
- Memory resident storage of PREPAREd dynamic SQL statements
- SQL text
- Statement ID
- Date/time, current status
- Resource consumption

→ Start IFCIDs 316, 317 (318) to access the data*

…once the data is gathered, decide on what you want to analyze…

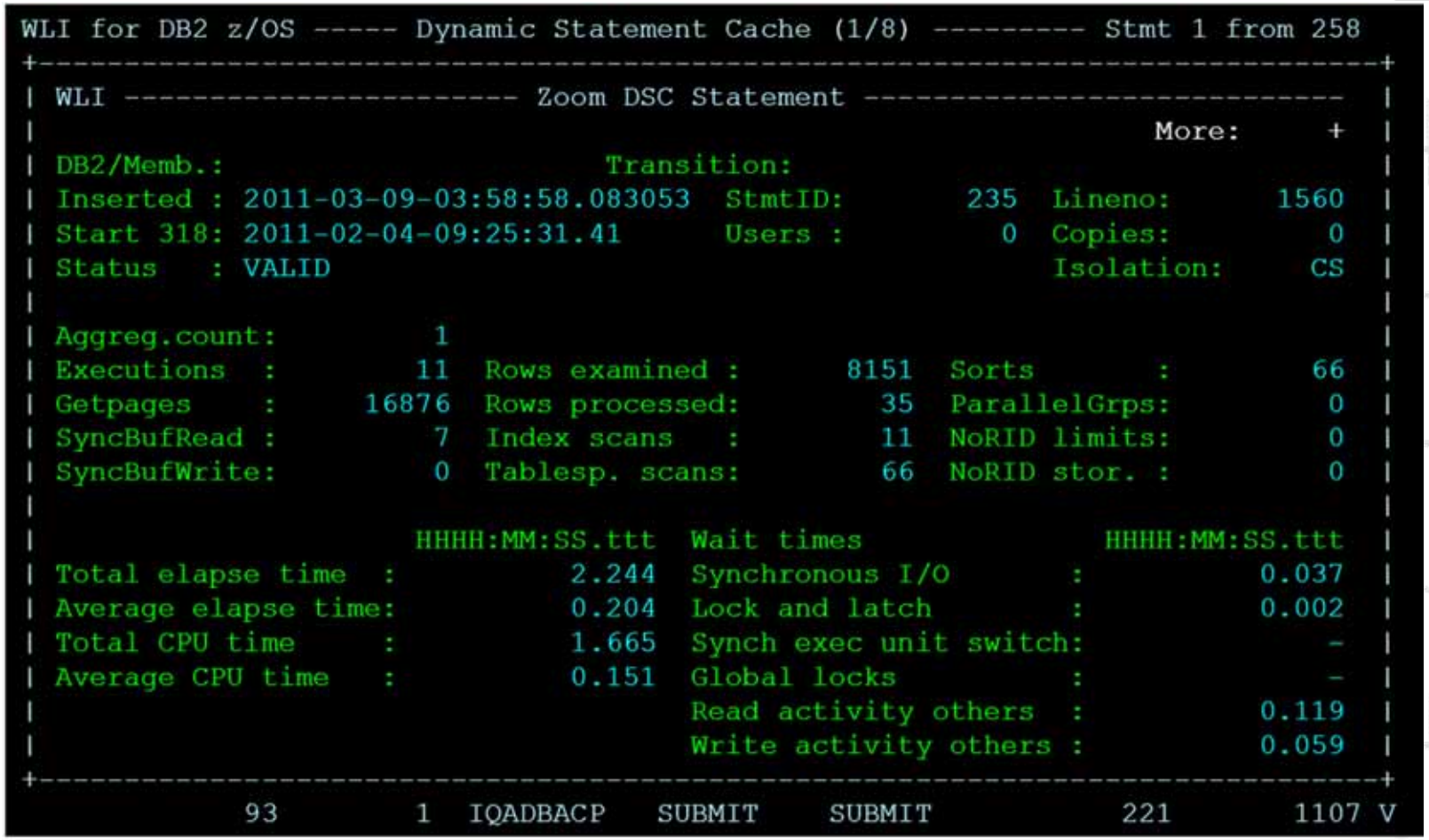
*These three IFCIDs are not really IFCIDs but more „switches“ to enable externalization of dynamic SQL metrics

DSC Tracing

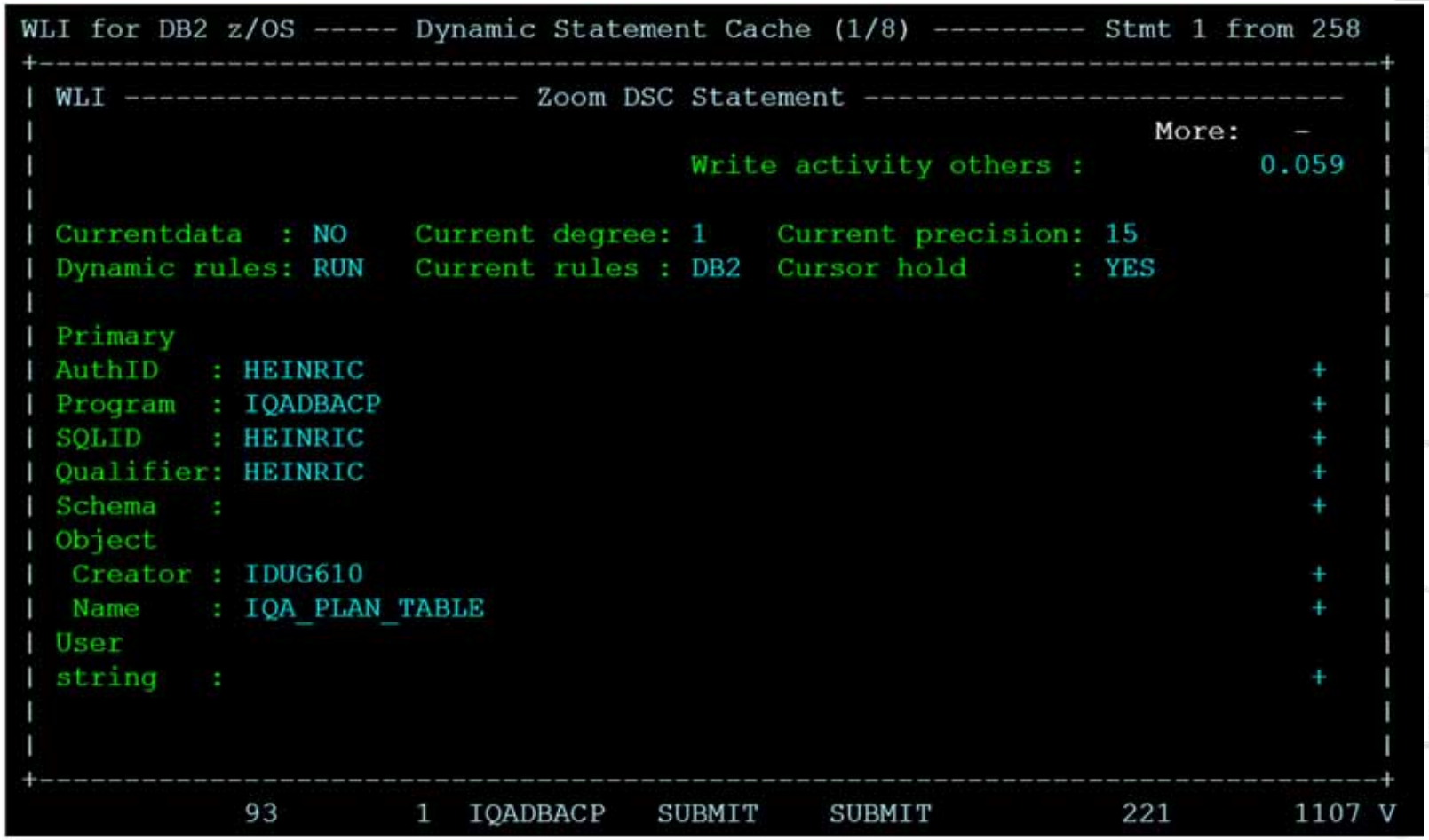


```
WLI for DB2 z/OS ----- Dynamic Statement Cache (1/8) --------- Stmt 1 from 258
Command ===> _____________________________________________ Scroll ===> CSR
                                                            DB2: SA10
 Primary cmd: END, SE(tup), Z(oom), L(ocate) stmtid
 Line    cmd: Z(oom), A(nalyze), E(dit Statement), S(tatement Text), T(able),
              X(EXecute)

     StmtID     Count     Program    PrAuthID  Qualifier Executes   Getpages   S
     ---------- --------  ---------  --------  --------- ---------- ---------- -
_          235         1  IQADBACP   HEINRIC   HEINRIC           11      16876 V
_           89         1  IQADBACP   SUBMIT    SUBMIT          1639      13300 V
_           87         1  IQADBACP   SUBMIT    SUBMIT          1644      11667 V
_          159         1  IQADBACP   HEINRIC   HEINRIC            5       9645 V
_           92         1  IQADBACP   SUBMIT    SUBMIT          2855       5712 V
_           91         1  IQADBACP   SUBMIT    SUBMIT          2783       5566 V
_           90         1  IQADBACP   SUBMIT    SUBMIT          1639       5266 V
_           88         1  IQADBACP   SUBMIT    SUBMIT          1639       4936 V
_           86         1  IQADBACP   SUBMIT    SUBMIT           221       4186 V
_           98         1  IQADBACP   HEINRIC   HEINRIC            1       3836 V
_          258         1  IQADBACP   HEINRIC   HEINRIC            2       3816 V
_          248         1  IQADBACP   HEINRIC   HEINRIC            1       1917 V
_          250         1  IQADBACP   HEINRIC   HEINRIC            1       1914 V
_          182         1  IQADBACP   HEINRIC   HEINRIC          114       1833 V
_           93         1  IQADBACP   SUBMIT    SUBMIT           221       1107 V
```

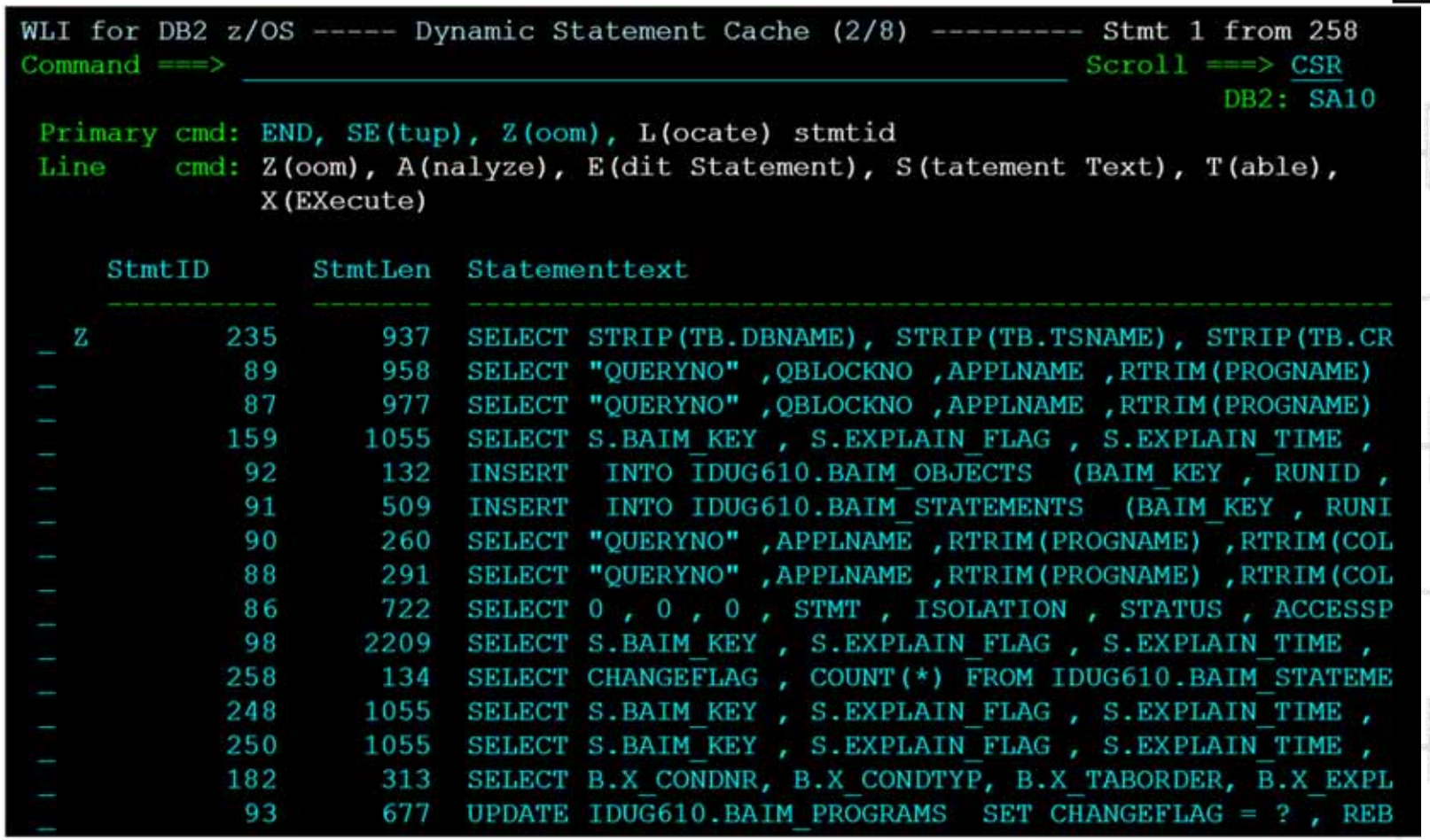
DSC Tracing



DSC Tracing

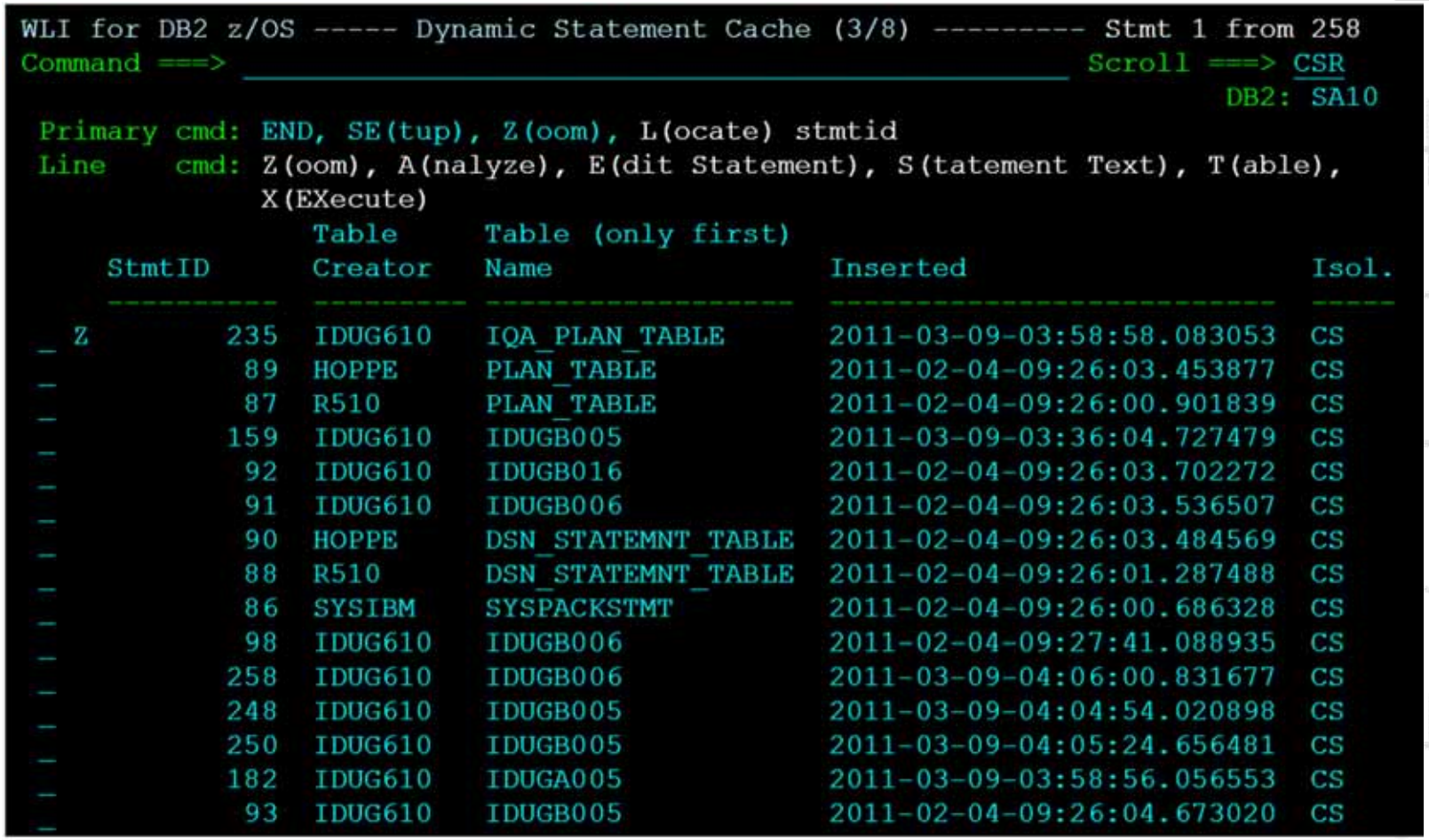


DSC Tracing



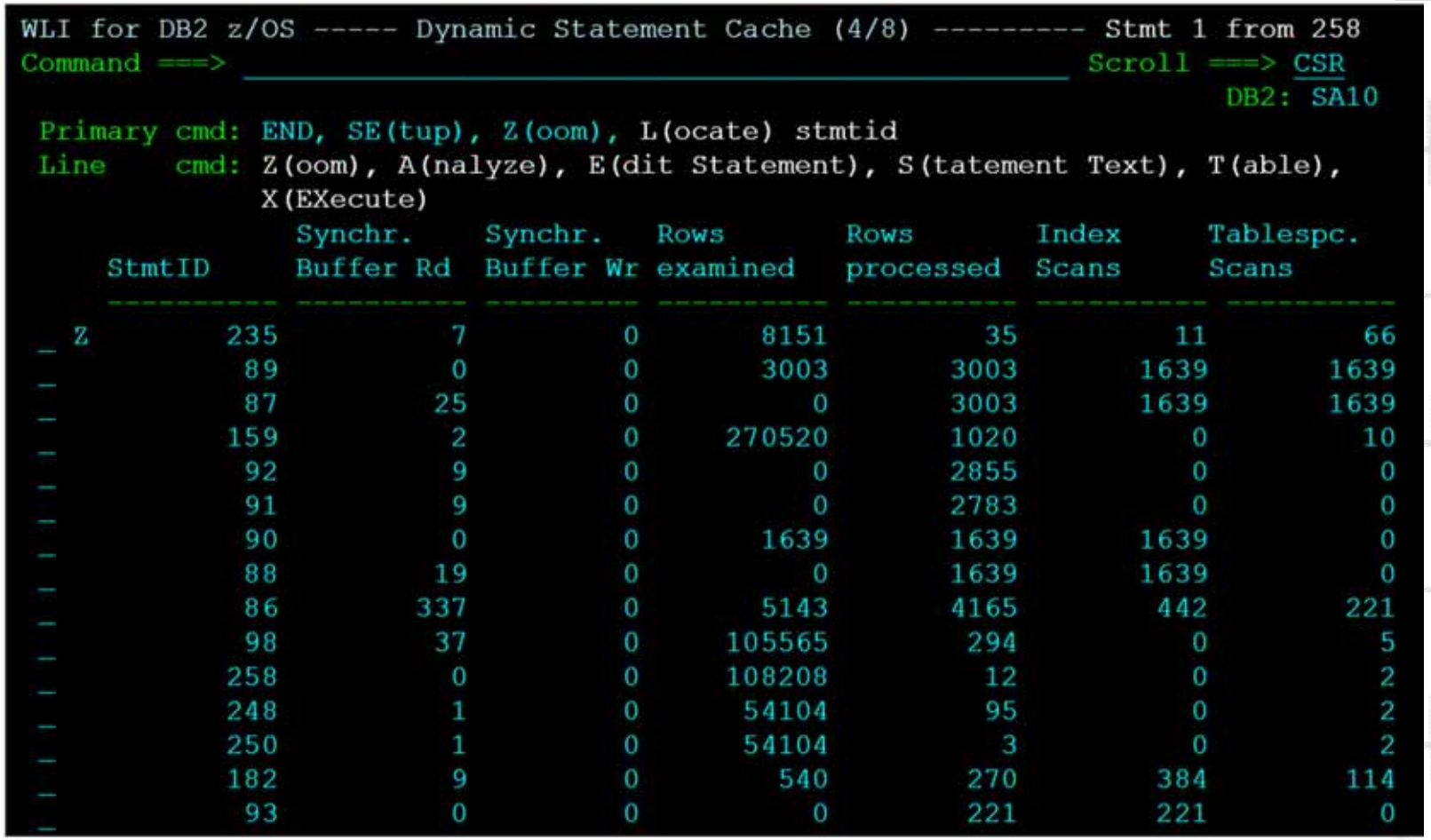
```
WLI for DB2 z/OS ----- Dynamic Statement Cache (2/8) --------- Stmt 1 from 258
Command ===> _____________________________________________ Scroll ===> CSR
                                                            DB2: SA10
 Primary cmd: END, SE(tup), Z(oom), L(ocate) stmtid
 Line    cmd: Z(oom), A(nalyze), E(dit Statement), S(tatement Text), T(able),
              X(EXecute)

     StmtID      StmtLen  Statementtext
     ----------  -------  ------------------------------------------------
 _ Z        235      937  SELECT STRIP(TB.DBNAME), STRIP(TB.TSNAME), STRIP(TB.CR
 _           89      958  SELECT "QUERYNO" ,QBLOCKNO ,APPLNAME ,RTRIM(PROGNAME)
 _           87      977  SELECT "QUERYNO" ,QBLOCKNO ,APPLNAME ,RTRIM(PROGNAME)
 _          159     1055  SELECT S.BAIM_KEY , S.EXPLAIN_FLAG , S.EXPLAIN_TIME ,
 _           92      132  INSERT  INTO IDUG610.BAIM_OBJECTS  (BAIM_KEY , RUNID ,
 _           91      509  INSERT  INTO IDUG610.BAIM_STATEMENTS  (BAIM_KEY , RUNI
 _           90      260  SELECT "QUERYNO" ,APPLNAME ,RTRIM(PROGNAME) ,RTRIM(COL
 _           88      291  SELECT "QUERYNO" ,APPLNAME ,RTRIM(PROGNAME) ,RTRIM(COL
 _           86      722  SELECT 0 , 0 , 0 , STMT , ISOLATION , STATUS , ACCESSP
 _           98     2209  SELECT S.BAIM_KEY , S.EXPLAIN_FLAG , S.EXPLAIN_TIME ,
 _          258      134  SELECT CHANGEFLAG , COUNT(*) FROM IDUG610.BAIM_STATEME
 _          248     1055  SELECT S.BAIM_KEY , S.EXPLAIN_FLAG , S.EXPLAIN_TIME ,
 _          250     1055  SELECT S.BAIM_KEY , S.EXPLAIN_FLAG , S.EXPLAIN_TIME ,
 _          182      313  SELECT B.X_CONDNR, B.X_CONDTYP, B.X_TABORDER, B.X_EXPL
 _           93      677  UPDATE IDUG610.BAIM_PROGRAMS  SET CHANGEFLAG = ? , REB
```

DSC Tracing



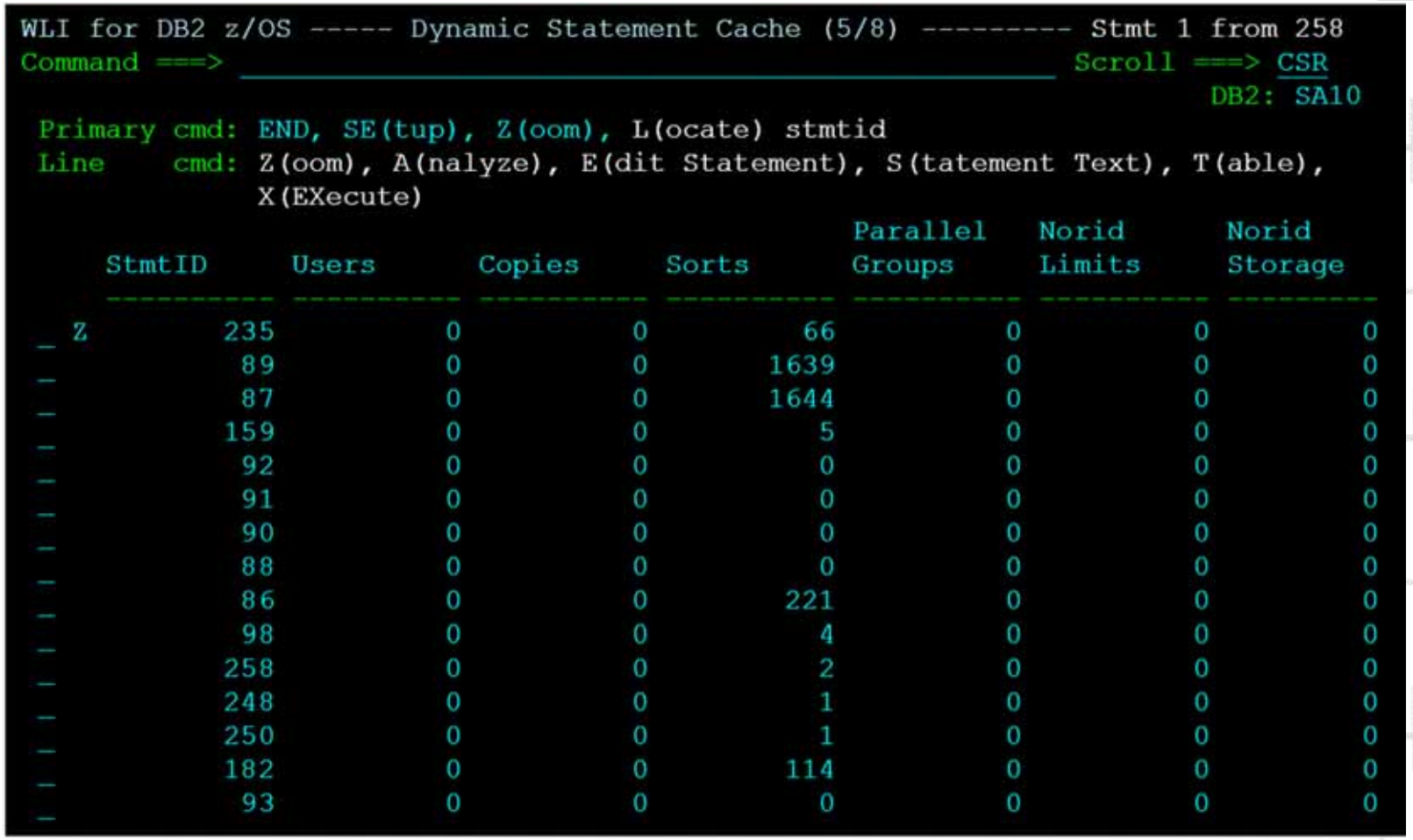
```
WLI for DB2 z/OS ----- Dynamic Statement Cache (3/8) --------- Stmt 1 from 258
Command ===> _____________________________________________ Scroll ===> CSR
                                                            DB2: SA10
 Primary cmd: END, SE(tup), Z(oom), L(ocate) stmtid
 Line    cmd: Z(oom), A(nalyze), E(dit Statement), S(tatement Text), T(able),
              X(EXecute)
                 Table     Table (only first)
     StmtID      Creator   Name                 Inserted                   Isol.
     ----------  --------  -------------------  -------------------------  -----
 _ Z        235  IDUG610   IQA_PLAN_TABLE       2011-03-09-03:58:58.083053  CS
 _           89  HOPPE     PLAN_TABLE           2011-02-04-09:26:03.453877  CS
 _           87  R510      PLAN_TABLE           2011-02-04-09:26:00.901839  CS
 _          159  IDUG610   IDUGB005             2011-03-09-03:36:04.727479  CS
 _           92  IDUG610   IDUGB016             2011-02-04-09:26:03.702272  CS
 _           91  IDUG610   IDUGB006             2011-02-04-09:26:03.536507  CS
 _           90  HOPPE     DSN_STATEMNT_TABLE   2011-02-04-09:26:03.484569  CS
 _           88  R510      DSN_STATEMNT_TABLE   2011-02-04-09:26:01.287488  CS
 _           86  SYSIBM    SYSPACKSTMT          2011-02-04-09:26:00.686328  CS
 _           98  IDUG610   IDUGB006             2011-02-04-09:27:41.088935  CS
 _          258  IDUG610   IDUGB006             2011-03-09-04:06:00.831677  CS
 _          248  IDUG610   IDUGB005             2011-03-09-04:04:54.020898  CS
 _          250  IDUG610   IDUGB005             2011-03-09-04:05:24.656481  CS
 _          182  IDUG610   IDUGA005             2011-03-09-03:58:56.056553  CS
 _           93  IDUG610   IDUGB005             2011-02-04-09:26:04.673020  CS
```

DSC Tracing



```
WLI for DB2 z/OS ----- Dynamic Statement Cache (4/8) --------- Stmt 1 from 258
Command ===> _____________________________________________ Scroll ===> CSR
                                                            DB2: SA10
 Primary cmd: END, SE(tup), Z(oom), L(ocate) stmtid
 Line    cmd: Z(oom), A(nalyze), E(dit Statement), S(tatement Text), T(able),
              X(EXecute)
                Synchr.    Synchr.   Rows       Rows       Index      Tablespc.
     StmtID     Buffer Rd  Buffer Wr examined   processed  Scans      Scans
     ---------- ---------- --------- ---------- ---------- ---------- ----------
 _ Z        235          7         0       8151         35         11         66
 _           89          0         0       3003       3003       1639       1639
 _           87         25         0          0       3003       1639       1639
 _          159          2         0     270520       1020          0         10
 _           92          9         0          0       2855          0          0
 _           91          9         0          0       2783          0          0
 _           90          0         0       1639       1639       1639          0
 _           88         19         0          0       1639       1639          0
 _           86        337         0       5143       4165        442        221
 _           98         37         0     105565        294          0          5
 _          258          0         0     108208         12          0          2
 _          248          1         0      54104         95          0          2
 _          250          1         0      54104          3          0          2
 _          182          9         0        540        270        384        114
 _           93          0         0          0        221        221          0
```

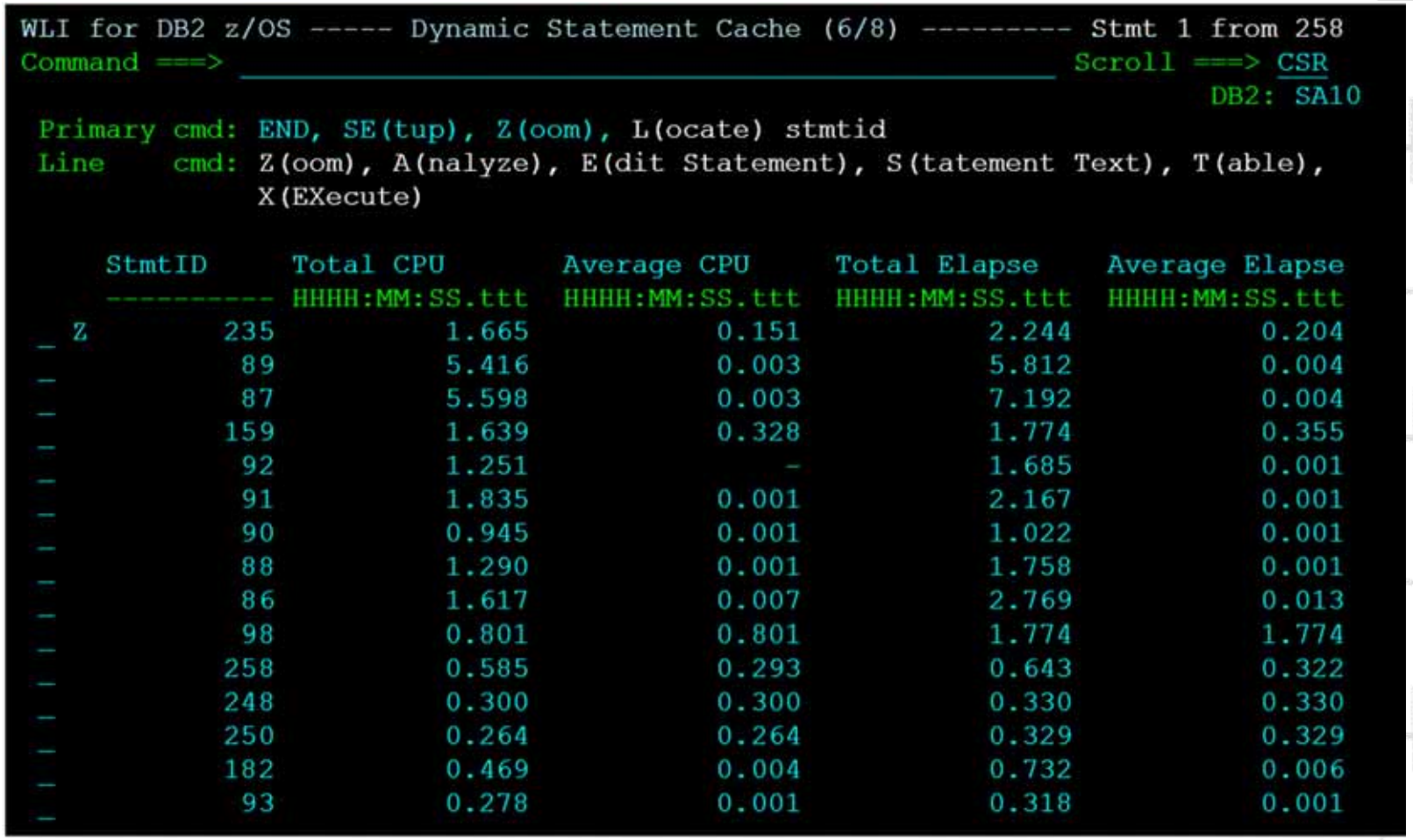
DSC Tracing



```
WLI for DB2 z/OS ----- Dynamic Statement Cache (5/8) --------- Stmt 1 from 258
Command ===> _____________________________________________ Scroll ===> CSR
                                                            DB2: SA10
 Primary cmd: END, SE(tup), Z(oom), L(ocate) stmtid
 Line    cmd: Z(oom), A(nalyze), E(dit Statement), S(tatement Text), T(able),
              X(EXecute)
```

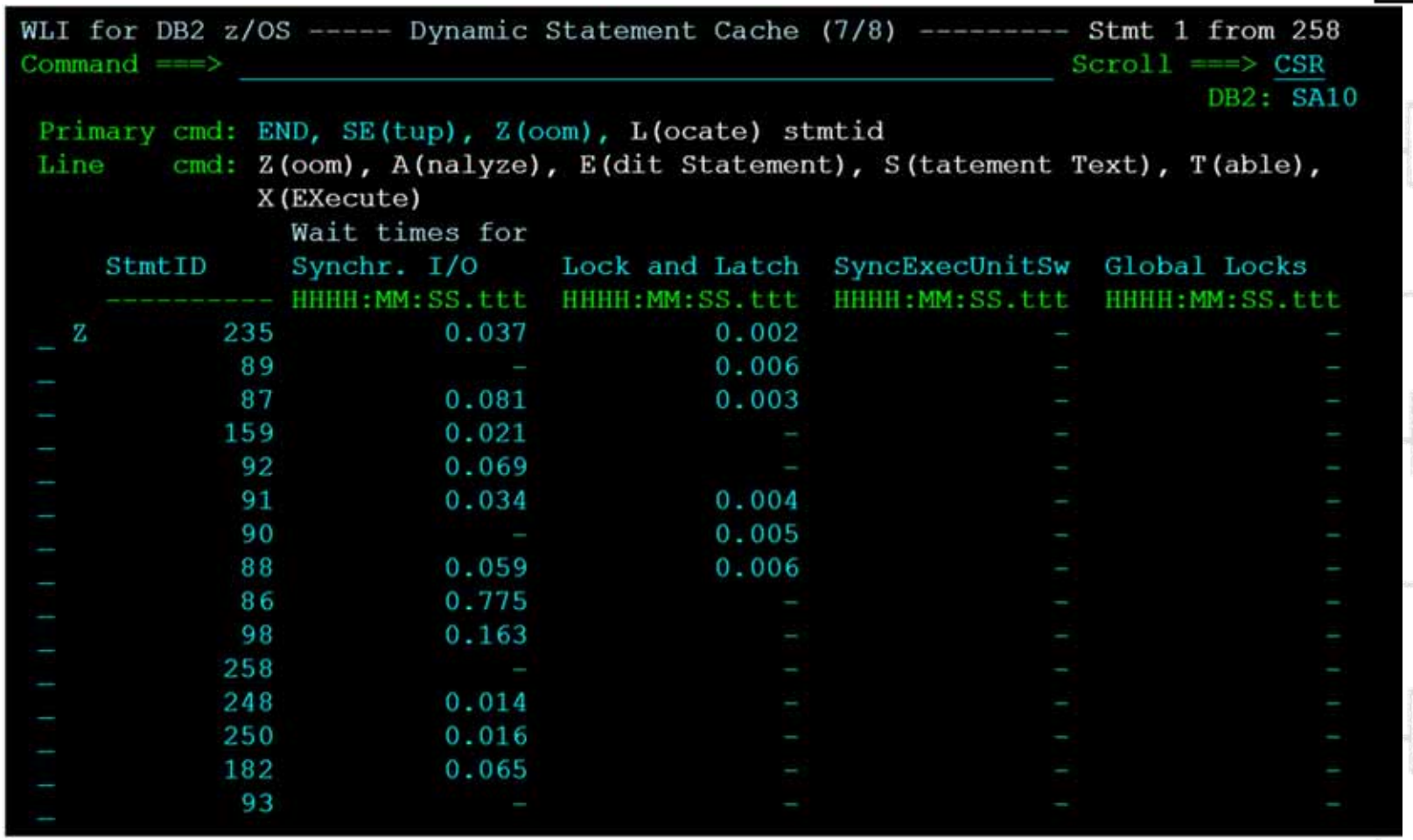
	StmtID	Users	Copies	Sorts	Parallel Groups	Norid Limits	Norid Storage
_ Z	235	0	0	66	0	0	0
_	89	0	0	1639	0	0	0
_	87	0	0	1644	0	0	0
_	159	0	0	5	0	0	0
_	92	0	0	0	0	0	0
_	91	0	0	0	0	0	0
_	90	0	0	0	0	0	0
_	88	0	0	0	0	0	0
_	86	0	0	221	0	0	0
_	98	0	0	4	0	0	0
_	258	0	0	2	0	0	0
_	248	0	0	1	0	0	0
_	250	0	0	1	0	0	0
_	182	0	0	114	0	0	0
_	93	0	0	0	0	0	0

DSC Tracing



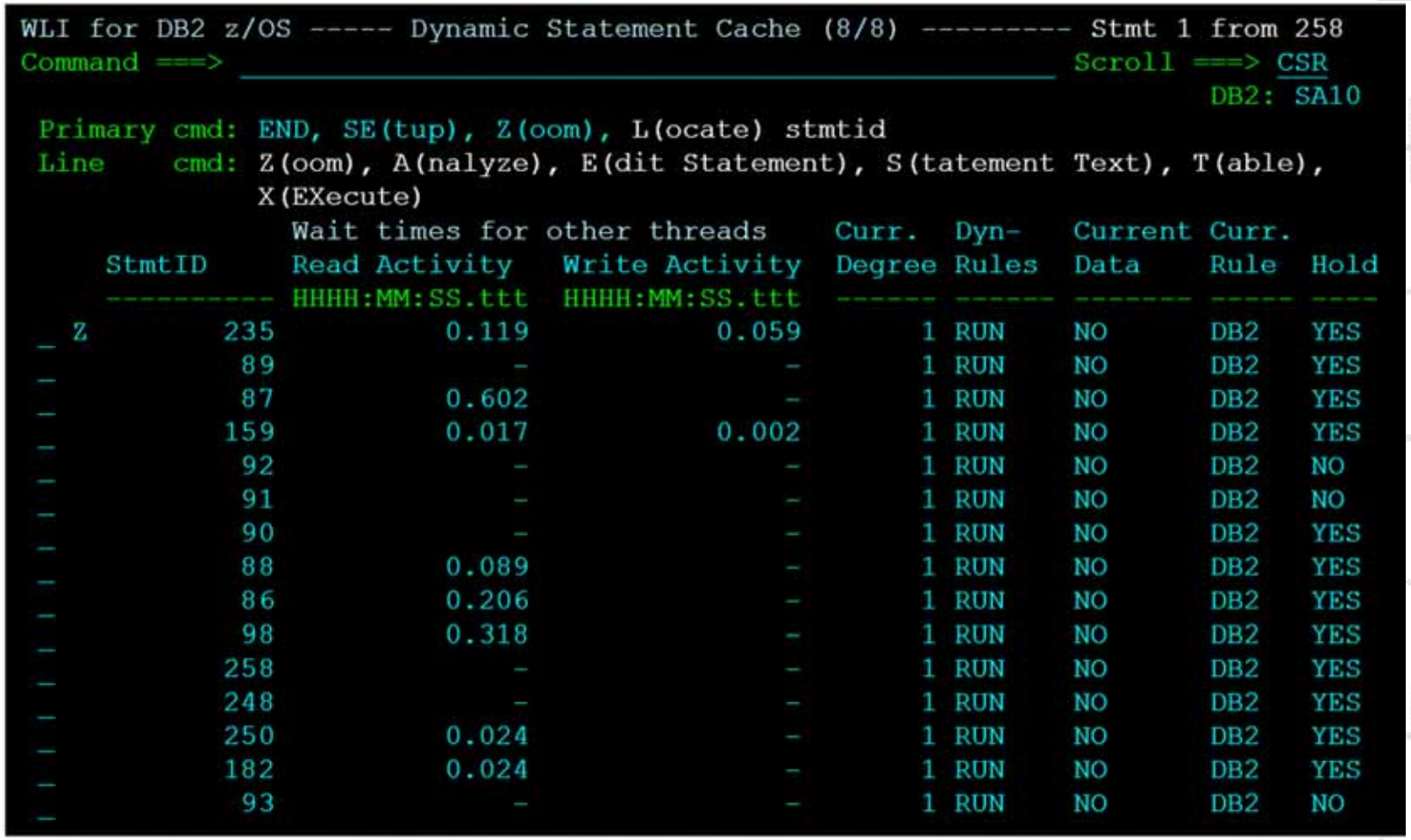
```
WLI for DB2 z/OS ----- Dynamic Statement Cache (6/8) --------- Stmt 1 from 258
Command ===> _____________________________________________ Scroll ===> CSR
                                                            DB2: SA10
 Primary cmd: END, SE(tup), Z(oom), L(ocate) stmtid
 Line    cmd: Z(oom), A(nalyze), E(dit Statement), S(tatement Text), T(able),
              X(EXecute)

     StmtID      Total CPU       Average CPU     Total Elapse    Average Elapse
     ---------- HHHH:MM:SS.ttt  HHHH:MM:SS.ttt  HHHH:MM:SS.ttt  HHHH:MM:SS.ttt
_ Z        235          1.665           0.151           2.244           0.204
_           89          5.416           0.003           5.812           0.004
_           87          5.598           0.003           7.192           0.004
_          159          1.639           0.328           1.774           0.355
_           92          1.251               -           1.685           0.001
_           91          1.835           0.001           2.167           0.001
_           90          0.945           0.001           1.022           0.001
_           88          1.290           0.001           1.758           0.001
_           86          1.617           0.007           2.769           0.013
_           98          0.801           0.801           1.774           1.774
_          258          0.585           0.293           0.643           0.322
_          248          0.300           0.300           0.330           0.330
_          250          0.264           0.264           0.329           0.329
_          182          0.469           0.004           0.732           0.006
_           93          0.278           0.001           0.318           0.001
```

DSC Tracing



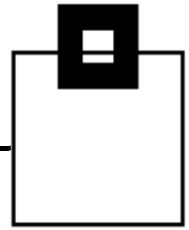
```
WLI for DB2 z/OS ----- Dynamic Statement Cache (7/8) --------- Stmt 1 from 258
Command ===> _____________________________________________ Scroll ===> CSR
                                                            DB2: SA10
 Primary cmd: END, SE(tup), Z(oom), L(ocate) stmtid
 Line    cmd: Z(oom), A(nalyze), E(dit Statement), S(tatement Text), T(able),
              X(EXecute)
                Wait times for
     StmtID     Synchr. I/O     Lock and Latch  SyncExecUnitSw  Global Locks
     ---------- HHHH:MM:SS.ttt  HHHH:MM:SS.ttt  HHHH:MM:SS.ttt  HHHH:MM:SS.ttt
 _ Z        235          0.037           0.002               -               -
 _           89              -           0.006               -               -
 _           87          0.081           0.003               -               -
 _          159          0.021               -               -               -
 _           92          0.069               -               -               -
 _           91          0.034           0.004               -               -
 _           90              -           0.005               -               -
 _           88          0.059           0.006               -               -
 _           86          0.775               -               -               -
 _           98          0.163               -               -               -
 _          258              -               -               -               -
 _          248          0.014               -               -               -
 _          250          0.016               -               -               -
 _          182          0.065               -               -               -
 _           93              -               -               -               -
```

DSC Tracing



```
WLI for DB2 z/OS ----- Dynamic Statement Cache (8/8) --------- Stmt 1 from 258
Command ===> _____________________________________________ Scroll ===> CSR
                                                            DB2: SA10
 Primary cmd: END, SE(tup), Z(oom), L(ocate) stmtid
 Line    cmd: Z(oom), A(nalyze), E(dit Statement), S(tatement Text), T(able),
              X(EXecute)
                Wait times for other threads    Curr.  Dyn-   Current Curr.
     StmtID     Read Activity   Write Activity  Degree Rules  Data    Rule  Hold
     ---------- HHHH:MM:SS.ttt  HHHH:MM:SS.ttt  ------ ------ ------- ----- ----
 _ Z        235          0.119           0.059       1 RUN    NO      DB2   YES
 _           89              -               -       1 RUN    NO      DB2   YES
 _           87          0.602               -       1 RUN    NO      DB2   YES
 _          159          0.017           0.002       1 RUN    NO      DB2   YES
 _           92              -               -       1 RUN    NO      DB2   NO
 _           91              -               -       1 RUN    NO      DB2   NO
 _           90              -               -       1 RUN    NO      DB2   YES
 _           88          0.089               -       1 RUN    NO      DB2   YES
 _           86          0.206               -       1 RUN    NO      DB2   YES
 _           98          0.318               -       1 RUN    NO      DB2   YES
 _          258              -               -       1 RUN    NO      DB2   YES
 _          248              -               -       1 RUN    NO      DB2   YES
 _          250          0.024               -       1 RUN    NO      DB2   YES
 _          182          0.024               -       1 RUN    NO      DB2   YES
 _           93              -               -       1 RUN    NO      DB2   NO
```

DSC Tracing



WOW!

This is great insight, without having to run a monitor,nor a trace!

BUT!

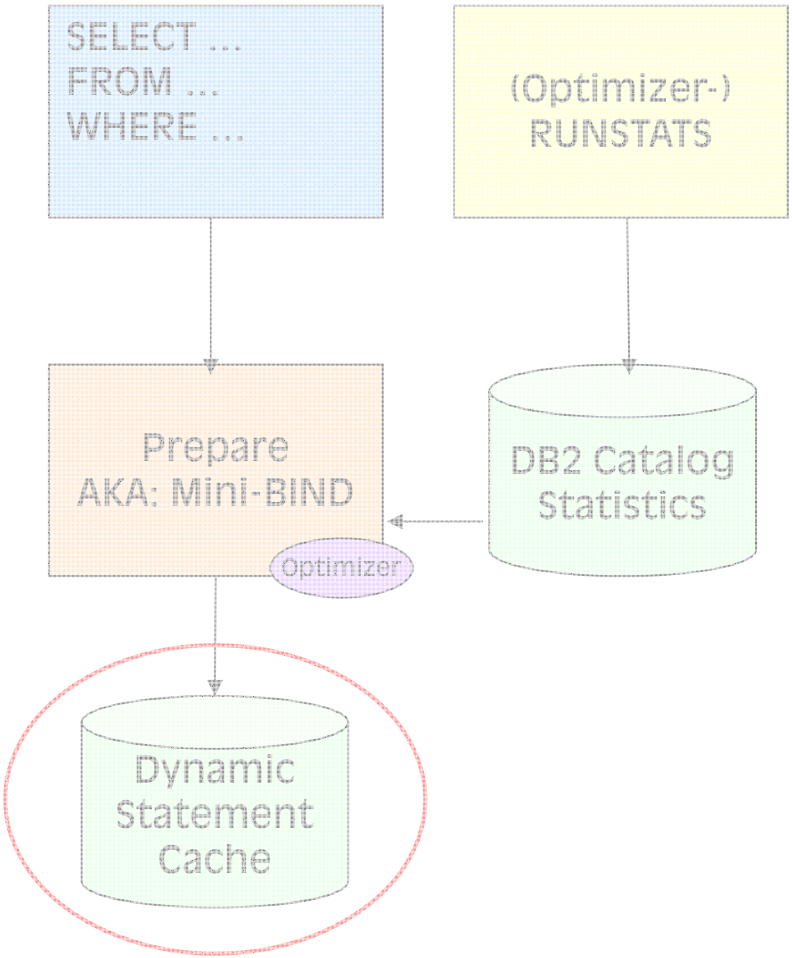
Does it really show ALL statements being executed?
Do you really get the full picture?

What about static SQL?

What was this flush thing from slide four?

DSC Tracing

Characteristics:



Access Paths for dynamic SQL are determined on the fly and stored in the DSC.

LRU, RUNSTATS, ALTER, DROP, REVOKE, DB2 RESTART invalidates and flushes the DSC for an object.

DSC Tracing

…How does the smarter database, how does DB2 10 for z/OS deal with it?…

New:

- IFCIDs 316 was enhanced to externalize the data from the DSC when a flushing situation occurs (LRU, RUNSTATs, ALTER, DROP, REVOKE, …)

SSC at a glance

→Start the new IFCIDs 400, 401 to access additional data* – let's call it the Static Statement Cache

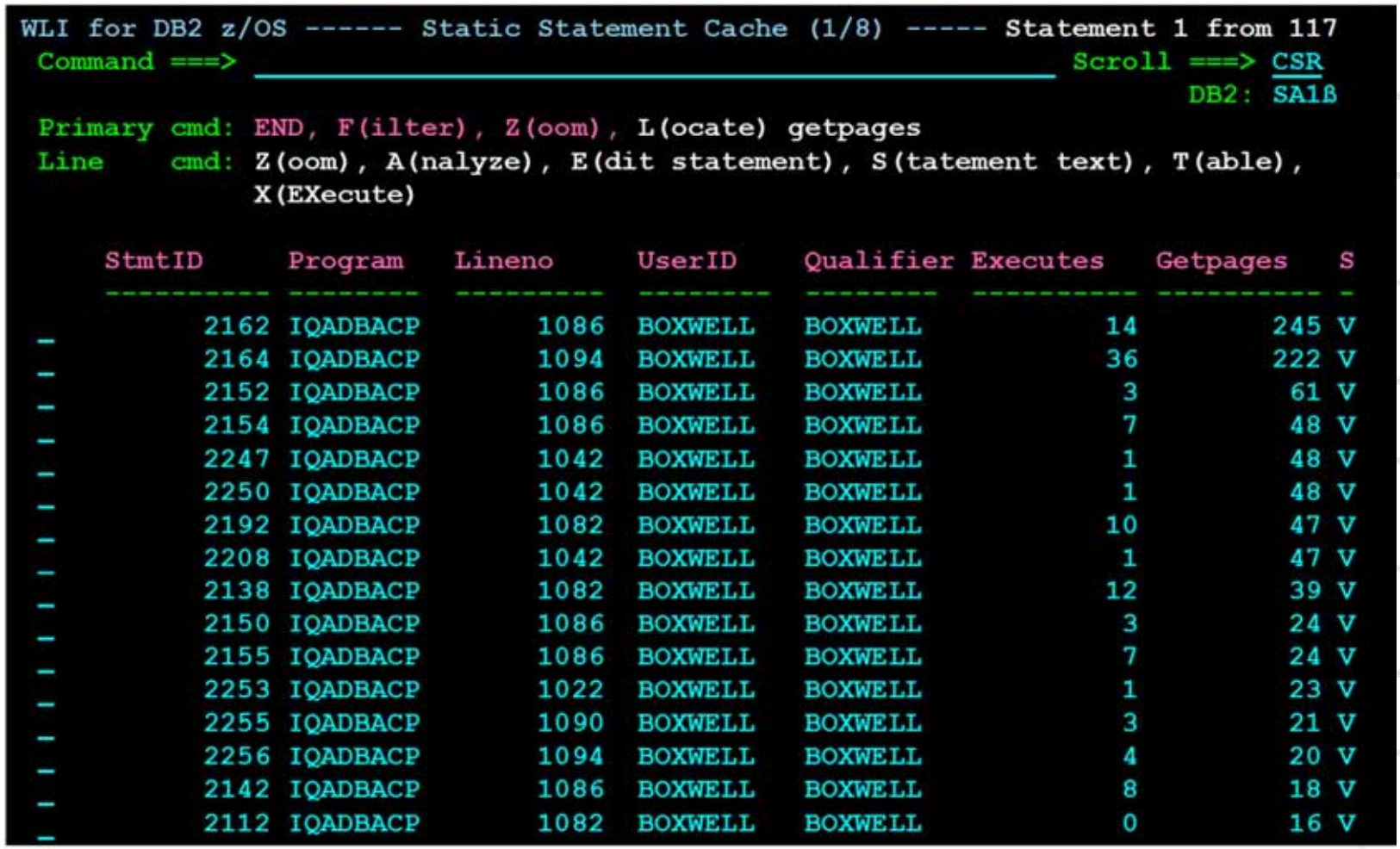
The SSC gives you the full picture:

- Memory resident storage of
- static SQL statements
- Statement ID
- Date/time, current status
- Resource consumption

Like with the enhanced 316, data is externalized when the memory is full. – NO DATA LOSS

*These two IFCIDs are not really IFCIDs but more „switches" to enable externalization of static SQL metrics

SSC at a glance



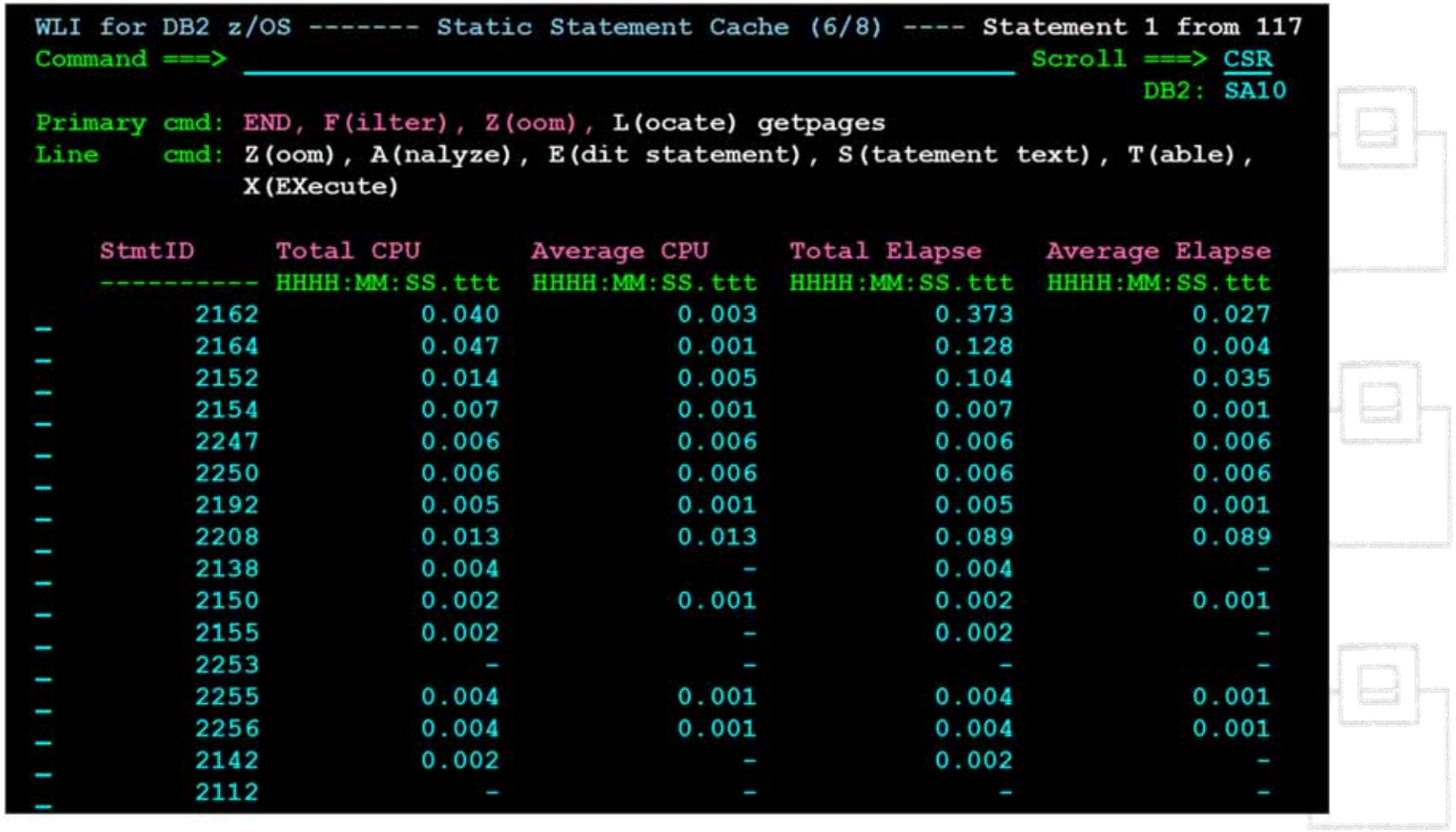
```
WLI for DB2 z/OS ------ Static Statement Cache (1/8) ----- Statement 1 from 117
 Command ===>                                                Scroll ===> CSR
                                                                 DB2: SA1ß

 Primary cmd: END, F(ilter), Z(oom), L(ocate) getpages
 Line     cmd: Z(oom), A(nalyze), E(dit statement), S(tatement text), T(able),
               X(EXecute)

      StmtID     Program   Lineno     UserID    Qualifier Executes   Getpages   S
      ---------- --------  ---------  --------  -------- ---------- ---------- -
_           2162 IQADBACP       1086  BOXWELL   BOXWELL          14        245 V
_           2164 IQADBACP       1094  BOXWELL   BOXWELL          36        222 V
_           2152 IQADBACP       1086  BOXWELL   BOXWELL           3         61 V
_           2154 IQADBACP       1086  BOXWELL   BOXWELL           7         48 V
_           2247 IQADBACP       1042  BOXWELL   BOXWELL           1         48 V
_           2250 IQADBACP       1042  BOXWELL   BOXWELL           1         48 V
_           2192 IQADBACP       1082  BOXWELL   BOXWELL          10         47 V
_           2208 IQADBACP       1042  BOXWELL   BOXWELL           1         47 V
_           2138 IQADBACP       1082  BOXWELL   BOXWELL          12         39 V
_           2150 IQADBACP       1086  BOXWELL   BOXWELL           3         24 V
_           2155 IQADBACP       1086  BOXWELL   BOXWELL           7         24 V
_           2253 IQADBACP       1022  BOXWELL   BOXWELL           1         23 V
_           2255 IQADBACP       1090  BOXWELL   BOXWELL           3         21 V
_           2256 IQADBACP       1094  BOXWELL   BOXWELL           4         20 V
_           2142 IQADBACP       1086  BOXWELL   BOXWELL           8         18 V
_           2112 IQADBACP       1082  BOXWELL   BOXWELL           0         16 V
```

SSC at a glance

```
WLI for DB2 z/OS ------ Static Statement Cache (4/8) ----- Statement 1 from 117
 Command ===> ________________________________________________ Scroll ===> CSR
                                                                      DB2: SA10

 Primary cmd: END, F(ilter), Z(oom), L(ocate) getpages
 Line    cmd: Z(oom), A(nalyze), E(dit statement), S(tatement text), T(able),
              X(EXecute)
```

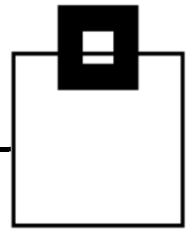
	StmtID	Synchr. Buffer Rd	Synchr. Buffer Wr	Rows examined	Rows processed	Index Scans	Tablespc. Scans
_	2162	0	0	74	37	52	15
_	2164	0	0	0	185	0	74
_	2152	4	0	38	19	30	4
_	2154	0	0	16	8	16	0
_	2247	0	0	101	2	2	1
_	2250	0	0	101	2	2	1
_	2192	0	0	844	2	4	11
_	2208	4	0	100	39	2	1
_	2138	0	0	13	13	13	0
_	2150	0	0	8	4	8	0
_	2155	0	0	8	0	8	0
_	2253	0	0	3	1	0	0
_	2255	0	0	0	7	3	3
_	2256	0	0	0	2	0	8
_	2142	0	0	0	9	0	9
_	2112	0	0	1	0	0	1

SSC at a glance



```
WLI for DB2 z/OS ------- Static Statement Cache (6/8) ---- Statement 1 from 117
Command ===> _____________________________________________ Scroll ===> CSR
                                                           DB2: SA10

Primary cmd: END, F(ilter), Z(oom), L(ocate) getpages
Line     cmd: Z(oom), A(nalyze), E(dit statement), S(tatement text), T(able),
              X(EXecute)

     StmtID      Total CPU       Average CPU     Total Elapse    Average Elapse
     ---------- HHHH:MM:SS.ttt  HHHH:MM:SS.ttt  HHHH:MM:SS.ttt  HHHH:MM:SS.ttt
_          2162          0.040           0.003           0.373           0.027
_          2164          0.047           0.001           0.128           0.004
_          2152          0.014           0.005           0.104           0.035
_          2154          0.007           0.001           0.007           0.001
_          2247          0.006           0.006           0.006           0.006
_          2250          0.006           0.006           0.006           0.006
_          2192          0.005           0.001           0.005           0.001
_          2208          0.013           0.013           0.089           0.089
_          2138          0.004               -           0.004               -
_          2150          0.002           0.001           0.002           0.001
_          2155          0.002               -           0.002               -
_          2253              -               -               -               -
_          2255          0.004           0.001           0.004           0.001
_          2256          0.004           0.001           0.004           0.001
_          2142          0.002               -           0.002               -
_          2112              -               -               -               -
```

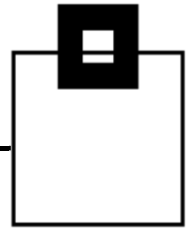
SSC at a glance



DB2 10 also provides some additional information from The DSC trace we all know today:

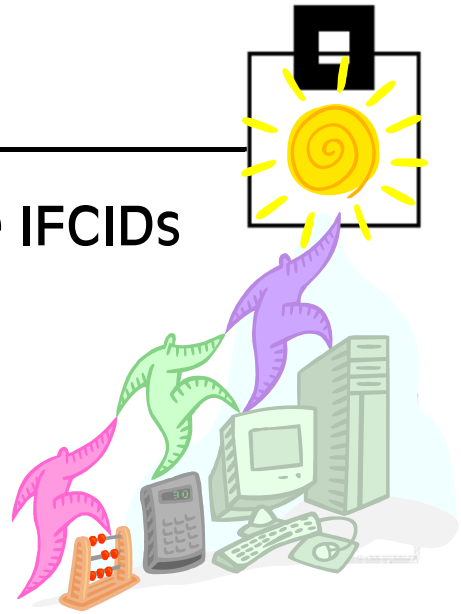
- Wait time accumulation for
  - Latch requests
  - Page latches
  - Drain locks
  - Drains during waits for claims to be released
  - Log writers

SSC at a glance



- Date and time in store clock format for Stmt insertion and update (along with internal format)
- Number of times that
  - a RID list overflowed because of
    - storage shortage
    - # of RIDs exceeded internal limit(s)
  - a RID list append for a hybrid join interrupted
    - because of RID pool storage shortage
    - # of RIDs exceeded internal limit(s)
  - a RID list retrieval failed for multiple IX access. The result of IX AND/Oring could not be determined

How to exploit DSC & SSC



Have a started task running 24x7 catching all the IFCIDs that DB2 will be throwing and stores the data.

Allow to exclude yourself from the system.

Externalize the data like every 60 min, or allow
-customization (e.g. 30 - 180 minutes
-ad hoc data refresh

How to exploit DSC & SSC

Triggered at the interval, the data captured needs to be processed locally, or at a different site:

Allow two levels of details for static SQL

- Full - Capture the SQL Text at trace time
- Ref - Use the SYSPACKSTMT table for the SQL Text

Referring to the Catalog should be the efficient default. However, in case of a very dynamic environment the SQL text needs to be captured real-time to avoid missing SQL text situations when using history and the original SQL isn't there anymore (e.g. a new pre-compile leads to a new STMTNO) .

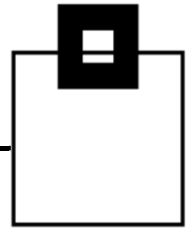
How to exploit DSC & SSC

Resulting benefits:

- Executed SQL in a plex report
- Workload and Performance warehouse repository that contains *all* executed SQL
- Threshold based warning system for
  - Exceeded resource limits
  - Unusual workload
- Powerful history and trending analysis

All of this is now available with the smallest ever possible overhead!

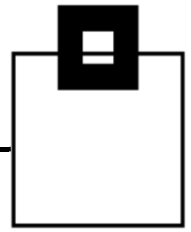
How to exploit it



Real world scenarios:

Scenario 1: Heavy hitters
Scenario 2: Executed SQL by program
Scenario 3: Objects used in SQL
Scenario 4: Never executed SQL
Scenario 5: RID Pool control
Auditor Scenario 1: Who did what?
Auditor Scenario 2: When did it happen?

Scenario 1: Heavy hitters



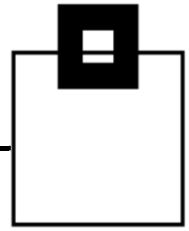
All SQL that uses over a threshold of CPU
All SQL that scans over a threshold of Getpages
All SQL that waits too long
Etc etc

History, Drill-down
Static and Dynamic

Scenario 1: Heavy hitters

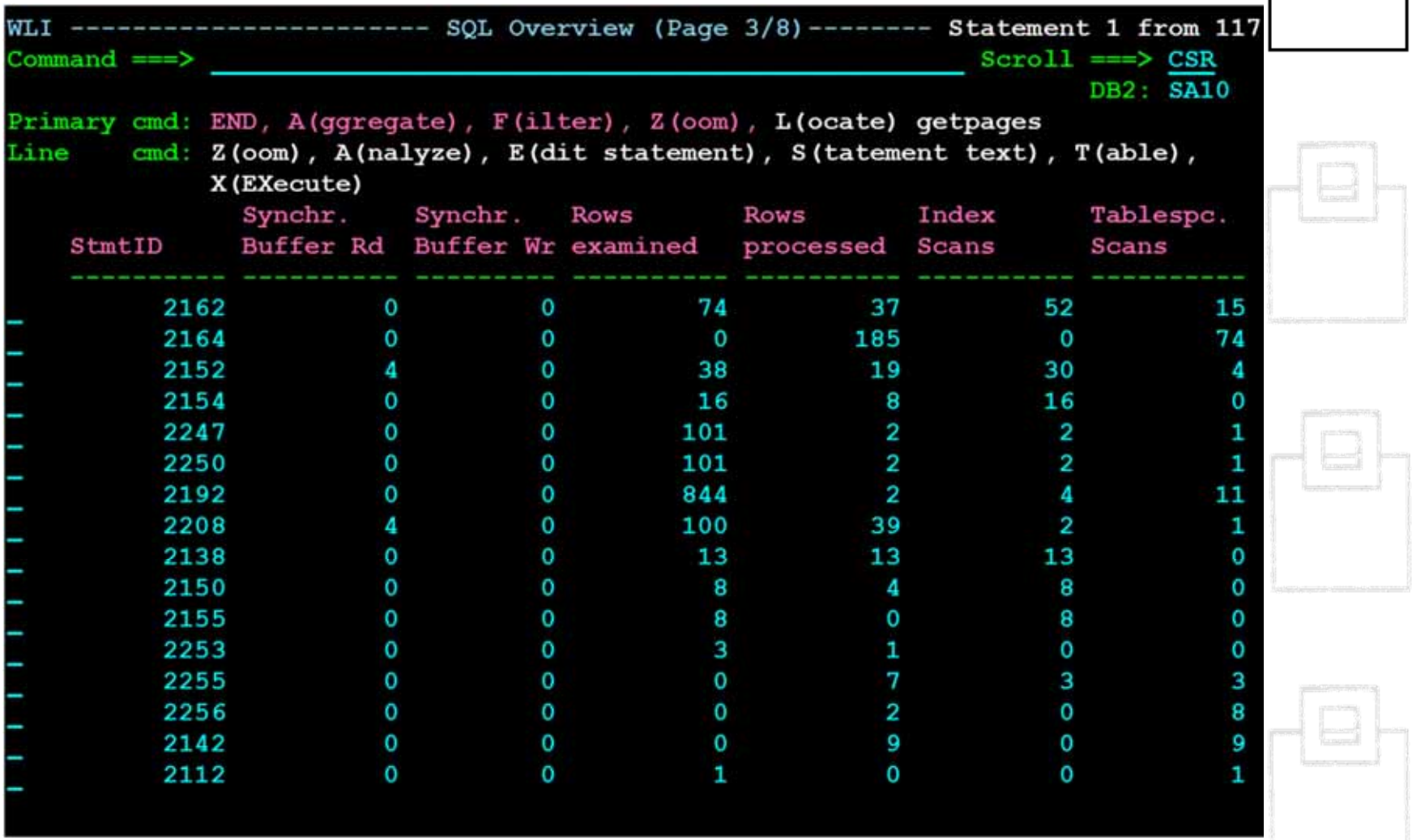
Scenario 1: Heavy hitters – data for SSC

Scenario 1: Heavy hitters




```
WLI ---------------------- Filter ----------------------------------------
Command ===> _____________________________________________ DB2: SA10

DYNAMIC        : _
STATIC         : X    COLLID: _________________  PACKAGE: ________

FROM           : ____________________________
TO             : ____________________________

(FIRST) TABLE  : ____________________________________________________
CREATOR          ____________________________________________________

(FIRST) TABLE  : ____________________________________________________
NAME             ____________________________________________________

QUALIFIER      : ____________________________________________________
                 ____________________________________________________

PRIMARY        : ____________________________________________________
AUTHID           ____________________________________________________

SELECT  X      CURRENT USERS    between _________ and _________ (Integer)
INSERT  X      STMT COUNT       between _________ and _________ (Integer)
UPDATE  X      AVG CPU TIME     between _________ and _________ (MM:SS.TTT)
DELETE  X      AVG ELAPSE TIME  between _________ and _________ (MM:SS.TTT)
               AVG GETPAGES     between _________ and _________ (Integer)

OUTPUT LIMIT   : 10000   0 - 25000 Max number of statements to be displayed
```

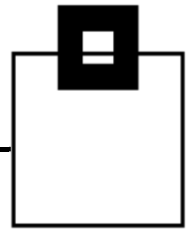
Scenario 1: Heavy hitters



```
WLI ---------------------- SQL Overview (Page 3/8)-------- Statement 1 from 117
Command ===> _____________________________________________ Scroll ===> CSR
                                                           DB2: SA10
Primary cmd: END, A(ggregate), F(ilter), Z(oom), L(ocate) getpages
Line    cmd: Z(oom), A(nalyze), E(dit statement), S(tatement text), T(able),
             X(EXecute)
```

	StmtID	Synchr. Buffer Rd	Synchr. Buffer Wr	Rows examined	Rows processed	Index Scans	Tablespc. Scans
_	2162	0	0	74	37	52	15
_	2164	0	0	0	185	0	74
_	2152	4	0	38	19	30	4
_	2154	0	0	16	8	16	0
_	2247	0	0	101	2	2	1
_	2250	0	0	101	2	2	1
_	2192	0	0	844	2	4	11
_	2208	4	0	100	39	2	1
_	2138	0	0	13	13	13	0
_	2150	0	0	8	4	8	0
_	2155	0	0	8	0	8	0
_	2253	0	0	3	1	0	0
_	2255	0	0	0	7	3	3
_	2256	0	0	0	2	0	8
_	2142	0	0	0	9	0	9
_	2112	0	0	1	0	0	1

Scenario 2: Executed SQL by program



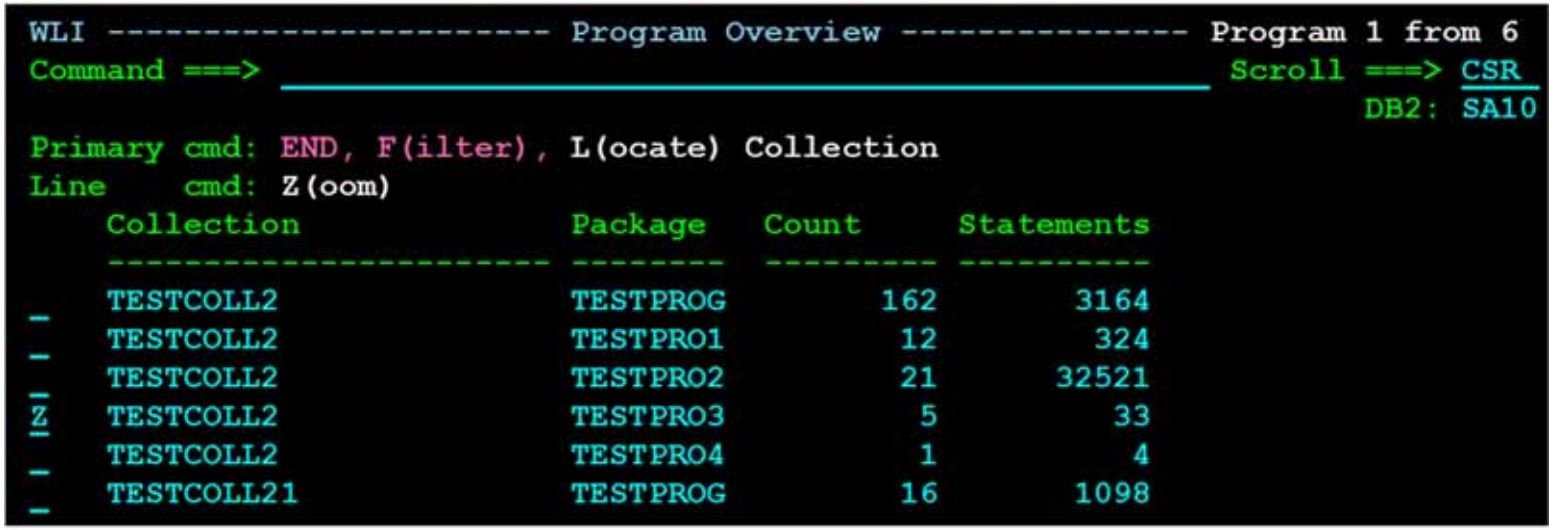
All SQL in programs with usage stats

History, Drill-down
Static

Scenario 2: Executed SQL by program

Scenario 2: Executed SQL by program

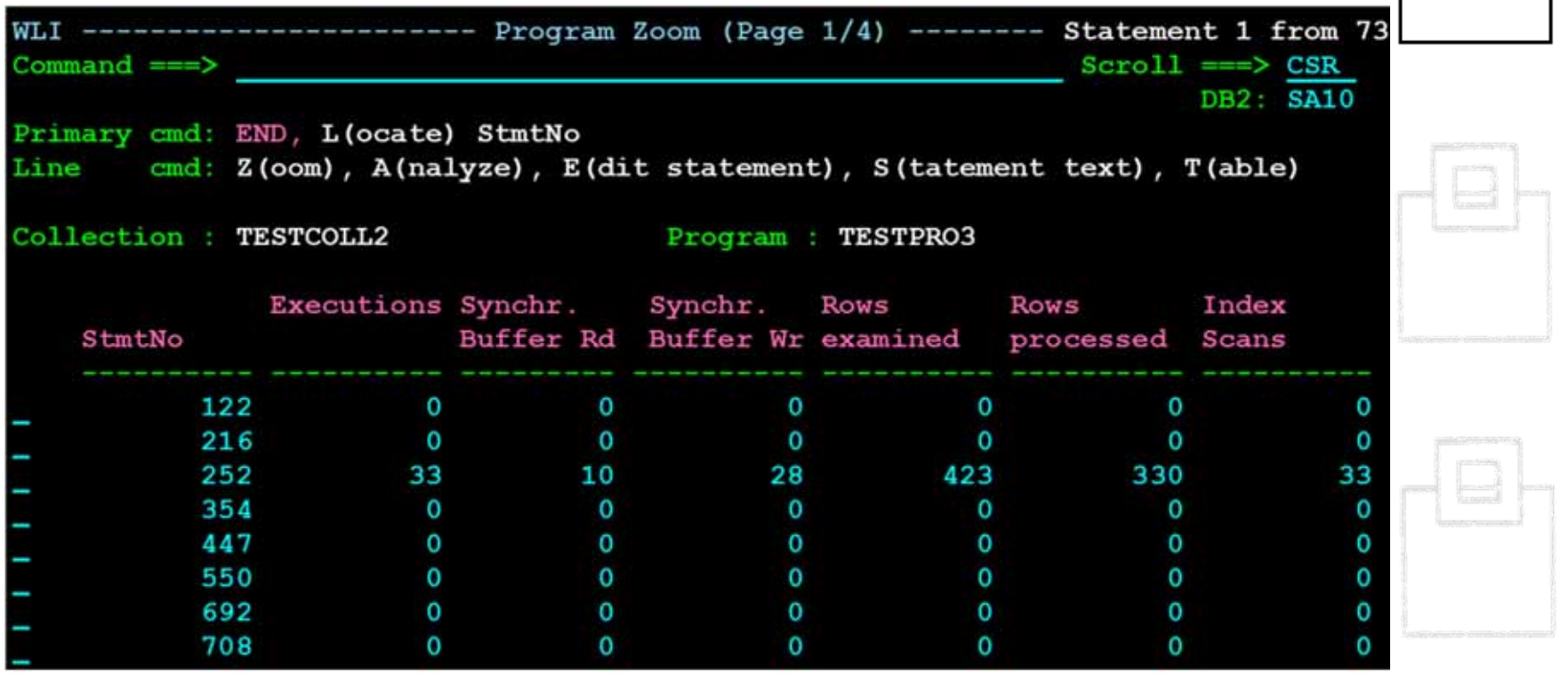
Scenario 2: Executed SQL by program



Here you can see the “Count” column

This gives the correct sequence in the BIND statement depending on the execution rate – Another bonus!

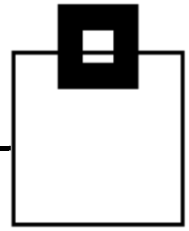
Scenario 2: Executed SQL by program



StmtNo	Executions	Synchr. Buffer Rd	Synchr. Buffer Wr	Rows examined	Rows processed	Index Scans
122	0	0	0	0	0	0
216	0	0	0	0	0	0
252	33	10	28	423	330	33
354	0	0	0	0	0	0
447	0	0	0	0	0	0
550	0	0	0	0	0	0
692	0	0	0	0	0	0
708	0	0	0	0	0	0

Again lots of data “off to the right”

Scenario 3: Objects used in SQL



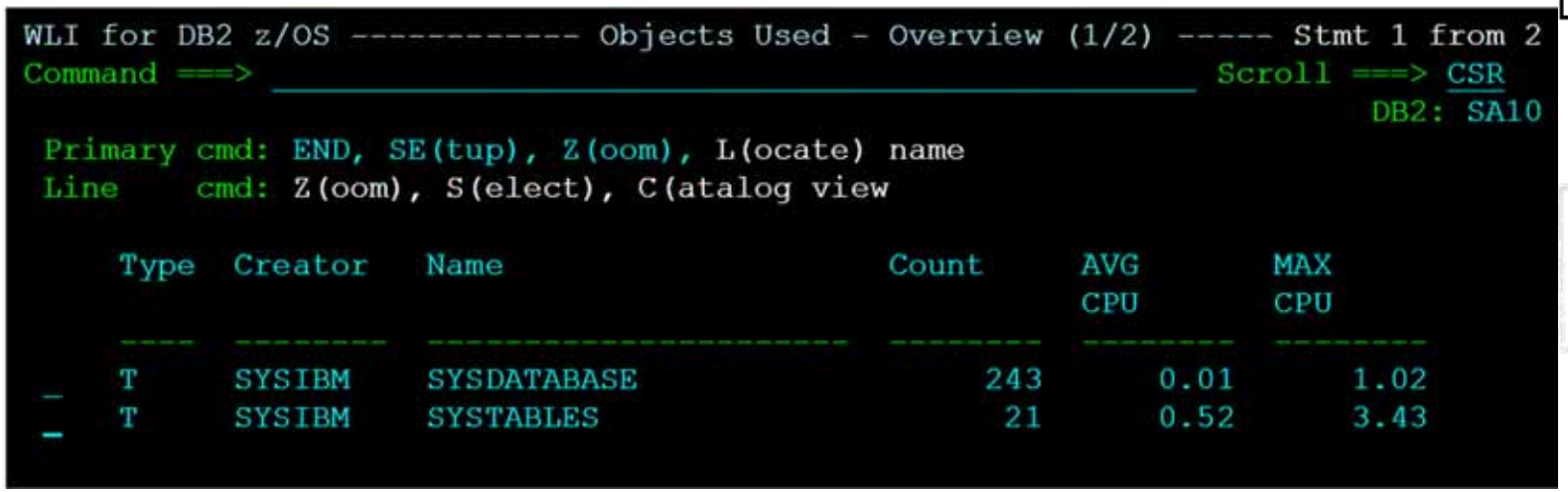
All SQL in programs with usage stats

If the SQL text has been processed you can then see what objects are being used in your system. Of course what is even better is to see those objects *not* being used!

History, Drill-down
Static and Dynamic

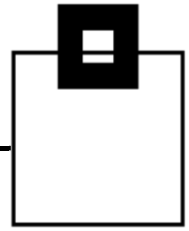
Scenario 3: Objects used in SQL

Scenario 3: Objects used in SQL



Here you can Zoom to the catalog object details or Select down to detail level to review SQL text and then link across to our Analyse+ tool.

Scenario 4: Never executed SQL



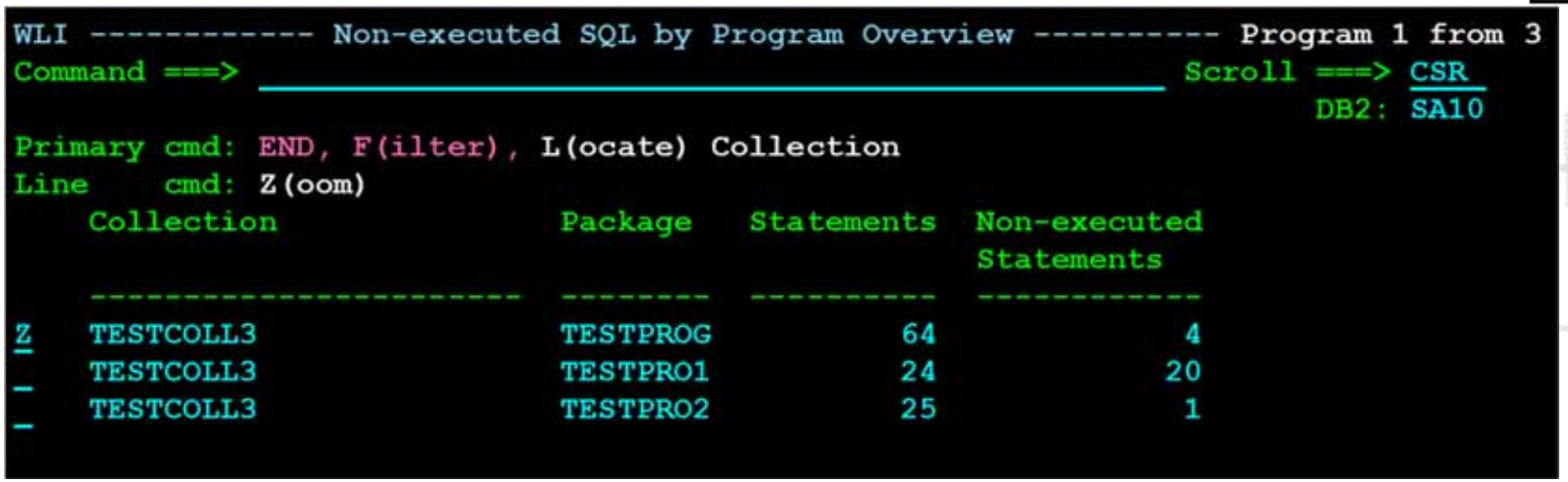
All SQL in programs that has *never* been executed!
Think of the possibilities here…
How often have you tuned and tuned a SQL to then find out that it is never actually executed?

History, Drill-down
Static

Scenario 4: Never executed SQL

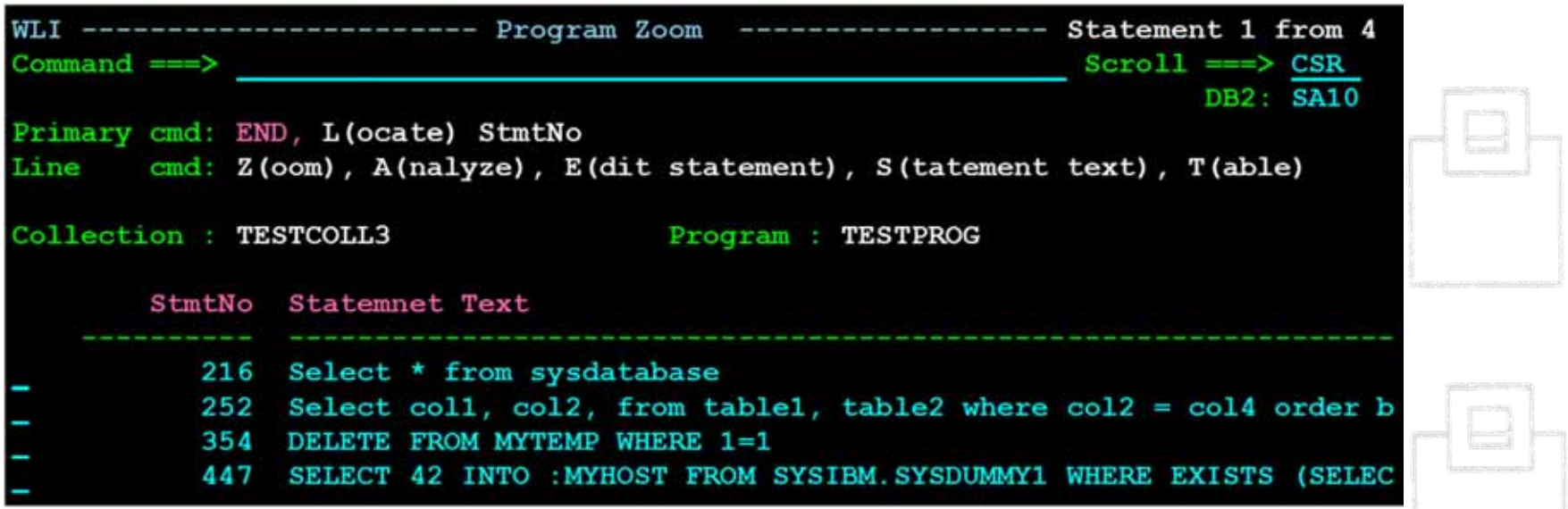
Scenario 4: Never executed SQL

Scenario 4: Never executed SQL

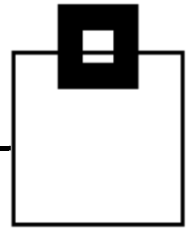


Here you can see the never executed statements in a given package

Scenario 4: Never executed SQL



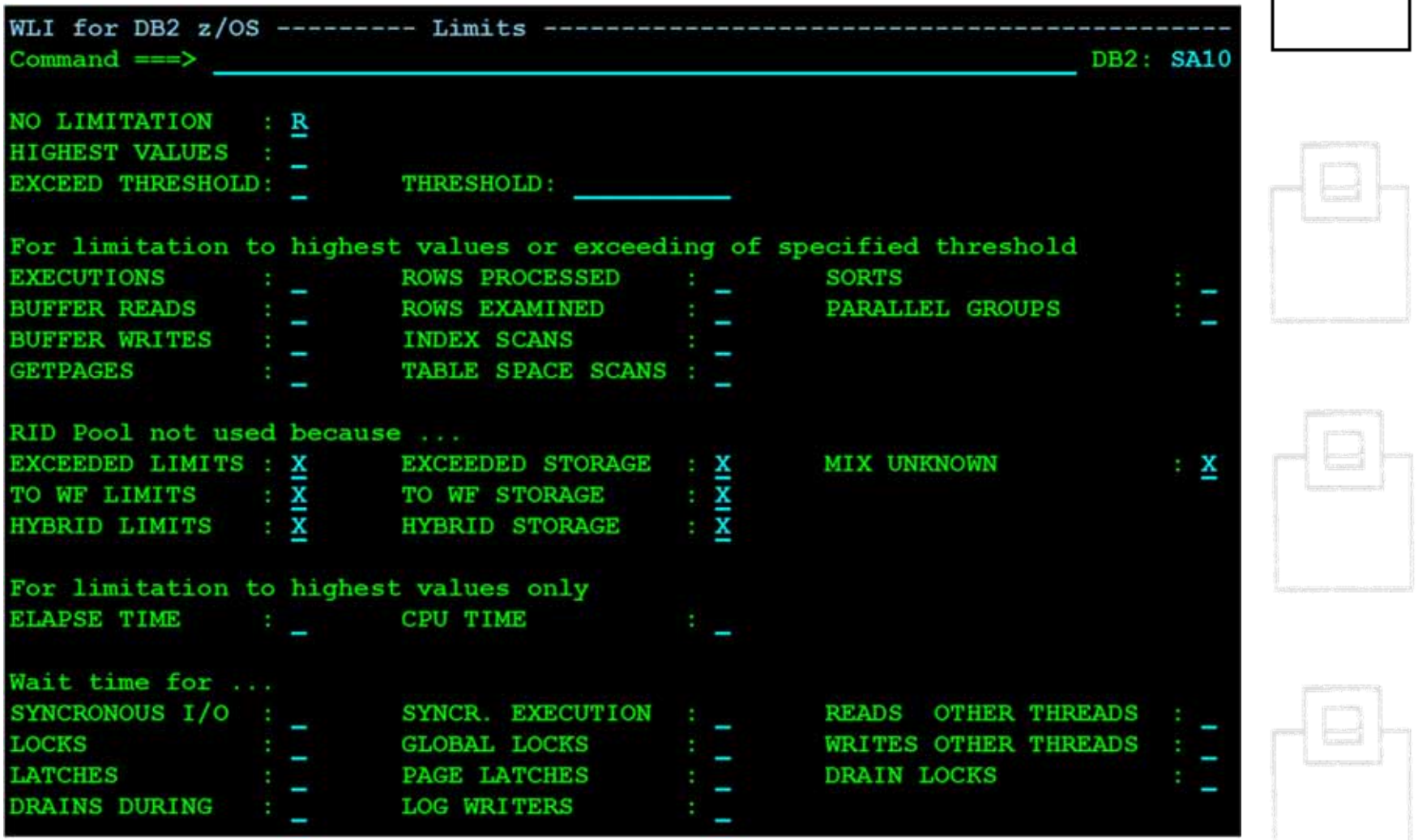
Scenario 5: RID Pool control



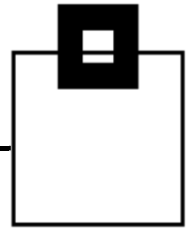
Where am I getting RID overflow or failure?
Can I solve it with better statistics and a REBIND?

History, Drill-down
Static and Dynamic

Scenario 5: RID Pool control

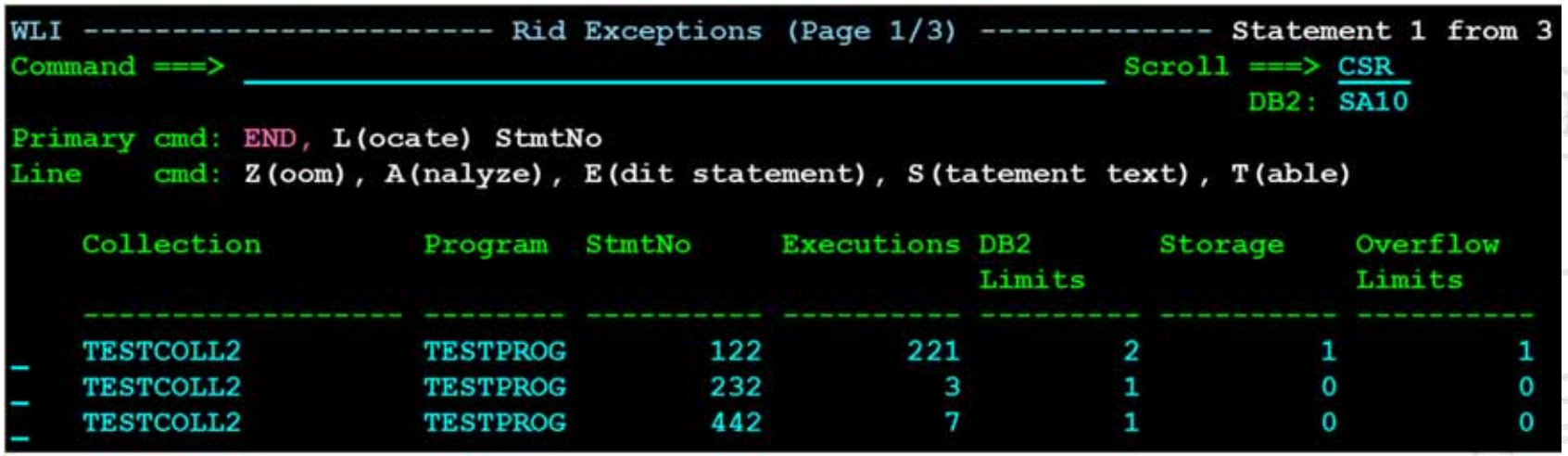


Scenario 5: RID Pool control



```
WLI ---------------------- Filter ----------------------------------------
Command ===>                                                  DB2: SA10

DYNAMIC         : X
STATIC          : X    COLLID:                PACKAGE:

FROM            :
TO              :

(FIRST) TABLE   :
CREATOR

(FIRST) TABLE   :
NAME

QUALIFIER       :


PRIMARY         :
AUTHID

SELECT  X       CURRENT USERS    between          and          (Integer)
INSERT  X       STMT COUNT       between          and          (Integer)
UPDATE  X       AVG CPU TIME     between          and          (MM:SS.TTT)
DELETE  X       AVG ELAPSE TIME  between          and          (MM:SS.TTT)
                AVG GETPAGES     between          and          (Integer)

OUTPUT LIMIT    : 10000    0 - 25000 Max number of statements to be displayed
```

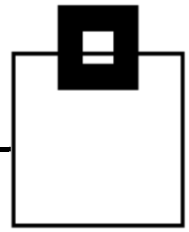
Scenario 5: RID Pool control



```
WLI ---------------------- Rid Exceptions (Page 1/3) ------------- Statement 1 from 3
Command ===> _____________________________________________ Scroll ===> CSR
                                                            DB2: SA10

Primary cmd: END, L(ocate) StmtNo
Line    cmd: Z(oom), A(nalyze), E(dit statement), S(tatement text), T(able)

    Collection          Program  StmtNo     Executions DB2        Storage    Overflow
                                                       Limits                 Limits
    ----------------- -------- ---------- ---------- --------- ---------- ----------
_   TESTCOLL2           TESTPROG        122        221         2          1          1
_   TESTCOLL2           TESTPROG        232          3         1          0          0
_   TESTCOLL2           TESTPROG        442          7         1          0          0
```

Again lots of data “off to the right”

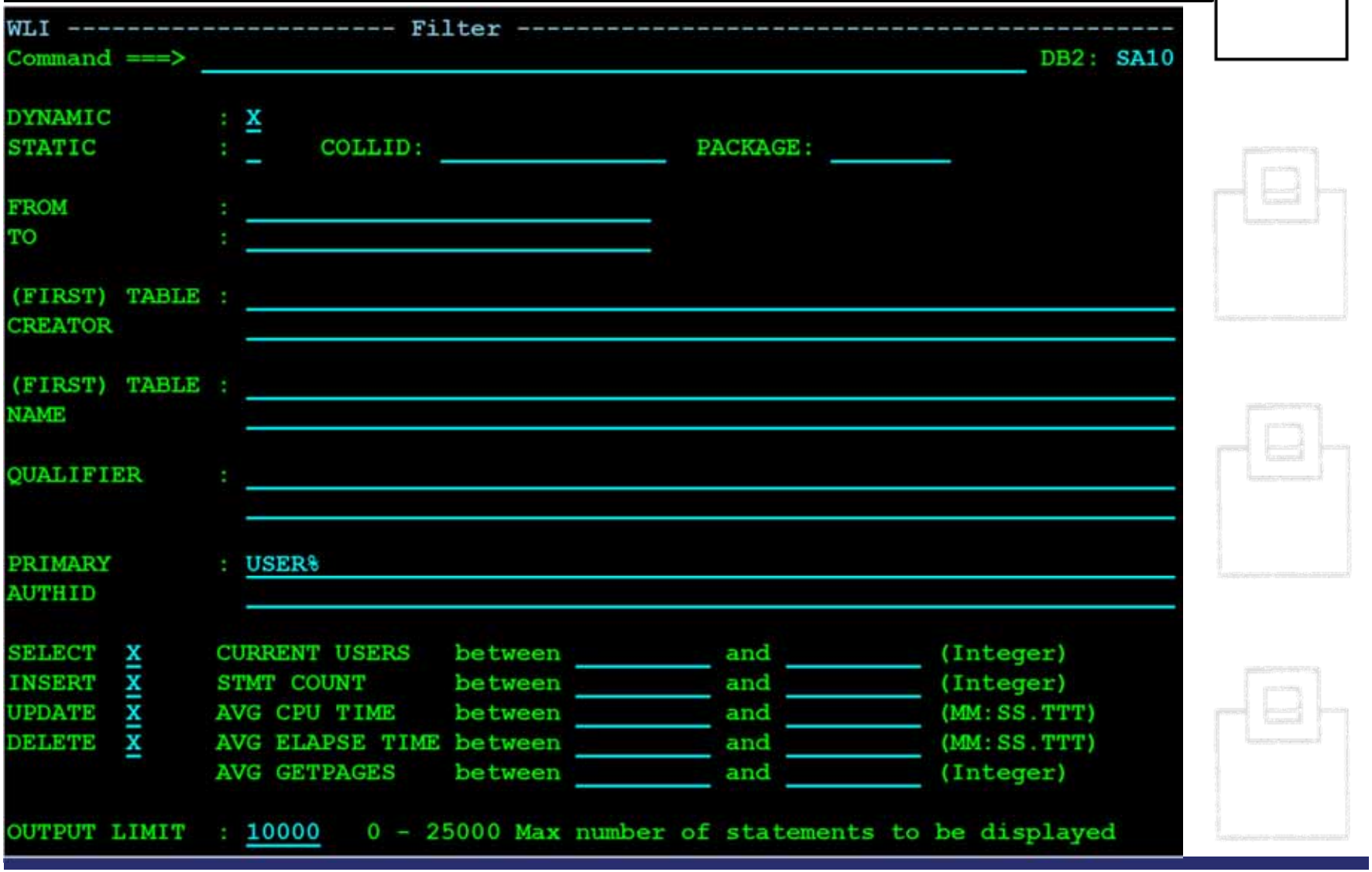
Auditor Scenario 1: Who did what?



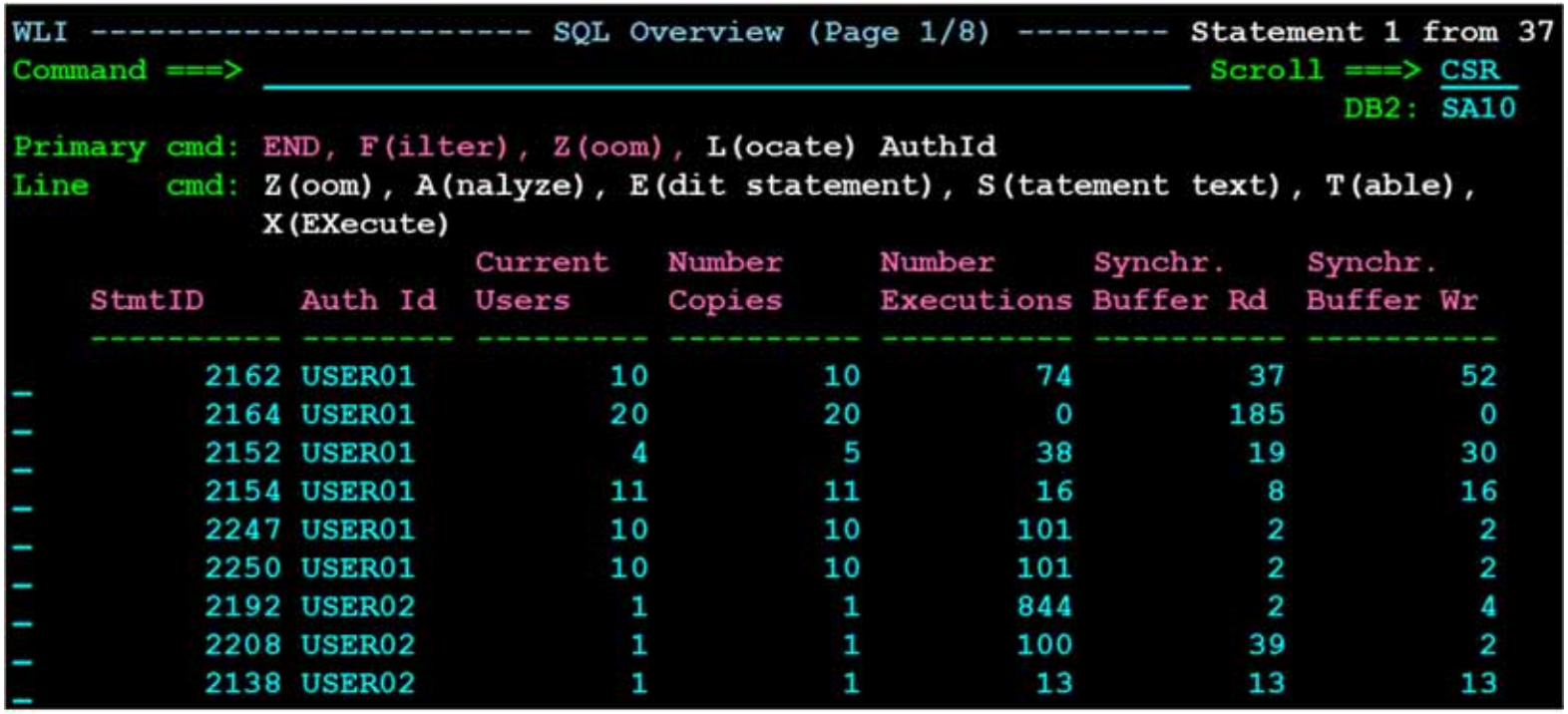
All executed SQL for a given Authorization

History, Drill-down
Dynamic

Auditor Scenario 1: Who did what?

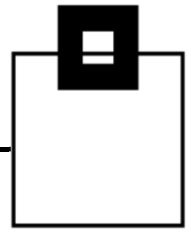


Auditor Scenario 1: Who did what?



```
WLI ---------------------- SQL Overview (Page 1/8) -------- Statement 1 from 37
Command ===> _____________________________________________ Scroll ===> CSR
                                                            DB2: SA10
Primary cmd: END, F(ilter), Z(oom), L(ocate) AuthId
Line    cmd: Z(oom), A(nalyze), E(dit statement), S(tatement text), T(able),
             X(EXecute)
                          Current   Number     Number     Synchr.    Synchr.
    StmtID     Auth Id    Users     Copies     Executions Buffer Rd  Buffer Wr
    ---------- -------- --------- ---------- ---------- ---------- ----------
_         2162 USER01          10         10         74         37         52
_         2164 USER01          20         20          0        185          0
_         2152 USER01           4          5         38         19         30
_         2154 USER01          11         11         16          8         16
_         2247 USER01          10         10        101          2          2
_         2250 USER01          10         10        101          2          2
_         2192 USER02           1          1        844          2          4
_         2208 USER02           1          1        100         39          2
_         2138 USER02           1          1         13         13         13
```

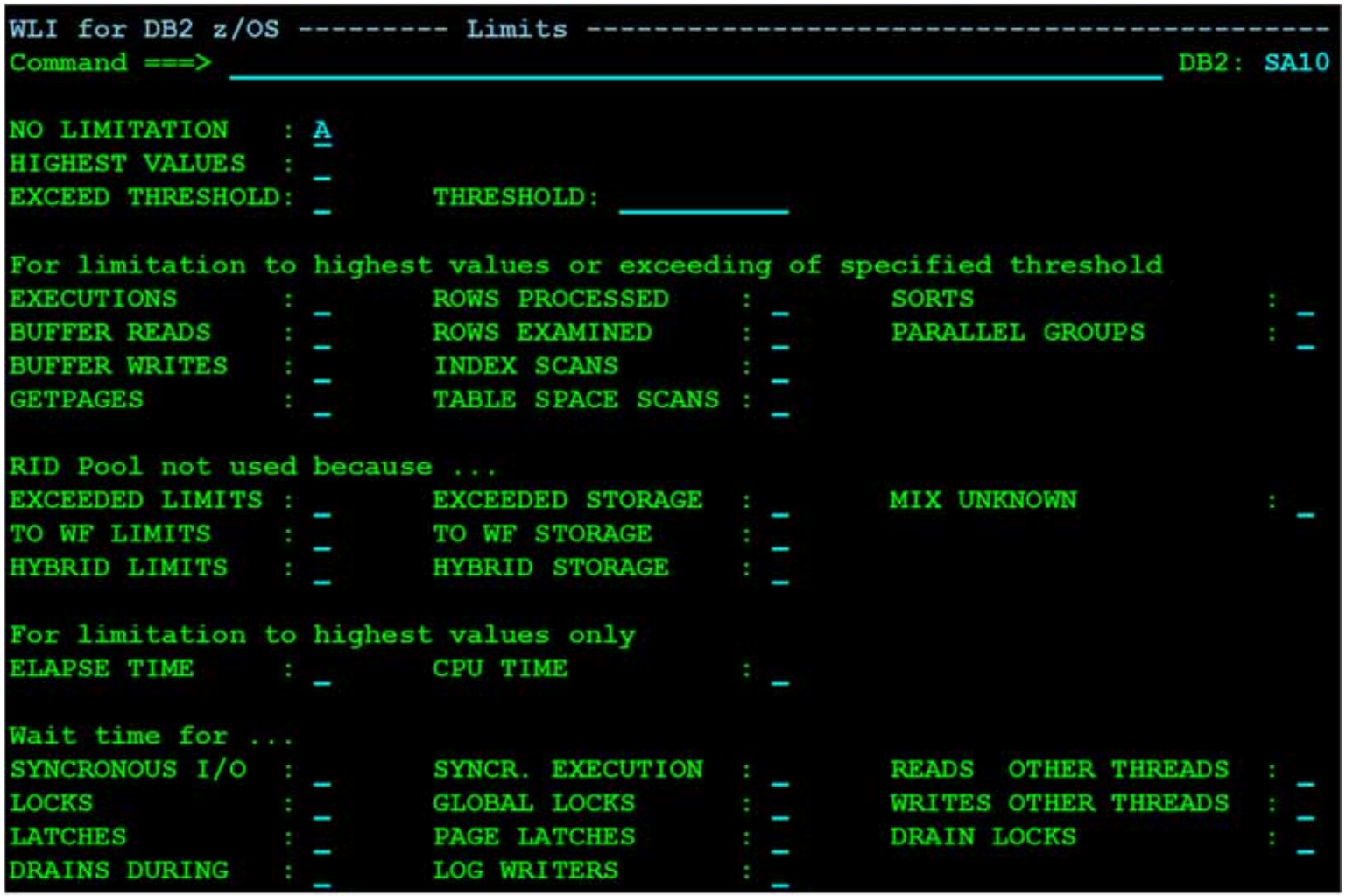
Auditor Scenario 2: When did it happen?



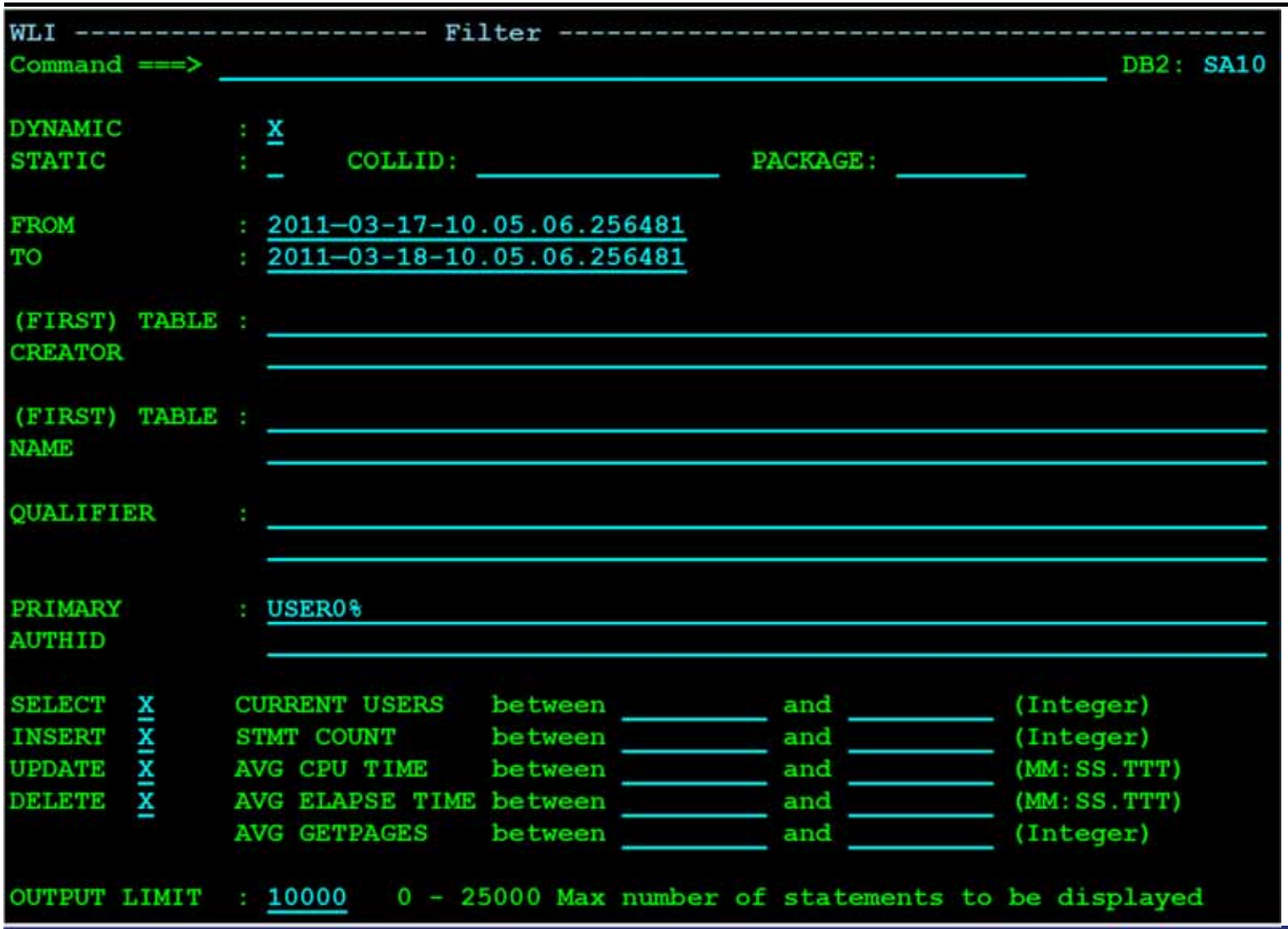
All executed SQL from a given (1st) time

History, Drill-down
Static and Dynamic

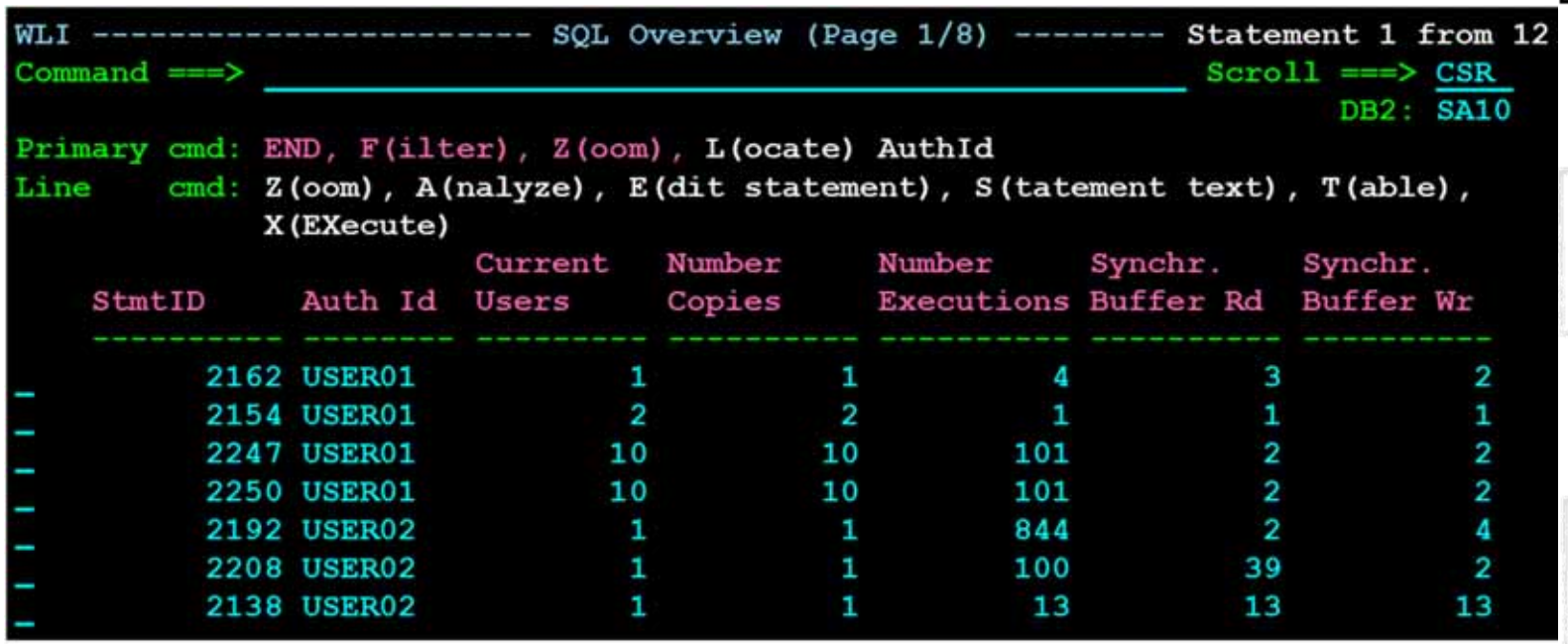
Auditor Scenario 2: When did it happen?



Auditor Scenario 2: When did it happen?



Auditor Scenario 2: When did it happen?



```
WLI ---------------------- SQL Overview (Page 1/8) -------- Statement 1 from 12
Command ===> _____________________________________________ Scroll ===> CSR
                                                            DB2: SA10

Primary cmd: END, F(ilter), Z(oom), L(ocate) AuthId
Line    cmd: Z(oom), A(nalyze), E(dit statement), S(tatement text), T(able),
             X(EXecute)
                          Current   Number     Number     Synchr.    Synchr.
   StmtID     Auth Id     Users     Copies     Executions Buffer Rd  Buffer Wr
   ---------- -------- --------- ---------- ---------- ---------- ----------
_        2162 USER01            1          1          4          3          2
_        2154 USER01            2          2          1          1          1
_        2247 USER01           10         10        101          2          2
_        2250 USER01           10         10        101          2          2
_        2192 USER02            1          1        844          2          4
_        2208 USER02            1          1        100         39          2
_        2138 USER02            1          1         13         13         13
```

Timestamps and other data off to the right remember!

Conclusion

- The DSC continues to increase DB2 efficiency
- An additional enhancement “SSC” closes the gap of out-of-the-box workload data
- DSC and SSC provide detailed production workload information ready-to-go