IBM Software
Information Management Skills Development

DB2® Setup a database for testing BLU acceleration for query processing

Version/Revision# **1**

Module ID **10301**

Length **1.5 hours**

IBM

Table of Contents

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 you're 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.

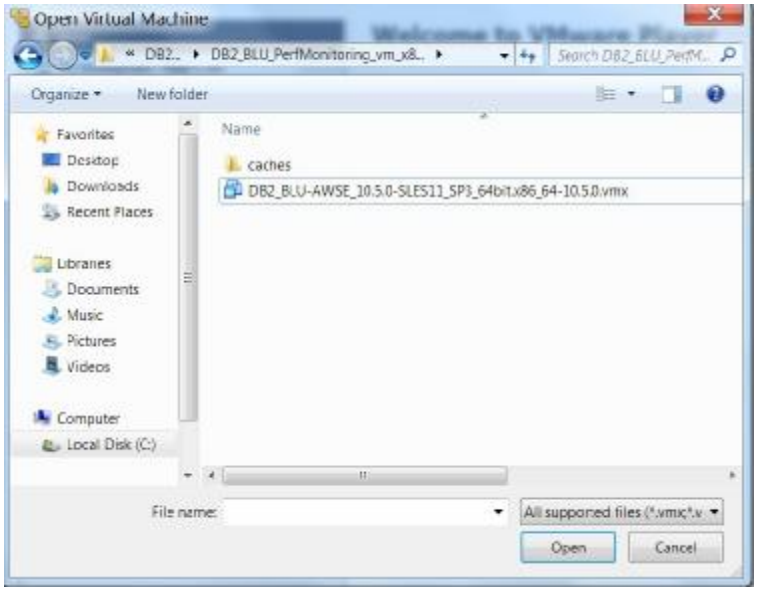

Figure 1 – Opening a VMware Image File

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.

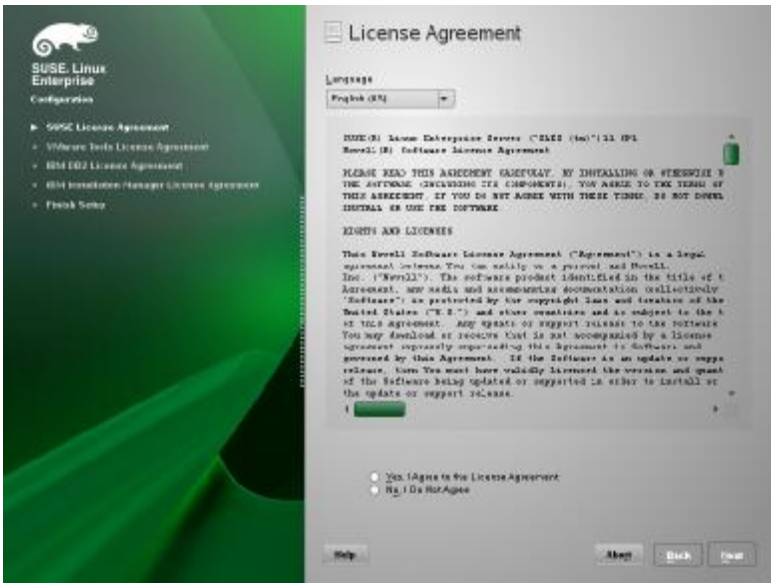

Figure 2 – License Agreements

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:

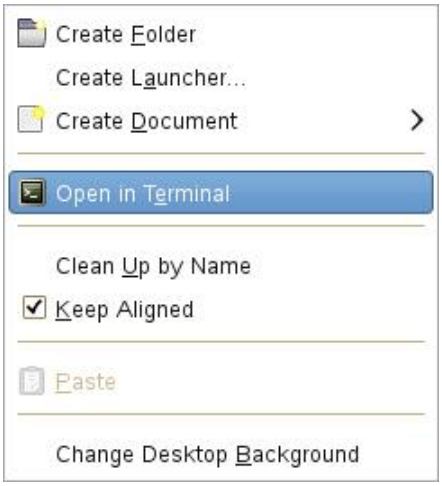

Figure 4 – Opening a Terminal

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 your lab scriptd.

```
cd /home/db2inst1/Documents/LabScripts/db2pt/db2blu
```

4 What this exercise is about

This exercise is an online lab which allows the student to implement DB2 BLU Acceleration for a new database. We will compare the disk space requirements for column-organized tables to standard row-organized tables, with and without compression. In the second part of the BLU lab you will run a set of sample queries to compare the access plans and

performance characteristics of query processing using DB2 BLU Acceleration with the processing techniques used for row-organized tables with indexes.

4.1 Setup a new DB2 database for testing DB2 BLU acceleration for query processing

1. In this exercise, new database DB2BLU will be used for implementing the DB2 BLU Acceleration feature. We will set the DB2 registry variable DB2_WORKLOAD to ANALYTICS before we create the new database.

   From your Linux VMware, login with the userid **db2inst1.** Start a new terminal session. In the terminal session, issued the following commands:

```
db2set db2_workload=ANALYTICS
db2set -all
db2 stop force
db2start
db2 terminate
db2 create database db2blu on /home/db2inst1
```

   Setting the DB2 Registry variable DB2_WORKLOAD to the value ANALYTICS before the databasae is created allows the database to be configured initially for DB2 BLU Acceleration. The database page size will be set to 32K rather than the 4K default and the default buffer pool will have a 32K page size.
   In the Linux terminal session issue the command:

```
db2 connect to db2blu
db2 –tvf checkcfg.sql
```

   The output should look similar to the following:

```
select name as CFG_option,
 varchar(value,30)  as configured_value
 from sysibmadm.dbcfg
 where name in ('sheapthres_shr','sortheap','util_heap_sz','dft_table_org'
 ,'self_tuning_mem','num_ioservers')
 order by name

CFG_OPTION                       CONFIGURED_VALUE
-------------------------------- ------------------------------
dft_table_org                    COLUMN
num_ioservers                    16
self_tuning_mem                  ON (Active)
sheapthres_shr                   32768
sortheap                         32768
util_heap_sz                     5000

  6 record(s) selected.
```

   Notice that the option DFT_TABLE_ORG is set to Column,so tables will be created as columns organized by default.

The database is configured to use self tuning memory management, but has the sort memory options SORTHEAP and SHEAPTHRES_SHR set manually since those can not be self tuned with DB2 BLU. Command.

We are going to create two sets of tables, one set defined with a schema of COLORG, and one set of row-organized tables using the schema ROWORG. First we will create tablespaces to store the tables and indexes. The file *tspace.dll* creates four new table spaces.

In the Linux terminal session, issue the command:

```
db2 –tvf tspace.ddl
```

The file *loadhistory.sql* contains the statements to load the table COLORG.HSITORY.

The output should look similar to the following:

```
CREATE TABLESPACE TSCOLD
DB20000I  The SQL command completed successfully.

CREATE TABLESPACE TSCOLI
DB20000I  The SQL command completed successfully.

CREATE TABLESPACE TSROWD
DB20000I  The SQL command completed successfully.

CREATE TABLESPACE TSROWI
DB20000I  The SQL command completed successfully.
```

We will initially create the first table COLORG.HISTORY as a row-organized table with standard indexes and then use ADMIN_MOVE_TABLE to convert the table to a column-organized table. Use the *create_colhist.ddl* to create the table COLORG.HISTORY.

In the Linux terminal session, issue the command:

```
db2 –tvf create_colhist.ddl
```

The output should look similar to the following:

```
set current schema = 'COLORG'
DB20000I  The SQL command completed successfully.
```

```
CREATE TABLE HISTORY (ACCT_ID          INTEGER         NOT NULL, TELLER_ID
SMALLINT        NOT NULL, BRANCH_ID        SMALLINT        NOT NULL, BALANCE
DECIMAL(15,2)   NOT NULL, DELTA            DECIMAL(9,2)    NOT NULL, PID
INTEGER         NOT NULL, TSTMP            TIMESTAMP       NOT NULL WITH
DEFAULT, ACCTNAME         CHAR(20)        NOT NULL, TEMP            CHAR(6)
NOT NULL) organize by row IN TSROWD INDEX IN TSROWI
DB20000I  The SQL command completed successfully.

CREATE INDEX HISTIX1 ON HISTORY (BRANCH_ID ASC) ALLOW REVERSE SCANS
DB20000I  The SQL command completed successfully.

CREATE INDEX HISTIX2 ON HISTORY (TELLER_ID ASC) ALLOW REVERSE SCANS
DB20000I  The SQL command completed successfully.
```

The **organize by row** clause is necessary in the new database since the default tableorganization was set to COLUMN.

Next we will use the LOAD utility to load a set of test data into the table.

The file *loadhistory.sql* contains the statements to load the table COLORG.HISTORY.

In the Linux terminal session, issue the command:

```
db2 –tvf loadhistory.sql
```

The output should look similar to the following:

```
load from histbranch.del of del messages loadihist.msg replace into
colorg.history

Number of rows read         = 513576
Number of rows skipped      = 0
Number of rows loaded       = 513576
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 513576
```

We can use ADMIN_MOVE_TABLE to convert the row-organized table to a column-organized table. The two indexes we created on the table COLORG.HISTORY are not unique indexes, so they will not be defined when the table is redefined. The file *convert_history.sql* contains the CALL statement for ADMIN_MOVE_TABLE to redefine the table as column-organized and changes the table space assignments. We will not keep a copy of the original table.

In the Linux terminal session, issue the command:

```
db2 –tvf convert_history.sql
```

The output should look similar to the following:

```
call SYSPROC.ADMIN_MOVE_TABLE ( 'COLORG', 'HISTORY', 'TSCOLD','TSCOLI','TSCOLD',
'ORGANIZE BY COLUMN',NULL,NULL,NULL, 'COPY_USE_LOAD,FORCE','MOVE')


  Result set 1
  --------------

  KEY                               VALUE
  --------------------------------- ----------------------------------------
----------------------------------------------------------------------------
-----
  AUTHID                            INST461
  CLEANUP_END                       2013-11-07-16.07.57.589451
  CLEANUP_START                     2013-11-07-16.07.57.296630
  COPY_END                          2013-11-07-16.07.56.649651
  COPY_OPTS                         LOAD,WITH_INDEXES,NON_CLUSTER
  COPY_START                        2013-11-07-16.07.46.659477
  COPY_TOTAL_ROWS                   513576
  INDEXNAME                         HISTIX1
  INDEXSCHEMA                       COLORG
  INDEX_CREATION_TOTAL_TIME         0
  INIT_END                          2013-11-07-16.07.46.558586
  INIT_START                        2013-11-07-16.07.44.198981
  ORIGINAL_TBLSIZE                  45696
  REPLAY_END                        2013-11-07-16.07.57.116417
  REPLAY_START                      2013-11-07-16.07.56.650246
  REPLAY_TOTAL_ROWS                 0
  REPLAY_TOTAL_TIME                 0
  STATUS                            COMPLETE
  SWAP_END                          2013-11-07-16.07.57.288726
  SWAP_RETRIES                      0
  SWAP_START                        2013-11-07-16.07.57.117257
  UTILITY_INVOCATION_ID
0100000002000000080000000000000000020131107160746560541000000000
  VERSION                           10.05.0000
```

```
  23 record(s) selected.

  Return Status = 0
```

The COPY_USE_LOAD option is important when converting to a column_organized table since the LOAD ANALYZE phase is needed to build the column dictionaries. The new table COLORG.HISTORY has catalog statistics collected, so a RUNSTATS is not necessary.

Notice: While ADMIN_MOVE_TABLE can be used to simplify converting a row-organized table to a column-organized table, it can not be used to convert a column-organized table to a row-organized table.

We will create the other three column-organized tables, ACCT, BRANCH, and TELLER using the COLORG schema and load them using the LOAD utility. Use the DB2 command file *tables.ddl* to create the tables.

In the Linux terminal session, issue the command:

```
db2 –tvf tables.ddl
```

The output should look similar to the following:

```
set current schema = 'COLORG'
DB20000I  The SQL command completed successfully.

CREATE TABLE ACCT (ACCT_ID          INT             NOT NULL, NAME
CHAR(20)        NOT NULL, ACCT_GRP         SMALLINT        NOT NULL, BALANCE
DECIMAL(15,2)   NOT NULL, ADDRESS          CHAR(30)        NOT NULL, TEMP
CHAR(40)        NOT NULL) IN TSCOLD INDEX IN TSCOLI
DB20000I  The SQL command completed successfully.

CREATE TABLE BRANCH (BRANCH_ID        SMALLINT        NOT NULL, BRANCH_NAME
CHAR(20)        NOT NULL, BALANCE          DECIMAL(15,2)   NOT NULL, AREA_CODE
CHAR(4)         NOT NULL, ADDRESS          CHAR(30)        NOT NULL, TEMP
CHAR(40)        NOT NULL) IN TSCOLD INDEX IN TSCOLI
DB20000I  The SQL command completed successfully.

CREATE TABLE TELLER (TELLER_ID        SMALLINT        NOT NULL, TELLER_NAME
CHAR(20)        NOT NULL, BRANCH_ID        SMALLINT        NOT NULL, BALANCE
DECIMAL(15,2)   NOT NULL, TELLER_CODE      CHAR(2)         NOT NULL, ADDRESS
CHAR(30)        NOT NULL, TEMP             CHAR(40)        NOT NULL) IN TSCOLD
INDEX IN TSCOLI
DB20000I  The SQL command completed successfully.
```

Next we will use the LOAD utility to load a set of test data into the table. The file *loadacct.*sql contains the statements to load the table COLORG.ACCT.

In the Linux terminal session, issue the command:

```
db2 –tvf loadacct.sql
```

The output should look similar to the following:

```
load from /home/db2inst1/Documents/Labscripts/db2pt/db2blu/reorg/acct.del of
del messages loadacct.msg replace into colorg.acct

Number of rows read         = 1000000
Number of rows skipped      = 0
Number of rows loaded       = 1000000
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 1000000
```

Review the messages generated by the LOAD utility processing for the table COLORG.ACCT:

In the Linux terminal session, issue the command:

```
cat loadacct.msg
```

The output should look similar to the following:

```
SQL3501W  The table space(s) in which the table resides will not be placed in
backup pending state since forward recovery is disabled for the database.

SQL3109N  The utility is beginning to load data from file
"/home/db2inst1/Documents/LabScripts/db2pt/db2blu/reorg/acct.del".

SQL3500W  The utility is beginning the "ANALYZE" phase at time "11/07/2013
16:09:36.394392".

SQL3519W  Begin Load Consistency Point. Input record count = "0".

SQL3520W  Load Consistency Point was successful.

SQL3515W  The utility has finished the "ANALYZE" phase at time "11/07/2013
16:09:41.737479".
```

```
SQL3500W  The utility is beginning the "LOAD" phase at time "11/07/2013
16:09:41.737852".

SQL3110N  The utility has completed processing.  "1000000" rows were read from
the input file.

SQL3519W  Begin Load Consistency Point. Input record count = "1000000".

SQL3520W  Load Consistency Point was successful.

SQL3515W  The utility has finished the "LOAD" phase at time "11/07/2013
16:09:49.197086".

SQL3500W  The utility is beginning the "BUILD" phase at time "11/07/2013
16:09:49.198691".

SQL3213I  The indexing mode is "REBUILD".

SQL3515W  The utility has finished the "BUILD" phase at time "11/07/2013
16:09:49.514587".


Number of rows read         = 1000000
Number of rows skipped      = 0
Number of rows loaded       = 1000000
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 1000000
```

Note that LOAD processing for the column-organized table includes the ANALYZE phase, needed to build the column compression dictionaries.

The tables BRANCH and TELLER will be loaded using the command file *loadothers.sql.*

In the Linux terminal session, issue the command:

```
db2 –tvf loadothers.sql
```

The output should look similar to the following:

```
load from branch.del of del messages loadbranch.msg replace into COLORG.branch

Number of rows read         = 100
Number of rows skipped      = 0
Number of rows loaded       = 100
```

```
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 100
```

```
load from teller.del of del messages loadteller.msg replace into COLORG.teller

Number of rows read         = 1000
Number of rows skipped      = 0
Number of rows loaded       = 1000
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 1000
```

Next we will create four row-organized tables and load them with the same test data used for the column-organized tables. The tables will use the schema ROWORG. The ACCT table will be created using adaptive compression.The other three tables, HISTORY, BRANCH; TELLER will not be defined as compressed. Use the DB2 command file *tablesrow.dll* to creat the tables.

In the Linux terminal session, issue the command:

```
db2 –tvf tablesrow.ddl
```

The output should look similar to the following:

```
set current schema = 'ROWORG'
DB20000I  The SQL command completed successfully.

CREATE TABLE ACCT (ACCT_ID          INT             NOT NULL, NAME             CHAR(20)
NOT NULL, ACCT_GRP         SMALLINT        NOT NULL, BALANCE          DECIMAL(15,2)   NOT NULL,
ADDRESS          CHAR(30)        NOT NULL, TEMP             CHAR(40)        NOT NULL) COMPRESS
YES ADAPTIVE organize by row IN TSROWD INDEX IN TSROWI
DB20000I  The SQL command completed successfully.

alter table acct add primary key (acct_id)
DB20000I  The SQL command completed successfully.

CREATE TABLE HISTORY (ACCT_ID          INTEGER         NOT NULL, TELLER_ID        SMALLINT
NOT NULL, BRANCH_ID        SMALLINT        NOT NULL, BALANCE          DECIMAL(15,2)   NOT NULL,
DELTA            DECIMAL(9,2)    NOT NULL, PID              INTEGER         NOT NULL, TSTMP
TIMESTAMP       NOT NULL WITH DEFAULT, ACCTNAME         CHAR(20)        NOT NULL, TEMP
CHAR(6)         NOT NULL) organize by row IN TSROWD INDEX IN TSROWI
DB20000I  The SQL command completed successfully.

CREATE INDEX HISTIX1 ON HISTORY (BRANCH_ID ASC) ALLOW REVERSE SCANS
```

```
DB20000I  The SQL command completed successfully.

CREATE INDEX HISTIX2 ON HISTORY (TELLER_ID ASC) ALLOW REVERSE SCANS
DB20000I  The SQL command completed successfully.

CREATE TABLE BRANCH (BRANCH_ID        SMALLINT        NOT NULL, BRANCH_NAME      CHAR(20)
NOT NULL, BALANCE          DECIMAL(15,2)   NOT NULL, AREA_CODE        CHAR(4)         NOT NULL,
ADDRESS          CHAR(30)        NOT NULL, TEMP             CHAR(40)        NOT NULL) organize
by row IN TSROWD
DB20000I  The SQL command completed successfully.

alter table branch add primary key (branch_id)
DB20000I  The SQL command completed successfully.

CREATE TABLE TELLER (TELLER_ID        SMALLINT        NOT NULL, TELLER_NAME      CHAR(20)
NOT NULL, BRANCH_ID        SMALLINT        NOT NULL, BALANCE          DECIMAL(15,2)   NOT NULL,
TELLER_CODE      CHAR(2)         NOT NULL, ADDRESS          CHAR(30)        NOT NULL, TEMP
CHAR(40)        NOT NULL) organize by row IN TSROWD
DB20000I  The SQL command completed successfully.

alter table teller add primary key (teller_id)
DB20000I  The SQL command completed successfully.
```

A few indexes were defined on these tables, that will be useful for the sample queries used for testing also for joining the tables together. Use the file *loadrowtabs.*dll to load the four tables and collect statistics using RUNSTATS.

In the Linux terminal session, issue the command:

```
db2 –tvf loadrowtabs.ddl
```

The output should look similar to the following:

```
load from /home/db2inst1/Documents/LabScripts/db2pt/db2blu/reorg/acct.del of del messages
loadacct.msg replace into roworg.acct

Number of rows read         = 1000000
Number of rows skipped      = 0
Number of rows loaded       = 1000000
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 1000000


reorg table roworg.acct use tempspace1 resetdictionary
DB20000I  The REORG command completed successfully.
```

```
runstats on table roworg.acct and indexes all
DB20000I  The RUNSTATS command completed successfully.

load from histbranch.del of del messages loadihist.msg replace into roworg.history

Number of rows read         = 513576
Number of rows skipped      = 0
Number of rows loaded       = 513576
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 513576


runstats on table roworg.history and indexes all
DB20000I  The RUNSTATS command completed successfully.

load from branch.del of del messages loadbranch.msg replace into ROWORG.branch

Number of rows read         = 100
Number of rows skipped      = 0
Number of rows loaded       = 100
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 100


runstats on table roworg.branch and indexes all
DB20000I  The RUNSTATS command completed successfully.

load from teller.del of del messages loadteller.msg replace into ROWORG.teller

Number of rows read         = 1000
Number of rows skipped      = 0
Number of rows loaded       = 1000
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 1000


runstats on table roworg.teller and indexes all
DB20000I  The RUNSTATS command completed successfully.
```

The processing included a REORG of the table ROWORG.ACCT following loading of data to create the most efficient compression dictionary. The RUNSTATS utility was run to collect table and index statistics for eacht table.

An important aspect of DB2 BLU Acceleration is the extreme compression for larger tables. We will use several queries to compare the storage requirements for the two sets of tables. All column-organized tables are compressed using the

column compression dictionaries. We defined the table ROWORG.ACCT to utilize Adaptive compression. Use the DB2 command file *qsyscat.sql* to query the statistics in the catalog table SYSCAT.TABLES.

In the Linux terminal session, issue the command:

```
db2 –tvf qsyscat.sql
```

The output should look similar to the following:

```
select varchar(tabschema,12) as schema,
  varchar(tabname,12) as table,
  card, tableorg,
  npages, fpages, mpages,
  pctpagessaved from syscat.tables
   where tabschema in ('COLORG','ROWORG')
   order by tabname

SCHEMA       TABLE        CARD                TABLEORG NPAGES              FPAGES               MPAGES
PCTPAGESSAVED
------------ ------------ ------------------- -------- ------------------- ------------------- --------
------------ -------------
COLORG       ACCT                     1000000 C                        137                 138
1            96
ROWORG       ACCT                     1000000 R                        568                 570
0            83
COLORG       BRANCH                       100 C                          8                   9
1             0
ROWORG       BRANCH                       100 R                          1                   2
0             0
COLORG       HISTORY                   513576 C                        171                 181
10            84
ROWORG       HISTORY                   513576 R                       1117                1118
0             0
COLORG       TELLER                      1000 C                          9                  10
1             0
ROWORG       TELLER                      1000 R                          4                   5
0             0

  8 record(s) selected.
```

The column-organized versions of the tables show most of the disk space is allocated in the column-organized object for the tables, the columns COL_OBJECT_P_SIZE AND COL_OBJECT_L_SIZE: The column dicionaries and other meta data are allocated in the data object, shown as DATA_OBJECT_P_SIZE.

The page map index for a column-organized table is shown as a small amount of space in the index object for each table, shown as INDEX_OBJECT_P_SIZE.

The amounts shown are kilobytes.

The larger row-organized tables ACCT and HISTORY show larger allocations for the data objects and index objects and no allocation for the column-organized object.

```
This concludes the first part of BLU Acceleration exercise.
```

5 Cleanup

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

```
db2 force applications all
db2 terminate
db2stop
```

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