PROGRESSIVE

Managing Distributed, Dynamic Workloads on DB2

Overview

- Our Environment
- Exploiting DB2 Traces to Manage Dynamic SQL
  - SMF
  - Performance Database – Summary Data
  - Detail Data
  - Traces
  - Buffer Pool Analysis
- Summary

Progressive Company Info

- Progressive has changed significantly from only a few years ago:
  - *New products, new means of distribution, new services, new facilities, and many new faces*
  - *Over 25,000 Progressive people.*
  - *Net income increased 88% over last year*
- Progressive is growing
  - *We grow as fast as possible, constrained only by our profit objective and our ability to provide high quality service*
- IT must support the business growth objectives

Progressive –

Quote from our 2003 note to shareholders:

In 1974, then CEO and now Chairman, Peter Lewis wrote ***"There is a simpler, easier way- - but it's not the Progressive way. Our way is not easy and we may even be crazy- - but we are going to do it and we will be proud."*** *Today, as then, our culture is defined by people who enjoy working hard, growing constantly, performing well and being rewarded competitively. Our objectives are demanding and hard to achieve, and our organization is proud when we do so.*

Glen Renwick, Progressive CEO

Production Gateway Configuration

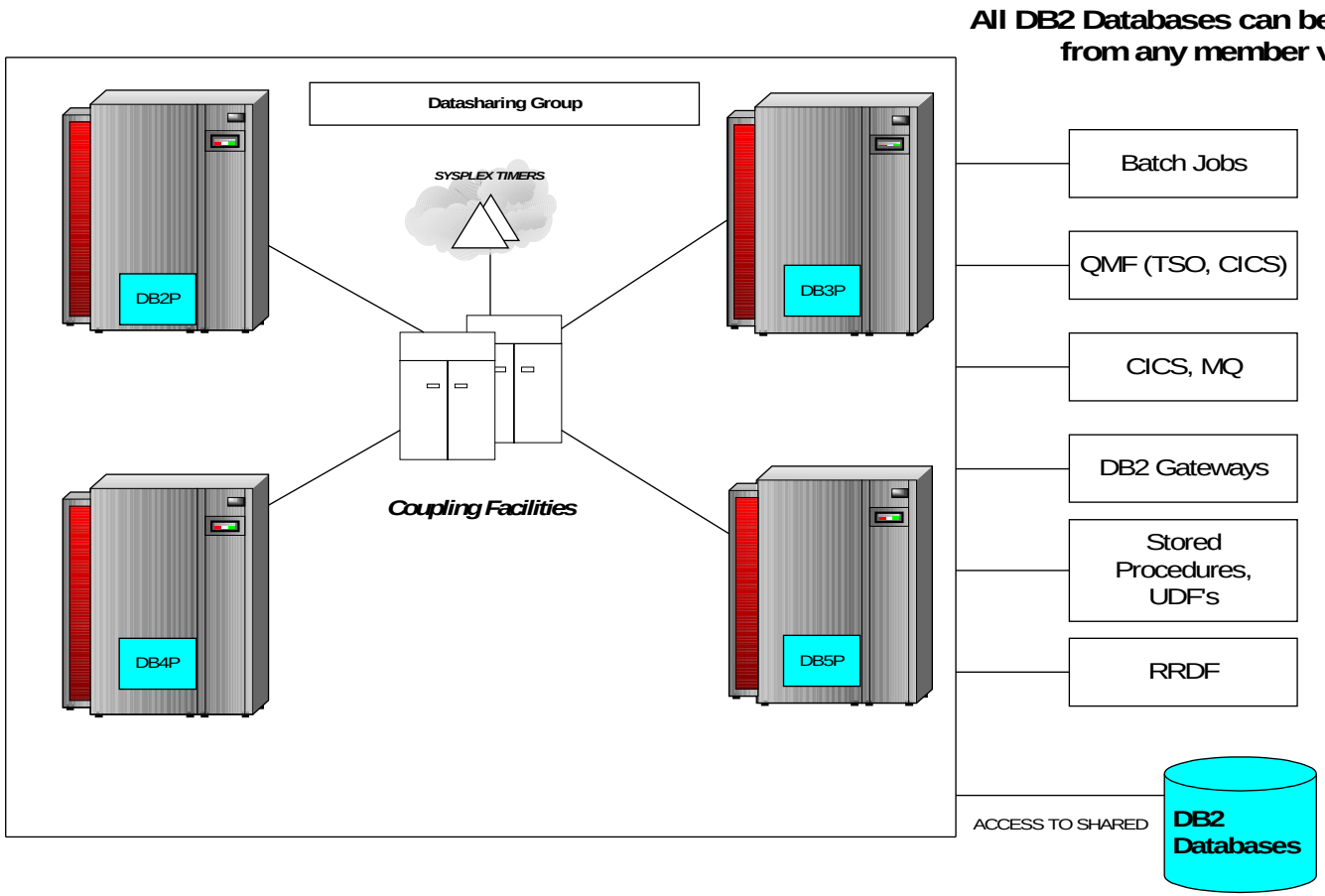
Decision Support Pool

Main Operational Pool

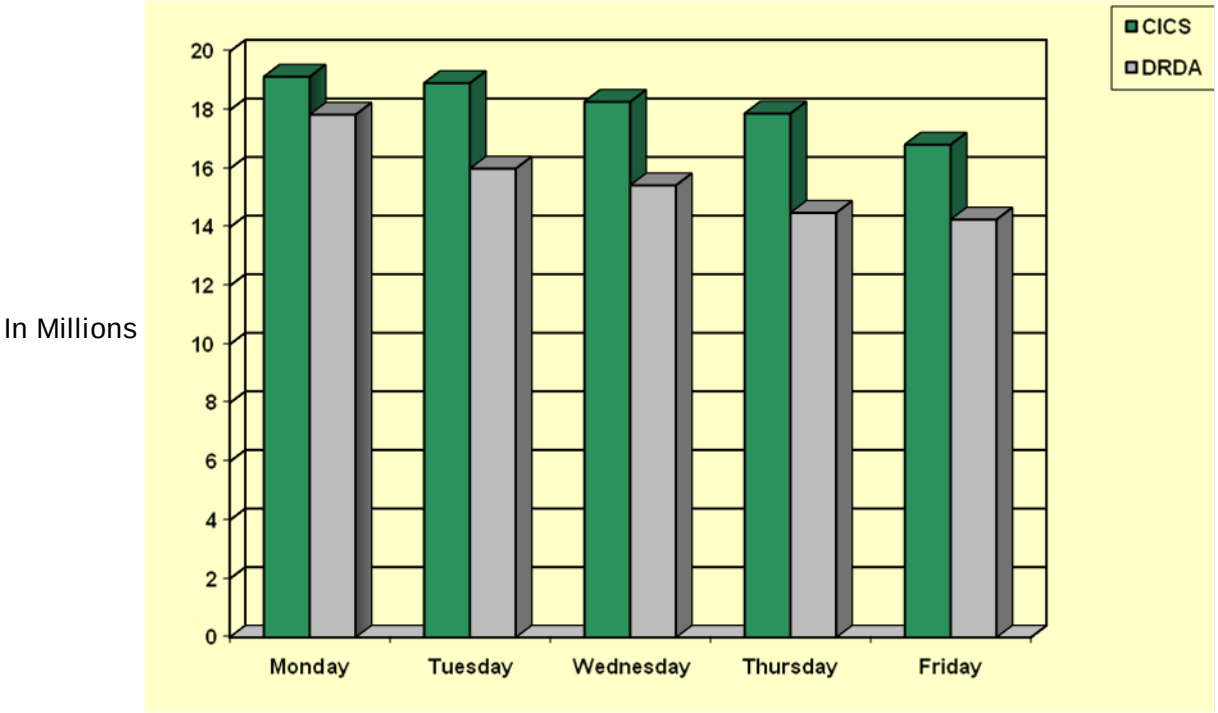
Operational Pool (Isolation)

Most of our distributed workload uses our main operational gateway pool. The isolation pool is used by one application. We will not focus on the decision support applications in this presentation.

DB2 ProdPlex Configuration



Traditional CICS vs Distributed Workload



The volume of distributed transactions is beginning to rival that of the CICS workload. However, numerous CICS transactions also use Dynamic SQL.

In an almost perfect world . . .

- A product that helps DBAs manage a large volume of Dynamic SQL should:
  - Quickly find problem SQL
  - User friendly

MIP Usage By Application

Based on a 15 minute interval

APPLICATION	TRANSACTIONS	MIPS
CLAIMS1	62,362	456.68
AGENT1	172,281	218.03
POLICY	23,523	54.877
CLAIMS2	762	54.091
CLAIMS3	3,895	35.153
AGENT2	44,688	151.15

MIP Usage Comparison

Based on a 15 minute interval

APPLICATION	TRANSACTIONS	AVERAGE CPU	MIPS per 100 trans
CLAIMS1	62,362	0.0369	0.732305
AGENT1	172,281	0.0064	0.126555
POLICY	23,523	0.0119	0.233291
CLAIMS2	762	0.3762	7.098556
CLAIMS3	3,895	0.0427	0.902516
AGENT2	44,688	.02	0.338234

MIPS per 100 transactions is an artificial measure, but this technique allows us to compare a distributed application to another application.

Loading the Performance Database

- The Summary Load process consists of multiple steps:
  - Save
  - Convert
  - Load

We summarize all DB2 SMF data on 15 minute intervals and keep between 1 and 2 months of accounting data.

Application Profiling Using Summary Data

- ACCT_GENERAL
  - Class 1, Class 2 and Class 3 times
- ACCT_BUFFER
- ACCT_PROGRAM
  - Class 7 and Class 8 times
- ACCT_DDF
- ACCT_GBUFFER
- ACCT_RLF

For the purposes of this presentation we will focus on the ACCT_GENERAL and ACCT_BUFFER tables.

Agent1 Application Summary Profile

Class 2 – Elapsed	30,219 Seconds
Class 2 – CPU	1,169 Seconds
Occurrences	194,291
Open, Fetch, Close	708,299
Insert, Update, Delete	51,474
Prep Statement Match	117,054
Prep statement no match	76,969

BUFFER POOL	GETPAGES
BP0	5,794,447
BP5 (SORT)	677,530
BP12	1,275,754
BP13	552,362

The profiles are built with simple queries that group by generic authids.

Agent1 Application Summary Profile

Class 2 – Elapsed	.16 Seconds
Class 2 – CPU	.006 Seconds
Occurrences	194,291
Open, Fetch, Close	4
Insert, Update, Delete	> 1
Prep Statement Match	N/A
Prep statement no match	N/A

BUFFER POOL	GETPAGES
BP0	30
BP5 (SORT)	3
BP12	7
BP13	3

Agent2 Application Summary Profile

Class 2 – Elapsed	1264 Seconds
Class 2 – CPU	710 Seconds
Occurrences	44,688
Open, Fetch, Close	265,380
Insert, Update, Delete	0
Prep Statement Match	19156
Prep statement no match	25457

BUFFER POOL	GETPAGES
BP0	3,396,288
BP5 (SORT)	290,472
BP12	437,942
BP13	272,596

Agent2 Application Summary Profile

Class 2 – Elapsed	.028 Seconds
Class 2 – CPU	.016 Seconds
Occurrences	44,688
Open, Fetch, Close	6
Insert, Update, Delete	0
Prep Statement Match	N/A
Prep statement no match	N/A

BUFFER POOL	GETPAGES
BP0	76
BP5 (SORT)	7
BP12	10
BP13	6

The Next Step:

- The Accounting Detail Tables
  - ACCT_GEN_DTL
    - Class 1, 2 and 3 times
  - ACCT_BUF_DTL
    - Contains 1 row for each buffer pool accessed for each thread
  - ACCT_PROG_DTL
    - Class 7, Class 8 times
  - ACCT_DDF_DTL
  - ACCT_GBUFFER_DTL
  - ACCT_RLF_DTL

Use the ACCOUNTING FILE command to create the load files for these tables. Sources of this data would include SMF or a trace with IFCID 3. The instance number and timestamp columns can be used to join the tables. Only the ACCT_GEN_DTL table contains correlation data.

Detail Analysis of CPU Usage

APPLICATION	CPU USAGE PROFILE
AGENT2	50% of transactions use 80% of CPU
CLAIMS1	12% of transactions use 50% CPU
CLAIMS2	25% of transactions use almost all of the CPU

Generally the 80/20 rule applies.....

Back to the Agent2 Application Profile

Class 2 – Elapsed	.028 Seconds
Class 2 – CPU	.016 Seconds
Occurrences	44,688
Open, Fetch, Close	6
Insert, Update, Delete	0
Prep Statement Match	N/A
Prep statement no match	N/A

BUFFER POOL	GETPAGES
BP0	76
BP5 (SORT)	7
BP12	10
BP13	6

Additional Detail Tables

There are other detail tables that can be loaded from record trace data. Two of these are:

- IFCID 22 - Mini-Bind
- IFCID 63 - SQLTEXT

Specialized Trace

- A record trace with just three IFCIDs will produce all the information needed to populate the detail tables. The trace can be qualified for an individual Authid or remote locations.
    - IFCID 3 - Accounting
    - IFCID 22 - Mini-bind
    - IFCID 63 – SQL Text

The Mini-bind information is essentially the explain for a given SQL statement.

Loading all the Detail Tables

We need to use the RECORD FILE and ACCOUNTING FILE commands to create two files for table loads.

- ACCOUNTING FILE output loads the accounting tables
- RECORD FILE output loads the SQLTEXT and Mind-Bind tables

Agent2 Application Profile

Class 2 – Elapsed	.028 Seconds
Class 2 – CPU	.016 Seconds
Occurrences	44,688
Open, Fetch, Close	6
Insert, Update, Delete	0
Prep Statement Match	N/A
Prep statement no match	N/A

BUFFER POOL	GETPAGES
BP0	76
BP5 (SORT)	7
BP12	10
BP13	6

Using the Detail Tables to find problem SQL

- MIP usage is high
- BP0 getpages are high relative to the other buffer pools
- Statement caching is not optimal
- CPU is high for some transactions – 50% of transactions use 80% of the CPU

The Culprit!!

The most frequently executed SQL statement had the form of:

Three Way Join Union…

This bad SQL was identified by joining the detail accounting general, accounting buffer, and SQL text tables.

```
SELECT
      G."TIMESTAMP"     AS ACCT_GEN_TSTAMP,
      G.INSTANCE_NBR    AS INSTANCE,
      G.PRIMAUTH,
      G.CLASS1_TIME_BEG AS BEGIN_TIME,
      G.CLASS1_TIME_END AS END_TIME,
      G.CLASS2_CPU_TOTAL,
      S.TIMESTAMP       AS SQL_TSTAMP,
      S.STATEMENT_TEXT
FROM ACCT_GEN_DTL G, SQLTEXT S, ACCT_BUFFER_DTL B
  WHERE  S.LUW_INSTANCE =  G.INSTANCE_NBR
    AND (G.INSTANCE_NBR =  B.INSTANCE_NBR AND
        G.TIMESTAMP =  B.TIMESTAMP)
    AND (S.TIMESTAMP >  G.CLASS1_TIME_BEG AND
        S.TIMESTAMP <  G.CLASS1_TIME_END )
    AND (B.BP_ID =  0 AND B.BP_GETPAGES  >  76  )
    ORDER BY  G.CLASS2_CPU_TOTAL DESC
```

CLAIMS2 Summary Data

Class 2 - Elapsed	1022 seconds
Class 2 – CPU	279 seconds
Occurrences	980

BUFFER POOL	GETPAGES	SYNC READ	ASYNC READ
0	79,716	378	0
3	344,095	12,163	71,012
4	3,375,667	63,448	767,002

Buffer Pool Analyzer

BPA is a component of DB2PE that enables the user to monitor, analyze and tune DB2 buffer pools. The product's flexibility differentiates it from other buffer pool tuning products.

BPA has many features but we are only interested in:

- Detail collection with record format standard
- Buffer pool activity reporting

There may be considerable overhead associated with the Detail collection.

BPA Detail Reporting

Global Include (Primauth (CLAIMS2))

BPACTIVITY

Report

Level (Detail)

Sort by (Planname, Getpage) Top 10))

BPACTIVITY reporting is extremely flexible. It supports input from SMF, GTF and, most importantly, data from DB2PE traces

BP Activity Report Sample

PLANNAME	DISTSERV	DISTSERV	DISTSERV	DISTSERV
BPID	BP4	BP4	BP4	BP3
QPAGESET	CDWDB003	CDWDB003	CDWDB003	CDWDB003
	CDWTS316	CDWTS306	CDWTS311	CDW311X1
BP Hit ratio(%)				
System	100.0	100.0	100.0	100.0
Application	100.0	94.1	80.0	98.7
Getpage	27468	4155	2423	2317
Sequential	27462	0	0	0
Random	6	2006	2395	2317
Ridlist	0	2149	28	0
Hit	27466	3911	1939	2286
Mss random	1	244	483	31
Mss asynch	1	0	1	0
Nbread	0	0	0	0
Read request	0	0	0	0
Synchronous	0	0	0	0
Seq prefetch	0	0	0	0
List pref	0	0	0	0
Dyn prefetch	0	0	0	0
Delay(msec)				
Synchronous	n/c	n/c	n/c	n/c
Seq pref	n/c	n/c	n/c	n/c
List pref	n/c	n/c	n/c	n/c
Dyn pref	n/c	n/c	n/c	n/c

"Combining" the Mini-Bind Data with BPA reports information

The BPACTIVITY report shows a significant amount of sequential access to one object:

Tablespace CDWDB003.CDWTS316

We can now use the data in the Mini-Bind Table to find statements that access this object and also use a lot of CPU. Additionally, we can find statements that access this object and have high getpage counts.

SQL Statement Analysis – The Results

The CLAIMS2 application actually had several problem queries. The issues included:

- Index tuning opportunities
- Mismatched data lengths

Use at Your Own "Risk"!!

There is a way to limit the buffer pool tracing and still gather the data that we need. Our initial trace included these IFCIDs:

- Accounting – IFCID 3
- Mind- Bind – IFCID 22
- SQL Text – IFCID 63

We will add two additional IFCIDS to the list:

- BP Getpages – IFCID 198
- DBID/ OBID Translation – IFCID 105

The trace will be qualified by an authid.

BP Activity Report with GP Trace

PLANNAME	DISTSERV	DISTSERV	DISTSERV	DISTSERV
BPID	BP4	BP4	BP4	BP3
QPAGESET	CDWDB003 CDWTS316	CDWDB003 CDWTS306	CDWDB003 CDWTS311	CDWDB003 CDW311X1
BP Hit ratio(%)				
System	100.0	100.0	100.0	100.0
Application	100.0	94.1	80.0	98.7
Getpage	27468	4155	2423	2317
Sequential	27462	0	0	0
Random	6	2006	2395	2317
Ridlist	0	2149	28	0
Hit	27466	3911	1939	2286
Miss random	1	244	483	31
Miss asynch	1	0	1	0
Noread	0	0	0	0
Read request	0	0	0	0
Synchronous	0	0	0	0
Seq prefetch	0	0	0	0
List pref	0	0	0	0
Dyn prefetch	0	0	0	0
Delay(msec)				
Synchronous	n/c	n/c	n/c	n/c
Seq pref	n/c	n/c	n/c	n/c
List pref	n/c	n/c	n/c	n/c
Dyn pref	n/c	n/c	n/c	n/c

The Complete Process

- Run a specialized trace
  - IFCIDs 3, 22,63,198 and 105 – for a specific Authid
- Load the detail tables
  - Accounting general and buffer
  - SQL_Text
  - Mini_Bind
- Run a BPA detail report.

Then analyze the BPA report and/or run a set of canned queries to find the bad SQL

The Results

- Better SQL
- Transaction performance improvements
- CPU savings

In Conclusion . . .

- A product that helps DBAs manage a large volume of Dynamic SQL should:
  - Quickly find problem SQL: YES
  - User friendly: WELL, MAYBE . . .

Questions ?

Samples – In SFPESAMP

For the Performance Database:

Members DGOACS* have the table DDL

Members DGOALS* contain the Load statements

Members DGOABS* have the field definitions. They
describe the contents of the data in each table.

Samples – In SFPESAMP

For the Accounting detail tables:

Members DGOACF* have the table DDL

Members DGOALF* contain the Load statements

Members DGOABF* have the field definitions. They
describe the contents of the data in each table.

Samples – In SFPESAMP

For the other detail tables:

Members DGONCF* have the table DDL

Members DGONLF* contain the Load statements

Members DGONBF* have the field definitions. They
describe the contents of the data in each table.

Samples

Once you run the trace for IFCIDs 3,22,63,105 & 198 run a job like this to load all the tables and
run the BPA report.
Step 1 – Create the load files.

```
//DB2PMR   EXEC PGM=DB2PM,REGION=0M
//STEPLIB  DD  DISP=(SHR,KEEP,KEEP),
//             DSN=SYS44.SFPELOAD
//INPUTDD  DD  DISP=(SHR,KEEP,KEEP),
//             DSN=<your trace dataset>
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//ACRPTDD  DD  SYSOUT=*
//STRPTDD  DD  SYSOUT=*
//JOBSUMDD DD  SYSOUT=*
//RTFILDD1 DD  DISP=(NEW,CATLG,DELETE),
//         DSN=&&TRACFILE,
//         UNIT=SYSALLDA,
//         SPACE=(CYL,(100,50),RLSE),
```

Samples

```
//          DCB=(RECFM=VB,LRECL=9072,BLKSIZE=32756)
//ACFILDD1  DD   DISP=(NEW,CATLG,DELETE),
//          DSN=&&ACCTFILE,
//          UNIT=SYSALLDA,
//          SPACE=(CYL,(100,10),RLSE),
//          DCB=(RECFM=VB,LRECL=9072,BLKSIZE=32756)
//DPMPARMS DD   DISP=(SHR,KEEP,KEEP),
//          DSN=SYS44.DPMPARMS
//SYSIN     DD   *
  ACCOUNTING
     FILE
 RECORD
     FILE
 EXEC
```

Samples

Step 2 – Run the BPACTIVITY report

```
//DB2BP   EXEC PGM=DB2BP
//STEPLIB  DD  DISP=(SHR,KEEP,KEEP),
//             DSN=SYS44.SFPELOAD
//INPUTDD  DD  DISP=SHR,DSN=<your trace dataset>
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//BPFILDD1 DD  DUMMY
//JOBSUMDD DD  SYSOUT=*
//DPMLOG   DD  SYSOUT=*
//BPAREP2  DD  SYSOUT=*
//SYSIN    DD  *
*    GLOBAL COMMAND TO ADJUST REPORTED GMT TO
     LOCAL TIME
```

Samples

```
GLOBAL
      TIMEZONE (-4:00)
BPACTIVITY
REPORT
         LEVEL(DETAIL)
         ORDER(PLANNAME- BPID- QPAGESET
      SORTBY(PLANNAME,GETPAGE) TOP(50))
```

Samples

Step 3

Load the ACCT_GEN_DTL and ACCT_BUFFER_DTL Tables with &&ACCTFILE.

Load parms for ACCT_GEN_DTL are in SFPESAMP(DGOALFGE)

Load parms for ACCT_BUFFER_DTL are in SFPESAMP(DGOALFBU)

Step 4

Load the Mini_bnd and SQLTEXT tables with &&TRACFILE.

Load parms for Mini_bnd are in SFPESAMP(DGONLFMB)

Load parms for SQLTEXT are in SFPESAMP(DGONLFSQ)

Samples

- Join the Detail accounting tables on instance and timestamp. You may need to include the fully qualified network name, but instance and timestamp should do the trick.
- The other two tables do not have timestamps that match the ones on the accounting tables. You have to use the CLASS1_TIME_BEG and CLASS1_TIME_END columns from the ACCT_GEN_DTL table to match the statements and mini-bind data to the accounting data.
- You can also run a RECTRACE report with the trace file to make sure that your queries are valid.
- Use these parms for the RECTRACE:

```
GLOBAL
    INCLUDE (INSTANCE(<instance nbr>) IFCID(3,22,63))
 RECTRACE
  TRACE
    LEVEL (LONG)
    SORTBY (TIMESTAMP)
  EXEC
```

The report can help sort out any issues.

Samples

Sample query for CPU usage

```
SELECT
      G."TIMESTAMP"       AS ACCT_GEN_TSTAMP,
      G.INSTANCE_NBR      AS ACCT_INSTANCE,
      G.PRIMAUTH,
      G.CLASS1_TIME_BEG AS ACCT_BEGIN_TIME,
      G.CLASS1_TIME_END AS ACCT_END_TIME,
      G.CLASS2_CPU_TOTAL,
      S.TIMESTAMP         AS SQL_TEXT_TSTAMP,
      S.STATEMENT_TEXT
FROM ACCT_GEN_DTL G, SQLTEXT S
  WHERE  S.LUW_INSTANCE =  G.INSTANCE_NBR
    AND (S.TIMESTAMP >  G.CLASS1_TIME_BEG AND
        S.TIMESTAMP <  G.CLASS1_TIME_END )
    AND G.CLASS2_CPU_TOTAL >  .3
ORDER BY G.CLASS2_CPU_TOTAL DESC
```

Samples

Sample query for buffer pool usage

```
SELECT
      G."TIMESTAMP"      AS ACCT_GEN_TSTAMP,
      G.INSTANCE_NBR     AS INSTANCE,
      G.PRIMAUTH,
      G.CLASS1_TIME_BEG AS BEGIN_TIME,
      G.CLASS1_TIME_END AS END_TIME,
      G.CLASS2_CPU_TOTAL,
      S.TIMESTAMP        AS SQL_TSTAMP,
      S.STATEMENT_TEXT
FROM ACCT_GEN_DTL G, SQLTEXT S, ACCT_BUFFER_DTL B
  WHERE  S.LUW_INSTANCE =  G.INSTANCE_NBR
    AND (G.INSTANCE_NBR =  B.INSTANCE_NBR AND
        G.TIMESTAMP =  B.TIMESTAMP)
    AND (S.TIMESTAMP >  G.CLASS1_TIME_BEG AND
        S.TIMESTAMP <  G.CLASS1_TIME_END )
    AND (B.BP_ID =  4 AND B.BP_GETPAGES  >  3000   )
    ORDER BY  G.CLASS2_CPU_TOTAL DESC
```

Samples

Sample query based on use of an Object:

```
SELECT
      G."TIMESTAMP",
      G.INSTANCE_NBR,
      G.PRIMAUTH,
      G.CLASS1_TIME_BEG, G.CLASS1_TIME_END,
      G.CLASS2_CPU_TOTAL,
      S.TIMESTAMP,
      S.STATEMENT_TEXT
FROM ACCT_GEN_DTL G, SQLTEXT S,
    ACCT_BUFFER_DTL B , MINIPLN M
  WHERE  S.LUW_INSTANCE =  G.INSTANCE_NBR
    AND (S.LUW_INSTANCE =  M.LUW_INSTANCE AND
        S.TIMESTAMP =  M.SQLTEXT_TIMESTAMP)
    AND (G.INSTANCE_NBR =  B.INSTANCE_NBR AND
        G.TIMESTAMP =  B.TIMESTAMP)
    AND (S.TIMESTAMP >  G.CLASS1_TIME_BEG AND
        S.TIMESTAMP <  G.CLASS1_TIME_END )
    AND (B.BP_ID =  4 AND B.BP_GETPAGES  >  3000)
 AND M.TNAME =          '<table name>'       (use ACCESSNAME for indexes)
  ORDER BY G.CLASS2_CPU_TOTAL DESC
```