bmcsoftware

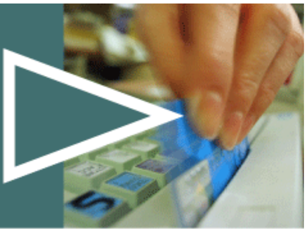
ACTIVATE BUSINESS WITH THE POWER OF I.T.™

Demystifying the DB2 Dynamic Statement Cache

For the Atlanta DB2 Users Group – March 2008

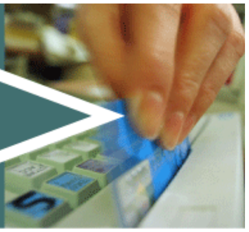
6/16/2008

What Will We Talk About?

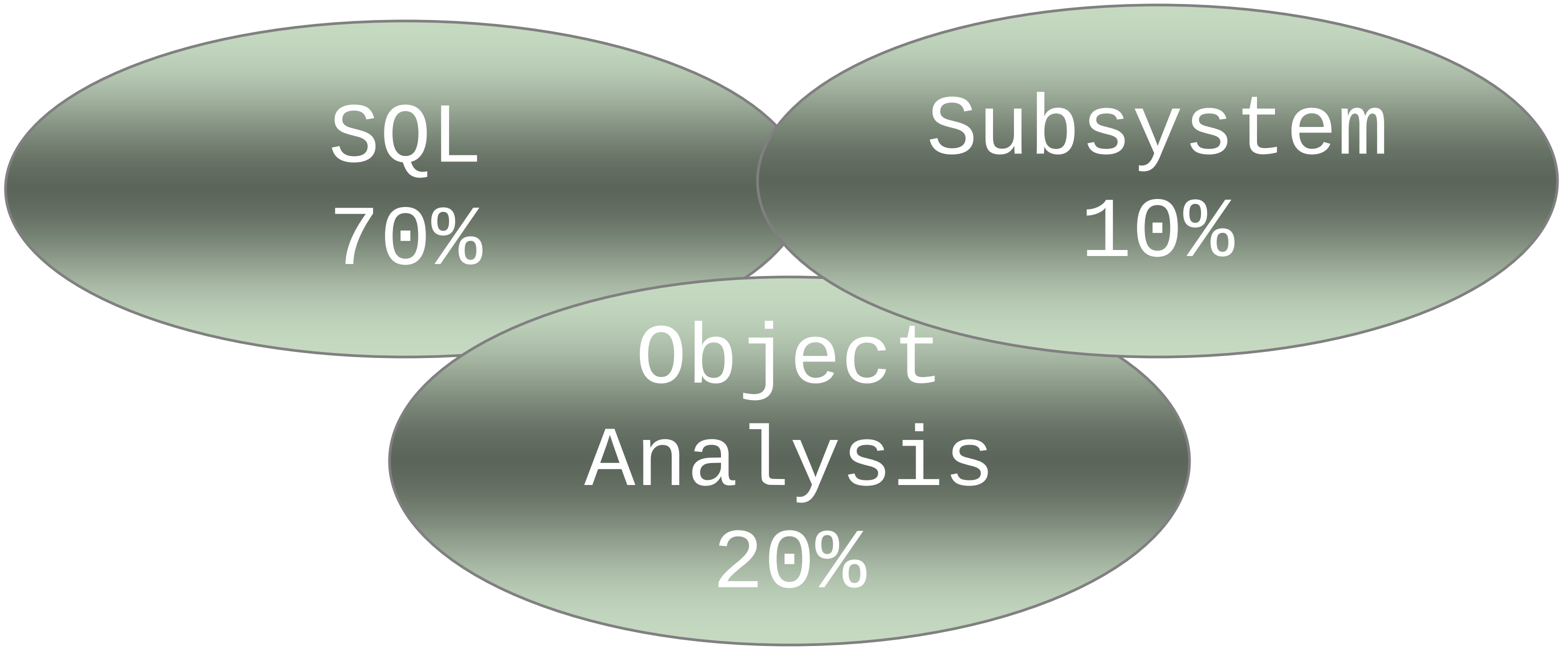


- Some SQL Tuning Fundamentals
- Dynamic SQL in More Detail
- Introduction to DB2 Statement Caching
- Mining for Gold in the Global Statement Cache

DB2 Tuning - Where Should You Spend Your Time



- What Can I tune in DB2



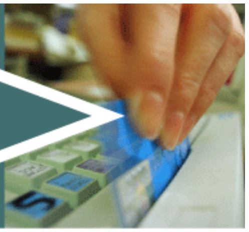
SQL
70%

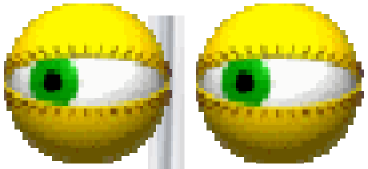
Subsystem
10%

Object
Analysis
20%

- Where are the biggest problems
  - Purely an estimate and your experience may vary
  - Many tuning efforts combine multiple areas
    - Especially true of SQL and Object Analysis

Solving the Problem
SQL Analysis Across the Application Life Cycle





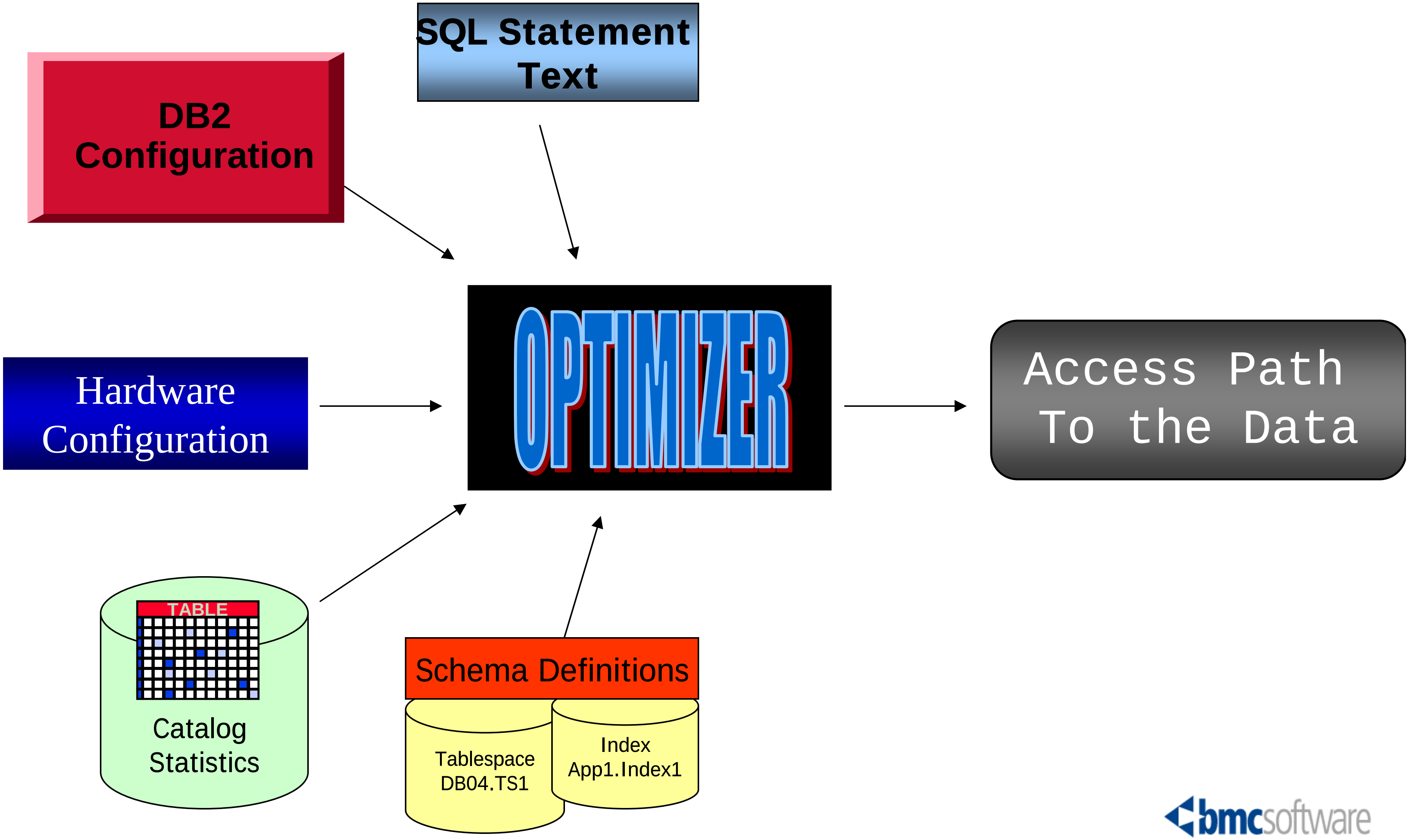
Atomic SQL

- Focus on individual SQL statements
  - Do they meet “best practice” coding standards
  - Do they use expected/accepted DB2 access paths
  - Do they deliver desired result set in acceptable time with acceptable resource consumption
- Developed and tested in controlled environment
- More predictive in nature

SQL Workload

- Focus on workload dynamics
  - How does concurrent execution affect response time/resource consumption
  - Does this SQL statement/program collide with other transactions
  - Same application
    - Other applications in a shared subsystem
- Real world unpredictability comes into play
- More focus on measuring the workload and rapidly reacting

SQL Tuning Fundamentals
DB2 Optimizer Determines SQL Performance



SQL Statement Text

DB2 Configuration

Hardware Configuration

OPTIMIZER

Access Path To the Data

TABLE

Catalog Statistics

Schema Definitions

Tablespace DB04.TS1

Index App1.Index1

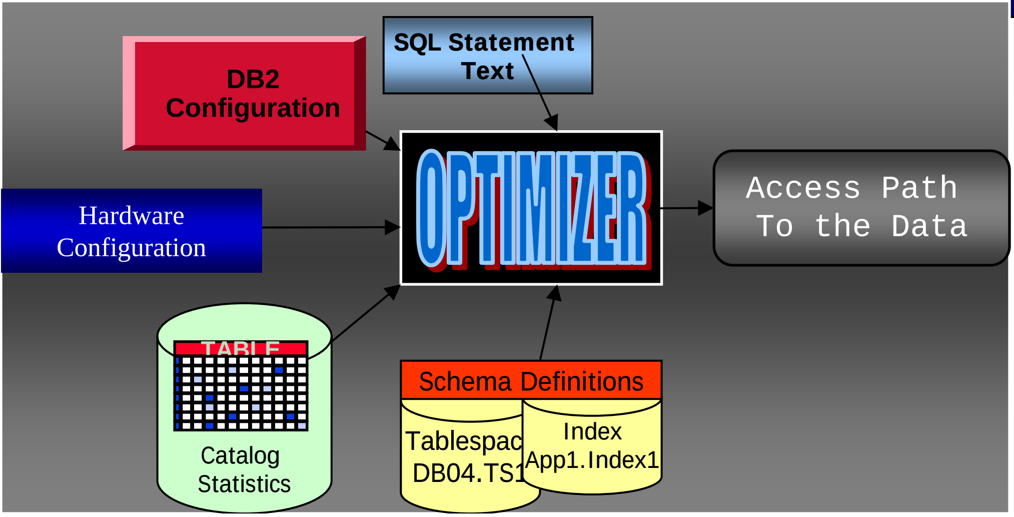
SQL Tuning Fundamentals
Access Path Selection

Static SQL

- Access path determined at bind time – better performance
  - Exceptions to the rule
    - REOPT (VARS) or (ALWAYS)
      - Access path determined at run time for those statements with host variables or parameter markers
    - PREPARE(DEFER)
      - Option useful in distributed environments for reducing message traffic
- Authorization for execution at the plan/package level
- Qualifiers passed via host variables
- SQLJ provides for bound static SQL in Java applications

For Dynamic SQL

- Access Path Selection determined at execution
  - That's the PREPARE
  - Exceptions to the Rule
    - KEEPDYNAMIC bind option
      - Holds prepared statements across commits to avoid cost of re-preparing statement
    - Global Dynamic Statement Cache
      - Maintains Skeleton of prepared statements
- Build and execute SQL on the fly
- User requires authorization to all accessed objects
- Parameter markers for passing variables



DB2 Configuration

SQL Statement Text

Hardware Configuration

OPTIMIZER

Access Path To the Data

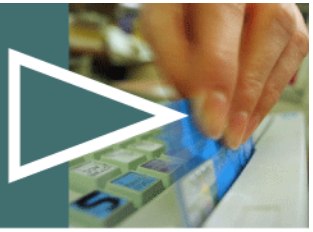
Catalog Statistics

Schema Definitions

Tablespac DB04.TS1

Index App1.Index1

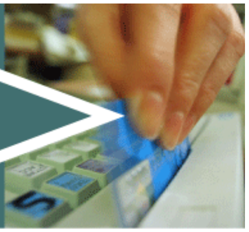
Trends in the Marketplace
Static vs. Dynamic SQL



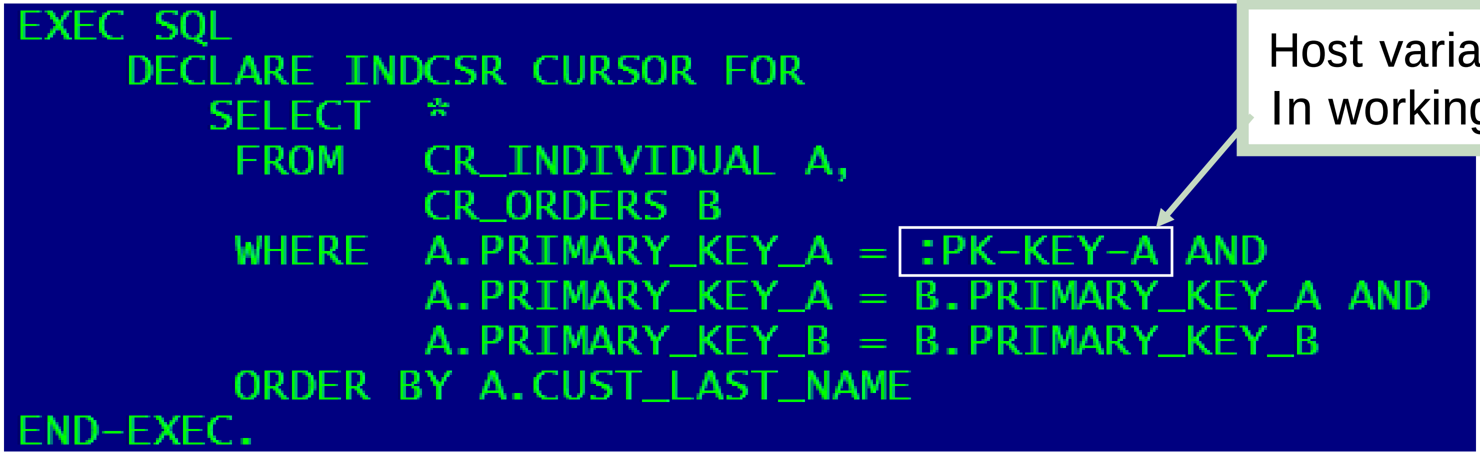
- Dynamic SQL usage is on the increase

- What’s driving it?
  - Dynamic SQL offers flexibility that can simplify developing complex applications
  - New applications being developed on distributed platforms using connections that only support dynamic SQL
    - DB2 CONNECT, etc.
  - ERP applications implemented with dynamic SQL
    - SAP, PeopleSoft, Siebel
  - New applications being developed on distributed platforms
    - New developers are much more familiar with GUI-based programming environments and don’t even sign on to the mainframe
      - More Java and C++

SQL Fundamentals - Static SQL



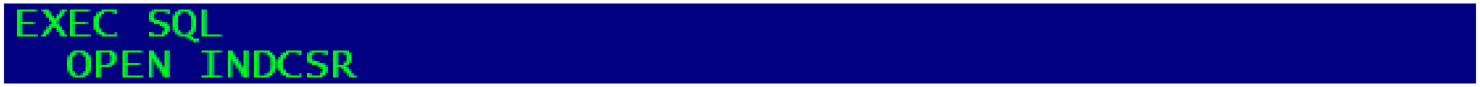
- Data access requirements well defined and predictable
- Static SQL cursor constructs
  - Define the Cursor



```
EXEC SQL
    DECLARE INDCSR CURSOR FOR
       SELECT  *
        FROM   CR_INDIVIDUAL A,
               CR_ORDERS B
        WHERE  A.PRIMARY_KEY_A = :PK-KEY-A AND
               A.PRIMARY_KEY_A = B.PRIMARY_KEY_A AND
               A.PRIMARY_KEY_B = B.PRIMARY_KEY_B
        ORDER BY A.CUST_LAST_NAME
END-EXEC.
```

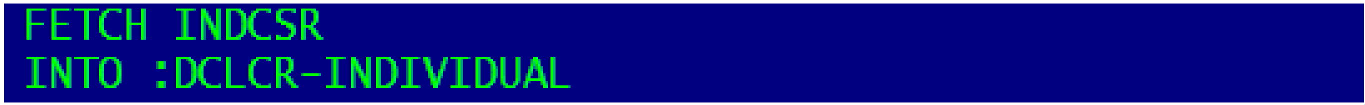
Host variable defined
In working storage

  - Open the cursor



```
EXEC SQL
  OPEN INDCSR
```

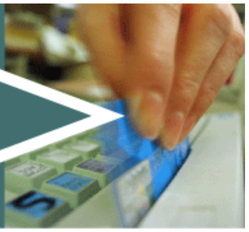
  - Fetch the rows from the result set



```
FETCH INDCSR
INTO :DCLCR-INDIVIDUAL
```

  - Close the cursor

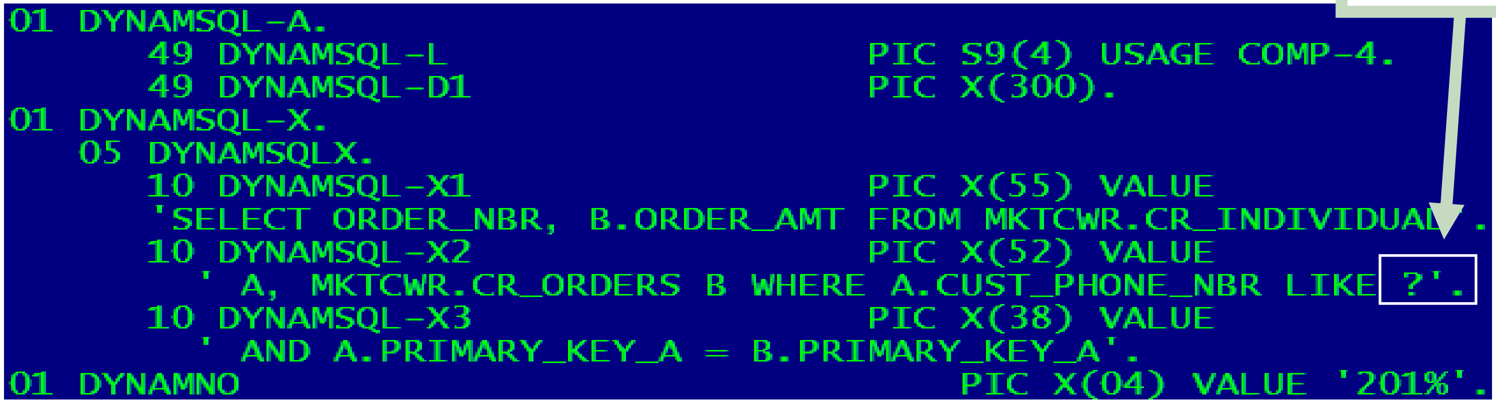
SQL Fundamentals - Dynamic SQL



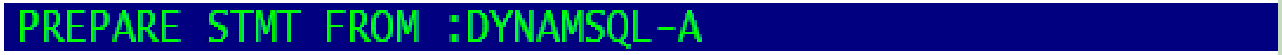
› Data access requirements are ad hoc in nature and identified on the fly

– SELECT Operations

Parameter marker provides placeholder for later substitution

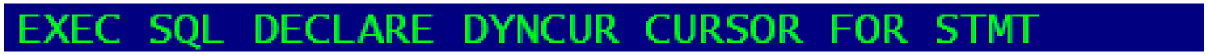


```
01 DYNAMSQL-A.
      49 DYNAMSQL-L                        PIC S9(4) USAGE COMP-4.
      49 DYNAMSQL-D1                       PIC X(300).
01 DYNAMSQL-X.
   05 DYNAMSQLX.
      10 DYNAMSQL-X1                       PIC X(55) VALUE
      'SELECT ORDER_NBR, B.ORDER_AMT FROM MKTCWR.CR_INDIVIDUAL .
      10 DYNAMSQL-X2                       PIC X(52) VALUE
        ' A, MKTCWR.CR_ORDERS B WHERE A.CUST_PHONE_NBR LIKE ?'.
      10 DYNAMSQL-X3                       PIC X(38) VALUE
        ' AND A.PRIMARY_KEY_A = B.PRIMARY_KEY_A'.
01 DYNAMNO                                     PIC X(04) VALUE '201%'.
```

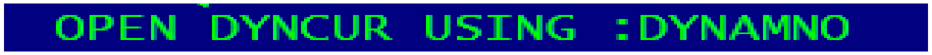


```
PREPARE STMT FROM :DYNAMSQL-A
```

Notice the literal

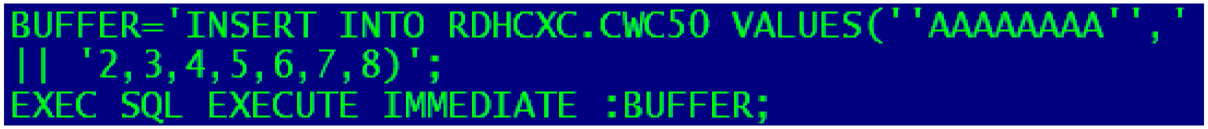


```
EXEC SQL DECLARE DYNCUR CURSOR FOR STMT
```



```
  OPEN DYNCUR USING :DYNAMNO
```

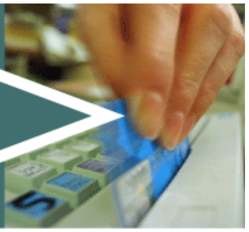
› Other operations

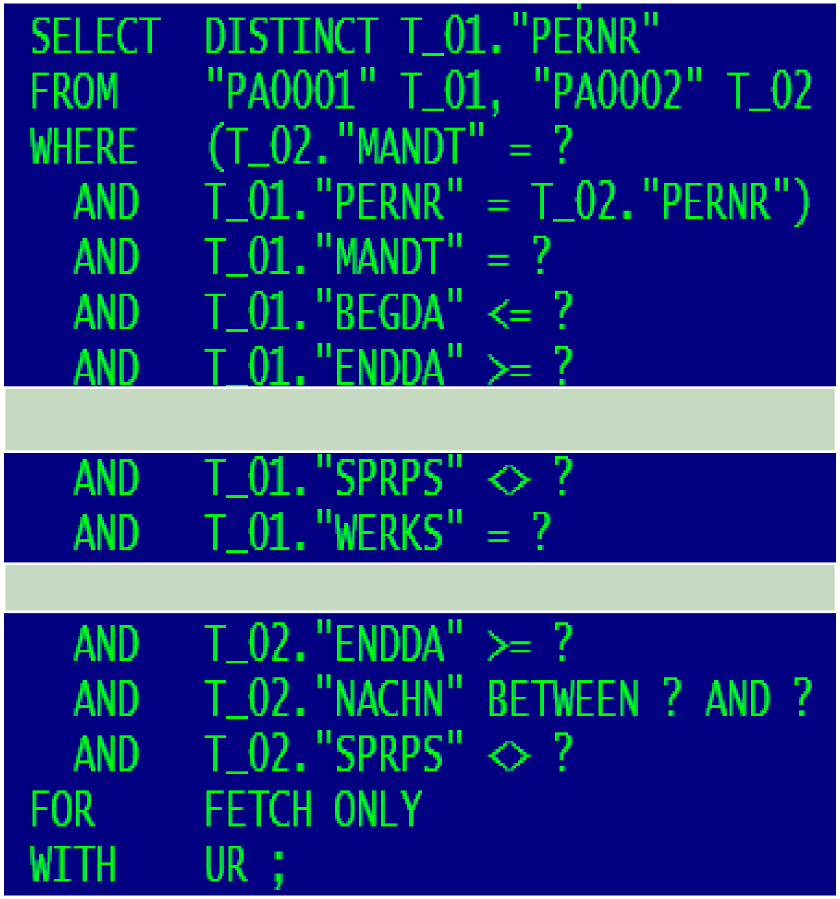


```
BUFFER='INSERT INTO RDHCXC.CWC50 VALUES(''AAAAAAAA'','
|| '2,3,4,5,6,7,8)';
EXEC SQL EXECUTE IMMEDIATE :BUFFER;
```

– Cause the INSERT statement to be prepared and executed immediately

SQL Fundamentals - Dynamic SQL In Practice

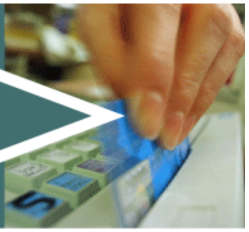




```
SELECT   DISTINCT T_01."PERNR"
FROM     "PA0001" T_01, "PA0002" T_02
WHERE    (T_02."MANDT" = ?
  AND    T_01."PERNR" = T_02."PERNR")
  AND    T_01."MANDT" = ?
  AND    T_01."BEGDA" <= ?
  AND    T_01."ENDDA" >= ?

  AND    T_01."SPRPS" <> ?
  AND    T_01."WERKS" = ?

  AND    T_02."ENDDA" >= ?
  AND    T_02."NACHN" BETWEEN ? AND ?
  AND    T_02."SPRPS" <> ?
FOR      FETCH ONLY
WITH     UR ;
```

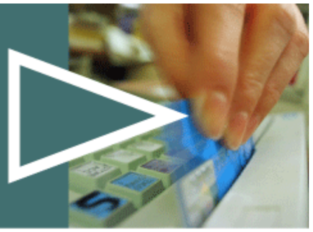
- A Statement from a major ERP application
- Built on the fly based on search criteria selected
- A complex statement with unpredictable input
  - Default statement syntax includes minimal number of search criteria
  - More search criteria the statement expands to include those search arguments
  - If using static SQL could require over 100 cursor definitions in the program

Dynamic SQL Operational Considerations



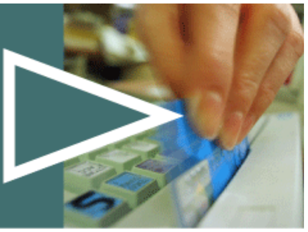
- Sensitive to DB2 statistics
  - Dynamic SQL always uses current catalog statistics for access path selection
    - Changes in DB2 statistics can cause unpredictable changes in access paths
  - Some DB2 customers collect catalog statistics to drive maintenance processes
    - May cause SQL performance to fluctuate unexpectedly
- Security is generally more complex with dynamic SQL
  - Application users generally require authorization to the objects being accessed
  - Auditing is also affected because statements are developed on the fly
- Governor capability may be required
  - Performance characteristics can vary widely for dynamic
  - DB2 Resource Limit Facility may be required
- Access path analysis difficult because access path is not available prior to execution

Dynamic SQL Considerations
PREPARE Yourself



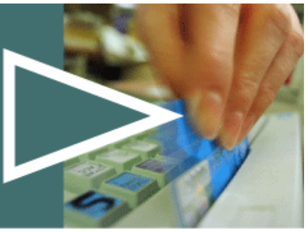
- Repeated PREPAREs drive up the cost of dynamic SQL
  - Prepared statements by default are not persistent across UOWs
  - Prepare costs vary widely but are significant
- Key requirement from anyone developing dynamic SQL applications to reduce or eliminate the cost of preparing dynamic SQL statements
  - Driven initially by SAP and other ERP vendors
  - More in-house dynamic SQL applications drive this requirement
- Enter Dynamic Statement Caching

Introduction to Dynamic Statement Caching

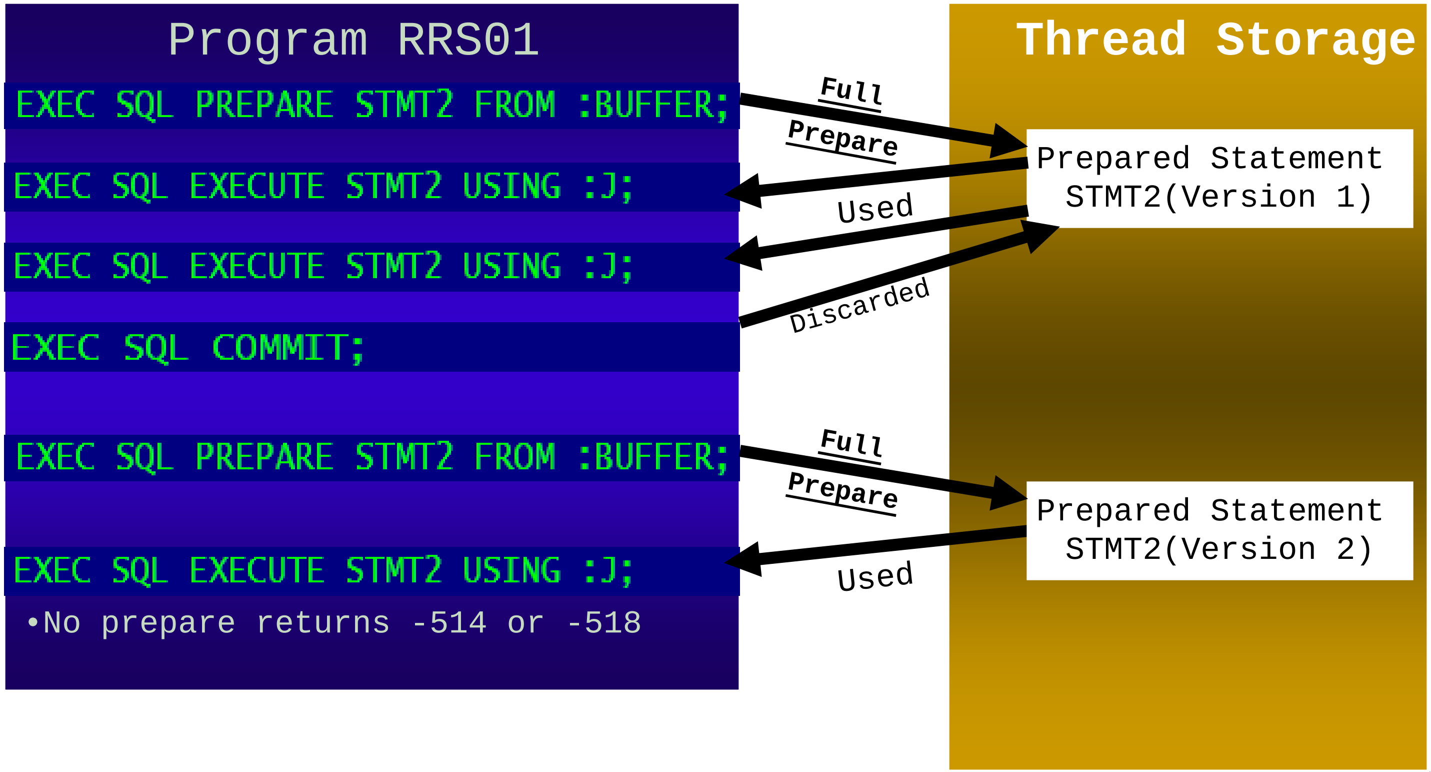


- Goal is to reduce or eliminate SQL Prepare operations required for dynamic SQL statements
- Implementation
  - Four kinds of caching
    - No caching
    - Local Dynamic Statement Caching
    - Global Dynamic Statement Caching
    - Full Caching
  - Cache prepared SQL statement and statement text for dynamic SQL statements in DBM1address space
    - Local Statement Cache
    - Global Dynamic Statement Cache
  - Controlled by various parameters
    - Bind options
    - DSNZPARMs
    - Application constructs

Dynamic Statement Caching
No Statement Caching



- Prepared statements do not persist across commits
  - Discarded at commit
  - Except for statements defined with CURSOR for HOLD
- Default mode of operation



Program RRS01

```
EXEC SQL PREPARE STMT2 FROM :BUFFER;
EXEC SQL EXECUTE STMT2 USING :J;
EXEC SQL EXECUTE STMT2 USING :J;
EXEC SQL COMMIT;
EXEC SQL PREPARE STMT2 FROM :BUFFER;
EXEC SQL EXECUTE STMT2 USING :J;
```

•No prepare returns -514 or -518

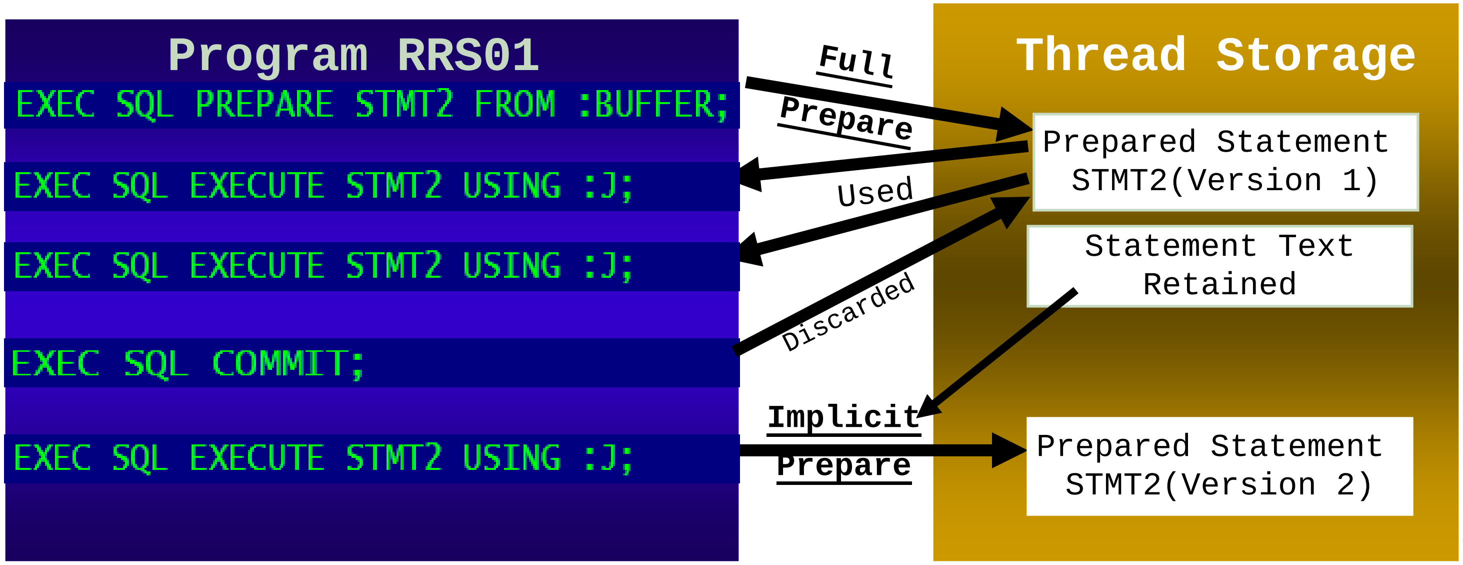
Thread Storage

Prepared Statement STMT2(Version 1)

Prepared Statement STMT2(Version 2)

Dynamic Statement Caching
With Local Statement Caching Only

- Eliminates need for application to do multiple prepares for same statement
  - Implicit prepares done by DB2
- Enabling Local Statement Caching
  - KEEPDYNAMIC(YES) Bind Parameter
  - MAXKEEPD DSNZPARM controls maximum prepared statements
    - Does not affect statement text which is always kept
- Differentiation between prepared statement and statement text
- Minimal benefit if used alone
  - Some reduction in message traffic in a distributed environment is possible

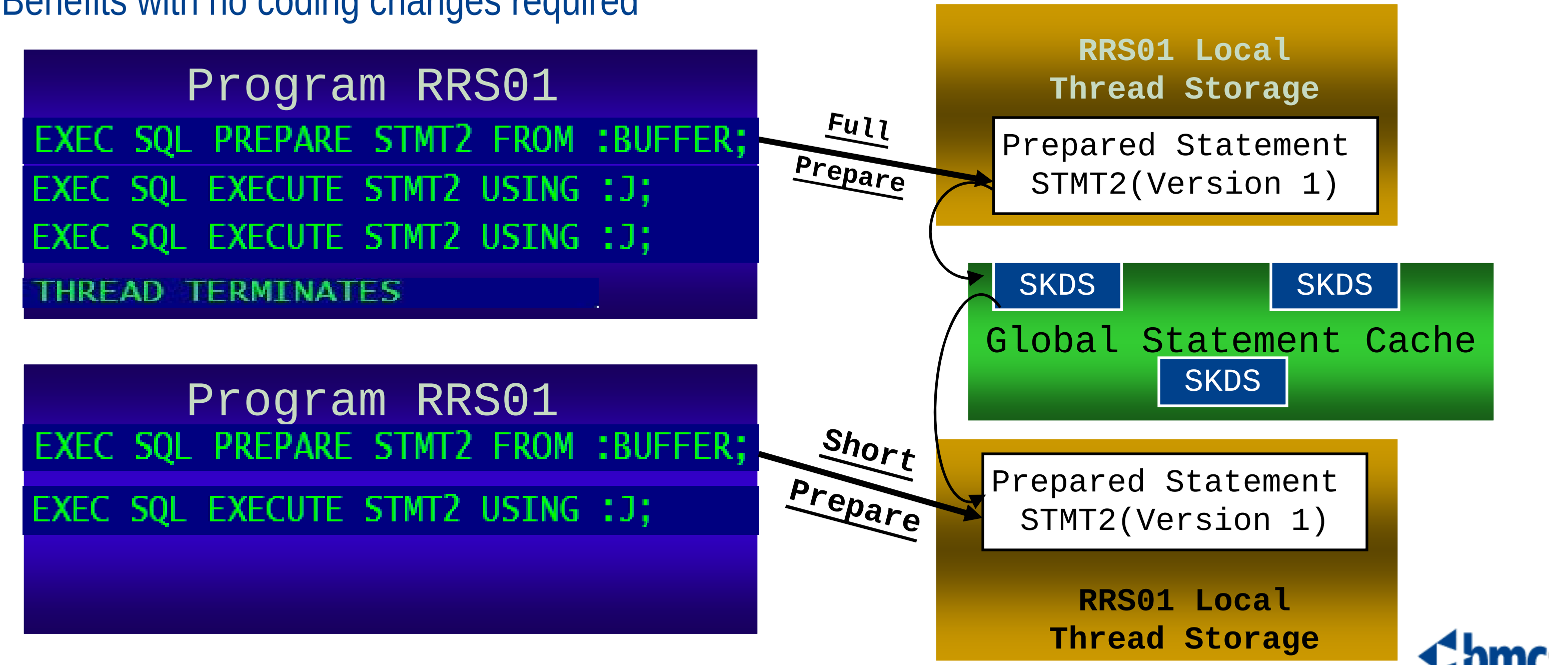


Program RRS01

```
EXEC SQL PREPARE STMT2 FROM :BUFFER;

EXEC SQL EXECUTE STMT2 USING :J;

EXEC SQL EXECUTE STMT2 USING :J;

EXEC SQL COMMIT;

EXEC SQL EXECUTE STMT2 USING :J;
```

Dynamic Statement Caching
Global Statement Caching Only

- Allows reuse of prepared statements across UOWs
  - Within and across program executions
  - Prepared statement (SKDS) cached in global dynamic statement cache
    - Copied into local storage when possible
    - <u>Short Prepare</u>
- Enabling global statement caching
  - CACHEDYN=YES DSNZPARM value
  - Storage allocation discussed later
- Big benefit for applications with frequent reuse of dynamic SQL
  - Benefits with no coding changes required



Program RRS01

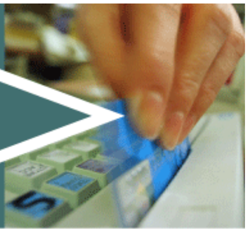
```
EXEC SQL PREPARE STMT2 FROM :BUFFER;
EXEC SQL EXECUTE STMT2 USING :J;
EXEC SQL EXECUTE STMT2 USING :J;
THREAD TERMINATES
```

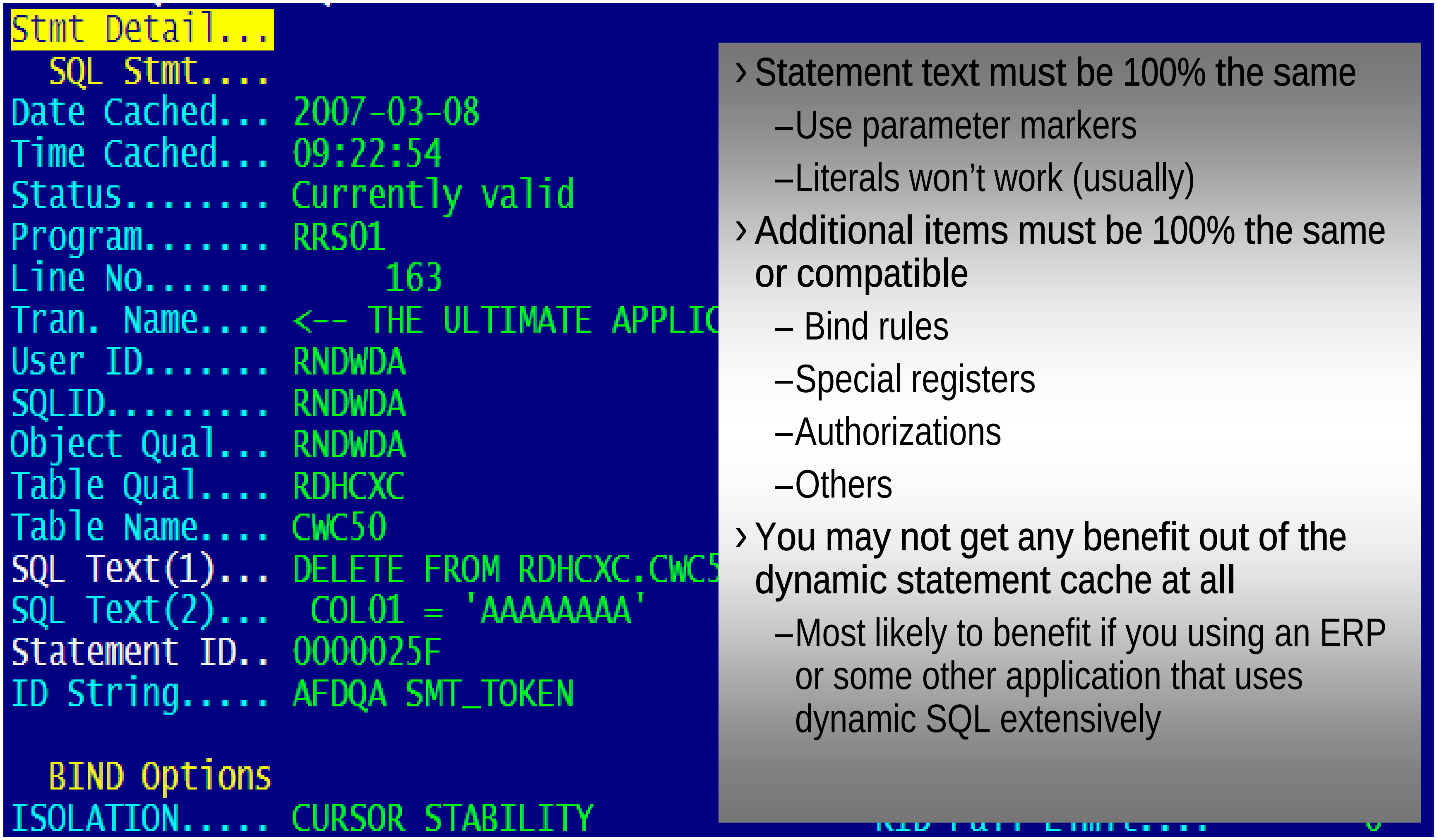
Program RRS01

```
EXEC SQL PREPARE STMT2 FROM :BUFFER;
EXEC SQL EXECUTE STMT2 USING :J;
```

Full Prepare
Short Prepare
RRS01 Local Thread Storage
Prepared Statement STMT2(Version 1)
SKDS
SKDS
Global Statement Cache
SKDS
Prepared Statement STMT2(Version 1)
RRS01 Local Thread Storage

Dynamic Statement Caching
Where Cached Statements can be Reused

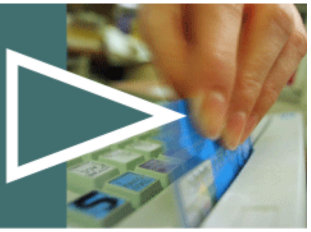




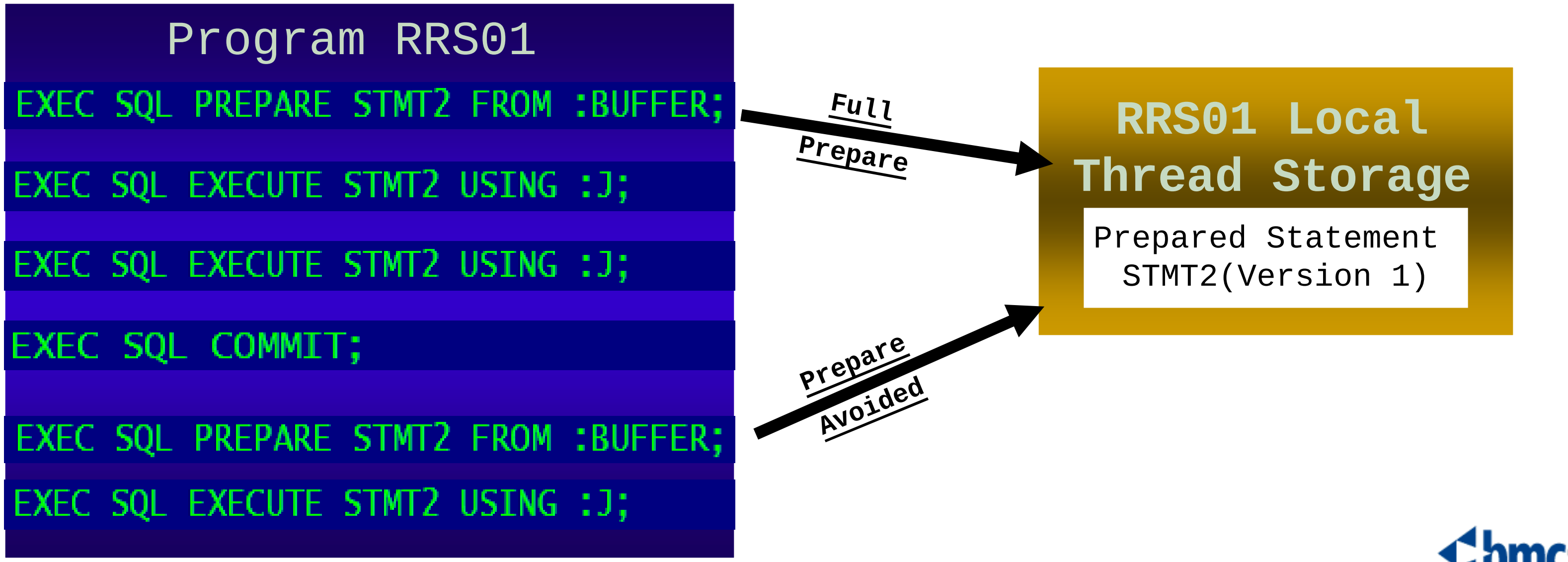
```
Stmt Detail...
  SQL Stmt....
Date Cached... 2007-03-08
Time Cached... 09:22:54
Status........ Currently valid
Program....... RRS01
Line No.......      163
Tran. Name.... <-- THE ULTIMATE APPLIC
User ID....... RNDWDA
SQLID......... RNDWDA
Object Qual... RNDWDA
Table Qual.... RDHCXC
Table Name.... CWC50
SQL Text(1)... DELETE FROM RDHCXC.CWC5
SQL Text(2)...  COL01 = 'AAAAAAAA'
Statement ID.. 0000025F
ID String..... AFDQA SMT_TOKEN

  BIND Options
ISOLATION..... CURSOR STABILITY
```

- Statement text must be 100% the same
  - Use parameter markers
  - Literals won't work (usually)
- Additional items must be 100% the same or compatible
  - Bind rules
  - Special registers
  - Authorizations
  - Others
- You may not get any benefit out of the dynamic statement cache at all
  - Most likely to benefit if you using an ERP or some other application that uses dynamic SQL extensively

Dynamic Statement Caching
Full Caching – A Final Flavor



- Combines benefits of local and global statement caching
  - Ability to completely avoid prepare operations
  - Prepared statement kept in local thread storage and not invalidated across commits
    - <u>Prepare Avoidance</u>
- Enabling global statement caching
  - CACHEDYN=YES, MAXKEEPD>0, KEEPDYNAMIC(YES)
- Maximum benefit within an application execution
  - Local thread storage is discarded at thread termination



Program RRS01

```
EXEC SQL PREPARE STMT2 FROM :BUFFER;
EXEC SQL EXECUTE STMT2 USING :J;
EXEC SQL EXECUTE STMT2 USING :J;
EXEC SQL COMMIT;
EXEC SQL PREPARE STMT2 FROM :BUFFER;
EXEC SQL EXECUTE STMT2 USING :J;
```

Full Prepare

Prepare Avoided

RRS01 Local Thread Storage

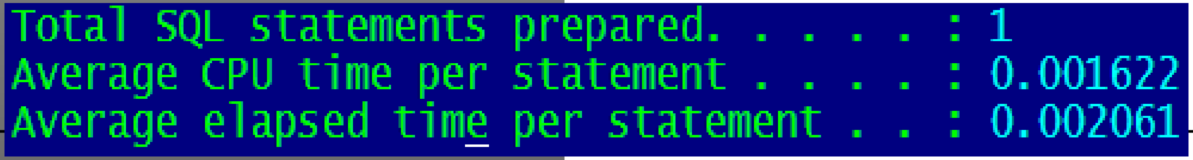
Prepared Statement STMT2(Version 1)

Dynamic Statement Caching Cost Impacts

Relative Cost

- Full Prepare
  - Statement not in cache
  - Global statement caching not active

100

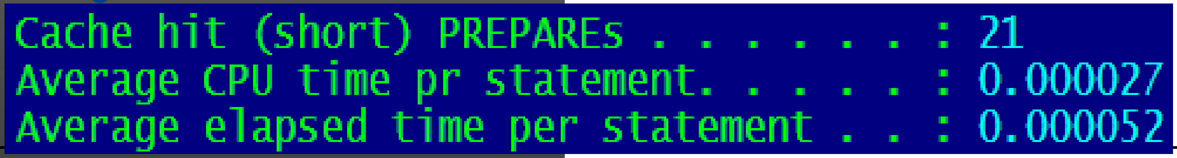


```
Total SQL statements prepared. . . . . . : 1
Average CPU time per statement . . . . . : 0.001622
Average elapsed time per statement . . . : 0.002061
```

- Short Prepare
  - Dynamic statement (SKDS) in the global cache
  - Global caching active

1

DB2 Execution Metrics

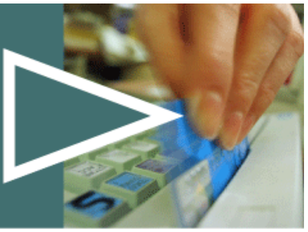


```
Cache hit (short) PREPAREs . . . . . . . : 21
Average CPU time pr statement. . . . . . : 0.000027
Average elapsed time per statement . . . : 0.000052
```

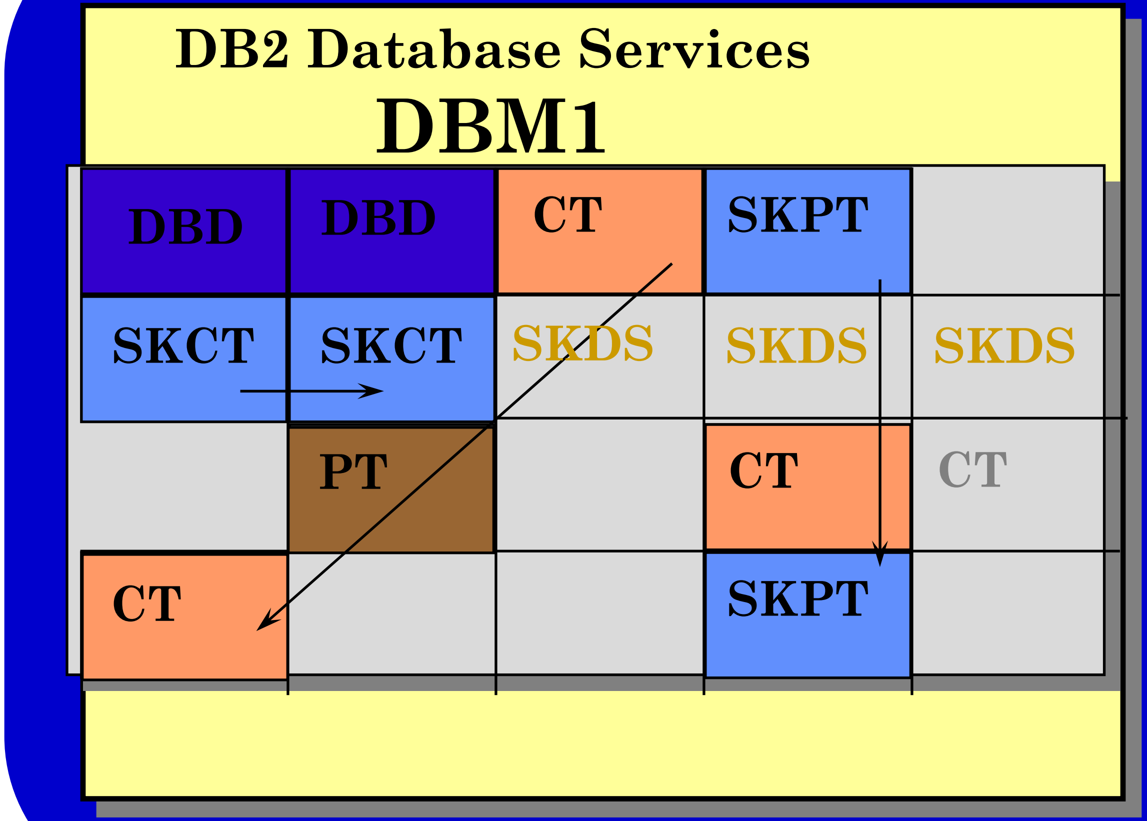
- Avoided Prepare
  - Local and global caching active

0

Dynamic Statement Caching Impacts on Storage



EDM Pool in DB2 V7



DB2 Database Services

DBM1

DBD DBD CT SKPT

SKCT SKCT SKDS SKDS SKDS

PT CT CT

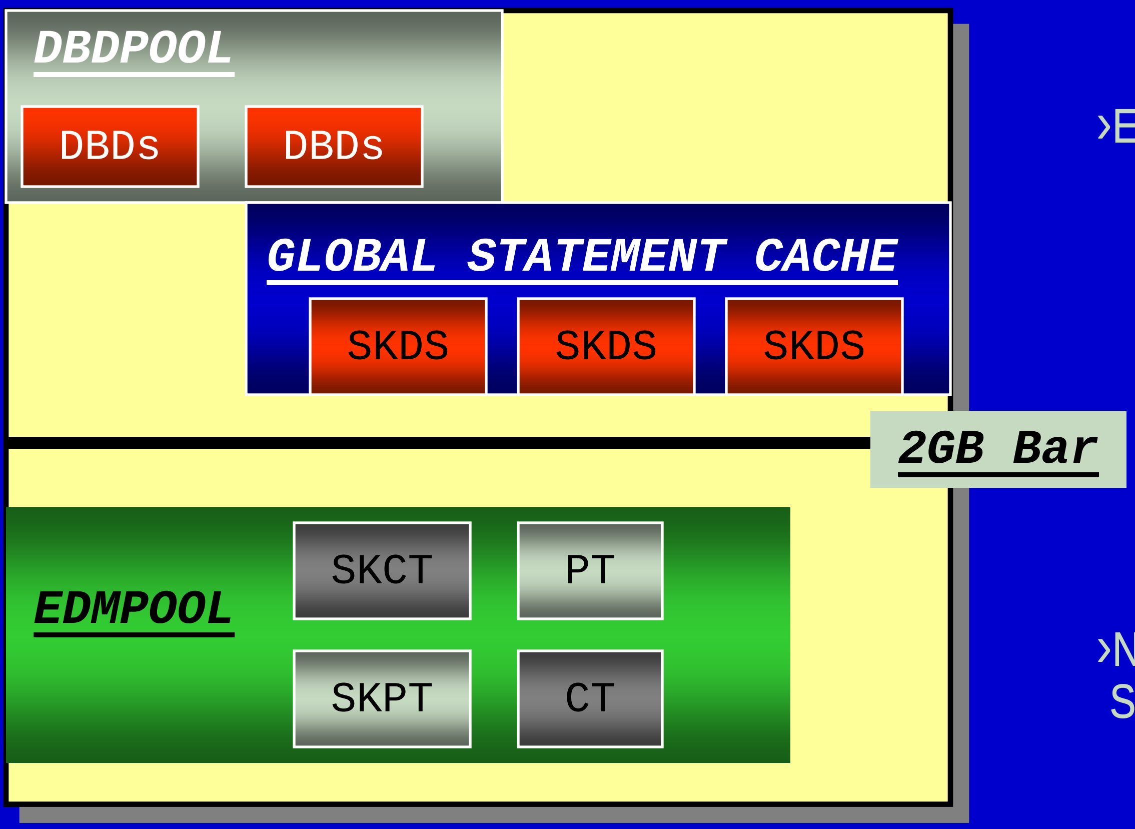
CT SKPT

- ›Caches access path & internal structure definitions
- ›This pool contains
  - –DBDs – database descriptors
  - –Skeleton Package and Cursor Tables (SKPT & SKCT)
  - –Package and Cursor Tables – (PT/CT)
  - –Authorization cache block for each plan (optional)
  - –SKDS - Skeletons of dynamic SQL for CACHE DYNAMIC SQL (optional)
    - ➢Optionally stored in a dataspace
  - –Trigger Packages

Dynamic Statement Caching Impacts on Storage

EDM Pool In DB2 V8

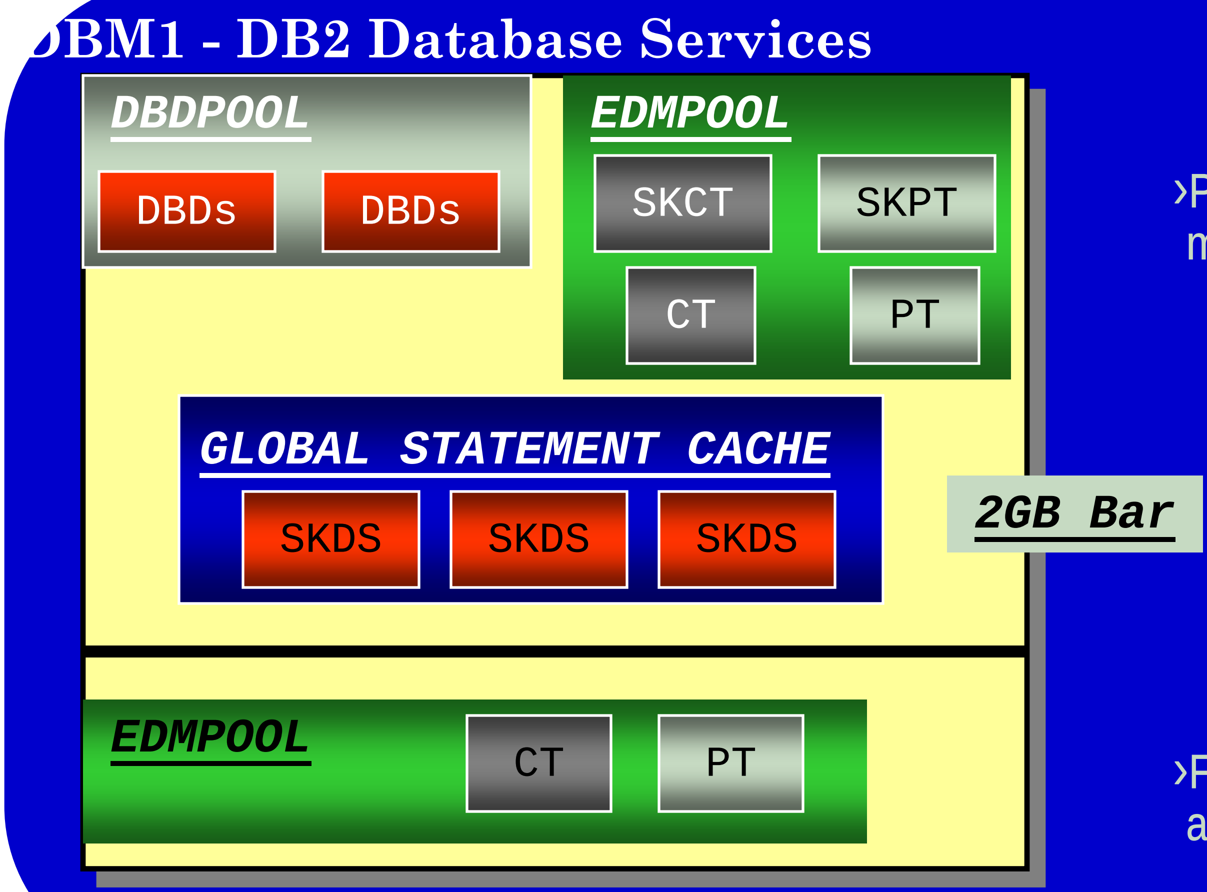
DBM1 - DB2 Database Services



- ›EDMPOOL now in 3 separate pools
  - – EDMDBDC – DBDs
    - Above the Bar
  - – EDMSTMTC – Dynamic Statements
    - Above the Bar
  - – EDMPOOL – Skeleton Package and Cursor Tables
    - Still below the bar and a potential source of VSC
- ›No dataspace option for Dynamic Statement Cache

Dynamic Statement Caching Impacts on Storage

EDM Pool In DB2 V9

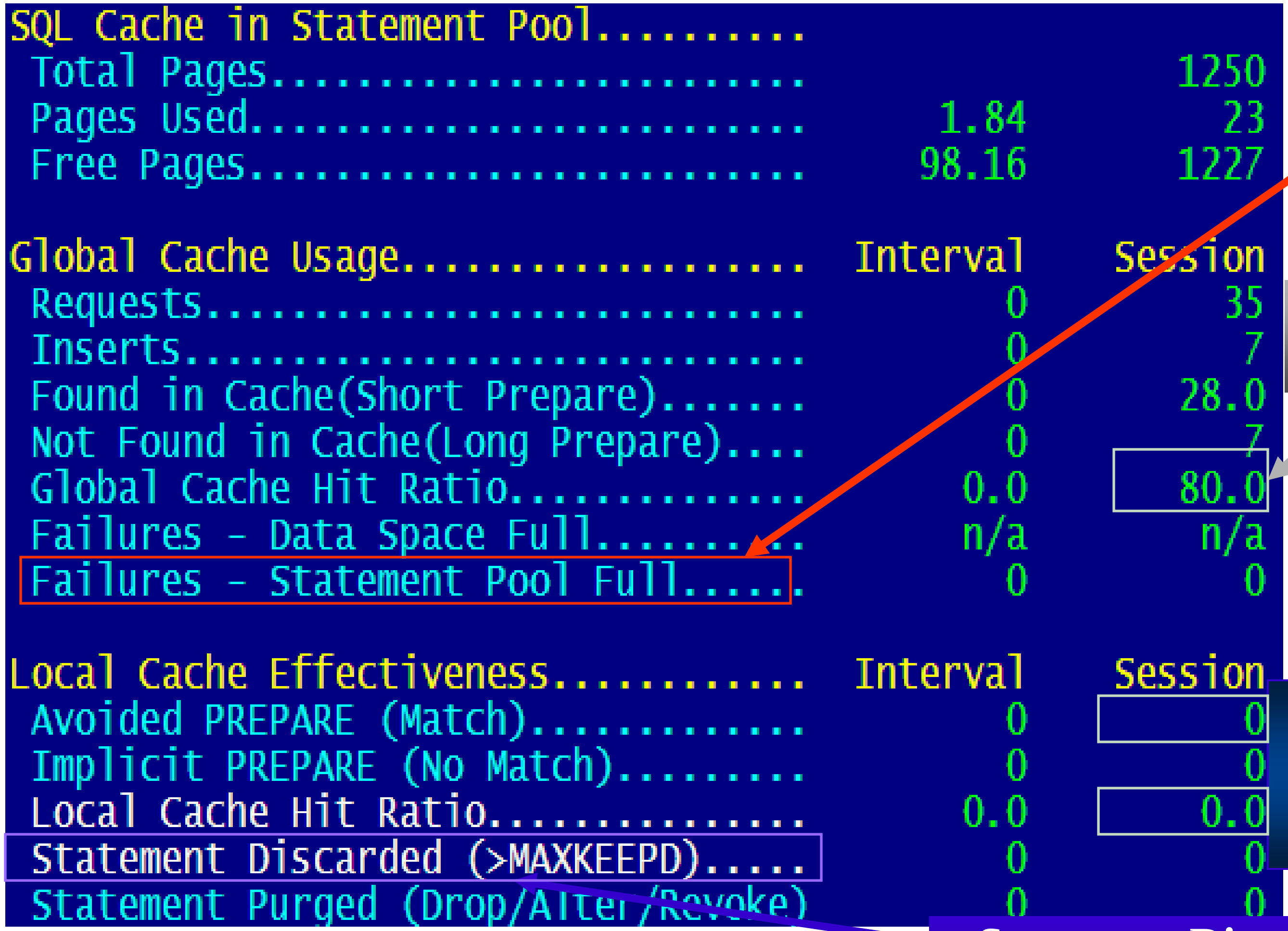


›Portions of runtime Components moved above the bar

- –Plan and package skeletons above the bar
- – Bound/Prepared DML Statements
  - • Statement Text
  - • SQLDA DESCRIBE output
  - • Portion of native SQL PL package
- – Portions of static SQL sections (CT/PT) are moved as well

›Further reduces VSC in the DBM1 address space

Dynamic SQL Statement Caching
DB2 Cache Statistics



```
SQL Cache in Statement Pool..........
 Total Pages.........................              1250
 Pages Used..........................    1.84        23
 Free Pages..........................   98.16      1227

Global Cache Usage.................. Interval   Session
 Requests............................       0        35
 Inserts.............................       0         7
 Found in Cache(Short Prepare).......       0      28.0
 Not Found in Cache(Long Prepare)....       0         7
 Global Cache Hit Ratio..............     0.0      80.0
 Failures - Data Space Full..........     n/a       n/a
 Failures - Statement Pool Full......       0         0

Local Cache Effectiveness........... Interval   Session
 Avoided PREPARE (Match).............       0         0
 Implicit PREPARE (No Match).........       0         0
 Local Cache Hit Ratio...............     0.0       0.0
 Statement Discarded (>MAXKEEPD).....       0         0
 Statement Purged (Drop/Alter/Revoke)       0         0
```

Statement Pool Full Failures

Should be 0

Increase Statement Pool Size if not

Global Cache Hit Ratio

Shoot for 80+%

Local Cache Hit Ratio

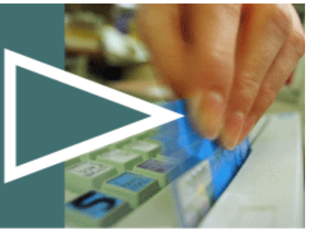
Specific for Applications bound with KEEPDYNAMIC(YES)

Statement Discarded

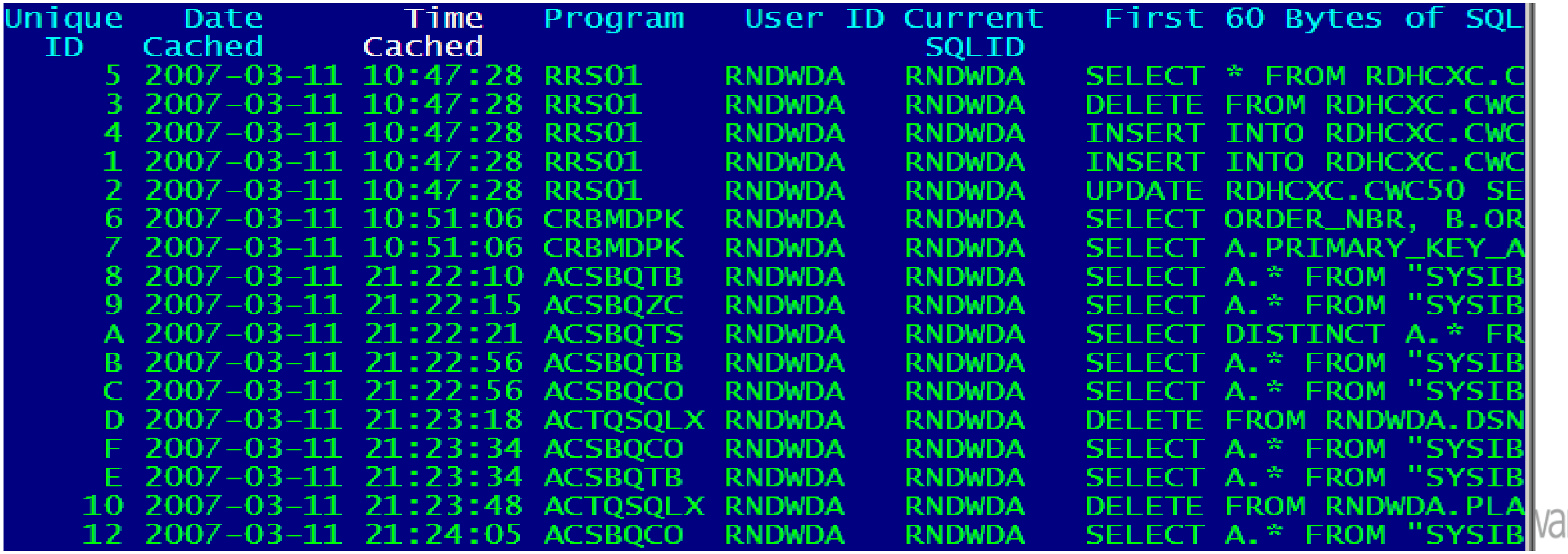
Shoot for 0

Increase MAXKEEPD

The Global Dynamic Statement Cache
What Goes In?

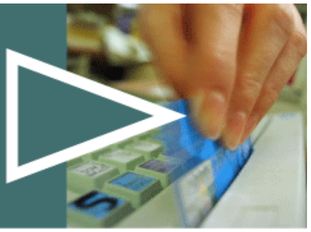


- Dynamic Statements
  - If the Global Cache is active (CACHEDYN=YES) and not a REOPT(ALWAYS) application
  - Reside in the till they are thrown out
    - DROP or ALTER
    - Authorization Revoked
    - LRU
    - RUNSTATS
    - DB2 is recycled



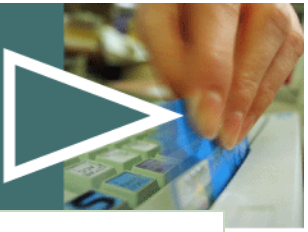
```
Unique    Date          Time     Program    User ID Current    First 60 Bytes of SQL
  ID    Cached        Cached                          SQLID
     5 2007-03-11 10:47:28 RRS01      RNDWDA    RNDWDA    SELECT * FROM RDHCXC.C
     3 2007-03-11 10:47:28 RRS01      RNDWDA    RNDWDA    DELETE FROM RDHCXC.CWC
     4 2007-03-11 10:47:28 RRS01      RNDWDA    RNDWDA    INSERT INTO RDHCXC.CWC
     1 2007-03-11 10:47:28 RRS01      RNDWDA    RNDWDA    INSERT INTO RDHCXC.CWC
     2 2007-03-11 10:47:28 RRS01      RNDWDA    RNDWDA    UPDATE RDHCXC.CWC50 SE
     6 2007-03-11 10:51:06 CRBMDPK    RNDWDA    RNDWDA    SELECT ORDER_NBR, B.OR
     7 2007-03-11 10:51:06 CRBMDPK    RNDWDA    RNDWDA    SELECT A.PRIMARY_KEY_A
     8 2007-03-11 21:22:10 ACSBQTB    RNDWDA    RNDWDA    SELECT A.* FROM "SYSIB
     9 2007-03-11 21:22:15 ACSBQZC    RNDWDA    RNDWDA    SELECT A.* FROM "SYSIB
     A 2007-03-11 21:22:21 ACSBQTS    RNDWDA    RNDWDA    SELECT DISTINCT A.* FR
     B 2007-03-11 21:22:56 ACSBQTB    RNDWDA    RNDWDA    SELECT A.* FROM "SYSIB
     C 2007-03-11 21:22:56 ACSBQCO    RNDWDA    RNDWDA    SELECT A.* FROM "SYSIB
     D 2007-03-11 21:23:18 ACTQSQLX RNDWDA    RNDWDA    DELETE FROM RNDWDA.DSN
     F 2007-03-11 21:23:34 ACSBQCO    RNDWDA    RNDWDA    SELECT A.* FROM "SYSIB
     E 2007-03-11 21:23:34 ACSBQTB    RNDWDA    RNDWDA    SELECT A.* FROM "SYSIB
    10 2007-03-11 21:23:48 ACTQSQLX RNDWDA    RNDWDA    DELETE FROM RNDWDA.PLA
    12 2007-03-11 21:24:05 ACSBQCO    RNDWDA    RNDWDA    SELECT A.* FROM "SYSIB
```

Retrieving Data From the Global Cache

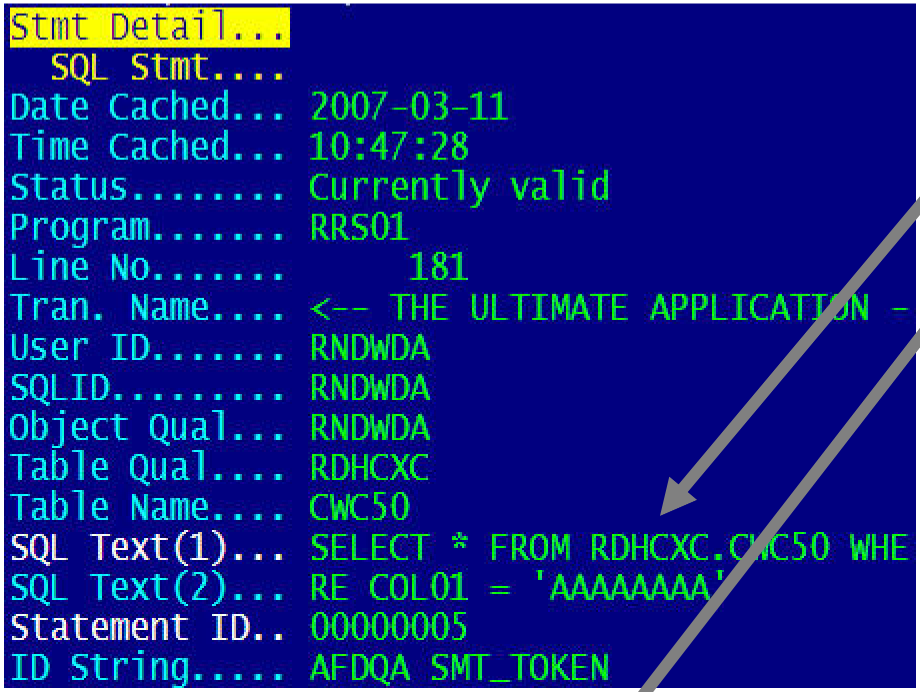


- As shown previously
  - Statement caching performance data in DB2 statistics records
  - Metrics show details about cache hit ratios and other useful data points that help you evaluate overall performance of your statement caches
- For more detail on Global Statement Cache usage the following instrumentation is provided
  - IFCID 316 – Provides details on statements in the cache
    - First 60 bytes of SQL text
    - Includes execution statistics (0 if not being collected)
  - IFCID 317 can then be used to retrieve the entire SQL statement from the cache once you have identified the statement of interest
- EXPLAIN STMTCACHE
  - V8 feature that exports Dynamic Statement Cache information to the DSN_STATEMENT_CACHE_TABLE
  - Nearly identical to the detail in IFCID 316 & 317
  - Multiple options including ALL, stmt-id, and stmt-token

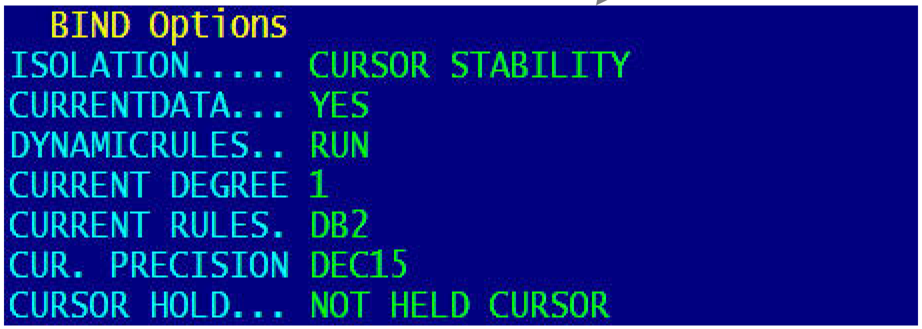
Reviewing Global Statement Cache Information IFCID 316 Results



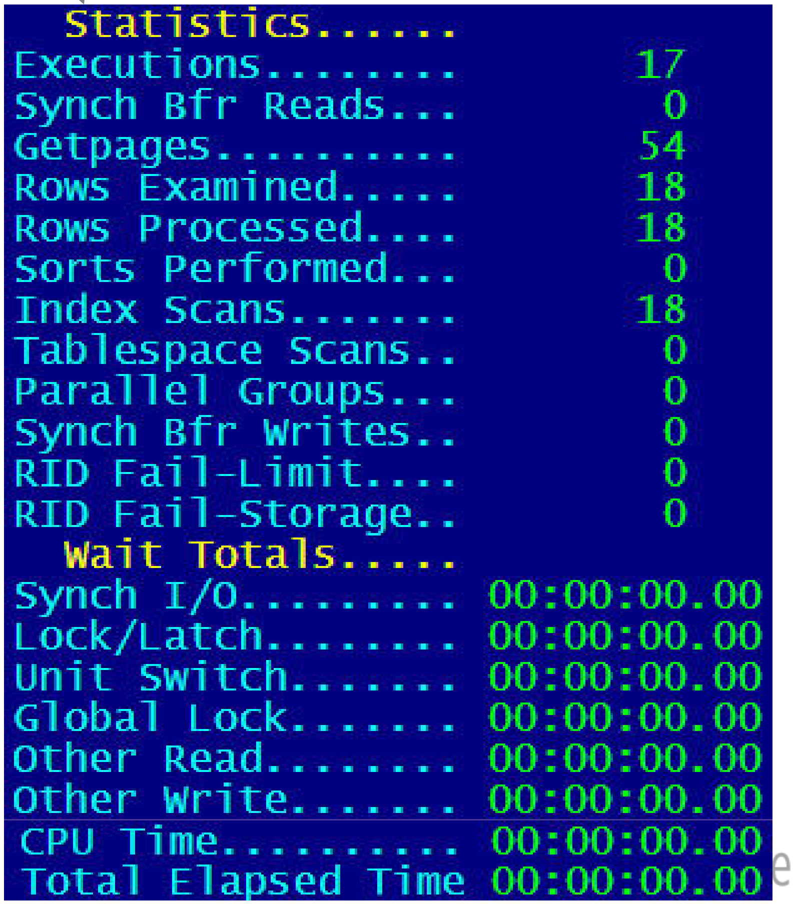
- First 60 Bytes of SQL Text
  - IFCID 317 gives full text
- Bind Options
- Statement Statistics (more later)



```
Stmt Detail...
  SQL Stmt....
Date Cached... 2007-03-11
Time Cached... 10:47:28
Status........ Currently valid
Program....... RRS01
Line No.......      181
Tran. Name.... <-- THE ULTIMATE APPLICATION -
User ID....... RNDWDA
SQLID......... RNDWDA
Object Qual... RNDWDA
Table Qual.... RDHCXC
Table Name.... CWC50
SQL Text(1)... SELECT * FROM RDHCXC.CWC50 WHE
SQL Text(2)... RE COL01 = 'AAAAAAAA'
Statement ID.. 00000005
ID String..... AFDQA SMT_TOKEN
```

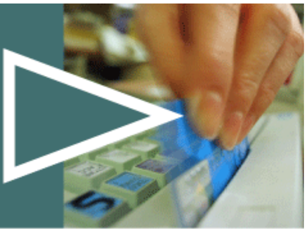


```
  BIND Options
ISOLATION..... CURSOR STABILITY
CURRENTDATA... YES
DYNAMICRULES.. RUN
CURRENT DEGREE 1
CURRENT RULES. DB2
CUR. PRECISION DEC15
CURSOR HOLD... NOT HELD CURSOR
```

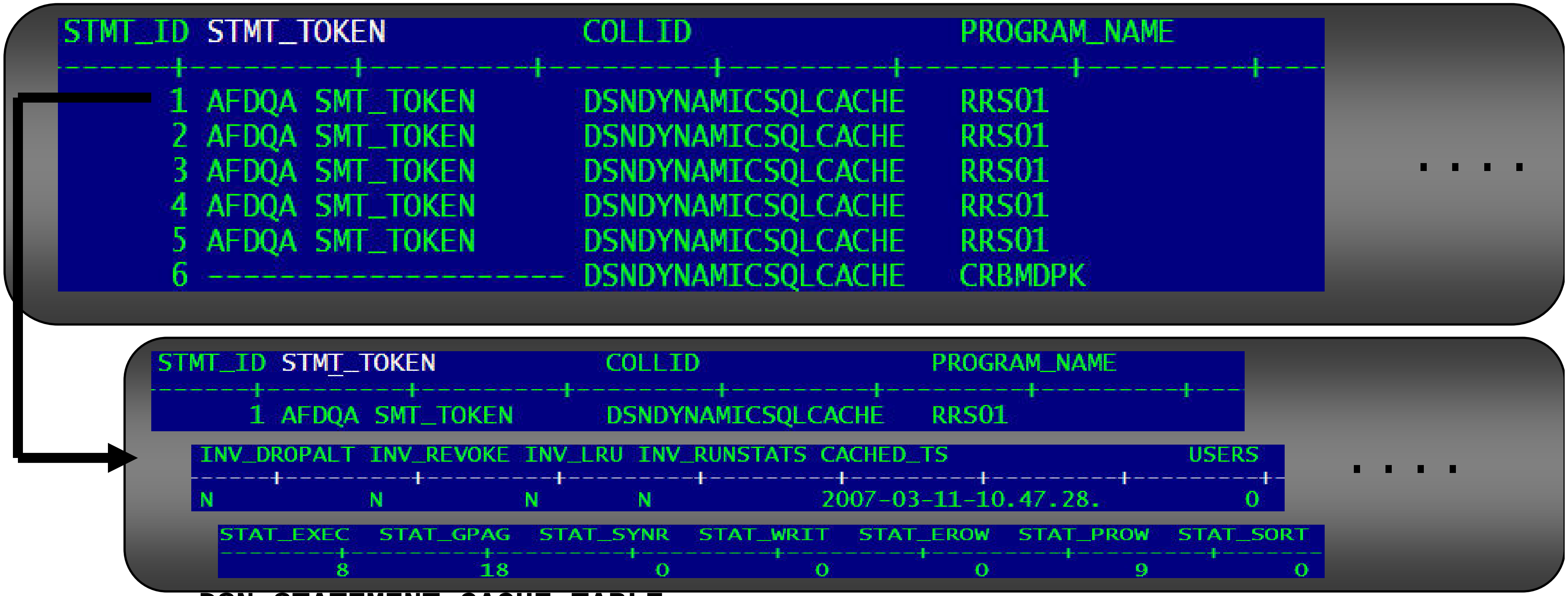


```
  Statistics.......
Executions.........        17
Synch Bfr Reads...          0
Getpages...........        54
Rows Examined.....         18
Rows Processed....         18
Sorts Performed...          0
Index Scans.......         18
Tablespace Scans..          0
Parallel Groups...          0
Synch Bfr Writes..          0
RID Fail-Limit....          0
RID Fail-Storage..          0
  Wait Totals.....
Synch I/O......... 00:00:00.00
Lock/Latch........ 00:00:00.00
Unit Switch....... 00:00:00.00
Global Lock....... 00:00:00.00
Other Read........ 00:00:00.00
Other Write....... 00:00:00.00
CPU Time.......... 00:00:00.00
Total Elapsed Time 00:00:00.00
```

Mining the Dynamic Statement Cache
EXPLAIN STMTCACHE ALL



- Extracts all statements from the global cache
- Inserts one row for each entry in the global DSC
  - Populates DSN_STATEMNT_CACHE_TABLE only
  - STMT_ID column matches the Unique ID in the global statement cache
  - Nearly exact match to the DSC with a few additional columns
  - STMT_TEXT is a 2M CLOB so be careful with that
  - COLLID set to DSNDYNAMICSQLCACHE



```
STMT_ID STMT_TOKEN           COLLID                PROGRAM_NAME
-------+---------+---------+---------+---------+---------+---------+---
      1 AFDQA SMT_TOKEN      DSNDYNAMICSQLCACHE    RRS01
      2 AFDQA SMT_TOKEN      DSNDYNAMICSQLCACHE    RRS01
      3 AFDQA SMT_TOKEN      DSNDYNAMICSQLCACHE    RRS01
      4 AFDQA SMT_TOKEN      DSNDYNAMICSQLCACHE    RRS01
      5 AFDQA SMT_TOKEN      DSNDYNAMICSQLCACHE    RRS01
      6 -------------------- DSNDYNAMICSQLCACHE    CRBMDPK
```

```
STMT_ID STMT_TOKEN           COLLID                PROGRAM_NAME
-------+---------+---------+---------+---------+---------+---------+---
      1 AFDQA SMT_TOKEN      DSNDYNAMICSQLCACHE    RRS01
```

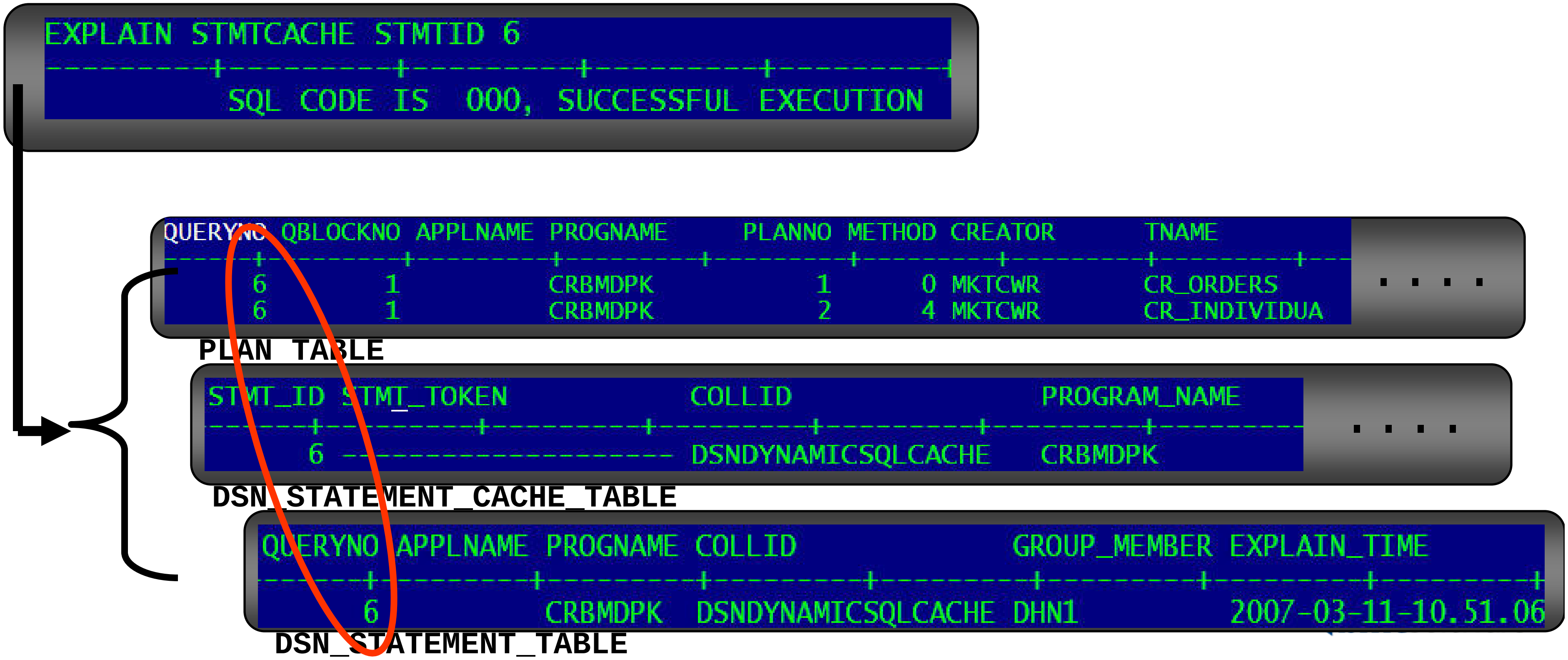
```
INV_DROPALT INV_REVOKE INV_LRU INV_RUNSTATS CACHED_TS                  USERS
-----+---------+---------+---------+---------+---------+---------+---------+-
N           N          N       N            2007-03-11-10.47.28.           0
```

```
STAT_EXEC   STAT_GPAG   STAT_SYNR   STAT_WRIT   STAT_EROW   STAT_PROW   STAT_SORT
---------+---------+---------+---------+---------+---------+---------+-------
        8          18           0           0           0           9           0
```

DSN_STATEMENT_CACHE_TABLE

Mining the Dynamic Statement Cache EXPLAIN STMTCACHE STMT_ID

- › Extracts a single statement from the global DSC
  - – Populates PLAN, DSN_DYNAMIC_STATEMNT, DSN_STATEMENT, and DSN_FUNCTION tables if they exist
  - – Access path is current access path for statement in the cache
  - – Numeric literal or host variable from program
  - – -248 SQL Return Code back to program is STMT_ID not found



```
EXPLAIN STMTCACHE STMTID 6
---------+---------+---------+---------+---------+
          SQL CODE IS 000, SUCCESSFUL EXECUTION
```

```
QUERYNO QBLOCKNO APPLNAME PROGNAME     PLANNO METHOD CREATOR      TNAME
-------+---------+---------+---------+---------+---------+---------+---------+---
      6        1          CRBMDPK          1      0 MKTCWR       CR_ORDERS
      6        1          CRBMDPK          2      4 MKTCWR       CR_INDIVIDUA
```

PLAN_TABLE

```
STMT_ID STMT_TOKEN           COLLID                  PROGRAM_NAME
-------+---------+---------+---------+---------+---------+---------
      6 -------------------- DSNDYNAMICSQLCACHE      CRBMDPK
```

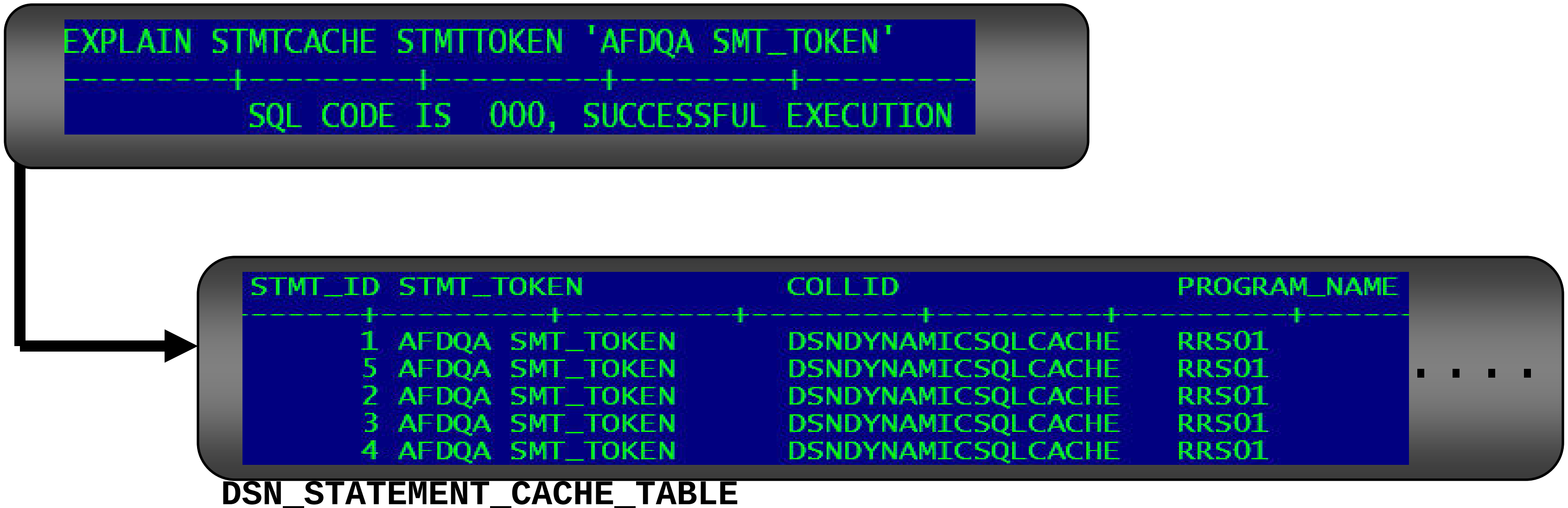
DSN_STATEMENT_CACHE_TABLE

```
QUERYNO APPLNAME PROGNAME COLLID                 GROUP_MEMBER EXPLAIN_TIME
-------+---------+---------+---------+---------+---------+---------+---------+
      6          CRBMDPK  DSNDYNAMICSQLCACHE     DHN1         2007-03-11-10.51.06
```

DSN_STATEMENT_TABLE

Mining the Dynamic Statement Cache
EXPLAIN STMTCACHE STMTTOKEN

- Extracts a group of statements from the global DSC
  - Populates PLAN, DSN_DYNAMIC_STATEMNT, DSN_STATEMENT, and DSN_FUNCTION tables if they exist
  - Access path is current access path for statement in the cache
  - Based on STMT_TOKEN value in the cache
  - Alphanumeric literal or host variable in program
  - -248 SQL Return Code returned if no qualifying entries found in cache

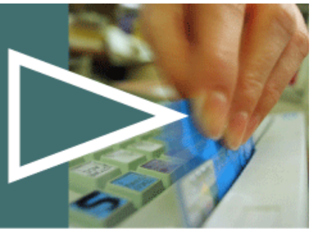


```
EXPLAIN STMTCACHE STMTTOKEN 'AFDQA SMT_TOKEN'
---------+---------+---------+---------+---------
           SQL CODE IS  000, SUCCESSFUL EXECUTION
```

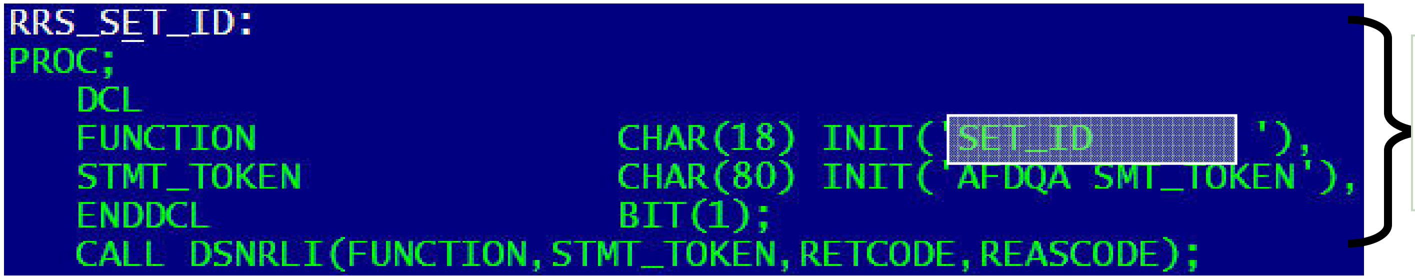
```
STMT_ID STMT_TOKEN           COLLID               PROGRAM_NAME
-------+---------+---------+---------+---------+---------+------
      1 AFDQA SMT_TOKEN      DSNDYNAMICSQLCACHE   RRS01
      5 AFDQA SMT_TOKEN      DSNDYNAMICSQLCACHE   RRS01
      2 AFDQA SMT_TOKEN      DSNDYNAMICSQLCACHE   RRS01
      3 AFDQA SMT_TOKEN      DSNDYNAMICSQLCACHE   RRS01
      4 AFDQA SMT_TOKEN      DSNDYNAMICSQLCACHE   RRS01
```

DSN_STATEMENT_CACHE_TABLE

Mining the Dynamic Statement Cache
More on the STMT_TOKEN in the Cache

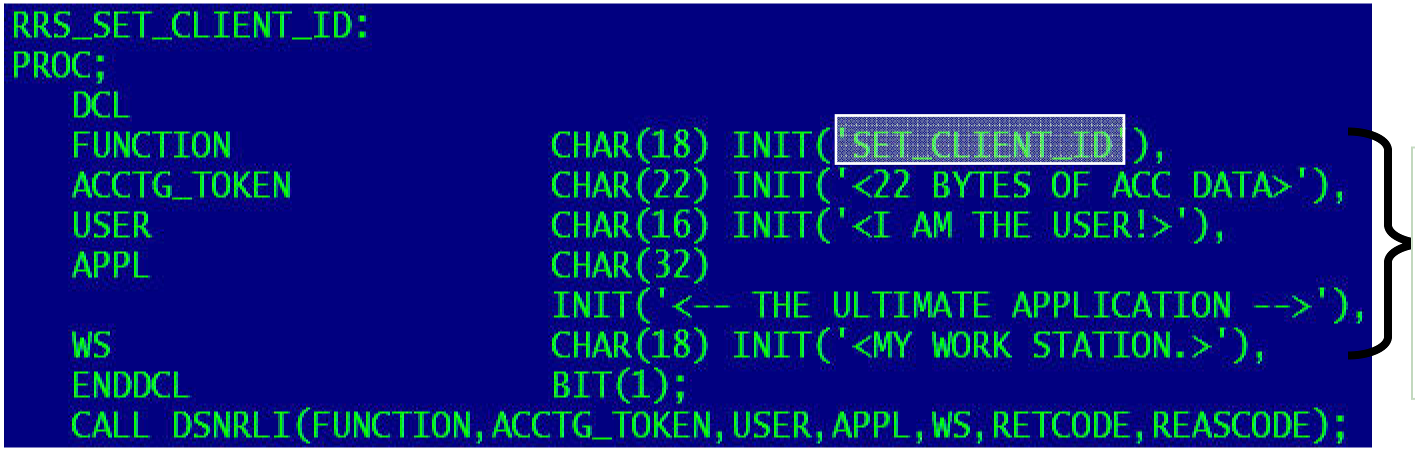


- Provides a method for grouping similar SQL statements
- STMTTOKEN values set using RRSAF or sqleseti functions
- Similar to Client special registers implemented in DB2 v8
- PL/1 RRSAF Example



```
RRS_SET_ID:
PROC;
   DCL
   FUNCTION                   CHAR(18) INIT('SET_ID       '),
   STMT_TOKEN                 CHAR(80) INIT('AFDQA SMT_TOKEN'),
   ENDDCL                     BIT(1);
   CALL DSNRLI(FUNCTION,STMT_TOKEN,RETCODE,REASCODE);
```

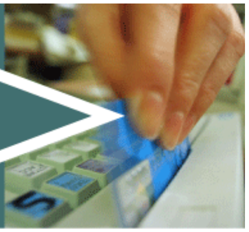
- Set STMTTOKEN Value
- Call to DSNRLI (RRSAF) with SET_ID function
- Error handling follows



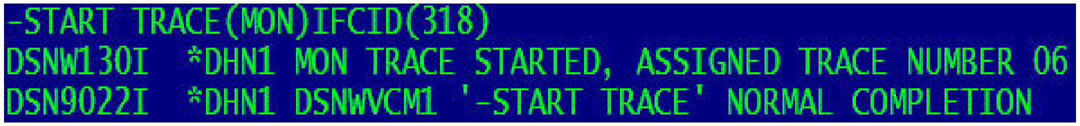
```
RRS_SET_CLIENT_ID:
PROC;
   DCL
   FUNCTION                   CHAR(18) INIT('SET_CLIENT_ID'),
   ACCTG_TOKEN                CHAR(22) INIT('<22 BYTES OF ACC DATA>'),
   USER                       CHAR(16) INIT('<I AM THE USER!>'),
   APPL                       CHAR(32)
                              INIT('<-- THE ULTIMATE APPLICATION -->'),
   WS                         CHAR(18) INIT('<MY WORK STATION.>'),
   ENDDCL                     BIT(1);
   CALL DSNRLI(FUNCTION,ACCTG_TOKEN,USER,APPL,WS,RETCODE,REASCODE);
```

- Set Client Special Registers
- Call to DSNRLI (RRSAF) with SET_CLIENT_ID function
- Error handling follows

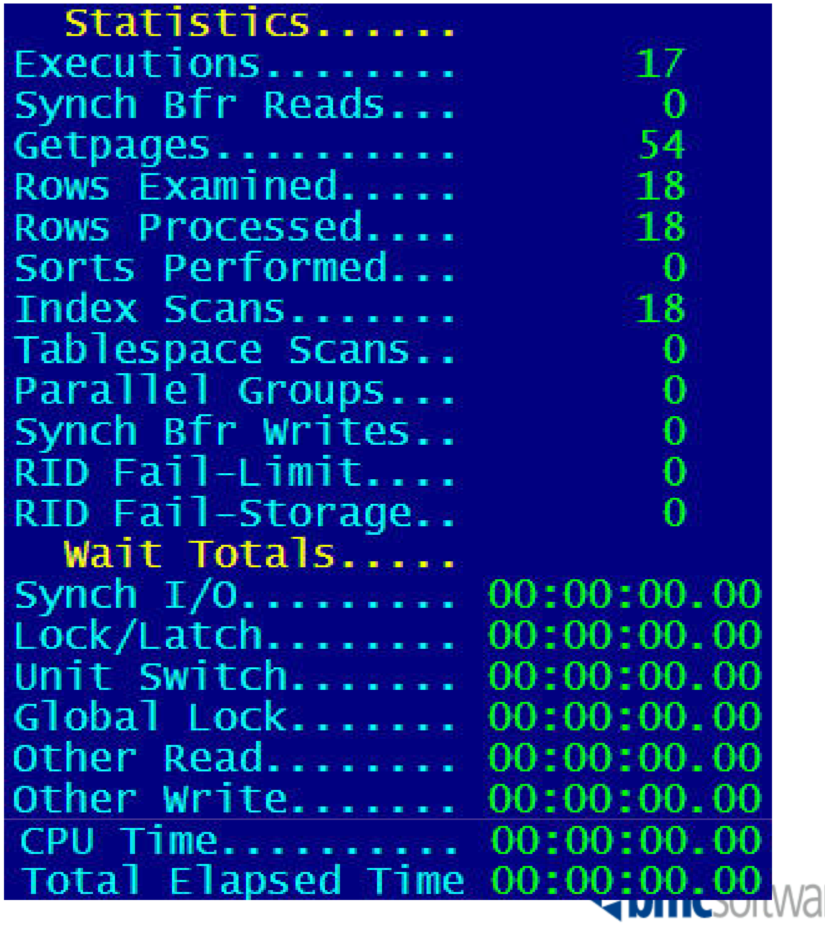
Reviewing Global Statement Cache Information IFCID 318



- Execution statistics for dynamic SQL statements
- Turn on collection with Monitor trace IFCID 318
  - Begins collecting statistics and accumulates them for the length of time the monitor trace is on
  - Stop Monitor trace resets all statistics
  - 2-4% overhead per dynamic SQL statement stored in the cache
- Recommended approach
  - Run the trace only when actively monitoring the cache
- Use EXPLAIN STMTCACHE to externalize data for evaluation

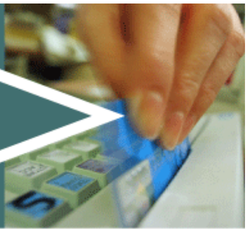


```
-START TRACE(MON)IFCID(318)
DSNW130I  *DHN1 MON TRACE STARTED, ASSIGNED TRACE NUMBER 06
DSN9022I  *DHN1 DSNWVCM1 '-START TRACE' NORMAL COMPLETION
```



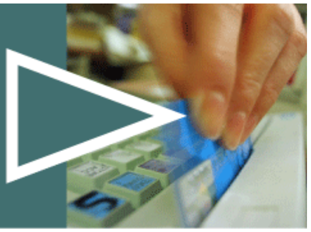
```
  Statistics......
Executions........        17
Synch Bfr Reads...         0
Getpages..........        54
Rows Examined.....        18
Rows Processed....        18
Sorts Performed...         0
Index Scans.......        18
Tablespace Scans..         0
Parallel Groups...         0
Synch Bfr Writes..         0
RID Fail-Limit....         0
RID Fail-Storage..         0
  Wait Totals.....
Synch I/O......... 00:00:00.00
Lock/Latch........ 00:00:00.00
Unit Switch....... 00:00:00.00
Global Lock....... 00:00:00.00
Other Read........ 00:00:00.00
Other Write....... 00:00:00.00
CPU Time.......... 00:00:00.00
Total Elapsed Time 00:00:00.00
```

Acknowledgements



- There are numerous documents that discuss SQL in general and dynamic SQL in particular, including:
  - DB2 technical publications
  - Technical articles by numerous DB2 Subject Matter Experts
  - IDUG List Server Archives
- IBM Redbooks on this topic were especially helpful in researching this presentation, including:
  - DB2 for z/OS and OS/390 : Squeezing the Most Out of Dynamic SQL
  - DB2 UDB for z/OS V8: Through the Looking Glass and What SAP Found There

Summary



- Dynamic SQL is growing in usage
  - ERP Vendors
  - Distributed applications
- DB2 offers multiple options for reducing the overhead traditionally associated with dynamic SQL
- These options include multiple types of statement caching
  - Local statement caching
  - Global statement caching
  - Full statement caching
- DB2 9 will see big changes in the way the SQL statement execution statistics discussed in this session will be used captured and used