

DB2® Performance Monitoring Essentials

Version/Revision# 1

Module ID 10302

Length 1.5 hours



Table of Contents

1	Introduction	3
2	Suggested Reading	3
3	Lab Preparation	4
3.1	Start Virtual Machine and DB2	4
3.2	Initialize Environment	6
4	Configuring Performance Monitoring	7
4.1	DB Configurations for Monitoring Framework	7
4.2	System Monitor Switches	9
5	Interfaces for database monitoring	11
5.1	CREATE EVENT MONITOR (change history) statement	11
5.2	Identifying the statements that affect a table	14
5.3	Creating event monitors that write to tables	16
5.4	Displaying a list of event monitors created in your database	19
5.5	Event Monitor Data Retention from Release to Release	20
6	Cleanup	23

1 Introduction

DB2's performance monitoring framework has been improved in DB2 10.5. Historically DB2's monitoring utilities and interfaces have been based on the monitoring elements presented by the Snapshot Monitoring stream. DB2's Health monitor, snapshot commands, administrative views and table functions all utilize the Snapshot Monitoring stream of monitoring elements.

DB2 V10 brings along with it a new and full complement of monitoring table functions and administrative views that enable you to take advantage of the monitoring framework. The monitoring framework is robust, light weight and features many identical or similar monitoring elements as the Snapshot monitoring.

This lab is intended to teach you the essentials to monitoring a DB2 database system. The scope of a "DB2 database system" extends capabilities through the Enterprise Edition, and up to and including version 10.5. This module is intended for someone who has some experience with DB2 and wishes to broaden their knowledge in the area of performance-related monitoring.

These labs are performed on a VMware image of a 64bit SUSE Linux machine with minimal resources declared. There are some commands used in the lab that are typically found on Linux/Unix systems, however basic Unix command-line tools can be downloaded and easily employed on Windows-based machines at no cost. Additionally, comparable commands are available in the Windows environment, like find in place of grep.

There are two main tips that will make the lab progress better.

1. Use the up arrow key to recall previous commands. These commands can be edited and re-executed as needed.
2. Follow all the instructions, don't skip steps. These labs have been created so that they can be rerun but to avoid that please follow all instructions.

2 Suggested Reading

Information Management Technical Library at developerWorks

Contains articles and tutorials and best practices

<http://www.ibm.com/developerworks/views/db2/library.jsp>

<http://www.ibm.com/developerworks/data/bestpractices>

An Expert's Guide to DB2 Technology

Great read and useful information.

<http://blogs.ittoolbox.com/database/technology>

DB2 10 Fundamentals Certification Study Guide

Learn the basics and get ready for certification

3 Lab Preparation

Before getting started with the lab exercises you will set up your environment then work through a series of exercises focused on DB2 performance monitoring techniques.

3.1 Start Virtual Machine and DB2

You may skip this section if you have started your VM and db2 instance in a previous lab or exercise.

1. Choose the **DB2_BLU-AWSE_10.5....vmx** file when first opening the lab image in the **/DB2_BLU_PerfMonitoring...** folder and run it.

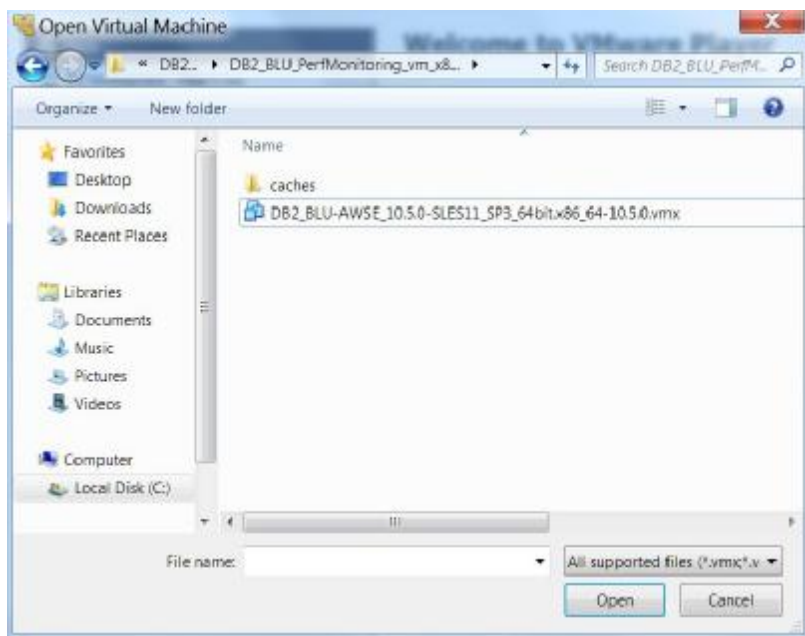


Figure 1 – Opening a VMware Image File

2. If this is the first time this virtual machine has been started, a series of licensing agreements will prompt you. To proceed, acknowledge by responding by selecting the “yes, I Agree to the License Agreement” selection.



Figure 2 – License Agreements

3. Login in at the console prompt as **db2inst1**. You will be prompted for the password which is **password**. You may be prompted for the root **password** as well, if so, it is also password.

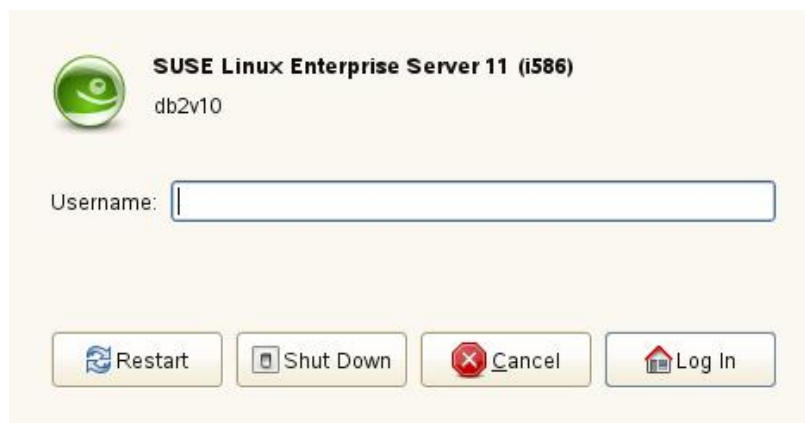


Figure 3 – Login credentials

4. Open a terminal window as follows:
 - Open a new terminal window by right-clicking on the **Desktop** and choosing the “**Open Terminal**” item:

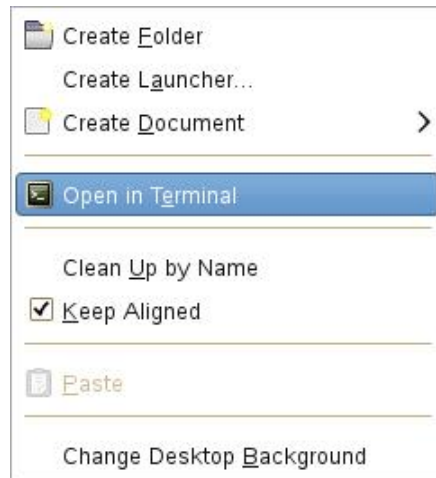


Figure 4 – Opening a Terminal

From the new terminal session start the Database Manager as follows:

- Enter the command “**db2start**” to start the database instance. Make sure the instance is started successfully. (**Note:** If you notice a message that says “The database manager is already active”, that’s “OK. Please continue to the next step.)
- In order to enable the aliases defined for this lab, copy the /home/db2inst1/labbashrc to /home/db2inst1/.bashrc and source that file.

```
cd ~  
cp labbashrc .bashrc  
. .bashrc
```

3.2 Initialize Environment

You have been provided with a script, which creates a database, the needed tablespaces and the tables that you will work with in this portion of the lab. **These are mandatory steps and must be completed, as mandatory steps so often are.**

1. If you no longer have the terminal opened from previous exercise: By right clicking on the Gnome desktop (as in 3.1) **open a new terminal window.**
2. While logged in as user “**db2inst1**”, change your working directory to the folder containing the script mentioned.

```
edb2scripts
```

The command will take you to directory /home/db2inst1/Documents/LabScripts/db2pt/essential/db2scripts.

3. Before executing, take a look at the script, and note how the database is created. It will first try to drop the db2pt database, in case one exists from previous labs, and then create a new version of the database.

```
cat createDB.db2
```

Note: You can use the arrow keys to scroll up and down in the script and type the letter “q” to exit anytime.

4. Execute the script as follows:

```
db2 -tvf createDB.db2
```

Note: Make sure that the create database SQL statement has finished successfully. The following message should be returned after the successful completion of this SQL statement:

```
DB20000I The CREATE DATABASE command completed successfully.
```

5. Turn off system monitor switches and restart the instance.

```
db2 -tvf dftmonOff.db2
db2stop force
db2start
db2 terminate
```

4 Configuring Performance Monitoring

Before you start performance monitoring, you want to make sure that your system and your session are configured to collect the information that you need. First you will examine and practice with the database configuration parameters that control the collection of data for the monitoring table functions. Then you will look at and enable the system monitor switches used to collect data for Snapshot monitoring.

4.1 DB Configurations for Monitoring Framework

DB2® Version 10 provides new monitor elements that enable you to perform more granular monitoring, without using the monitor switches or snapshot interfaces. Database-wide monitoring control is provided by database configuration parameters.

With the new monitor elements and infrastructure, you can use SQL statements to efficiently collect monitor data from memory to determine whether specific aspects of the system are working correctly and to help you diagnose performance problems. With the new access methods, you can get the data you need without using the snapshot interfaces. The increased monitoring granularity gives you more control over the data collection process; collect the data you want from the source you want. The monitoring framework also reduces performance monitoring overhead to a minimum. **Though the monitoring framework was designed to work with the Performance Optimization feature pack, it can still be used with the default workloads that are predefined for each database created.**

Monitoring information is collected about the work performed by your applications and reported through table functions and administrative views at the following three levels:

System level

These monitoring elements provide details about all work being performed on the system. Monitor-element access points include service subclass, workload definition, unit of work, and connection.

Activity level

These monitor elements provide details about activities being performed on the system (a specific subset of the work being performed on the system). You can use these elements to understand the behaviour and performance of activities. Monitor-element access points include individual activities, and entries in the database package cache.

Data object level

These monitoring elements provide details about the work being processed by the database system within specific database objects such as indexes, tables, buffer pools, table spaces, and containers, thereby enabling you to quickly identify issues with particular data objects that might be causing system problems. Monitor-element access points include buffer pool, container, index, table, and table space.

For database-wide control over the collection of monitoring data at the System, Activity and Data Object Levels, and the generation of events in the new event monitors, ten database configuration parameters have been added. With the workload management feature enabled, these controls can be enforced at both the database and at activity levels defined in workloads.

In this short exercise you will learn how to look at a database configuration file, examine the configurations for the monitoring framework, and then actually make a few updates to the lock monitoring configurations.

1. If you no longer have the terminal opened from the previous exercise, right click on the Gnome desktop (as in 3.1) and **open a new terminal window**.
2. Enter the command to get and display the database configuration file. The new configurations will appear at the bottom of the file's listing under Monitor Collect Settings.

```
db2start
db2 get db cfg for db2pt | grep -i MON
```



```
File Edit View Terminal Help
SQL1028N The database manager is already active.
db2inst1@db2v10:~> db2 get db cfg for db2pt | grep -i MON
Monitor Collect Settings
Request metrics              (MON_REQ_METRICS) = BASE
Activity metrics             (MON_ACT_METRICS) = BASE
Object metrics               (MON_OBJ_METRICS) = EXTENDED
Unit of work events          (MON_UOW_DATA) = NONE
UOW events with package list (MON_UOW_PKGLIST) = OFF
UOW events with executable list (MON_UOW_EXECLIST) = OFF
Lock timeout events          (MON_LOCKTIMEOUT) = NONE
Deadlock events              (MON_DEADLOCK) = WITHOUT_HIST
Lock wait events             (MON_LOCKWAIT) = NONE
Lock wait event threshold    (MON_LW_THRESH) = 5000000
Number of package list entries (MON_PKGLIST_SZ) = 32
Lock event notification level (MON_LCK_MSG_LVL) = 1
db2inst1@db2v10:~>
```

Figure 5 – Getting the Monitor Parameters

The request (REQ), activity (ACT) and Object Metrics (OBJ) are all set on (BASE) by default, which means all metrics are collected for all requests executed on the data server.

3. Next you are going to turn on maximum collection for lock wait and deadlock monitoring and set the threshold for lockwaits to 3 seconds.

Lab Tip: Use the up arrow on your keyboard to recall commands.

```
db2 update db cfg for db2pt using mon_lockwait hist_and_values
db2 update db cfg for db2pt using mon_lw_thresh 3000000
db2 update db cfg for db2pt using mon_deadlock hist_and_values
```

4. Show the new database configurations You may want to “get db cfg show detail” and explain:


```
db2 get db cfg for db2pt
```

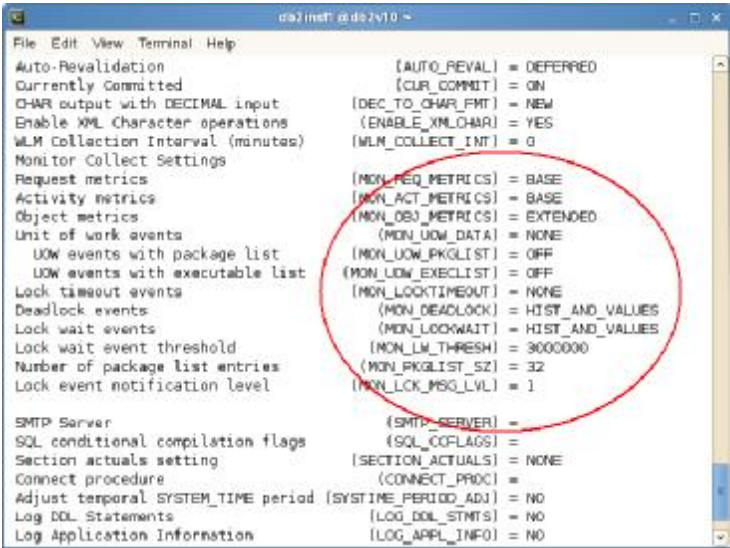


Figure 6 – Getting the Values Parameters

4.2 System Monitor Switches

If you use the Snapshot Monitor or any related interfaces you should make sure that your system monitor switches are set so you can collect the monitoring data you are after. The system monitor switches can be turned on at the instance level by updating the database manager configurations. They can also be controlled at the session level. Each monitoring application has its own logical view of the monitor switches (and the system monitor data). Upon startup, each application inherits its monitor switch settings from the parameters in the database manager configuration file (at the instance level). A monitoring application can alter its monitor switch settings with the UPDATE MONITOR SWITCHES USING MONSWITCH OFF/ON command. The MONSWITCH parameter holds values found in the Monitor Switch column in the Snapshot Monitor Switches table below. Changes to the switch settings at the application level affect only the application from which the switch was changed.

Table 1. Snapshot Monitor Switches

Monitor Switch	DBM Parameter	Information Provided
BUFFERPOOL	DFT_MON_BUFPOOL	Number of reads and writes, time taken
LOCK	DFT_MON_LOCK	Lock wait times, deadlocks
SORT	DFT_MON_SORT	Number of heaps used, sort performance
STATEMENT	DFT_MON_STMT	Start/stop time, statement identification
TABLE	DFT_MON_TABLE	Measure of activity(rows read/written)

UOW	DFT_MON_UOW	Start/end times, completion status
TIMESTAMP	DFT_MON_TIMESTAMP	Timestamps

This short exercise will teach you how to set and reset these controls.

1. By right clicking on the Gnome desktop (as in 3.1) **open a new terminal window** or you may use the one from 4.1 if still open.
2. Connect to the database and list the system monitor switches from the database manager configuration file. You will find most switches are set to OFF.

```
db2 connect to db2pt
db2 get dbm cfg | grep -i DFT_MON
```

3. Set monitor switches to ON and restart instance. Check new values.

Lab Tip: Use the up arrow on your keyboard to recall commands.

```
db2 update dbm cfg using dft_mon_bufpool on
db2 update dbm cfg using dft_mon_lock on
db2 update dbm cfg using dft_mon_stmt on
db2 update dbm cfg using dft_mon_uow on
db2 update dbm cfg using dft_mon_sort on
db2 update dbm cfg using dft_mon_table on
db2 update dbm cfg using dft_mon_timestamp on
db2stop force
db2start
db2 get dbm cfg | grep -i DFT_MON
```

The monitor switches should be on now.



Figure 7 – Getting the Values Parameters

4. Now you will turn OFF the switch for bufferpools at the **session level**. It should currently be set to ON (session level), adopted from the DFT_MON_BUFPOOL value in the db manager configuration file. First activate the database and take a snapshot of the database, then turn the switch ON and look again to see that bufferpool metrics are being collected for your session.

```
db2 get monitor switches
```

Notice that bufferpool is ON

```
db2 update monitor switches using bufferpool off
```

Set switch at session level

```
db2 get monitor switches
```

Notice that bufferpool is OFF

```
db2 activate database db2pt
```

Database **must be active** for monitoring

```
db2 get snapshot for bufferpools on db2pt
```

Notice how most metrics come up as Not Collected Database

```
db2 update monitor switches using bufferpool on
```

Updating session

```
db2 get monitor switches
```

Notice that the bufferpool is ON

```
db2 get snapshot for bufferpools on db2pt
```

Counters now contain numeric values, with probably a lot of zeros at this point, since there is little activity on the system. This indicates that collection for these metrics is now active

5 Interfaces for database monitoring

i **Note:** Your results may be a little different.

There are two ways to monitor operations in your database. You can view information that shows the state of various aspects of the database at a specific point in time. Or, you can set up event monitors to capture historical information as specific types of database events take place.

You can monitor your database operations in real-time using monitoring table functions. For example, you can use a monitoring table function to examine the total amount of space used in a table space. These table functions let you examine *monitor elements* and metrics that report on virtually all aspects of database operations using SQL.

5.1 CREATE EVENT MONITOR (change history) statement

Generally, the process of creating and using event monitors to capture information about the system when certain events occur is similar for all event monitor types. First you create the event monitor, then you enable data collection, and finally, you access the data gathered.

1. Create the event monitor. To create an event monitor, use the appropriate version of the CREATE EVENT MONITOR statement. When you create an event monitor, you must choose how to record the data that the event monitor collects. All event monitors can write their output to relational tables; however, depending on your specific purposes, there are different options that might be more appropriate.

```
db2 connect to db2pt
db2 "CREATE EVENT MONITOR EM_CHANGE_HIST
FOR CHANGE HISTORY WHERE EVENT IN (DBMCFG, DDLALL, DDLDATA, DDLSQL)
WRITE TO TABLE
CHANGESUMMARY (TABLE CHG_SUMMARY_HISTORY),
```

```
DDLSTMTEEXEC (TABLE DDLSTM_HISTORY)
AUTOSTART "
```

The previous statement creates the following tables:

```
db2 "describe table chg_summary_history"
db2 "describe table ddlstm_history"
```

Changes the following parameters again

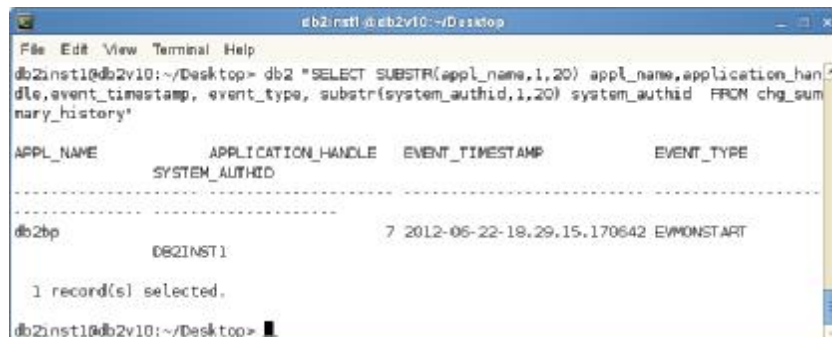
```
db2 update dbm cfg using dft_mon_bufpool off
db2 update dbm cfg using dft_mon_lock off
db2 update dbm cfg using dft_mon_stmt off
```



```
db2inst1@db2v10: ~/Desktop
File Edit View Terminal Help
db2inst1@db2v10:~/Desktop> db2 update dbm cfg using dft_mon_bufpool off
DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.
db2inst1@db2v10:~/Desktop> db2 update dbm cfg using dft_mon_lock off
DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.
db2inst1@db2v10:~/Desktop> db2 update dbm cfg using dft_mon_stmt off
DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.
db2inst1@db2v10:~/Desktop>
```

Figure 8 – Change of parameters

```
db2stop force
db2start
db2 connect to db2pt
db2 "SELECT SUBSTR(appl_name,1,20)
appl_name,application_handle,event_timestamp, event_type,
substr(system_authid,1,20) system_authid FROM chg_summary_history"
```



```
db2inst1@db2v10:~/Desktop
File Edit View Terminal Help
db2inst1@db2v10:~/Desktop> db2 "SELECT SUBSTR(appl_name,1,20) appl_name,application_han
dle,event_timestamp, event_type, substr(system_authid,1,20) system_authid FROM chg_sum
mary_history"
APPL_NAME          APPLICATION_HANDLE  EVENT_TIMESTAMP    EVENT_TYPE
-----
db2pt              DB2INST1              7 2012-06-22-18.29.15.170642 EVMONSTART
1 record(s) selected.
db2inst1@db2v10:~/Desktop>
```

Figure 9 – Show the change history

Changes the following parameters

```
db2 update dbm cfg using dft_mon_bufpool on
db2 update dbm cfg using dft_mon_lock on
```

```
db2 update dbm cfg using dft_mon_stmt on
db2stop force
db2start
db2 connect to db2pt
db2 "SELECT SUBSTR(appl_name,1,20)
appl_name,application_handle,event_timestamp, event_type,
SUBSTR(system_authid,1,20) system_authid FROM chg_summary_history"
```

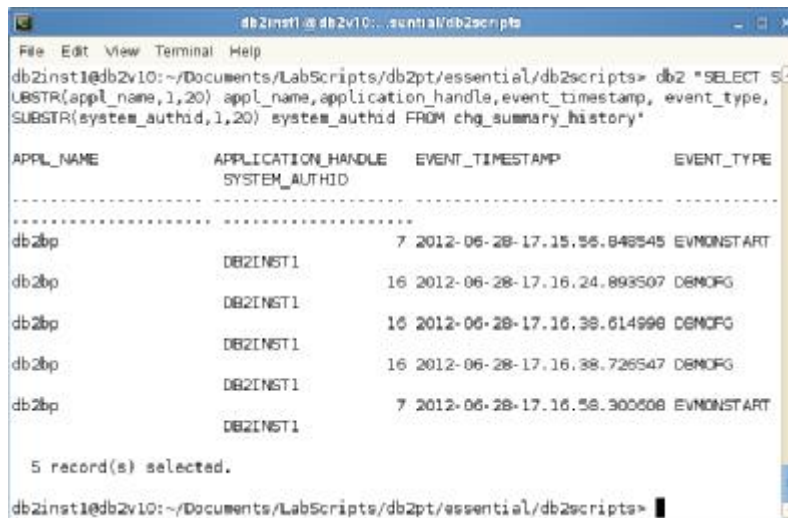


Figure 10 – Show the change history

Execute the follow script to generate the next objects:

- Table DOCUMENTS
- View V_DOCUMENTS
- Type name IMAGE.

```
edb2scripts
db2 -tvf create_objects_evm.sql
```



Figure 11 – Creation of Objects using DDL

```
db2 "SELECT event_id, event_timestamp, SUBSTR(event_type,1,20) event_type,
ddl_classification, SUBSTR(local_transaction_id,1,20) local_transaction_id,
SUBSTR(stmt_text,1,50) stmt_text
FROM ddlstm_history"
```



Figure 12 – Show the change history for Event Type DDLSTMEEXEC

5.2 Identifying the statements that affect a table

Use usage lists to identify DML statement sections that affect a particular table when the statement sections execute. You can view statistics for each statement and use these statistics to determine where additional monitoring or tuning might be required.

Do the following tasks:

1. Identify a table for which you want to view object usage statistics. You can use the MON_GET_TABLE table function to view monitor metrics for one or more tables.

```
db2 update DATABASE CONFIGURATION using MON_OBJ_METRICS EXTENDED
```

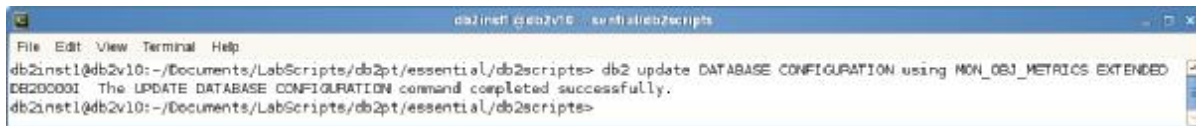


Figure 2 – Set the mon_obj_metrics configuration parameter

2. Create an usage list for the table by using the CREATE USAGE LIST statement

```
db2 "CREATE USAGE LIST DOCUMENTSUL FOR TABLE DOCUMENTS"
```

3. Activate the collection of object usage statistics by using the SET USAGE LIST STATE statement

```
db2 "SET USAGE LIST DOCUMENTSUL STATE = ACTIVE"
```

4. During the collection of object statistics, ensure that the usage list is active and that sufficient memory is allocated for the usage list by using the MON_GET_USAGE_LIST_STATUS table function. To check the status of the DOCUMENTSUL usage list, issue the following command:

```
db2 "SELECT MEMBER, STATE, LIST_SIZE, USED_ENTRIES, WRAPPED
FROM TABLE(MON_GET_USAGE_LIST_STATUS('DOCUMENTS', 'DOCUMENTSUL', -2))"
```

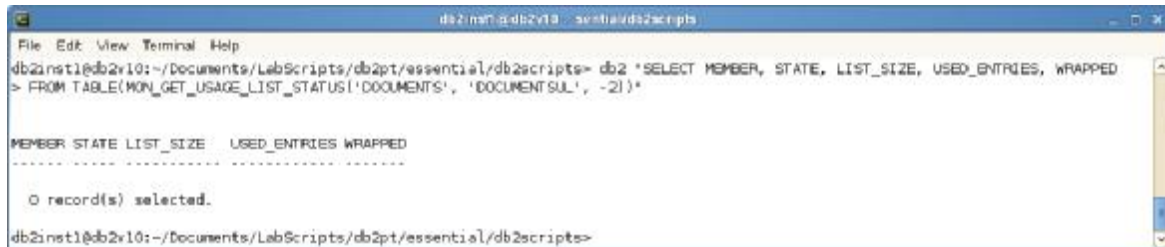



Figure 3 – Status of the Documentsul usage list

5. Execute the follow script to create data

```
db2 -tvf create_data.db2
```

6. When the time period for which you want to collect object usage statistics is elapsed, deactivate the collection of usage list data by using the `SET USAGE LIST STATE` statement.

```
db2 "SET USAGE LIST DOCUMENTSUL STATE = INACTIVE"
```

7. View the information that you collected by using the `MON_GET_TABLE_USAGE_LIST` function. You can view statistics for a subset or for all of the statements that affected the table during the time period for which you collected statistics

```
db2 "SELECT MEMBER, EXECUTABLE_ID, NUM_REFERENCES, NUM_REF_WITH_METRICS,  
ROWS_READ, ROWS_INSERTED, ROWS_UPDATED, ROWS_DELETED  
FROM TABLE(MON_GET_TABLE_USAGE_LIST('DB2INST1', 'DOCUMENTSUL', -2))  
ORDER BY ROWS_READ DESC  
FETCH FIRST 10 ROWS ONLY"
```

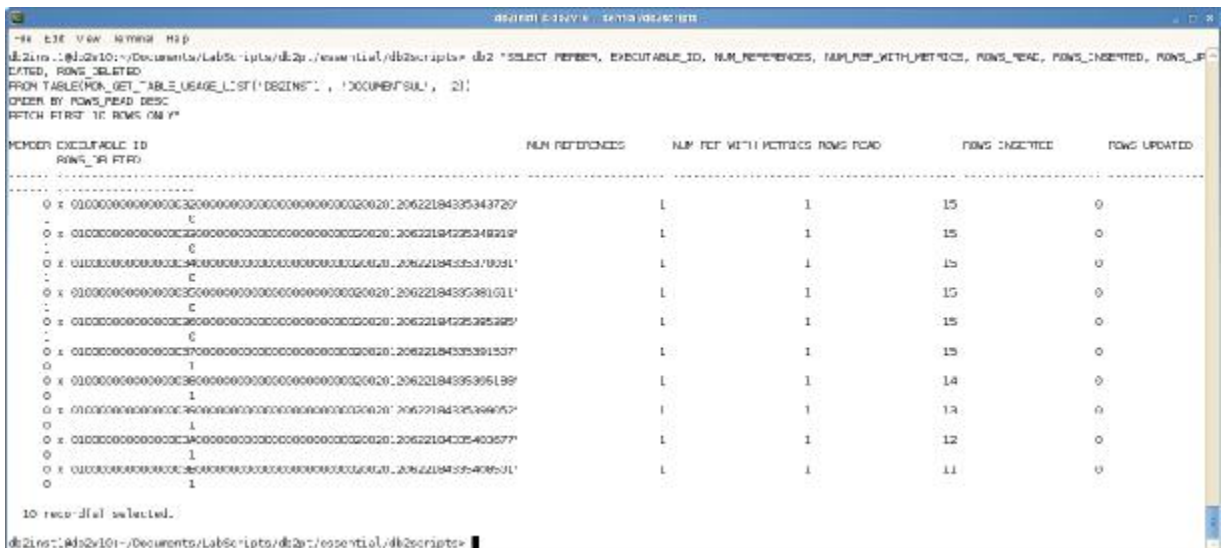


Figure 4 – Information collected by using MON GET TABLE USAGE LIST

8. If you want to view the text of a statement that affected the table, **use the value of the executable_id** element in the MON GET TABLE USAGE LIST output as input for the MON GET PKG CACHE STMT table function

[illegible]

Figure 5 – Text of statement that affected the table

9. Drop the objects generated for this lab.

```
db2 -tvf drop_objects_evm.sql
```

5.3 Creating event monitors that write to tables

There are different forms of this statement that you use, depending on the type of events that you intend to monitor. The table to which the event monitor writes its output must be a non-partitioned table.

The various options for table event monitors are set in the CREATE EVENT MONITOR statement. For further assistance in generating CREATE EVENT MONITOR SQL statements for write-to-table event monitors, you can use the db2evtbl command. Simply provide the name of the event monitor and the required event type (or types), and the CREATE EVENT MONITOR statement is generated, complete with listings of all the target tables. You can then copy the generated statement, make modifications, and then execute the statement from the command line processor.

Formulate a CREATE EVENT MONITOR statement using the WRITE TO TABLE clause to indicate that event monitor data is to be collected in a table (or set of tables).

The event type is one of the following values:

- ACTIVITIES
- BUFFERPOOLS
- CHANGE HISTORY
- CONNECTIONS
- DATABASE
- DEADLOCKS
- LOCKING
- PACKAGE CACHE
- STATEMENTS
- STATISTICS
- TABLE
- TABLESPACE
- THRESHOLD VIOLATIONS
- TRANSACTIONS
- UNIT OF WORK

1. To create a unit of work event monitor called myevmon, use a statement like the one that follows

```
db2 "CREATE EVENT MONITOR myevmon FOR UNIT OF WORK WRITE TO TABLE "
```

The preceding statement creates a unit of work event monitor that uses defaults for the logical groups of monitor elements collected, the corresponding output table names, and the target table spaces for the table.

```
db2 "CREATE EVENT MONITOR evm_test FOR STATEMENTS WRITE TO TABLE "
```

Event monitors using the STATEMENTS event type collect data from the event_connheader, event_stmt, and event_subsection logical data groups. Tables representing logical data groups that are specific to individual event types are created, along with a control table for every write-to-table event monitor. For the event monitor test, created by user db2inst1, the database manager creates the following tables:

- connheader_evm_test
- stmt_test
- subsection_test
- control_evm_test

```
db2 "describe table connheader_evm_test"  
db2 "describe table stmt_evm_test"  
db2 "describe table control_evm_test"  
db2 "set event monitor evm_test state 1"
```

2. Optional: Specify the logical groups for which you want data collected. By default, event data is collected for all logical data groups for the event monitor type. (See Target tables, control tables, and event monitor table management for details.)

If you want only data for selected logical groups collected, you can specify the names of the logical groups to include in the CREATE EVENT MONITOR statement. For example, with a locking event monitor, you might want to collect only the information associated with the LOCK and PARTICIPANT logical groups. To include only these logical groups, you could use a statement like the one that follows:

```
db2 "CREATE EVENT MONITOR mylocks FOR LOCKING WRITE TO TABLE  
LOCK,LOCK_PARTICIPANTS "
```

3. Execute the next scripts to see the event monitor statement

```
db2 -tvf create_objects_evm.sql  
db2 -tvf create_data.db2
```

```
File Edit View Terminal Help
db2inst1@db2v10:~/Documents/LabScripts/ch2pt/essential/db2scripts> db2 -tvf create_data.db2
INSERT INTO documents (summary, report) VALUES ('10','1')
00200001 The SQL command completed successfully.

INSERT INTO documents (summary, report) VALUES ('20','2')
00200001 The SQL command completed successfully.

INSERT INTO documents (summary, report) VALUES ('30','3')
00200001 The SQL command completed successfully.

INSERT INTO documents (summary, report) VALUES ('40','4')
00200001 The SQL command completed successfully.

INSERT INTO documents (summary, report) VALUES ('50','5')
00200001 The SQL command completed successfully.

INSERT INTO documents (summary, report) VALUES ('60','6')
00200001 The SQL command completed successfully.

INSERT INTO documents (summary, report) VALUES ('70','7')
00200001 The SQL command completed successfully.

INSERT INTO documents (summary, report) VALUES ('80','8')
```

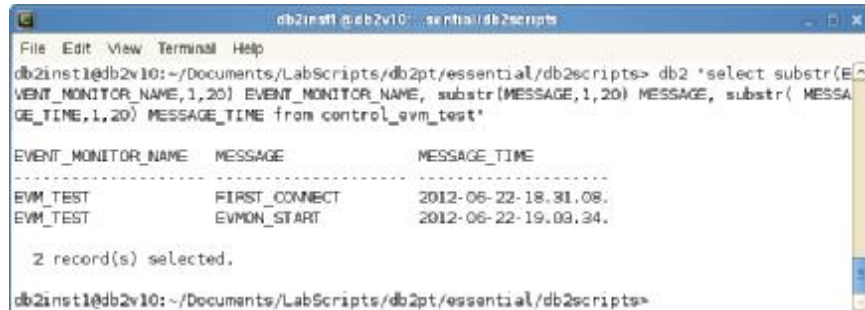
Figure 6 – Creation of Objects and generation of data

```
db2 "select * from connheader_evm_test"
```

[illegible]

Figure 7 – Logical Data Grouping connheader

```
db2 "select substr(EVENT_MONITOR_NAME,1,20) EVENT_MONITOR_NAME,  
substr(MESSAGE,1,20) MESSAGE, substr( MESSAGE_TIME,1,20) MESSAGE_TIME from  
control evm test"
```



```

db2inst1@db2v10: ~/Documents/LabScripts/db2pt/essential/db2scripts
File Edit View Terminal Help
db2inst1@db2v10:~/Documents/LabScripts/db2pt/essential/db2scripts> db2 "select substr(EVENT_MONITOR_NAME,1,20) EVENT_MONITOR_NAME, substr(MESSAGE,1,20) MESSAGE, substr(MESSAGE_TIME,1,20) MESSAGE_TIME from control_ewm_test"

EVENT_MONITOR_NAME  MESSAGE                                MESSAGE_TIME
-----
EVM_TEST            FIRST_CONNECT                         2012-06-22-18.31.08.
EVM_TEST            EVMON_START                           2012-06-22-19.03.34.

2 record(s) selected.
db2inst1@db2v10:~/Documents/LabScripts/db2pt/essential/db2scripts>

```

Figure 8 – Logical Data Grouping stmt

```
db2 -tvf drop_objects_ewm.sql
```

5.4 Displaying a list of event monitors created in your database

You can see what event monitors are already defined in your database. To view a list of the event monitors that you defined on your system, query the catalog view SYSCAT.EVENTMONITORS.

```
db2 "SELECT SUBSTR(EVMONNAME,1,20) AS EVMON_NAME, TARGET_TYPE,
SUBSTR(OWNER,1,20) OWNER FROM SYSCAT.EVENTMONITORS"
```



```

db2inst1@db2v10: ~/Documents/LabScripts/db2pt/essential/db2scripts
File Edit View Terminal Help
db2inst1@db2v10:~/Documents/LabScripts/db2pt/essential/db2scripts> db2 "SELECT SUBSTR(EVMONNAME,1,20) AS EVMON_NAME, TARGET_TYPE, SUBSTR(OWNER,1,20) OWNER FROM SYSCAT.EVENTMONITORS"

EVMON_NAME          TARGET_TYPE  OWNER
-----
DB2DETAILDEADLOCK   F           DB2INST1
EM_CHANGE_HIST      T           DB2INST1
MYEVMON             T           DB2INST1
EVM_TEST            T           DB2INST1
MYLOCKS             T           DB2INST1

5 record(s) selected.
db2inst1@db2v10:~/Documents/LabScripts/db2pt/essential/db2scripts>

```

Figure 20 – Event monitors defined in the database

You can also use a catalog view to see which event monitors exist for monitoring a specific type of event. The SYSCAT.EVENTS view returns a list of event monitors and the type of events for which they record data.

```
db2 "SELECT SUBSTR(TYPE,1,20) AS EVENT_TYPE, SUBSTR(EVMONNAME,1,20) AS
EVENT_MONITOR_NAME FROM SYSCAT.EVENTS ORDER BY TYPE"
```



Figure 29 – Event monitor for a specific type of event

5.5 Event Monitor Data Retention from Release to Release

You can upgrade event monitor output tables after you upgrade the DB2 product. This capability lets you retain any data that might exist in event monitor tables that you had before you upgraded.

As event monitors are enhanced in the DB2 product, the tables they produce might change. For example, new columns might be added to a table for reporting new monitor elements. Before Version 10, if you had existing event monitors that wrote to tables that contained data that you wanted to retain, and you wanted to collect the data in the newly-added columns, you were required to manually alter them after upgrading to the new release. This alteration involved adding any of the new columns that you might want to use. If you did not add the new columns, the event monitor would work as it had in the previous release, capturing only the data supported by that the event monitor in that release.

Unformatted event tables that had changed could not be upgraded at all; you were required to drop them and then re-create them.

The EVMON_UPGRADE_TABLES stored procedure upgrades the definitions of existing event monitor tables to match those produced by the current level of the DB2 product. This feature lets you keep any existing tables that you might have, along with all the data they contain, eliminating the need to manually alter, or to drop and re-create tables.

Note: Starting in Version 10, you can also use the ALTER EVENT MONITOR statement to add new logical groups to an event monitor. You can use this approach as an alternative to EVMON_UPGRADE_TABLES to add logical data groups added in a new release. However, you cannot use ALTER EVENT MONITOR to modify logical groups that are already associated with the event monitor; if a logical data group already associated with the event monitor has changed, the only way to modify the event monitor is using the EVMON_UPGRADE_TABLES procedure.

The EVMON_UPGRADE_TABLES procedure works with both regular and UE tables. For regular tables, the procedure adds any new columns needed, drops old columns that are no longer required, and alters any columns as needed. For UE tables, the procedure adds new columns and modifies existing columns as needed to allow the UE table to be processed by the db2evmonfmt tool, or the EVMON_FORMAT_UE_TO_TABLES or EVMON_FORMAT_UE_TO_XML routines.

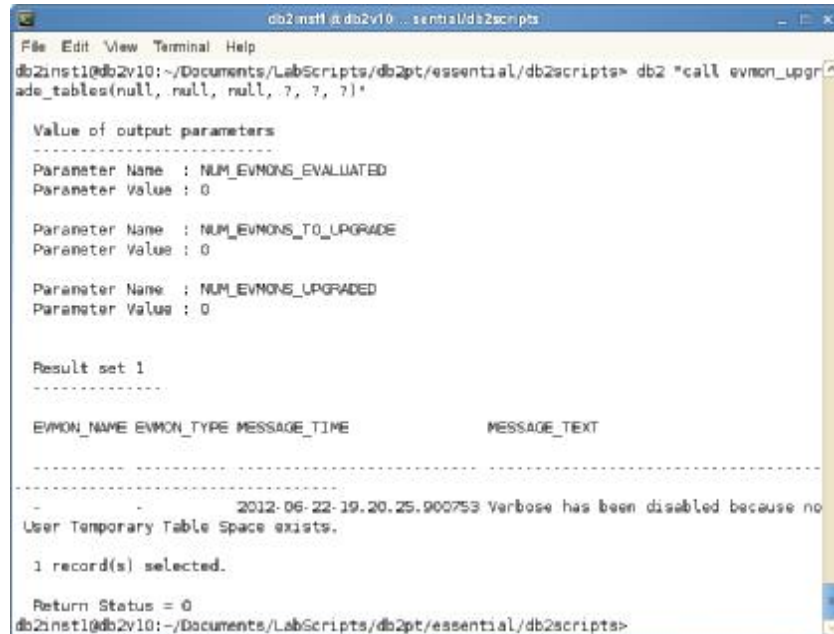
For example, a user created the following event monitors in DB2:

```
db2 "create event monitor lock for locking write to unformatted event table"
db2 "create event monitor act for activities write to table control"
db2 "create event monitor stat for statistics write to table"
db2 "create event monitor conn for connections write to table"
```

After upgrading the database to the current release they upgrade all the event monitor tables using the following command:

```
db2 "call evmon_upgrade_tables(null, null, null, ?, ?, ?)"
```

Note: The Parameter Value is 0



```
db2inst1@db2v10:~/essential/db2scripts> db2 "call evmon_upgrade_tables(null, null, null, ?, ?, ?)"

Value of output parameters
-----
Parameter Name : NUM_EVMONS_EVALUATED
Parameter Value : 0

Parameter Name : NUM_EVMONS_TO_UPGRADE
Parameter Value : 0

Parameter Name : NUM_EVMONS_UPGRADED
Parameter Value : 0

Result set 1
-----
EVMON_NAME EVMON_TYPE MESSAGE_TIME          MESSAGE_TEXT
-----
-          -          2012-06-22-19.20.25.900753 Verbose has been disabled because no
User Temporary Table Space exists.

1 record(s) selected.

Return Status = 0
db2inst1@db2v10:~/Documents/LabScripts/db2pt/essential/db2scripts>
```

Figure 22 – Upgrade all the event monitor tables

If instead they only wanted to upgrade act, they could use this command:

```
db2 "call evmon_upgrade_tables(null, 'ACTIVITIES', null, ?, ?, ?)"
```



```
db2inst1@db2v10:~/essential/db2scripts> db2 "call evmon_upgrade_tables(null, 'ACTIVITIES', null, ?, ?, ?)"

Value of output parameters
-----
Parameter Name : NUM_EVMONS_EVALUATED
Parameter Value : 0

Parameter Name : NUM_EVMONS_TO_UPGRADE
Parameter Value : 0

Parameter Name : NUM_EVMONS_UPGRADED
Parameter Value : 0

Result set 1
-----
EVMON_NAME EVMON_TYPE MESSAGE_TIME          MESSAGE_TEXT
-----
-          -          2012-06-22-19.23.27.329903 Verbose has been disabled because no User Temporary Table Space exists.

1 record(s) selected.

Return Status = 0
db2inst1@db2v10:~/Documents/LabScripts/db2pt/essential/db2scripts>
```

Figure 10 – To update only activities

Alternatively they could choose to upgrade only the statistic event monitors by using this command:

```
db2 "call evmon_upgrade_tables(null,'STATISTICS', null, ?, ?, ?)"
```



Figure 11 – To update only statistics

- Close up all process and terminals. This is the end of the lab exercises.

LAB FILES:

cleanRange.db2 - Cleans all objects created in IO lab
checkTableIndexStats.db2 - Queries syscat.tables and syscat indexes for stats_time
createDB.db2 - drop recreate db2pt
createRSTATS_T1.db2 - Creates RSTAT schema and table
dftmonOff.db2 - Turn off system monitor switches
dftmonOn.db2 - Not used now, manual entry in lab
get_bphits_all.db2 - Get bp hit ratio with snapdb and MON_GET_BUFFERPOOL()
HighestReadTimePerContainer.db2 queries with MON_GET_CONTAINERS()
manual_cleanup_bpmontune.db2 - clean up bufferpool tablespace table if WE cleanup fails in we_run_BPMonTune.sh
Mon_Get_Workload_Details.db2 - Get lock wait statistics
reads_per_trx.db2 - queries reads per trx from snapdb
rread_per_rrtrnd.db2 - queries rows read per rows returned MON_GET_WORKLOAD()
setUpLockWait.db2 - Create table insert row for lock wait
showHoldingApp.db2 - Admin view snaplockwait
tbl_bpmontune - dummy data for tables
we_run_BPMonTune.sh - Start WE scenario BPMonTune.sh
we_run_live_random.sh - Start WE scenario live_random (gui)
we_run_live_simple.sh - Start WE scenario live_simple (gui)
we_run_RangeNoSpread.sh - Start WE scenario RangeNoSpread.scn
we_run_RangeSpread.scn - Start WE scenario RangeSpread.scn
create_data.db2 – create data in the table documents

6 Cleanup

If you want to cleanup your work you can use these commands.

```
db2 force applications all  
db2 terminate  
db2 drop db sample  
db2stop
```



© Copyright IBM Corporation 2014
All Rights Reserved.

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

IBM, the IBM logo, ibm.com and Tivoli are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at ibm.com/legal/copytrade.shtml

Other company, product and service names may be trademarks or service marks of others.

References in this publication to IBM products and services do not imply that IBM intends to make them available in all countries in which IBM operates.

No part of this document may be reproduced or transmitted in any form without written permission from IBM Corporation.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS
DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER
EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
PARTICULAR PURPOSE OR NON-INFRINGEMENT.

IBM products are warranted according to the terms and conditions of the agreements (e.g. IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided.