Sérgio Carvalho Fonseca

De:	Sérgio Carvalho Fonseca
Enviado em:	sexta-feira, 10 de outubro de 2008 12:12
Para:	Sérgio Carvalho Fonseca
Assunto:	Locks
Sinalizador de acompanhamento:	Acompanhar
Status do sinalizador:	Concluída
Categorias:	UTIL

http://www.sommarskog.se/sqlutil/aba_lockinfo.html

```
DROP PROCEDURE aba_lockinfo

/*------------------------------------------------------------------

$Header: /abasql/aba_lockinfo_sqlmm_sp3.sp 8 06-05-23 21:27 Sommar $

This SP lists locking information for all active processes, that is

processes that have a lock or are not AWAITING COMMAND. Information

about all locked objects are included, as well the last command sent

from the client. Note that this command is tacked out afterwards with

DBCC INPUTBUFFER, and may be out of sync with the rest of the data.

The original source for the SP was taken from the undocumented

system procedure sp_lockinfo.

This version works only in SQL2000 SP3. There are separate versions for

SQL6.5, SQL7 and SQL 2000 pre-SP3.

$History: aba_lockinfo_sqlmm_sp3.sp $

*

* ***************** Version 8 *****************

* User: Sommar Date: 06-05-23 Time: 21:27

* Updated in $/abasql

* suser_name(sid) does not always return the login name, so use

* sysprocess.loginame as a fallback.

*
```

```
* ***************** Version 7 *****************

* User: Sommar Date: 06-03-21 Time: 21:09

* Updated in $/abasql

* Bug fix: I did not quote database names with quotename(), so you would

* get a syntax error with databases with special characters.

*

* ***************** Version 6 *****************

* User: Sommar Date: 04-12-27 Time: 13:26

* Updated in $/abasql

*

* ***************** Version 5 *****************

* User: Sommar Date: 04-02-23 Time: 11:08

* Updated in $/abasql

* Remove line break from curstmt.

*

* ***************** Version 4 *****************

* User: Sommar Date: 04-02-22 Time: 23:20

* Updated in $/abasql

* Thorough rewrite for better performance.

*

* ***************** Version 3 *****************

* User: Sommar Date: 03-05-25 Time: 11:22

* Updated in $/abasql

* stmt_start is an offset, so we should add 1 to it.

*

* ***************** Version 2 *****************

* User: Sommar Date: 02-12-21 Time: 23:25

* Updated in $/abasql
* stmt_end = -1 means that current statement extents until end of text.
*
* ***************** Version 1 *****************
* User: Sommar Date: 02-12-21 Time: 23:08
* Created in $/abasql
*
* ***************** Version 10 *****************
* User: Sommar Date: 02-05-04 Time: 18:35
* Updated in $/abasql
* Current time was saved in a format that later would cause conversion
* error with some dateformat settings.
*
* ***************** Version 9 *****************
* User: Sommar Date: 02-03-24 Time: 0:39
* Updated in $/abasql
* The setting of @minspid had disappeared.
*
* ***************** Version 8 *****************
* User: Sommar Date: 02-03-22 Time: 16:02
* Updated in $/abasql
* Performance enhancements. No longer need for separate database, uses
* temp tables. Lots of KEEPFIXED PLAN to avoid recompilations.
* Unnecessary use of dynamic SQL removed. Support for SQL7 removed.
*
* ***************** Version 7 *****************
* User: Sommar Date: 01-11-26 Time: 15:38
* Updated in $/abasql
* Now that's news! There might be more than one ecid in sysprocesses per
* spid. Let's handle that!
*
* ***************** Version 6 *****************
* User: Sommar Date: 01-07-16 Time: 22:27
* Updated in $/abasql
* Extensive rewrite. Default is now to group locks to reduce the amount
* data when there are many locks. Also handling the case that a process
* may not exist in sysprocesses. Handle also application locks.
*
* ***************** Version 5 *****************
* User: Sommar Date: 01-03-17 Time: 22:07
* Updated in $/abasql
* Added SET QUOTED_IDENTIFIER OFF for the benefit of people outside
* Abaris who might have this on.
*
* ***************** Version 4 *****************
* User: Sommar Date: 00-11-07 Time: 10:59
* Updated in $/projects/dbverktyg/abasql
* Adaptions for SQL2000. Define processes that only hold a lock on a
* databaes as passive. Translate object names per database, not per
* object. Handle that last_since may overflow.
*
* ***************** Version 2 *****************
* User: Sommar Date: 00-02-09 Time: 13:19
* Updated in $/projects/dbverktyg/abasql
* Stupid bug: last_since was 10 times too big.
*
* ***************** Version 1 *****************
* User: Sommar Date: 00-01-06 Time: 17:46

* Created in $/projects/dbverktyg/abasql

*

* ***************** Version 2 *****************

* User: Sommar Date: 99-12-21 Time: 19:39

* Updated in $/projects/dbverktyg/abasql

* Hide system processes.

*

* ***************** Version 1 *****************

* User: Sommar Date: 99-12-21 Time: 19:33

* Created in $/projects/dbverktyg/abasql

---------------------------------------------------------------------*/

CREATE PROCEDURE aba_lockinfo @processes tinyint = 0,

@details bit = 0,

@fancy bit = 0 AS

----------------------------------------------------------------------

-- The following temp tables are work tables that are involved in dynamic

-- SQL or INSERT EXEC, and therefore cannot be table variables.

----------------------------------------------------------------------

-- Output from DBCC INPUTBUFFER.

CREATE TABLE #inputbuffer (eventtype nvarchar(30) NULL,

params int NULL,

eventinfo nvarchar(255) NULL)

-- Holds all object to be identified.

CREATE TABLE #objects (dbid smallint NOT NULL,

objid int NOT NULL,

indid tinyint NOT NULL,

objname nvarchar(170) NULL,
PRIMARY KEY CLUSTERED (dbid, objid, indid))

-- Used for the fancy result.

CREATE TABLE #result (

ident int IDENTITY,

spid smallint NOT NULL,

ecid smallint NOT NULL,

cnt int NULL,

login sysname NOT NULL,

prcstatus nvarchar(30) NOT NULL,

command nvarchar(16) NOT NULL,

dbname sysname NOT NULL,

host nvarchar(128) NOT NULL,

appl nvarchar(128) NOT NULL,

opntrn varchar(5) NOT NULL,

lvl char(3) NOT NULL,

blkby varchar(5) NOT NULL,

locktype nvarchar(70) NOT NULL,

ownertype nvarchar(70) NOT NULL,

object nvarchar(170) NULL,

rsctype nvarchar(70) NOT NULL,

lstatus nvarchar(70) NOT NULL,

waittime varchar(10) NOT NULL,

waittype binary(2) NULL,

cpu varchar(10) NOT NULL,

physio varchar(10) NOT NULL,

memusg varchar(10) NOT NULL,

now char(12) NOT NULL,

login_time char(16) NOT NULL,
last_batch char(16) NOT NULL,

last_since varchar(11) NOT NULL,

delay varchar(10) NOT NULL,

inputbuffer varchar(255) NOT NULL,

current_sp nvarchar(255) NOT NULL,

curstmt nvarchar(255) NOT NULL,

stmtoff varchar(15) NOT NULL,

last bit NOT NULL DEFAULT 0)

---------------------------------------------------------------------

-- Then table variables for locks and processes. Input from syslockinfo and

-- sysprocesses augmented with other material.

---------------------------------------------------------------------

DECLARE @procs TABLE (

spid smallint NOT NULL,

ecid smallint NOT NULL,

active bit NOT NULL DEFAULT 1,

login sysname NULL,

status nvarchar(30) NULL,

dbname sysname NULL,

host nvarchar(128) NULL,

command nvarchar(16) NULL,

appl nvarchar(128) NULL,

opntrn smallint NULL,

blking smallint NOT NULL,

blkby smallint NULL,

blklvl smallint NOT NULL,

waittime int NULL,

waittype binary(2) NULL,
cpu int NULL,

physio bigint NULL,

memusage int NULL,

now datetime NOT NULL,

login_time char(16) NULL,

last_batch char(16) NULL,

last_since numeric(10,3) NULL,

sql_handle binary(20) NOT NULL,

stmt_start int NOT NULL,

stmt_end int NOT NULL,

current_sp int NULL,

curdbid smallint NULL,

curstmt nvarchar(255) NULL,

delay int NOT NULL DEFAULT 0,

inputbuffer nvarchar(255) NOT NULL DEFAULT '',

PRIMARY KEY (spid, ecid))

DECLARE @locks TABLE (

ident int IDENTITY,

spid smallint NOT NULL,

ecid smallint NOT NULL,

cnt int NULL,

req_mode tinyint NOT NULL,

rsc_type tinyint NOT NULL,

req_status tinyint NOT NULL,

req_ownertype smallint NOT NULL,

dbid smallint NOT NULL,

objid int NOT NULL,

indid tinyint NOT NULL,
rsc_text nchar(32) NULL,

refcnt smallint NULL,

activelock bit NOT NULL)

-- We tried indexing, but they seem to cost more than give.

-- UNIQUE NONCLUSTERED (spid, ecid, activelock, ident))

-- UNIQUE NONCLUSTERED (dbid, objid, indid, ident))

-----------------------------------------------------------------------

-- Local variables.

-----------------------------------------------------------------------

DECLARE @minspid int,

@objid int,

@dbid smallint,

@dbname sysname,

@qdbname nvarchar(256),

@stmt varchar(8000),

@spid smallint,

@sql_handle binary(20),

@stmt_start int,

@stmt_end int,

@spidstr varchar(10),

@inputbuff varchar(255),

@blklvl tinyint,

@now datetime

-----------------------------------------------------------------------

-- All reads are dirty! The most important reason for this is tempdb..sysobjects.

-----------------------------------------------------------------------

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

SET NOCOUNT ON
```

```
-- Processes below @minspid are system processes.

SELECT @minspid = 50, @now = getdate()

----------------------------------------------------------------------

-- First caputure all locks. These can be aggregate, or we can get all of them.

----------------------------------------------------------------------

IF @details = 0

BEGIN

INSERT @locks (spid, ecid, req_mode, rsc_type, req_status, req_ownertype,

dbid, objid, indid,

rsc_text,

activelock, cnt)

SELECT req_spid, req_ecid, req_mode, rsc_type, req_status, req_ownertype,

rsc_dbid, rsc_objid, rsc_indid,

CASE rsc_type WHEN 10 THEN rsc_text END,

CASE WHEN rsc_type = 2 AND req_status = 1 THEN 0 ELSE 1 END,

COUNT(*)

FROM master.dbo.syslockinfo

GROUP BY req_spid, req_ecid, req_mode, rsc_type, req_status, req_ownertype,

rsc_dbid, rsc_objid, rsc_indid,

CASE rsc_type WHEN 10 THEN rsc_text END,

CASE WHEN rsc_type = 2 AND req_status = 1 THEN 0 ELSE 1 END

END

ELSE

BEGIN

INSERT @locks (spid, ecid, req_mode, rsc_type, req_status, req_ownertype,

dbid, objid, indid, rsc_text, refcnt,

activelock)

SELECT req_spid, req_ecid, req_mode, rsc_type, req_status, req_ownertype,
rsc_dbid, rsc_objid, rsc_indid, rsc_text, req_refcnt,

CASE WHEN rsc_type = 2 AND req_status = 1 THEN 0 ELSE 1 END

FROM master.dbo.syslockinfo

END

---------------------------------------------------------------------

-- Then get the processes. We filter here for active processes once for all

---------------------------------------------------------------------

INSERT @procs(spid, ecid, login, status,

dbname,

host, command, appl, opntrn,

blking, blkby, blklvl,

waittime, waittype, cpu, physio, memusage, now,

login_time,

last_batch,

last_since, sql_handle, stmt_start, stmt_end)

SELECT p.spid, p.ecid, coalesce(suser_sname(p.sid), p.loginame), rtrim(p.status),

CASE WHEN p.dbid > 0 THEN db_name(p.dbid) ELSE '' END,

rtrim(p.hostname), rtrim(p.cmd), rtrim(p.program_name), p.open_tran,

0, p.blocked, 0,

p.waittime, p.waittype, p.cpu, p.physical_io, p.memusage, @now,

convert(char(7), p.login_time, 12) + convert(char(8), p.login_time, 8),

convert(char(7), p.last_batch, 12) + convert(char(8), p.last_batch, 8),

CASE WHEN datediff(DAY, p.last_batch, @now) > 20

THEN NULL

ELSE datediff(MS, p.last_batch, @now) / 1000.000

END, sql_handle, stmt_start, stmt_end

FROM master.dbo.sysprocesses p

WHERE @processes > 0 OR
```

```
(upper(p.cmd) <> 'AWAITING COMMAND' AND

p.spid >= @minspid AND

p.spid <> @@spid) OR

p.open_tran > 0 OR

p.blocked > 0 OR

(EXISTS (SELECT *

FROM @locks l

WHERE l.spid = p.spid

AND l.activelock = 1) AND spid <> @@spid)

----------------------------------------------------------------------

-- Mark inactive processes; this is only interesting if @processes = 1,

-- because with @processes = 0 we only have active now.

----------------------------------------------------------------------

IF @processes = 1

BEGIN

UPDATE @procs

SET active = 0

FROM @procs p

WHERE NOT EXISTS (SELECT *

FROM @locks l

WHERE p.spid = l.spid

AND p.ecid = l.ecid

AND l.activelock = 1

AND p.spid <> @@spid

AND p.spid >= @minspid)

AND (p.command = 'AWAITING COMMAND' OR p.spid < @minspid OR p.spid = @@spid)

AND p.blkby = 0

END
```

```
----------------------------------------------------------------------
-- Get input buffers and fn_get_sql data. Note that only the main thread,
-- ecid = 0 is of interest.
----------------------------------------------------------------------
DECLARE C1 CURSOR LOCAL FOR
SELECT str(spid), spid, sql_handle, stmt_start, stmt_end
FROM @procs
WHERE (@processes = 2 OR active = 1)
AND ecid = 0
AND login IS NOT NULL
OPEN C1
WHILE 1 = 1
BEGIN
FETCH C1 INTO @spidstr, @spid, @sql_handle, @stmt_start, @stmt_end
IF @@fetch_status <> 0
BREAK
DELETE #inputbuffer
INSERT #inputbuffer
EXEC ('DBCC INPUTBUFFER (' + @spidstr + ') WITH NO_INFOMSGS')
SELECT @inputbuff = ' '
SELECT @inputbuff = rtrim(eventinfo) FROM #inputbuffer
-- Replace line breaks with spaces.
SET @inputbuff = replace(@inputbuff, char(10) + char(13), ' ')
SET @inputbuff = replace(@inputbuff, char(10), ' ')
SET @inputbuff = replace(@inputbuff, char(13), ' ')
IF @sql_handle <> 0x0
BEGIN
SELECT @objid = objectid,
@dbid = dbid,

@stmt = substring(

CASE WHEN @stmt_start >= 0

THEN substring(

text, (@stmt_start + 2)/2,

CASE @stmt_end

WHEN -1 THEN 255

ELSE (@stmt_end - @stmt_start + 2) / 2

END)

END, 1, 255)

FROM ::fn_get_sql(@sql_handle)

SET @stmt = replace(@stmt, char(10) + char(13), ' ')

SET @stmt = replace(@stmt, char(10), ' ')

SET @stmt = replace(@stmt, char(13), ' ')

END

ELSE

SELECT @stmt = '', @objid = NULL, @dbid = NULL

UPDATE @procs

SET inputbuffer = coalesce(@inputbuff, ''),

delay = datediff(ms, now, @now),

current_sp = @objid,

curdbid = @dbid,

curstmt = @stmt

FROM @procs p

WHERE spid = @spid

AND ecid = 0

END

DEALLOCATE C1
```

```
----------------------------------------------------------------------

-- Delete inactive processes from @locks.

----------------------------------------------------------------------

IF @processes = 0

BEGIN

DELETE @locks

FROM @locks l

WHERE NOT EXISTS (SELECT *

FROM @procs p

WHERE p.spid = l.spid

AND p.active = 1)

END

----------------------------------------------------------------------

-- Get name of objects. Need to do this per database.

----------------------------------------------------------------------

INSERT #objects (dbid, objid, indid)

SELECT dbid, objid, indid

FROM @locks

WHERE dbid > 0 AND objid > 0

UNION

SELECT curdbid, current_sp, 0

FROM @procs

WHERE curdbid > 0 AND current_sp > 0

DECLARE C2 CURSOR LOCAL FOR

SELECT DISTINCT dbid, db_name(dbid), quotename(db_name(dbid)) FROM #objects

OPEN C2

WHILE 1 = 1

BEGIN
```

```
FETCH C2 INTO @dbid, @dbname, @qdbname

IF @@fetch_status <> 0

BREAK

-- Set database.owner.name(.index) of all objects in #objects.

SELECT @stmt =

' UPDATE #objects

SET objname = ''' + @dbname + '.'' + u.name + ''.'' + o.name +

CASE coalesce(t.indid, 0) WHEN 0 THEN '''' ELSE ''.'' + i.name END

FROM #objects t

JOIN ' + @qdbname + '.dbo.sysobjects o ON t.objid = o.id

JOIN ' + @qdbname + '.dbo.sysusers u ON u.uid = o.uid

LEFT JOIN ' + @qdbname + '.dbo.sysindexes i ON t.indid = i.indid

AND t.objid = i.id

WHERE t.dbid = ' + str(@dbid) + '

AND t.objid > 0 '

EXEC (@stmt)

END

DEALLOCATE C2

--------------------------------------------------------------------

-- Flag blocking and blocked processes

--------------------------------------------------------------------

UPDATE @procs

SET blking = 1

FROM @procs p

WHERE EXISTS (SELECT *

FROM @procs p2

WHERE p.spid = p2.blkby)

UPDATE @procs
```

```
SET blklvl = 1

WHERE blking = 1

AND blkby = 0

SELECT @blklvl = 1

-- Find out place in the queue for blocked processes.

WHILE EXISTS (SELECT * FROM @procs WHERE blkby > 0 AND blklvl = 0) AND

@blklvl < 20

BEGIN

UPDATE p1

SET blklvl = @blklvl + 1

FROM @procs p1

JOIN @procs p2 ON p1.blkby = p2.spid

WHERE p1.blkby > 0

AND p1.blklvl = 0

AND p2.blklvl = @blklvl

SELECT @blklvl = @blklvl + 1

END


---------------------------------------------------------------------

-- For Plain results we are ready to return now.

---------------------------------------------------------------------

IF @fancy = 0

BEGIN

SELECT spid = coalesce(p.spid, l.spid),

ecid = coalesce(p.ecid, l.ecid),

cnt = CASE @details

WHEN 0 THEN coalesce(l.cnt, 0)

WHEN 1 THEN coalesce(l.refcnt, 0)
```

```
END,

login = coalesce(p.login, ''),

prcstatus = coalesce(p.status, ''),

command = coalesce(p.command, ''),

dbname = coalesce(p.dbname, ''),

host = coalesce(p.host, ''),

appl = coalesce(p.appl, ''),

opntrn = coalesce(convert(varchar(5), p.opntrn), ''),

lvl = CASE coalesce(p.blklvl, 0)

WHEN 0 THEN ''

WHEN 1 THEN '!!'

ELSE convert(varchar(3), p.blklvl - 1)

END,

blkby = CASE coalesce(p.blkby, 0)

WHEN 0 THEN ''

ELSE convert(varchar(5), p.blkby)

END,

locktype = coalesce(v1.name, ''),

ownertype = coalesce(v2.name, ''),

object = CASE WHEN l.rsc_type = 10 THEN rtrim(l.rsc_text)

WHEN l.rsc_type = 2 THEN rtrim(db_name(l.dbid))

WHEN l.rsc_type IS NOT NULL

THEN coalesce(o1.objname,

db_name(l.dbid) + '.MISSING?')

ELSE ''

END,

rsctype = coalesce(v3.name, ''),

lstatus = coalesce(v4.name, ''),
waittime = CASE coalesce(p.waittime, 0)

WHEN 0 THEN ''

ELSE convert(varchar(10), p.waittime)

END,

p.waittype,

cpu = coalesce(convert(varchar(10), p.cpu), ''),

physio = coalesce(convert(varchar(10), p.physio), ''),

memusg = coalesce(convert(varchar(10), p.memusage), ''),

now = convert(char(12), p.now, 114),

login_time = coalesce(p.login_time, ''),

last_batch = coalesce(p.last_batch, ''),

last_since = coalesce(str(p.last_since, 11, 3), ''),

delay = coalesce(convert(varchar(10), p.delay), ''),

inputbuffer = coalesce(p.inputbuffer, ''),

current_sp = coalesce(o2.objname, ''),

curstmt = coalesce(p.curstmt, ''),

stmtoff = coalesce(ltrim(str(stmt_start/2)), '') + '/' +

coalesce(ltrim(str(stmt_end/2)), '')

FROM (@procs p

LEFT JOIN #objects o2 ON p.curdbid = o2.dbid

AND p.current_sp = o2.objid

AND o2.indid = 0)

FULL JOIN (@locks l

LEFT JOIN master.dbo.spt_values v1 ON v1.number = l.req_mode + 1

AND v1.type = 'L'

LEFT JOIN master.dbo.spt_values v2 ON v2.number = l.req_ownertype

AND v2.type = 'LO'

LEFT JOIN master.dbo.spt_values v3 ON v3.number = l.rsc_type
AND v3.type = 'LR'

LEFT JOIN master.dbo.spt_values v4 ON v4.number = l.req_status

AND v4.type = 'LS'

LEFT JOIN #objects o1 ON l.dbid = o1.dbid

AND l.objid = o1.objid

AND l.indid = o1.indid)

ON p.spid = l.spid

AND p.ecid = l.ecid

ORDER BY spid, ecid, lstatus DESC, object

END

ELSE

BEGIN

---------------------------------------------------------------------

-- For fancy result, we save to #result, and to find suitable lengths.

---------------------------------------------------------------------

DECLARE @spidlen varchar(5),

@ecidlen varchar(5),

@cntlen varchar(5),

@loginlen varchar(5),

@statuslen varchar(5),

@dbnamelen varchar(5),

@hostlen varchar(5),

@cmdlen varchar(5),

@appllen varchar(5),

@waitlen varchar(5),

@waitreslen varchar(5),

@locktlen varchar(5),

@restlen varchar(5),
@lkstatlen varchar(5),

@lkobjlen varchar(5),

@ownertlen varchar(5),

@cpulen varchar(5),

@physiolen varchar(5),

@memlen varchar(5),

@delaylen varchar(5),

@curobjlen varchar(5),

@stmtlen varchar(5),

@stmtofflen varchar(5),

@inputlen varchar(5)

INSERT #result (spid, ecid, cnt, login, prcstatus, command, dbname, host,

appl, opntrn, lvl, blkby, locktype, ownertype, object,

rsctype, lstatus, waittime, p.waittype, cpu, physio, memusg,

now, login_time, last_batch, last_since, delay, inputbuffer,

current_sp, curstmt, stmtoff)

SELECT spid = coalesce(p.spid, l.spid),

ecid = coalesce(p.ecid, l.ecid),

cnt = CASE @details

WHEN 0 THEN coalesce(l.cnt, 0)

WHEN 1 THEN coalesce(l.refcnt, 0)

END,

login = coalesce(p.login, ''),

prcstatus = coalesce(p.status, ''),

command = coalesce(p.command, ''),

dbname = coalesce(p.dbname, ''),

host = coalesce(p.host, ''),

appl = coalesce(p.appl, ''),
opntrn = coalesce(convert(varchar(5), p.opntrn), ''),

lvl = CASE coalesce(p.blklvl, 0)

WHEN 0 THEN ''

WHEN 1 THEN '!!'

ELSE convert(varchar(3), p.blklvl - 1)

END,

blkby = CASE coalesce(p.blkby, 0)

WHEN 0 THEN ''

ELSE convert(varchar(5), p.blkby)

END,

locktype = coalesce(v1.name, ''),

ownertype = coalesce(v2.name, ''),

object = CASE WHEN l.rsc_type = 10 THEN rtrim(l.rsc_text)

WHEN l.rsc_type = 2 THEN rtrim(db_name(l.dbid))

WHEN l.rsc_type IS NOT NULL

THEN coalesce(o1.objname,

db_name(l.dbid) + '.MISSING?')

ELSE ''

END,

rsctype = coalesce(v3.name, ''),

lstatus = coalesce(v4.name, ''),

waittime = CASE coalesce(p.waittime, 0)

WHEN 0 THEN ''

ELSE convert(varchar(10), p.waittime)

END,

p.waittype,

cpu = coalesce(convert(varchar(10), p.cpu), ''),

physio = coalesce(convert(varchar(10), p.physio), ''),
memusg = coalesce(convert(varchar(10), p.memusage), ''),

now = convert(char(12), p.now, 114),

login_time = coalesce(p.login_time, ''),

last_batch = coalesce(p.last_batch, ''),

last_since = coalesce(str(p.last_since, 11, 3), ''),

delay = coalesce(convert(varchar(10), p.delay), ''),

inputbuffer = coalesce(p.inputbuffer, ''),

current_sp = coalesce(o2.objname, ''),

curstmt = coalesce(p.curstmt, ''),

stmtoff = coalesce(ltrim(str(p.stmt_start / 2)), '') + '/' +

coalesce(ltrim(str(p.stmt_end/2)), '')

FROM (@procs p

LEFT JOIN #objects o2 ON p.curdbid = o2.dbid

AND p.current_sp = o2.objid

AND o2.indid = 0)

FULL JOIN (@locks l

LEFT JOIN master.dbo.spt_values v1 ON v1.number = l.req_mode + 1

AND v1.type = 'L'

LEFT JOIN master.dbo.spt_values v2 ON v2.number = l.req_ownertype

AND v2.type = 'LO'

LEFT JOIN master.dbo.spt_values v3 ON v3.number = l.rsc_type

AND v3.type = 'LR'

LEFT JOIN master.dbo.spt_values v4 ON v4.number = l.req_status

AND v4.type = 'LS'

LEFT JOIN #objects o1 ON l.dbid = o1.dbid

AND l.objid = o1.objid

AND l.indid = o1.indid)

ON p.spid = l.spid
```

```
AND p.ecid = l.ecid

ORDER BY spid, ecid, lstatus DESC, object

-- Mark last row.

UPDATE #result

SET last = 1

FROM #result r1

JOIN (SELECT spid, ident = MAX(ident)

FROM #result

GROUP BY spid) AS r2 ON r2.ident = r1.ident

OPTION (KEEPFIXED PLAN)

-- Get all maxlengths

SELECT @spidlen = convert(varchar(5), coalesce(max(len(ltrim(str(spid)))), 1)),

@ecidlen = convert(varchar(5), coalesce(max(len(ltrim(str(ecid)))), 1)),

@cntlen = convert(varchar(5), coalesce(max(len(ltrim(str(cnt)))), 1)),

@loginlen = convert(varchar(5), coalesce(nullif(max(len(login)), 0), 1)),

@cntlen = convert(varchar(5), coalesce(nullif(max(len(ltrim(str(cnt)))), 0), 1)),

@statuslen = convert(varchar(5), coalesce(nullif(max(len(prcstatus)), 0), 1)),

@dbnamelen = convert(varchar(5), coalesce(nullif(max(len(dbname)), 0), 1)),

@hostlen = convert(varchar(5), coalesce(nullif(max(len(host)), 0), 1)),

@cmdlen = convert(varchar(5), coalesce(nullif(max(len(command)), 0), 1)),

@appllen = convert(varchar(5), coalesce(nullif(max(len(appl)), 0), 1)),

@waitlen = convert(varchar(5), coalesce(nullif(max(len(waittime)), 0), 1)),

@locktlen = convert(varchar(5), coalesce(nullif(max(len(locktype)), 0), 1)),

@lkobjlen = convert(varchar(5), coalesce(nullif(max(len(object)), 0), 1)),

@ownertlen = convert(varchar(5), coalesce(nullif(max(len(ownertype)), 0), 1)),

@restlen = convert(varchar(5), coalesce(nullif(max(len(rsctype)), 0), 1)),

@lkstatlen = convert(varchar(5), coalesce(nullif(max(len(lstatus)), 0), 1)),

@cpulen = convert(varchar(5), coalesce(nullif(max(len(cpu)), 0), 1)),
@physiolen = convert(varchar(5), coalesce(nullif(max(len(physio)), 0), 1)),

@memlen = convert(varchar(5), coalesce(nullif(max(len(memusg)), 0), 1)),

@delaylen = convert(varchar(5), coalesce(nullif(max(len(delay)), 0), 1)),

@curobjlen = convert(varchar(5), coalesce(nullif(max(len(current_sp)), 0), 1)),

@inputlen = convert(varchar(5), coalesce(nullif(max(len(inputbuffer)), 0), 1)),

@stmtlen = convert(varchar(5), coalesce(nullif(max(len(curstmt)), 0), 1)),

@stmtofflen = convert(varchar(5), coalesce(nullif(max(len(stmtoff)), 0), 1))

FROM #result

OPTION (KEEPFIXED PLAN)

-- Return the #results table with dynamic lengths.

EXEC ('SELECT spid = str(spid, ' + @spidlen + '),

ecid = str(ecid, ' + @ecidlen + '),

cnt = convert(varchar( ' + @cntlen + '), cnt),

login = convert(varchar( ' + @loginlen + '), login),

prcstatus = convert(varchar( ' + @statuslen + '), prcstatus),

command = convert(varchar( ' + @cmdlen + '), command),

dbname = convert(varchar( ' + @dbnamelen + '), dbname),

host = convert(varchar( ' + @hostlen + '), host),

appl = convert(varchar( ' + @appllen + '), appl),

opntrn,

lvl,

blkby,

locktype = convert(varchar( ' + @locktlen + '), locktype),

ownertype = convert(varchar( ' + @ownertlen + '), ownertype),

object = convert(varchar( ' + @lkobjlen + '), object),

rsctype = convert(varchar( ' + @restlen + '), rsctype),

lstatus = convert(varchar( ' + @lkstatlen + '), lstatus),

waittime = convert(varchar( ' + @waitlen + '), waittime),
waittype,

cpu = convert(varchar( ' + @cpulen + '), cpu),

io = convert(varchar( ' + @physiolen + '), physio),

memusg = convert(varchar( ' + @memlen + '), memusg),

now,

login_time,

last_batch,

last_since,

delay = convert(varchar( ' + @delaylen + '), delay),

intputbuffer = convert(varchar( ' + @inputlen + '), inputbuffer),

current_sp = convert(varchar( ' + @curobjlen + '), current_sp),

curstmt = convert(varchar( ' + @stmtlen + '), curstmt),

stmtoff = convert(varchar( ' + @stmtofflen + '), stmtoff),

CASE last WHEN 1 THEN char(10) ELSE " " END

FROM #result

ORDER BY ident')

END
```

Atenciosamente,

Sérgio Carvalho Fonseca
ITGROUP - MS GOLD Certified Partner
Tel: +55 (11) 3346-5711
Cel: +55 (11) 8331-0362