

Course Guide

DB2 10.5 Administration Workshop for Windows

Course code: CL235 ERC 2.0



March 2016 edition**NOTICES**

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

TRADEMARKS

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, and the Adobe logo, are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.



UNIX is a registered trademark of The Open Group in the United States and other countries.


© Copyright International Business Machines Corporation 2016.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Unit 3 The DB2 database manager instance





The DB2 database manager instance

DB2 10.5 Administration Workshop for Windows

© Copyright IBM Corporation 2016
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Demonstration 1

Create a new DB2 instance

- Run the **db2icrt** command to create a new DB2 database manager instance.
- Issue **db2set** and **db2 update** commands to configure the DB2 instance.
- Utilize the **db2start** and **db2stop** commands to start and stop a DB2 instance.
- Run **db2pd** commands to check the DB2 instance configuration and status.

Demonstration 1: Create a new DB2 instance

Demonstration 1: Create a new DB2 instance

Purpose:

This demonstration will create a new DB2 instance that will support the database that we will create and manage in the following exercises. We will use the DB2 command line processor to create the new instance, configure some of the instance level options and issue commands to start and stop the instance.

Task 1. Logon to the Windows system and create the DB2 instance, **inst23**

In this exercise we will create a new DB2 instance with a name of **inst23**. The Windows systems used for the lab exercise has a predefined user named **inst23**.

On Windows systems, DB2 does not require an *'instance owner'*, but we will use the system logon user name **inst23** to create and manage the DB2 instance and database for the lab exercises.

1. Logon to the Windows system using the user id **inst23**, with a password of **ibm2blue**.

The DB2 command line processor will be used to issue the DB2 and system commands. You will need to start the DB2 command window in administrator mode to issue these DB2 commands.

2. To start the DB2 command window, click the Windows Start button, and then Navigate to **All programs > IBM DB2 DB2COPY1 > DB2 Command Window - Administrator**. When prompted with the question, 'Do you want to allow the following program to make changes to this computer?' select **Yes**.

When the DB2 software was installed the option to create a DB2 instance named **DB2** was chosen. This instance was used to create a sample database named **SAMPLE**.

The IBM Data Server Manager software can be configured to store data into a DB2 database for monitoring over a period of time. A second database named **DSMDATA** was created and configured to support the DSM tool.

3. Issue the following series of commands using the DB2 command line processor.
 - db2 get instance
 - db2 list db directory

The current instance should be DB2. The two databases - SAMPLE and DSMDATA - should be listed.

We will now create a new DB2 Database Manager Instance named **inst23**, using the **db2icrt** command to create the instance. We can use the Windows **SET** command to change the current instance variable to the new instance. Use the **db2set** command to change the default instance to **inst23**.

Changing the default instance registry variable, **db2instdef**, will direct the command activity to this default instance by default. You can always change the current DB2 instance for the command window using the SET command to alter the **DB2INSTANCE** variable.

4. Issue the following series of commands using the DB2 command line processor.
 - db2icrt inst23
 - set db2instance=inst23
 - db2set db2instdef=inst23
 - db2set -all

Task 2. Configure the new DB2 instance, inst23, to allow tcpip application connections and define a specific diagnostic data path.

The new DB2 instance has a DBM configuration with default settings.

We need to enable tcpip application connections to the new DB2 instance. We will need to assign a tcpip port number, for application connections to the instance, using the **SVCENAME** option in the DBM configuration.

We will assign the port number **50230**, which cannot be used by any other DB2 instance or other application on the same server.

The **db2set** command will be used to set the DB2 registry variable DB2COMM to 'tcpip'.

1. Issue the following series of commands using the DB2 command line processor.
 - db2 get instance
 - db2set db2comm=tcpip
 - db2set -all
 - db2 update dbm cfg using svcename 50230

We need to issue the db2start command to activate the new DB2 instance. The db2pd command with the **-edus** option can be used to list the active processes and threads for a DB2 instance.

2. Issue the following series of commands using the DB2 command line processor.
 - db2start
 - db2pd -edus

The output from this command will look similar to the following:

```
Database Member 0 -- Active -- Up 0 days 00:01:31 -- Date 2015-08-03-
15.15.45.266000
```

```
List of all EDUs for database member 0
```

```
db2sysc PID: 5620
```

```
db2acd PID: 1252
```

EDU ID	TID	EDU Name
=====		
0	0	db2pdbe
6188	6188	db2spmlw
6548	6548	db2spmrsy
4936	4936	db2resync
7136	7136	db2tcpcm
4220	4220	db2ipccm
6848	6848	db2wlmtm
3980	3980	db2wlmt
4652	4652	db2licc
7012	7012	db2aiothr
5620	5620	db2sysc

We can set a specific disk location to store the diagnostic files generated by the new instance using the **DIAGPATH** configuration option. We can limit the amount of diagnostic data using a series of rotating diagnostic log files using the **DIAGSIZE** configuration option. We will need to restart the DB2 instance to implement these changes. The db2pd command can be used to verify the configuration settings.

3. Issue the following series of commands using the DB2 command line processor.
 - md c:\inst23\diag
 - db2 update dbm cfg using diagpath c:\inst23\diag diagsize 20
 - db2stop force
 - db2start
 - db2pd -dbmcfg | more

The output from this command will look similar to the following:

```
Database Member 0 -- Active -- Up 0 days 00:00:40 -- Date 2015-08-03-
15.28.55.657000
```

Database Manager Configuration Settings:

Description	Memory Value	Disk Value
RELEASE	0x1000	0x1000
CPUSPEED(millisec/instruction)	2.519169e-007	2.519169e-007
NUMDB	32	32
NUMDB_INT	NEEDS RECOMPUTE(32)	NEEDS RECOMPUTE(32)
FEDERATED	NO	NO

TP_MON_NAME		
DFT_ACCOUNT_STR		
JDK_PATH (memory)	C:\PROGRA~1\IBM\SQLLIB\java\jdk	
JDK_PATH (disk)	C:\PROGRA~1\IBM\SQLLIB\java\jdk	
DIAGLEVEL	3	3
NOTIFYLEVEL	3	3
DIAGPATH (memory)	c:\inst23\diag\	
DIAGPATH (disk)	c:\inst23\diag\	
DIAGPATH_RESOLVED (memory)	c:\inst23\diag\	
DIAGPATH_RESOLVED (disk)	c:\inst23\diag\	
ALT_DIAGPATH (memory)		
ALT_DIAGPATH (disk)		
ALT_DIAGPATH_RESOLVED (memory)		
ALT_DIAGPATH_RESOLVED (disk)		
DIAGSIZE (MB)	20	20
DFT_MON_BUFPOOL	OFF	OFF
DFT_MON_LOCK	OFF	OFF
DFT_MON_SORT	OFF	OFF
DFT_MON_STMT	OFF	OFF
DFT_MON_TABLE	OFF	OFF
DFT_MON_TIMESTAMP	ON	ON
DFT_MON_UOW	OFF	OFF
HEALTH_MON	OFF	OFF
SYSADM_GROUP (memory)		
SYSADM_GROUP (disk)		
SYSCTRL_GROUP (memory)		
SYSCTRL_GROUP (disk)		
SYSMAINT_GROUP (memory)		
SYSMAINT_GROUP (disk)		
SYSMON_GROUP (memory)		
SYSMON_GROUP (disk)		
CLNT_PW_PLUGIN		
CLNT_KRB_PLUGIN	IBMkrb5	IBMkrb5
GROUP_PLUGIN		
LOCAL_GSSPLUGIN		
SRV_PLUGIN_MODE	UNFENCED	UNFENCED

SRVCON_GSSPLUGIN_LIST		
SRVCON_PW_PLUGIN		
SRVCON_AUTH		
AUTHENTICATION	SERVER	SERVER
ALTERNATE_AUTH_ENC		
CATALOG_NOAUTH	NO	NO
TRUST_ALLCLNTS	YES	YES
TRUST_CLNTAUTH	CLIENT	CLIENT
FED_NOAUTH	NO	NO
DFTDBPATH (memory)	C:	
DFTDBPATH (disk)	C:	
MON_HEAP_SZ (4KB)	AUTOMATIC(66)	AUTOMATIC(66)
JAVA_HEAP_SZ (4KB)	2048	2048
AUDIT_BUF_SZ (4KB)	0	0
INSTANCE_MEMORY (4KB)	AUTOMATIC(838861)	AUTOMATIC(838861)
RSTRT_LIGHT_MEM (4KB)	AUTOMATIC(10)	AUTOMATIC(10)
RSTRT_LIGHT_MEM_INT (4KB)	NEEDS RECOMPUTE(0)	NEEDS RECOMPUTE(0)
AGENT_STACK_SZ	16	16
BACKBUFSZ (4KB)	1024	1024
RESTBUFSZ (4KB)	1024	1024
SHEAPTHRES (4KB)	0	0
DIR_CACHE	YES	YES
ASLHEAPSZ (4KB)	15	15
RQRIOBLK (bytes)	65535	65535
UTIL_IMPACT_LIM	10	10
AGENTPRI	SYSTEM	SYSTEM
NUM_POOLAGENTS	AUTOMATIC(100)	AUTOMATIC(100)
NUM_INITAGENTS	0	0
MAX_COORDAGENTS	AUTOMATIC(200)	AUTOMATIC(200)
MAX_CONNECTIONS	AUTOMATIC(MAX_COORDAGENTS)	
	AUTOMATIC(MAX_COORDAGENTS)	
KEEPFENCED	YES	YES
FENCED_POOL	AUTOMATIC(MAX_COORDAGENTS)	
	AUTOMATIC(MAX_COORDAGENTS)	
NUM_INITFENCED	0	0
INDEXREC	RESTART	RESTART
TM_DATABASE	1ST_CONN	1ST_CONN

RESYNC_INTERVAL (secs)	180	180
SPM_NAME	IBMCLAS1	IBMCLAS1
SPM_LOG_FILE_SZ	256	256
SPM_MAX_RESYNC	20	20
SPM_LOG_PATH		
SVCENAME	50230	50230
DISCOVER	SEARCH	SEARCH
DISCOVER_INST	ENABLE	ENABLE
SSL_SVR_KEYDB (memory)		
SSL_SVR_KEYDB (disk)		
SSL_SVR_STASH (memory)		
SSL_SVR_STASH (disk)		
SSL_SVR_LABEL (memory)		
SSL_SVR_LABEL (disk)		
SSL_SVCENAME		
SSL_CIPHERSPECS (memory)		
SSL_CIPHERSPECS (disk)		
SSL_VERSIONS (memory)		
SSL_VERSIONS (disk)		
SSL_CLNT_KEYDB (memory)		
SSL_CLNT_KEYDB (disk)		
SSL_CLNT_STASH (memory)		
SSL_CLNT_STASH (disk)		
MAX_QUERYDEGREE	ANY	ANY
INTRA_PARALLEL	NO	NO
FCM_NUM_BUFFERS (4KB)	AUTOMATIC(1024)	AUTOMATIC(1024)
FCM_NUM_CHANNELS	AUTOMATIC(512)	AUTOMATIC(512)
FCM_PARALLELISM	1	1
FCM_PARALLELISM_INT	1	1
WLM_DISPATCHER	NO	NO
WLM_DISP_CONCUR	COMPUTED(4)	COMPUTED
WLM_DISP_CPU_SHARES	NO	NO
WLM_DISP_MIN_UTIL	5	5
KCFD_CFG_SIGNATURE	5	5
COMM_EXIT_LIST (memory)		
COMM_EXIT_LIST (disk)		

```

KEYSTORE_TYPE                NONE                NONE
KEYSTORE_LOCATION (memory)
KEYSTORE_LOCATION (disk)

```

The database manager configuration contains settings and status information for the DB2 instance. The DB2 command GET DBM CFG can be used to list this information. Check the current default database path, DFTDBPATH setting to see the default database path that would be used if a new database is created.

4. Issue the following command using the DB2 command line processor.

- `db2 get dbm cfg | find "path"`

The output from this command will look similar to the following:

```

Java Development Kit installation path      (JDK_PATH) =
C:\PROGRA~1\IBM\SQLLIB\java\jdk
Diagnostic data directory path              (DIAGPATH) = c:\inst23\diag\
Alternate diagnostic data directory path (ALT_DIAGPATH) =
Default database path                      (DFTDBPATH) = C:
SPM log path                              (SPM_LOG_PATH) =

```

Now that we have created and configured the new DB2 instance, **inst23**, we will use that instance to create a new database.

Results:



You created a new DB2 instance that will support the database that we will create and manage in the following exercises. You used the DB2 command line processor to create the new instance, configured some of the instance level options and issued commands to start and stop the instance.


Unit summary

- Specify the key features of an instance
- Create and drop a DB2 instance
- Use db2start and db2stop commands to manage a DB2 instance
- Display and set DB2 registry variables
- Describe and modify the database manager configuration

Units summary

Unit 4 Creating databases and data placement





Creating databases and data placement

DB2 10.5 Administration Workshop for Windows

© Copyright IBM Corporation 2016
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Demonstration 1

Creating databases and data placement

- Create a new DB2 database.
- Change some of the default database configuration options.
- Create a new Storage Group to support application storage.
- Create a set of tablespaces to support the database objects we plan to create.
- Use SQL queries and db2pd commands to review the options and disk storage associated with the new database.

Demonstration 1: Creating databases and data placement

Demonstration 1: Creating databases and data placement

Purpose:

This demonstration will create a new DB2 database named **MUSICDB** that will support the database objects that we will create and manage in the following exercises. You will create a new storage group and use DB2 commands and SQL queries to investigate database storage.

Task 1. Logon to the Windows system and create the DB2 database, **MUSICDB**.

In this exercise we will create a new DB2 database with a name of **MUSICDB**. The course uses a set of sample tables, so we will create a series of tablespaces to provide the storage for those tables. The database will be created in the instance, **inst23**, created in the first lab exercise.

In most cases the lab guide will include the step-by-step instructions to perform each step using the IBM Data Server Manager tool. You will also have the option to complete some tasks using a DB2 command line processor. Some commands can only be executed using a DB2 command line processor.

1. Logon to the Windows system using the user id **inst23**, with a password of **ibm2blue**.

The DB2 command line processor will be used to issue some DB2 and system commands. You will need to start the DB2 command window in administrator mode to issue these DB2 commands.

2. To start the DB2 command window, click the Windows Start button, then Navigate to **All programs > IBM DB2 DB2COPY1 > DB2 Command Window - Administrator**. When prompted with the question, 'Do you want to allow the following program to make changes to this computer?' select **Yes**.

A set of course files are located in the directory **c:\inst23\ddl**. Change to this directory to make it easier to access these files that contain DB2 commands or SQL statements. Check the current DB2 instance.

3. Issue the following series of commands using the DB2 command line processor:

- **cd c:\inst23\ddl**
- **db2 get instance**

The current instance should be **INST23**.

At this point, you may choose to create the database using the CREATE DATABASE command, using either the DB2 command line processor or the Data Server Manager tool. Follow the steps for your chosen tool only.

1A. Use the DB2 command line processor.

1. Issue the following series of command using the DB2 command line processor.
 - **db2 create database musicdb on c:** (may take several seconds/minutes)
 - **db2 connect to musicdb**
2. You can now skip to **Task 2**.

1B. Use the Data Server Manager tool.

The IBM Data Server Manager tool, referred to as DSM, was installed on the class system to provide a simple graphical interface to manage the DB2 database server. Access to DSM is through a Web Browser. When the software is installed, a tcpip port number was associated with the DSM service, **11080**. Use the Microsoft Internet Explorer browser to start the DSM application.

A shortcut to the **Data Server Manager** may be available on the Windows desktop.

If no shortcut is provided, open the Internet Explorer Browser and type the following URL text to start DSM, **http://localhost:11080**.

When DSM was installed, a local user id and password were selected to be associated with the application. For this course the following were configured:

Windows user id: **db2admin**

User password: **ibm2blue**

1. Use this userid and password to logon to DSM.
2. Use DSM to Create the MUSICDB database
3. In order to create a new DB2 database, we need to select the DB2 instance, **inst23**. Click the **Administer** option on the left side of the DSM application, then select **Explore Databases**.
 - The first drop down list at the top should be set to **Host Port Instance**.
 - The next drop down list is labeled *Select a Database or an Instance*. You can select **localhost 30230 INST23 (Instance)**.

You should see buttons that could be used to start or stop this DB2 instance.

4. Click **Create a new database**.

You can now select a set of database options for the new database.

- In the section **DETAIL** enter the database name **MUSICDB**.
- Expand the options for the **STORAGE** section, click on **BROWSE** to enter a disk path to be used for automatic tablespace storage. Select the following disk directory, **C:**
- Expand the options in the **Locale** section and use the drop down list to select the *Country or Region* option, for example select **United States of America**.

5. Click **Next** at the bottom.

The generated CREATE DATABASE command text is shown at the bottom of the window.

6. Click **Finish** to begin processing the generated CREATE DATABASE command.

Wait for the database creation to complete. This may take several minutes to complete processing.

Task 2. Configure a database connection to the DB2 database MUSICDB from Data Server Manager.

We will create a database connection profile for the new database, MUSICDB to be used by the Data Server Manager tool (see previous task, for accessing DSM).

1. Click **Setup** on the left side of the DSM application, and then select **Database Connections**.
2. Click (+) **Add a Database Connection**.
3. In the **Add Database Connection** window, make the following entries, scrolling to access all fields, if required:
 - database connection name: **MUSICDB**
 - database name: **MUSICDB**
 - host name: **localhost**
 - port number: **50230**
 - user id : **inst23**
 - password: **ibm2blue**
 - **Leave other options with the default values**

4. Click **Test Connection** and wait for the response: 'The connection to MUSICDB was successful'. Click **OK**.
5. Click **OK** to complete the connection.

Task 3. Change default database configuration options for the new database MUSICDB.

The database creation sets configuration options to default values or automatically configured values. We can make adjustments to these values to better match the intended usage of the database.

At this point, you may choose to make database configuration changes using the UPDATE DB CFG command using the DB2 command line processor or the Data Server Manager tool. Follow the steps for your chosen tool only.

We will reconfigure the number of primary, (Initial) and secondary (Additional) database log files for the database.

3A. Use the DB2 command line processor.

1. Issue the following series of commands using the DB2 command line processor.
 - **db2 connect to musicdb**
 - **db2 update db cfg using logprimary 5 logsecond 10**

The response includes the warning message *SQL1363W*, indicating that one or more changes will not take effect until the database is restarted. In this case the LOGPRIMARY option cannot be changed dynamically.

- **db2 get db cfg show detail | more**
2. You can now skip to **Task 4**.

3B. Use the Data Server Manager tool.

We can use the DSM tool to review and make changes to the database configuration.

If you have not started the DSM application, start it now using the following URL text, **http://localhost:11080**. Use the following user and password for DSM.

When DSM was installed, a local user id and password were selected to be associated with the application. For this course the following were configured:

Windows user id: **db2admin**

User password: **ibm2blue**

Some database configuration changes can take effect immediately, while others will require the database to be restarted.

1. Click **Home** on the left side of the DSM application.

You should see the database MUSICDB listed with some basic statistics on resource usage including, CPU, I/O and Memory. The database name MUSICDB provides a drop down list to perform different tasks.

2. Select **Administer: Explore Database** from the drop-down list of the MUSICDB database.

You will be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of **ibm2blue**.

3. Click **Configuration parameters**.

4. Expand **Logging**.

5. Scroll down the list of logging related configuration options to locate *LOGPRIMARY* and *LOGSECOND*. Make the following changes by entering new values in the column *Pending Value*:

- For *LOGPRIMARY*: **5**
- For *LOGSECOND*: **10** ; clear the Immediate checkbox.

The DSM tool generates an UPDATE DB CFG command to make these changes. The deferred option is added to the command.

6. Click **Next**.
7. Select **Run with SQL editor** for the Run schedule, and then click **Finish**.

Wait for the command to be processed (this may take several seconds or minutes).

The result should show that the command processing succeeded.

Task 4. Create a new storage group to support application storage.

The CREATE DATABASE command created one storage group, IBMSTOGROUP. We will create a new storage group to support several of the automatic storage table spaces.

The file `c:\inst23\ddl\create_stogroup.ddl` contains the CREATE STOGROUP statement. The file contains the following statement text:

```
CREATE STOGROUP app_data ON 'C:\dbauto\path1','C:\dbauto\path2';
```

At this point, you may choose to create the storage group using the DB2 command line processor or the Data Server Manager tool. Follow the steps for your chosen tool only.

4A. Use the DB2 command line processor.

1. Issue the following series of commands using the DB2 command line processor.
 - **cd c:\inst23\ddl**
 - **db2 connect to musicdb**
 - **db2 -tvf create_stogroup.ddl**

The db2pd command can be used to list the storage groups for an active database.

- **db2pd -db musicdb -storage**

The output from this command will look similar to the following:

```
Database Member 0 -- Database MUSICDB -- Active -- Up 6 days 19:57:39 --
Date 2015-08-11-10.43.12.024000
```

Storage Group Configuration:

Address	SGID	Default	DataTag	Name
0x0000000001214820	0	Yes	0	IBMSTOGROUP
0x00000000021B7F000	1	No	0	APP_DATA

Storage Group Statistics:

Address	SGID	State	Numpaths	NumDropPen
0x0000000001214820	0	0x00000000	1	0
0x00000000021B7F000	1	0x00000000	2	0

Storage Group Paths:

Address	SGID	PathID	PathState	PathName
0x0000000001238000	0	0	InUse	C:
0x00000000021B82000	1	1024	NotInUse	C:\dbauto\path1
0x00000000021B84000	1	1025	NotInUse	c:\dbauto\path2

2. You can now skip to **Task 5**.

4B. Use the Data Server Manager tool.

We can use the DSM tool to create the new storage group.

1. Click the **Home** option on the left side of the DSM application.
You should see the database MUSICDB listed with some basic statistics on resource usage including, CPU, I/O and Memory. The database name MUSICDB provides a drop down list to perform different tasks.
2. Select **Administer: Explore Database** from the drop down list of the MUSICDB database.
You will be prompted for a userid and password for the MUSICDB database connection; use **inst23** with a password of **ibm2blue**.
A tree structure of database objects for this database is shown on the left side of the DSM view.
3. Click **Storage Groups**.
The storage group IBMSTOGROUP is the default storage group defined when the database was created.

4. Click **Create** to open a list of options to define a new storage group.
Enter the following values for the new storage group:

- For storage group name: **APP_DATA**
- For default storage group: **NO**
- For storage paths: **C:\dbauto\path1,C:\dbauto\path2**

The DSM tool generates the CREATE STOGROUP statement that will define the new storage group named APP_DATA, with two storage paths.

The command text should appear as follows:

```
CREATE STOGROUP "APP_DATA" ON 'C:\dbauto\path1','C:\dbauto\path2';
```

- Click **Next**.
5. Select **Run with SQL Editor** for the Run Schedule, and then click **Finish**.
 6. Wait for the command to be processed.
The result should show that the command processing succeeded.
 7. Close the **Create Storage Group** tab.

Task 5. Create a new tablespace to support application objects.

We will create a new table space that has the following attributes:

- Table space name is **TSP01**
- Table Space Type is **Large**
- Buffer Pool should be **IBMDEFAULTBP** (which is also the default)
- Table Space management is **Automatic_Storage**
- Storage group: **APP_DATA**
- Initial size should be **1MB** (256 **4 KB** pages)
- Container should be a **File**
- Table space extent size should be **4**
- AUTORESIZE should increase the size by **100 KB** when more disk storage is needed. No maximum size will be specified.

At this point, you may choose to create the table space **TSP01** using the DB2 command line processor or the Data Server Manager tool. Follow the steps for your chosen tool only.

5A. Use the DB2 command line processor.

The file `c:\inst23\ddl\create_tablespace_tsp01.ddl` contains the CREATE TABLESPACE statement. The file contains the following statement text:

```
CREATE TABLESPACE TSP01 using stogroup APP_DATA
INITIALSIZE 1M INCREASESIZE 100 K
EXTENTSIZE 4 ;
```

1. Issue the following series of commands using the DB2 command line processor.

- **cd c:\inst23\ddl**
- **db2 connect to musicdb**
- **db2 -tvf create_tablespace_tsp01.ddl**

The db2pd command can be used to list the tablespaces for an active database.

- **db2pd -db musicdb -tablespaces | more**

2. You can now skip to **Task 6**.

5B. Use the Data Server Manager tool.

We can use the DSM tool to create the new tablespace, **TSP01**.

1. Click **Home** on the left side of the DSM application.

You should see the database MUSICDB listed with some basic statistics on resource usage including, CPU, I/O and Memory. The database name MUSICDB provides a drop down list to perform different tasks.

2. Select **Administer: Explore Database** from the drop down list of the **MUSICDB** database.

You will be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of **ibm2blue**.

A tree structure of database objects for this database is shown on the left side of the DSM view.

3. Click **Table Spaces**. The system created table spaces will be listed.

4. Click **Create** to open a list of options to define a new storage group. Enter the following values for the new storage group:

- For table space name: **TSP01 (Use uppercase text)**
- Accept the default values for *Type of Data*, *Managed By*, *Buffer Pool* and *Database partition Group*.
- For storage group name: **APP_DATA**
- For extent size: **4**
- For initial size: **1 M**
- For increase size: **100 K**

The DSM tool generates the CREATE TABLESPACE statement that will define the new table space named TSP01.

The command text should be similar to the following:

```
CREATE TABLESPACE "TSP01" MANAGED BY AUTOMATIC STORAGE
USING STOGROUP APP_DATA AUTORESIZE YES
INITIALSIZE 1 M INCREASESIZE 100 K
EXTENTSIZE 4 ;
;
```

5. Click **Next**.
6. Select **Run with SQL Editor** for the Run Schedule and click on **Finish**.

Wait for the command to be processed.

The result should show that the command processing succeeded.

7. Close the **Create Table Space** tab.

If you refresh the table space list, you should see the new tablespace, TSP01 included in the list.

Task 6. Create a set of new table spaces using a SQL file.

We will create a group of table spaces using a SQL file containing the CREATE TABLESPACE statements.

The file *c:\inst23\ddl\create_tablespaces.ddl* contains five CREATE TABLESPACE statements.

The file contains the following statement text:

```
create tablespace tsp02
managed by database
using (file 'tsp02' 128)
extentsize 2 autoresize yes maxsize 2 M ;
```

```
create tablespace tsp03
managed by database
using (file 'tsp03' 1024)
extentsize 8 autoresize yes maxsize 10 M ;
```

```
create tablespace tsp04
managed by automatic storage using stogroup app_data
initialsize 100 K maxsize none
extentsize 2;
```

```
create tablespace tsp05 using stogroup app_data
initialsize 64 K maxsize 1 M
extentsize 2;
```

```
create regular tablespace tsp06
extentsize 4;
;
```

At this point, you may choose to create the table spaces using the DB2 command line processor or the Data Server Manager tool. Follow the steps for your chosen tool only.

6A. Use the DB2 command line processor.

1. Issue the following series of commands using the DB2 command line processor.

- **cd c:\inst23\ddl**
- **db2 connect to musicdb**
- **db2 -tvf create_tablespaces.ddl**

The LIST TABLESPACES command can be used to list the tablespaces for an active database.

- **db2 list tablespaces | more**

2. You can now skip to **Task 7**.

6B. Use the Data Server Manager tool.

We can use the DSM tool to execute the SQL file containing the CREATE TABLESPACE statements.

1. Click the **Home** option on the left side of the DSM application.
You should see the database MUSICDB listed. The database name MUSICDB provides a drop down list to perform different tasks.
2. Select **Administer: Explore Database** from the drop down list of the MUSICDB database.
You may be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of **ibm2blue**.
A tree structure of database objects for this database is shown on the left side of the DSM view.
3. Click **Open SQL editor**.
4. Click **Upload** (which may be part of a list labelled *More Actions*). Use the **Browse** button for the Open SQL Script window to locate and select the file *C:\inst23\ddl\create_tablespaces.ddl*. Click **Open**.
5. Click **OK** to complete loading of the SQL text into the SQL editor.
Review the options specified for the five CREATE TABLESPACE statements.
6. Click **Run**, and then wait for the five SQL statements to be processed.
The result should show that all five statements succeeded.

Task 7. Use DB2 commands and SQL queries to show table space related information.

We can use DB2 commands like **db2pd** and SQL queries to retrieve information about the table spaces in a DB2 database.

The file *dbpaths.sql* uses a DB2 system view SYSIBMADM.DBPATHS which shows the disk paths currently used by a database.

The DB2 system catalog views can be used to list information about data objects, like table spaces. The view SYSCAT.TABLESPACES contains table space attributes.

Use the file *select_tablespace.sql* to query information using this catalog view. The selected fields that contain table space information are:

- TBSPACE: Name of primary table space for this table.
- DEFINER: Authid of table space creator.
- TBSPACEID: Internal table space identifier.
- TBSPACETYPE: Type of table space. D for DMS or S for SMS.
- DATATYPE: Type of data that can be stored in the table space. L for large table spaces, A for Regular table spaces, or T for temporary table spaces.
- SGNAME: The storage group name for automatic storage table spaces.

Note: Using this view, Table spaces managed by Automatic Storage will appear as DMS for regular and large table spaces and SMS for temporary table spaces.

At this point, you may choose to execute the commands and SQL statements using the DB2 command line processor or the Data Server Manager tool. Follow the steps for your chosen tool only.

7A. Use the DB2 command line processor.

1. Issue the following series of commands using the DB2 command line processor.

- **cd c:\inst23\ddl**
- **db2 connect to musicdb**
- **db2 -tvf select_tablespaces.sql**

The output from this command will look similar to the following:

```
select substr(tbspacename,1,18) as tbspacename,
       substr(definer,1,10) as definer, tbspacename, tbpacetype, datatype, sgname
from syscat.tablespace
```

TBSPACE	DEFINER	TBSPACEID	TBSPACETYPE	DATATYPE	SGNAME
SYSCATSPACE	SYSIBM	0	D	A	IBMSTOGROUP
TEMPSPACE1	SYSIBM	1	S	T	IBMSTOGROUP
USERSPACE1	SYSIBM	2	D	L	IBMSTOGROUP
SYSTOOLSPACE	SYSTEM	3	D	L	IBMSTOGROUP
TSP01	INST23	4	D	L	APP_DATA
TSP02	INST23	5	D	L	-
TSP03	INST23	6	D	L	-
TSP04	INST23	7	D	L	APP_DATA
TSP05	INST23	8	D	L	APP_DATA
TSP06	INST23	9	D	A	IBMSTOGROUP

10 record(s) selected.

Notice that the DMS managed tablespaces do not have an assigned storage group. Each tablespace is assigned a unique tablespace ID by DB2.

- **db2 -tvf select_mon_get_cont.sql**

The output from this command will look similar to the following:

```
select varchar(container_name, 80) as container_name,
       varchar(tbsp_name, 20) as tbsp_name,
       pool_read_time
from table(mon_get_container('', -2)) as t order by tbsp_id
```

```
CONTAINER_NAME
Tbsp_NAME      POOL_READ_TIME
```

```
-----
```

```
C:\INST23\NODE0000\MUSICDB\T0000000\C0000000.CAT
SYSCATSPACE      1157
```

```
C:\INST23\NODE0000\MUSICDB\T0000001\C0000000.TMP
TEMPSPACE1      0
```

```
C:\INST23\NODE0000\MUSICDB\T0000002\C0000000.LRG
USERSPACE1      20
```

```
C:\INST23\NODE0000\MUSICDB\T0000003\C0000000.LRG
SYSTOOLSPACE    9
```

```
C:\DBAUTO\PATH1\INST23\NODE0000\MUSICDB\T0000004\C0000000.LRG
TSP01           10
```

```
C:\DBAUTO\PATH2\INST23\NODE0000\MUSICDB\T0000004\C0000001.LRG
TSP01           5
```

```
C:\INST23\NODE0000\SQL00001\tsp02
TSP02           4
```

```
C:\INST23\NODE0000\SQL00001\tsp03
TSP03           1
```

```
C:\DBAUTO\PATH1\INST23\NODE0000\MUSICDB\T0000007\C0000000.LRG
TSP04           1
```

```
C:\DBAUTO\PATH2\INST23\NODE0000\MUSICDB\T0000007\C0000001.LRG
TSP04           0
```

```
C:\DBAUTO\PATH1\INST23\NODE0000\MUSICDB\T0000008\C0000000.LRG
TSP05           0
```

```
C:\DBAUTO\PATH2\INST23\NODE0000\MUSICDB\T0000008\C0000001.LRG
TSP05           0
```

```
C:\INST23\NODE0000\MUSICDB\T00000009\C0000000.USR
TSP06                                0
```

```
13 record(s) selected.
```

Notice that the tablespaces using the APP_DATA storage group have two containers assigned because the storage group has two paths defined.

- **db2 -tvf dbpaths.sql**

The output from this command will look similar to the following:

```
select substr(type,1,30) as db_path_type,
substr(path,1,50) as path_name
from sysibmadm.dbpaths order by 1
```

DB_PATH_TYPE	PATH_NAME
DBPATH	C:\INST23\NODE0000\SQL00001\
DBPATH	C:\INST23\NODE0000\SQL00001\MEMBER0000\
DB_STORAGE_PATH	c:\dbauto\path2\
DB_STORAGE_PATH	C:\dbauto\path1\
DB_STORAGE_PATH	C:\
LOCAL_DB_DIRECTORY	C:\INST23\NODE0000\SQLDBDIR\
LOGPATH	C:\INST23\NODE0000\SQL00001\LOGSTREAM0000\
TBSP_CONTAINER	C:\INST23\NODE0000\SQL00001\tsp03
TBSP_CONTAINER	C:\INST23\NODE0000\SQL00001\tsp02

```
9 record(s) selected.
```

The result shows the disk locations that contain the components of the database, including the log files, database control files and tablespace storage.

- **db2pd -db musicdb -storage | more**

The output from this command will look similar to the following:

```
Database Member 0 -- Database MUSICDB -- Active -- Up 0 days 00:08:11 --
Date 2015-11-05-06.21.49.889000
```

Storage Group Configuration:

Address	SGID	Default	DataTag	Name
0x0000000012E00820	0	Yes	0	IBMSTOGROUP
0x0000000012E00940	1	No	0	APP_DATA

Storage Group Statistics:

Address	SGID	State	Numpaths	NumDropPen
0x0000000012E00820	0	0x00000000	1	0
0x0000000012E00940	1	0x00000000	2	0

Storage Group Paths:

Address	SGID	PathID	PathState	PathName
0x0000000012E24000	0	0	InUse	C:
0x0000000012E48000	1	1024	InUse	C:\DBAUTO\PATH1
0x0000000012E46000	1	1025	InUse	C:\DBAUTO\PATH2

The db2pd command report provides information about the two storage groups currently defined for the MUSICDB database.

- **db2pd -db musicdb -tablespaces | more**

The output from this command will look similar to the following:

```
Database Member 0 -- Database MUSICDB -- Active -- Up 0 days 00:08:11 --
Date 2015-11-05-06.21.49.889000
```

Tablespace Configuration:

Address	Id	Type	Content	PageSz	ExtentSz	Auto	Prefetch	BufID
BufID	Disk	FSC	NumCntrs	MaxStripe	LastConsecPg	RSE	Name	
0x00000000141D2F40	0	DMS	Regular	4096	4	Yes	4	1
Off 1	0	3	Yes	SYSCATSPACE				1
0x00000000141E00E0	1	SMS	SysTmp	4096	32	Yes	32	1
On 1	0	31	No	TEMPSPACE1				1
0x00000000141ED280	2	DMS	Large	4096	32	Yes	32	1
Off 1	0	31	Yes	USERSPACE1				1
0x00000000141FA420	3	DMS	Large	4096	4	Yes	4	1
Off 1	0	3	Yes	SYSTOOLSPACE				1
0x00000000142075C0	4	DMS	Large	4096	4	Yes	8	1
Off 2	0	3	Yes	TSP01				1

0x00000000142232C0	5	DMS	Large	4096	2	Yes	2	1	1
Off 1	0	1		Yes	TSP02				
0x0000000014230460	6	DMS	Large	4096	8	Yes	8	1	1
Off 1	0	7		Yes	TSP03				
0x000000001423D600	7	DMS	Large	4096	2	Yes	4	1	1
Off 2	0	1		Yes	TSP04				
0x000000001424A7A0	8	DMS	Large	4096	2	Yes	4	1	1
Off 2	0	1		Yes	TSP05				
0x0000000018070080	9	DMS	Regular	4096	4	Yes	4	1	1
Off 1	0	3		Yes	TSP06				

Tablespace Statistics:

Address	Id	TotalPgs	UsablePgs	UsedPgs	PndFreePgs
FreePgs	HWM	Max HWM	State	MinRecTime	NQuiescers
PathsDropped	TrackmodState				
0x00000000141D2F40	0	32768	32764	28220	0
28220	28220	0x00000000	0	0	No
					4544
0x00000000141E00E0	1	1	1	1	0
-	-	0x00000000	0	0	No
					0
0x00000000141ED280	2	8192	8160	96	0
96	96	0x00000000	0	0	No
					8064
0x00000000141FA420	3	8192	8188	152	0
152	152	0x00000000	0	0	No
					8036
0x00000000142075C0	4	256	248	12	0
12	12	0x00000000	0	0	No
					236
0x00000000142232C0	5	128	126	6	0
6	6	0x00000000	0	0	No
					120
0x0000000014230460	6	1024	1016	24	0
24	24	0x00000000	0	0	No
					992
0x000000001423D600	7	24	20	6	0
6	6	0x00000000	0	0	No
					14
0x000000001424A7A0	8	16	12	6	0
6	6	0x00000000	0	0	No
					6
0x0000000018070080	9	8192	8188	12	0
12	12	0x00000000	0	0	No
					8176
					n/a

Tablespace Autoresize Statistics:

Address	Id	AS	AR	InitSize	IncSize
IIP MaxSize		LastResize			LRF
0x00000000141D2F40	0	Yes	Yes	33554432	-1
No None		None			No
0x00000000141E00E0	1	Yes	No	0	0
No 0		None			No
0x00000000141ED280	2	Yes	Yes	33554432	-1
No None		None			No
0x00000000141FA420	3	Yes	Yes	33554432	-1
No None		None			No

0x00000000142075C0	4	Yes Yes	1048576	102400
No None		None		No
0x00000000142232C0	5	No Yes	-4096	-1
No 2097152		None		No
0x0000000014230460	6	No Yes	-4096	-1
No 10485760		None		No
0x000000001423D600	7	Yes Yes	102400	-1
No None		None		No
0x000000001424A7A0	8	Yes Yes	65536	-1
No 1048576		None		No
0x0000000018070080	9	Yes Yes	33554432	-1
No None		None		No

Tablespace Storage Statistics:

Address	Id	DataTag	Rebalance	SGID	SourceSGID
0x00000000141D2F40	0	0	No	0	-
0x00000000141E00E0	1	0	No	0	-
0x00000000141ED280	2	-1	No	0	-
0x00000000141FA420	3	-1	No	0	-
0x00000000142075C0	4	-1	No	1	-
0x00000000142232C0	5	0	No	-	-
0x0000000014230460	6	0	No	-	-
0x000000001423D600	7	-1	No	1	-
0x000000001424A7A0	8	-1	No	1	-
0x0000000018070080	9	-1	No	0	-

Containers:

Address	TspId	ContainNum	Type	TotalPgs	UseablePgs	PathID
StripeSet	Container					
0x00000000141CE960	0	0	File	32768	32764	0
0	C:\INST23\NODE0000\MUSICDB\T0000000\C0000000.CAT					
0x0000000014258000	1	0	Path	1	1	0
0	C:\INST23\NODE0000\MUSICDB\T0000001\C0000000.TMP					
0x00000000141C4640	2	0	File	8192	8160	0
0	C:\INST23\NODE0000\MUSICDB\T0000002\C0000000.LRG					
0x00000000141C4C80	3	0	File	8192	8188	0
0	C:\INST23\NODE0000\MUSICDB\T0000003\C0000000.LRG					
0x00000000141C5460	4	0	File	128	124	1024
0	C:\DBAUTO\PATH1\INST23\NODE0000\MUSICDB\T0000004\C0000000.LRG					
0x00000000141C5688	4	1	File	128	124	1025
0	C:\DBAUTO\PATH2\INST23\NODE0000\MUSICDB\T0000004\C0000001.LRG					
0x00000000141C5CE0	5	0	File	128	126	-
0	C:\INST23\NODE0000\SQL00001\tsp02					
0x00000000141C64C0	6	0	File	1024	1016	-
0	C:\INST23\NODE0000\SQL00001\tsp03					
0x00000000141C27A0	7	0	File	12	10	1024
0	C:\DBAUTO\PATH1\INST23\NODE0000\MUSICDB\T0000007\C0000000.LRG					

```

0x00000000141C29C8 7      1      File      12      10      1025
0      C:\DBAUTO\PATH2\INST23\NODE0000\MUSICDB\T0000007\C0000001.LRG
0x0000000012EDB7C0 8      0      File      8      6      1024
0      C:\DBAUTO\PATH1\INST23\NODE0000\MUSICDB\T0000008\C0000000.LRG
0x0000000012EDB9E8 8      1      File      8      6      1025
0      C:\DBAUTO\PATH2\INST23\NODE0000\MUSICDB\T0000008\C0000001.LRG
0x0000000012ED59A0 9      0      File      8192     8188     0
0      C:\INST23\NODE0000\MUSICDB\T0000009\C0000000.USR

```

The db2pd command report provides information about the tablespaces for the MUSICDB database, including disk space usage, container assignments and storage group usage.

2. You can now skip to the end of this exercise.

7B. Use the Data Server Manager tool.

We can use the DSM tool to execute the SQL file containing the SQL query.

1. Click the **Home** option on the left side of the DSM application.
You should see the database MUSICDB listed. The database name MUSICDB provides a drop down list to perform different tasks.

2. Select **Administer - Explore Database** from the drop down list of the MUSICDB database.

You will be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of *ibm2blue*.

A tree structure of database objects for this database is shown on the left side of the DSM view.

3. Click on **Open SQL Editor**.
4. Click on **Upload** (which may be part of a list labelled *More Actions*). Use the **Browse** button for the Open SQL Script window to locate and select the file **C:\inst23\ddl\select_tablespaces.sql**. Click on **OK** to complete loading the SQL text into the SQL editor.

Review the SQL query text.

5. Click **Run**, and then wait for the SQL statement to be processed.

The result should show that the SQL statement succeeded. The result also contains the query result in a columnar report.

Review the report. The result area contains an icon for *open* that will show the query results in its own view.

We will now utilize the SQL query text in the file

c:\inst23\ddl\select_mon_get_tbsp.sql to retrieve table space information and statistics.

6. Click **Open SQL Editor**.
7. Click on **Upload** (which may be part of a list labelled *More Actions*). Use the **Browse** button for the Open SQL Script window to locate and select the file ***select_mon_get_tbsp.sql***. Click on **OK** to complete loading the SQL text into the SQL editor.
Review the SQL query text.
8. Click **Run**, and then wait for the SQL statement to be processed.
The result should show that the SQL statement succeeded. The result also contains the query result in a columnar report.
9. Review the report. The result area contains an icon for *open* that will show the query results in its own view.
Notice that the two DMS managed table spaces, TSP02 and TSP03, do not have a storage group, since storage groups only apply to automatic storage managed table spaces.
We will now utilize the SQL query text in the file *c:\inst23\ddl\select_mon_get_cont.sql* to retrieve a list of disk containers for each table space in the database.
10. Click **Open SQL Editor**.
11. Click on **Upload** (which may be part of a list labelled *More Actions*). Use the **Browse** button for the Open SQL Script window to locate and select the file ***select_mon_get_cont.sql***. Click on **OK** to complete loading the SQL text into the SQL editor.
12. Review the SQL query text.
13. Click **Run**, and then wait for the SQL statement to be processed.
The result should show that the SQL statement succeeded. The result also contains the query result in a columnar report.
Review the report. The result area contains an icon for *open* that will show the query results in its own view.
Notice that the tablespaces using the APP_DATA storage group have two containers assigned because the storage group has two paths defined.
14. Use the SQL query text in the file *c:\inst23\ddl\dbpaths.sql* to retrieve information about the disk paths used to support components of this database.
15. Click **Open SQL Editor**.
16. Click **Upload** (which may be part of a list labelled *More Actions*). Use the **Browse** button for the Open SQL Script window to locate and select the file *c:\inst23\ddl\dbpaths.sql*. Click on **OK** to complete loading the SQL text into the SQL editor.


17. Review the SQL query text.
18. Click **Run**, and then wait for the SQL statement to be processed.
The result should show that the SQL statement succeeded. The result also contains the query result in a columnar report.
19. Review the report. The result area contains an icon for open that will show the query results in its own view.
The query result shows the disk locations that contain the components of the database, including the log files, database control files and tablespace storage.


Results:

You created a new DB2 database named MUSICDB that will support the database objects that we will create and manage in the following exercises. You created a new storage group and a set of table spaces. You used DB2 commands and SQL queries to investigate database storage.

Unit 5 Creating database objects

IBM Training





Creating database objects

DB2 10.5 Administration Workshop for Windows

© Copyright IBM Corporation 2016
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Demonstration 1

Creating database objects

- Create a set of tables using the Data Server Manager to build the CREATE TABLE statement or using saved DDL in a file.
- Create indexes for tables using the CREATE INDEX statement.
- Create views and alias objects using SQL statements in a file.
- Create foreign key and check constraints for a table using SQL statements in a file.
- Use the db2look utility to extract database object definitions from a DB2 database.

Demonstration 1: Creating database objects

Demonstration 1: Creating database objects

Purpose:

This demonstration will create a group of database objects in the DB2 database named MUSICDB.

Task 1. Logon to the Windows system and start a DB2 Command Line processor

1. Logon to the Windows system using the user id **inst23**, with a password of **ibm2blue**.

The DB2 command line processor will be used to issue some DB2 and system commands. You will need to start the DB2 command window in administrator mode to issue these DB2 commands.

2. To start the DB2 command window, click on the Windows start icon, then Navigate to: **All programs > IBM DB2 DB2COPY1 > DB2 Command Window - Administrator**. When prompted with the question, 'Do you want to allow the following program to make changes to this computer?' select **Yes**.

A set of course files are located in the directory **c:\inst23\ddl**. You will change to this directory to make it easier to access these files that contain DB2 commands or SQL statements.

Task 2. Create the ALBUMS table.

At this point, you may choose to create the ALBUMS table by using the DB2 command line processor or the Data Server Manager tool. Follow the steps for your chosen tool only.

2A. Use the DB2 command line processor.

Use the file **c:\inst23\ddl\create_table_albums.ddl** to create the ALBUMS table with a Primary key defined on the ITEMNO column.

The file contains the following statement text:

```
create table music.albums
(title      varchar (50),
artno      smallint not null,
itemno     smallint not null)
in tsp04
index in tsp05;

alter table music.albums primary key (itemno) ;
```


1. Issue the following series of commands using the DB2 command line processor:
 - **cd c:\inst23\ddl**
 - **db2 connect to musicdb**
 - **db2 -tvf create_table_albums.ddl**
 - **db2 describe table music.albums**
2. You can now skip to **Task 3**.

2B. Use the Data Server Manager tool.

We can use the Data Server Manager tool to create tables either by defining the table definition or by running a SQL file that contains the Data Definition Language statements.

If you have not started the DSM application, start it now using the following URL text, **http://localhost:11080**. Use the following user and password for DSM.

When DSM was installed, a local user id and password were selected to be associated with the application. For this course the following were configured:

Windows user id: **db2admin**

User password: **ibm2blue**

1. Use **DSM** to create a new table in the **MUSICDB** database.
2. In order to access the MUSICDB database, click the **Administer** option on the left side of the **DSM** application, then select **Explore Databases**.
3. The first drop down list at the top should be set to **Host Port Instance**.
4. The next drop down list is labeled *Select a Database or an Instance*. You can select **MUSICDB**.
5. When prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of *ibm2blue*.

A tree structure of database objects for this database is shown on the left side of the DSM view.

The tables we will create will use a single schema name of MUSIC. We will create the new schema named MUSIC.

6. Click **Schemas**.

The schemas created to support the new database will be listed.

7. Click **Create** to open a list of options to define a new schema. and then enter the following values for the new schema:

- For schema name: **MUSIC** (use upper case)
- Allow other options to take default values

The DSM tool generates the CREATE SCHEMA statement.

The command text should be the following:

```
CREATE SCHEMA "MUSIC" ;
```

8. Click **Next**.
9. Select **Run with SQL Editor** for the Run Schedule, and then click **Finish**.
The result should show that the command processing succeeded.
10. **Close** the **Create Schema** tab.

Now we can create a table, ALBUMS in the new schema, MUSIC.

11. Using the object tree structure, click **Tables**.
12. Click **Create** to open a list of options to define a new table.
13. Enter the following values for the **PROPERTIES** of the new table:
 - For name: **ALBUMS** (*enter the name using uppercase characters*)
 - For schema: **MUSIC**
 - For table space: **TSP04**
 - For index table space: **TSP05**
14. Click **Columns** to define the table columns.
15. Click **Add column** to define a column, using:
 - Name: **TITLE**
 - Data Type: **Varchar**
 - Length: **50**
16. Click **Columns** to return to the column list.
17. Click **Add column** to define a column, using:
 - Name: **ARTNO**
 - Data Type: **Smallint**
 - Nullable: **No**
18. Click **Columns** to return to the column list.

19. Click **Add column** to define a column, using:
 - Name: **ITEMNO**
 - Data Type: **Smallint**
 - Nullable: **No**
20. Click **Constraints**.
 - Click **Add constraint** to define a constraint, select **Primary key** as the constraint type, using:
 - Name: **ALBUM_ITEM**
21. Click **Add**, and then select the column **ITEMNO**.
The DSM tool generates the CREATE TABLE statement and the ALTER TABLE statement to add the primary key.
22. Click **Next**.
23. Select **Run with SQL Editor** for the Run Schedule, and then click **Finish**.
The result should show that the command processing succeeded.
24. **Close** the **Create table** tab.

Task 3. Create a set of new tables using a SQL file.

We will create a group of tables using a SQL file containing the CREATE TABLE statements.

At this point, you may choose to create the ALBUMS table by using the DB2 command line processor or the Data Server Manager tool. Follow the steps for your chosen tool only.

3A. Use the DB2 command line processor.

The file `c:\inst23\ddl\create_tables.ddl` contains five CREATE TABLE statements.

The file contains the following statement text:

```
create table MUSIC.concerts
(artno      smallint not null,
 date       date not null,
 city       varchar (25) not null with default)
in tsp04;

create table MUSIC.reorder
(itemno      smallint not null,
 timestamp   timestamp)
in TSP02;
```

```
create table MUSIC.artists
(artno      smallint not null,
 name       varchar(50),
 classification char(1) not null,
 bio        clob(100K) logged compact,
 picture    blob(500K) not logged compact,
 primary key (artno))
in tsp01
index in tsp02
long in tsp03 ;
```

```
create table MUSIC.stock
(ITEMNO SMALLINT NOT NULL ,
 TYPE CHAR(1) NOT NULL ,
 PRICE DECIMAL(5,2) NOT NULL WITH DEFAULT ,
 QTY INTEGER NOT NULL WITH DEFAULT,
 SYS_START TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN
IMPLICITLY HIDDEN,
 SYS_END TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END
IMPLICITLY HIDDEN,
 TX_START TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION
START ID IMPLICITLY HIDDEN,
 PERIOD SYSTEM_TIME (SYS_START,SYS_END) )
in tsp06;
```

```
CREATE TABLE MUSIC.STOCK_HISTORY LIKE MUSIC.STOCK IN tsp06 ;
```

```
ALTER TABLE MUSIC.STOCK ADD VERSIONING USE HISTORY TABLE
MUSIC.STOCK_HISTORY ;
```

1. Issue the following series of commands using the DB2 command line processor.
 - **cd C:\inst23\ddl**
 - **db2 connect to musicdb**
 - **db2 -tvf create_tables.ddl | more**
 - **db2 list tables for schema music**
2. You can now skip to **Task 4**.

3B. Use the Data Server Manager tool.

We can use the DSM tool to execute the SQL file containing the CREATE TABLE statements.

1. Click the **Home** option on the left side of the DSM application.
You should see the database MUSICDB listed. The database name MUSICDB provides a drop down list to perform different tasks.
2. Select **Administer - Explore Database** from the drop down list of the **MUSICDB** database.
You will be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of **ibm2blue**.
A tree structure of database objects for this database is shown on the left side of the DSM view.
3. Click **Open SQL Editor**.
4. Click **Upload** (which may be part of a list labelled *More Actions*). Use the Browse button for the Open SQL Script window to locate and select the file `C:\inst23\ddl\create_tables.ddl`.
5. Click **OK** to complete loading the SQL text into the SQL editor.
Review the options specified for the five CREATE TABLE statements.
The table MUSIC.STOCK will be defined as a System-period temporal table.
The table MUSIC.STOCK_HISTORY will be used as the history table for MUSIC.STOCK.
Notice the assignments of tables to table spaces.
For DMS and Automatic Storage Managed table spaces, multiple table spaces can be used for the data, index and large object components of a single table.
Which table space(s) will be used for data, indexes and large objects for the table STOCK? For CONCERTS? For ARTISTS?
6. Click **Run** and wait for the SQL statements to be processed.
The result should show that all of the statements succeeded.

Task 4. Create index, view and alias objects for the application tables.

Next we will create an index on the STOCK table based on the column ITEMNO.

We will also use the SQL statements in the file *create_View_Alias.ddl* to create view and alias objects.

The file *c:\inst23\ddl\create_stock_ix.ddl* contains the CREATE INDEX statement.

The file contains the following statement text:

```
create index music.stockitem_ix on music.stock(itemno) ;
```

The file *c:\inst23\ddl\create_VIEW_ALIAS.ddl* contains the following statements:

```
create view music.music as select title, classification, name
from music.albums alb, music.artists art
where art.artno = alb.artno ;
```

```
create view music.inventory (type, itemno, totcost, totqty)
as select type, itemno, sum (price * qty), sum (qty)
from music.stock group by type, itemno;
```

```
create alias music.singers for music.artists ;
```

```
create alias music.emptystock for music.reorder ;
```

At this point, you may choose to create the new index, view and alias objects using the DB2 command line processor or the Data Server Manager tool. Follow the steps for your chosen tool only.

4A. Use the DB2 command line processor.

1. Issue the following series of commands using the DB2 command line processor.
 - **cd c:\inst23\ddl**
 - **db2 connect to musicdb**
 - **db2 -tvf create_stock_ix.ddl**
 - **db2 describe indexes for table music.stock**
 - **db2 -tvf create_VIEW_ALIAS.ddl**
 - **db2 list tables for schema music**
2. You can now skip to **Task 5**.

4B. Use the Data Server Manager tool.

We can use the DSM tool to create the index, view and alias objects.

1. Click the **Home** option on the left side of the DSM application.
You should see the database MUSICDB listed with some basic statistics on resource usage including, CPU, I/O and Memory. The database name MUSICDB provides a drop down list to perform different tasks.
2. Select **Administer - Explore Database** from the drop down list of the MUSICDB database.
You will be prompted for a userid and password for the MUSICDB database connection; use **inst23** with a password of **ibm2blue**.
A tree structure of database objects for this database is shown on the left side of the DSM view.
3. Click **Indexes**. The current index objects will be listed.
4. Click **Create** to open a list of options to define a new storage group. Select the **MUSIC.STOCK** table from the table list and enter the following values for the new storage group:

- For name: **STOCKITEM_IX (Use uppercase text)**
- For Members: select **Add**, then select the **ITEMNO** column
- *Leave other options with default values*

The DSM tool generates the CREATE INDEX statement that will define the new index named STOCKITEM_IX.

The command text should be similar to the following:

```
CREATE INDEX "MUSIC"."STOCKITEM_IX" ON "MUSIC"."STOCK" ("ITEMNO"
ASC) ;
```

5. Click **Next**.
6. Select **Run with SQL Editor** for the Run Schedule, and then click **Finish**.
7. Wait for the command to be processed.
The result should show that the command processing succeeded.
8. **Close** the **Create Index** tab.
If you refresh the index list, you should see the new index included in the list.
Now we will use DSM to execute the SQL statements in the file *create_View_Alias.ddl* to create view and alias objects.
9. Click **Open SQL Editor**.

10. Click **Upload** (which may be part of a list labelled *More Actions*). Use the Browse button for the Open SQL Script window to locate and select the file *create_View_Alias.ddl*.
11. Click **OK** to complete loading of the SQL text into the SQL editor.
Review the options specified for the CREATE VIEW and CREATE ALIAS statements.
12. Click **Run** and wait for the SQL statements to be processed.
The result should show that all of the statements succeeded.

Task 5. Create several table constraints and a trigger.

We will now create the following database objects:

- A foreign key constraint for the ALBUMS table that references the ARTNO column of the ARTISTS table.
- A foreign key constraint for the STOCK table that references the ITEMNO column of the ALBUMS table.
- A check constraint for the CCTYPE column of the STOCK table.
- A TRIGGER for the STOCK table that inserts a row into the REORDER table when the QTY column of the STOCK table is updated with a value of 5 or less.

At this point, you may choose to create the database objects using the DB2 command line processor or the Data Server Manager tool. Follow the steps for your chosen tool only.

5A. Use the DB2 command line processor.

The file *c:\inst23\ddl\create_ri.cc.ddl* contains following statements.

```
alter table music.albums add constraint ARTNO_FK
foreign key (artno) references music.artists (artno)
on delete cascade on update no action ;

alter table music.stock
foreign key ITEMNO_FK (itemno)
references music.albums on delete restrict;

alter table music.stock
add constraint cctype check (type in ('D', 'C', 'R')) ;
```


The file `c:\inst23\ddl\create_trigger.ddl` contains following statement.

```
create trigger music.reorder
after update of qty on music.stock
referencing new as n
for each row
mode db2sql
when (n.qty <= 5)
insert into music.reorder values (n.itemno, current timestamp);
```

1. Issue the following series of commands using the DB2 command line processor.

- **cd C:\inst23\ddl**
- **db2 connect to musicdb**
- **db2 -tvf create_ri_cc.ddl**
- **db2 -tvf create_trigger.ddl**

2. You can now skip to **Task 6**.

5B. Use the Data Server Manager tool.

We can use the DSM tool to execute the SQL file containing the ALTER TABLE statements to create referential integrity and check constraints.

1. Click **Home** on the left side of the **DSM** application.
You should see the database MUSICDB listed. The database name MUSICDB provides a drop down list to perform different tasks.
2. Select **Administer - Explore Database** from the drop down list of the MUSICDB database.
You will be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of **ibm2blue**.
A tree structure of database objects for this database is shown on the left side of the DSM view.
3. Click **Open SQL Editor**.
4. Click **Upload** (which may be part of a list labelled *More Actions*). Use the Browse button for the Open SQL Script window to locate and select the file `C:\inst23\ddl\create_ri_cc.ddl`.
5. Click **OK** to complete the loading of the SQL text into the SQL editor.
Review the options specified for the SQL statements.

6. Click **Run** and wait for the SQL statements to be processed.

The result should show that all statements succeeded.

You can define a new trigger using DSM. You could use the file `c:\inst23\ddl\create_trigger.ddl` in the SQL Editor or define the trigger using the create trigger function of DSM.

7. Select **Administer - Explore Database** from the drop down list of the **MUSICDB** database.

You will be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of **ibm2blue**.

A tree structure of database objects for this database is shown on the left side of the DSM view.

8. Click **Triggers**. The current index objects will be listed.
9. Click **Create** to open a list of options to define a new storage group. Select the **MUSIC.STOCK** table from the table list and enter the following values for the new storage group:

- For name: **REORDER (Use uppercase text)**
- For Action Time, select **After**
- For Trigger Event, check **Update**
- For Update columns: select **Add**, then select the **QTY** column
- For New row, enter **n**
- For Search condition, enter **n.qty <= 5**
- For SQL Procedure statement, enter:
Insert into music.reorder values (n.itemno, current timestamp)
- *Leave other options with default values*

The DSM tool generates the CREATE TRIGGER statement that will define the new index named REORDER.

The command text should be similar to the following:

```
create trigger music.reorder
after update of qty on music.stock
referencing new as n
for each row
mode db2sql
when (n.qty <= 5)
insert into music.reorder values (n.itemno, current timestamp);
```

10. Click **Next**.
11. Select **Run with SQL Editor** for the Run Schedule, and then click **Finish**.
12. Wait for the command to be processed.

The result should show that the command processing succeeded.

13. **Close** the **Create Trigger** tab.

If you refresh the index list, you should see the new trigger included in the list.

Task 6. Use the db2look command line tool to generate the DDL statements to create database objects.

You will use the **db2look** tool to extract selected DDL statements from the MUSICDB database.

1. Issue the following series of commands using the DB2 command line processor to generate the DDL associated with the ALBUMS table and save the output in a file named ALBUMS.DDL:

- **cd c:\inst23**
- **db2look -d musicdb -e -z music -t albums -o ALBUMS.DDL**

2. Review the generated statements placed into the file ALBUMS.DDL, by issuing the command:

- **more ALBUMS.DDL**

The file ALBUMS.DDL will contain DDL statements similar to the following:

```
-- This CLP file was created using DB2LOOK Version "10.5"
-- Timestamp: 8/17/2015 7:42:17 AM
-- Database Name: MUSICDB
-- Database Manager Version: DB2/NT64 Version 10.5.5
-- Database Codepage: 1208
-- Database Collating Sequence is: IDENTITY
-- Alternate collating sequence(alt_collate): null
-- varchar2 compatibility(varchar2_compat): OFF
```

```
CONNECT TO MUSICDB;
```

```
-----
-- DDL Statements for Table "MUSIC  "."ALBUMS"
-----
```

```
CREATE TABLE "MUSIC  "."ALBUMS" (
    "TITLE" VARCHAR(50 OCTETS) ,
    "ARTNO" SMALLINT NOT NULL ,
    "ITEMNO" SMALLINT NOT NULL )
IN "TSP04" INDEX IN "TSP05"
ORGANIZE BY ROW;
```

```
-- DDL Statements for Primary Key on Table "MUSIC  "."ALBUMS"
```

```
ALTER TABLE "MUSIC  "."ALBUMS"
    ADD CONSTRAINT "ALBUM_ITEM" PRIMARY KEY
        ("ITEMNO");
```

```
-- DDL Statements for Foreign Keys on Table "MUSIC  "."ALBUMS"
```

```
ALTER TABLE "MUSIC  "."ALBUMS"
    ADD CONSTRAINT "ARTNO_FK" FOREIGN KEY
        ("ARTNO")
    REFERENCES "MUSIC  "."ARTISTS"
        ("ARTNO")
    ON DELETE CASCADE
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;
```



```
-----
-- DDL Statements for Views
```


```
-----  
SET CURRENT SCHEMA = "INST23  ";  
SET CURRENT PATH = "SYSIBM","SYSFUN","SYSPROC","SYSIBMADM","INST23";  
create view music.music as select title, classification, name  
from music.albums alb, music.artists art  
where art.artno = alb.artno;  
  
COMMIT WORK;  
  
CONNECT RESET;  
  
TERMINATE;
```

Results:

You created a group of database objects in the DB2 database named MUSICDB.

Unit 6 Moving data





Moving data

DB2 10.5 Administration Workshop for Windows

© Copyright IBM Corporation 2016
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Demonstration 1

Moving data

- Use the DB2 IMPORT command to load data into a DB2 table.
- Run the INGEST command to efficiently load data into a DB2 table.
- Invoke the LOAD utility to process input files and load data into DB2 tables.
- Run SET INTEGRITY commands to resolve the set integrity pending conditions resulting from loading data into tables with constraints defined using a LOAD utility.

Demonstration 1: Moving data

Demonstration 1: Moving data

Purpose:

This demonstration uses several methods to load data into DB2 tables including the **IMPORT**, **INGEST** and **LOAD** commands. You will use the **SET INTEGRITY** statement to resolve constraint checking after using a **LOAD** command.

Task 1. Use the **IMPORT** utility to load data into a table.

1. Logon to the Windows system using the user id **inst23**, with a password of **ibm2blue**.

The DB2 command line processor will be used to issue some DB2 and system commands. You will need to start the DB2 command window in administrator mode to issue these DB2 commands.

2. To start the DB2 command window, click on the Windows start icon, then Navigate to **All programs > IBM DB2 DB2COPY1 > DB2 Command Window - Administrator**. When prompted with the question, 'Do you want to allow the following program to make changes to this computer?' select **Yes**.

A set of course files are located in the directory **c:\inst23\ddl**. Change to this directory to make it easier to access these files that contain DB2 commands or SQL statements.

The data to be loaded into the **ARTISTS** table is in the delimited file named **c:\inst23\artists.del**

At this point you may choose to use the DB2 command line processor or the Data Server Manager tool. Follow the steps for your chosen tool only, to run the **IMPORT** command to load data into the **ARTISTS** table.

1A. Use the DB2 command line processor.

1. Issue the following series of commands using the DB2 command line processor.
 - **cd c:\inst23\ddl**
 - **db2 connect to musicdb**
 - **db2 IMPORT from c:\inst23\artists.del of del insert into music.artists**

The output generated will look similar to the following:

```
import from c:\inst23\artists.del of del insert into music.artists
SQL3109N  The utility is beginning to load data from file
"c:\inst23\artists.del".

SQL3110N  The utility has completed processing.  "79" rows were read from the
input file.

SQL3221W  ...Begin COMMIT WORK. Input Record Count = "79".

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

SQL3149N  "79" rows were processed from the input file.  "79" rows were
successfully inserted into the table.  "0" rows were rejected.

Number of rows read           = 79
Number of rows skipped        = 0
Number of rows inserted       = 79
Number of rows updated        = 0
Number of rows rejected       = 0
Number of rows committed     = 79
```

2. You can now skip to **Task 2**.

1B. Use the Data Server Manager tool.

We can use the Data Server Manager tool to run DB2 utilities like IMPORT and LOAD can be executed using the SQL Editor function. These utilities need to be invoked using the **ADMIN_CMD** procedure interface.

If you have not started the DSM application, start it now using the following URL text, **http://localhost:11080**. Use the following user and password for DSM.

When DSM was installed, a local user id and password were selected to be associated with the application. For this course the following were configured:

Windows user id: **db2admin**

User password: **ibm2blue**

1. Click the **Home** option on the left side of the DSM application.
You should see the database MUSICDB listed. The database name MUSICDB provides a drop down list to perform different tasks.

2. Select **Administer - Explore Database** from the drop down list of the MUSICDB database.

You will be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of **ibm2blue**.

A tree structure of database objects for this database is shown on the left side of the DSM view.

3. Click **Open SQL Editor**.
4. Click **Upload** (which may be part of a list labelled *More Actions*). Use the Browse button for the Open SQL Script window to locate and select the file **c:\inst23\ddl\import_artists.sql**

5. Click **OK** to complete loading the SQL text into the SQL editor.

The CALL to the procedure SYSPROC.ADMIN_CMD, will be used to invoke the IMPORT command. The INSERT option of IMPORT is being used.

Notice the MESSAGES ON SERVER option is included, which allows the IMPORT to run from a client system but the generated messages are stored on the database server.

6. Click **Run** and wait for the SQL CALL statement to be processed.

The result area shows that 79 data rows were read from the file and inserted into the table MUSIC.ARTISTS.

Task 2. Use the INGEST command to load data into a DB2 table.

We can use the DB2 command line processor to run the INGEST command to load data files into a DB2 table. The Data Server Manager tool does not currently provide a method to invoke the INGEST command.

The **INGEST** command uses a work table **SYSTOOLS.INGESTRESTART** to save command restart data.

The file *cr_toolspace.ddl* will be used to create this table prior to running the INGEST command. This only needs to be performed once per DB2 database.

The file *c:\inst23\ddl\ingest_albums.ddl* contains the following INGEST command:

```
ingest from file c:\inst23\albums.del
format delimited messages ingest_albums.txt
RESTART NEW 'ingest_alb' INSERT INTO music.albums ;
```

1. Issue the following series of commands using the DB2 command line processor.

- **cd c:\inst23\ddl**
- **db2 connect to musicdb**
- **db2 -tvf cr_toolspace.ddl**
- **db2 -tvf ingest_albums.ddl**

After a few seconds, the output generated will look similar to the following:

```
ingest from file c:\inst23\albums.del format delimited messages
ingest_albums.txt RESTART NEW 'ingest_alb' INSERT INTO music.albums
```

```
Number of rows read           = 264
Number of rows inserted       = 264
Number of rows rejected       = 0
```

```
SQL2980I  The ingest utility completed successfully at timestamp
"08/18/2015
09:26:51.608177"
```

Task 3. Use the DB2 LOAD utility to load data into a table that has a foreign key constraint defined.

Next we will utilize the DB2 LOAD utility to add data to the STOCK table. The ALBUMS table and ARTISTS table contain the data added using the IMPORT utility and INGEST in the previous tasks.

We will use the LOAD command in the file *C:\inst23\ddl\load_stock1.ddl* to invoke the load processing.

```
CALL SYSPROC.ADMIN_CMD (
  'LOAD FROM "c:\inst23\stock.del" of del
  MODIFIED BY GENERATEDMISSING METHOD P (1,2,3,4) MESSAGES ON SERVER
  INSERT INTO MUSIC.STOCK (ITEMNO,TYPE,PRICE,QTY) ' ) ;
```

The file *C:\inst23\ddl\set_integrity_stock.sql* will be used to run SET INTEGRITY to perform constraint checking not performed by the LOAD utility.

At this point you may choose to use the DB2 command line processor or the Data Server Manager tool, to execute the LOAD and SET INTEGRITY commands. Follow the steps for your chosen tool only.

3A. Use the DB2 command line processor.

1. Issue the following series of commands using the DB2 command line processor.
 - **cd c:\inst23\ddl**
 - **db2 connect to musicdb**
 - **db2 -tvf load_stock1.ddl**
 - **db2 -tvf set_integrity_stock.sql**
2. You can now skip to **Task 4**.

3B. Use the Data Server Manager tool.

We can use the DSM tool to execute the file containing the LOAD utility command.

1. Click the **Home** option on the left side of the DSM application.
You should see the database MUSICDB listed with some basic statistics on resource usage including, CPU, I/O and Memory. The database name MUSICDB provides a drop down list to perform different tasks.
2. Select **Administer - Explore Database** from the drop down list of the MUSICDB database.
You will be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of *ibm2blue*.
A tree structure of database objects for this database is shown on the left side of the DSM view.
3. Click **Open SQL Editor**.
4. Click **Upload** (which may be part of a list labelled *More Actions*). Use the Browse button for the Open SQL Script window to locate and select the file *c:\inst23\ddl\load_stock1.ddl*. Click on **OK** to complete loading the SQL text into the SQL editor.
The CALL to the procedure SYSPROC.ADMIN_CMD, will be used to invoke the LOAD command. The INSERT option of LOAD is being used.
The GENERATEDMISSING option is necessary because the input file does not include data for the three columns SYS_START, SYS_END and TX_START. These are the columns used to support the STOCK table as a system period temporal table.

5. Click **Run** and wait for the SQL CALL statement to be processed.
The result area shows that 777 data rows were read from the file and inserted into the table MUSIC.STOCK.

The LOAD utility placed the STOCK table into set integrity pending state because the table has constraints defined. We can use the SET INTEGRITY command to resolve this state.

6. Click **Upload** (which may be part of a list labelled *More Actions*). Use the Browse button for the Open SQL Script window to locate and select the file *c:\inst23\ddl\set_integrity_stock.sql*. Click on **OK** to complete loading the SQL text into the SQL editor.

The file contains the following SQL text:

```
SET INTEGRITY FOR MUSIC.STOCK
  ALLOW NO ACCESS IMMEDIATE CHECKED
;
```

The SET INTEGRITY statement is needed to check the foreign key and check constraints defined for the STOCK table

7. Click on **Run** and wait for the SQL statement to be processed.
The result should show that the SET INTEGRITY statement succeeded.

Task 4. Run a file containing a series of LOAD commands and SET INTEGRITY statements.

You may want to periodically refresh or extend tables with new data. If you use the DB2 LOAD utility and the target tables include referential or check constraints you will need to run SET INTEGRITY statements following the load processing.

We will utilize a command script that contains LOAD commands and SET INTEGRITY statements to load data into several tables.

It is normally a good idea to use exception tables during a LOAD operation. We have provided a script (named **create_exception_tables.ddl**) that will create exception tables for ARTISTS and ALBUMS.

At this point you may choose to use the DB2 command line processor or the Data Server Manager tool, to perform the LOAD and SET INTEGRITY processing.

Follow the steps for your chosen tool only.

4A. Use the DB2 command line processor.

The file **c:\inst23\ddl\load_tables_clp.ddl** contains LOAD and SET INTEGRITY statements that perform the following:

- LOAD data in the CONCERTS table using the REPLACE option, this table does not have any constraints to be checked.
 - LOAD data into the ARTISTS table using the REPLACE option.
 - A SQL query that searches the SYSCAT.TABLES view for any tables with set integrity pending status.
 - The SET INTEGRITY statement to check the ARTISTS and ALBUMS tables.
 - The SET INTEGRITY statement to check the STOCK table.
 - A second SQL query to verify that all set integrity pending states have been resolved.
1. Issue the following series of commands using the DB2 command line processor.

- **cd c:\inst23\ddl**
- **db2 connect to musicdb**
- **db2 -tvf create_exception_tables.ddl**
- **db2 -tvf load_tables_clp.ddl**

The output produced will look similar to the following:

```
LOAD FROM "C:\INST23\concerts.del" OF del METHOD P (1, 2, 3) MESSAGES
load_concert.txt REPLACE INTO MUSIC.CONCERTS (ARTNO, DATE, CITY)
```

```
Number of rows read      = 10
Number of rows skipped   = 0
Number of rows loaded    = 10
Number of rows rejected  = 0
Number of rows deleted   = 0
Number of rows committed = 10
```

```
load from "C:\INST23\artists.del" of del messages load_art.txt replace into
music.artists for exception music.artexp
```

```
Number of rows read      = 79
Number of rows skipped   = 0
Number of rows loaded    = 79
Number of rows rejected  = 0
Number of rows deleted   = 0
Number of rows committed = 79
```

```
select substr(tabname,1,18), status, substr(const_checked,1,1) as FK_CHECKED,
substr(const_checked,2,1) as CC_CHECKED from syscat.tables where status='C'
```

```
1          STATUS FK_CHECKED CC_CHECKED
-----
ARTISTS          C          Y          Y
```

1 record(s) selected.

```
SET INTEGRITY FOR MUSIC.ARTISTS, MUSIC.ALBUMS ALLOW NO ACCESS IMMEDIATE CHECKED
FOR EXCEPTION IN MUSIC.albums use MUSIC.albexp , in MUSIC.artists use MUSIC.artexp
SQL3601W The statement caused one or more tables to automatically be placed
in the Set Integrity Pending state.  SQLSTATE=01586
```

```
SET INTEGRITY FOR MUSIC.STOCK ALLOW NO ACCESS IMMEDIATE CHECKED
DB20000I The SQL command completed successfully.
```

```
select substr(tabname,1,18), status, substr(const_checked,1,1) as FK_CHECKED,
substr(const_checked,2,1) as CC_CHECKED from syscat.tables where status='C'
```

```
1          STATUS FK_CHECKED CC_CHECKED
-----
```

0 record(s) selected.

2. You have now completed this lab exercise.

4B. Use the Data Server Manager tool.

We can use the DSM tool to execute a SQL file that includes LOAD and SET INTEGRITY processing.

First we will execute the SQL file *create_exception_tables.ddl* that creates exception tables for the ALBUMS and ARTISTS tables. These can be referenced by LOAD and SET INTEGRITY to be able to complete processing if there are exceptions to table constraints.

1. Click the **Home** option on the left side of the DSM application.
You should see the database MUSICDB listed. The database name MUSICDB provides a drop down list to perform different tasks.
2. Select **Administer - Explore Database** from the drop down list of the MUSICDB database.
You will be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of **ibm2blue**.
A tree structure of database objects for this database is shown on the left side of the DSM view.
3. Click **Open SQL Editor**.

4. Click **Upload** (which may be part of a list labelled *More Actions*). Use the Browse button for the Open SQL Script window to locate and select the file `C:\inst23\ddl\create_exception_tables.ddl`. Click on **OK** to complete loading the SQL text into the SQL editor.

Review the options specified for the SQL statements.

5. Click **Run** and wait for the SQL statements to be processed.

The result should show that all statements succeeded.

Now we will use DSM to execute the SQL file that will perform the following steps:

- Use ADMIN_CMD to LOAD data in the CONCERTS table using the REPLACE option, this table does not have any constraints to be checked.
- Use ADMIN_CMD to LOAD data into the ARTISTS table using the REPLACE option.
- A SQL query that searches the SYSCAT.TABLES view for any tables with set integrity pending status.
- The SET INTEGRITY statement to check the ARTISTS and ALBUMS tables.
- The SET INTEGRITY statement to check the STOCK table.
- A second SQL query to verify that all set integrity pending states have been resolved.

The ADMIN-CMD procedure calls allow the LOAD utility to be initiated by applications like DSM.

6. Click **Open SQL Editor**.
7. Click **Upload** (which may be part of a list labelled *More Actions*). Use the Browse button for the Open SQL Script window to locate and select the file `C:\inst23\ddl\load_tables.sql`. Click on **OK** to complete loading the SQL text into the SQL editor.

Review the series of statements.

8. Click **Run** and wait for the SQL statements to be processed.
- Browse the Results view to make sure all of the processing completed successfully. Each LOAD will generate statistics showing the number of rows of data read and the number of rows loaded into the table.

Results:

You utilized several methods to load data into DB2 tables including the IMPORT, INGEST and LOAD commands. You used SET INTEGRITY statements to resolve set integrity pending states for tables that were loaded using the LOAD utility.

Unit 7 Backup and recovery

IBM Training



Backup and recovery

DB2 10.5 Administration Workshop for Windows

© Copyright IBM Corporation 2016
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Demonstration 1

Backup and recovery

- Create a backup image of a database including all of the table spaces.
- Configure a database to change from circular logging to enable archive logging to make the database fully recoverable.
- Restore a database from a backup image and Roll forward a database to perform point in time recovery using the RECOVER command.

Demonstration 1: Backup and recovery

Demonstration 1: Backup and recovery

Purpose:

This demonstration utilizes the database MUSICDB to perform backup and recovery related tasks. During this exercise, you will prepare the database for archival logging, create a backup image and recover the database to a selected point in time. You will also configure the database log space to handle applications with large transactions.

Task 1. Configure the available log space for a database.

1. Logon to the Windows system using the user id **inst23**, with a password of **ibm2blue**.

The DB2 command line processor will be used to issue some DB2 and system commands. You will need to start the DB2 command window in administrator mode to issue these DB2 commands.

2. To start the DB2 command window, click on the Windows start icon, then Navigate to **All programs > IBM DB2 DB2COPY1 > DB2 Command Window - Administrator**. When prompted with the question, 'Do you want to allow the following program to make changes to this computer?' select **Yes**.

We will start by configuring the database with a small database log space to test the effect on processing application requests that exceed the database log space limits.

This task will be performed using the DB2 command line processor.

3. Issue the following series of commands, using the DB2 command line processor.

- **cd c:\inst23\ddl**
- **db2 connect to musicdb**
- **db2 update db cfg using logprimary 3 logsecond 1 logfilsiz 6**
- **db2 get db cfg show detail | more**

The SQL1363W message indicates that at least one of the configuration changes requires a database restart to take effect. In this case, LOGFILSIZ and LOGPRIMARY cannot be changed dynamically.

The db2pd command can be used to show the current and any deferred database configuration settings.

4. Issue the following series of commands using the DB2 command line processor.

- **db2pd -db musicdb -dbcfg | find "LOG"**

The command output will look similar to the following:

CATALOGCACHE_SZ (4KB)	300	300
LOGBUFSZ (4KB)	2149	2149
LOGFILSIZ (4KB)	1024	6
LOGPRIMARY	5	3
LOGSECOND	1	1
NEWLOGPATH (memory)		
NEWLOGPATH (disk)		
Path to log files (memory)	C:\INST23\NODE0000\SQL00001\LOGSTREAM0000\	
OVERFLOWLOGPATH (memory)		
OVERFLOWLOGPATH (disk)		
MIRRORLOGPATH (memory)		
MIRRORLOGPATH (disk)		
First active log file	S0000000.LOG	S0000000.LOG
BLK_LOG_DSK_FUL	NO	NO
BLOCKNONLOGGED	NO	NO
MAX_LOG	0	0
NUM_LOG_SPAN	0	0
LOGARCHMETH1 (memory)	OFF	
LOGARCHMETH1 (disk)	OFF	
LOGARCHCOMPR1	OFF	OFF
LOGARCHOPT1		
LOGARCHMETH2 (memory)	OFF	
LOGARCHMETH2 (disk)	OFF	
LOGARCHCOMPR2	OFF	OFF
LOGARCHOPT2		
LOGINDEXBUILD	0	0
LOG_DDL_STMTS	YES	YES
LOG_APPL_INFO	YES	YES

Now we will connect to the MUSICDB database and run two SQL command files using the DB2 command line processor.

For the change processing, you will include the **+C** option to process the statements as a single unit of work or transaction.

The DB2 command line processor will automatically commit each statement processed by default. The +C option turns autocommit off so that all of the changes made by multiple statements must be retained in the DB2 active log files to support rolling back the changes, if necessary.

5. Issue the following series of commands using the DB2 command line processor:

- **db2 force application all**
- **db2 terminate**
- **db2 connect to musicdb**
- **db2pd -db musicdb -logs**
- **db2 -tvf create_temp_stock.sql**
- **db2 +C -tvf stock_insert.sql**

The command output will look similar to the following:

```
set current schema music
```

```
DB20000I The SQL command completed successfully.
```

```
insert into temp_stock select itemno,type,price,qty from stock where itemno < 100
```

```
DB20000I The SQL command completed successfully.
```

```
insert into temp_stock select itemno,type,price,qty from stock where itemno < 100
```

```
DB21034E The command was processed as an SQL statement because it was not a  
valid Command Line Processor command. During SQL processing it returned:
```

```
SQL0964C The transaction log for the database is full. SQLSTATE=57011
```

```
insert into temp_stock select itemno,type,price,qty from stock where itemno < 100
```

```
DB21034E The command was processed as an SQL statement because it was not a  
valid Command Line Processor command. During SQL processing it returned:
```

```
SQL0964C The transaction log for the database is full. SQLSTATE=57011
```

```
delete from temp_stock
```

```
DB21034E The command was processed as an SQL statement because it was not a  
valid Command Line Processor command. During SQL processing it returned:
```

```
SQL0964C The transaction log for the database is full. SQLSTATE=57011
```

Once our limited database log space is filled, the database changes fail with the SQL0964 message indicating that “The transaction log for the database is full”.

Use the DB2 command file *increase_logs.ddl* to allocate sufficient database log space to process the database changes.

6. Issue the following series of commands, using the DB2 command line processor.

- **db2 terminate**
- **db2 connect to musicdb**
- **db2 -tvf increase_logs.ddl**
- **db2 terminate**

Now will retry the command file containing the table changes.

You will include the **+C** option to process the statements as a single unit of work or transaction.

7. Issue the following series of commands using the DB2 command line processor.

- **db2 connect to musicdb**
- **db2 +C -tvf stock_insert.sql**
- **db2 terminate**

The command output will look similar to the following:

```
set current schema music
```

```
DB20000I The SQL command completed successfully.
```

```
insert into temp_stock select itemno,type,price,qty from stock where itemno < 100
```

```
DB20000I The SQL command completed successfully.
```

```
insert into temp_stock select itemno,type,price,qty from stock where itemno < 100
```

```
DB20000I The SQL command completed successfully.
```

```
insert into temp_stock select itemno,type,price,qty from stock where itemno < 100
```

```
DB20000I The SQL command completed successfully.
```

```
delete from temp_stock
```

```
DB20000I The SQL command completed successfully.
```

The increased database log space allows the statements to process without errors.

Task 2. Database Recovery Support with Archive Logging.

In this section, we will implement archive logging. This will allow the database to be more completely recovered, including the logged changes.

By default, our database MUSICDB is using circular logging which only supports database level recovery to a previous backup copy. We will perform several sets of database changes and then perform a point-in-time database recovery to include some of the changes. This could be used to handle application failures that cause tables to contain incorrect data.

Switching from circular logging to archive logging requires an offline database backup in order to establish the starting point where all new logged changes will be kept.

The disk location **C:\inst23\archive** will be used as the location to archive logs.

The disk location **C:\inst23\backup** will be used to store the database backup file.

At this point you may choose to use the DB2 command line processor or the Data Server Manager tool, to configure the MUSICDB database to perform archive logging.

2A. Use the DB2 command line processor.

1. Issue the following series of commands using the DB2 command line processor.
 - **cd c:\inst23\ddl**
 - **db2 connect to musicdb**
 - **db2 UPDATE DATABASE CONFIGURATION USING logarchmeth1 "DISK:C:\inst23\archive" logprimary 3 logsecond 10 logfilsiz 1000 LOGINDEXBUILD OFF**
 - **db2 connect reset**
 - **db2 force application all**
 - **db2 terminate**
 - **db2 deactivate database musicdb**
 - **db2 BACKUP DATABASE MUSICDB TO C:\inst23\backup COMPRESS**
2. You can now skip to **Task 3**.

2B. Use the Data Server Manager tool.

We can use the DSM tool to configure the database to support archived log files. A shortcut to **Data Server Manager** (DSM) may be available on the Windows desktop.

If no shortcut is provided, open Internet Explorer and type the following URL text to launch DSM: **http://localhost:11080**.

When DSM was installed, a local user profile was created, and associated to the application: userid: **db2admin**, password: **ibm2blue**

1. Log into **DSM**.
2. Click the **Home** option on the left side of the DSM application.
You should see the database MUSICDB listed with some basic statistics on resource usage including, CPU, I/O and Memory. The database name MUSICDB provides a drop down list to perform different tasks.
3. Select **Administer - Explore Database** from the drop down list of the MUSICDB database.
You will be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of *ibm2blue*.
A tree structure of database objects for this database is shown on the left side of the DSM view.
4. Click **Configure Logging** (which may be part of a list labelled *More Actions*). Enter the following values for Configure Database Logging Options:
 - For Configure Database Logging Type : **Archive**
 - For Archive Log Handling: **Automatic DB2 Archive**
 - For Media Type: **File System**
 - For Primary Archive Log: Use the Browse button to locate and select the directory *C:\inst23\archive*, Click on **OK**
 - Under Choose the Number and Size of the Log files, select these options:
 - For Number of Primary log files : **3**
 - For Number of Secondary log files: **10**
 - For Size of Each Log File: **1000**

The DSM tool generates the UPDATE DATABASE CONFIGURATION statement and the commands that will deactivate the database to allow these changes to take effect.

5. Click **Next**.

6. Select **Command Line Processor** for the Run Method Schedule, and then click **Finish**.
7. Wait for the commands to be processed.
8. You can close the **Configure Database Logging** tab.
The database configuration has been changed, but the database is now in a Backup pending state. We will create an offline database backup to allow the database to begin archive logging.
9. Under Explore Database, click **Backup**.
10. Enter the following values for Back Up Options:
 - Image Type : **File System**
 - select the backup location, using the **Browse** button to locate and select the directory *C:\inst23\backup*,
 - click **OK**
 - under **Options**, use select **Compress Backup Image**

The DSM tool generates the BACKUP DATABASE command.
11. Click **Next**.
12. Select **Command Line Processor** for the Run Method Schedule and click on **Finish**.
13. Click **Run** and wait for the BACKUP command to be processed.
You can close the Backup database tab.

Task 3. Generate multiple sets of table changes noting the point in time for each change.

In this section we will perform several sets of changes, noting the current time when each set completes. These changes are now being archived by DB2 automatically.

A script file named *stock_insert2.sql* contains two SQL statements.

- The INSERT statement adds a set of rows from the MUSIC.STOCK table to the MUSIC.TEMP_STOCK table.
- A SELECT statement returns the current row count for the TEMP_STOCK table and the current local date and time information.

Record these date and time values.

At this point you may choose to use the DB2 command line processor or the Data Server Manager tool, to execute the SQL script file that makes the changes to the table TEMP_STOCK.

3A. Use the DB2 command line processor.

1. Issue the following series of commands using the DB2 command line processor.

- **cd c:\inst23\ddl**
- **db2 connect to musicdb**
- **db2 -tvf stock_insert2.sql**

2. Record the following from the SQL result:

- Current Table size: _____ (result 1)
- Local date: _____ (result 1)
- Local time: _____ (result 1)

The command output will look similar to the following:

```
set current schema music
```

```
DB20000I The SQL command completed successfully.
```

```
insert into temp_stock select itemno,type,price,qty from stock where TYPE = 'C'
```

```
DB20000I The SQL command completed successfully.
```

```
SELECT COUNT(*) AS CURRENT_SIZE, CURRENT DATE , CURRENT TIME FROM temp_stock
```

```
CURRENT_SIZE 2          3
-----
259 09/16/2015 06:55:25
```

```
1 record(s) selected.
```

3. Run the command file a second time to make additional logged changes and note the results.

- **db2 -tvf stock_insert2.sql**

4. Record the following from the SQL result:

- Current Table size: _____ (result 2)
- Local date: _____ (result 2)
- Local time: _____ (result 2)

5. You can now skip to **Task 4**.

3B. Use the Data Server Manager tool.

We will use the Data Server Manager tool to run the SQL scripts that perform the database changes.

1. Click the **Home** option on the left side of the DSM application.
You should see the database MUSICDB listed. The database name MUSICDB provides a drop down list to perform different tasks.
2. Select **Administer - Explore Database** from the drop down list of the MUSICDB database.
You will be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of *ibm2blue*.
A tree structure of database objects for this database is shown on the left side of the DSM view.
3. Click **Open SQL Editor**.
4. Click **Upload** (which may be part of a list labelled *More Actions*). Use the Browse button for the Open SQL Script window to locate and select the file *C:\inst23\ddl\stock_insert2.sql*.
5. Click **OK** to complete loading of the SQL text into the SQL editor.
Review the SQL statements.
6. Click **Run** and then wait for the SQL statements to be processed.
7. Record the following from the SQL result:
 - Current Table size: _____ (result 1)
 - Local date: _____ (result 1)
 - Local time: _____ (result 1)
8. Now run the INSERT script a second time to generate a second set of logged changes.
9. Click **Run** and wait for the SQL statements to be processed.
10. Record the following from the SQL result:
 - Current Table size: _____ (result 2)
 - Local date: _____ (result 2)
 - Local time: _____ (result 2)

Task 4. Recover the database to a specific point in time.

Now you will use the RECOVER database command to restore the MUSICDB database to the point in time of the first execution of the insert SQL script. The RECOVER DATABASE command will locate the required backup image using the database recovery history file. A RESTORE DATABASE command would require the backup location to be specified.

The recovery target time value is specified as a time stamp, a 7-part character string that identifies a combined date and time.

The format is yyyy-mm-dd-hh.mm.ss.nnnnnn (year, month, day, hour, minutes, seconds, microseconds).

db2 recover database musicdb to yyyy-mm-dd-hh.mm.ss

(Use the Date and time recorded in Task 3 for the first execution).

1. Issue the following series of commands using the DB2 command line processor.

- **db2 terminate**
- **db2 force application all**
- **db2 recover database musicdb to yyyy-mm-dd-hh.mm.ss**

Use the Date and time recorded for the first execution of *stock_insert2.sql* (Results 1). Wait about 1 minute before trying to perform the 'recover' command - giving time for the database to shut down.

2. Now connect to the database and check the count of rows in the MUSIC.TEMP_STOCK table.

The count should match the number of rows recorded for the first execution of the SQL insert script.

3. Issue the following series of commands using the DB2 command line processor.

- **db2 connect to musicdb**
- **db2 “select count(*) from music.temp_stock”**
- **db2 terminate**

Results:

You utilized the database MUSICDB to perform backup and recovery related tasks, including configuring the database for archival logging, creating a backup image and recovered the database to a selected point in time. You also configured the database log space to handle applications with large transactions.

Unit 8

Database maintenance, monitoring and problem determination

IBM Training



Database maintenance, monitoring and problem determination

DB2 10.5 Administration Workshop for Windows

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the written permission of IBM.

Demonstration 1

Using DB2 tools for performance

- Use an explain tool to analyze the access plan for an SQL statement.
- Create an index for a table to reduce processing costs for SQL.
- Invoke the DB2 Design Advisor to suggest new indexes for an SQL workload.
- Use the DB2 RUNSTATS and REORG commands to reorganize a table based on an index and collect new catalog statistics.

Demonstration 1 Using DB2 tools for performance

Demonstration 1: Using DB2 tools for performance

Purpose:

This demonstration uses several DB2 tools and utilities to improve the performance of a SQL query. You will use the Data Server Manager Visual Explain tool to review the access plans and the estimated costs for processing SQL statements. You will use the DB2 design advisor to suggest a new index to reduce processing costs. You will execute a DB2 REORG utility to reorganize a table to improve performance.

Task 1. Load a test table and create a set of explain tables for query analysis.

1. Logon to the Windows system using the user id **inst23**, with a password of **ibm2blue**.

The DB2 command line processor will be used to issue some DB2 and system commands. You will need to start the DB2 command window in administrator mode to issue these DB2 commands.

2. To start the DB2 command window, click on the Windows start icon, then Navigate to **All programs > IBM DB2 DB2COPY1 > DB2 Command Window - Administrator**. When prompted with the question, 'Do you want to allow the following program to make changes to this computer?' select **Yes**.

In this section, you will run a SQL script that creates and loads a DB2 table, TEST.HISTORY that will be used to analyze the performance of a test query. We will also create a set of explain tables that can be used to support explain tool access plan analysis. The file *explain.ddl* calls the SYSINSTALLOBJECTS procedure to create a set of explain tables matching the current DB2 release.

This task will be performed using the DB2 command line processor.

3. Issue the following series of commands using the DB2 command line processor.

- **cd c:\inst23\ddl**
- **db2 connect to musicdb**
- **db2 -tvf explain.ddl**
- **db2 -tvf create_testhist.ddl**

The SQL statements show a sum from the QTY column for a set of rows before and after the UPDATE processing.

The output will look similar to the following:

CONNECT TO MUSICDB

Database Connection Information

```
Database server      = DB2/NT64 10.5.5
SQL authorization ID = INST23
Local database alias = MUSICDB
```

```
CREATE TABLE "TEST"    "."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 CURRENT TIMESTAMP ,
"ACCTNAME" CHAR(20) NOT NULL , "TEMP" CHAR(6) NOT NULL ) IN "USERSPACE1"
```

DB20000I The SQL command completed successfully.

```
LOAD FROM C:\INST23\HISTDATA.IXF OF IXF REPLACE INTO TEST.HISTORY
NONRECOVERABLE
```

SQL3109N The utility is beginning to load data from file
"C:\INST23\HISTDATA.IXF".

SQL3500W The utility is beginning the "LOAD" phase at time "08/20/2015
14:45:17.365469".

SQL3150N The H record in the PC/IXF file has product "DB2 02.00", date
"20120817", and time "154700".

SQL3050W Conversions on the data will be made between the IXF file code
page

"1252" and the application code page "1208".

SQL3153N The T record in the PC/IXF file has name "histdata.del",
qualifier

"", and source " ".

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 "08/20/2015
14:45:20.930460".

```
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
```

```
RUNSTATS ON TABLE TEST.HISTORY
DB20000I  The RUNSTATS command completed successfully.
```

```
CONNECT RESET
DB20000I  The SQL command completed successfully.
```

Task 2. Use the Explain SQL capability of Data Server Manager to review the access plan and estimated costs for processing a SQL statement.

The file *C:\inst23\ddl\query_history.sql* will be used as a sample application SQL statement.

We will use the DSM Explain tool to review the access plan for a SQL statement

1. Click **Home** on the left side of the DSM application.
You should see the MUSICDB database listed, with some basic statistics on resource usage including, CPU, I/O and Memory. The database name MUSICDB provides a drop down list to perform different tasks.
2. Select **Administer: Explore Database** from the drop down list of the MUSICDB database.
When prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of **ibm2blue**.
A tree structure of database objects for this database is shown on the left side of the DSM view.
3. Click **Open SQL editor**.
4. Click **Upload** (which may be part of a list labelled *More Actions*). Use the **Browse** button for the Open SQL Script window to locate and select the file *C:\inst23\ddl\query_history.sql*
5. Click **OK** to complete loading of the SQL text into the SQL editor.
The SQL statement is a SELECT from the table test.HISTORY with two predicates, one on the BRANCH_ID column and one on the TELLER_ID column.

6. Click **Explain** to generate the access plan report.

The screenshot shows the Oracle SQL Developer interface. The 'Explain' tab is active, displaying a query plan for a table scan operation. A tooltip is visible over the 'TBSCAN' operation, showing the following statistics:

Operation	Table scan
Estimated cardinality	162.025
Actual cardinality	-
Cumulative total cost (timeron)	3,238.60
Cumulative CPU cost (timeron)	568,579,712.00
Cumulative I/O cost (timeron)	3,575.00
Total cost (timeron)	3,238.60
CPU cost (timeron)	568,579,712.00
I/O cost (timeron)	3,575.00

The tooltip also includes a description: 'Avoid table space scans (Operator ID = 2) on table TEST.HISTORY. The table is accessed by a table space scan. Consider running the Statistics Advisor, because the improving statistics might improve the access path. Also, consider running the Index Advisor to determine whether creating an index might improve the access path.'

7. Select the **TBSCAN** operation and look at the cost statistics that appear to the left of the TBSCAN octagon.

A TBSCAN operation will read every page in the table to produce the result. Look at the following:

- Estimated Cardinality: _____ (test result 162)
- Cumulative Total Cost: _____ (test result 3,328)
- Cumulative I/O Cost : _____ (test result 3,575)

Your results may be slightly different, but a test result produced the estimated cardinality of 162 rows, and an estimated I/O cost of 3,575, which is the number of pages in the table being accessed. There are 200,000 rows in the test table, so the access plan will scan every row and only return about 162 rows of result. We should be able to tune this query to run more efficiently.

8. Close the **Explain** tab to return to the SQL Editor.

Task 3. Use the **db2adv** command to recommend additional table indexes that could reduce processing costs for the SQL query.

The **db2adv** command can be used to suggest changes, like adding new table indexes that could reduce processing costs for an SQL workload.

We will utilize the SQL query from the file *C:\inst23\ddl\query_history.sql*, as the workload to input to the **db2adv** command.

1. Issue the following series of commands using the DB2 command line processor.
 - **cd c:\inst23\ddl**
 - **db2adv -d musicdb -i query_history.sql | more**
 - **db2 -tvf create_testhist.ddl**

The output will include results similar to the following:

```

execution started at timestamp 2016-01-25-11.26.19.388694
found [3] SQL statements from the input file
Recommending indexes...
total disk space needed for initial set [ 10.981] MB
total disk space constrained to [ 24.837] MB
Trying variations of the solution set.
1 indexes in current solution
[3444.0000] timerons (without recommendations)
[ 22.0000] timerons (with current solution)
[99.36%] improvement
--
--
-- LIST OF RECOMMENDED INDEXES
-- =====
-- index[1], 10.981MB
CREATE INDEX "INST23"."IDX1208291526220" ON "TEST"."HISTORY"
("BRANCH_ID" ASC, "TELLER_ID" ASC, "ACCTNAME" ASC,
"BALANCE" ASC, "ACCT_ID" ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK ;

```

The db2advis report includes a recommendation to create one new index. A CREATE INDEX statement is listed that is estimated to significantly reduce SQL execution costs.

The suggested new index should look similar to the following:

```
CREATE INDEX "INST23  "."IDX1208202015070" ON "TEST  "."HISTORY"
("BRANCH_ID" ASC, "TELLER_ID" ASC, "ACCTNAME" ASC,
"BALANCE" ASC, "ACCT_ID" ASC) ALLOW REVERSE SCANS COLLECT STATISTICS;
```

This index may be very efficient for processing this one SQL statement, but we will create a simple two column index, on BRANCH_ID and TELLER_ID, that will require less disk space.

We will start a second SQL Editor view in Data Server Manager to run the CREATE INDEX and RUNSTATS statements, using the file, *C:\inst23\ddl\create_testhist_ix.ddl*, that contains the following commands:

```
create index test.histix on test.history (branch_id, teller_id) ;
```

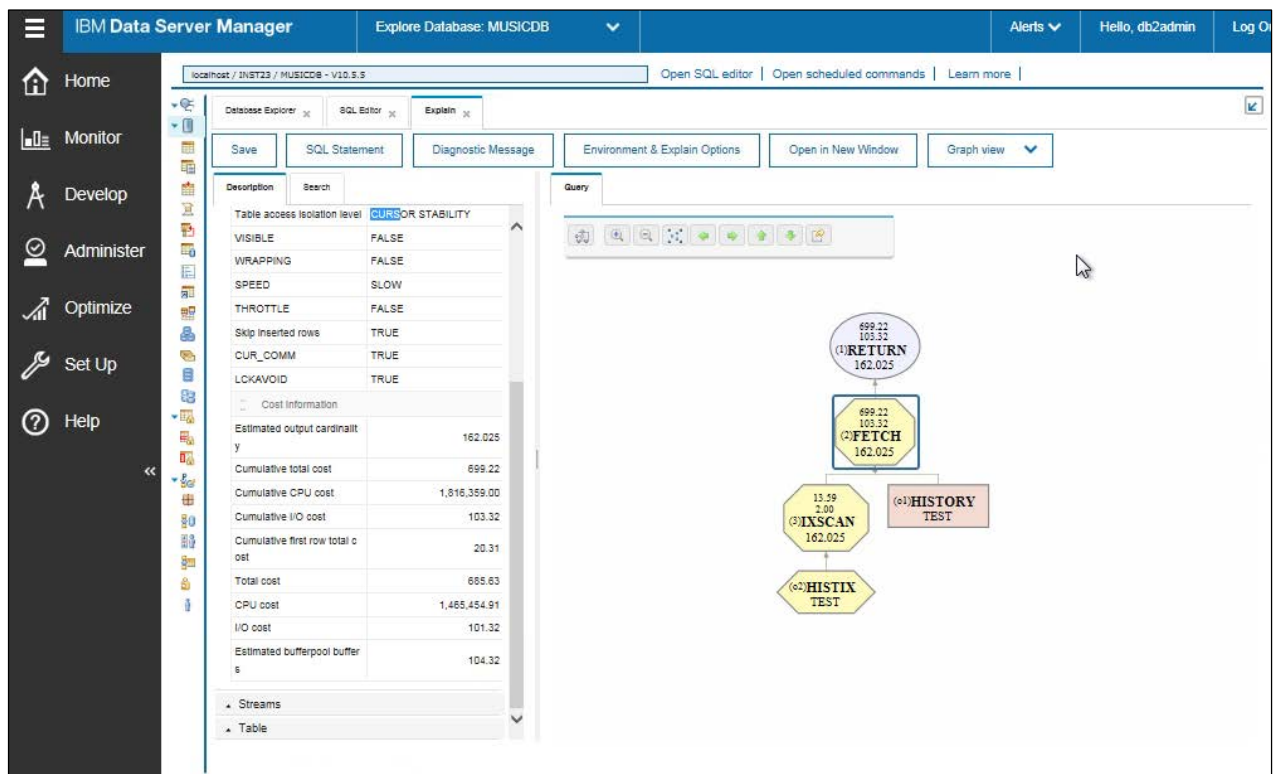
```
Call SYSPROC.ADMIN_CMD( 'runstats on table test.history and indexes all
' ) ;
```

2. Using **DSM**, click **Open SQL editor**.
3. Click **Upload** (which may be part of a list labelled *More Actions*). Use the **Browse** button for the Open SQL Script window to locate and select the file *C:\inst23\ddl\create_testhist_ix.ddl*.
4. Click **OK** to complete loading the SQL text into the SQL editor.
5. Review the CREATE INDEX statement and the ADMIN_CMD procedure call that is used to invoke the RUNSTATS processing.
6. Click **Run** and wait for the SQL statements to be processed.

Task 4. Use the Explain SQL capability of DSM to check the access plan and estimated costs with the new index in place.

We will be using the SQL in the file *C:\inst23\ddl\query_history.sql* which should still be in a SQL Editor view in DSM. If not, open a new SQL Editor and upload the SQL text from the file.

1. Click **Explain** and wait for the access plan report to get generated.
The new access plan utilizes the new index with an IXSCAN, or index scan operation, followed by the FETCH to retrieve data rows using the index pointers.
2. Select the **FETCH** operation and look at the cost statistics that appear.



Look at the following:

- Estimated Cardinality: _____ (test result 162)
- Cumulative Total Cost: _____ (test result 699)
- Cumulative I/O Cost : _____ (test result 103)

Your results may be slightly different, but a test result produced the estimated cardinality of 162 rows, and estimated I/O cost at 103.

These estimated costs are much lower than those required for the table scan in the previous access plan.

The relatively high I/O count, over 100, in order to read the estimated 162 rows suggests that the data we need spans many pages.

3. Close the **Explain** view and return to the SQL Editor.

Task 5. Reorganize a table using a REORG utility.

The DB2 REORG utility can reorganize, by sorting, a table's data rows to match the sequence of an index. This is called a reclustering reorganization.

This type of reorganization can improve the performance of applications that need to retrieve sets of rows using an index since the results may be located together in a relatively small number of pages.

We will use DSM to invoke the table reorganization.

1. Click the **Home** option on the left side of the DSM application.
You should see the database MUSICDB listed with some basic statistics on resource usage including, CPU, I/O and Memory. The database name MUSICDB provides a drop down list to perform different tasks.
2. Select **Administer: Explore Database** from the drop down list of the MUSICDB database.
You will be prompted for a userid and password for the MUSICDB database connection, use **inst23** with a password of **ibm2blue**.
A tree structure of database objects for this database is shown on the left side of the DSM view.
3. Click **Tables**.
The current table objects are listed.
4. Locate and select the table **TEST.HISTORY**. You may need to resize the schema and table columns.
5. Select **Reorganize** (which may be listed under the More Actions list), and then click **Next**.
6. Select these options:
 - Reorganization Method: **Classic**
 - Indexes, Table Spaces, and Dictionary: **Reorganize by using an existing index** (the only index, TEST.HISTIX, is listed); **Temporally store a copy of the reorganized table in a temporary table space**
 - *Leave other options with default values*

7. Review the result near the bottom of the window. The DSM tool generates the REORG TABLE command with a procedure call to ADMIN_CMD.

The command text should be similar to the following:

```
CALL SYSPROC.ADMIN_CMD ('REORG TABLE TEST.HISTORY INDEX TEST.HISTIX
USE TEMPSPACE1');
```

8. Click **Next**.
9. Select **Run with SQL Editor** for Run schedule, and then click **Finish**.
The result should show that the command processing succeeded.
10. **Close** the **Reorganize Table Data** tab.
We need to collect new table and index statistics, so that the DB2 Optimizer can accurately plan access to the newly reorganized table.
11. Click the **Database Explorer** tab.
12. Select **Collect Statistics** (which may be listed under the More Actions list).
13. Click **Next**. (if 'Next' does not appear, you may have to clear your browser History and Cache, and log in again after closing your browser)
14. Leave all of the options with default values, and then click **Next**.
15. Select **Run with SQL Editor** for the Run schedule, and then click **Finish**.
The result should show that the command processing succeeded.
16. Close the **Statistics Collection** tab.

Task 6. Use the Explain SQL capability of DSM to recheck the access plan and estimated costs after table reorganization.

We will be using the SQL in the file *C:\inst23\ddl\query_history.sql* which should still be in a SQL Editor view in DSM. If not, open a new SQL Editor and upload the SQL text from the file.

1. Click **Explain** to generate the access plan report.
The access plan utilizes the index with an IXSCAN, or index scan operation, followed by the FETCH to retrieve data rows using the index pointers. The new estimated costs are much lower with a reorganized table.

2. Select the **FETCH** operation and look at the cost statistics that appear.

The screenshot shows the IBM Data Server Manager interface. On the left is a navigation pane with options like Home, Monitor, Develop, Administer, Optimize, Set Up, and Help. The main area is titled 'Explore Database: MUSICDB'. It shows a tree view of database objects, with 'Tables' selected. The 'Query' tab is active, displaying a visual execution plan. The plan consists of several nodes: a 'RETURN' node at the top, followed by a 'FETCH' node (highlighted in yellow), then a 'HISTIX TEST' node, and finally a 'HISTORY TEST' node. Below the plan, a table provides cost statistics for the selected 'FETCH' operation.

Operator type	Description
Operator type	Sequential read
Type of prefetch	SEQUENTIAL, READ AHEAD
Maximum pages for prefetch	2
Table lock intent	INTENT SHARE
Row lock intent	SHARE (CS/RS)
Table access isolation level	CURSOR STABILITY
VISIBILITY	FALSE
WRAPPING	FALSE
SPEED	SLOW
THROTTLE	FALSE
Skip inserted rows	TRUE
CUR_COMM	TRUE
LOCKAVOID	TRUE
Estimated output cardinality	162.025
Cumulative total cost	33.20
Cumulative CPU cost	536,896.50

Look at the following:

- Estimated Cardinality: _____ (test result 162)
- Cumulative Total Cost: _____ (test result 33)
- Cumulative I/O Cost : _____ (test result 4.9)

Your results may be slightly different, but a test result produced the estimated cardinality of 162 rows with an estimated I/O cost at 4.9. These estimated costs are much lower than those required for the index scan in the previous access plan before the table was reorganized.

The relatively small count, about 5, shows that DB2 expects to find the rows in a small number of pages, which drastically reduces the estimated costs.

2 record(s) selected with 2 warning messages printed.

Results:

You used several DB2 tools and utilities to improve the performance of a SQL query. You invoked the Data Server Manager Visual Explain tool to review the access plans and the estimated costs for processing SQL statements. You used the DB2 design advisor to suggest a new index to reduce processing costs. You executed a DB2 REORG utility to reorganize a table to improve performance.

Unit 9 Locking and concurrency

IBM Training



Locking and concurrency

DB2 10.5 Administration Workshop for Windows

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the written permission of IBM.

Demonstration 1

Investigating DB2 locking

- Explain how SET ISOLATION can be used to select an isolation level for an application and the effect that selection can have on results returned for SQL statements.
- Use the db2pd commands to monitor locks being held by active applications and current lock wait conditions.
- Monitor a lock wait condition using the Data Server Manager application.
- Create a LOCKING event monitor to capture diagnostic information about lock related events.

Database maintenance, monitoring and problem determination

© Copyright IBM Corporation 2016

Demonstration 1: Investigating DB2 locking

Demonstration 1: Investigating DB2 locking

Purpose:

This demonstration uses several SQL scripts to demonstrate the types of locks used for processing SQL statements with different application isolation levels. You will create and analyze a lock wait condition. A LOCKING event monitor will be used to capture diagnostic data about lock related events.

Task 1. Monitor the locks acquired for a SQL UPDATE.

1. Logon to the Windows system using the user id **inst23**, with a password of **ibm2blue**.

The DB2 command line processor will be used to issue some DB2 and system commands. You will need to start the DB2 command window in administrator mode to issue these DB2 commands.

2. To start the DB2 command window, click on the Windows start icon, then Navigate to **All programs > IBM DB2 DB2COPY1 > DB2 Command Window - Administrator**. When prompted with the question, 'Do you want to allow the following program to make changes to this computer?' select **Yes**.

In this section, you will run a SQL script that updates a DB2 table, MUSIC.STOCK to understand the locks acquired for the UPDATE processing. We will also observe the effect those locks can have on applications that need to read the updated rows before the changes are committed.

The DB2 Command Line Processor can execute a series of SQL statements with auto-commit turned off (the +C option). This allows the locks to be held longer and is useful to our testing purposes.

We will grant SELECT access to the table MUSIC.STOCK by a user logon **user23**. We will use this user to access the table for read purposes while a portion of the table is locked for updates.

This task will be performed using the DB2 command line processor.

3. Issue the following series of commands using the DB2 command line processor.

- **cd c:\inst23\ddl**
- **db2 connect to musicdb**
- **db2 grant select on table music.stock to user user23**
- **db2 +C -tvf stock_update.sql**

The SQL statements show a sum from the QTY column for a set of rows before and after the UPDATE processing.

The output will look similar to the following:

```
SELECT sum(qty) from music.stock where itemno=10
1
-----
109
1 record(s) selected.
update music.stock set qty = qty + 1 where itemno between 10 and 15
DB20000I The SQL command completed successfully.
SELECT sum(qty) from music.stock where itemno=10
1
-----
112
1 record(s) selected.
```

The **db2pd** command can be used to show the locks currently held by DB2 applications for a database.

4. Issue the following series of commands using the DB2 command line processor.

- **db2pd -db musicdb -locks | more**

The command output will look similar to the following:

```
Database Member 0 -- Database MUSICDB -- Active -- Up 0 days 00:01:49 -- Date
2015-09-18-10.10.14.527000
```

Locks:

Address	TranHdl	Lockname	Type	Mode	Sts
Owner	Dur HoldCount Att	ReleaseFlg rrIID			
0x0000000001131F80	3	090004002800010000000000052	RowLock	..X	G 3
1 0	0x00200000 0x40000000 0				
0x0000000001137280	3	090005001500000000000000052	RowLock	..X	G 3
1 0	0x00200008 0x40000000 0				
0x0000000001132600	3	090004002000010000000000052	RowLock	..X	G 3
1 0	0x00200000 0x40000000 0				
0x0000000001136A00	3	090005000D00000000000000052	RowLock	..X	G 3
1 0	0x00200008 0x40000000 0				
0x0000000001132C00	3	090004002B00010000000000052	RowLock	..X	G 3
1 0	0x00200000 0x40000000 0				
0x0000000001132E00	3	090005000500000000000000052	RowLock	..X	G 3
1 0	0x00200008 0x40000000 0				
0x0000000001131980	3	4141414141664164FE8BC714C1	PlanLock	..S	G 3
1 0	0x00000000 0x40000000 0				
0x0000000001131880	3	090004002300010000000000052	RowLock	..X	G 3
1 0	0x00200000 0x40000000 0				
0x0000000001136D00	3	090005001000000000000000052	RowLock	..X	G 3
1 0	0x00200008 0x40000000 0				
0x0000000001132800	3	090004001B00010000000000052	RowLock	..X	G 3
1 0	0x00200000 0x40000000 0				

0x0000000001136500	3	0900050008000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001131E80	3	0900040026000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				
0x0000000001137080	3	0900050013000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001132100	3	090004001E000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				
0x0000000001136800	3	090005000B000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001132A00	3	0900040029000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				
0x0000000001132300	3	0900040021000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				
0x0000000001136B00	3	090005000E000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001132D00	3	090004002C000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				
0x0000000001132F00	3	0900050006000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001131680	3	0900040024000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				
0x0000000001136E00	3	0900050011000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001132400	3	090004001C000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				
0x0000000001136600	3	0900050009000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001131B80	3	0900040027000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				
0x0000000001137180	3	0900050014000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001132200	3	090004001F000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				
0x0000000001136900	3	090005000C000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001132B00	3	090004002A000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				
0x0000000001132500	3	0900050004000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001131C80	3	0900040022000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				
0x0000000001136C00	3	090005000F000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001136400	3	0900050007000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001131D80	3	0900040025000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				
0x0000000001136F00	3	0900050012000000000000000052	RowLock	..X	G	3
1 0	0x00200008	0x40000000 0				
0x0000000001131A80	3	090004001D000100000000000052	RowLock	..X	G	3
1 0	0x00200000	0x40000000 0				

```

0x0000000001136700 3          090005000A00000000000000052 RowLock      ..X  G   3
1    0          0x00200008 0x40000000 0
0x0000000001132900 3          090005000000000000000000054 TableLock    .IX  G   3
1    0          0x00202000 0x40000000 0
0x0000000001132700 3          090004000000000000000000054 TableLock    .IX  G   3
1    0          0x00202000 0x40000000 0

```

5. Review the list of locks that were acquired to execute the SQL UPDATE.

- You should see a series of Exclusive (X) row locks and several table locks.
- You should see two Intent Exclusive (IX) table locks.

The UPDATE SQL statement updated the table MUSIC.STOCK, so one Intent Exclusive (IX) table lock is needed for that table. We defined the table MUSIC.STOCK as a System period temporal table with a history table of MUSIC.STOCK_HISTORY. The UPDATE of MUSIC.STOCK needs to update the system history table, so an additional IX lock is needed for MUSIC.STOCK_HISTORY.

Task 2. Processing SQL SELECT statements using different locking isolation levels.

The execution of the stock update script in the previous task acquired a set of locks that were not released by the DB2 command line processor. We could use the COMMIT WORK statement to release those locks, but we want to run several SQL SELECT scripts that use different locking isolation levels to see how an application might be impacted by locks held by other applications running at the same time.

We will need a second DB2 command line processor session to be able to start a second database connection that can execute SQL scripts while the first session continues to hold its row and table locks. We will connect to the MUSICDB database using a different system userid, **user23**, to make it easier to monitor the locking for each connection.

We will create a LOCKING event monitor to capture information about lock waits and lock timeouts. On your Windows Database Server, a script file named create_lock_monitor.sql contains the statements to create a LOCKING event monitor and to activate the monitor. We will need to update the database configuration to generate the diagnostic locking data that will be captured.

1. Issue the following series of commands using the DB2 command line processor:

- **db2 update db cfg using mon_lockwait without_hist**
- **db2 update db cfg using mon_locktimeout without_hist**
- **db2 -tvf create_lock_monitor.sql**

The command output will look similar to the following:

```
create event monitor mon_locks for locking write to table autostart
DB20000I  The SQL command completed successfully.
```

```
set event monitor mon_locks state 1
DB20000I  The SQL command completed successfully.
```

```
select varchar(tabschema,10) as schema, varchar(tabname,40) as mon_table from
syscat.eventtables where evmonname = 'MON_LOCKS'
```

SCHEMA	MON_TABLE
INST23	CONTROL_MON_LOCKS
INST23	LOCK_MON_LOCKS
INST23	LOCK_PARTICIPANTS_MON_LOCKS
INST23	LOCK_PARTICIPANT_ACTIVITIES_MON_LOCKS
INST23	LOCK_ACTIVITY_VALUES_MON_LOCKS

2. Issue the following series of commands using the DB2 command line processor:

- **db2 connect to musicdb**
- **db2 +C -tvf stock_update.sql**

3. Start a **second** DB2 command window, click on the Windows start icon, then Navigate to **All programs > IBM DB2 DB2COPY1 > DB2 Command Window - Administrator**. When prompted with the question, 'Do you want to allow the following program to make changes to this computer?' select **Yes**.

4. Using the **second DB2 command window**, issue the following series of commands:

- **cd C:\inst23\ddl**
- **db2 connect to musicdb user user23 using ibm2blue**

We will start by running the SQL script, *stock_select_ur.sql*. The script starts by setting the special registers CURRENT LOCK TIMEOUT and CURRENT ISOLATION. The statement **SET CURRENT ISOLATION ur** sets the locking isolation level to Uncommitted Read, the lowest level for isolation. This allows the connection to access data rows updated by other applications that have not committed those changes.

5. Using the **second DB2 command window**, issue the following series of commands:

- **db2 -tvf stock_select_ur.sql**

The command output will look similar to the following:

```
set current lock timeout 300
DB20000I  The SQL command completed successfully.

set current isolation ur
DB20000I  The SQL command completed successfully.

select sum(qty) from music.STOCK where itemno = 10

1
-----
      115

1 record(s) selected.
```

The result for the sum(QTY) using isolation level UR reflects the changes made by the uncommitted UPDATE in the other CLP session.

Next we will run the SQL script, *stock_select_cs.sql*. The script starts by setting the special registers CURRENT LOCK TIMEOUT and CURRENT ISOLATION. The statement **SET CURRENT ISOLATION cs**, sets the locking isolation level to Cursor Stability, the default level for isolation. This does not allow the connection to access data rows updated by other applications that have not committed those changes, but DB2 can use special log records to retrieve the data rows contents before the changes were made.

6. Using the **second DB2 command window**, issue the following series of commands:

- **db2 -tvf stock_select_cs.sql**

The command output will look similar to the following:

```
set current lock timeout 300
DB20000I  The SQL command completed successfully.

set current isolation cs
DB20000I  The SQL command completed successfully.

select sum(qty) from music.STOCK where itemno = 10

1
-----
      112

1 record(s) selected.
```

The result for the sum(QTY) using isolation level CS *does not* include the changes made by the uncommitted UPDATE in the other CLP session.

Next we will run the SQL script, *stock_select_rs.sql*. The script starts by setting the special registers CURRENT LOCK TIMEOUT and CURRENT ISOLATION. The statement **SET CURRENT ISOLATION rs**, sets the locking isolation level to Read Stability. This isolation level requires row locks for any row of data included in the SQL result. Since the UPDATE executed in the first CLP session was not committed, the SELECT for that data in this second session will cause a lock wait condition.

7. Using the **second DB2 command window**, issue the following series of commands:

- **db2 connect to musicdb user user23 using ibm2blue**
- **db2 -tvf stock_select_rs.sql**

The SELECT SQL statement cannot complete because row locks are needed to produce the result that are held by the other CLP session.

We can use the **db2pd** command to look for active lock waits, but we will need to run the command from the first CLP session.

8. Switching back to the **first DB2 command window**, issue the following series of commands:

- **db2pd -db musicdb -wlock**

The command output will look similar to the following:

```
Database Member 0 -- Database MUSICDB -- Active -- Up 0 days 00:12:13 -- Date
2015-09-18-10.20.38.667000
```

Locks being waited on :

AppHndl	[nod-index]	TranHdl	Lockname	Type	Mode	Conv
Sts	CoorEDU	AppName	AuthID	AppID		
23254	[000-23254]	3	090004001B000100000000000052	RowLock	..X	G
6700	db2bp.ex	INST23	*LOCAL.INST23.150918140903			
23265	[000-23265]	12	090004001B000100000000000052	RowLock	.NS	W
236	db2bp.ex	USER23	*LOCAL.INST23.150918141210			

The db2pd command output shows that a row lock is held (G) by one user connection, inst23, but a NS read lock is waiting (W) for the connection user user23.

We can use the -lock report of **db2pd** to see the full list of locks held by the two connections, including the lock that is being waited for.

9. Using the **first DB2 command window**, issue the following series of commands:

- **db2pd -db musicdb -lock**

The command output will look similar to the following:

```
Database Member 0 -- Database MUSICDB -- Active -- Up 0 days 00:12:40 -- Date
2015-09-18-10.21.05.386000
```

Locks:

Address	TranHdl	Lockname	Type	Mode	Sts
Owner	Dur HoldCount	Att	ReleaseFlg rrIID		
0x0000000001131F80	3	0900040028000100000000000052	RowLock	..X	G 3
1 0	0x00200000	0x40000000 0			
0x0000000001137280	3	0900050015000000000000000052	RowLock	..X	G 3
1 0	0x00200008	0x40000000 0			
0x0000000001132600	3	0900040020000100000000000052	RowLock	..X	G 3
1 0	0x00200000	0x40000000 0			
0x0000000001136A00	3	090005000D000000000000000052	RowLock	..X	G 3
1 0	0x00200008	0x40000000 0			
0x0000000001132C00	3	090004002B000100000000000052	RowLock	..X	G 3
1 0	0x00200000	0x40000000 0			
0x0000000001132E00	3	0900050005000000000000000052	RowLock	..X	G 3
1 0	0x00200008	0x40000000 0			
0x0000000001131980	3	4141414141664164FE8BC714C1	PlanLock	..S	G 3
1 0	0x00000000	0x40000000 0			

0x000000000113B080 12	4141414141664164FE8BC714C1	PlanLock	..S	G	
12 1 0	0x00000000 0x40000000 0				
0x0000000001131880 3	090004002300010000000000052	RowLock	..X	G	3
1 0	0x00200000 0x40000000 0				
0x0000000001136D00 3	090005001000000000000000052	RowLock	..X	G	3
1 0	0x00200008 0x40000000 0				
0x0000000001132800 3	090004001B00010000000000052	RowLock	..X	G	3
1 0	0x00200000 0x40000000 0				
0x0000000001139A00 12	090004001B00010000000000052	RowLock	.NS	W	3
0 0	0x00000000 0x00000000 0				
0x0000000001136500 3	090005000800000000000000052	RowLock	..X	G	3
1 0	0x00200008 0x40000000 0				
0x0000000001131E80 3	090004002600010000000000052	RowLock	..X	G	3
1 0	0x00200000 0x40000000 0				
0x0000000001137080 3	090005001300000000000000052	RowLock	..X	G	3
1 0	0x00200008 0x40000000 0				
0x0000000001132100 3	090004001E00010000000000052	RowLock	..X	G	3
1 0	0x00200000 0x40000000 0				
0x0000000001136800 3	090005000B00000000000000052	RowLock	..X	G	3
1 0	0x00200008 0x40000000 0				
0x0000000001132A00 3	090004002900010000000000052	RowLock	..X	G	3
1 0	0x00200000 0x40000000 0				
0x0000000001132300 3	090004002100010000000000052	RowLock	..X	G	3
1 0	0x00200000 0x40000000 0				
0x0000000001136B00 3	090005000E00000000000000052	RowLock	..X	G	3
1 0	0x00200008 0x40000000 0				
0x000000000113AE00 12	01000000030000000100A032D6	VarLock	..S	G	
12 1 0	0x00000000 0x40000000 0				
0x0000000001132D00 3	090004002C00010000000000052	RowLock	..X	G	3
1 0	0x00200000 0x40000000 0				
0x0000000001132F00 3	090005000600000000000000052	RowLock	..X	G	3
1 0	0x00200008 0x40000000 0				
0x0000000001131680 3	090004002400010000000000052	RowLock	..X	G	3
1 0	0x00200000 0x40000000 0				
0x0000000001136E00 3	090005001100000000000000052	RowLock	..X	G	3
1 0	0x00200008 0x40000000 0				
0x0000000001132400 3	090004001C00010000000000052	RowLock	..X	G	3
1 0	0x00200000 0x40000000 0				
0x0000000001136600 3	090005000900000000000000052	RowLock	..X	G	3
1 0	0x00200008 0x40000000 0				
0x0000000001131B80 3	090004002700010000000000052	RowLock	..X	G	3
1 0	0x00200000 0x40000000 0				
0x0000000001137180 3	090005001400000000000000052	RowLock	..X	G	3
1 0	0x00200008 0x40000000 0				
0x0000000001132200 3	090004001F00010000000000052	RowLock	..X	G	3
1 0	0x00200000 0x40000000 0				
0x0000000001136900 3	090005000C00000000000000052	RowLock	..X	G	3
1 0	0x00200008 0x40000000 0				
0x0000000001132B00 3	090004002A00010000000000052	RowLock	..X	G	3
1 0	0x00200000 0x40000000 0				
0x0000000001132500 3	090005000400000000000000052	RowLock	..X	G	3
1 0	0x00200008 0x40000000 0				

```

0x0000000001131C80 3          090004002200010000000000052 RowLock      ..X  G   3
1  0          0x00200000 0x40000000 0
0x0000000001136C00 3          090005000F00000000000000052 RowLock      ..X  G   3
1  0          0x00200008 0x40000000 0
0x0000000001136400 3          090005000700000000000000052 RowLock      ..X  G   3
1  0          0x00200008 0x40000000 0
0x0000000001131D80 3          090004002500010000000000052 RowLock      ..X  G   3
1  0          0x00200000 0x40000000 0
0x0000000001136F00 3          090005001200000000000000052 RowLock      ..X  G   3
1  0          0x00200008 0x40000000 0
0x0000000001131A80 3          090004001D00010000000000052 RowLock      ..X  G   3
1  0          0x00200000 0x40000000 0
0x0000000001136700 3          090005000A00000000000000052 RowLock      ..X  G   3
1  0          0x00200008 0x40000000 0
0x0000000001132900 3          090005000000000000000000054 TableLock    .IX  G   3
1  0          0x00202000 0x40000000 0
0x0000000001132700 3          090004000000000000000000054 TableLock    .IX  G   3
1  0          0x00202000 0x40000000 0
0x0000000001139B00 12         090004000000000000000000054 TableLock    .IS  G
12          1  0          0x00003000 0x00000001 0

```

The SQL command file included the SET CURRENT LOCK TIMEOUT statement to limit the time that a SQL statement will wait for a lock, before a SQL error return code is generated.

10. In the **second DB2 command window**, wait for the lock timeout condition to occur and the DB2 error message to be returned.

The command output will look similar to the following:

```
set current lock timeout 180
```

```
DB20000I The SQL command completed successfully.
```

```
set current isolation rs
```

```
DB20000I The SQL command completed successfully.
```

```
select sum(qty) from music.STOCK where itemno = 10
```

```
SQL0911N The current transaction has been rolled back because of a deadlock
or timeout. Reason code "68". SQLSTATE=40001
```

Task 3. Monitor a DB2 database for lock wait conditions using the Data Server Manager.

In this section we use the Data Server Manager to monitor a lock wait condition for a DB2 database.

We are going to be using the two DB2 Command Line Processor sessions that were opened in the preceding tasks.

- One session uses the user inst23 to run a command file that updates the MUSIC.STOCK table and holds those locks.
- The second session uses the user user23 to select rows that were locked by the first connection

1. Using the **first DB2 command window**, issue the following series of commands:

- **db2 commit**
- **db2 connect to musicdb**
- **db2 +C -tvf stock_update.sql**

Next we will run the SQL script, *stock_select_rs.sql*. Since the UPDATE executed in the first CLP session was not committed, the SELECT for that data in this second session will cause a lock wait condition.

2. Using the **second DB2 command window**, issue the following series of commands:

- **db2 connect to musicdb user user23 using ibm2blue**
- **db2 -tvf stock_select_rs.sql**

The SELECT SQL statement cannot complete because row locks are needed to produce the result that are held by the other CLP session.

We will use the Data Server Manager tool to monitor lock wait conditions.

If no shortcut is provided, open the Internet Explorer Browser and type the following URL text to start DSM, **http://localhost:11080**.

When DSM was installed, a local user id and password were selected to be associated with the application. For this course the following were configured:

Windows user id: **db2admin**

User password: **ibm2blue**

3. Use the aforementioned userid and password to logon to DSM.
4. Click the **Monitor** option on the left side of the DSM application.
5. Select **Database** from the options. When prompted to select a database, select **MUSICDB**.

6. Click **Locking** from the menu options at the top.
The *Blocking and Waiting Connections* view shows that the user INST23 is the blocker
7. Use the arrow next to **Blocker** to expand the data and show the WAITER, user23.
8. Select the **Locked Objects with Waiting Connections** view.
The data shown indicates that a row lock for the MUSIC.STOCK table is the lock object for an active lock wait.
9. Select the **Connection Statistics** view.
The data shown lists show locking related statistics for all connections, including lock waits per minute and lock wait time percentage.
Wait for the lock timeout condition to occur and the DB2 error message to be returned for the second CLP session with user USER23.
You can use **Log Out** to exit the Data Server Manager.

Task 4. Analyze the data collected by a LOCKING event monitor for locking related issues.

Next we will use a SQL script to review the locking information saved by the LOCKING event monitor.

The file *query_lock_events.sql* contains the SELECT SQL statements that use the tables associated with the LOCKING event monitor.

The SQL script has three SQL SELECT statements.

```
select event_id , event_type, event_timestamp
      from lock_mon_locks ;
```

```
select participant_no,
      varchar(auth_id,10) as auth_id,
      varchar(appl_name,20) as appl_name,
      varchar(table_name,12) as tablename,
      varchar(table_schema,12) as tabschema,
      lock_object_type , participant_type , lock_status
from lock_participants_mon_locks
      where event_type='LOCKTIMEOUT' ;
```

```
select
      participant_no, effective_isolation,
      varchar(stmt_operation,20) as operation,
      varchar(stmt_text,50) as sql_text
```

```
from lock_participant_activities_mon_locks
where event_type='LOCKTIMEOUT' ;
```

The first query uses the LOCK_MON_LOCKS table to show the type of each locking event and the time stamp when the data was captured. There should be at least one LOCKTIMEOUT and one or more LOCKWAIT events.

The next query looks at the LOCK_PARTICIPANTS_MON_LOCKS table to see which user, application and table were associated with the lock timeout.

The third query uses the LOCK_PARTICIPANT_ACTIVITY_MON_LOCKS table to see the SQL text and isolation level for the application that received the lock timeout condition.

You should use the SET EVENT MONITOR command to stop collecting the locking data.

1. Using the **first DB2 command window**, issue the following series of commands:

- **db2 commit**
- **db2 connect to musicdb**
- **db2 set event monitor mon_locks state 0**
- **db2 -tvf query_lock_events.sql | more**

The command output will look similar to the following:

```
select event_id , event_type, event_timestamp from lock_mon_locks
```

```
EVENT_ID          EVENT_TYPE
EVENT_TIMESTAMP
```

```
-----
              1 LOCKWAIT
2015-12-03-13.05.57.075574

              2 LOCKTIMEOUT
2015-12-03-13.09.09.809327

              3 LOCKWAIT
2015-12-03-13.13.06.622011

              4 LOCKTIMEOUT
2015-12-03-13.16.24.172684
```

```
4 record(s) selected.
```

```
select participant_no, varchar(auth_id,10) as auth_id,
       varchar(appl_name,20) as appl_name,
```

```

    varchar(table_name,12) as tabname, varchar(table_schema,12) as tabschema,
    lock_object_type , participant_type , lock_status
from lock_participants_mon_locks where event_type='LOCKTIMEOUT'

```

PARTICIPANT_NO	AUTH_ID	APPL_NAME	TABNAME	TABSCHEMA	LOCK_OBJECT_TYPE	PARTICIPANT_TYPE	LOCK_STATUS
1	USER23	db2bp.exe	STOCK	MUSIC	REQUESTER	2	ROW
2	INST23	db2bp.exe	-	-	OWNER	-	-
1	USER23	db2bp.exe	STOCK	MUSIC	REQUESTER	2	ROW
2	INST23	db2bp.exe	-	-	OWNER	-	-

4 record(s) selected.

```

select participant_no, effective_isolation,
varchar(stmt_operation,20) as operation,
varchar(stmt_text,50) as sql_text
from lock_participant_activities_mon_locks
where event_type='LOCKTIMEOUT'

```

PARTICIPANT_NO	EFFECTIVE_ISOLATION	OPERATION	SQL_TEXT
1	RS	DML, Select (blockab select sum(qty) from music.STOCK where itemno = 10	
SQL0445W Value "DML, Select (blockable)" has been truncated. SQLSTATE=01004			
1	RS	DML, Select (blockab select sum(qty) from music.STOCK where itemno = 10	
SQL0445W Value "DML, Select (blockable)" has been truncated. SQLSTATE=01004			

2 record(s) selected with 2 warning messages printed.

Results:

You created and analyzed a lock wait condition. A LOCKING event monitor was used to capture diagnostic data about lock related events.

Unit 10 Security

IBM Training

IBM

Security

DB2 10.5 Administration Workshop for Windows

© Copyright IBM Corporation 2016
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Demonstration 1

Database security

- Describe the default privileges that were granted to PUBLIC and the database creator when the database was created.
- Implement a SECADM user for the database that does not have either SYSADM or DBADM authorities
- Grant privileges to individuals, groups or roles
- Compare the privileges that can be used to run utilities like LOAD, IMPORT or RUNSTATS.

Database maintenance, monitoring and problem determination

© Copyright IBM Corporation 2016

Demonstration 1: Database security

Demonstration 1: Database security

Purpose:

This demonstration allows the student to manage the security privileges of the DB2 database to support different types of users. A security administrator with the SECADM authority will be implemented. Another user will be given the DBADM authority for this database. An application developer will be granted specific privileges to support their assignment. Several database roles will be created to provide access based on role membership rather than managing the authorizations at the user level.

Task 1. Granting specific privileges to a set of database users.

In previous exercises we used the DB2 instance owner **inst23**, to perform most of the tasks. As a member of the SYSADM group for the instance, we used this logon to create the MUSICDB database. As the database creator, **inst23** was granted the DBADM and SECADM authorities for the database. These combined privileges allowed this user to perform all of the administration tasks for the MUSICDB database.

We are going to assign specific authorities to a list of users to see how DB2 security checking is performed for processing DB2 commands and SQL statements.

We will use the following system user logons:

- **dba23** - will be granted DBADM authority, but will not be granted ACCESSCTRL
- **ctrl23** - will be granted SECADM authority as a new security administrator.
- **user23** - will be granted various specific privileges to perform selected tasks as a new developer.

1. Logon to the Windows system using the user id **inst23**, with a password of **ibm2blue**.

The DB2 command line processor will be used to issue some DB2 and system commands. You will need to start the DB2 command window in administrator mode to issue these DB2 commands.

2. To start the DB2 command window, click on the Windows start icon, then Navigate to **All programs > IBM DB2 DB2COPY1 > DB2 Command Window - Administrator**. When prompted with the question, 'Do you want to allow the following program to make changes to this computer?' select **Yes**.

We will grant the system user logon **ctrl23** the security administrator (SECADM) authority for the database MUSICDB. The database creator, **inst23**, is currently the only user with SECADM authority.

We can use the catalog view SYSCAT.DBAUTH to check current database authorizations. On your system a script file named *select_dbauth.sql* contains the query statement.

The file *select_dbauth.sql* contains the following SQL text:

```
select varchar(grantee,12) as grantee,
       varchar(grantor,12) as grantor,
       connectauth,loadauth,dbadmauth,securityadmauth
from syscat.dbauth ;
```

This task will be performed using the DB2 command line processor.

3. Issue the following series of commands using the DB2 command line processor.

- **cd c:\inst23\ddl**
- **db2 connect to musicdb**
- **db2 grant secadm on database to user ctrl23**
- **db2 -tvf select_dbauth.sql**

The output will look similar to the following:

GRANTEE	GRANTOR	CONNECTAUTH	LOADAUTH	DBADMAUTH	SECURITYADMAUTH
-----	-----	-----	-----	-----	-----
INST23	SYSIBM	N	N	Y	Y
PUBLIC	SYSIBM	Y	N	N	N
CTRL23	INST23	N	N	N	Y

3 record(s) selected.

We will need a second DB2 command line processor session to be able to start a second database connection that will be used by the security administrator, **ctrl23**. We will connect to the MUSICDB database using a different system userid, **ctrl23**, to make it easier to manage the security privileges for the database.

4. Start a **second** DB2 command line window; click on the Windows start icon, then Navigate to **All programs > IBM DB2 DB2COPY1 > DB2 Command Window - Administrator**. When prompted with the question, 'Do you want to allow the following program to make changes to this computer?' select **Yes**.

5. Using the **second DB2 command line window**, issue the following series of commands.

- **cd C:\inst23\ddl**
- **db2 connect to musicdb user ctrl23 using ibm2blue**

Rather than grant the DBADM authority directly to the user dba23, we will create a new database role named dba_role. This role can be used to define a set of database authorizations that new staff can utilize as needed. The WITHOUT ACCESSCTRL option will be used to provide broad access to table data but not allow users of this role to grant and revoke privileges. The new security administrator can perform this work.

6. Using the **second DB2 command line window**, issue the following series of commands:

- **db2 create role dba_role**
- **db2 grant dba_role to user dba23**
- **db2 grant dbadm without accessctrl on database to role dba_role**

The user system user **user23** does not have the SELECT authority for the table MUSIC.ALBUMS. We could grant the authority to the user directly, but we will grant the access to the developer role **dev_role**, so all the developers can perform the access if needed. We will grant the SELECT privilege on the table MUSIC.ALBUMS to the role dev_role.

7. Using the **second DB2 command line window**, issue the following series of commands:

- **db2 create role dev_role**
- **db2 grant dev_role to user user23**
- **db2 grant load on database to role dev_role**
- **db2 -tvf select_dbauth.sql**

The output should look similar to the following:

GRANTEE	GRANTOR	CONNECTAUTH	LOADAUTH	DBADMAUTH	SECURITYADMAUTH
INST23	SYSIBM	N	N	Y	Y
PUBLIC	SYSIBM	Y	N	N	N
CTRL23	INST23	N	N	N	Y
DBA_ROLE	CTRL23	N	N	Y	N
DEV_ROLE	CTRL23	N	Y	N	N

5 record(s) selected.

Task 2. Use the newly defined security privileges to create and access database objects.

Next we will use the new database administrator id, **dba23** to access our database.

We will start a second terminal session that the new database administrator, dba20, can use.

We will need a **third** DB2 command line processor session to be able to start a third database connection that can execute SQL scripts and SQL statements. We will connect to the MUSICDB database using a different system userid, **dba23**.

1. Start a **third** DB2 command window; click on the Windows start icon, then Navigate to **All programs > IBM DB2 DB2COPY1 > DB2 Command Window - Administrator**. When prompted with the question, 'Do you want to allow the following program to make changes to this computer?' select **Yes**.
2. Using the **third DB2 command line window**, issue the following series of commands:

- **cd C:\inst23\ddl**
- **db2 connect to musicdb user dba23 using ibm2blue**

Try to create a new table space using this database user.

3. Using the **third DB2 command line window**, issue the following command:
 - **db2 create tablespace testdata**

The CREATE TABLESPACE fails with a SQL0552N message. The DBADM database authority does not allow a user to create new table spaces. Currently the **inst23** user would need to perform that task.

The user dba23 can be used to create a new table and load data into the new table.

Issue the following SQL statements using the third terminal session connected as dba23.

4. Using the **third DB2 command line window**, issue the following series of commands:
 - **db2 create table test.albums like music.albums in userspace1**
 - **db2 “export to album.del of del select * from music.albums”**
 - **db2 “import from album.del of del insert into test.albums”**
 - **db2 runstats on table test.albums**
 - **db2 runstats on table music.albums**

As a DBADM user, we were able to access the table MUSIC.ALBUMS, and execute RUNSTATS commands for any table.

The user named user23 was placed in a role for developers named dev_role. We can run some SQL statements using the user23 user to test its ability to access data and perform development tasks.

Issue the following SQL statements using the third terminal session connected as dba23.

5. Using the **third DB2 command line window**, issue the following series of commands:

- **db2 connect to musicdb user user23 using ibm2blue**
- **db2 create table test.album2 like music.albums in tsp01**

The CREATE TABLE fails with a SQL0551N message. The user user23 does not have the USE authority for the tablespace TSP01. When the database was created, USE authority for the tablespace USERSPACE1 was granted to PUBLIC. Create the new test table using the USERSPACE1 table space.

- **db2 create table test.album2 like music.albums in userspace1**

Now attempt to copy some rows from the table MUSIC.ALBUMS to the new test table.

6. Using the **third DB2 command line window**, issue the following command:

- **db2 “insert into test.album2 select * from music.albums where artno = 42 “**

The INSERT fails with a SQL0551N message

The user user23 does not have the SELECT authority for the table MUSIC.ALBUMS. We could grant the authority to the user directly, but we will grant the access to the developer role **dev_role**, so all the developers can perform the access if needed. The security administrator will perform the task.

7. Using the **second DB2 command line window**, issue the following series of commands:

- **db2 grant select on table music.albums to role dev_role**
- **db2 grant select,update on table test.albums to role dev_role**

We run some additional SQL statements using the **user23** user to test its ability to access data in the database.

8. Using the **third DB2 command line window**, issue the following series of commands:
- **db2 connect to musicdb user user23 using ibm2blue**
 - **db2 "insert into test.album2 select * from music.albums where artno = 42"**

The INSERT is now successful. The user user23 now has the authority to SELECT from the table MUSIC.ARTISTS as a member of the role **dev_role**. As the creator of the table TEST.ALBUM2 the user has all SQL privileges for that table.

Next run the SQL statements in the script file *test_user23.sql*.

9. Using the **third DB2 command line window**, issue the following series of commands:

- **db2 connect to musicdb user user23 using ibm2blue**
- **db2 -tvf test_user23.sql | more**

The output should look similar to the following:

```
select title from test.albums where itemno =97
```

```
TITLE
```

```
-----
```

```
1962 - 1966
```

```
1 record(s) selected.
```

```
update test.albums set artno = 1 where itemno = 97
```

```
DB20000I The SQL command completed successfully.
```

```
delete from test.album2 where itemno = 97
```

```
DB20000I The SQL command completed successfully.
```

```
delete from test.albums where itemno = 97
```

```
DB21034E The command was processed as an SQL statement because it was not a
```

```
valid Command Line Processor command. During SQL processing it returned:
```

```
SQL0551N "USER23" does not have the required authorization or privilege to perform operation "DELETE" on object "TEST.ALBUMS". SQLSTATE=42501
```



```
runstats on table test.album2
```

```
DB20000I  The RUNSTATS command completed successfully.
```

```
runstats on table test.albums
```

```
DB20000I  The RUNSTATS command completed successfully.
```

```
drop table test.album2
```

```
DB20000I  The SQL command completed successfully.
```

The only statement execution that failed security tests was the DELETE from the table TEST.ALBUMS. The user **user23**, as a member of the role **dev_role** was able to perform SELECT and UPDATE access to the table TEST.ALBUMS that was created by the user dba23. As creator of the table TEST.ALBUM2, the user **user23** could perform any type of access and could drop the table.

Now that we have created and configured the new DB2 instance, **inst23**, we will use that instance to create a new database.

Results:

You managed the security privileges of the DB2 database to support different types of users. A security administrator with the SECADM authority was implemented. Another user was given the DBADM authority for this database. An application developer was granted specific privileges to support their assignment. Several database roles were created to provide access based on role membership rather than managing the authorizations at the user level.



IBM Training

