IDUG Leading the DB2 User Community since 1988

IDUG DB2 Tech Conference
Barcelona, Spain - October 2013

Table Monitors
Who, What, When, Where, Why, How

Leo Pedron
Fourth Millennium Technologies

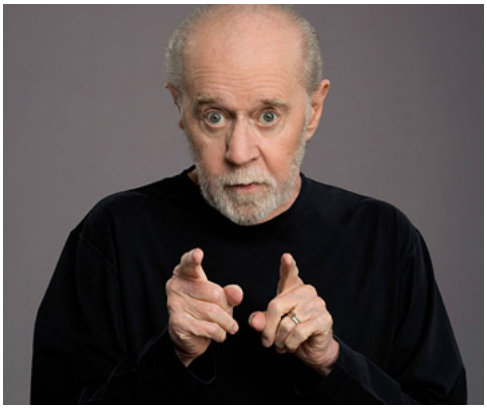
Session Code: D12
16/10/2013, 14:45 - 15:45 | Platform: DB2 for LUW

INTERNATIONAL DB2 USERS GROUP
Independent • Not-for-Profit • User Run
Founded 1988

AGENDA:

- Who?
- What?
- When?
- Where?
- Why?
- How!

- You, that's who!

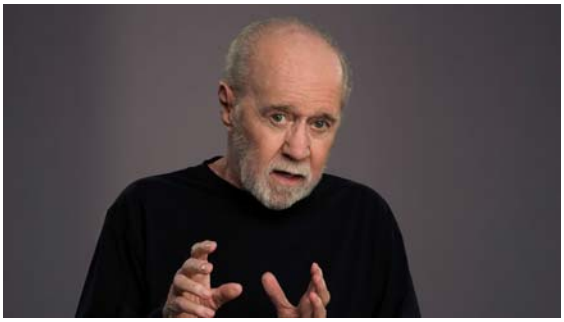


What?

- Routine monitoring
- Historical data gathering
- Problem determination

When?

- When are you looking?
  - Daily monitoring
  - The phone is ringing!



Where?

- OLD WAY:

  Log into the server and start taking snapshots
  - Could be trouble on Windows due to RDC issues
  - Need to know lots of shell commands on unix

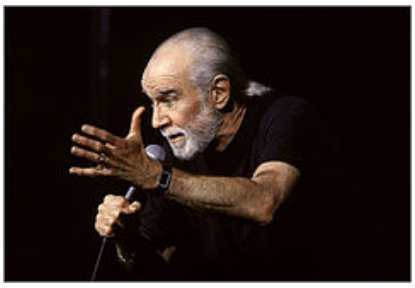
- NEW WAY:

  From anyplace you can establish a database connection
  - Locally on the server
  - From a remote server
  - From a desktop

Why?

- But why do I need to give you my old snapshots?

1. Table functions have less overhead.
2. Snaphot monitoring has been deprecated.



How?
How am I going to deal with this???

- How do I reset monitor switches?
- I have a CPU problem.
- I have IO issues.
- I have space issues.
- I have a locking problem.
- Other handy views

Use a Global Temp Table

```
declare global temporary table  base_mon_get_table as
      ( select * from table( mon_get_table( '' , '' , -2 )) x )
definition only
on commit preserve rows
not logged
on rollback preserve rows
with replace ;

insert into session.base_mon_get_table ( select * from table( mon_get_table( '' , '' , -2 )) x ) ;

create index session.x1_base_mon_get_table on session.base_mon_get_table
      ( tabname, tabschema , dbpartitionnum )
      allow reverse scans  ;

runstats on table session.base_mon_get_table and indexes all ;
```

Read data on a single partition

```
select substr(x.tabname,1,30) as tabname
      , x.TABLE_SCANS - b.TABLE_SCANS as table_scans
      , x.ROWS_READ - b.ROWS_READ as rows_read
      , x.OVERFLOW_ACCESSES  - b.OVERFLOW_ACCESSES  as over_accesses
      , x.ROWS_INSERTED - b.ROWS_INSERTED as ROWS_INSERTED
      , x.ROWS_UPDATED - b.ROWS_UPDATED as ROWS_UPDATED
      , x.ROWS_DELETED - b.ROWS_DELETED as ROWS_DELETED
      ,x.dbpartitionnum
from table(mon_get_table( '', '', 0 )) x ,
       session.base_mon_get_table b
where b.tabname = x.tabname
   and b.tabschema = x.tabschema
   and b.dbpartitionnum = x.dbpartitionnum
   and b.tabschema  in ( 'SCHEMA1', 'SCHEMA2' )
order by 3 desc
fetch first 20 rows only with ur  ;
```

Get the global view

```
select  substr(x.tabschema,1,10) as tabschema ,
        substr(x.tabname,1,30) tabname
       , sum(x.TABLE_SCANS) - sum(b.TABLE_SCANS) as table_scans
       , sum(x.ROWS_READ) - sum(b.ROWS_READ) as rows_read
       , sum(x.OVERFLOW_ACCESSES)  - sum(b.OVERFLOW_ACCESSES)  as over_accesses
       , sum(x.ROWS_INSERTED) - sum(b.ROWS_INSERTED) as ROWS_INSERTED
       , sum(x.ROWS_UPDATED) - sum(b.ROWS_UPDATED) as ROWS_UPDATED
       , sum(x.ROWS_DELETED) - sum(b.ROWS_DELETED) as ROWS_DELETED
from table(mon_get_table('', '', -2 )) x ,
        session.base_mon_get_table b
where b.tabname = x.tabname
   and b.tabschema = x.tabschema
   and b.dbpartitionnum = x.dbpartitionnum
   and b.tabschema  in ( 'SCHEMA1', 'SCHEMA2' )
group by x.tabschema, x.tabname
order by ROWS_READ desc
-- order by over_accesses desc
fetch first 20 rows only with ur ;
```

HOW?
- How am I going to deal this this???

- I have a CPU problem.
- I have IO issues.
- I have space issues.
- I have a locking problem.
- Other handy views

The OLD Way
db2 get snapshot for database ….

- Great for a top level overview
  - Connections
  - Sorting
  - Overall bufferpool
  - Locking
  - Logging
- It does many things, but there is a better way!

The NEW Way
SNAP_GET_DB (SYSIBMADM.SNAPDB)

- The direct replacement for 'get snapshot for database'
- Select * from table(snap_get_db (current server , -1)) t ;
  - This select gets back all the data as the snapshot
  - Best used by breaking down the query into subsections
  - Most of this information is best retrieved from other objects

The OLD Way
Break out the calculator….

- Take the data from the database snapshot and check percentages.
- Take several snapshots and run the output through a list of grep, awk, sed, and sort commands to get the data you want.
- Time consuming when the phone is ringing and you have to act fast.

The NEW Way
MON_DB_SUMMARY

- Now there is one system table to query and see what's happening!

```
select TOTAL_BP_HIT_RATIO_PERCENT as bp_hit_ratio,
    ROWS_READ_PER_ROWS_RETURNED,
    AVG_LOCK_WAITS_PER_ACT,
    COMPILE_PROC_TIME_PERCENT,
    IO_WAIT_TIME_PERCENT,
    LOCK_WAIT_TIME_PERCENT,
    AGENT_WAIT_TIME_PERCENT,
    NETWORK_WAIT_TIME_PERCENT,
    SECTION_PROC_TIME_PERCENT,
    SECTION_SORT_PROC_TIME_PERCENT
from sysibmadm.mon_db_summary ;
```

The OLD Way
get snapshot for dynamic sql ….

- Get you grep, awk, sort command strings ready.
- Have a heavy impact on CPU bound system.
- Writing data to a file causes more IO on a IO constrained system
- The file can get quite large based on the size of the package cache.

The NEW Way
SNAPDYN_SQL

- This is the direct replacement for the snapshot.

```
select  rank() over ( order by NUM_EXECUTIONS DESC ) as rnk,
            num_executions,
            rows_read,
            rows_written,
            stmt_sorts,
      rtrim(char(total_exec_time)) || '.' || rtrim(char(total_exec_time_ms)) as total_exec_time ,
      rtrim(char(total_USR_CPU_time)) || '.' || rtrim(char(total_USR_CPU_time_ms)) as
            usr_cpu_time ,
      rtrim(char(total_SYS_CPU_time)) || '.' || rtrim(char(total_SYS_CPU_time_ms)) as
            sys_cpu_time ,
      chr(10) || char(substr(x.stmt_text,1,150)) || chr(10) as stmt_txt
from sysibmadm.snapdyn_sql s
fetch first 10 rows only ;
```

The NEW Way
MON_GET_PKG_CACHE_STMT

```
select  rank() over ( order by QUERY_COST_ESTIMATE desc ) as rnk,
                x.NUM_EXECUTIONS,
                x.QUERY_COST_ESTIMATE,
                decimal(( float(x.STMT_EXEC_TIME) / 1000 ),30,3) as STMT_EXEC_TIME ,
                decimal(( float(x.TOTAL_ACT_TIME) / 1000 ),30,3) as TOTAL_ACT_TIME,
                decimal(( float(x.TOTAL_ACT_WAIT_TIME) / 1000 ),30,3) as TOTAL_ACT_WAIT_TIME,
                (case when TOTAL_ACT_TIME > 0 then dec((1-( float(TOTAL_ACT_TIME - TOTAL_ACT_WAIT_TIME ) /
float(TOTAL_ACT_TIME ))) * 100,5,2) else null end ) as Pct_Wait,
                ROWS_MODIFIED,
                ROWS_READ,
                ROWS_RETURNED,
                POOL_DATA_L_READS,
                POOL_INDEX_L_READS,
                POOL_READ_TIME, -- time spent reading in pages from tablespace containers
                POOL_WRITE_TIME,
                TOTAL_SORTS,
                SORT_OVERFLOWS
                chr(10) || char(substr(x.stmt_text,1,150)) || chr(10) as stmt_txt
from table(mon_get_pkg_cache_stmt ( null, null, null, -1)) x
fetch first 10 rows only ;
```

The NEW Way
MON_GET_PKG_CACHE_STMT + SNAPDYN_SQL

```
select  rank() over ( order by STMT_EXEC_TIME DESC ) as rnk,
      x.NUM_EXECUTIONS,
      x.QUERY_COST_ESTIMATE,
      decimal(( float(x.STMT_EXEC_TIME) / 1000 ),30,3) as STMT_EXEC_TIME ,
      decimal(( float(x.TOTAL_ACT_TIME) / 1000 ),30,3) as TOTAL_ACT_TIME,
      decimal(( float(x.TOTAL_ACT_WAIT_TIME) / 1000 ),30,3) as TOTAL_ACT_WAIT_TIME,
      (case when TOTAL_ACT_TIME > 0 then dec((1-( float(TOTAL_ACT_TIME - TOTAL_ACT_WAIT_TIME ) /
float(TOTAL_ACT_TIME ))) * 100,5,2) else null end ) as Pct_Wait,
      rtrim(char(total_exec_time)) || '.' || rtrim(char(total_exec_time_ms)) as total_exec_time ,
      rtrim(char(total_USR_CPU_time)) || '.' || rtrim(char(total_USR_CPU_time_ms)) as usr_cpu_time ,
      rtrim(char(total_SYS_CPU_time)) || '.' || rtrim(char(total_SYS_CPU_time_ms)) as sys_cpu_time ,
      chr(10) || char(substr(x.stmt_text,1,150)) || chr(10) as stmt_txt
from table(mon_get_pkg_cache_stmt ( null, null, null, -1)) x
      , sysibmadm.snapdyn_sql              s
where x.stmt_text = s.stmt_text
fetch first 10 rows only ;
```

The OLD Way
What's running right now?

- Get snapshot for all applications
- db2pd

The NEW Way
SNAPSHOT_STATEMENT

```
select  count(*) as cnt ,
              QUERY_COST_ESTIMATE,
              txt
from   ( select QUERY_COST_ESTIMATE,
                           varchar(substr(STMT_TEXT,1,400)) as txt,
                           STMT_START,
                           substr(PACKAGE_NAME,1,20) as pkg_name
       from table(SNAPSHOT_STATEMENT(current server,-1)) x
       where QUERY_COST_ESTIMATE > 0
      ) u
group by QUERY_COST_ESTIMATE, txt
order by cnt desc, QUERY_COST_ESTIMATE desc fetch first 20 rows only ;

select agent_id, substr(stmt_text,1,400) as txt
from table(snapshot_statement ( current server , -1 )) x
where query_cost_estimate = value-of-interest ;
```

HOW?
- How am I going to deal this this???

- I have a CPU problem.
- I have IO issues.
- I have space issues.
- I have a locking problem.
- Other handy views

The OLD Way
How to figure out an IO problem

- Get snapshot for tables
- Get snapshot for tablespaces
- Get snapshot for bufferpools
- Topas, vmstat, iostat, etc

The OLD Way
Which tables are being read the most?

- Get snapshot for tables on …..
- Get snapshot for tables on | grep –i “rows read”
- Get snapshot for tables on …. | grep –p *big-number*
- Get snapshot for tables on …| sed -e '/./{H;$!d;}' -e 'x;/AAA/!d;'

The NEW Way
MON_GET_TABLE

```
select char(tabname,30) tabname,
        table_scans,
        rows_read,
        rows_inserted,
        rows_updated,
        rows_deleted,
        overflow_accesses,
        overflow_creates,
        page_reorgs
from table ( mon_get_table ( 'SCHEMA_NAME', null , -1)) as t
order by rows_read
fetch first 20 rows only ;
```

The OLD Way
What are my bufferpool hit ratios

- db2 get snapshot for bufferpools on …. > bp.snap
- Break out a spreadsheet to perform calculations to get the bufferpool hit ratios.
- Suspect to copy/paste errors

The NEW Way MON_GET_BP

```
with bpsum as (
    select bp_name,
        pool_data_l_reads + pool_index_l_reads + pool_temp_data_l_reads + pool_temp_index_l_reads as
lreads,
        pool_data_p_reads + pool_index_p_reads + pool_temp_data_p_reads + pool_temp_index_p_reads as
preads,
        pool_temp_data_l_reads + pool_temp_index_l_reads as tlreads,
        pool_temp_data_p_reads + pool_temp_index_p_reads as tpreads
    from table( mon_get_bufferpool ( null, -1 )) as m
      )
select char(b.bp_name,20) as bpname,
    lreads,
    preads,
    ( case when preads > 0 then dec((1-(float(preads) / float(lreads))) * 100,5,2 ) else null end ) as glbl_hit_ratio,
    ( case when pool_data_p_reads > 0 then dec((1-(float(pool_data_p_reads) / float(pool_data_l_reads))) *
100,5,2 ) else null end ) as data_hit_ratio,
    ( case when pool_index_p_reads > 0 then dec((1-(float(pool_index_p_reads) / float(pool_index_l_reads))) *
100,5,2 ) else null end ) as indx_hit_ratio,
    ( case when tpreads > 0 then dec((1-(float(tpreads) / float(tlreads))) * 100,5,2 ) else null end ) as
temp_hit_ratio,
    block_ios,
```

The NEW Way
MON_GET_BP (continued)

```
( case when tpreads > 0 then dec((1-(float(tpreads) / float(tlreads))) * 100,5,2 ) else null end ) as temp_hit_ratio,
      block_ios,
      pages_from_block_ios,
      pages_from_vectored_ios,
      ( case when pages_from_block_ios > 0 then dec( (float(pages_from_block_ios) / float(block_ios)),5,2) else
null end ) as avg_block_rd_sz,
      ( case when pages_from_block_ios > 0 and pages_from_vectored_ios > 0 then ( case when
pages_from_block_ios > pages_from_vectored_ios then 'BLOCK' when pages_from_vectored_ios >
pages_from_block_ios then 'VECTOR' end ) else null end ) as V_vs_B,
      sys.npages,
      sys.blocksize,
      sys.numblockpages
from bpsum,
   table( mon_get_bufferpool ( null, -1 )) b,
   syscat.bufferpools sys
where bpsum.bp_name = b.bp_name
 and b.bp_name not like 'IBMSYS%'
 and sys.bpname = b.bp_name
```

The OLD Way
What tablespaces are being accessed most frequently?

- db2 get snapshot for tablespaces on ….

The NEW Way
MON_GET_TABLESPACE

```
select  char(tbsp_name,30) tbspace,
      tbsp_type,
      -- FS_CACHING,
      pool_data_l_reads,
      pool_data_p_reads,
      pool_async_data_reads,
      pool_index_l_reads,
      pool_index_p_reads,
      pool_async_index_reads,
      unread_prefetch_pages,
      pool_read_time,
      pool_write_time
from table( mon_get_tablespace ( null, -1 )) ts
order by pool_data_p_reads   desc
fetch first 20 rows only
;
```

The NEW Way
MON_GET_TABLESPACE (continued)

```
with ts as ( select ( SELECT LIST FROM PREVIOUS SLIDE )
                from table( mon_get_tablespace ( null, -1 )) ts
                order by pool_data_l_reads   desc
                fetch first 5 rows only )
select char(t.tabname,30) tabname,
      ts.tbspace,
      ts.pool_data_l_reads,
      ts.unread_prefetch_pages,
      ts.pool_read_time as pool_read_time_ms,
      dec( (float(ts.pool_read_time) / float(1000) ),25,3 ) as pool_read_time_sec,
      t.table_scans,
      t.rows_read
from table ( mon_get_table ( 'SAPR3', null , -1)) as t,
    ts ts,
    syscat.tables sys
where -- table_scans > 0 and
     sys.tbspace = ts.tbspace
  and sys.tabname = t.tabname
  and t.rows_read >= 10000000
 order by ts.unread_prefetch_pages desc,  t.rows_read desc ;
```

The OLD Way
New means to answer an old question

- So, now that I can see which tables are being scanned, and which SQL is doing the scanning, how do I fix the problem?
- Change the SQL or perhaps add an index.
- But my tables have “too many” indexes already. How do I know which are being used?

The NEW Way
MON_GET_INDEX

```
select char(i.tabname,30) tabname,
            char(i.indname,30) indname,
            t.index_scans,
            t.index_only_scans,
            t.key_updates,
            t.include_col_updates,
            t.page_allocations ,
            t.root_node_splits,
            t.int_node_spilts,
            t.boundary_leaf_node_splits
from table ( mon_get_index ( 'SCHEMA_NAME', null , -1)) as t,
     syscat.indexes i
where i.iid = t.iid
  and i.tabname = t.tabname
  and i.tabschema = t.tabschema
  and  key_updates > 0 and index_scans = 0 and index_only_scans = 0
order by key_updates ;
```

The NEW Way
How much space do I have in use?

```
SELECT char(current_server,20) as db_name,
        SMALLINT(TBSP_ID) as id
        ,CHAR(TBSP_NAME,20) as tbsp_name
        ,DECIMAL(ROUND(TBSP_USABLE_PAGES*TBSP_PAGE_SIZE/DECIMAL(1048576),2),9,2)
AS totalmb
        ,DECIMAL(ROUND(TBSP_FREE_PAGES *TBSP_PAGE_SIZE/DECIMAL(1048576),2),9,2) AS
freemb
        ,DECIMAL(100 *
ROUND(DECIMAL(TBSP_FREE_PAGES)/NULLIF(TBSP_USABLE_PAGES,0),4),5,2) AS pctfree
        ,DECIMAL(100 *
ROUND(DECIMAL(TBSP_USED_PAGES)/NULLIF(TBSP_USABLE_PAGES,0),4),5,2) AS pctused
        ,DECIMAL(ROUND((TBSP_PAGE_TOP - TBSP_USED_PAGES) *
TBSP_PAGE_SIZE/DECIMAL(1048576),2),9,2) AS mb_under_hwm

from table(mon_get_tablespace ( null , -1 )

WHERE TBSP_TYPE = 'DMS'
```

HOW?
- How am I going to deal this this???

- I have a CPU problem.
- I have IO issues.
- I have space issues.
- I have a locking problem.
- Other handy views

The OLD Way
get snapshot for locks for application agentid ….

- db2 list applications show detail | grep –v Wait | grep –v Connect
- Then get a snapshot for locks on an agent id that in lock wait
- Then get a snapshot for the agent that's holding the lock
- Hope you are fast enough to do all that before the locks are released!

The NEW Way
Show me lock escalations!

```
select a.AGENT_ID ,
        L.LOCK_ESCALATION,
        L.TABNAME,
        chr(10) || substr(STMT_TEXT,1,400) || chr(10)  as txt
from table ( SNAP_GET_APPL_INFO ( current server , -1)) a ,
       table ( SNAP_GET_STMT ( current server , -1)) x ,
       table ( SNAP_GET_LOCK ( current server , -1)) L
where x.agent_id = a.agent_id
   and L.agent_id = a.agent_id
   and a.appl_status = 'UOWEXEC'
   and LOCK_ESCALATION <> 0
;
```

The NEW Way
Who's holding the most locks?

```
select  y.agent_id,
                char(y.tabname,30) as tabname
                y.lock_mode,
                y.lock_status,
                y.lock_count,
                y.lock_current_mode,
                y.lock_hold_count
                x.STMT_OPERATION,
               x.QUERY_COST_ESTIMATE,
               x.STMT_START,
                rtrim(char(STMT_USR_CPU_TIME_S)) || '.' || char(STMT_USR_CPU_TIME_MS),
                chr(10) || substr(stmt_text,1,100) || chr(10) as stmt_text
from table (SNAP_GET_STMT ( current server, -1 )) x
      ,table (SNAP_GET_LOCK ( current server, -1 )) y
where lock_mode not in ( 'IX', 'IS', 'IN' )
 and LOCK_OBJECT_TYPE not like 'INTERN%'
 and x.agent_id = y.agent_id ;
```

The NEW Way
OK, but what's really the problem transaction?

- See the notes on this slide for the SQL.

AGENT_ID	HOLDING_AGENT	TABNAME	APPL_STATUS	TYPE
19647	0	TABLE_AA	UOWEXEC	HOLDER
56502	0	TABLE_AA	UOWEXEC	HOLDER
25900	0	TABLE_AA	UOWEXEC	HOLDER
29816	0	TABLE_AA	UOWWAIT	HOLDER
5816	0	TABLE_AA	UOWEXEC	HOLDER
18910	56502	TABLE_AA	LOCKWAIT	WAITER
58448	1564	TABLE_AA	LOCKWAIT	WAITER
17537	24064	TABLE_AA	LOCKWAIT	WAITER
10827	61877	TABLE_AA	LOCKWAIT	WAITER
42177	29816	TABLE_AA	LOCKWAIT	WAITER
55614	5816	TABLE_AA	LOCKWAIT	WAITER
61181	19647	TABLE_AA	LOCKWAIT	WAITER

12 record(s) selected.

```
with H ( agent_id,
            agent_id_holding_lk,
            tabname,
            appl_status,
            locks_held,
            SQL_REQS_SINCE_COMMIT,
            UOW_START_TIME,
            UOW_TIME ,
            lock_wait_UOW_START_TIME,
            snapshot_timestamp,
            Type ) as (
 select lw.agent_id,
       lw.agent_id_holding_lk ,
       substr(lw.tabname,1,30) tabname,
       ai.appl_status ,
       a.locks_held ,
       a.SQL_REQS_SINCE_COMMIT ,
       a.UOW_START_TIME ,
       rtrim(char(UOW_ELAPSED_TIME_S)) || '.' ||
char(UOW_ELAPSED_TIME_MS) ,
```

HOW?
- How am I going to deal this this???

- I have a CPU problem.
- I have IO issues.
- I have space issues.
- I have a locking problem.
- Other handy views.

The NEW Way
MON_FCM

- Easy to use in a DPF environment and useful when using intra-parallelism.

```
select  member,
      buff_total,
      buff_free,
      buff_free_bottom
from table( MON_GET_FCM ( -2 )) x
order by member ;
```

- The use of -2 in this example gathers data from all database partitions.

The NEW Way
REORG Monitoring

```
select char(tabname,40) as tabname,
                reorg_status,
                reorg_start,
                reorg_end,
                reorg_current_counter,
                reorg_max_counter
from table(snap_get_tab_reorg ( current server , -1 )) x
where date(reorg_start) >= ( current date - 1 day )
order by 4, 3, 2 ;
```

The NEW Way
MON_GET_TRANSACTION_LOG

```
Select member,
        TOTAL_LOG_AVAILABLE,
        TOTAL_LOG_USED,
        LOG_READS,
        LOG_READTIME,
        LOG_WRITES,
        LOG_WRITE_TIME,
        NUM_LOG_BUFFER_FULL,
        CUR_COMMIT_LOG_BUFF_READS
        CUR_COMMIT_DISK_LOG_READS,
        CUR_COMMIT_TOTAL_LOG_READS
From table( mon_get_transaction_log ( -1 ) ;
```

The NEW Way
MON_GET_ROUTINE

```
Select char(routine_name,30) as routine,
        total_times_routine_invoked,
        total_routine_corrd_time,
        total_sorts,
        total_CPU_time,
        total_WAIT_time
From table( mon_get_routine ( 'P', 'YOUR_SCHEMA', null, null, -1 ) ;
```

- DB2 10.1 function.

The NEW Way
HADR Cluster Status

```
Select HADR_ROLE,
      HADR_SYSCMODE,
      STANDBY_ID,
      HADR_STATE,
      HADR_CONNECT_STATUS,
      HADR_CONNECT_STATUS_TIME,
      PRIMARY_LOG_FILE,
      PRIMARY_LOG_PAGE,
      PRIMARY_LOG_POS,
      STANDBY_LOG_FILE,
      STANDBY_LOG_PAGE,
      STANDBY_LOG_POS,
      LOG_HADR_WAIT_TIME,
      LOG_HADR_WAITS_TOTAL
From MON_GET_HADR( -2 ) ;
```

Leo Pedron
Fourth Millennium Technologies
leo@fmtusa.com

Session Number: D12

Table Monitors
Who, What, When, Where, Why, How

INTERNATIONAL DB2 USERS GROUP
Independent • Not-for-Profit • User Run
Founded 1988