

DB2[®] Maintenance Utilities and Performance

Version/Revision# 1
Module ID 10306
Length 2 hours



Table of Contents

1	Introduction.....	3
2	Suggested Reading	3
3	Lab Setup	3
4	Create Database.....	4
4.1	Create Database.....	4
4.1.1	Activate the Database.....	5
4.2	REORGCHK, REORG and RUNSTATS	6
4.2.1	REORGCHK.....	6
4.2.2	REORG.....	8
4.2.3	RUNSTATS.....	8
5	Configure Utility Performance	10
5.1	Utility Throttling.....	10
5.2	Backup.....	11
5.3	EXPORT Utility	18
5.4	IMPORT Utility	20
5.4.1	ALLOW NO WRITE ACCESS	20
5.4.2	COMMITCOUNT n AUTOMATIC	20
5.5	INGEST Utility.....	22
5.6	LOAD utility.....	26
§	SAVECOUNT n.....	27
§	DATA BUFFER buffer – size	27
§	SORT BUFFER buffer - size	27
§	CPU_PARALLELISM n	27
§	DISK_PARALLELISM	27
§	FETCH_PARALLELISM.....	27
§	INDEXING MODE.....	27
§	ALLOW NO READ ACCESS.....	27
§	LOCK WITH FORCE.....	27
6	Further maintenance utility discussions	28
7	Cleanup	29

1 Introduction

DB2 10 contains innovative features for delivering information on demand and scaling databases to new levels.

In this lab, you will learn about

- Automatic tuning DB2 performs for you when you use utilities
- Manual tuning considerations and options for utilities
- Monitoring the utilities DB2 provided for maintaining the integrity of your data
- Features and facilities available for configuring and monitoring the performance of DB2 and your applications

Performance and security are often left out of initial design considerations for new or migrated databases. We will address these as we proceed through the workshop.

Some of the information in this document has been taken directly from the DB2 Version 10 Information Center. The information may or may not have been modified for the purposes of this document.

2 Suggested Reading

Information Management Technical Library at developerWorks

Contains articles and tutorials

<http://www.ibm.com/developerworks/views/data/libraryview.jsp>

An Expert's Guide to DB2 Technology

Great read and useful information

<http://it.toolbox.com/blogs/db2luw>

DS4000 Best Practices and Performance Tuning Guide

<http://www.redbooks.ibm.com/abstracts/sg246363.html>

3 Lab Setup

Power on the virtual machine. Login as **db2inst1**. The password is **password**. Many of the commands you use to configure performance for DB2 can be quite long and complicated. We've put these commands into text files so you don't have to do as much typing in this lab. Please review each shell script or SQL file before you use it. To make it easier to review the code you'll be executing throughout the lab, open the files with SQL extension in an editor. Write permission for these files has been removed so you won't damage them while they're open in the editor. Alternatively, refer to the end of this document for the source code.

- Click **Computer** on the task bar (lower left corner).
- Click the **More applications...** pushbutton.
- Click **Tools** in the left navigation pane.
- Click the **gedit** text editor icon in the right pane.
- Click **File -> Open** on the menu bar.
- Navigate to the next directory:
/home/db2inst1/Documents/LabScripts/db2pt/utilities
- Select all the files with SQL extension in the directory and click the **Open** pushbutton.

4 Create Database

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

In this section we will create our database from a backup.

4.1 Create Database

In the previous exercise we placed our database logs in directory /logs. This directory will be reused in this exercise. Therefore, remove any old log files in this directory:

1. Open a terminal window as by right-clicking on the **Desktop** area and choose the “**Open Terminal**” item.

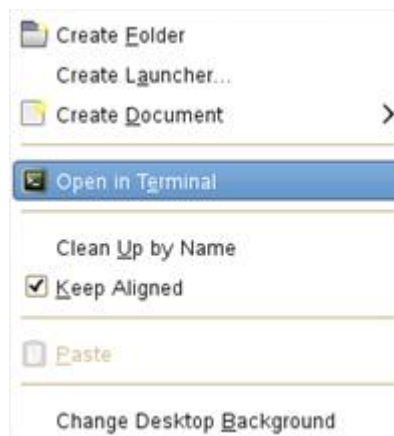


Figure 1 – Opening a Terminal

```
rm -rf /logs/*
```

We provide a script that contains the commands to create the database by restoring a backup of the xmldb database. Open a terminal session and enter the following commands to restore the database and the database objects needed for the lab exercises.

```
cd ~/Documents/LabScripts/db2pt/utilities  
db2start  
db2 -tvf restoreXMLdb.sql
```

4.1.1 *Activate the Database*

Deployment consideration: Database buffer pools, agents and other objects are not allocated in memory until the database is activated or an application connects to the database.

If a database has not been activated, the first application to connect to the database will incur a performance penalty as it waits for the database objects to be allocated and instantiated. In addition, when the last application disconnects from the database, the database objects are de-allocated. This means the next application to connect to the database will have to wait until the database objects are allocated. Another reason to activate the database is for monitoring purposes; performance metrics are typically accumulated upon either activation or first connection to the database.

You can remove this performance inhibitor by activating the database when the system starts. Database objects will remain allocated until the `deactivate database` command is issued. This will also give DB2's autonomic computing features the best opportunity to provide their benefits. Refer to the `activate database` command for more information on this subject

```
db2stop  
db2start  
db2 activate db xmldb  
db2 connect to xmldb
```



```
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities
File Edit View Terminal Help
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 -tvf restoreXMLdb.s
1
force applications all
DB20000I The FORCE APPLICATION command completed successfully.
DB21024I This command is asynchronous and may not be effective immediately.

deactivate database xmldb
SQL1013N The database alias name or database name "XMLDB" could not be found.
SQLSTATE=42705

drop database xmldb
SQL1013N The database alias name or database name "XMLDB" could not be
found. SQLSTATE=42705

restore database xmldb from . on /home/db2inst1
DB20000I The RESTORE DATABASE command completed successfully.

activate database xmldb
DB20000I The ACTIVATE DATABASE command completed successfully.

db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2stop
06/22/2012 15:12:19 0 0 SQL1054N DB2STOP processing was successful.
SQL1054N DB2STOP processing was successful.
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2start
06/22/2012 15:13:38 0 0 SQL1053N DB2START processing was successful.
SQL1053N DB2START processing was successful.
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 activate db xmldb
DB20000I The ACTIVATE DATABASE command completed successfully.
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 connect to xmldb

Database Connection Information

Database server      = DB2/LINUX 10.1.0
SQL authorization ID = DB2INST1
Local database alias = XMLDB

db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities>
```

Figure 2 – Restore y Activate database

4.2 REORGCHK, REORG and RUNSTATS

Upon restoration of the database, the database will have newly created indexes on the table, db2inst1.xmlfiles. The indexes were created to enhance application performance. However, our data is not arranged optimally in the tablespaces because the indexes were created after the data was inserted. The utilities in this section will help us cluster the data based on the indexes, compress the indexes and rebuild the compression dictionary.

4.2.1 REORGCHK

The REORGCHK command calculates statistics on the database to determine if tables, indexes or both need to be reorganized or cleaned up.

We will run the reorgchk command twice. The first time we will use the current statistics for the database, and then we will supply an option that will update the statistics and then check if a reorg is needed. Enter the following command **on a single line**:

```
db2 reorgchk current statistics on table db2inst1.xmlfiles >reorgchk_1.txt
```

You can review the reorgchk command's results by opening the reorgchk_1.txt by displaying its contents with the `cat`, `more` or `less` commands. Results during lab development showed that table did not require a reorg (F1 F2 F3 REORG results were ---), but that an index reorg was recommended (F4 F5 F6 F7 F8 REORG results were *----).

```
cat reorgchk_1.txt
```

```

Table: DB2INST1.XMLFILES
Index: DB2INST1.IDX1
      101      982      3      0      2      0      982      101
      3534      3534      50      97      82
      0      0      ---
Index: DB2INST1.IDX2
      4      882      1      0      1      0      715      4
      5122      5122      0      70      -
      0      0      *----
Index: DB2INST1.IDX3
      97      982      2      0      2      0      982      97
      4452      4452      33      97      221
      0      0      ---
Index: SYSIBM.SQLO91221174225250
      0      0      1      0      1      0      0      0
      7355      7355      0      100      -
      0      0      ---
Index: SYSIBM.SQLO91221174225700
      44      4065      12      0      2      0      4065      48
      1242      1328      0      -      66
      0      0      ---
CLUSTERFATOR or normalized CLUSTERFACTOR (F4) will indicate REORG is necessary
for indexes that are not in the same sequence as the base table. When multiple
indexes are defined on a table, one or more indexes may be flagged as needing
REORG. Specify the most important index for REORG sequencing.

```

Figure 3 – REORGCHK output

The `UPDATE STATISTICS` parameter calls the `RUNSTATS` routine to update table and index statistics, and then uses the updated statistics to determine if table or index reorganization is required. Run the command to update the statistics and review the results. Enter the following command **on a single line**:

```
db2 reorgchk update statistics on table db2inst1.xmlfiles >reorgchk_2.txt
```

Results during lab development showed similar results for both commands, meaning that the statistics for this table were already up-to-date. This is a small table with relatively simple statistics. Results for production databases may show that you should use the `UPDATE STATISTICS` option for more accurate results, provided that you have a large enough window. (`RUNSTATS` may be costly in terms of lapsed time and catalog locking.) Alternatively, you can enable automatic `RUNSTATS`, so that DB2 will gather statistics in the background as needed.

4.2.2 REORG

The `reorg` command is used to reorganize (defragment) an index or a table. Running a `reorg` for indexes rebuilds the index data into unfragmented, physically contiguous pages. Running a `reorg` for a table reconstructs, or rebuilds the rows, compacting the table by eliminating fragmented data. In addition, you can rebuild the data compression dictionary and recompress the data accordingly.

On a partitioned table, you can reorganize a single partition. You can reorganize a specific nonpartitioned index, or you can reorganize all the partitioned indexes on a specific data partition.

If you specify the `CLEANUP` option of the index clause, cleanup of logically deleted indexes is performed without rebuilding the indexes. This command cannot be used against indexes on declared temporary tables or created temporary tables (SQLSTATE 42995).

Enter the following commands to `reorg` the indexes and table data and create a new compression dictionary.

```
db2 reorg indexes all for table db2inst1.xmlfiles
db2 reorg table db2inst1.xmlfiles resetdictionary
```

4.2.3 RUNSTATS

There are two essential times you should execute the `runstats` utility against a table: after the table has had a lot of updates or after you reorganize the table. You should execute the `runstats` utility against statistical views that have been enabled for optimization when modifications to their underlying tables affect the rows returned by the views.

You do not need any explicit privilege to use this command on any declared temporary table that exists within its connection.

We want to update the statistics for the database because we have just run a `reorg` operation. We can base the statistics on a sample, or percentage of the data in the table. DB2's `runstats` command offers two types of sampling: Bernoulli and system. Bernoulli sampling evaluates a sample set of rows. System sampling evaluates a sample set of data pages. System sampling introduces less overhead than Bernoulli sampling but may not produce as accurate results when your data is highly clustered.

In this example we specify the `util_impact_priority` parameter with its lowest value so it will run at a very low priority. We specify “allow read access” so other applications will be able to access the data while the command runs. We specify 100 for both the sampling method and the repeatable parameter so that every row (100 percent of the rows) will be analyzed. You will probably want to use a much lower value for larger tables. You may need to compare results with no sampling to find the lowest percentage that will yield “good” statistics. This can become critical to performance when hundreds of millions of rows exist. Enter the following command to capture the new statistics for our database.

```
db2 -tvf runstats.sql
```



```

db2inst1@db2v10:~/db2pt/utilities
File Edit View Terminal Help
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 reorgchk update sta
tistics on table db2inst1.xmlfiles >reorgchk_2.txt
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 reorg indexes all f
or table db2inst1.xmlfiles
DB20000I The REORG command completed successfully.
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 reorg table db2inst
1.xmlfiles resetdictionary
DB20000I The REORG command completed successfully.
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 -tvf runstats.sql
runstats on table db2inst1.xmlfiles on all columns and detailed indexes all all
ow read access table sample bernoulli (100) repeatable (100) util_impact_priorit
y 1
DB20000I The RUNSTATS command completed successfully.
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities>

```

Figure 4 – REORGCHK, REORG, and RUNSTATS

The first `reorg` command we ran in the last section was for the indexes. Review that `reorg` command's results by running the `reorgchk` command again, as shown below. Enter the following command **on a single line**:

```
db2 reorgchk update statistics on table db2inst1.xmlfiles >reorgchk_3.txt
```

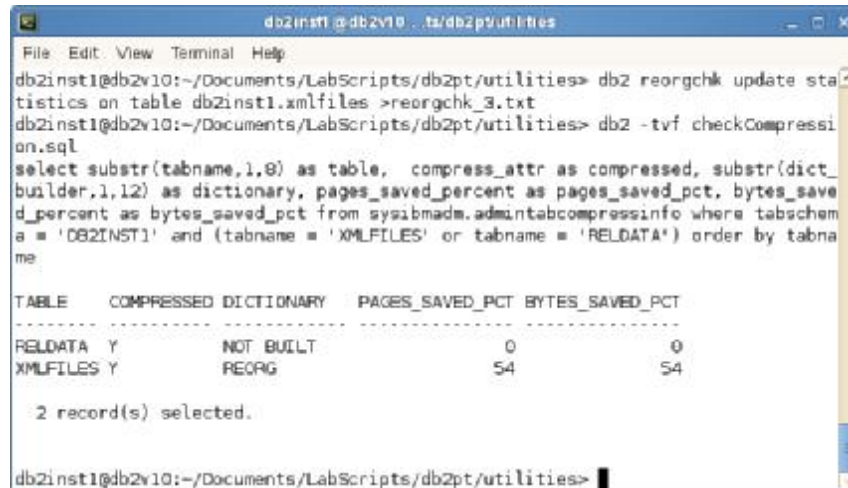
Compare files `reorgchk_2.txt` and `reorgchk_3.txt`. There should no longer be any indexes needing a `reorg`

The second `reorg` command we ran in the last section reorganized the data, creating a new compression dictionary. Enter the following command to display the current compression statistics. Record the results in to table below.

```
db2 -tvf checkCompression.sql
```

TYPE	DICTIONARY	PAGES SAVED %	BYTES SAVED %
DATA			
XML			

Table 1 XMLDB compression statistics after reorg and runstats



```
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 reorgchk update statistics on table db2inst1.xmlfiles >reorgchk_3.txt
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 -tvf checkCompression.sql
select substr(tabname,1,8) as table, compress_attr as compressed, substr(dict_builder,1,12) as dictionary, pages_saved_percent as pages_saved_pct, bytes_saved_percent as bytes_saved_pct from sysibmadm.admintabcompressinfo where tabschema = 'DB2INST1' and (tabname = 'XMLFILES' or tabname = 'RELODATA') order by tabname

TABLE      COMPRESSED DICTIONARY      PAGES_SAVED_PCT BYTES_SAVED_PCT
-----
RELODATA   Y          NOT BUILT          0                0
XMLFILES   Y          REORG              54               54

2 record(s) selected.

db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities>
```

Figure 5 – REORGCHK and compression check

5 Configure Utility Performance

In the previous section we reviewed the RUNSTATS, REORG, REORGCHK. These utilities impact directly the runtime performance of the database. In this section we will focus on the data management utilities for data movement.

5.1 Utility Throttling

Utility throttling regulates the performance impact of maintenance utilities, so that it is possible to run them concurrently during production periods. Although the impact policy, a setting that allows utilities to run in throttled mode, is defined by default, you must set an impact priority for each utility which you want to throttle.

The throttling system ensures that the throttled utilities are run as frequently as possible when necessary without violating the impact policy. You can throttle statistics collection, backup operations, rebalancing operations, and asynchronous index cleanups.

First, you define the impact policy by setting the UTIL_IMPACT_LIM configuration parameter.

Check the current setting for the impact policy with the following command.

```
db2 get dbm cfg | grep -i util
```



```
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 get dbm cfg | grep -i util
Workload impact by throttled utilities(UTIL_IMPACT_LIM) = 10
WLM dispatcher min. utilization (%) (WLM_DISP_MIN_UTIL) = 5
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities>
```

Figure 6 – UTIL_IMPACT_LIM

The value of the `util_impact_lim` parameter determines what impact all running utilities can have on the total database workload. For example, if the value of this parameter is 10, then all running utilities together can have no more than a 10% average impact upon the total workload for the database. If the value is set to 100, then no throttling will occur.

Refining this a bit further, you can manage the priority of individual utilities within the average impact allotment. For example, you could give more resources to `RUNSTATS` and fewer to `BACKUP`, or vice versa. You use the `UTIL_IMPACT_PRIORITY` clause of the `RUNSTATS` or `BACKUP` command to set the relative priority of an individual utility within the `UTIL_IMPACT_LIM` percentage. If you omit the `UTIL_IMPACT_PRIORITY` clause, the utility runs unthrottled, regardless of the setting of `UTIL_IMPACT_LIM`.

You may also use the `SET UTIL_IMPACT_PRIORITY` command or the `db2UtilityControl` API to enable throttling, or to set or change the relative priority of an individual utility. By default, within the impact allotment, each utility has a utility impact priority of 50 (acceptable values are between 1 and 100, with 0 indicating no throttling).

5.2 Backup

DB2 allows you to backup an entire database or selected tablespaces. You can perform a hot (online) or cold (offline) backup depending on how the transaction logging is configured.

Data, backups, active transaction logs and archived transaction logs should be stored on separate physical devices for two reasons:

- First, if you lose the active database you don't lose the backup and transaction logs, too. If you did there would be no way to recover or restore the database.
- Second, disk I/O will not be a bottleneck as you read from the database and write to the backup and/or transaction logs.

We have configured separate drives for active transaction logs (`/logs`), archived transaction logs (`/archive`), and database backups (`/backups`).

Let's suppose that the Service Level Agreement (SLA) states that the database should remain online at all times except during the quarterly maintenance periods. To reduce the amount of time it would take to recover a database after a disaster you should perform a full database offline backup before putting the database online. Then you could take another full offline backup during each maintenance period.

Before performing a backup operation we need to change transaction logging from circular mode to archival mode. This will allow you to take an online backup of the database. In order to do this we need to set the `logarchmeth1` parameter.

```
db2 connect to xmldb
db2 update db cfg using logarchmeth1 disk:/archive
```

We have a disk drive set aside for transaction logs. Specify the new transaction log setting with the following command.

```
db2 update db cfg using newlogpath /logs
```



```
db2inst1@db2v10:~/db2pt/utilities
File Edit View Terminal Help
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 connect to xmldb

Database Connection Information

Database server          = DB2/LINUX 10.1.0
SQL authorization ID    = DB2INST1
Local database alias    = XMLDB

db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 update db cfg using
logarchmeth1 disk:/archive
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
SQL1363W  Database must be deactivated and reactivated before the changes to
one or more of the configuration parameters will be effective.
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 update db cfg using
newlogpath /logs
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
SQL1363W  Database must be deactivated and reactivated before the changes to
one or more of the configuration parameters will be effective.
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities>
```

Figure 7 – Enable log archiving and move logs

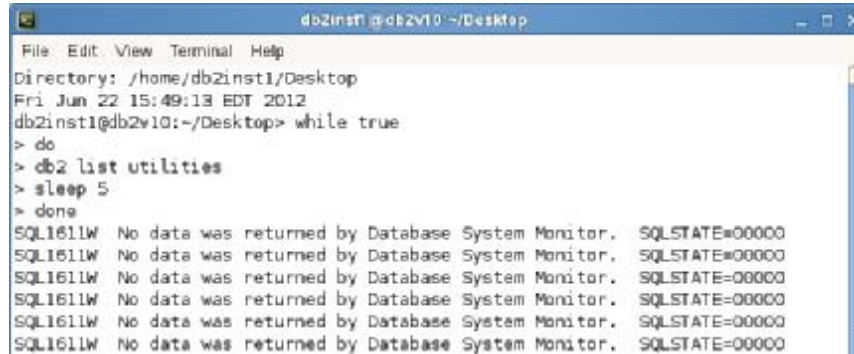
The messages issued when you executed the previous commands indicate that the changes have not yet taken effect. We need to terminate all connections to the database to activate the changes. To be certain, and since we aren't in production mode, we'll also restart DB2. Enter the following commands to activate the logging changes.

```
db2 connect reset
db2 deactivate db xmldb
db2stop
db2start
db2 activate db xmldb
```

i **Note:** We cannot activate the database, nor can applications connect to the database, which is now in a **BACKUP PENDING** state. A message to this effect is returned by the activate command and is also placed in the DB2 diagnostics log. By default, this log can be found at `~/sqllib/db2dump/db2diag.log`

Start a **new terminal** session or terminal session tab. We'll use it to monitor utilities as they run. Start a **WHILE** loop in the new session. Press Enter after each of the following lines. This loop will return message SQL1611W when no utilities are running.

```
while true
do
db2 list utilities
sleep 5
done
```



```
db2inst1@db2v10:~/Desktop
File Edit View Terminal Help
Directory: /home/db2inst1/Desktop
Fri Jun 22 15:49:13 EDT 2012
db2inst1@db2v10:~/Desktop> while true
> do
> db2 list utilities
> sleep 5
> done
SQL1611W No data was returned by Database System Monitor.  SQLSTATE=00000
SQL1611W No data was returned by Database System Monitor.  SQLSTATE=00000
SQL1611W No data was returned by Database System Monitor.  SQLSTATE=00000
SQL1611W No data was returned by Database System Monitor.  SQLSTATE=00000
SQL1611W No data was returned by Database System Monitor.  SQLSTATE=00000
SQL1611W No data was returned by Database System Monitor.  SQLSTATE=00000
```

Figure 8 – LIST UTILITIES

DB2 automatically calculates the numbers of buffers it needs for a backup and the parallelism to be used to read tablespaces concurrently. DB2 records the actions it takes to optimize backup performance using its autonomic computing features in the db2diag.log. Enter the following command to backup the database, and then switch to the other terminal session to monitor the backup process.

```
db2 backup db xmldb to /backup compress
```

In the diagnostics log (/home/db2inst1/sqllib/db2dump/db2diag.log) you can see STMM modifying memory consumers parameters. We observed, during the creation of the lab materials, that DB2 set the parallelism to 2 and used 4 backup buffers of size 4096.

```
2012-06-22-15.53.15.981725-240 E53859946G537      LEVEL: Info
PID      : 24783      TID : 2883578736      PROC : db2sysc 0
INSTANCE: db2inst1      NODE : 000      DB : XMLDB
APPHDL   : 0-9      APPID: *LOCAL.db2inst1.120622195231
AUTHID   : DB2INST1      HOSTNAME: db2v10
EDUID    : 22      EDUNAME: db2agent (XMLDB) 0
FUNCTION: DB2 UDB, database utilities, sqlubTuneBuffers, probe:903
DATA #1 : <preformatted>
Autonomic backup - tuning enabled.
Using buffer size = 4097, number = 4.

2012-06-22-15.53.16.594853-240 E53860484G485      LEVEL: Info
PID      : 24783      TID : 2883578736      PROC : db2sysc 0
INSTANCE: db2inst1      NODE : 000      DB : XMLDB
APPHDL   : 0-9      APPID: *LOCAL.db2inst1.120622195231
AUTHID   : DB2INST1      HOSTNAME: db2v10
EDUID    : 22      EDUNAME: db2agent (XMLDB) 0
FUNCTION: DB2 UDB, database utilities, sqlubSetupJobControl, probe:1687
MESSAGE : Starting an offline db backup.

2012-06-22-15.54.00.592112-240 E53860970G458      LEVEL: Info
PID      : 24783      TID : 2883578736      PROC : db2sysc 0
INSTANCE: db2inst1      NODE : 000      DB : XMLDB
APPHDL   : 0-9      APPID: *LOCAL.db2inst1.120622195231
AUTHID   : DB2INST1      HOSTNAME: db2v10
EDUID    : 22      EDUNAME: db2agent (XMLDB) 0
FUNCTION: DB2 UDB, database utilities, sqlubcka, probe:882
MESSAGE : Backup complete.

2012-06-22-15.54.00.880837-240 E53861429G515      LEVEL: Event
PID      : 24783      TID : 2883578736      PROC : db2sysc 0
INSTANCE: db2inst1      NODE : 000      DB : XMLDB
APPHDL   : 0-9      APPID: *LOCAL.db2inst1.120622195231
AUTHID   : DB2INST1      HOSTNAME: db2v10
EDUID    : 22      EDUNAME: db2agent (idle) 0
FUNCTION: DB2 UDB, base sys utilities,
sqeLocalDatabase::FreeResourcesOnDBShutdown, probe:13101
STOP     : DATABASE: XMLDB      : DEACTIVATED: NO
```

Figure 9 – db2diag.log

The output of the LIST UTILITIES command shows that the backup was unthrottled. It also gave us an estimated percentage complete.

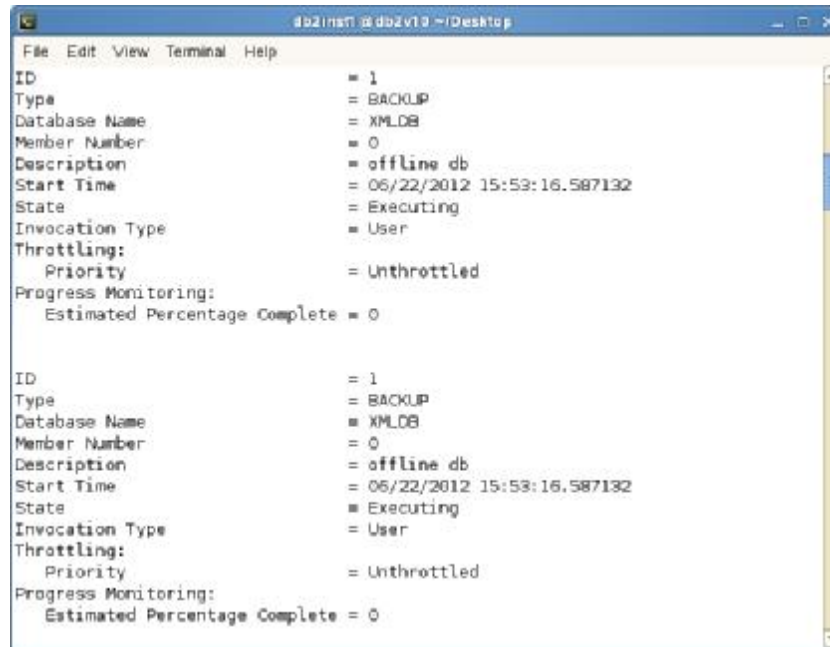
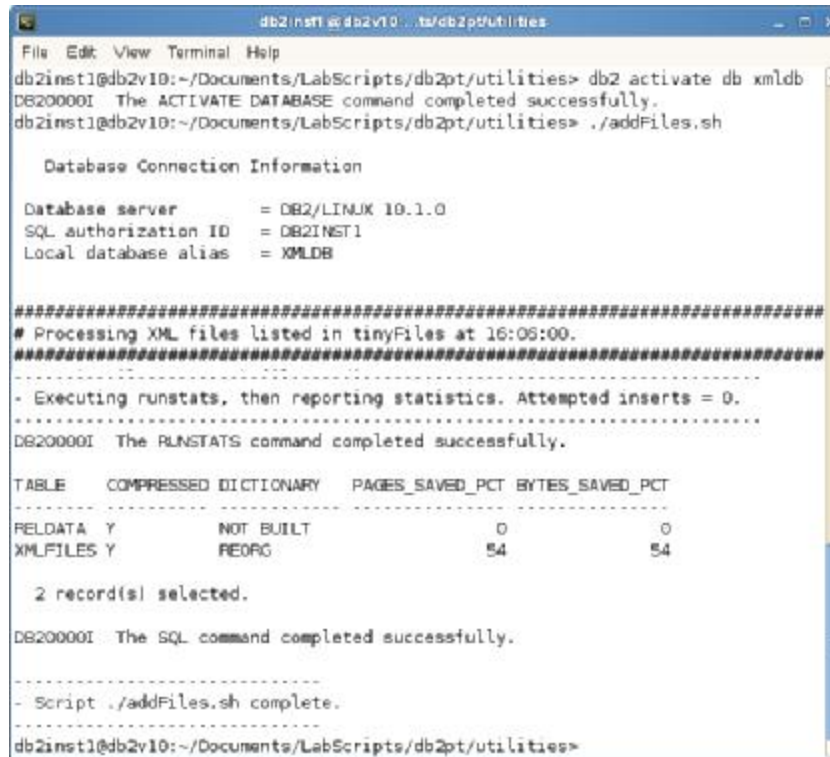


Figure 10 – LIST UTILITIES

Switch to the terminal session tab that isn't monitoring activities. In this session you will activate the database (it's ready for production use now), add more files to the database, and then perform an online backup which you will throttle. Add more data with the following command.

```
db2 activate db xmldb
./addFiles.sh
```



```
db2inst1@db2v10...t/db2pt/utilities
File Edit View Terminal Help
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 activate db xmldb
DB20000I The ACTIVATE DATABASE command completed successfully.
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> ./addFiles.sh

Database Connection Information

Database server      = DB2/LINUX 10.1.0
SQL authorization ID = DB2INST1
Local database alias = XMLDB

#####
# Processing XML files listed in tinyFiles at 16:06:00.
#####
- Executing runstats, then reporting statistics. Attempted inserts = 0.
DB20000I The RUNSTATS command completed successfully.

TABLE    COMPRESSED DICTIONARY    PAGES_SAVED_PCT    BYTES_SAVED_PCT
-----
RELDATA  Y          NOT BUILT              0              0
XMLFILES Y          REORG              54             54

2 record(s) selected.

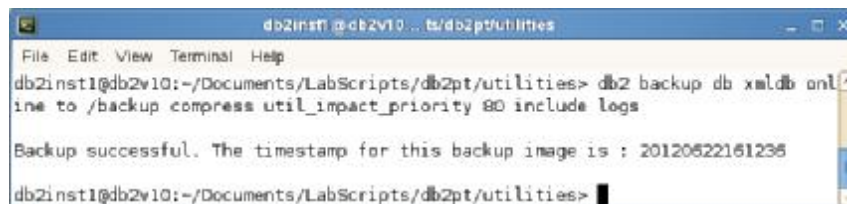
DB20000I The SQL command completed successfully.

-----
- Script ./addFiles.sh complete.
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities>
```

Figure 11 – Activate and add data

After the script completes, enter the following command on a single line to backup the database, and then switch to the other terminal session to monitor the backup process. Providing the util_impact_priority clause and value will throttle the backup operation.

```
db2 backup db xmldb online to /backup compress util_impact_priority 80
include logs
```



```
db2inst1@db2v10...t/db2pt/utilities
File Edit View Terminal Help
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities> db2 backup db xmldb onl
ine to /backup compress util_impact_priority 80 include logs

Backup successful. The timestamp for this backup image is : 20120622161236
db2inst1@db2v10:~/Documents/LabScripts/db2pt/utilities>
```

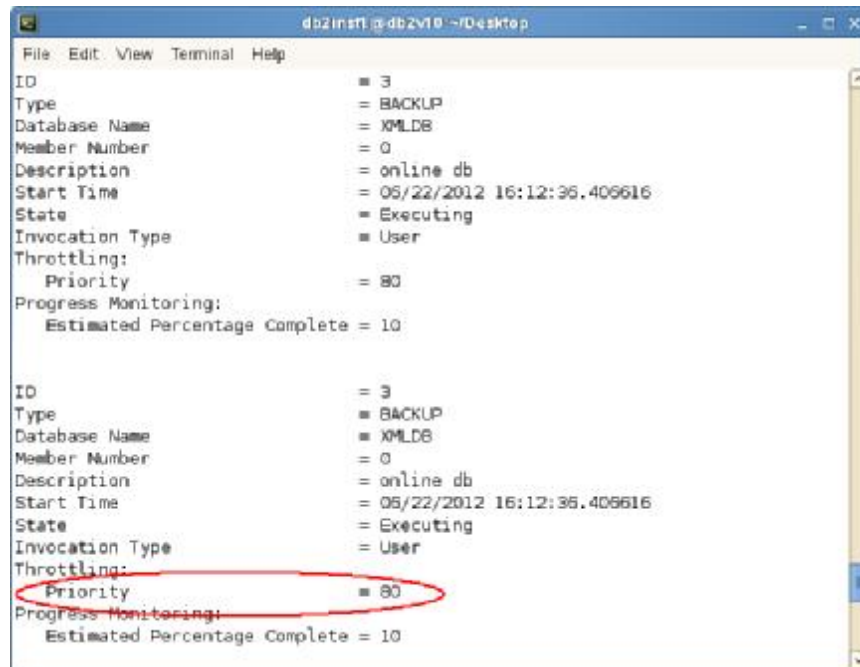
Figure 12 – online BACKUP

You should notice in db2diag.log that an online backup was started and the transaction log was archived. The WHILE loop should show you that the backup was started in throttled mode using the percentage you supplied, and that specifying the UTIL_IMPACT_PRIORITY on the BACKUP command changed to the priority to 80.

Note: Relationship between UTIL_IMPACT_LIM and UTIL_IMPACT_PRIORITY

The database manager configuration parameter UTIL_IMPACT_LIM sets the limit on the amount of impact throttled utilities can have on the overall workload on the machine. 0-99 is a throttled percentage, 100 is no throttling.

UTIL_IMPACT_PRIORITY is set using the SET UTIL_IMPACT_PRIORITY command. It sets the priority that a particular utility has over the resources available to throttled utilities as defined by UTIL_IMPACT_LIM. (0 = unthrottled)



```
db2inst1@db2v10: ~/Desktop
File Edit View Terminal Help
ID                                     = 3
Type                                 = BACKUP
Database Name                        = XMLDB
Member Number                       = 0
Description                          = online db
Start Time                          = 06/22/2012 16:12:36.406616
State                               = Executing
Invocation Type                     = User
Throttling:
  Priority                           = 80
Progress Monitoring:
  Estimated Percentage Complete = 10

ID                                     = 3
Type                                 = BACKUP
Database Name                        = XMLDB
Member Number                       = 0
Description                          = online db
Start Time                          = 06/22/2012 16:12:36.406616
State                               = Executing
Invocation Type                     = User
Throttling:
  Priority                           = 80
Progress Monitoring:
  Estimated Percentage Complete = 10
```

Figure 13 – LIST UTILITIES

Press Ctrl+C in this terminal session to cancel the WHILE loop.

In addition to the storage savings you can achieve through row compression in your active database, you can also use backup compression to reduce the size of your database backups.

Whereas row compression works on a table-by-table basis, when you use compression for your backups, all of the data in the backup image is compressed, including catalog tables, index objects, LOB objects, auxiliary database files and database meta-data.

You can use backup compression with tables that use row compression. Keep in mind, however, that backup compression requires additional CPU resources and extra time. It may be sufficient to use table compression alone to achieve a reduction in your backup storage requirements. If you are using row compression, consider using backup compression only if storage optimization is of higher priority than the extra time it takes to perform the backup.

Tip: Consider using backup compression only on table spaces that do not contain compressed data if the following conditions apply:

- Data and index objects are separate from LOB and long field data, and

- You use row and index compression on the majority of your data tables and indexes, respectively

To use compression for your backups, use the `COMPRESS` option on the `BACKUP DATABASE` command.

5.3 EXPORT Utility

The `EXPORT` utility, or `EXPORT` option of the `DB2MOVE` utility, is cross-platform compatible. This utility is best suited for situations where you want to store data in an external file, either to process it further or to move data to another table. You can export large objects and XML columns in addition to relational data. `DB2MOVE` exports tables in PC/IXF format. If you want to save the data in the delimited ASCII format you should use the `EXPORT` command instead of `DB2MOVE`.

The `EXPORT` command has no performance optimization parameters. There are a few ways to improve the export utility's performance. As the export utility is an embedded SQL application and does SQL fetches internally, optimizations that apply to SQL operations apply to the export utility as well. Consider taking advantage of large buffer pools, plus indexing and sort heap if a `WHERE` clause is used. In addition, try to minimize device contention on the output files by placing them away from the containers and log devices. Large objects and XML files can be exported in round robin fashion by specifying multiple directories.

`EXPORT` does not move database objects associated with the tables such as aliases, views, or triggers, or the objects tables may depend on, such as user-defined types or user-defined functions. You will need to manually move these objects. They can be created with the `DB2LOOK` command.

Create a directory to store the extracted data in and list the number of records in the table with the following commands.

```
mkdir /tmp/db2move
cd /tmp/db2move
db2 connect to xmldb
db2 "select count(*) from reldata"
db2 connect reset
```



```
db2inst1@db2v10:/tmp/db2move
File Edit View Terminal Help
db2inst1@db2v10:/tmp/db2move> db2 connect to xmldb

Database Connection Information

Database server      = DB2/LINUX 10.1.0
SQL authorization ID = DB2INST1
Local database alias = XMLDB

db2inst1@db2v10:/tmp/db2move> db2 "select count(*) from reldata"

1
-----
      1107

1 record(s) selected.

db2inst1@db2v10:/tmp/db2move> db2 connect reset
DB200000I The SQL command completed successfully.
db2inst1@db2v10:/tmp/db2move>
```

Figure 14 – Query into table RELDATA

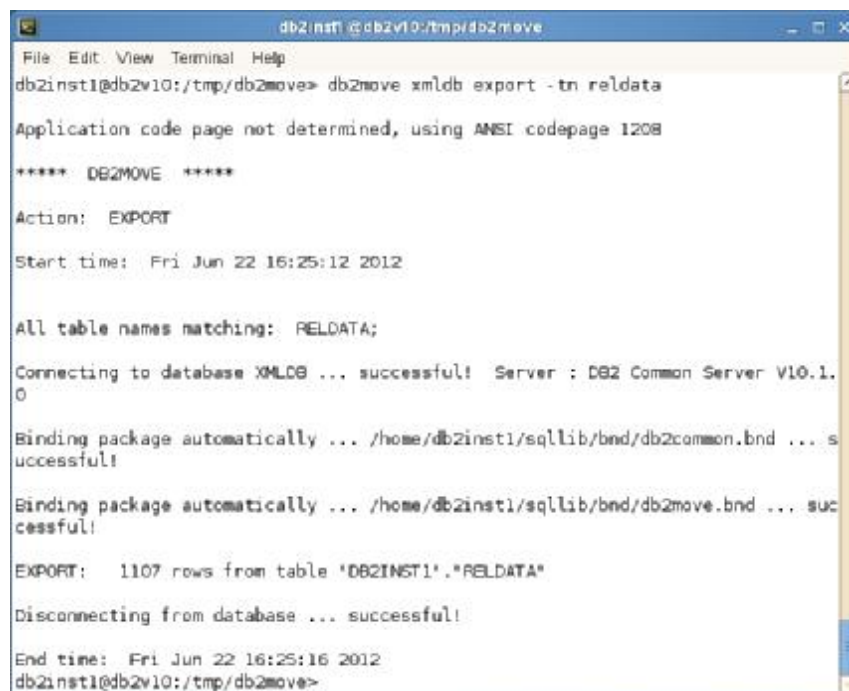
Choose one of the following commands to export the data. Both versions of this command should create identical output and results. Notice that the DB2MOVE command syntax is simpler. The EXPORT command should be entered on a single line.

Either:

```
db2 connect to xmldb
db2move xmldb export -tn reldata
```

Or:

```
db2 connect to xmldb
db2 "export to tabl.ixf of ixf messages tabl.msg select * from reldata"
```



```
db2inst1@db2v10:/tmp/db2move
File Edit View Terminal Help
db2inst1@db2v10:/tmp/db2move> db2move xmldb export -tn reldata

Application code page not determined, using ANSI codepage 1208

**** DB2MOVE ****

Action: EXPORT

Start time: Fri Jun 22 16:25:12 2012

All table names matching: RELDAPA;

Connecting to database XMLDB ... successful! Server : DB2 Common Server V10.1.0

Binding package automatically ... /home/db2inst1/sqllib/bnd/db2common.bnd ... successful!

Binding package automatically ... /home/db2inst1/sqllib/bnd/db2move.bnd ... successful!

EXPORT: 1107 rows from table 'DB2INST1'.RELDAPA

Disconnecting from database ... successful!

End time: Fri Jun 22 16:25:16 2012
db2inst1@db2v10:/tmp/db2move>
```

Figure 15 – Export Data with db2move

You can review the files created by the EXPORT command with the following commands. The EXPORT.out file is available if you used the DB2MOVE command.

```
ls
cat EXPORT.out
cat tabl.msg
```

Delete the data in the table and verify the records have been deleted with the following commands.

```
db2 connect to xmldb
db2 truncate table reldata reuse storage immediate
db2 "select count(*) from reldata"
db2 connect reset
```



```
db2inst1@db2v10:/tmp/db2move
File Edit View Terminal Help
db2inst1@db2v10:/tmp/db2move> db2 connect to xmldb

Database Connection Information

Database server      = DB2/LINUX 10.1.0
SQL authorization ID = DB2INST1
Local database alias = XMLDB

db2inst1@db2v10:/tmp/db2move> db2 truncate table reldata reuse storage immediate
DB20000I The SQL command completed successfully.
db2inst1@db2v10:/tmp/db2move> db2 "select count(*) from reldata"

1
-----
0

1 record(s) selected.

db2inst1@db2v10:/tmp/db2move> db2 connect reset
DB20000I The SQL command completed successfully.
db2inst1@db2v10:/tmp/db2move>
```

Figure 16 – Truncate table RELDATA

5.4 IMPORT Utility

The **IMPORT** utility, or **IMPORT** option of the **DB2MOVE** utility, is cross-platform compatible. The **IMPORT** utility inserts data from an external file with a supported file format into a table, hierarchy, view or nickname. The **LOAD** utility is a faster alternative, but it does not support loading data at the hierarchy level.

You can improve the performance of the **IMPORT** utility if the data is on separate drives from the log files and tablespace containers. You can reduce the impact to other applications by importing individual tables or doing hierarchical imports when system utilization is lower. The following two parameters impact the performance of this utility.

5.4.1 ALLOW NO|WRITE ACCESS

ALLOW NO ACCESS is the default for **IMPORT** operations. **DB2** will acquire an exclusive (X) lock on the table before any rows are inserted. In this case, the table is considered to be offline to other applications and they will not be able to access the table or its contents. This allows for the fastest **IMPORT** operation.

Specifying **WRITE** causes the **IMPORT** to run in online mode, allowing other applications to access the table. There are several **IMPORT** options which are not compatible with online mode. These are **REPLACE**, **CREATE**, or **REPLACE_CREATE**.

Additionally, online mode is not compatible with buffered inserts.

5.4.2 COMMITCOUNT n AUTOMATIC

The **COMMITCOUNT** parameter specifies how frequently the data is to be written to disk. **IMPORT** performs a **COMMIT** after every **n** records are imported if a number is provided. When **AUTOMATIC** is specified, **IMPORT** determines when a commit needs to be performed. The **IMPORT** utility will automatically commit for either one of two reasons:

- to avoid running out of active log space
- to avoid lock escalation from row level to table level

If the `ALLOW WRITE ACCESS` option is specified and the `COMMITCOUNT` option is not specified, the `IMPORT` utility will perform commits as if `COMMITCOUNT AUTOMATIC` had been specified.

In order to meet our SLA we need to keep the database online. Import the data back into the `reldata` table using the following commands and performance options. The `IMPORT` command should be entered **on a single line**.

```
db2 connect to xmldb
db2 import from tab1.ixf of ixf allow write access commitcount 50 messages
import.msg insert into reldata
db2 "select count(*) from reldata"
db2 connect reset
```



```
db2inst1@db2v10:/tmp/db2move
File Edit View Terminal Help
db2inst1@db2v10:/tmp/db2move> db2 import from tab1.ixf of ixf allow write access
commitcount 50 messages import.msg insert into reldata

Number of rows read      = 1107
Number of rows skipped   = 0
Number of rows inserted  = 1107
Number of rows updated   = 0
Number of rows rejected  = 0
Number of rows committed = 1107

db2inst1@db2v10:/tmp/db2move> db2 "select count(*) from reldata"

1
-----
1107

1 record(s) selected.

db2inst1@db2v10:/tmp/db2move> db2 connect reset
DB20000I The SQL command completed successfully.
db2inst1@db2v10:/tmp/db2move>
```

Figure 17 – Importing data back into table RELDATA

There should be a message for each `COMMIT` the `IMPORT` utility made after each 50 rows, and when the operation ended.

```
less import.msg
```

Press 'q' on your keyboard to quit viewing `import.msg`.

For reference, notice that the `DB2MOVE` command syntax is much simpler than the `IMPORT` command. `DB2MOVE` will use the information in file `db2move.lst` to determine which files and tables to access.

i Note: DO NOT EXECUTE THIS COMMAND

```
db2move xmldb import -io insert
```

5.5 INGEST Utility

The ingest utility (sometimes referred to as continuous data ingest, or CDI) is a high-speed client-side DB2® utility that streams data from files and pipes into DB2 target tables. Because the ingest utility can move large amounts of real-time data without locking the target table, you do not need to choose between the data currency and availability.

The ingest utility ingests pre-processed data directly or from files output by ETL tools or other means. It can run continually and thus it can process a continuous data stream through pipes. The data is ingested at speeds that are high enough to populate even large databases in partitioned database environments.

An `INGEST` command updates the target table with low latency in a single step. The ingest utility uses row locking, so it has minimal interference with other user activities on the same table.

With this utility, you can perform DML operations on a table using an SQL-like interface without locking the target table. These ingest operations support the following SQL statements: `INSERT`, `UPDATE`, `MERGE`, `REPLACE`, and `DELETE`. The ingest utility also supports the use of SQL expressions to build individual column values from more than one data field.

Other important features of the ingest utility include:

- **Commit by time or number of rows**
- **Support for copying rejected records to a file or table, or discarding them**
- **Support for restart and recovery**

The `INGEST` command supports the following input data formats:

- Delimited text
- Positional text and binary
- Columns in various orders and formats

A single `INGEST` command goes through three major phases:

- 1. Transport**
- 2. Format**
- 3. Flush**

During this exercise we will just replace data into a table using `import`, `ingest` and `load`, each one several times to see and compare execution times.

Depending on hardware infrastructure, you may reach to the conclusion that the fastest one is `LOAD`. However, `INGEST` not only is as fast as `IMPORT`, but also has the option to automatically restart from the point of cancellation if it occurs. If we disable this functionality in `INGEST`, it will run twice as fast as `IMPORT`. Furthermore, `INGEST` can input data from several files at the same time, or from pipes, etc.

We first need to create the table that `INGEST` tool uses for restart information, and the table that we will use for our example, to input data:

Open a terminal window and execute the following commands:

```
cd ~/Documents/LabScripts/db2pt/utilities  
./compare_times.sh
```

You will see output similar to the results below (result times may vary depending on the physical machine where VMware resides). Notice the execution time for each operation:

```
Database Connection Information  
Database server      = DB2/LINUX 10.1.0  
SQL authorization ID = DB2INST1  
Local database alias = XMLDB  
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.  
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.  
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.  
SQL3418W The NOCHARDEL file type modifier should not be specified if the data was exported  
using DB2. It is provided to support vendor data files that do not have character delimiters.
```

STARTING LOAD

SQL3109N The utility is beginning to load data from file

"/home/db2inst1/Documents/LabScripts/db2pt/utilities/ingest_data_200k".

SQL3500W The utility is beginning the "LOAD" phase at time "03/23/2012

13:57:07.219370".

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

SQL3520W Load Consistency Point was successful.

SQL3110N The utility has completed processing. "200000" rows were read from the input file.

SQL3519W Begin Load Consistency Point. Input record count = "200000".

SQL3520W Load Consistency Point was successful.

SQL3515W The utility has finished the "LOAD" phase at time "03/23/2012

13:57:11.797938".

Number of rows read = 200000

Number of rows skipped = 0

Number of rows loaded = 200000

Number of rows rejected = 0

Number of rows deleted = 0

Number of rows committed = 200000

real 0m2.632s

user 0m0.004s

sys 0m0.008s

DB20000I The SQL command completed successfully.

STARTING INGEST UTILITY

SQL2979I The ingest utility is starting at "03/23/2012 13:57:14.571600".

SQL2914I The ingest utility has started the following ingest job:

"DB21001:20120323.135714.571600:00003:00006".

Number of rows read = 200000

Number of rows inserted = 200000

Number of rows rejected = 0

SQL2980I The ingest utility completed successfully at timestamp "03/23/2012

13:57:28.934178"

real 0m14.512s

user 0m0.004s

sys 0m0.008s

DB20000I The SQL command completed successfully.

SQL2979I The ingest utility is starting at "03/23/2012 13:57:29.156538".

SQL2914I The ingest utility has started the following ingest job:

"DB21001:20120323.135729.156538:00003:00006".

Number of rows read = 200000

Number of rows inserted = 200000

Number of rows rejected = 0

SQL2980I The ingest utility completed successfully at timestamp "03/23/2012

13:57:40.456536"

real 0m11.531s

user 0m0.008s

sys 0m0.008s

DB20000I The SQL command completed successfully.

SQL3109N The utility is beginning to load data from file "ingest_data_200k".

SQL3418W The NOCHARDEL file type modifier should not be specified if the data
was exported using DB2. It is provided to support vendor data files that do

```
not have character delimiters.

SQL3110N The utility has completed processing. "200000" rows were read from
the input file.

SQL3221W ...Begin COMMIT WORK. Input Record Count = "200000".

SQL3222W ...COMMIT of any database changes was successful.

SQL3149N "200000" rows were processed from the input file. "200000" rows
were successfully inserted into the table. "0" rows were rejected.

Number of rows read      = 200000
Number of rows skipped   = 0
Number of rows inserted  = 200000
Number of rows updated   = 0
Number of rows rejected  = 0
Number of rows committed = 200000
real    0m19.712s
user    0m0.004s
sys     0m0.008s
DB20000I The SQL command completed successfully.
```

Figure 18 – IMPORT/INGEST OUTPUT

As you can see, the execution time of `IMPORT` and `INGEST` with restart capability is almost the same, but `INGEST` provides the functionality of managing jobs. Also, notice that `INGEST` execution time is about HALF of `IMPORT` execution time, without the limitations of the `LOAD` command.

5.6 LOAD utility

The `LOAD` utility is best suited to situations where performance is your primary concern. It is much faster than the `IMPORT` utility because it writes formatted pages directly into the database rather than using SQL `INSERTS`. The load utility also allows you the option of not logging the transactions. Load operations can fully exploit resources such as memory and multiple processors.

The `LOAD`, like the `EXPORT` and `IMPORT` utilities, is cross-platform compatible. This utility can be used as an alternative to the `IMPORT` utility. Use the `LOAD` utility if you want to move large quantities of data into empty tables or tables that already contain data.

The following parameters impact the LOAD utility's performance.

- § **SAVECOUNT *n***
- § **DATA BUFFER *buffer – size***
- § **SORT BUFFER *buffer - size***
- § **CPU_PARALLELISM *n***
- § **DISK_PARALLELISM**
- § **FETCH_PARALLELISM**
- § **INDEXING MODE**
- § **REBUILD**
- § **INCREMENTAL**
- § **AUTOSELECT**
- § **DEFERRED**
- § **ALLOW NO/READ ACCESS**
- § **LOCK WITH FORCE**

If a value is not specified for DATA BUFFER, CPU_PARALLELISM, or DISK_PARALLELISM, an intelligent default is calculated by the utility at run time.

As shown in Figure 18, the record order in the source data can be preserved when CPU_PARALLELISM is set to allow multiple processes or threads.



Figure 19 – LOAD source data order preserved

During the development of this lab we were not able to capture much information with the `LOAD QUERY` command because our table is small and the `LOAD` completes very quickly. You can try if you like with the following loop, or you can skip this set of commands. If you want to try, open a new terminal session and enter these commands to monitor the progress of the `LOAD` operation. Press Enter after each line.

```
db2 connect to xmldb
while true
do
sleep 2
echo -e "\n### New Load Query ###"
db2 load query table reldata | tee -a loadQuery.txt
done
```

Switch to an unused terminal session. Delete the data in the table and verify the records have been deleted with the following commands.

```
db2 connect to xmldb
db2 truncate table reldata reuse storage immediate
db2 "select count(*) from reldata"
```

Start the `LOAD` and verify the records were loaded with the following commands. The `LOAD` command should be entered on a single line.

```
cd /tmp/db2move
db2 load from tabl.ixf of ixf savecount 50 messages load.msg insert into
reldata cpu_parallelism 2
db2 "select count(*) from reldata"
db2 connect reset
```

If you tried to monitor the `LOAD` progress with the loop, switch back to that terminal session and press `Ctrl+C` to cancel the loop. You can review the output of the command in file `loadQuery.txt`. Note: you may have to repeat the load a couple of times for the query to show results since it is such a small load.

6 Further maintenance utility discussions

DB2 provides other utilities for moving data. These include stored procedures such as `ADMIN_COPY_SCHEMA`, `ADMIN_MOVE_TABLE` and `ADMIN_MOVE_TABLE_UTIL`, DB2 system commands such as `db2relocatedb` and several administrative APIs.

If you require 24x7 availability then you will probably implement a form of high availability for your database. It is possible to eliminate some maintenance activity against a primary production server by applying maintenance to the standby, or backup server.

Many companies and organizations have established Service Level Agreements or SLAs. These formalized requirements will have a significant bearing on how you will configure your DB2 system and commands, often requiring you to trade availability for performance.

7 Cleanup

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

```
db2 connect reset
db2 force application all
db2 terminate
db2 deactivate db xmldb
db2 drop db xmldb
db2stop
rm -rf ~/sqllib/db2dump/*
rm -rf /backup/*
rm -rf /archive/*
rm -rf /logs/*
rm -rf /tmp/restore/*
rm -rf /tmp/db2move/*
```



© 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.