How a minor change had a major impact

Siegfried Fürst - SOFTWARE ENGINEERING GmbH

Like clockwork, every Thursday evening, an international supermarket conglomorate systematically "pushes" the latest weekly updates into Production.

These updates consist of approved program changes and functional upgrades for the data center of two different supermarket chains. During one of these routine cycles in mid July, an unintentional change was introduced which had far-reaching consequences.

Early Friday morning, reports of users experiencing problems began to trickle in: they were having issues with the login to the merchandising system—which runs under DB2 z/OS. The initial complaints were about delays in the login process. Later, complaints of poor performance in general were reported too.

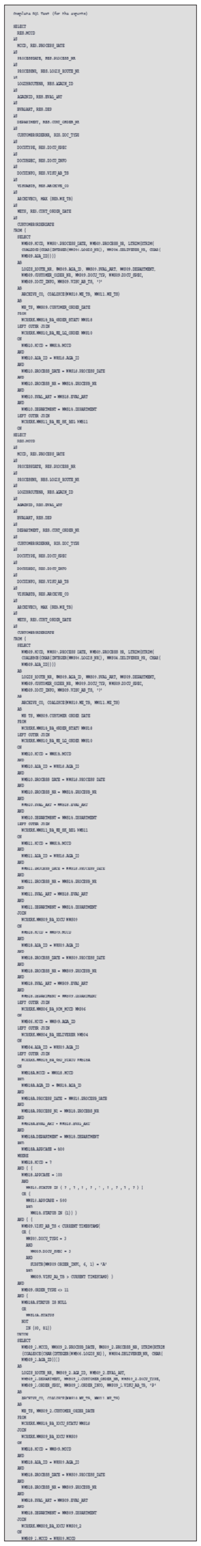
A cursory check of the system revealed excessive CPU consumption. The initial conclusion was that the Thursday evening update had somehow led to the problems.

As would be expected under these circumstances, the initial response was to "rollback" the prior day's release. The feeling among the IT staff was that a program "bug" had somehow been unintentionally introduced.

Unfortunately, even after the data had been restored, the problem persisted further. The login process and general usage of the system was still unacceptable to the user base. The regional branch managers were none too happy.

The IT team continued with their resolution endeavors and a further analysis was undertaken, with a complete review of the IBM Websphere Application Servers, and all of their Java components.

The assessment of all programs running at the time revealed one SQL statement in particular that really stood out. However, after further analysis, it was found that even though this SQL was related to the long wait times during user login, it was not involved in the Thursday evening release. The DBA staff were able to confirm that the execution of this SQL statement was also coincidental to the observed "spikes" in CPU consumption.



```
Template SQL Text (for the experts)

SELECT
 RES.MCCD
AS
 MCCD, RES.PROCESS_DATE
AS
 PROCESSDATE, RES.PROCESS_NR
AS
 PROCESSNR, RES.LOGIS_ROUTE_NR
AS
 LOGISROUTENR, RES.AGAIN_ID
AS
 AGAINID, RES.EVAL_ART
AS
 EVALART, RES.DEP
AS
 DEPARTMENT, RES.CUST_ORDER_NR
AS
 CUSTOMERORDERNR, RES.DOC_TYPE
AS
 DOCUTYPE, RES.DOCU_SPEC
AS
 DOCUSPEC, RES.DOCU_INFO
AS
 DOCUINFO, RES.VISU_AB_TS
AS
 VISUABTS, RES.ARCHIVE_CO
AS
 ARCHIVECO, MAX (RES.ME_TS)
AS
 METS, RES.CUST_ORDER_DATE
AS
 CUSTOMERORDERDATE
FROM (
 SELECT
  MM809.MCCD, MM809.PROCESS_DATE, MM809.PROCESS_NR, LTRIM(RTRIM(
  COALESCE(CHAR(INTEGER(MM804.LOGIS_NR)), MM804.DELIVERER_NR, CHAR(
  MM809.AGA_ID))))
 AS
  LOGIS_ROUTE_NR, MM809.AGA_ID, MM809.EVAL_ART, MM809.DEPARTMENT,
  MM809.CUSTOMER_ORDER_NR, MM809.DOCU_TYP, MM809.DOCU_SPEC,
  MM809.DOCU_INFO, MM809.VISU_AB_TS, '?'
 AS
  ARCHIVE_CO, COALESCE(MM810.ME_TS, MM811.ME_TS)
 AS
  ME_TS, MM809.CUSTOMER_ORDER_DATE
 FROM
  MCRXRX.MM818_BA_ORDER_STATU MM818
 LEFT OUTER JOIN
  MCRXRX.MM810_BA_NE_LG_ORDER MM810
 ON
  MM810.MCCD = MM818.MCCD
 AND
  MM810.AGA_ID = MM818.AGA_ID
 AND
  MM810.PROCESS_DATE = MM818.PROCESS_DATE
 AND
  MM810.PROCESS_NR = MM818.PROCESS_NR
 AND
  MM810.EVAL_ART = MM818.EVAL_ART
 AND
  MM810.DEPARTMENT = MM818.DEPARTMENT
 LEFT OUTER JOIN
  MCRXRX.MM811_BA_NE_SK_BEL MM811
 ON
SELECT
 RES.MCCD
AS
 MCCD, RES.PROCESS_DATE
AS
 PROCESSDATE, RES.PROCESS_NR
AS
 PROCESSNR, RES.LOGIS_ROUTE_NR
AS
 LOGISROUTENR, RES.AGAIN_ID
AS
 AGAINID, RES.EVAL_ART
AS
 EVALART, RES.DEP
AS
 DEPARTMENT, RES.CUST_ORDER_NR
AS
 CUSTOMERORDERNR, RES.DOC_TYPE
AS
 DOCUTYPE, RES.DOCU_SPEC
AS
 DOCUSPEC, RES.DOCU_INFO
AS
 DOCUINFO, RES.VISU_AB_TS
AS
 VISUABTS, RES.ARCHIVE_CO
AS
 ARCHIVECO, MAX (RES.ME_TS)
AS
 METS, RES.CUST_ORDER_DATE
AS
 CUSTOMERORDERDATE
FROM (
 SELECT
  MM809.MCCD, MM809.PROCESS DATE, MM809.PROCESS NR, LTRIM(RTRIM(
  COALESCE(CHAR(INTEGER(MM804.LOGIS_NR)), MM804.DELIVERER_NR, CHAR(
  MM809.AGA_ID))))
 AS
  LOGIS_ROUTE_NR, MM809.AGA_ID, MM809.EVAL_ART, MM809.DEPARTMENT,
  MM809.CUSTOMER_ORDER_NR, MM809.DOCU_TYP, MM809.DOCU_SPEC,
  MM809.DOCU_INFO, MM809.VISU_AB_TS, '?'
 AS
  ARCHIVE_CO, COALESCE(MM810.ME_TS, MM811.ME_TS)
 AS
  ME TS, MM809.CUSTOMER ORDER DATE
 FROM
  MCRXRX.MM818_BA_ORDER_STATU MM818
 LEFT OUTER JOIN
  MCRXRX.MM810_BA_NE_LG_ORDER MM810
 ON
  MM810.MCCD = MM818.MCCD
 AND
  MM810.AGA_ID = MM818.AGA_ID
 AND
  MM810.PROCESS DATE = MM818.PROCESS DATE
 AND
  MM810.PROCESS_NR = MM818.PROCESS_NR
 AND
  MM810.EVAL_ART = MM818.EVAL_ART
 AND
  MM810.DEPARTMENT = MM818.DEPARTMENT
 LEFT OUTER JOIN
  MCRXRX.MM811_BA_NE_SK_BEL MM811
 ON
  MM811.MCCD = MM818.MCCD
 AND
  MM811.AGA_ID = MM818.AGA_ID
 AND
  MM811.PROCESS_DATE = MM818.PROCESS_DATE
 AND
  MM811.PROCESS_NR = MM818.PROCESS_NR
 AND
  MM811.EVAL_ART = MM818.EVAL_ART
 AND
  MM811.DEPARTMENT = MM818.DEPARTMENT
 JOIN
  MCRXRX.MM809_BA_DOCU MM809
 ON
  MM818.MCCD = MM809.MCCD
 AND
  MM818.AGA_ID = MM809.AGA_ID
 AND
  MM818.PROCESS_DATE = MM809.PROCESS_DATE
 AND
  MM818.PROCESS_NR = MM809.PROCESS_NR
 AND
  MM818.EVAL_ART = MM809.EVAL_ART
 AND
  MM818.DEPARTMENT = MM809.DEPARTMENT
 LEFT OUTER JOIN
  MCRXRX.MM806_BA_SCM_MCCD MM806
 ON
  MM806.MCCD = MM809.AGA_ID
 LEFT OUTER JOIN
  MCRXRX.MM804_BA_DELIVERER MM804
 ON
  MM804.AGA_ID = MM809.AGA_ID
 LEFT OUTER JOIN
  MCRXRX.MM818_BA_ORD_STATU MM818A
 ON
  MM818A.MCCD = MM818.MCCD
 AND
  MM818A.AGA_ID = MM818.AGA_ID
 AND
  MM818A.PROCESS_DATE = MM818.PROCESS_DATE
 AND
  MM818A.PROCESS_NR = MM818.PROCESS_NR
 AND
  MM818A.EVAL_ART = MM818.EVAL_ART
 AND
  MM818A.DEPARTMENT = MM818.DEPARTMENT
 AND
  MM818A.APPCASE = 800
 WHERE
  MM818.MCCD = ?
 AND ( (
  MM818.APPCASE = 100
  AND
   MM818.STATUS IN ( ? , ? , ? , ? , ? , ? , ? , ? , ? ) )
  OR (
   MM818.APPCASE = 500
   AND
    MM818.STATUS IN (1)) )
 AND ( (
  MM809.VISU_AB_TS < CURRENT TIMESTAMP)
  OR (
   MM809.DOCU_TYPE = 3
   AND
    MM809.DOCU_SPEC = 3
   AND
    SUBSTR(MM809.ORDER_INFO, 6, 1) = 'A'
   AND
    MM809.VISU_AB_TS > CURRENT TIMESTAMP) )
 AND
  MM809.ORDER_TYPE <> 11
 AND (
  MM818A.STATUS IS NULL
  OR (
   MM818A.STATUS
  NOT
   IN (80, 81))
 UNION
 SELECT
  MM809_2.MCCD, MM809_2.PROCESS_DATE, MM809_2.PROCESS_NR, LTRIM(RTRIM
  (COALESCE(CHAR(INTEGER(MM806.LOGIS_NR)), MM804.DELIVERER_NR, CHAR(
  MM809_2.AGA_ID))))
 AS
  LOGIS_ROUTE_NR, MM809_2.AGA_ID, MM809_2.EVAL_ART,
  MM809_2.DEPARTMENT, MM809_2.CUSTOMER_ORDER_NR, MM809_2.DOCU_TYPE,
  MM809_2.ORDER_SPEC, MM809_2.ORDER_INFO, MM809_2.VISU_AB_TS, 'P'
 AS
  ARCHIVE_CO, COALESCE(MM810.ME_TS, MM811.ME_TS)
 AS
  ME_TS, MM809_2.CUSTOMER_ORDER_DATE
 FROM
  MCRXRX.MM818_BA_DOCU_STATU MM818
 JOIN
  MCRXRX.MM809_BA_DOCU MM809
 ON
  MM818.MCCD = MM809.MCCD
 AND
  MM818.AGA_ID = MM809.AGA_ID
 AND
  MM818.PROCESS_DATE = MM809.PROCESS_DATE
 AND
  MM818.PROCESS_NR = MM809.PROCESS_NR
 AND
  MM818.EVAL_ART = MM809.EVAL_ART
 AND
  MM818.DEPARTMENT = MM809.DEPARTMENT
 JOIN
  MCRXRX.MM809_BA_DOCU MM809_2
 ON
  MM809_2.MCCD = MM809.MCCD
```

Figure 1

The purpose of the suspect SQL statement was to list all open merchandise issues from the prior day, for each store, (e.g. orders, returns, etc.) The SQL is executed at different times for different stores accessing the system—depending on which chain the stores belonged to.

For example: for store #500, the SQL is executed during the login procedure. After a successful login, the user would then immediately be presented with all open issues. For store #947, the user may optionally execute the SQL statement by selecting a menu item on the screen.

In ALL stores, however, each login created an enormous increase in the Thread usage on the IBM Websphere Servers—greatly affecting the performance of all user dialogs on these servers.

Once it was clear that the SQL itself was involved in the huge increase in resource consumption, the focus of attention was turned to the tables that the SQL interacted with. The offending SQL queried extremely large tables. A REORG with inline RUNSTATS of the biggest tables did not cure the problem either.

The "offending" SQL Statement is executed 4,000 times an hour.

Average CPU times per call:

Before Thursday evening - 0.041 sec
After Thursday evening – 9.446 sec

Figure 1, on the left, shows *some* of the actual SQL statement text. To read the statement in full, please visit www.segus.com/wlxucfss.

As of Saturday morning, the chain was suffering from poor performance across its network of stores. Now the regional branch managers were starting to panic, Saturday morning being one of the busiest grocery shopping days, they envisioned the long lines of irate customers and the disruption to the supply chain. The DBA's were getting wor-ried.

By pure coincidence, around this time, a performance project team had been looking into the issue of "nudging" access paths and their general stability. They were working together with the company SOFTWARE ENGINEERING GmbH (SE) to automate their complex requirements and benefit from SE's long experience with large IBM DB2 z/OS customers around the globe.

SE had installed products that were using fast and powerful methods, including the new DB2 10 IFCID technology and access path control.

Management agreed to involve the performance project team in the problem diagnosis to help pinpoint the issues faster.

Enter SQL WorkloadExpert (aka WLX) and Bind ImpactExpert (BIX). Using these new, innovative solutions, the project team was able to very quickly confirm the "suspect" SQL as the culprit. Continuing with the tools, both teams were able to see that there had been a dramatic change to the access path. It was soon determined that the performance had degraded by a factor of 230!

As WLX captures all historical information on SQL and the associated access paths, users are able to view "point-in-time" or "point-to-point" scenarios.

In this case, the DBA's were able to a) analyze the stored Dynamic Statement Cache (DSC) and confirm the offending SQL statement was one of the top consumers at the point in time, and b) correctly confirm that the offending statement was previously seen as operating within acceptable limits.

Armed with this information from WLX and BIX, the DBA team was then able to quickly pinpoint the reason for the dramatic degradation: On the Thursday evening in question, a RUNSTATS utility had been executed across the tables used by the offending SQL statement. Following this, new catalog statistics had been created, and the DB2 Optimizer had, unfortunately, selected a slightly different but dramatically worse access path than before.

Using the SE Tools, the team was easily able to compare "before" and "after" access paths side by side: Following the update of the catalog data on Thursday evening, the DB2 optimizer had chosen WW09_ORDER as the initial table, whereas previously WWS18_ORDER_STATUS had been used. (Please refer to the PLAN_Tables in Figure 2 and Figure 3 on pages 6 and 7 respectively for more details.)

The company employs more than 400,000 people in the German-speaking stores alone.

They run a three-tier 16+ way data-sharing environment with over 50,000 MIPS and DB2 z/OS on the mainframe.

Over 30 logistics centers deliver "Just in Time" to around 3,500 supermarkets. The supply chain is extremely important to the business for guaranteeing product freshness and availability, and it is also a key factor for competitive pricing.

In addition, BIX showed in the SQL Statement Cache after the Thursday evening update, that the SQL statement had a changed access path. Prior to this, the analysis showed that the statement was unchanged.

The small change with the large impact is highlighted in red in the PLAN_TABLE comparison below.

```
Access path OLD ---------------------- ! Access path NEW ----------------------
                                       !
TABLE             QB PN AC MA ME IX PR ! TABLE             QB PN AC MA ME IX PR
  INDEX                 TY CO TH ON FT !   INDEX                 TY CO TH ON FT
----------------- -- -- -- -- -- -- -- ! ----------------- -- -- -- -- -- -- --
RES                1  1 R   0  0 N  S  ! RES                1  1 R   0  0 N  S
                   1  2     0  3 N     !                    1  2     0  3 N
WWS18_ORDER_STATUS 3  1 I   1  0 Y  S  ! WWS09_ORDER        3  1 I   1  0 N  S
  WWS18X01                             !   WWS09X01
.                                      ! .
.                                      ! .
.                                      ! .
Milliseconds:           25             ! Milliseconds:           38
Serviceunits:         1269             ! Serviceunits:         1861
-------------------------------------------------------------------------------
```

The simple solution to the problem, was to create an additional Index for the table WWS18_ORDER_STATUS, in order to nudge the DB2 Optimizer into choosing that table as the initial table.

The new index was quickly created and RUNSTATed in Production in a matter of minutes, then everything starting running as normal. They were then able to restart all the WebSphere servers and all was well.

In Summary

The preceding (true!) story illustrates how just one small change can have dire consequences for the business.

Simply by following their routine procedures, unsuspecting teams were able to wreak havoc. This can happen anywhere.

In this case, the problems were 3-fold:

1. On Thursday night, RUNSTATS were run and the Optimizer chose a very slightly different, but nonetheless extremely poorly-performing access path.
2. Because historical SQL data wasn't readily available to the DBAs, the consequences of this were not immediately recognized.
3. As, we suspect, with most sites, time was wasted in a) rolling back the release, and b) attempting to remediate by running REORG and RUNSTATS utilities. This is not atypical, and in many cases may actually remediate similar issues, but not in this case.

The two-step solution was a) the implementation of an SQL Warehouse (WLX), allowing DBA's to very quickly assess offending SQL's in order to remediate and improve on them. b) To implement a methodology to quickly determine access path changes (BIX) and either approve them or disallow them because of access path degradation.

Now, not only does the performance team benefit from these solutions, but the DBAs are able to quickly respond to any future problems.

Traditional Tools	**SQL WorkloadExpert**
ISPF	GUI-Interface
Classical Drill Down	> 20 Uses Cases out-of-the-box
High CPU Consumption	Low CPU consumption
Allows restricted Monitoring	Allows 24/7 IFCID Catching
High license costs	Moderate license costs

WLX

- Catch all SQL running on a typical system, without the overhead of a trace or utilizing monitors.
- Analyze SQL Workload by aggregate, or in detail.
- Comparatively analyze access paths and SQL.
- Report and chart metrics via a standard Eclipse interface—an inexpensive and commonly-accepted desktop tool.

BIX

- Compares before and after access paths side-by-side.
- Provides an overview of historical data.

DB2 access path after the RUNSTATS on Thursday evening (As of 2013-07-13):

Dynamic Statement Cache DB2/Group member: DBAP

RunID : 1307200911   Created : 2013-07-13-09.11.12.969221   StmtID :   7322372

Milliseconds: 38 Service Units: 1861 Cost Category: B

```
 QBNO QBTYPE CREATOR  TABLE NAME               MTCH IX METH PRNT TABL PRE  MXO
PLNO TABNO  XCREATOR INDEX NAME         ACTYP COLS ON OD   QBLK TYPE FTCH PSQ
---- ------ -------- ------------------ ----- ---- -- ---- ---- ---- ---- ---
   1 SELECT SBXXA00P RES                R        0 N     0    0 W    S      0
   1 2
   1 SELECT                                      0 N     3    0 -           0
   2 0
   2 UNION                                       0       3    1 -           0
   1 0
   3 NCOSUB WCRKRK   WWS09_ORDER        I        1 N     0    2 T    S      0
   1 6      WCRKRK   WWS09X01
   3 NCOSUB WCRKRK   WWS06_SCM_MCCD     I        1 N     1    2 T           0
   2 8      WCRKRK   WWS06X01
   3 NCOSUB WCRKRK   WWS04_DELIVERER    I        1 N     1    2 T           0
   3 1      WCRKRK   WWS04X0A
   3 NCOSUB WCRKRK   WWS18_ORDER_STATUS I        1 Y     1    2 T    S      0
   4 1      WCRKRK   WWS18X01
   3 NCOSUB WCRKRK   WWS10_WE_DEL_ORDER I        6 N     1    2 T           0
   5 2      WCRKRK   WWS10X01
   3 NCOSUB WCRKRK   WWS11_WE_CAN_ORDER I        6 N     1    2 T           0
   6 4      WCRKRK   WWS11X01
   3 NCOSUB WCRKRK   WWS18_ORDER_STATUS I        7 Y     1    2 T    S      0
   7 1      WCRKRK   WWS18X03
   9 NCOSUB WCRKRK   WWS09_ORDER        I        1 N     0    2 T    S      0
   1 1      WCRKRK   WWS09X01
   9 NCOSUB WCRKRK   WWS18_ORDER_STATUS I        1 Y     1    2 T    S      0
   2 1      WCRKRK   WWS18X01
   9 NCOSUB WCRKRK   WWS09_ORDER        I        4 N     1    2 T           0
   3 1      WCRKRK   WWS09X01
   9 NCOSUB WCRKRK   WWS10_WE_DEL_ORDER I        6 N     1    2 T           0
   4 1      WCRKRK   WWS10X01
   9 NCOSUB WCRKRK   WWS06_SCM_MCCD     I        1 N     1    2 T           0
   5 2      WCRKRK   WWS06X01
   9 NCOSUB WCRKRK   WWS04_DELIVERER    I        1 N     1    2 T           0
   6 2      WCRKRK   WWS04X0A
   9 NCOSUB WCRKRK   WWS11_WE_DEL_ORDER I        6 N     1    2 T           0
   7 1      WCRKRK   WWS11X01
   9 NCOSUB WCRKRK   WWS18_ORDER_STATUS I        7 Y     1    2 T    S      0
   8 2      WCRKRK   WWS18X03
```

Figure 2

DB2 access path before the RUNSTATS on Thursday evening (As of 2013-07-11):
Dynamic Statement Cache DB2/Group member: DBAP
RunID : 1307111618   Created : 2013-07-11-16.18.12.544794   StmtID :   5837498
Milliseconds: 25 Service Units: 1269 Cost Category: B

```
QBNO QBTYPE CREATOR  TABLE NAME               MTCH IX METH PRNT TABL PRE  MXO
PLNO TABNO  XCREATOR INDEX NAME         ACTYP COLS ON OD   QBLK TYPE FTCH PSQ
---- ------ -------- ------------------ ----- ---- -- ---- ---- ---- ---- ---
   1 SELECT SBXXA00P RES                R        0 N     0    0 W    S      0
   1 2
   1 SELECT                                      0 N     3    0 -           0
   2 0
   2 UNION                                       0       3    1 -           0
  1 0
   3 NCOSUB WCRKRK   WWS18_ORDER_STATUS I        1 Y     0    2 T    S      0
   1 1      WCRKRK   WWS18X01
   3 NCOSUB WCRKRK   WWS09_ORDER        I        6 N     1    2 T           0
   2 6      WCRKRK   WWS09X01
   3 NCOSUB WCRKRK   WWS10_WE_DEL_ORDER I        6 N     1    2 T           0
   3 2      WCRKRK   WWS10X01
   3 NCOSUB WCRKRK   WWS11_WE_CAN_ORDER I        6 N     1    2 T           0
   4 4      WCRKRK   WWS11X01
   3 NCOSUB WCRKRK   WWS06_SCM_MCCD     I        1 N     1    2 T           0
   5 8      WCRKRK   WWS06X01
   3 NCOSUB WCRKRK   WWS04_DELIVERER    I        1 N     1    2 T           0
   6 1      WCRKRK   WWS04X0A
   3 NCOSUB WCRKRK   WWS18_ORDER_STATUS I        7 Y     1    2 T    S      0
   7 1      WCRKRK   WWS18X03
   9 NCOSUB WCRKRK   WWS18_ORDER_STATUS I        1 Y     0    2 T    S      0
   1 1      WCRKRK   WWS18X01
   9 NCOSUB WCRKRK   WWS09_ORDER        I        6 N     1    2 T           0
   2 1      WCRKRK   WWS09X01
   9 NCOSUB WCRKRK   WWS18_ORDER_STATUS I        7 Y     1    2 T    S      0
   3 2      WCRKRK   WWS18X03
   9 NCOSUB WCRKRK   WWS09_ORDER        I        4 N     1    2 T           0
   4 1      WCRKRK   WWS09X01
   9 NCOSUB WCRKRK   WWS10_WE_DEL_ORDER I        6 N     1    2 T           0
   5 1      WCRKRK   WWS10X01
   9 NCOSUB WCRKRK   WWS11_WE_CAN_ORDER I        6 N     1    2 T           0
   6 1      WCRKRK   WWS11X01
   9 NCOSUB WCRKRK   WWS06_ERP_MCCD     I        1 N     1    2 T           0
   7 2      WCRKRK   WWS06X01
   9 NCOSUB WCRKRK   WWS04_DELIVERER    I        1 N     1    2 T           0
   8 2      WCRKRK   WWS04X0A
```

Figure 3

Meet the SOFTWARE ENGINEERING GMBH “Expert” Tools:

SQL PerformanceExpert® for DB2 z/OS (SPX)
- What-If Analysis
- SQL recommendations
- Statistics Check
- Standards-conformity (Extensible rules supported)

Bind ImpactExpert® for DB2 z/OS (BIX)
- Compares access paths for static and dynamic SQL
- Allows only improved access paths for REBINDS and BINDs
- Helps to secure DB2 Version migration and PTFs/APARs application

SQL WorkloadExpert™ for DB2 z/OS (WLX)
- Workload catcher
- Comparison, tuning and trending
- Out-of-the-box use cases

The Expert Performance Products are “stand-alone” or may be integrated as a complete solution package. They are sold in North America by SEGUS Inc.

For more information, please visit www.segus.com or www.seg.de.

SEGUS Inc
14151 Park Meadow Dr
Chantilly, VA 20151
USA

(800) 327-9650
www.segus.com
info@segus.com

SOFTWARE ENGINEERING GMBH
Robert-Stolz-Strasse 5
40470 Düsseldorf
Germany

+49-211-961-49-0
www.seg.de
info@seg.de