Sharing IBM i Knowledge

System Administration **Modernisation**

Unleash the power of the Query Supervisor

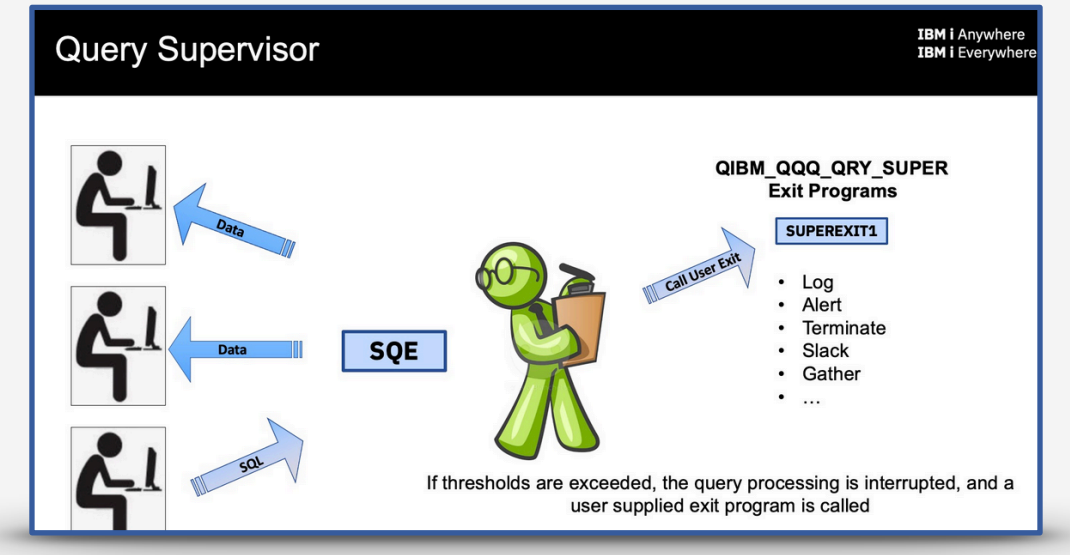
DATE
02 June 2026

Host
Rudi van Helvoirt

"You only know what it is, when it is no longer there.."

The ultimate definition of an invisible foundation.

IBM i 7.3 TR10 and 7.4 TR4 on April 13 2021



Db2 for i - Services (new)

- QSYS2.ADD_QUERY_THRESHOLD
- QSYS2.END_IDLE_SQE_THREADS
- QSYS2.QUERY_SUPERVISOR
- QSYS2.REMOVE_QUERY_THRESHOLD

Where the journey started

✓ **Abstract**

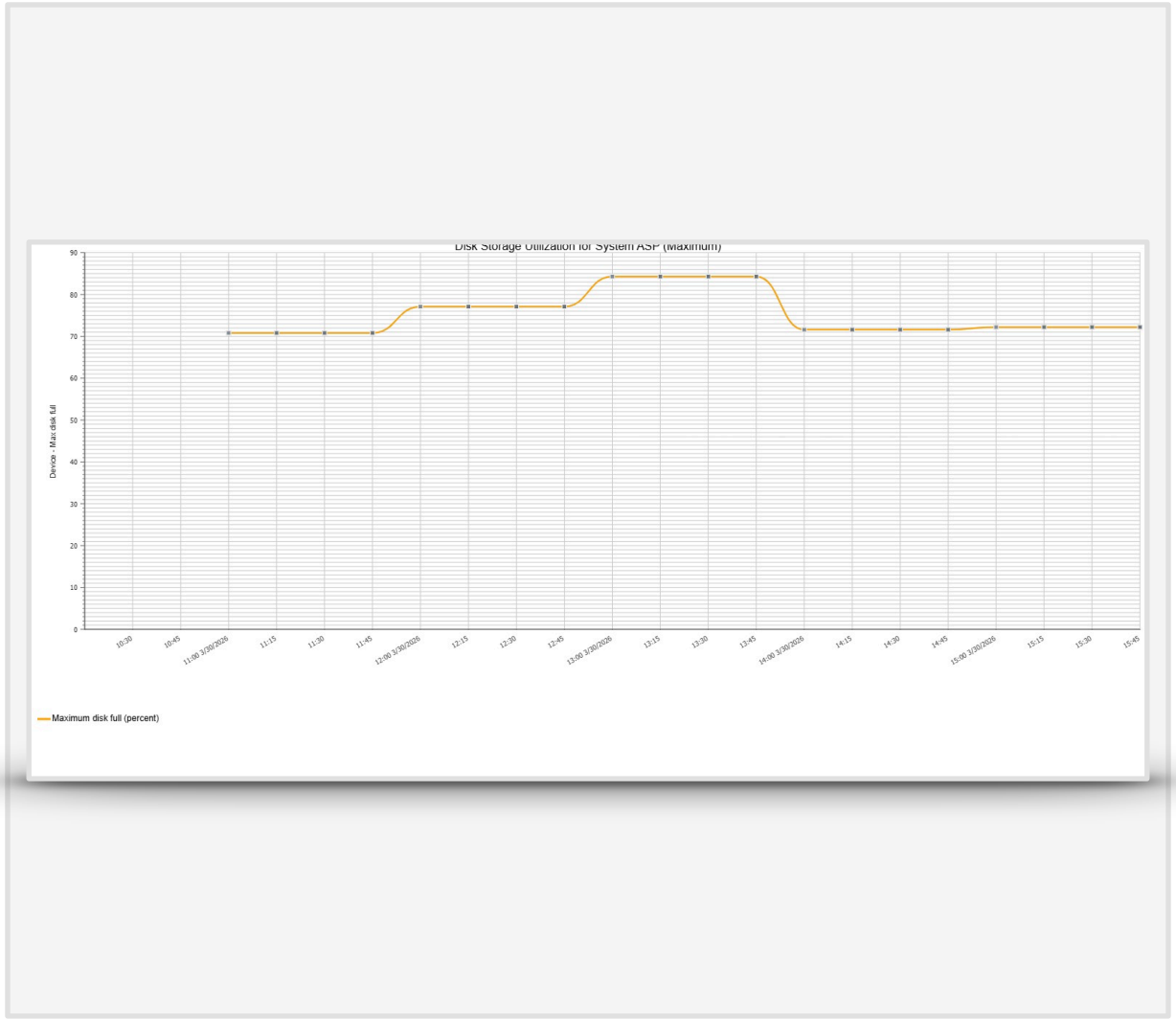
The Db2 for i SQL Query Engine (SQE) provides a Query Supervisor which enables real-time monitoring of resource consumption by SQL and native queries.

✓ **Documentation**

Query Supervisor

✓ **For complete details**

Query Supervisor
ADD_QUERY_THRESHOLD procedure
REMOVE_QUERY_THRESHOLD procedure
QUERY_SUPERVISOR view
Query Supervisor Exit Program
(updated to include CL, to join RPG and C, versions of sample exit programs)



Potential Situation

This could happen to you or not?

- **Disk space usage 90 %**

  After investigation afterwards!!!

  - An ad hoc SQL query was initiated (in this case from an Excel sheet)
  - The user did “cancel” the query
  - In the background the query continued
  - An automated trigger process “killed” all ODBC tasks QZDASOINIT

- **A rough remedy**

  Nothing in place to detect the real cause.

- **Tailor-made solution**

  The Query Supervisor is part of this solution.

The agenda

Three Parts

The Theory 01

What is the Query Supervisor

- The thresholds
- The triggers

The Explanation 02

How does the Query Supervisor work

- Components
- How to activate?
- How to review?
- How to check?

The Building 03

How to build and implement the Query Supervisor

- Schemas
- Tables
- Procedures
- Programs
- Bring it all together
- Testing

The Theory

The Threshold

When running SQL statements they:

- Have an I/O count
- Use CPU time.
- Run for an amount of time
- Use temporary storage

How to determine the thresholds?

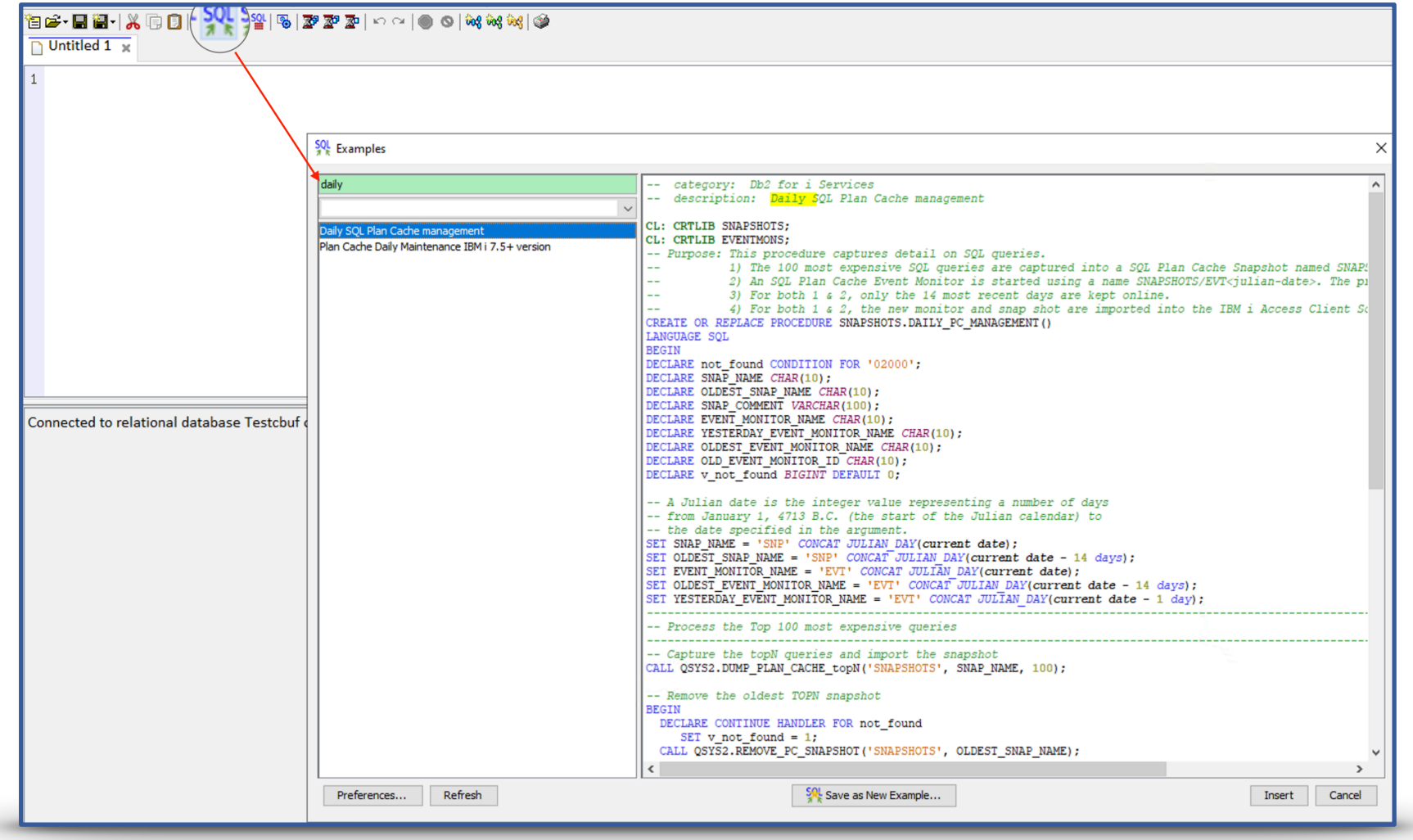
What is “too much”?

The answer as always is “it depends”.

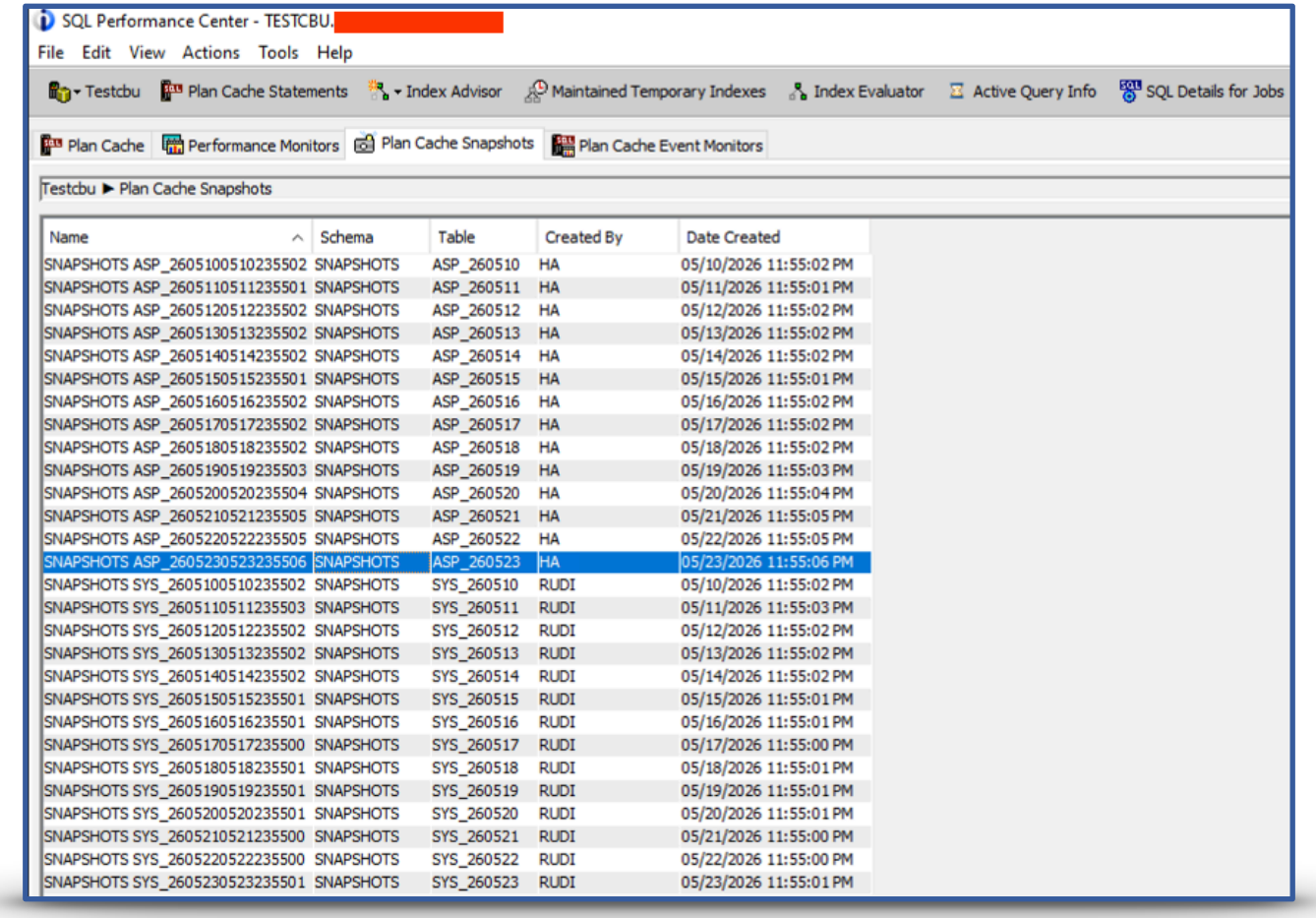
What can we do to get insight?

Daily Plan cache Management.

Daily Plan Cache Management

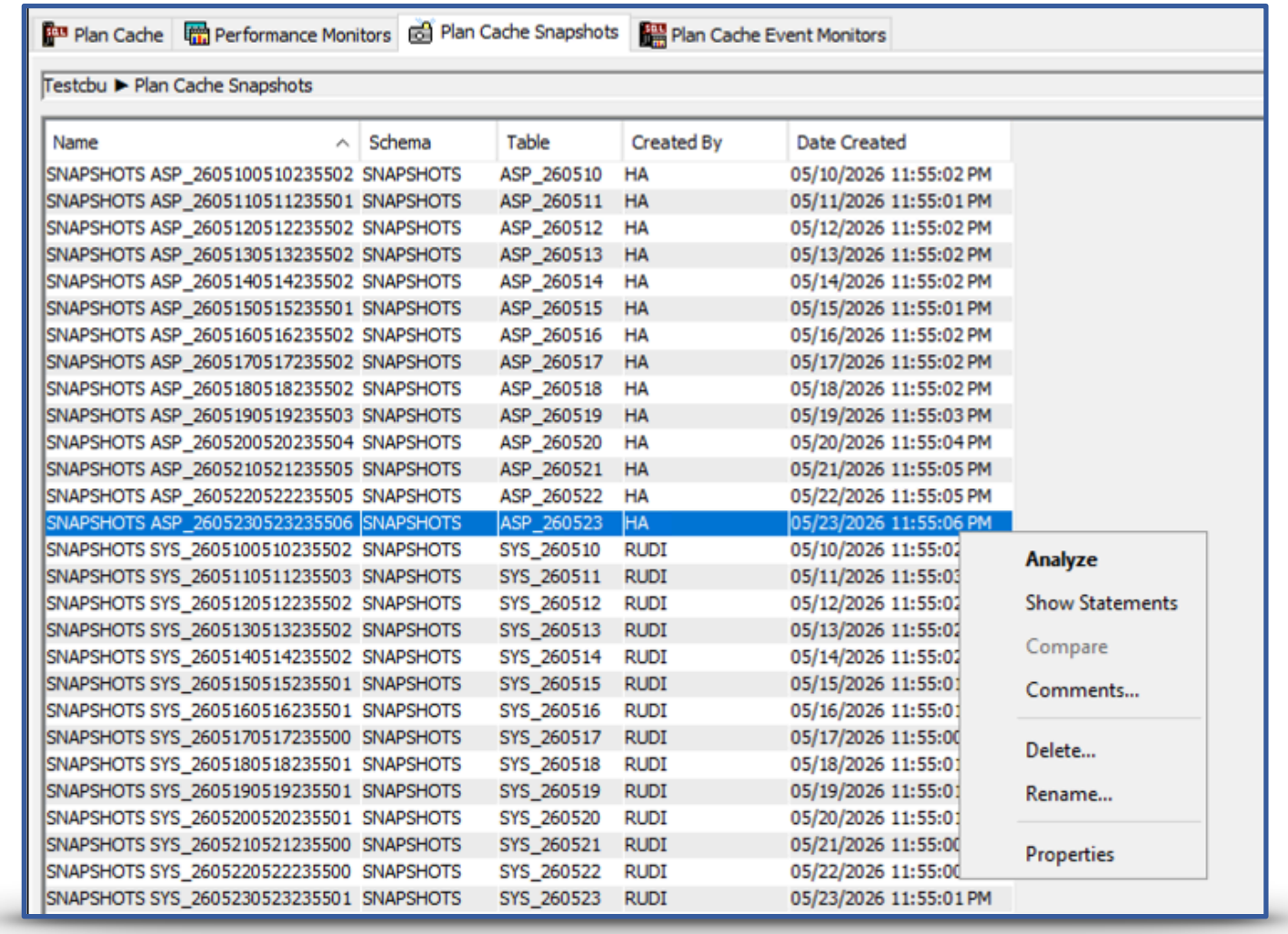


Daily Plan Cache Management



Name	Schema	Table	Created By	Date Created
SNAPSHOTS ASP_2605100510235502	SNAPSHOTS	ASP_260510	HA	05/10/2026 11:55:02 PM
SNAPSHOTS ASP_2605110511235501	SNAPSHOTS	ASP_260511	HA	05/11/2026 11:55:01 PM
SNAPSHOTS ASP_2605120512235502	SNAPSHOTS	ASP_260512	HA	05/12/2026 11:55:02 PM
SNAPSHOTS ASP_2605130513235502	SNAPSHOTS	ASP_260513	HA	05/13/2026 11:55:02 PM
SNAPSHOTS ASP_2605140514235502	SNAPSHOTS	ASP_260514	HA	05/14/2026 11:55:02 PM
SNAPSHOTS ASP_2605150515235501	SNAPSHOTS	ASP_260515	HA	05/15/2026 11:55:01 PM
SNAPSHOTS ASP_2605160516235502	SNAPSHOTS	ASP_260516	HA	05/16/2026 11:55:02 PM
SNAPSHOTS ASP_2605170517235502	SNAPSHOTS	ASP_260517	HA	05/17/2026 11:55:02 PM
SNAPSHOTS ASP_2605180518235502	SNAPSHOTS	ASP_260518	HA	05/18/2026 11:55:02 PM
SNAPSHOTS ASP_2605190519235503	SNAPSHOTS	ASP_260519	HA	05/19/2026 11:55:03 PM
SNAPSHOTS ASP_2605200520235504	SNAPSHOTS	ASP_260520	HA	05/20/2026 11:55:04 PM
SNAPSHOTS ASP_2605210521235505	SNAPSHOTS	ASP_260521	HA	05/21/2026 11:55:05 PM
SNAPSHOTS ASP_2605220522235505	SNAPSHOTS	ASP_260522	HA	05/22/2026 11:55:05 PM
SNAPSHOTS ASP_2605230523235506	SNAPSHOTS	ASP_260523	HA	05/23/2026 11:55:06 PM
SNAPSHOTS SYS_2605100510235502	SNAPSHOTS	SYS_260510	RUDI	05/10/2026 11:55:02 PM
SNAPSHOTS SYS_2605110511235503	SNAPSHOTS	SYS_260511	RUDI	05/11/2026 11:55:03 PM
SNAPSHOTS SYS_2605120512235502	SNAPSHOTS	SYS_260512	RUDI	05/12/2026 11:55:02 PM
SNAPSHOTS SYS_2605130513235502	SNAPSHOTS	SYS_260513	RUDI	05/13/2026 11:55:02 PM
SNAPSHOTS SYS_2605140514235502	SNAPSHOTS	SYS_260514	RUDI	05/14/2026 11:55:02 PM
SNAPSHOTS SYS_2605150515235501	SNAPSHOTS	SYS_260515	RUDI	05/15/2026 11:55:01 PM
SNAPSHOTS SYS_2605160516235501	SNAPSHOTS	SYS_260516	RUDI	05/16/2026 11:55:01 PM
SNAPSHOTS SYS_2605170517235500	SNAPSHOTS	SYS_260517	RUDI	05/17/2026 11:55:00 PM
SNAPSHOTS SYS_2605180518235501	SNAPSHOTS	SYS_260518	RUDI	05/18/2026 11:55:01 PM
SNAPSHOTS SYS_2605190519235501	SNAPSHOTS	SYS_260519	RUDI	05/19/2026 11:55:01 PM
SNAPSHOTS SYS_2605200520235501	SNAPSHOTS	SYS_260520	RUDI	05/20/2026 11:55:01 PM
SNAPSHOTS SYS_2605210521235500	SNAPSHOTS	SYS_260521	RUDI	05/21/2026 11:55:00 PM
SNAPSHOTS SYS_2605220522235500	SNAPSHOTS	SYS_260522	RUDI	05/22/2026 11:55:00 PM
SNAPSHOTS SYS_2605230523235501	SNAPSHOTS	SYS_260523	RUDI	05/23/2026 11:55:01 PM

Daily Plan Cache Management



Query Supervisor limitations & scope

The Query Supervisor enables monitoring and management of queries that reach pre-determined thresholds of resource consumption. Unlike the Predictive Query Governor, which intervenes on the basis of *estimated* resource usage *before* a query runs, the Query Supervisor reacts to *actual* resource usage *while* the query runs.

The Query Supervisor monitors any SQL queries that are run to implement user SQL statements, including native database queries that are processed by SQE. SQL statements that do not require query processing such as a simple INSERT (for example, INSERT INTO <table> VALUES(123)) or COMMIT are not monitored by the Query Supervisor. Query supervision does not occur for queries initiated by the IBM® i operating system or for SQL that has been classified as specific to operating system processing.

The Query Supervisor allows a Database Engineer (DBE) to:

- Deploy real-time notification solutions when a query exceeds a specific resource threshold
- Take actions to terminate queries based on real-time query consumption of system resources
- Capture and log performance details for long-running queries

By configuring the Query Supervisor, a DBE can proactively manage excessive resource usage, monitor for unexpected workload variation, and automatically end run-away queries. The Query Supervisor provides the infrastructure for establishing and detecting thresholds. The specific action(s) taken when a threshold is reached is determined by one or more exit programs that the DBE provides.

The Theory

The Triggers

Action to be taken when a threshold is exceeded?

- Ability to investigate the SQL statement
- End or Hold the SQL Statement
- Get notified
- Automate the action

Maximum of flexibility:

- Using custom build software
- You determine, but IBM offers example software

The Explanation

Components

Thresholds

- Define Threshold
- Maintain the Thresholds
- Active the Thresholds
- Remove the Threshold

Query Supervisor configuration and operation

Query Supervisor thresholds are configured using the add and remove procedures:
ADD_QUERY_THRESHOLD procedure
REMOVE_QUERY_THRESHOLD procedure.
The QUERY_SUPERVISOR view shows the defined thresholds.

The four threshold types which can be monitored by the Query Supervisor are:

1. **CPU Time** – The total processing unit time used by the query, in seconds
2. **Elapsed Time** – The total clock time, in seconds
3. **Temporary storage** – The amount of storage, in megabytes (MB), that the query allocates
4. **Total I/O count** – The total number of I/O operations

The Explanation

Components

Exit Point Programs

- Query Supervisor Exit Program
- Add Query Supervisor Exit Program
- Remove Query Supervisor Exit Program
- Review Query Supervisor Exit Program

Query Supervisor Exit Program

The Query Supervisor exit program is called when a job is running a query and a resource threshold defined by the Query Supervisor is met or exceeded.
This exit is called in the thread that is running the query.

The resource thresholds monitored by the Query Supervisor may be configured by using the ADD_QUERY_THRESHOLD procedure and the REMOVE_QUERY_THRESHOLD procedure.

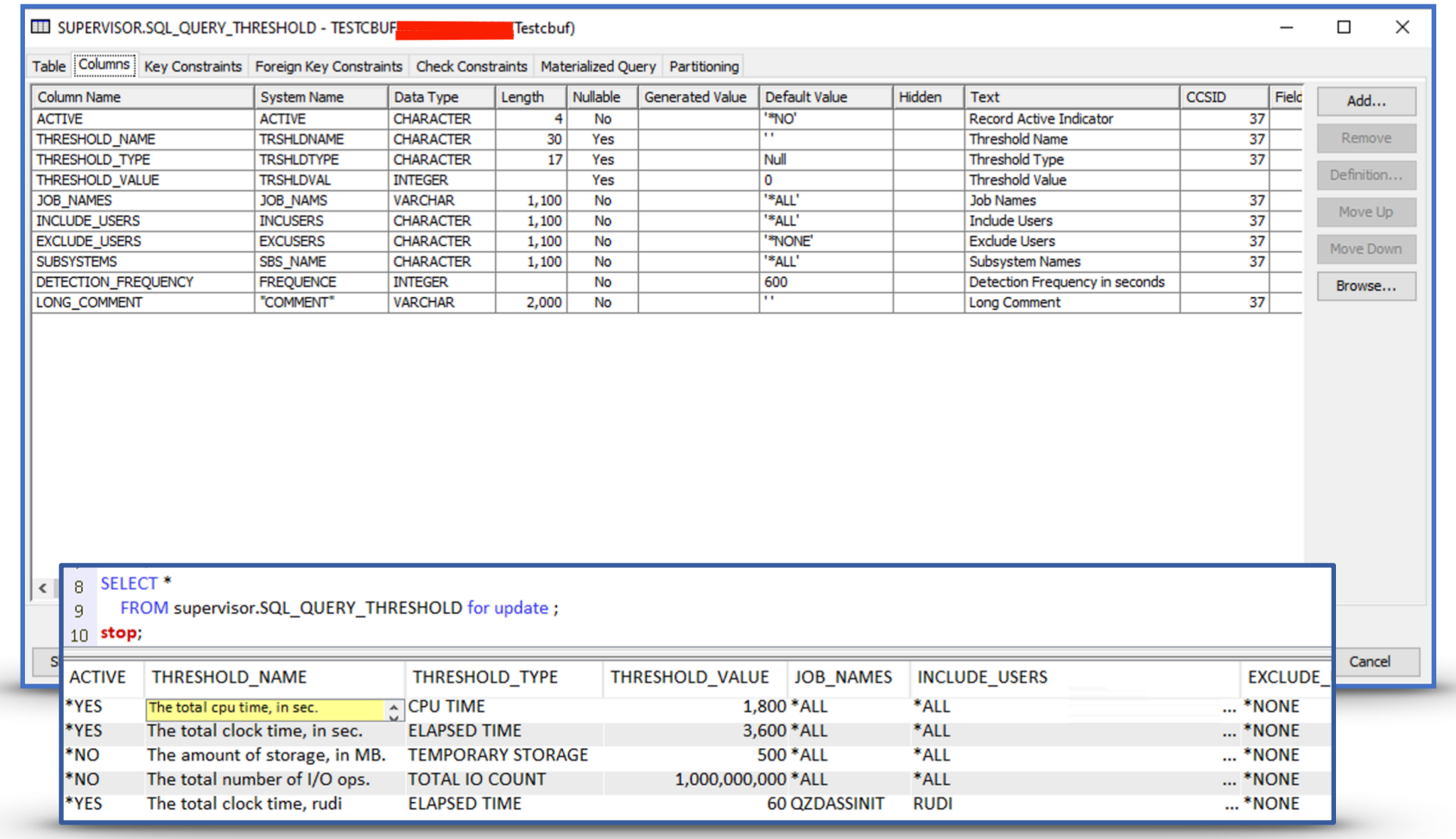
```
SELECT *
   FROM qsys2.EXIT_PROGRAM_INFO
   WHERE exit_point_name LIKE 'QIBM_QQQ_QRY_SUPER';
```

The Building

Schemas

Why a SQL Query Threshold table?

- Flexibility
- Transparency
- Security
- Traceability



SQL_QUERY_THRESHOLD Table

Field definition based on ADD_QUERY_THRESHOLD procedure

threshold-name

A character or graphic string that provides a name for the threshold. The name can be up to 30 characters long and can contain any characters including blanks. The name cannot be the same as an existing threshold name.

threshold-type

A character or graphic string that specifies the type of the threshold to be enforced.

CPU TIME

The total processing unit time used by the query, in seconds.

ELAPSED TIME

The total clock time, in seconds.

TEMPORARY STORAGE

The amount of storage, in megabytes (MB), that the query allocates.

TOTAL IO COUNT

The total number of I/O operations.

threshold-value

A big integer value that contains the threshold value, in units defined by the specified *threshold-type*. The value must be greater than 0.

job-names

A character or graphic string that specifies up to 100 job names separated by either a blank or a comma. Each job name can end with a wildcard character. For example, 'QPADEV*' indicates that any job name starting with the characters 'QPADEV' is a match.

Only jobs running with a matching job name are eligible to be supervised. Can contain the following special value:

***ALL**

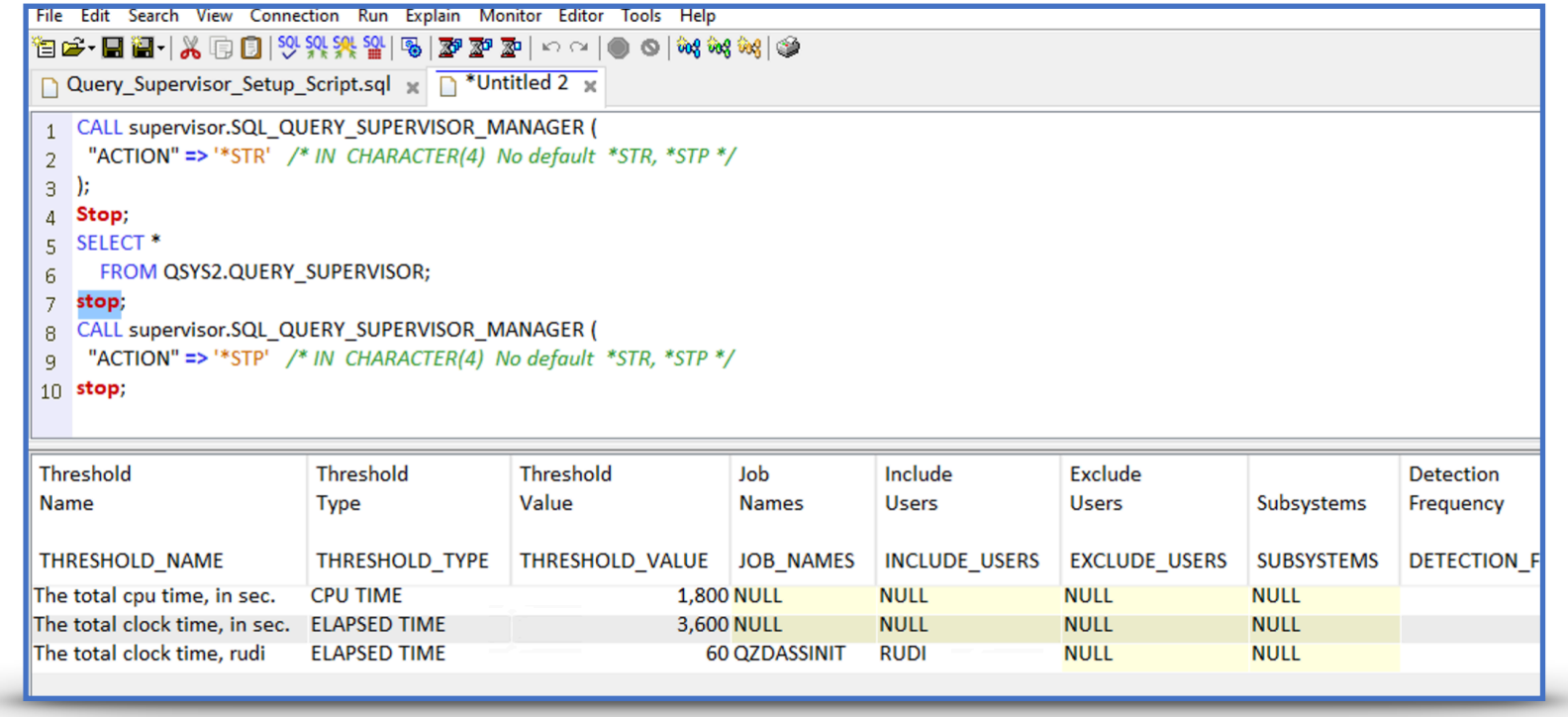
All jobs are supervised. This is the default.

The Building

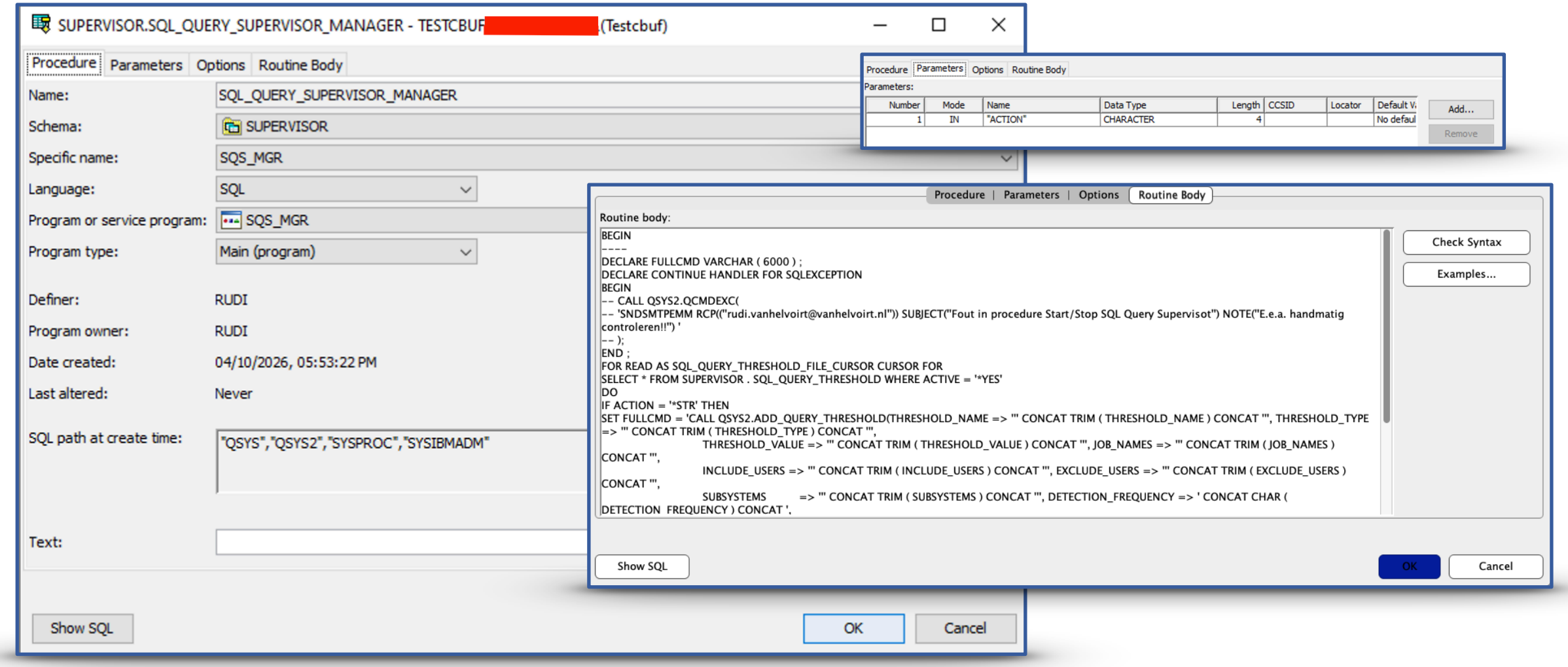
Schemas

A SQL Procedure using the Query Threshold table

- Flexibility
- Transparency
- Security
- Traceability



SQL_QUERY_SUPERVISOR_MANAGER Procedure



The Building

Exit Programs

Do not reinvent the wheel

- Get started
- Start learning
- Build confidence
- Use the Query Supervisor example exit programs

Query Supervisor example exit programs

Last Updated: 2025-10-07

The exit programs in this section demonstrate some of the ways an exit program can be used when a Query Supervisor threshold is reached. They are provided to enable quick and easy adoption of Query Supervisor.

The details of the exit program interface can be found here: Query Supervisor Exit Program

- **Exit program to send message to QSYSOPR**
  This Query Supervisor exit program sends an informational message to the QSYSOPR message queue.
- **Exit program to end query**
  This Query Supervisor exit program shows how to request that the query that was running when the threshold was reached is to be stopped.
- **Exit program to dump plan cache information for query**
  This Query Supervisor exit program dumps plan cache information about the query that reached a threshold.
- **Exit program to log information using a data queue**
  This Query Supervisor exit program shows how to use an asynchronous job to log information about the query that reported a threshold.

Exit program to send message to QSYSOPR

SQS_SNDMSG.CLLE

SUPERVISOR > QCLLESRC > SQS_SNDMSG.CLLE

```
57  /*****************************************************************/
58  /**********                  Execute                    **********/
59  /*****************************************************************/
60
61   /* INIT RETURN CODE TO CONTINUE RUNNING QUERY */
62
63              CHGVAR     VAR(&RC) VALUE(0)
64
65   /* GET VALUES FROM THE INPUT FORMAT */
66              CHGVAR     VAR(&JOBNAME) VALUE(%SST(&QRYS0100 13 10))
67              CHGVAR     VAR(&JOBUSER) VALUE(%SST(&QRYS0100 23 10))
68              CHGVAR     VAR(&JOBNBR)  VALUE(%SST(&QRYS0100 33  6))
69              CHGVAR     VAR(&THRESHTYPE) VALUE(%SST(&QRYS0100 135 30))
70              CHGVAR     VAR(&BINHIGH) VALUE(%BINARY(&QRYS0100 165 4))
71              CHGVAR     VAR(&BINLOW)  VALUE(%BINARY(&QRYS0100 169 4))
72              CHGVAR     VAR(&THRESHVAL) VALUE(%DEC(&BINHIGH 15 0) * +
73                           &MAXINT + %DEC(&BINLOW 15 0))
74
75    /* GENERATE MESSAGE TEXT STRING  */
76              CHGVAR     VAR(&MSGTXT) VALUE('QUERY SUPERVISOR +
77                           THRESHOLD TYPE ' *BCAT &THRESHTYPE *TCAT  +
78                           ', THRESHOLD VALUE ' *BCAT                +
79                           %CHAR(&THRESHVAL) *BCAT 'REACHED IN JOB ' +
80                           *BCAT &JOBNBR *TCAT '/' *TCAT &JOBUSER    +
81                           *TCAT '/' *TCAT &JOBNAME  )
82              CHGVAR     VAR(&MSGLEN) VALUE(%LEN(&MSGTXT))
83              CHGVAR     VAR(%BINARY(&MSGDTA 1 2)) VALUE(&MSGLEN)
84              CHGVAR     VAR(&MSGDTA) VALUE(&MSGDTA *TCAT &MSGTXT)
85              SNDPGMMSG  MSGID(SQL7064) MSGF(QSQLMSG) MSGDTA(&MSGDTA) +
86                           TOMSGQ(*SYSOPR)
87
88  /*****************************************************************/
89  /**********            Normal end of program            **********/
90  /*****************************************************************/
```

Exit program to end query

SQS_ENDQRY.SQLRPGLE

SUPERVISOR > QRPGLESRC > SQS_ENDQRY.SQLRPGLE > SQS_endqry

```
55
56  dcl-proc SQS_endqry; // Main procedure
57    dcl-pi *n;
58      input likeds(QQQ_QRYSV_QRYS0100_t);
59      rc int(10);
60    end-pi;
61
62  //*************************************************************************
63  // Init the return code to continue running the query
64  //*************************************************************************
65    rc = 0;
66
67  //* Check Threshold type against SQL Query Supervisor Thresholds Table  */
68
69    EXEC SQL
70        SELECT THRESHOLD_VALUE into :sqs_threshold_value FROM SUPERVISOR.SQS_TRSHLD
71         WHERE THRESHOLD_NAME = :input.Threshold_Name and ACTIVE = '*YES';
72
73    If SQLSTATE = '00000';
74      //note 00000 = no errors or warning
75      if input.Threshold_Consumption_Value >= sqs_threshold_value;
76        rc = 1; //* Terminate the query */
77      //     02000 = no data
78      // <do somthing?>
79    endif;
80    endif;
81     return;
82  end-proc;
```

Exit program to dump plan cache information for query

SQS_DMPQRY.CLLE

SUPERVISOR > QCLLESRC > SQS_DMPQRY.CLLE

```
52  /*****************************************************************/
54  /* INIT RETURN CODE TO CONTINUE RUNNING QUERY */
55
56      CHGVAR     VAR(&RC) VALUE(0)
57
58  /* GET VALUES FROM THE INPUT FORMAT */
59
60      CHGVAR     VAR(&BINHIGHU) VALUE(%BINARY(&QRYS0100 67 4))
61      CHGVAR     VAR(&BINLOW) VALUE(%BINARY(&QRYS0100 71 4))
62      CHGVAR     VAR(&PLANID) VALUE(%DEC(&BINHIGHU 15 0)  * +
63                   &MAXINT + %DEC(&BINLOW 15 0))
64
65  /* SUBMIT JOB TO DUMP THE PLAN CACHE FOR THIS QUERY */
66      SBMJOB     CMD(RUNSQL SQL('CALL QSYS2.DUMP_PLAN_CACHE(' +
67                   *BCAT 'FILESCHEMA => ''SUPERVISOR'', +
68                   FILENAME => ''PLANDUMPS'', +
69                   PLAN_IDENTIFIER=> ''' *TCAT +
70                   %CHAR(&PLANID) *TCAT ''')')) +
71                   JOBQ(QUSRNOMAX)
72
73  /*****************************************************************/
74  /**********            Normal end of program            **********/
75  /*****************************************************************/
```

SQL procedure used to log information using a data queue

```
DECLARE V_QUERY_IDENTIFIER VARCHAR ( 8 ) CCSID 1208 ;
DECLARE V_QUERY_PLAN_IDENTIFIER DECIMAL ( 20 , 0 ) ;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
BEGIN
GET DIAGNOSTICS CONDITION 1
LOCAL_SQLCODE = DB2_RETURNED_SQLCODE , LOCAL_SQLSTATE = RETURNED_SQLSTATE ,
V_MESSAGE_TEXT = MESSAGE_TEXT ;
CALL SYSTOOLS . LPRINTF ( 'process_SQS_data_queue() - failed with SQLCODE:' CONCAT
ERROR_SQLCODE CONCAT ' SQLSTATE:' CONCAT ERROR_SQLSTATE CONCAT ' and MESSAGE:'
CONCAT V_MESSAGE_TEXT ) ;
END ; /* end of continue handler */
WHILE ( 1 = 1 ) DO
SELECT MESSAGE_DATA_BINARY
INTO V_MESSAGE_DATA_BINARY
FROM
TABLE (
QSYS2 . RECEIVE_DATA_QUEUE (
DATA_QUEUE => 'SQS_DQ' , DATA_QUEUE_LIBRARY => 'SUPERVISOR' ,
WAIT_TIME => - 1 )  -- any negative means to wait forever for the next message
) ;

-- Convert the UTF16 data to UTF8
SET V_MESSAGE_DATA_BINARY_LENGTH = LENGTH ( V_MESSAGE_DATA_BINARY ) / 2 ;
SET V_MESSAGE_DATA_UTF8 = (
SELECT
INTERPRET (
VARBINARY_FORMAT ( HEX ( V_MESSAGE_DATA_BINARY_LENGTH ) ) CONCAT V_MESSAGE_DATA_BINARY
AS DBCLOB ( 63 K ) CCSID 1200 )
FROM SYSIBM . SYSDUMMY1 ) ;
```

SQL procedure used to log information using a data queue

```
CREATE TABLE SUPERVISOR.SUPERVISOR_LOG FOR SYSTEM NAME SUPERLOG (
        THRESHOLD_TIMESTAMP FOR COLUMN WHENHIT    TIMESTAMP DEFAULT NULL ,
        JOB_NAME VARCHAR(10) CCSID 37 DEFAULT NULL ,
        JOB_USER VARCHAR(10) CCSID 37 DEFAULT NULL ,
        JOB_NUMBER FOR COLUMN JOBNUM     VARCHAR(6) CCSID 37 DEFAULT NULL ,
        SUBSYSTEM VARCHAR(10) CCSID 37 DEFAULT NULL ,
        USER_NAME VARCHAR(10) CCSID 37 DEFAULT NULL ,
        THRESHOLD_NAME FOR COLUMN THRESHNAME VARCHAR(30) CCSID 1208 DEFAULT NULL ,
        THRESHOLD_TYPE FOR COLUMN THRESHTYPE VARCHAR(30) CCSID 1208 DEFAULT NULL ,
        THRESHOLD_CONSUMPTION_VALUE FOR COLUMN THRESHVAL  BIGINT DEFAULT NULL ,
        OPERATION_TYPE FOR COLUMN OP        SMALLINT DEFAULT NULL ,
        SQL_STATEMENT_TEXT FOR COLUMN STMTTEXT   VARCHAR(10000) CCSID 1208 DEFAULT NULL ,
        HOST_VARIABLE_LIST FOR COLUMN HVLIST     VARCHAR(10000) CCSID 1208 DEFAULT NULL ,
        CLIENT_ACCTNG FOR COLUMN "ACCTNG" VARCHAR(255) CCSID 1208 DEFAULT NULL ,
        CLIENT_APPLNAME FOR COLUMN "APPLNAME" VARCHAR(255) CCSID 1208 DEFAULT NULL ,
        CLIENT_PROGRAMID FOR COLUMN "PROGRAMID" VARCHAR(255) CCSID 1208 DEFAULT NULL ,
        CLIENT_USERID FOR COLUMN "USERID" VARCHAR(255) CCSID 1208 DEFAULT NULL ,
        CLIENT_WRKSTNNAME FOR COLUMN "WRKSTNNAME" VARCHAR(255) CCSID 1208 DEFAULT NULL ,
        QUERY_IDENTIFIER FOR COLUMN QRO_HASH   VARCHAR(8) CCSID 1208 DEFAULT NULL ,
        QUERY_PLAN_IDENTIFIER FOR COLUMN PLAN_ID    DECIMAL(20, 0) DEFAULT NULL )

        RCDFMT SUPERLOG   ;

LABEL ON TABLE SUPERVISOR.SUPERVISOR_LOG
        IS 'SQL Query Supervisor Log Table' ;

GRANT ALTER , DELETE , INDEX , INSERT , REFERENCES , SELECT , UPDATE
ON SUPERVISOR.SUPERVISOR_LOG TO RUDI WITH GRANT OPTION ;
```

```
-- Run once only
CL:CRTDTAQ DTAQ(SUPERVISOR/SQS_DQ) MAXLEN(45000) FORCE(*YES) SIZE(*MAX2GB 1000)
   AUTORCL(*YES) TEXT('SQL Query Supervisor Threshold Processor');
```

```
-- Make sure this job is submitted after every IPL
CL:SBMJOB CMD(RUNSQL SQL('call supervisor.process_SQS_data_queue()') COMMIT(*NONE)) JOB(SQS_MGR)
JOBQ(QUSRNOMAX) INLASPGRP(*NONE) LOG(4 00 *SECLVL) JOBMSGQFL(*PRTWRAP) CCSID(37);
stop;
```

Bring it all together

```
SELECT *
   FROM QSYS2.QUERY_SUPERVISOR;
Stop;
SELECT *
   FROM supervisor.SQL_QUERY_THRESHOLD for update ;
stop;
```

```
CALL
supervisor.SQL_QUERY_SUPERVISOR_MANAGER (
  "ACTION" => '*STR');
```

Bring it all together

```
-- Review Query Supervisor Exit Point Program
SELECT *
   FROM qsys2.EXIT_PROGRAM_INFO
   WHERE exit_point_name LIKE 'QIBM_QQQ_QRY_SUPER';
stop;
-- Unregister Exit Point Programs
CL:RMVEXITPGM EXITPNT(QIBM_QQQ_QRY_SUPER) FORMAT(QRYS0100) PGMNBR(*ALL);
stop;
```

Bring it all together

```
-- Register Exit Point Programs
CL:ADDEXITPGM EXITPNT(QIBM_QQQ_QRY_SUPER) FORMAT(QRYS0100)
                PGMNBR(*LOW) PGM(supervisor/SQS_SNDDTQ) THDSAFE(*YES)
                TEXT('Query Supervisor send data to *DTAQ');

-- Activation Command => SQL Query Supervisor - Send Message to QSYSOPR
CL:ADDEXITPGM EXITPNT(QIBM_QQQ_QRY_SUPER) FORMAT(QRYS0100)
                PGMNBR(*LOW) PGM(supervisor/SQS_SNDMSG) THDSAFE(*YES)
                TEXT('Query Supervisor send message to QSYSOPR');

-- Activation Command => SQL Query Supervisor - Dump Query
CL:ADDEXITPGM EXITPNT(QIBM_QQQ_QRY_SUPER) FORMAT(QRYS0100)
                PGMNBR(*LOW) PGM(supervisor/SQS_DMPQRY) THDSAFE(*YES)
                TEXT('Query supervisor dump plan cache');

-- Activation Command => SQL Query Supervisor - End Query
CL:ADDEXITPGM EXITPNT(QIBM_QQQ_QRY_SUPER) FORMAT(QRYS0100)
                PGMNBR(*LOW) PGM(supervisor/SQS_ENDQRY) THDSAFE(*YES)
                TEXT('Query Supervisor End Query');
stop;
```

Bring it all together

```
-- testing SQL statement
SELECT qsys2.qcmdexc('DLYJOB 5'),
       HIST.*
    FROM TABLE (
           QSYS2.HISTORY_LOG_INFO(START_TIME => CURRENT TIMESTAMP - 1 HOUR)
        ) HIST
    FETCH FIRST 150 ROWS ONLY;
```

Q & A

Thank you