

25 years of missed opportunities? SQL Tuning Revisited

SOFTWARE ENGINEERING GmbH

Platform: z/OS DB2



AGENDA

1. Tuning SQL

- How we have always done it
- Single SQL, Package, Application...
- Year 2004 – AP comparisons and simulation

2. Tuning SQL Revisited – A new methodology

3. Harvesting the low hanging fruit

Tuning SQL – How we have always done it

- Get an SQL from development
- EXPLAIN it
- Tune it if required
- Stage to next Level (Dev -> QA, QA -> Prod)
- Fire fight

Single SQL, Package, Application...

- Get an SQL, Package or list of Packages from development
- Fight for (and against!) Dynamic SQL
- EXPLAIN them all
- See if any have gone belly up
- Tune it if required and if you have the time
- Stage to next Level (Dev -> QA, QA -> Prod)
- Fire fight

Tuning SQL - Year 2004

- Get an SQL, Package or list of Packages from development
- Propagate Production statistics down the environment chain (Prod -> QA, Prod -> Dev)
- Simulate Hardware, ZPARMS, and BUFFERPOOLS
- Fight for (and against!) Dynamic SQL
- EXPLAIN them all
- Compare with existing Access Paths – Reject any that have got worse
- Tune it if required and if you have the time
- Stage to next Level (Dev -> QA, QA -> Prod)
- Fire fight

Tuning SQL Revisited

- Get *all* Dynamic and Static SQL running in the Plex
- Propagate Production statistics down the environment chain (Prod -> QA, Prod -> Dev)
- Simulate Hardware, ZPARMS, and BUFFERPOOLS
- EXPLAIN them all
- Compare with existing Access Paths – Tune any that have got worse
 - Pick the „low hanging fruit“
- Stage to next Level (Dev -> QA, QA -> Prod)

Tuning SQL Revisited

So how to get there?

1. Collect as much data as you can
2. Store it in a Data Warehouse
3. Analyze it
4. Take Actions!

Collect as much data as you can

- How many resources do you spend on capturing DB2 SQL workload and its metrics?
- There seems to be out-of-the-box metrics delivered by DB2, but does it give me all the data I need, when I need it?
- How does the smarter database, how does DB2 10, or 11 for z/OS deal with it?...

Collect as much data as you can

- DB2 10 Monitoring Enhancements and Changes:
 - **Statement Level Statistics**
 - Enhanced messages and traces to capture statement level information
 - **Statement information in real-time**
 - STMT_ID – unique statement identifier assigned when statement first inserted into DSC
 - Statement type – static or dynamic
 - Bind TS – 10 byte TS when stmt was bound, or prepared
 - **Statement level execution statistics (per execution)**
 - New Monitor class 29 for statement detail level monitoring
 - **Monitor Class 29 (overhead is ~1-3%)**
 - New for statement level detail

Collect as much data as you can

What's exactly new since DB2 10:



- IFCID 316 was enhanced to externalize the data from the Dynamic Statement Cache (DSC) when a flushing situation occurs (LRU, RUNSTATs, ALTER, DROP, REVOKE, ...)
– NO DATA LOSS



- New IFCIDs 400* and 401 additionally EDM pool data – let's call it the **Static Statement Cache**
 - Memory resident storage of static SQL statements
 - Like with the enhanced 316, data is externalized when the EDM pool is full.
– NO DATA LOSS

*This IFCID is not really an IFCID but more of a „switch“ to enable externalization of static SQL metrics

Collect as much data as you can

DSC and EDM provide detailed workload insights:

- SQL text
- Statement ID
- Date/time
- Current status
- Resource consumption
- Identification/environmental data



Collect as much data as you can

DB2 10 also introduced 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

Collect as much data as you can

- 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/OR-ing could not be determined

Collect as much data as you can

Counters # EXECUTIONS OF THE STATEMENT. FOR A CURSOR STATEMENT, THIS IS THE # OF OPENS. # OF SYNCHRONOUS BUFFER READS PERFORMED FOR STATEMENT. # OF GETPAGE OPERATIONS PERFORMED FOR STATEMENT. # OF ROWS EXAMINED FOR STATEMENT. # OF ROWS PROCESSED FOR STATEMENT - FOR EXAMPLE, THE # OF ROWS RETURNED FOR A SELECT, OR THE NUMBER OF ROWS AFFECTED BY AN INSERT, UPDATE, OR DELETE. # OF SORTS PERFORMED FOR STATEMENT. # OF INDEX SCANS PERFORMED FOR STATEMENT. # OF TABLESPACE SCANS PERFORMED FOR STATEMENT. * # OF PARALLEL GROUPS CREATED FOR STATEMENT. # OF SYNCHRONOUS BUFFER WRITE OPERATIONS PERFORMED FOR STATEMENT. # OF TIMES THAT RID LIST RETRIEVAL FOR MULTIPLE INDEX ACCESS WAS NOT DONE BECAUSE DB2 COULD DETERMINE THE OUTCOME OF INDEX ANDING OR ORING*.

O Counters # OF TIMES THAT A RID LIST WAS NOT USED BECAUSE THE # OF RIDS EXCEEDED ONE OR MORE INTERNAL DB2 LIMITS, AND THE # OF RID BLOCKS EXCEEDED THE VALUE OF SUBSYSTEM PARAMETER MAXTEMP. RID. # OF TIMES THAT A RID LIST WAS NOT USED BECAUSE NOT ENOUGH STORAGE WAS AVAILABLE TO HOLD THE RID LIST, OR WORK FILE STORAGE OR RESOURCES WERE NOT AVAILABLE. # OF TIMES THAT A RID LIST OVERFLOWED TO A WORK FILE BECAUSE NO RID POOL STORAGE WAS AVAILABLE TO HOLD THE LIST OF RIDS*. # OF TIMES THAT A RID LIST OVERFLOWED TO A WORK FILE BECAUSE THE NUMBER OF RIDS EXCEEDED ONE OR MORE INTERNAL LIMITS*. # OF TIMES THAT APPENDING TO A RID LIST FOR A HYBRID JOIN WAS INTERRUPTED BECAUSE NO RID POOL STORAGE WAS AVAILABLE TO HOLD THE LIST OF RIDS*. # OF TIMES THAT APPENDING TO A RID LIST FOR A HYBRID JOIN WAS INTERRUPTED BECAUSE THE NUMBER OF RIDS EXCEEDED ONE OR MORE INTERNAL LIMITS*.

TIMINGS ACCUMULATED CPU TIME. THIS VALUE INCLUDES CPU TIME THAT IS CONSUMED ON AN IBM SPECIALTY ENGINE. ACCUMULATED ELAPSED TIME USED FOR STATEMENT. ACCUMULATED WAIT TIME FOR LATCH REQUESTS*. ACCUMULATED WAIT TIME FOR PAGE LATCHES*. ACCUMULATED WAIT TIME FOR DRAIN LOCKS*. ACCUMULATED WAIT TIME FOR DRAINS DURING WAITS FOR CLAIMS TO BE RELEASED*. ACCUMULATED WAIT TIME FOR LOG WRITERS. ACCUMULATED WAIT TIME FOR SYNCHRONOUS I/O. ACCUMULATED WAIT TIME FOR LOCK REQUESTS. ACCUMULATED WAIT TIME FOR A SYNCHRONOUS EXECUTION UNIT SWITCH. ACCUMULATED WAIT TIME FOR GLOBAL LOCKS. ACCUMULATED WAIT TIME FOR READ ACTIVITY THAT IS DONE BY ANOTHER THREAD. ACCUMULATED WAIT TIME FOR WRITE ACTIVITY THAT IS DONE BY ANOTHER THREAD.

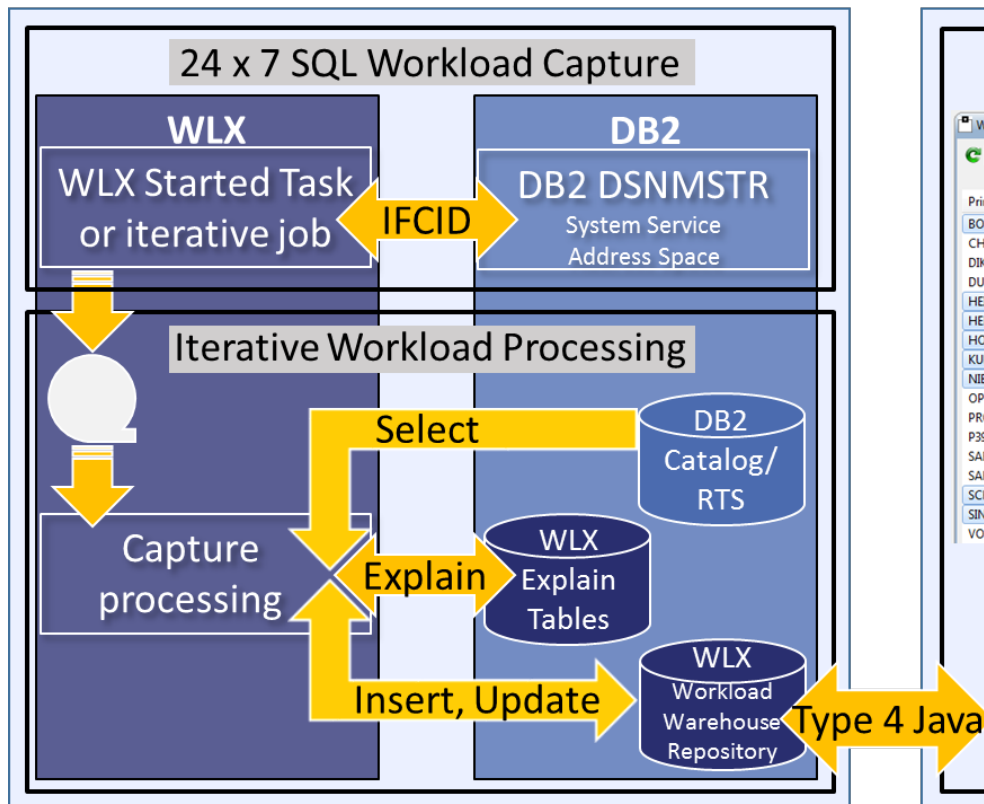
IDENTIFICATION DATA SHARING MEMBER THAT CACHED THE SQL STATEMENT*. PROGRAM NAME. PROGRAM NAME IS THE NAME OF THE PACKAGE OR DBRM THAT PERFORMED THE PREPARE/SQL. PRECOMPILER LINE NUMBER FOR THE PREPARE STATEMENT OR SQL STATEMENT. TRANSACTION NAME. THIS VALUE IS PROVIDED DURING RRS SIGNON OR RE-SIGNON. END USER ID*. THIS VALUE IS PROVIDED DURING RRS SIGNON OR RE-SIGNON. WORKSTATION NAME*. THIS VALUE IS PROVIDED DURING RRS SIGNON OR RE-SIGNON. USER ID. USER ID IS THE PRIMARY AUTH. ID OF THE USER WHO DID THE INITIAL PREPARE. USER GROUP. USER GROUP IS THE CURRENT SQLID OF THE USER WHO DID THE INITIAL PREPARE. USER-PROVIDED IDENTIFICATION STRING.

ENVIRONMENTAL REFERENCED TABLE NAME. FOR STATEMENTS THAT REFERENCE MORE THAN ONE TABLE, ONLY THE NAME OF THE FIRST TABLE THAT IS REFERENCED IS REPORTED. (ALL REFERENCED OBJECTS ARE STORED IN THE WLI DATA MODEL) LITERAL REPLACEMENT FLAG*. CURRENT SCHEMA. QUALIFIER THAT IS USED FOR UNQUALIFIED TABLE NAMES. BIND OPTIONS: ISOLATION, CURRENT DATA, AND DYNAMIC RULES. SPECIAL REGISTER VALUES: CURRENT DEGREE, CURRENT RULES, AND CURRENT PRECISION. WHETHER THE STATEMENT CURSOR IS A HELD CURSOR. TIMESTAMP WHEN STATISTICS COLLECTION BEGAN. DATA COLLECTION BEGINS WHEN A TRACE FOR IFCID 318 IS STARTED. DATE AND TIME WHEN THE STATEMENT WAS INSERTED INTO THE CACHE IN STORE CLOCK FORMAT. DATE AND TIME WHEN THE STATEMENT WAS UPDATED, IN STORE CLOCK FORMAT. DATE AND TIME WHEN THE STATEMENT WAS UPDATED, IN INTERNAL FORMAT.

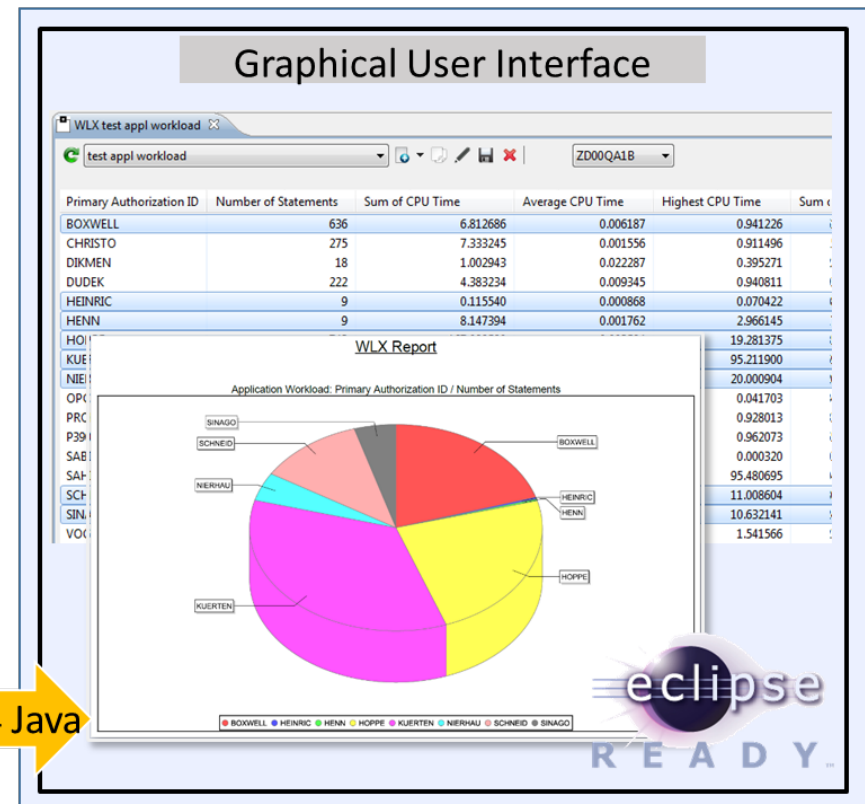
Store it in a SQL Workload Data Warehouse

Captures the hard to get SQLs,
even the ones that disappear ...

Mainframe Engine



Workstation Engine



Store it in a SQL Workload Data Warehouse

Capture the data e.g. using a STC:

Run a started task 24x7 to catch all the IFCIDs that DB2 will be throwing and store the data.

Process the workload:

Externalize and process the data, such as every 60 min:



- customizable (e.g. 30 - 180 minutes)
- allow Ad hoc data refresh triggered via operator command for the started task (MODIFY)
- capture the SQL Text at trace time
- gather additional catalog and RTS data
- add explain data if needed



Store it in a SQL Workload Data Warehouse

Use a GUI front end, preferably Eclipse:

Exploit and integrate into Eclipse based GUI front ends

- GUIs can come as a Plug-in for
 - IBM Rational
 - IBM Data Studio
 - Eclipse native
- Existing DB2 connections are used to connect to the mainframe
- Interactive dialogs allow complex and powerful analysis
- Export features can create PDF reports and allow MS Excel hand over
- Additional plug-ins interface with other tools,
such as  **SQL PerformanceExpert** (SPX) and  **Bind ImpactExpert** (BIX)

Store it in a SQL Workload Data Warehouse

Make the SQL Workload Warehouse Repository a set of DB2 tables that can also be created in LUW on a x86 server (E.g. DB2 Express-C).

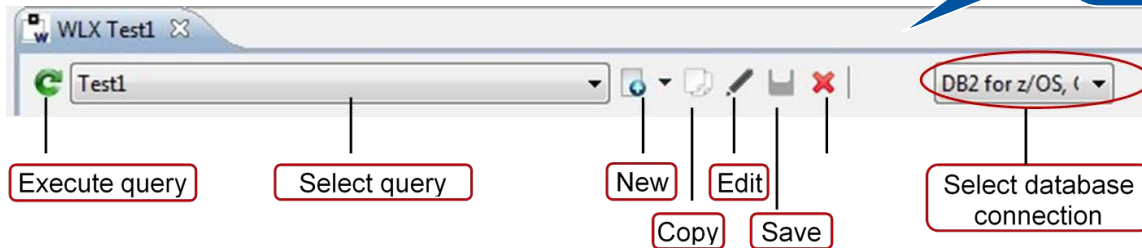
If this is done then you can simply unload from the z/OS DB2 tables and then load the LUW Tables directly from within the GUI which enables you to run all the analytics queries “locally”.

This can obviously save a lot of space on the z/OS side!

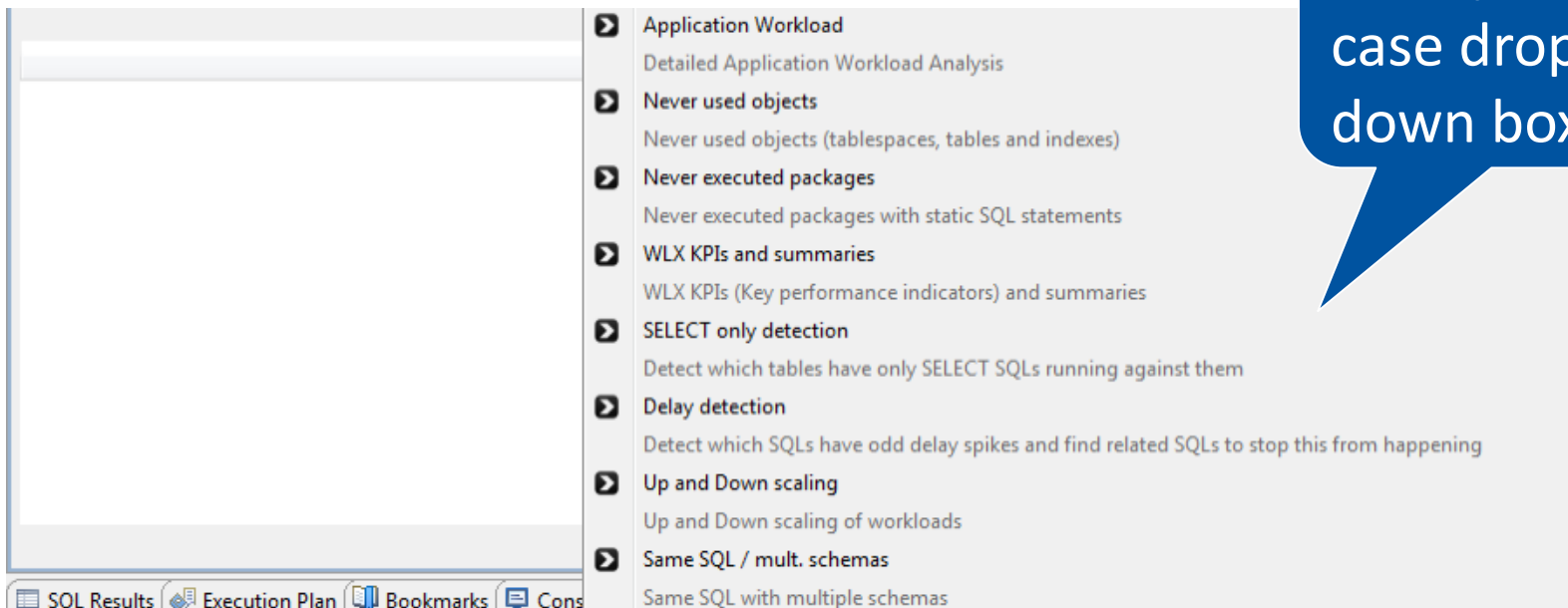
And remember that all of the Type 4 JAVA SQL is ZiiP eligible!

Analyze it

GUI features –
button overview



Example use
case drop
down box



Analyze it

Example of application workload and SQL text drill down

Database Development - Eclipse

File Edit Navigate Search Project Run Window Help

newQuery12

Primary Authorization ID	CPU Time	Elapsed Time	Executions	GETPAGES	Synchronous Buffer Reads	Synchronous Buffer Writes	Rows examined	Rows processed	Index scans	WF and Tablespace Scans	Sorts performed	Parallel Group
BOXWELL	6.562974	8.074171	1	13,922	481	0	766,371	3	0	1	1	
BOXWELL	5.982266	17.237795	2	21,640	98	0	881,114	20	0	2	2	
BOXWELL	4.850175	5.619810	1	9,298	13	0	288,444	42	255	8	6	
BOXWELL	4.344587	5.979491	1	13,922	0	0	766,371	12	0	1	1	
BOXWELL	2.736454	3.798705	2	7,059	84	0	170,121	1,699	11,779	0	0	
BOXWELL	1.568500	1.926214	1	3,529	17	0	85,061	1	5,890	0	0	
BOXWELL	1.030146	1.465298	3	7,223	70	0	4,623	9	18	18	12	
BOXWELL	0.909670	1.002262	1	2,792	0	0	0	12	8	6	4	
BOXWELL	0.843634	0.939291	2	3,779	12	0	0	266	4	4	4	
BOXWELL	0.607773	3.096399	1	4,571	442	0	150,679	10	0	1	0	
BOXWELL	0.392377	1.753672	2	2,418	697	0	2,320	24	1,288	12	8	
BOXWELL	0.384967	1.113439	1	2,660	28	0	88,236	963	963	1	0	
BOXWELL	0.379904	0.606013	1	4,974	0	0	265,800	10	0	1	0	
BOXWELL	0.367102	0.411977	3	474	0	0	69	33	75	18	12	

Result counter: 545

Application Workload

Connection profile

Type: Name: Database: Status: Disconnected, Auto Commit

```
WITH
PRIME_KEY (WLX_TIMESTAMP, STMT_GROUP_SSID, STMT_ID, STMT_ORIGIN,
STMT_TIMESTAMP, STMT_TYPE, SCALE_ADJ) AS
(SELECT WLX_TIMESTAMP, STMT_GROUP_SSID, STMT_ID, STMT_ORIGIN, STMT_TIMESTAMP,
STMT_TYPE, CASE WHEN STMT_TIMESTAMP >= (SELECT MAX(IQAO610.WLXT009.WLX_TIMESTAMP)
FROM IQAO610.WLXT009) - 1 HOURS THEN 3.6E+03 / CASE TIMESTAMPOFF(2, CAST (STMT_STATS_UPD - STMT_TIMESTAMP AS CHAR (22))) WHEN 0 THEN 1 ELSE TIMESTAMPOFF(2, CAST (STMT_STATS_UPD - STMT_TIMESTAMP AS CHAR (22)))
WHERE EXECUTIONS > 2
AND ((1 = 1)
AND (WLX_TIMESTAMP = (SELECT MAX(IQAO610.WLXT009.WLX_TIMESTAMP)
FROM IQAO610.WLXT009))))))
SELECT *
FROM (SELECT DECIMAL(K.SCALE_ADJ, 9, 3) AS SCALE_ADJ, DECIMAL((A.CPU_TIME * 1.000000) / 1E+06, 12, 6) AS CPU_TIME,
DECIMAL((A.CPU_TIME * 1E+02) / (CASE W.DCPU_TIME WHEN 0 THEN 1 ELSE W.DCPU_TIME END),
12, 6) AS PCT_CPU_TIME, CASE WHEN A.STMT_TIMESTAMP >= (SELECT MAX(IQAO610.WLXT009.WLX_TIMESTAMP)
FROM IQAO610.WLXT009) - 1 HOURS THEN DECIMAL(A.CPU_TIME * K.SCALE_ADJ * 1.000000 / 1E+06, 12, 6) ELSE DECIMAL(A.CPU_TIME / (CASE K.SCALE_ADJ WHEN 0 THEN 1 ELSE K.SCALE_ADJ END) * 1.000000 / 1E+06,
12, 6) END AS CPU_TIME_ADJ, A.GETP_OPERATIONS, DECIMAL((A.GETP_OPERATIONS * 1E+02) / (CASE W.DGETP_OPERATIONS WHEN 0 THEN 1 ELSE W.DGETP_OPERATIONS END),
12, 6) AS PCT_GETP_OPERATIONS, CASE WHEN A.STMT_TIMESTAMP >= (SELECT MAX(IQAO610.WLXT009.WLX_TIMESTAMP)
FROM IQAO610.WLXT009) - 1 HOURS THEN BIGINT(A.GETP_OPERATIONS * K.SCALE_ADJ) ELSE BIGINT(A.GETP_OPERATIONS / (CASE K.SCALE_ADJ WHEN 0 THEN 1 ELSE K.SCALE_ADJ END)) END AS GETP_OPERATIONS_ADJ,
DECIMAL((A.ELAPSED_TIME * 1.000000) / 1E+06, 12, 6) AS ELAPSED_TIME,
DECIMAL((A.ELAPSED_TIME * 1E+02) / (CASE W.DELAPSED_TIME WHEN 0 THEN 1 ELSE W.DELAPSED_TIME END),
12, 6) AS PCT_ELAPSED_TIME, CASE WHEN A.STMT_TIMESTAMP >= (SELECT MAX(IQAO610.WLXT009.WLX_TIMESTAMP)
FROM IQAO610.WLXT009) - 1 HOURS THEN DECIMAL((A.ELAPSED_TIME * K.SCALE_ADJ) * 1.000000 / 1E+06, 12,
6) ELSE DECIMAL(A.ELAPSED_TIME / (CASE K.SCALE_ADJ WHEN 0 THEN 1 ELSE K.SCALE_ADJ END) * 1.000000 / 1E+06,
12, 6) END AS ELAPSED_TIME_ADJ, A.EXECUTIONS, DECIMAL((A.EXECUTIONS * 1E+02) / (CASE W.DEXECUTIONS WHEN 0 THEN 1 ELSE W.DEXECUTIONS END),
12, 6) AS PCT_EXECUTIONS, CASE WHEN A.STMT_TIMESTAMP >= (SELECT MAX(IQAO610.WLXT009.WLX_TIMESTAMP)
FROM IQAO610.WLXT009) - 1 HOURS THEN DECIMAL(A.EXECUTIONS * K.SCALE_ADJ * 1.000000 / 1E+06, 12, 6) ELSE DECIMAL(A.EXECUTIONS / (CASE K.SCALE_ADJ WHEN 0 THEN 1 ELSE K.SCALE_ADJ END) * 1.000000 / 1E+06,
12, 6) END AS EXECUTIONS_ADJ, A.WF_AND_TABLESPACE_SCANS, DECIMAL((A.WF_AND_TABLESPACE_SCANS * 1E+02) / (CASE W.DWF_AND_TABLESPACE_SCANS WHEN 0 THEN 1 ELSE W.DWF_AND_TABLESPACE_SCANS END),
12, 6) AS PCT_WF_AND_TABLESPACE_SCANS, CASE WHEN A.WF_AND_TABLESPACE_SCANS >= (SELECT MAX(IQAO610.WLXT009.WF_AND_TABLESPACE_SCANS)
FROM IQAO610.WLXT009) - 1 HOURS THEN DECIMAL(A.WF_AND_TABLESPACE_SCANS * K.SCALE_ADJ * 1.000000 / 1E+06, 12, 6) ELSE DECIMAL(A.WF_AND_TABLESPACE_SCANS / (CASE K.SCALE_ADJ WHEN 0 THEN 1 ELSE K.SCALE_ADJ END) * 1.000000 / 1E+06,
12, 6) END AS WF_AND_TABLESPACE_SCANS_ADJ, A.SORTS_PERFORMED, DECIMAL((A.SORTS_PERFORMED * 1E+02) / (CASE W.DSORTS_PERFORMED WHEN 0 THEN 1 ELSE W.DSORTS_PERFORMED END),
12, 6) AS PCT_SORTS_PERFORMED, CASE WHEN A.SORTS_PERFORMED >= (SELECT MAX(IQAO610.WLXT009.SORTS_PERFORMED)
FROM IQAO610.WLXT009) - 1 HOURS THEN DECIMAL(A.SORTS_PERFORMED * K.SCALE_ADJ * 1.000000 / 1E+06, 12, 6) ELSE DECIMAL(A.SORTS_PERFORMED / (CASE K.SCALE_ADJ WHEN 0 THEN 1 ELSE K.SCALE_ADJ END) * 1.000000 / 1E+06,
12, 6) END AS SORTS_PERFORMED_ADJ, A.PARALLEL_GROUPS, DECIMAL((A.PARALLEL_GROUPS * 1E+02) / (CASE W.DPARALLEL_GROUPS WHEN 0 THEN 1 ELSE W.DPARALLEL_GROUPS END),
12, 6) AS PCT_PARALLEL_GROUPS, CASE WHEN A.PARALLEL_GROUPS >= (SELECT MAX(IQAO610.WLXT009.PARALLEL_GROUPS)
FROM IQAO610.WLXT009) - 1 HOURS THEN DECIMAL(A.PARALLEL_GROUPS * K.SCALE_ADJ * 1.000000 / 1E+06, 12, 6) ELSE DECIMAL(A.PARALLEL_GROUPS / (CASE K.SCALE_ADJ WHEN 0 THEN 1 ELSE K.SCALE_ADJ END) * 1.000000 / 1E+06,
12, 6) END AS PARALLEL_GROUPS_ADJ, A.CPU_TIME, A.ELAPSED_TIME, A.EXECUTIONS, A.GETPAGES, A.SYNC_BUFFER_READS, A.SYNC_BUFFER_WRITES, A.ROWS_EXAMINED, A.ROWS_PROCESSED, A.INDEX_SCANS, A.WF_AND_TABLESPACE_SCANS, A.SORTS_PERFORMED, A.PARALLEL_GROUPS
FROM IQAO610.WLXT009 A, IQAO610.WLXT009 B, IQAO610.WLXT009 C, IQAO610.WLXT009 D, IQAO610.WLXT009 E, IQAO610.WLXT009 F, IQAO610.WLXT009 G, IQAO610.WLXT009 H, IQAO610.WLXT009 I, IQAO610.WLXT009 J, IQAO610.WLXT009 K, IQAO610.WLXT009 L, IQAO610.WLXT009 M, IQAO610.WLXT009 N, IQAO610.WLXT009 O, IQAO610.WLXT009 P, IQAO610.WLXT009 Q, IQAO610.WLXT009 R, IQAO610.WLXT009 S, IQAO610.WLXT009 T, IQAO610.WLXT009 U, IQAO610.WLXT009 V, IQAO610.WLXT009 W, IQAO610.WLXT009 X, IQAO610.WLXT009 Y, IQAO610.WLXT009 Z, IQAO610.WLXT009 AA, IQAO610.WLXT009 AB, IQAO610.WLXT009 AC, IQAO610.WLXT009 AD, IQAO610.WLXT009 AE, IQAO610.WLXT009 AF, IQAO610.WLXT009 AG, IQAO610.WLXT009 AH, IQAO610.WLXT009 AI, IQAO610.WLXT009 AJ, IQAO610.WLXT009 AK, IQAO610.WLXT009 AL, IQAO610.WLXT009 AM, IQAO610.WLXT009 AN, IQAO610.WLXT009 AO, IQAO610.WLXT009 AP, IQAO610.WLXT009 AQ, IQAO610.WLXT009 AR, IQAO610.WLXT009 AS, IQAO610.WLXT009 AT, IQAO610.WLXT009 AU, IQAO610.WLXT009 AV, IQAO610.WLXT009 AW, IQAO610.WLXT009 AX, IQAO610.WLXT009 AY, IQAO610.WLXT009 AZ, IQAO610.WLXT009 BA, IQAO610.WLXT009 BB, IQAO610.WLXT009 BC, IQAO610.WLXT009 BD, IQAO610.WLXT009 BE, IQAO610.WLXT009 BF, IQAO610.WLXT009 BG, IQAO610.WLXT009 BH, IQAO610.WLXT009 BI, IQAO610.WLXT009 BJ, IQAO610.WLXT009 BK, IQAO610.WLXT009 BL, IQAO610.WLXT009 BM, IQAO610.WLXT009 BN, IQAO610.WLXT009 BO, IQAO610.WLXT009 BP, IQAO610.WLXT009 BQ, IQAO610.WLXT009 BR, IQAO610.WLXT009 BS, IQAO610.WLXT009 BT, IQAO610.WLXT009 BU, IQAO610.WLXT009 BV, IQAO610.WLXT009 BW, IQAO610.WLXT009 BX, IQAO610.WLXT009 BY, IQAO610.WLXT009 BZ, IQAO610.WLXT009 CA, IQAO610.WLXT009 CB, IQAO610.WLXT009 CC, IQAO610.WLXT009 CD, IQAO610.WLXT009 CE, IQAO610.WLXT009 CF, IQAO610.WLXT009 CG, IQAO610.WLXT009 CH, IQAO610.WLXT009 CI, IQAO610.WLXT009 CJ, IQAO610.WLXT009 CK, IQAO610.WLXT009 CL, IQAO610.WLXT009 CM, IQAO610.WLXT009 CN, IQAO610.WLXT009 CO, IQAO610.WLXT009 CP, IQAO610.WLXT009 CQ, IQAO610.WLXT009 CR, IQAO610.WLXT009 CS, IQAO610.WLXT009 CT, IQAO610.WLXT009 CU, IQAO610.WLXT009 CV, IQAO610.WLXT009 CW, IQAO610.WLXT009 CX, IQAO610.WLXT009 CY, IQAO610.WLXT009 CZ, IQAO610.WLXT009 DA, IQAO610.WLXT009 DB, IQAO610.WLXT009 DC, IQAO610.WLXT009 DD, IQAO610.WLXT009 DE, IQAO610.WLXT009 DF, IQAO610.WLXT009 DG, IQAO610.WLXT009 DH, IQAO610.WLXT009 DI, IQAO610.WLXT009 DJ, IQAO610.WLXT009 DK, IQAO610.WLXT009 DL, IQAO610.WLXT009 DM, IQAO610.WLXT009 DN, IQAO610.WLXT009 DO, IQAO610.WLXT009 DP, IQAO610.WLXT009 DQ, IQAO610.WLXT009 DR, IQAO610.WLXT009 DS, IQAO610.WLXT009 DT, IQAO610.WLXT009 DU, IQAO610.WLXT009 DV, IQAO610.WLXT009 DW, IQAO610.WLXT009 DX, IQAO610.WLXT009 DY, IQAO610.WLXT009 DZ, IQAO610.WLXT009 EA, IQAO610.WLXT009 EB, IQAO610.WLXT009 EC, IQAO610.WLXT009 ED, IQAO610.WLXT009 EE, IQAO610.WLXT009 EF, IQAO610.WLXT009 EG, IQAO610.WLXT009 EH, IQAO610.WLXT009 EI, IQAO610.WLXT009 EJ, IQAO610.WLXT009 EK, IQAO610.WLXT009 EL, IQAO610.WLXT009 EM, IQAO610.WLXT009 EN, IQAO610.WLXT009 EO, IQAO610.WLXT009 EP, IQAO610.WLXT009 EQ, IQAO610.WLXT009 ER, IQAO610.WLXT009 ES, IQAO610.WLXT009 ET, IQAO610.WLXT009 EU, IQAO610.WLXT009 EV, IQAO610.WLXT009 EW, IQAO610.WLXT009 EX, IQAO610.WLXT009 EY, IQAO610.WLXT009 EZ, IQAO610.WLXT009 FA, IQAO610.WLXT009 FB, IQAO610.WLXT009 FC, IQAO610.WLXT009 FD, IQAO610.WLXT009 FE, IQAO610.WLXT009 FF, IQAO610.WLXT009 FG, IQAO610.WLXT009 FH, IQAO610.WLXT009 FI, IQAO610.WLXT009 FJ, IQAO610.WLXT009 FK, IQAO610.WLXT009 FL, IQAO610.WLXT009 FM, IQAO610.WLXT009 FN, IQAO610.WLXT009 FO, IQAO610.WLXT009 FP, IQAO610.WLXT009 FQ, IQAO610.WLXT009 FR, IQAO610.WLXT009 FS, IQAO610.WLXT009 FT, IQAO610.WLXT009 FU, IQAO610.WLXT009 FV, IQAO610.WLXT009 FW, IQAO610.WLXT009 FX, IQAO610.WLXT009 FY, IQAO610.WLXT009 FZ, IQAO610.WLXT009 GA, IQAO610.WLXT009 GB, IQAO610.WLXT009 GC, IQAO610.WLXT009 GD, IQAO610.WLXT009 GE, IQAO610.WLXT009 GF, IQAO610.WLXT009 GG, IQAO610.WLXT009 GH, IQAO610.WLXT009 GI, IQAO610.WLXT009 GJ, IQAO610.WLXT009 GK, IQAO610.WLXT009 GL, IQAO610.WLXT009 GM, IQAO610.WLXT009 GN, IQAO610.WLXT009 GO, IQAO610.WLXT009 GP, IQAO610.WLXT009 GQ, IQAO610.WLXT009 GR, IQAO610.WLXT009 GS, IQAO610.WLXT009 GT, IQAO610.WLXT009 GU, IQAO610.WLXT009 GV, IQAO610.WLXT009 GW, IQAO610.WLXT009 GX, IQAO610.WLXT009 GY, IQAO610.WLXT009 GZ, IQAO610.WLXT009 HA, IQAO610.WLXT009 HB, IQAO610.WLXT009 HC, IQAO610.WLXT009 HD, IQAO610.WLXT009 HE, IQAO610.WLXT009 HF, IQAO610.WLXT009 HG, IQAO610.WLXT009 HH, IQAO610.WLXT009 HI, IQAO610.WLXT009 HJ, IQAO610.WLXT009 HK, IQAO610.WLXT009 HL, IQAO610.WLXT009 HM, IQAO610.WLXT009 HN, IQAO610.WLXT009 HO, IQAO610.WLXT009 HP, IQAO610.WLXT009 HQ, IQAO610.WLXT009 HR, IQAO610.WLXT009 HS, IQAO610.WLXT009 HT, IQAO610.WLXT009 HU, IQAO610.WLXT009 HV, IQAO610.WLXT009 HW, IQAO610.WLXT009 HX, IQAO610.WLXT009 HY, IQAO610.WLXT009 HZ, IQAO610.WLXT009 IA, IQAO610.WLXT009 IB, IQAO610.WLXT009 IC, IQAO610.WLXT009 ID, IQAO610.WLXT009 IE, IQAO610.WLXT009 IF, IQAO610.WLXT009 IG, IQAO610.WLXT009 IH, IQAO610.WLXT009 II, IQAO610.WLXT009 IJ, IQAO610.WLXT009 IK, IQAO610.WLXT009 IL, IQAO610.WLXT009 IM, IQAO610.WLXT009 IN, IQAO610.WLXT009 IO, IQAO610.WLXT009 IP, IQAO610.WLXT009 IQ, IQAO610.WLXT009 IR, IQAO610.WLXT009 IS, IQAO610.WLXT009 IT, IQAO610.WLXT009 IU, IQAO610.WLXT009 IV, IQAO610.WLXT009 IW, IQAO610.WLXT009 IX, IQAO610.WLXT009 IY, IQAO610.WLXT009 IZ, IQAO610.WLXT009 JA, IQAO610.WLXT009 JB, IQAO610.WLXT009 JC, IQAO610.WLXT009 JD, IQAO610.WLXT009 JE, IQAO610.WLXT009 JF, IQAO610.WLXT009 JG, IQAO610.WLXT009 JH, IQAO610.WLXT009 JI, IQAO610.WLXT009 JJ, IQAO610.WLXT009 JK, IQAO610.WLXT009 JL, IQAO610.WLXT009 JM, IQAO610.WLXT009 JN, IQAO610.WLXT009 JO, IQAO610.WLXT009 JP, IQAO610.WLXT009 JQ, IQAO610.WLXT009 JR, IQAO610.WLXT009 JS, IQAO610.WLXT009 JT, IQAO610.WLXT009 JU, IQAO610.WLXT009 JV, IQAO610.WLXT009 JW, IQAO610.WLXT009 JX, IQAO610.WLXT009 JY, IQAO610.WLXT009 JZ, IQAO610.WLXT009 KA, IQAO610.WLXT009 KB, IQAO610.WLXT009 KC, IQAO610.WLXT009 KD, IQAO610.WLXT009 KE, IQAO610.WLXT009 KF, IQAO610.WLXT009 KG, IQAO610.WLXT009 KH, IQAO610.WLXT009 KI, IQAO610.WLXT009 KJ, IQAO610.WLXT009 KK, IQAO610.WLXT009 KL, IQAO610.WLXT009 KM, IQAO610.WLXT009 KN, IQAO610.WLXT009 KO, IQAO610.WLXT009 KP, IQAO610.WLXT009 KQ, IQAO610.WLXT009 KR, IQAO610.WLXT009 KS, IQAO610.WLXT009 KT, IQAO610.WLXT009 KU, IQAO610.WLXT009 KV, IQAO610.WLXT009 KW, IQAO610.WLXT009 KX, IQAO610.WLXT009 KY, IQAO610.WLXT009 KZ, IQAO610.WLXT009 LA, IQAO610.WLXT009 LB, IQAO610.WLXT009 LC, IQAO610.WLXT009 LD, IQAO610.WLXT009 LE, IQAO610.WLXT009 LF, IQAO610.WLXT009 LG, IQAO610.WLXT009 LH, IQAO610.WLXT009 LI, IQAO610.WLXT009 LJ, IQAO610.WLXT009 LK, IQAO610.WLXT009 LL, IQAO610.WLXT009 LM, IQAO610.WLXT009 LN, IQAO610.WLXT009 LO, IQAO610.WLXT009 LP, IQAO610.WLXT009 LQ, IQAO610.WLXT009 LR, IQAO610.WLXT009 LS, IQAO610.WLXT009 LT, IQAO610.WLXT009 LU, IQAO610.WLXT009 LV, IQAO610.WLXT009 LW, IQAO610.WLXT009 LX, IQAO610.WLXT009 LY, IQAO610.WLXT009 LZ, IQAO610.WLXT009 MA, IQAO610.WLXT009 MB, IQAO610.WLXT009 MC, IQAO610.WLXT009 MD, IQAO610.WLXT009 ME, IQAO610.WLXT009 MF, IQAO610.WLXT009 MG, IQAO610.WLXT009 MH, IQAO610.WLXT009 MI, IQAO610.WLXT009 MJ, IQAO610.WLXT009 MK, IQAO610.WLXT009 ML, IQAO610.WLXT009 MN, IQAO610.WLXT009 MO, IQAO610.WLXT009 MP, IQAO610.WLXT009 MQ, IQAO610.WLXT009 MR, IQAO610.WLXT009 MS, IQAO610.WLXT009 MT, IQAO610.WLXT009 MU, IQAO610.WLXT009 MV, IQAO610.WLXT009 MW, IQAO610.WLXT009 MX, IQAO610.WLXT009 MY, IQAO610.WLXT009 MZ, IQAO610.WLXT009 NA, IQAO610.WLXT009 NB, IQAO610.WLXT009 NC, IQAO610.WLXT009 ND, IQAO610.WLXT009 NE, IQAO610.WLXT009 NF, IQAO610.WLXT009 NG, IQAO610.WLXT009 NH, IQAO610.WLXT009 NI, IQAO610.WLXT009 NJ, IQAO610.WLXT009 NK, IQAO610.WLXT009 NL, IQAO610.WLXT009 NM, IQAO610.WLXT009 NO, IQAO610.WLXT009 NP, IQAO610.WLXT009 NQ, IQAO610.WLXT009 NR, IQAO610.WLXT009 NS, IQAO610.WLXT009 NT, IQAO610.WLXT009 NU, IQAO610.WLXT009 NV, IQAO610.WLXT009 NW, IQAO610.WLXT009 NX, IQAO610.WLXT009 NY, IQAO610.WLXT009 NZ, IQAO610.WLXT009 OA, IQAO610.WLXT009 OB, IQAO610.WLXT009 OC, IQAO610.WLXT009 OD, IQAO610.WLXT009 OE, IQAO610.WLXT009 OF, IQAO610.WLXT009 OG, IQAO610.WLXT009 OH, IQAO610.WLXT009 OI, IQAO610.WLXT009 OJ, IQAO610.WLXT009 OK, IQAO610.WLXT009 OL, IQAO610.WLXT009 OM, IQAO610.WLXT009 ON, IQAO610.WLXT009 OO, IQAO610.WLXT009 OP, IQAO610.WLXT009 OQ, IQAO610.WLXT009 OR, IQAO610.WLXT009 OS, IQAO610.WLXT009 OT, IQAO610.WLXT009 OU, IQAO610.WLXT009 OV, IQAO610.WLXT009 OW, IQAO610.WLXT009 OX, IQAO610.WLXT009 OY, IQAO610.WLXT009 OZ, IQAO610.WLXT009 PA, IQAO610.WLXT009 PB, IQAO610.WLXT009 PC, IQAO610.WLXT009 PD, IQAO610.WLXT009 PE, IQAO610.WLXT009 PF, IQAO610.WLXT009 PG, IQAO610.WLXT009 PH, IQAO610.WLXT009 PI, IQAO610.WLXT009 PJ, IQAO610.WLXT009 PK, IQAO610.WLXT009 PL, IQAO610.WLXT009 PM, IQAO610.WLXT009 PN, IQAO610.WLXT009 PO, IQAO610.WLXT009 PP, IQAO610.WLXT009 PQ, IQAO610.WLXT009 PR, IQAO610.WLXT009 PS, IQAO610.WLXT009 PT, IQAO610.WLXT009 PU, IQAO610.WLXT009 PV, IQAO610.WLXT009 PW, IQAO610.WLXT009 PX, IQAO610.WLXT009 PY, IQAO610.WLXT009 PZ, IQAO610.WLXT009 QA, IQAO610.WLXT009 QB, IQAO610.WLXT009 QC, IQAO610.WLXT009 QD, IQAO610.WLXT009 QE, IQAO610.WLXT009 QF, IQAO610.WLXT009 QG, IQAO610.WLXT009 QH, IQAO610.WLXT009 QI, IQAO610.WLXT009 QJ, IQAO610.WLXT009 QK, IQAO610.WLXT009 QL, IQAO610.WLXT009 QM, IQAO610.WLXT009 QN, IQAO610.WLXT009 QO, IQAO610.WLXT009 QP, IQAO610.WLXT009 QQ, IQAO610.WLXT009 QR, IQAO610.WLXT009 QS, IQAO610.WLXT009 QT, IQAO610.WLXT009 QU, IQAO610.WLXT009 QV, IQAO610.WLXT009 QW, IQAO610.WLXT009 QX, IQAO610.WLXT009 QY, IQAO610.WLXT009 QZ, IQAO610.WLXT009 RA, IQAO610.WLXT009 RB, IQAO610.WLXT009 RC, IQAO610.WLXT009 RD, IQAO610.WLXT009 RE, IQAO610.WLXT009 RF, IQAO610.WLXT009 RG, IQAO610.WLXT009 RH, IQAO610.WLXT009 RI, IQAO610.WLXT009 RJ, IQAO610.WLXT009 RK, IQAO610.WLXT009 RL, IQAO610.WLXT009 RM, IQAO610.WLXT009 RN, IQAO610.WLXT009 RO, IQAO610.WLXT009 RP, IQAO610.WLXT009 RQ, IQAO610.WLXT009 RR, IQAO610.WLXT009 RS, IQAO610.WLXT009 RT, IQAO610.WLXT009 RU, IQAO610.WLXT009 RV, IQAO610.WLXT009 RW, IQAO610.WLXT009 RX, IQAO610.WLXT009 RY, IQAO610.WLXT009 RZ, IQAO610.WLXT009 SA, IQAO610.WLXT009 SB, IQAO610.WLXT009 SC, IQAO610.WLXT009 SD, IQAO610.WLXT009 SE, IQAO610.WLXT009 SF, IQAO610.WLXT009 SG, IQAO610.WLXT009 SH, IQAO610.WLXT009 SI, IQAO610.WLXT009 SJ, IQAO610.WLXT009 SK, IQAO610.WLXT009 SL, IQAO610.WLXT009 SM, IQAO610.WLXT009 SN, IQAO610.WLXT009 SO, IQAO610.WLXT009 SP, IQAO610.WLXT009 SQ, IQAO610.WLXT009 SR, IQAO610.WLXT009 SS, IQAO610.WLXT009 ST, IQAO610.WLXT009 SU, IQAO610.WLXT009 SV, IQAO610.WLXT009 SW, IQAO610.WLXT009 SX, IQAO610.WLXT009 SY, IQAO610.WLXT009 SZ, IQAO610.WLXT009 TA, IQAO610.WLXT009 TB, IQAO610.WLXT009 TC, IQAO610.WLXT009 TD, IQAO610.WLXT009 TE, IQAO610.WLXT009 TF, IQAO610.WLXT009 TG, IQAO610.WLXT009 TH, IQAO610.WLXT009 TI, IQAO610.WLXT009 TJ, IQAO610.WLXT009 TK, IQAO610.WLXT009 TL, IQAO610.WLXT009 TM, IQAO610.WLXT009 TN, IQAO610.WLXT009 TO, IQAO610.WLXT009 TP, IQAO610.WLXT009 TQ, IQAO610.WLXT009 TR, IQAO610.WLXT009 TS, IQAO610.WLXT009 TT, IQAO610.WLXT009 TU, IQAO610.WLXT009 TV, IQAO610.WLXT009 TW, IQAO610.WLXT009 TX, IQAO610.WLXT009 TY, IQAO610.WLXT009 TZ, IQAO610.WLXT009 UA, IQAO610.WLXT009 UB, IQAO610.WLXT009 UC, IQAO610.WLXT009 UD, IQAO610.WLXT009 UE, IQAO610.WLXT009 UF, IQAO610.WLXT009 UG, IQAO610.WLXT009 UH, IQAO610.WLXT009 UI, IQAO610.WLXT009 UJ, IQAO610.WLXT009 UK, IQAO610.WLXT009 UL, IQAO610.WLXT009 UM, IQAO610.WLXT009 UN, IQAO610.WLXT009 UO, IQAO610.WLXT009 UP, IQAO610.WLXT009 UQ, IQAO610.WLXT009 UR, IQAO610.WLXT009 US, IQAO610.WLXT009 UT, IQAO610.WLXT009 UU, IQAO610.WLXT009 UV, IQAO610.WLXT009 UW, IQAO610.WLXT009 UX, IQAO610.WLXT009 UY, IQAO610.WLXT009 UZ, IQAO610.WLXT009 VA, IQAO610.WLXT009 VB, IQAO610.WLXT009 VC, IQAO610.WLXT009 VD, IQAO610.WLXT009 VE, IQAO610.WLXT009 VF, IQAO610.WLXT009 VG, IQAO610.WLXT009 VH, IQAO610.WLXT009 VI, IQAO610.WLXT009 VJ, IQAO610.WLXT009 VK, IQAO610.WLXT009 VL, IQAO610.WLXT009 VM, IQAO610.WLXT009 VN, IQAO610.WLXT009 VO, IQAO610.WLXT009 VP, IQAO610.WLXT009 VQ, IQAO610.WLXT009 VR, IQAO610.WLXT009 VS, IQAO610.WLXT009 VT, IQAO610.WLXT009 VU, IQAO610.WLXT009 VV, IQAO610.WLXT009 VW, IQAO610.WLXT009 VX, IQAO610.WLXT009 VY, IQAO610.WLXT009 VZ, IQAO610.WLXT009 WA, IQAO610.WLXT009 WB, IQAO610.WLXT009 WC, IQAO610.WLXT009 WD, IQAO610.WLXT009 WE, IQAO610.WLXT009 WF, IQAO610.WLXT009 WG, IQAO610.WLXT009 WH, IQAO610.WLXT009 WI, IQAO610.WLXT009 WJ, IQAO610.WLXT009 WK, IQAO610.WLXT009 WL, IQAO610.WLXT009 WM, IQAO610.WLXT009 WN, IQAO610.WLXT009 WO, IQAO610.WLXT009 WP, IQAO610.WLXT009 WQ, IQAO610.WLXT009 WR, IQAO610.WLXT009 WS, IQAO610.WLXT009 WT, IQAO610.WLXT009 WU, IQAO610.WLXT009 WV, IQAO610.WLXT009 WW, IQAO610.WLXT009 WX, IQAO610.WLXT009 WY, IQAO610.WLXT009 WZ, IQAO610.WLXT009 XA, IQAO610.WLXT009 XB, IQAO610.WLXT009 XC, IQAO610.WLXT009 XD, IQAO610.WLXT009 XE, IQAO610.WLXT009 XF, IQAO610.WLXT009 XG, IQAO610.WLXT009 XH, IQAO610.WLXT009 XI, IQAO610.WLXT009 XJ, IQAO610.WLXT009 XK, IQAO610.WLXT009 XL, IQAO610.WLXT009 XM, IQAO610.WLXT009 XN, IQAO610.WLXT009 XO, IQAO610.WLXT009 XP
```

Analyze it

Compare view:
Select any two SQLs to
generate graphs

Database Development - Eclipse

File Edit Navigate Search Project Run Window Help

WLX WLX int

int New DB2 for z

Sum of Executions	Sum of CPU Time	Average CPU Time	Percentage CPU Time	Sum of GETPAGES
85	307.602598	3.618854	13.55	2798277
76	199.839950	2.629473	8.80	1947851
76	199.839950	2.629473	8.80	1947851
76	199.839950	2.629473	8.80	1947851
76	199.839950	2.629473	8.80	1947851
76	199.839950	2.629473	8.80	1947851
8	172.930322	21.616290	7.62	555208
2	158.202192	79.101096	6.97	173132

Result counter: 12

Compare view

Selection 1

Sum of Executions	Sum of CPU Time	Average CPU Time	Percentage CPU Time	Sum of GETPAGES	Average G
85	307.602598	3.618854	13.55	2798277	
76	199.839950	2.629473	8.80	1947851	
76	199.839950	2.629473	8.80	1947851	
76	199.839950	2.629473	8.80	1947851	
76	199.839950	2.629473	8.80	1947851	
76	199.839950	2.629473	8.80	1947851	

Selection 2

Sum of Executions	Sum of CPU Time	Average CPU Time	Percentage CPU Time	Sum of GETPAGES	Average G
8	172.930322	21.616290	7.62	555208	
2	158.202192	79.101096	6.97	173132	
2	158.202192	79.101096	6.97	173132	
2	158.202192	79.101096	6.97	173132	
2	158.202192	79.101096	6.97	173132	

Selection details

Column name	Selection value 1	Selection value 2
Sum of Executions	76	2
Sum of CPU Time	199.839950	158.202192
Average CPU Time	2.629473	79.101096
Percentage CPU Time	8.80	6.97
Sum of GETPAGES	1947851	173132
Average GETPAGES	25,629	86,566
Percentage GETPAGES	13.95	1.24
Sum of Elapsed Time	299.131558	183.349670

Value 1

```
WITH
INPUT (WLX_TIMESTAMP, STMT_TIMESTAMP, STMT_ORIGIN, STMT_TYPE,
STMT_ID, STMT_GROUP_SSID, EXECUTIONS, ELAPSE_TIME, CPU_TIME,
GRTP_OPERATIONS) AS
```

Value 2

```
WITH
PRIME_KEY (WLX_TIMESTAMP, STMT_GROUP_SSID, STMT_ID, STMT_ORIGIN,
STMT_TIMESTAMP, STMT_TYPE, SCALE_ADJ) AS
(SELECT WLX_TIMESTAMP, STMT_GROUP_SSID, STMT_ID, STMT_ORIGIN, STMT_TIMESTAMP,
```

Analyze it

WLX Report

Main Title: WLX Report

Subtitle: 2013-07-15 12:21:02.424463

Category Column: Primary Authorization ID

Value Column: Sum of CPU Time

Chart Type: ☒ Bar chart ☐ Pie chart ☐ Line chart

WLX Report

Main Title: WLX Report

Subtitle: CPU intensive statements

Selection 1: 2013-07-10

Selection 2: 2013-06-27

Generate Close

Report generation dialog and selection

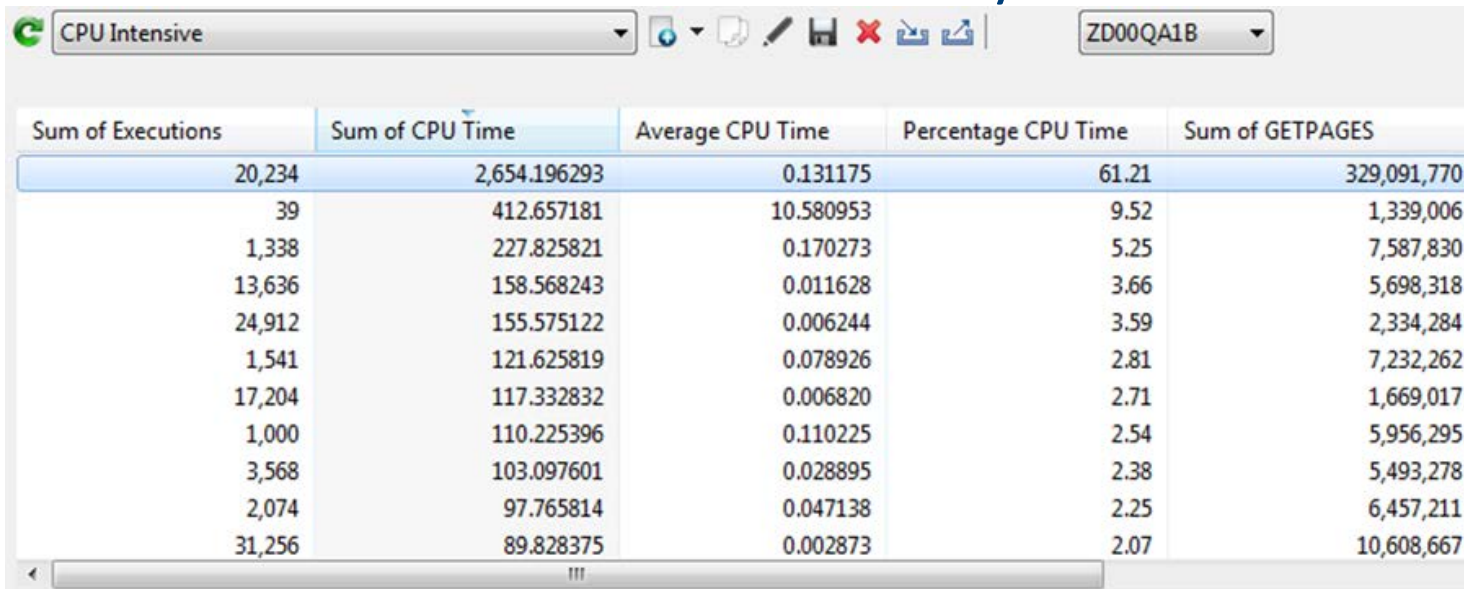
Harvesting the low hanging fruit

OK, so assuming you have all the data where shall we begin???

1. How about Intensive CPU?

Harvesting the low hanging fruit

The definition is simply the percentage of the CPU for a given period of time for an SQL. Here is two days of data:



Sum of Executions	Sum of CPU Time	Average CPU Time	Percentage CPU Time	Sum of GETPAGES
20,234	2,654.196293	0.131175	61.21	329,091,770
39	412.657181	10.580953	9.52	1,339,006
1,338	227.825821	0.170273	5.25	7,587,830
13,636	158.568243	0.011628	3.66	5,698,318
24,912	155.575122	0.006244	3.59	2,334,284
1,541	121.625819	0.078926	2.81	7,232,262
17,204	117.332832	0.006820	2.71	1,669,017
1,000	110.225396	0.110225	2.54	5,956,295
3,568	103.097601	0.028895	2.38	5,493,278
2,074	97.765814	0.047138	2.25	6,457,211
31,256	89.828375	0.002873	2.07	10,608,667

As you can see one SQL executed over 20,000 times and soaked up the lion's share of the machine! Drilling down reveals the SQL:

WHERE KA_BEARB_ZK <> '1' AND KA_BEARB_ZK <> 'L' WITH CS

Harvesting the low hanging fruit

OK, so assuming you have all the data where shall we begin???

1. How about Intensive CPU?
2. What about by Application?

Harvesting the low hanging fruit

The Application definition is simply the Primary Authorization Id or the Collection/Package. Here is one snapshot of data:

Number of Statements	Sum of CPU Time	Average CPU Time	Highest CPU Time	Sum of Elapsed Time	Average Elapsed Time
41	1,145.929112	28.648227	673.160930	2,171.410912	54.285272
6	122.393241	20.398873	38.379085	674.223872	112.370645

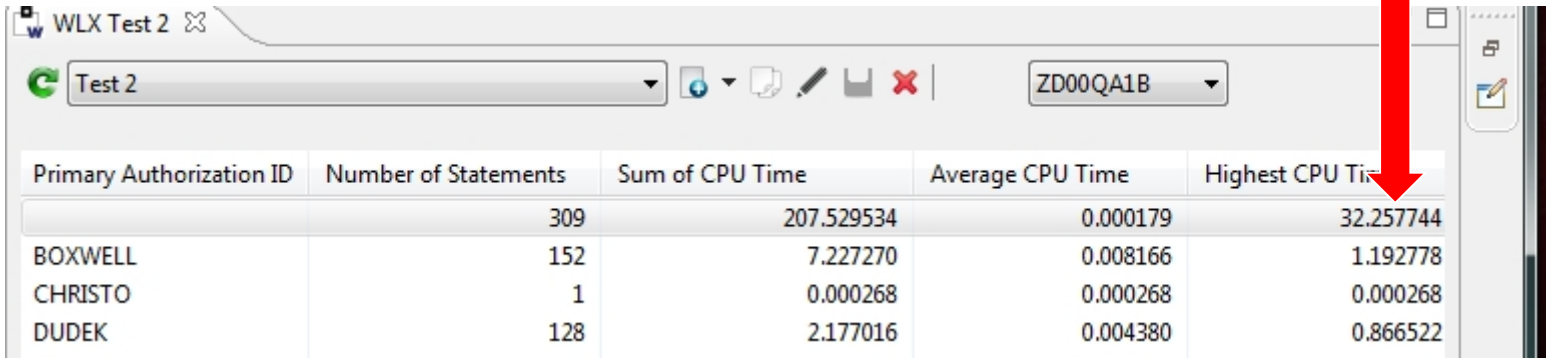
The average CPU is pretty high and the „highest“ is very high!
 Drilling on down:

CPU Time	Elapsed Time	Executions	GETPAGES	Synchronous Buffer Reads
673.160930	1,076.620276	1	9,731,686	42,481

Only one execution for this guy and the SQL was a pretty horrible three table join with about 20 predicates.

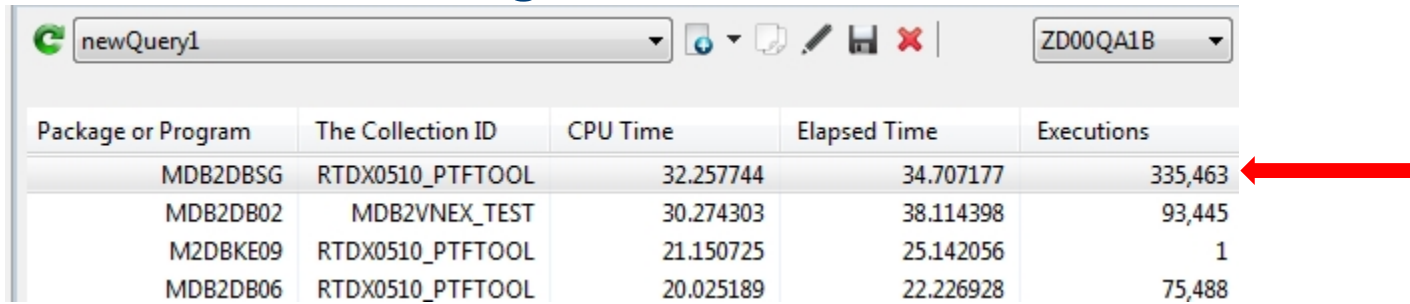
Harvesting the low hanging fruit

Here is a high CPU Static application:



Primary Authorization ID	Number of Statements	Sum of CPU Time	Average CPU Time	Highest CPU Time
	309	207.529534	0.000179	32.257744
BOXWELL	152	7.227270	0.008166	1.192778
CHRISTO	1	0.000268	0.000268	0.000268
DUDEK	128	2.177016	0.004380	0.866522

Drill down to Package level:



Package or Program	The Collection ID	CPU Time	Elapsed Time	Executions
MDB2DBSG	RTDX0510_PTFTOOL	32.257744	34.707177	335,463
MDB2DB02	MDB2VNEX_TEST	30.274303	38.114398	93,445
M2DBKE09	RTDX0510_PTFTOOL	21.150725	25.142056	1
MDB2DB06	RTDX0510_PTFTOOL	20.025189	22.226928	75,488

Harvesting the low hanging fruit

Drill down to SQL level:

```
SELECT CHAR ( SUBSTR ( DIGITS ( YEAR ( STATSTIME ) ) , 9 , 2 ) CONCAT  
              SUBSTR ( DIGITS ( DAYOFYEAR ( STATSTIME ) ) , 8 , 3 ) , 5 ) INTO : H  
FROM SE_STOGROUP  
WHERE NAME = : H  
WITH UR
```

For every physical object a select from SYSSTOGROUP... Rewrite to a LEFT OUTER JOIN and the problem is solved!

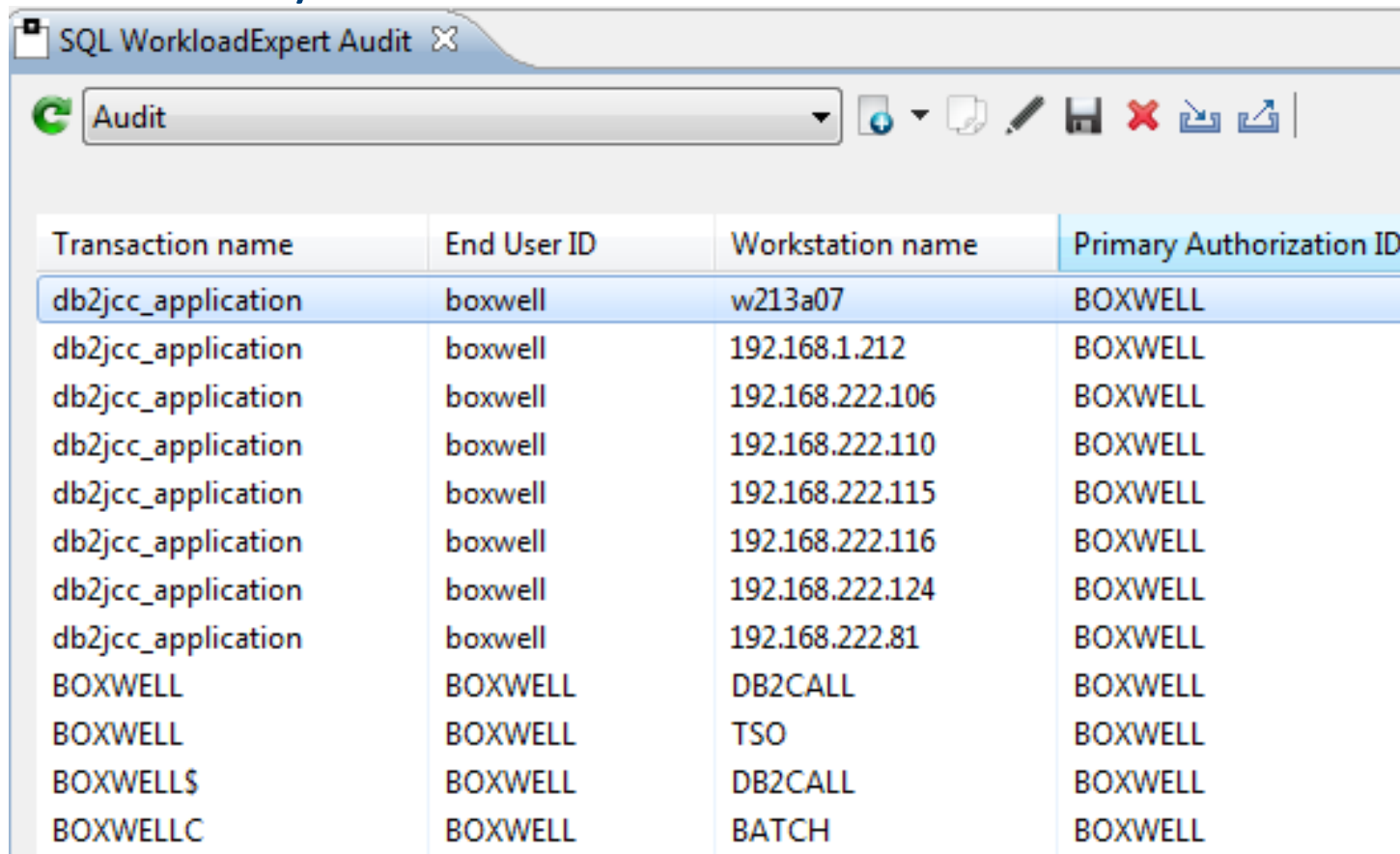
Harvesting the low hanging fruit

OK, so assuming you have all the data where shall we begin???

1. How about Intensive CPU?
2. What about by Application?
3. Auditing???

Harvesting the low hanging fruit

Choose how you like to find out who did what and when...



The screenshot shows the 'SQL WorkloadExpert Audit' window. It has a toolbar with icons for refresh, save, print, edit, delete, and export. Below the toolbar is a table with four columns: Transaction name, End User ID, Workstation name, and Primary Authorization ID. The table contains 13 rows of audit data.

Transaction name	End User ID	Workstation name	Primary Authorization ID
db2jcc_application	boxwell	w213a07	BOXWELL
db2jcc_application	boxwell	192.168.1.212	BOXWELL
db2jcc_application	boxwell	192.168.222.106	BOXWELL
db2jcc_application	boxwell	192.168.222.110	BOXWELL
db2jcc_application	boxwell	192.168.222.115	BOXWELL
db2jcc_application	boxwell	192.168.222.116	BOXWELL
db2jcc_application	boxwell	192.168.222.124	BOXWELL
db2jcc_application	boxwell	192.168.222.81	BOXWELL
BOXWELL	BOXWELL	DB2CALL	BOXWELL
BOXWELL	BOXWELL	TSO	BOXWELL
BOXWELL\$	BOXWELL	DB2CALL	BOXWELL
BOXWELLC	BOXWELL	BATCH	BOXWELL

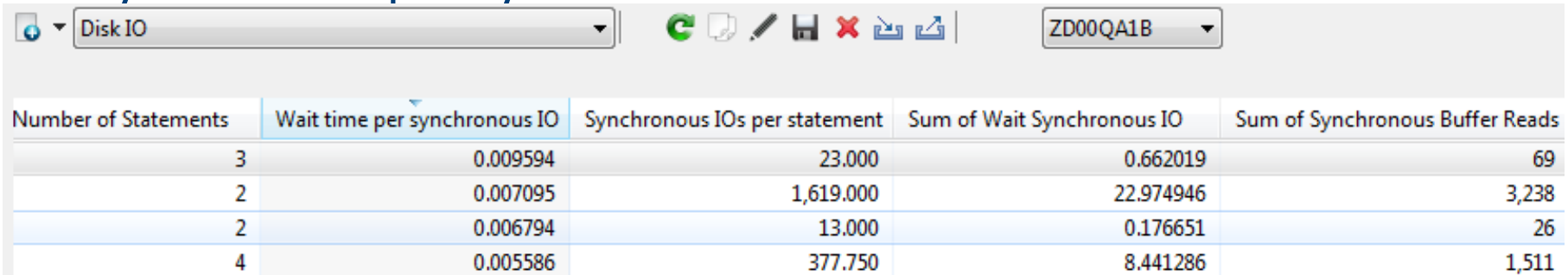
Harvesting the low hanging fruit

OK, so assuming you have all the data where shall we begin???

1. How about Intensive CPU?
2. What about by Application?
3. Auditing?
4. Disk I/O Performance?

Harvesting the low hanging fruit

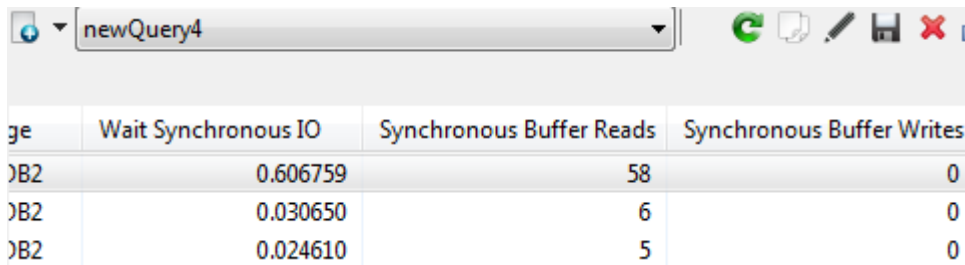
Any Wait time per synchronous IO over 0.002 seconds is bad:



Report: Disk IO | Filter: ZD00QA1B

Number of Statements	Wait time per synchronous IO	Synchronous IOs per statement	Sum of Wait Synchronous IO	Sum of Synchronous Buffer Reads
3	0.009594	23.000	0.662019	69
2	0.007095	1,619.000	22.974946	3,238
2	0.006794	13.000	0.176651	26
4	0.005586	377.750	8.441286	1,511

For OLTP transactions any with more than one Synchronous IOs per statement is “sub optimal”! Drill down shows details:



Report: newQuery4

Statement	Wait Synchronous IO	Synchronous Buffer Reads	Synchronous Buffer Writes
DB2	0.606759	58	0
DB2	0.030650	6	0
DB2	0.024610	5	0

Harvesting the low hanging fruit

OK, so assuming you have all the data where shall we begin???

1. How about Intensive CPU?
2. What about by Application?
3. Auditing?
4. Disk I/O Performance?
5. Up and Down Scaling?

Harvesting the low hanging fruit

Up and Down scaling is all about getting a “level playing field” when looking at the cache data. Simply displaying the data for SQLs that have been in the cache a week next to SQLs that have been in the cache for only 10 minutes is a bit biased!

CPU Time	Percentage CPU Time	CPU time adjusted	GETPAGES	Percentage GETPAGES	GETPAGES adjusted
29.231328	1.857795	1.238178	681,568	4.427895	28,869
23.371722	1.485388	0.989977	593,016	3.852606	25,118
16.904098	1.074338	0.604954	446,936	2.903578	15,994
174.386924	11.083150	0.558840	1,622,158	10.538562	5,198

Here you can easily see the “normal Top 10” values and the “adjusted” values. Your “Top 10” suddenly contains completely new candidates that you were ***never*** even aware of!

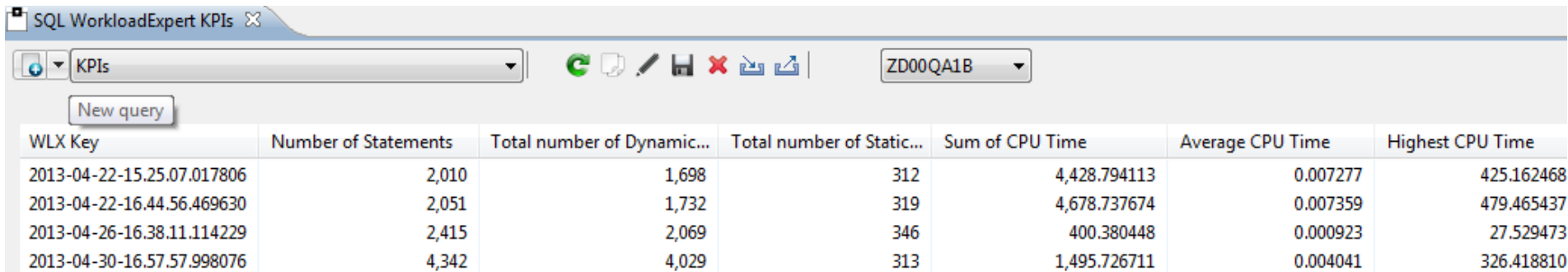
Harvesting the low hanging fruit

OK, so assuming you have all the data where shall we begin???

1. How about Intensive CPU?
2. What about by Application?
3. Auditing?
4. Disk I/O Performance?
5. Up and Down Scaling?
6. KPIs for your Enterprise?

Harvesting the low hanging fruit

Naturally all this data also lets you build up a great set of KPIs to keep track of how many, what type, and how CPU & I/O hungry everything is:



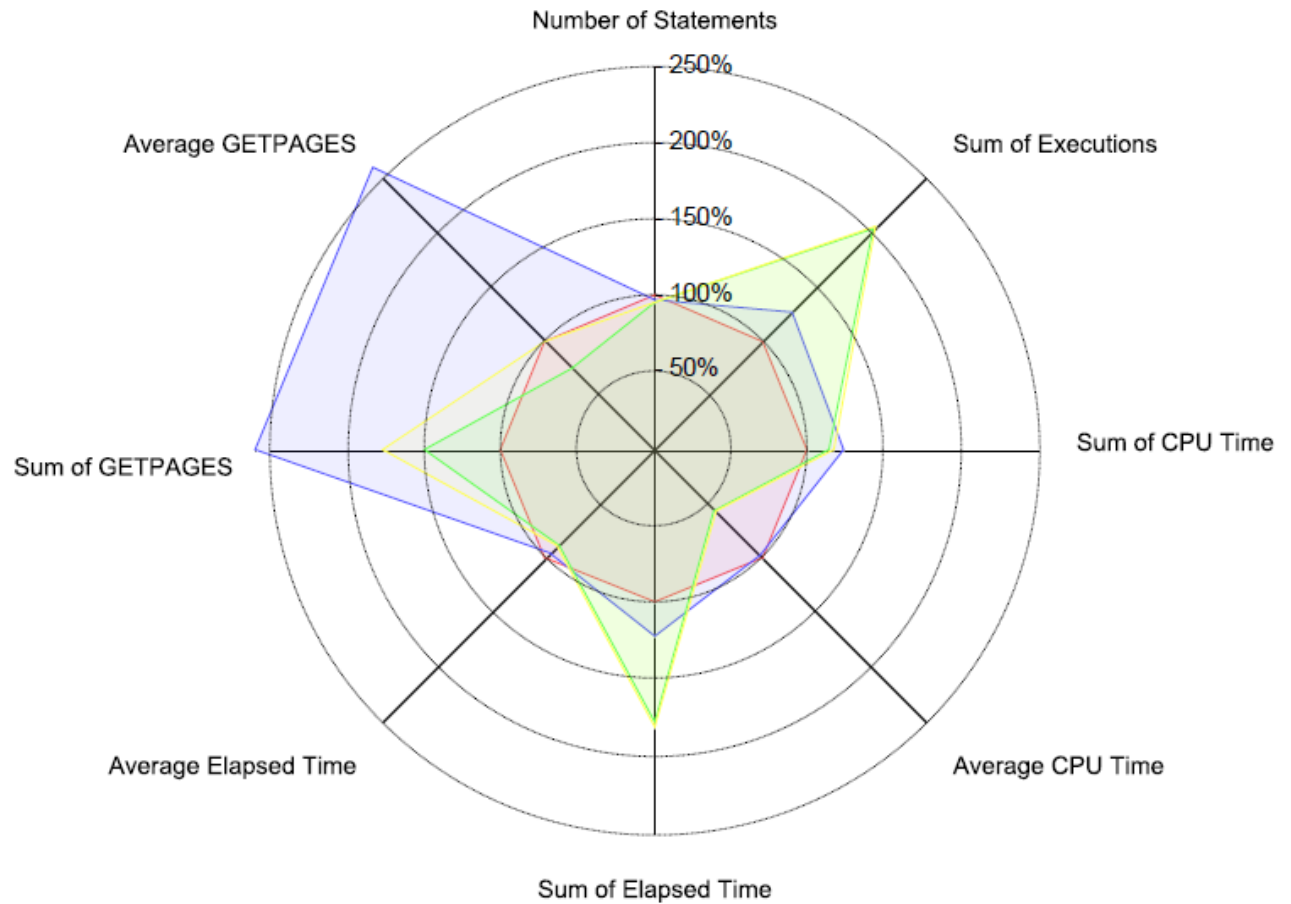
The screenshot shows the 'SQL WorkloadExpert KPIs' window. It has a toolbar with icons for refresh, copy, paste, save, delete, and print. A dropdown menu is set to 'KPIs' and a search box contains 'ZD00QA1B'. A 'New query' button is visible. Below the toolbar is a table with the following data:

WLX Key	Number of Statements	Total number of Dynamic...	Total number of Static...	Sum of CPU Time	Average CPU Time	Highest CPU Time
2013-04-22-15.25.07.017806	2,010	1,698	312	4,428.794113	0.007277	425.162468
2013-04-22-16.44.56.469630	2,051	1,732	319	4,678.737674	0.007359	479.465437
2013-04-26-16.38.11.114229	2,415	2,069	346	400.380448	0.000923	27.529473
2013-04-30-16.57.57.998076	4,342	4,029	313	1,495.726711	0.004041	326.418810

Not just CPU but GetPages etc. are also available.

Harvesting the low hanging fruit

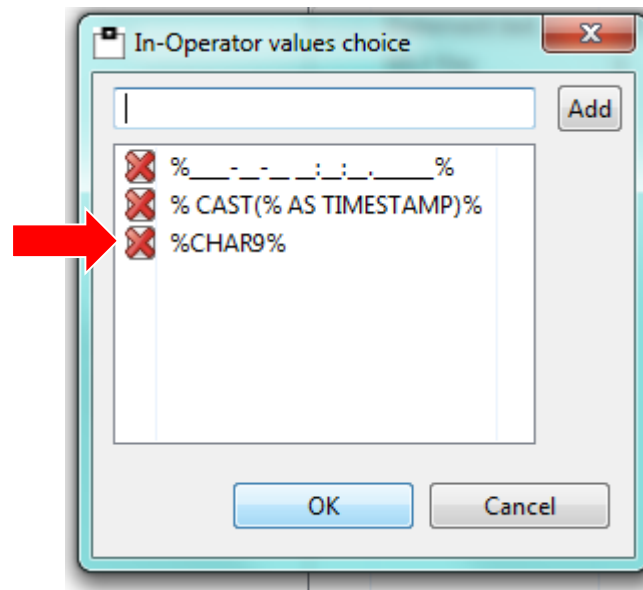
Then you can play
with radar charts:



Harvesting the low hanging fruit

OK, so assuming you have all the data where shall we begin???

1. How about Intensive CPU?
2. What about by Application?
3. Auditing?
4. Disk I/O Performance?
5. Up and Down Scaling?
6. KPIs for your Enterprise?
7. Searching for SQL?



Harvesting the low hanging fruit

And then...


SQL WorkloadExpert SQL text search

SQL text search ZD00QA1B

WLX Key	SQL type	Statement text	WLX DB2 SSID
2014-09-24-15.43.21.998200	SELECT	select * from IQA0610.BAIM_STATEMENTS where (RUNID = '2014-07-02 10:03:55.541206') FETCH FIRST 500 ROWS ONLY	QA1B
2014-09-24-15.43.21.998200	SELECT	SELECT COUNT_BETTER_PROG,COUNT_WORSE_PROG,COUNT_BETTER_STMT,COUNT_WORSE_STMT FROM IQA0610...	QA1B

Drill down to get a better view

```
SELECT COUNT_BETTER_PROG, COUNT_WORSE_PROG, COUNT_BETTER_STMT,
COUNT_WORSE_STMT
FROM IQA0610.BAIM_RUNIDS
WHERE RUN_MODE IN ('DYNA') AND (RUNID = '2014-04-22 14:27:18.84815')
ORDER BY RUNID DESC
```



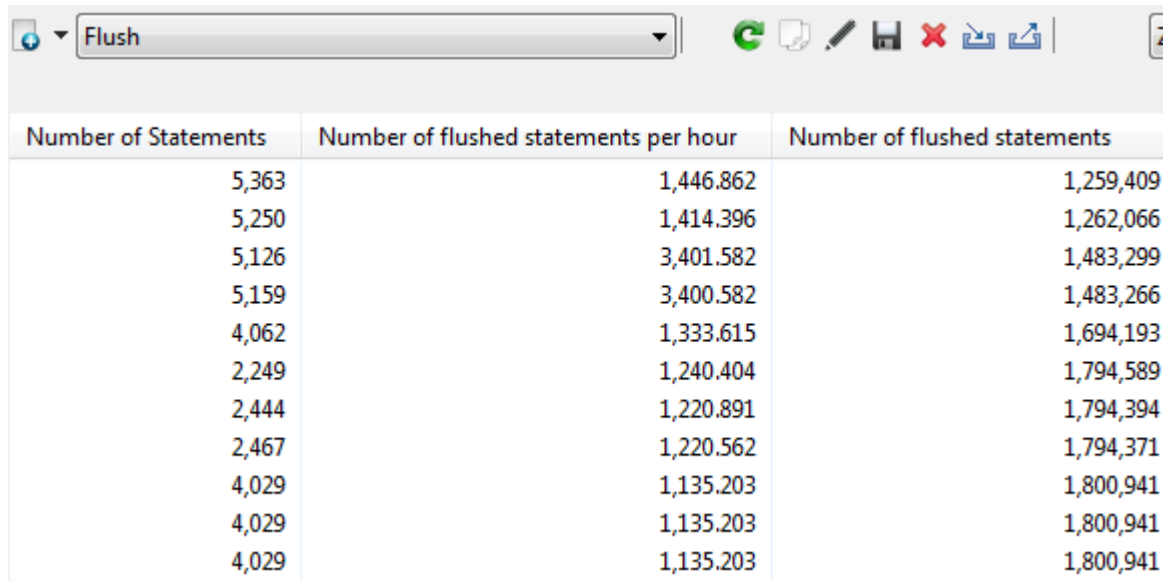
Harvesting the low hanging fruit

OK, so assuming you have all the data where shall we begin???

1. How about Intensive CPU?
2. What about by Application?
3. Auditing?
4. Disk I/O Performance?
5. Up and Down Scaling?
6. KPIs for your Enterprise?
7. Searching for SQL?
8. Flushed with success?

Harvesting the low hanging fruit

If you are catching and storing all the SQL then you can easily see how good the size and performance of your cache is:



Number of Statements	Number of flushed statements per hour	Number of flushed statements
5,363	1,446.862	1,259,409
5,250	1,414.396	1,262,066
5,126	3,401.582	1,483,299
5,159	3,400.582	1,483,266
4,062	1,333.615	1,694,193
2,249	1,240.404	1,794,589
2,444	1,220.891	1,794,394
2,467	1,220.562	1,794,371
4,029	1,135.203	1,800,941
4,029	1,135.203	1,800,941
4,029	1,135.203	1,800,941

Rule of thumb is to make the EDMSTMTC as big as it can be!
 200,000 is a good start!

Harvesting the low hanging fruit

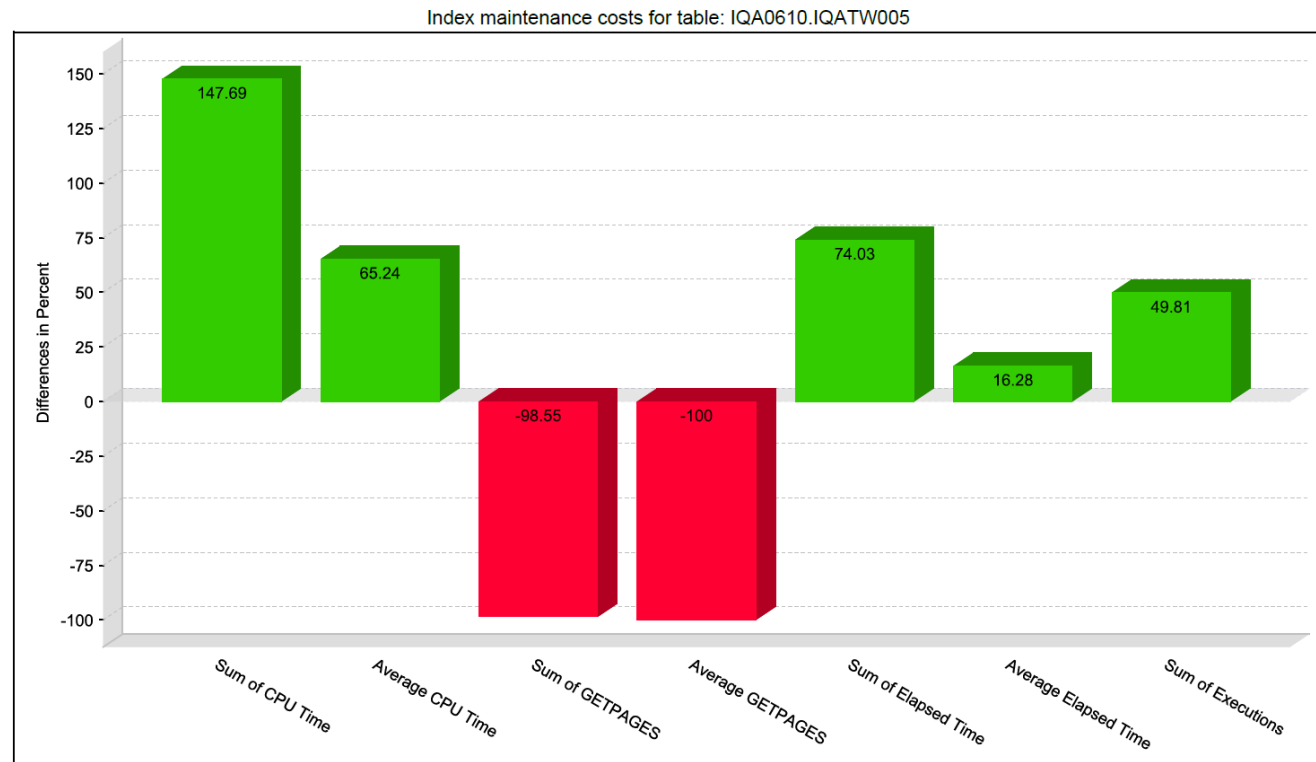
OK, so assuming you have all the data where shall we begin???

1. How about Intensive CPU?
2. What about by Application?
3. Auditing?
4. Disk I/O Performance?
5. Up and Down Scaling?
6. KPIs for your Enterprise?
7. Searching for SQL?
8. Flushed with success?
9. Index Comparison?

Harvesting the low hanging fruit

Compare KPIs before and after Index creation. Especially twinned with Virtual Index usage this is a real winner! Did that new Index help or hinder my DB2?

WLX Report



Harvesting the low hanging fruit

OK, so assuming you have all the data where shall we begin???

1. How about Intensive CPU?
2. What about by Application?
3. Auditing?
4. Disk I/O Performance?
5. Up and Down Scaling?
6. KPIs for your Enterprise?
7. Searching for SQL?
8. Flushed with success?
9. Index Comparison?
10. Miscellaneous other possibilities...

Harvesting the low hanging fruit

Again, if you are catching and storing all the SQL then you can do:

- Sub-system loading checking
- Delay detection
- Object Quiet Times – Alter & Reorg
- Find all non-executed Packages - Free
- Never executed SQLs within executed Packages - Delete
- Never referenced Tables/Indexes - Drop
- Select only usage of objects – Locksize tuning

Harvesting the low hanging fruit

Why stop with just these IFCIDs? If you have a technology for high speed catching and writing why not expand it to handle:

172 – Deadlocks

196 – Timeouts

337 – Lock Escalations

359 – Index page Splits

366/376 – BIF Usage


Harvesting the low hanging fruit

Some real world numbers to amaze and astound:

- On one member of a Data Sharing group the SQLs that normally ran fast were running 45% slower than on other members. After using a WLX it was discovered that this member had orders of magnitude more updates – Increase Log Buffer, Active Log, and Archive Log sizes then redirect some traffic. Et Voila!
- 450,000,000 Get pages per hour saved! -- New index created which gave a knock on performance boost effect to the whole DB2 sub-system
- CPU Reduction from 17,111 seconds per hour to 16 seconds per hour! – One “Bad Guy” query tuned
- Elapsed time from 30,000 seconds per hour to 30 seconds per hour! – Another single SQL “Bad Guy” query tuned

WLX typical use cases

Application Development:

- Application Workload Analysis: E.g. which machine load is produced by a certain Application?
- Explain Tool link (e.g.  **SQL PerformanceExpert**, IBM DataStudio)
- Show same SQL on Multiple Schemas to detect “heavy-hitters”
- SQL Text Analysis for free text search (e.g.: BIF [Built-in Function] and UDF [User-Defined Functions] -usage, Java-formatted timestamps, etc.)
- View to detect “heavy-hitters” resulting from identical statements using different predicates
- Find unused (orphaned) SQL

WLX typical use cases

Workload/Performance management:

- Workload-Change, Problem-Detection and Trending, Comparison of CPU consumption, I/O, execution rates, current KPIs and deltas – Calculated and summarized to the costs of multiple apps
- Disc Problem Detection – I/O Rates
- SQL KPIs – Background Noise and Exceptions
- SELECT Only Table Detection (READ only activity)
- Delay Detection (All queries which are delayed)
- Up and Down Scaling of SQL Workloads
- DSC Flush Analysis
- CPU Intensive Statements
- Index Maintenance Costs

WLX functional packages of use cases

Database Administration:

- Find never used Objects (Tables, Indexes, and Tablespaces)
- Find never executed Packages

Audit and Security:




- AUDIT tables being accessed
- AUDIT DB2 data being accessed
- AUDIT data manipulation (insert/update/delete)
- See where updates came from (inside or outside the local network)
- See where data is being accessed from (IP address, intra-/extranet, etc.)
- SQL Text Analysis for free text search (BIF [Built-in Function] and UDF [User-Defined Functions] -usage, Java-formatted timestamps, etc.)

Appendix

- DB2 APARs to check for:
 - PM77114 DB2 10 UK91560 – Abend S04E
 - PM78143 DB2 10 UK93065 – SOS – HIPER
 - PM80371 DB2 10 UK93127 – Serviceability for SHTE
 - PM83370 DB2 10 UK94511 – Fields TB and IX sometimes wrong
 - PM85376 DB2 10 UK96310 – Abend S04E
 - PM89121 DB2 10 UK95683 – Storage leak leading to abend – HIPER
 - PM91159 DB2 10 UK97197 – Improve accuracy of IFCID 316 and 401
 - PM92610 DB2 11 UK96376 – Abend with IFCID 400 or 401
 - PM93437 DB2 11 UK97361 – IFCID 316 fields length value problem
 - PM97922 DB2 10 UI12375 DB2 11 UI12376 – Invalid or empty IFCID 316
 - PI07461 OPEN – Inconsistent QA0401EU, GL and GP
 - PI09147 DB2 10 UI15679 DB2 11 UI15680 – Abend S04E
 - PI09408 DB2 10 UI15740 DB2 11 UI15741 – Abend S04E
 - PI09788 DB2 11 UI15739 – SOS with IFCID400
 - PI16183 OPEN MISSING IFCID401

Advertisement ;-)

So now you know...

- Of course it is easier with  **SQL WorkLoadExpert for DB2 z/OS**
 - Data Warehouse
 - Extensible and Extendable
 - Low CPU cost
- For Single SQL tuning it links to  **SQLPerformanceExpert for DB2 z/OS**
- Both work with  **BindImpactExpert for DB2 z/OS** for Access Path comparison and release control

Ulf Heinric

SOFTWARE ENGINEERING GmbH

u.heinrich@seg.de



**25 years of missed opportunities?
SQL Tuning Revisited**

