Hello World: DB2 for Linux, UNIX, and Windows

Introduction to basic features and concepts

Skill Level: Introductory

Claire Hong
Advisory Software Developer
IBM Toronto Lab

Renata Kupresak
Staff Software Developer
IBM Toronto Lab

11 Dec 2006

This tutorial provides high-level overviews of various IBM software products. You'll get an introduction to some introduction to some of the basic features and concepts in DB2® for Linux®, UNIX® and Windows® and learn you how to implement a database in DB2 environment. The exercises offer step-by-step instructions on how to create a database, how to create DB2 objects such as tables, indexes, views and constraints, how to get data in and out of the database, and how to query the data in the database.

Section 1. Before you start

About this series

This series is for software developers who want high-level overviews of IBM software products. The modules are designed to introduce the products, and draw your interest for further exploration. The exercises cover only the basic concepts, but are enough to get you started.

About this tutorial

This tutorial shows you how to implement a database in DB2 environment. The exercises offer step-by-step instructions on how to create a database, how to create DB2 objects such as tables, indexes, views and constraints, how to get data in and out of the database, and how to query the data in the database.

Objectives

At the end of this tutorial, you will be able to create a database and other objects, such as tablespaces, tables, indexes and views.

Prerequisites

This tutorial is designed for developers with minimum or no knowledge of DB2.

System requirements

Before starting this tutorial, make sure DB2 Express 9 is installed. Download a no charge version of DB2 Express-C 9.

To view the demos included in this tutorial, enable JavaScript in your browser and install Macromedia Flash Player 6 or higher. Download the latest Flash Player at http://www.macromedia.com/go/getflashplayer/.

Animated demos

If this is your first encounter with a developerWorks tutorial that includes demos, here are a few helpful hints:

- Demos are an optional way to see the same steps described in the tutorial. To see an animated demo, click the 🎞 Show me link. The demo opens in a new browser window.
- Each demo contains a navigation bar at the bottom of the screen. Use the navigation bar to pause, exit, rewind, or fast forward portions of the demo.
- The demos are 800 x 600 pixels. If this is the maximum resolution of your screen, or if your resolution is lower than this, scrolling is necessary to see some areas of the demo.

Watch the demo *before* you start the associated exercise to gain a clearer understanding of the tasks at hand.

Section 2. Introduction

What is DB2?

DB2 is a Relational Database Management System (RDBMS), part of the Information Management product family. DB2 is considered the first database product to use SQL (a language that provides an interface to an RDBMS). DB2 was first released in 1982 and is now available for a wide range of operating system platforms, including Linux, UNIX and Windows.

DB2 has different versions to meet a variety of needs:

- DB2 Everyplace provides mobile users with a version of DB2 with a small footprint and high performance.
- DB2 for z/OS provides all the features of DB2 for host systems.
- DB2 for Linux, UNIX and Windows comes in several flavors.
  - Enterprise Server Edition (ESE) is a complete RDBMS with client/server setup. DB2 ESE is intended for medium to large size businesses.
  - Workgroup Server Edition (WSE) is primarily intended for small to medium size businesses with all the features of DB2 ESE, except for mainframe connectivity.
  - Personal Edition provides a single-user deployment on a personal computer.
  - DB2 Express is the newest addition to the DB2 family and is available for download free of charge (if you did not download DB2 Express as directed in System requirements, go back and do so before beginning the tutorial). Express is for the DB2 community, offering the same core features and functions as DB2 WSE.

This tutorial uses DB2 Express, based on the newest release, DB2 9.

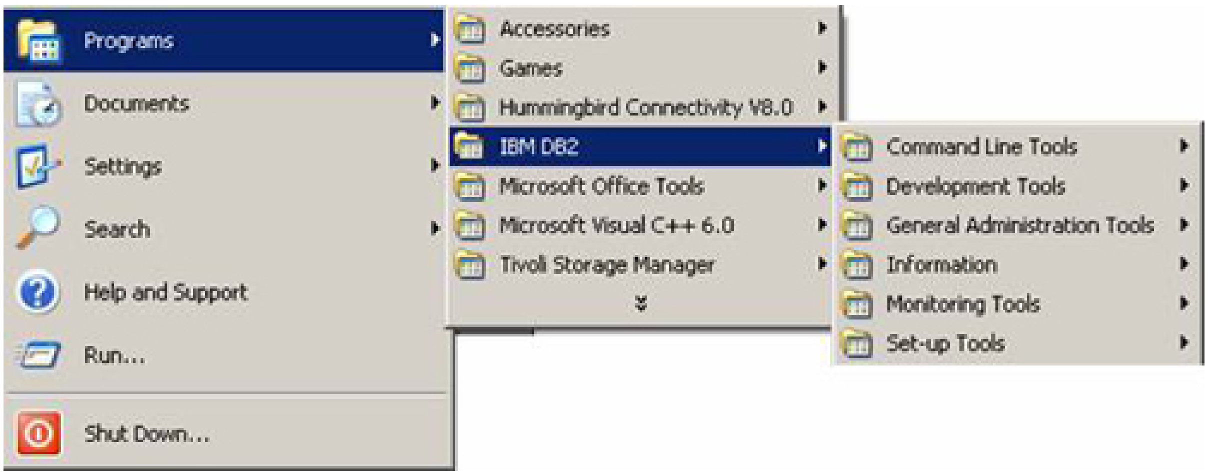
You can access DB2 databases from any application using Microsoft Open Database Connectivity (ODBC) and Java™ Database Connectivity (JDBC). The APIs are available for a wide range of programming languages, including C, C++, Java and FORTRAN. In this module you are not going to create any applications using APIs. See More information on APIs on the DB2 Information Center site (see Resources for a link).

A database can be used to store any kind of information, whether it is relational data, such as the names and addresses of your employees, or binary data, such as their identification photo. DB2 9 now also supports pureXML™ for easy storage and retrieval of XML data. Data can be loaded and maintained using available utilities or manipulated using queries to retrieve information you are interested in. You can control the integrity of the data with some of the tools described in this tutorial.

DB2 tools

DB2 offers many tools for daily administration and maintenance, which are categorized under Command Line Tools, General Administration Tools, Information, Monitoring Tools, and Set-up Tools. These tools are available either through the Windows Start menu as shown in Figure 1 or the Tools menu in the Control Center (CC), described in more detail later in this section. Each category is introduced briefly below.

Figure 1. DB2 GUI Tools



Command Line Tools
: Includes the Command Editor and Command Line Processor (CLP), each described in more detail later on, as well as the Command Window. The Command Window can be used in exactly the same way as the CLP, except all DB2 commands must be prefixed by 'db2' because the CLP is started in the interactive input mode and Command Window is not.

General Administration Tools
: Includes the Control Center (discussed shortly), Journal, Replication Center and Task Center. Journal can be used to view previous tasks, database actions, and messages. Replication Center allows users to administer the replication environment for the database. Finally, the Task Center can be used to schedule tasks in the form of scripts and notify others about the outcome.

Information
: Includes the Information Center, which can be used to locate help for any SQL or DB2 command and syntax trees.

Monitoring Tools
: Lets users monitor the state of your system either as a snapshot or constantly. With the Event Analyzer, users can follow the output of performance event monitors. With the Health Center, users can see the overall state and health of the database and receive alerts when resources, such as memory or locks, reach a predefined threshold. The Indoubt Transaction Manager allows users to perform necessary actions on *indoubt transactions* (a transaction which is left in an incomplete state due to a crash for example). The transaction can be committed, rolled back, or removed from the logs altogether. Finally, using the

Memory Visualizer, administrators can monitor memory-related health of the system and databases, such as current memory use and allocation.

Setup Tools

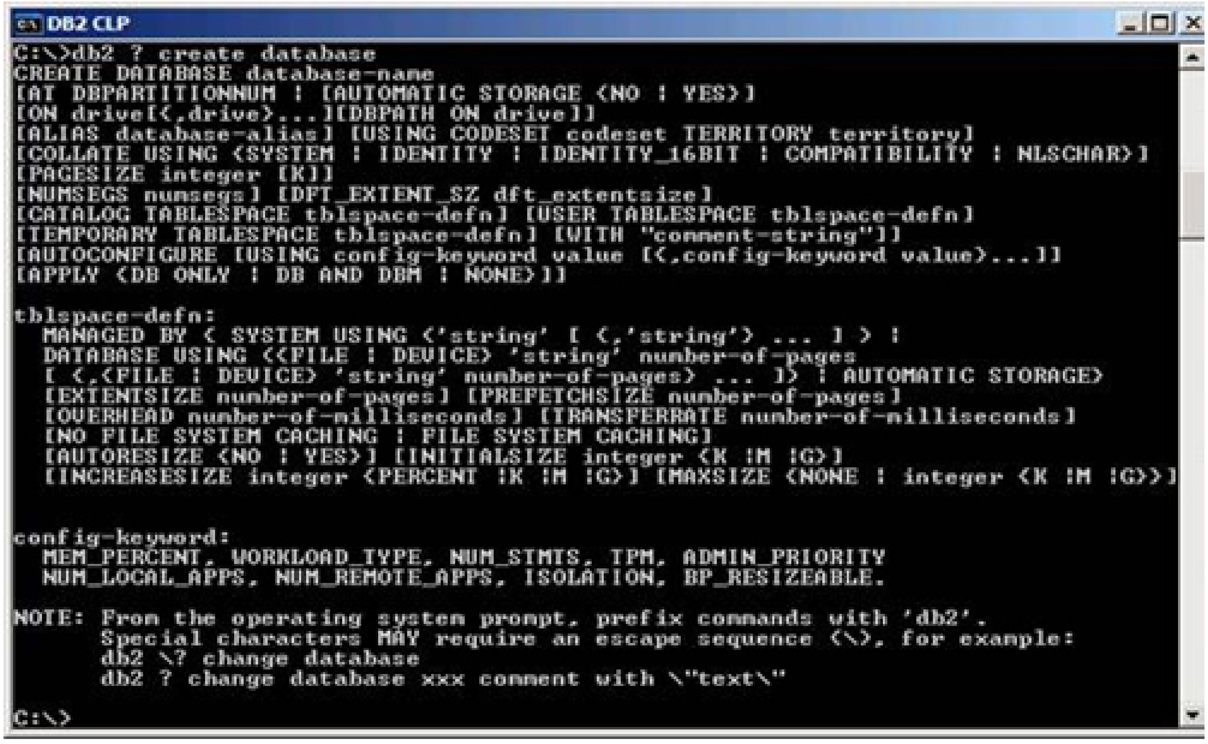
Helps users set up their system and databases. The Configuration Assistant helps users configure the databases so that they can be accessed from different clients or applications. First Steps contains links to the sources of information users need to learn about DB2. Under Set-up Tools users can also find the option to register their database development environment with Microsoft Visual Studio using their add-ins. Lastly, the Satellite Synchronizer lets users upload the satellite configuration information to its control server.

DB2 can be administered either from the text-based CLP or using Graphical User Interface (GUI). The Control Center, described below, is one of the main GUI administration tools, which allows users to perform various operations on DB2 objects, such as create and alter. Most of these tasks can also be performed using the Command Line Processor (CLP), depending on user's preference. The tools used in this tutorial are described in more detail in subsequent sections.

Command Line Processor

The Command Line Processor (CLP) is a text-based interface to DB2. It allows users to enter any command and receive the text output to the standard output. It also offers help on syntax for any command available from the command line as well as explanations of SQL codes. Following is an example of a DB2 command to find the syntax for the create database command:

Figure 2. DB2 command line processor

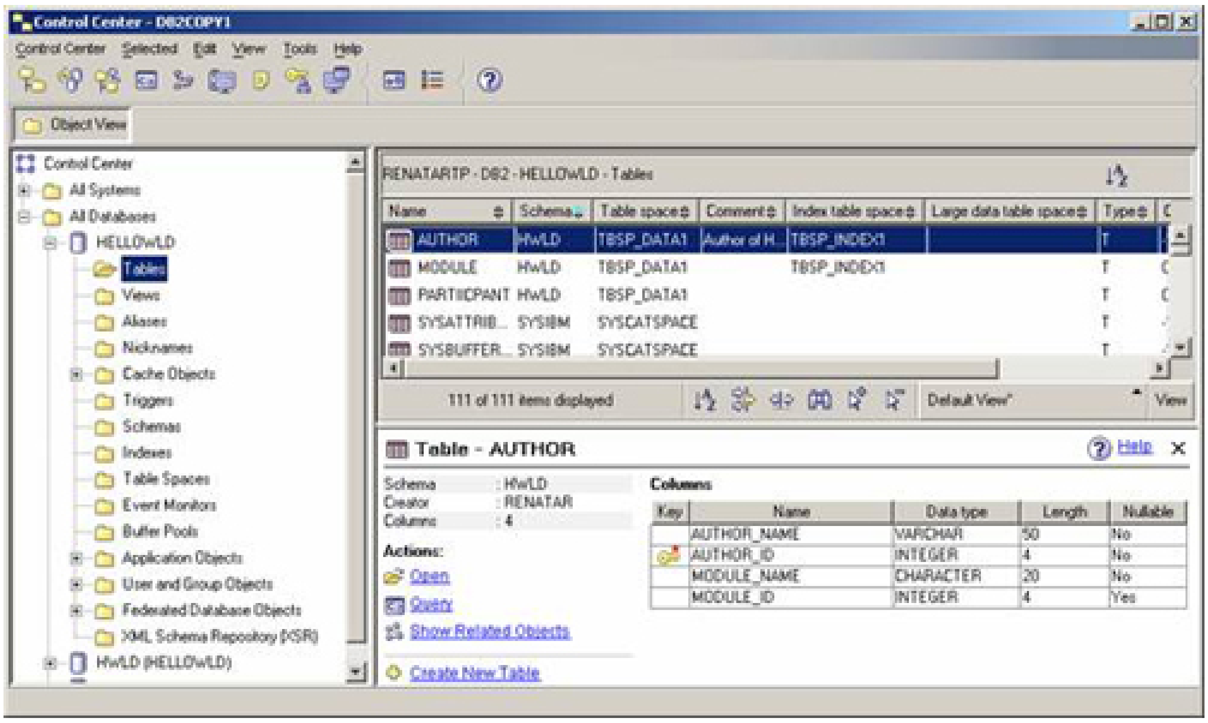


```
C:\>db2 ? create database
CREATE DATABASE database-name
[AT DBPARTITIONNUM | [AUTOMATIC STORAGE {NO | YES}]
[ON drive[{,drive}...][DBPATH ON drive]]
[ALIAS database-alias] [USING CODESET codeset TERRITORY territory]
[COLLATE USING {SYSTEM | IDENTITY | IDENTITY_16BIT | COMPATIBILITY | NLSCHAR}]
[PAGESIZE integer [K]]
[NUMSEGS numsegs] [DFT_EXTENT_SZ dft_extentsize]
[CATALOG TABLESPACE tblspace-defn] [USER TABLESPACE tblspace-defn]
[TEMPORARY TABLESPACE tblspace-defn] [WITH "comment-string"]]
[AUTOCONFIGURE [USING config-keyword value [{,config-keyword value}...]]
[APPLY {DB ONLY | DB AND DBM | NONE}]]

tblspace-defn:
  MANAGED BY { SYSTEM USING {'string' [ {,'string'} ... ] } |
  DATABASE USING {{FILE | DEVICE} 'string' number-of-pages
  [ {,{FILE | DEVICE} 'string' number-of-pages} ... ]} | AUTOMATIC STORAGE}
  [EXTENTSIZE number-of-pages] [PREFETCHSIZE number-of-pages]
  [OVERHEAD number-of-milliseconds] [TRANSFERRATE number-of-milliseconds]
  [NO FILE SYSTEM CACHING | FILE SYSTEM CACHING]
  [AUTORESIZE {NO | YES}] [INITIALSIZE integer {K |M |G}]
  [INCREASESIZE integer {PERCENT |K |M |G}] [MAXSIZE {NONE | integer {K |M |G}}]


config-keyword:
  MEM_PERCENT, WORKLOAD_TYPE, NUM_STMTS, TPM, ADMIN_PRIORITY
  NUM_LOCAL_APPS, NUM_REMOTE_APPS, ISOLATION, BP_RESIZEABLE.

NOTE: From the operating system prompt, prefix commands with 'db2'.
      Special characters MAY require an escape sequence (\), for example:
      db2 \? change database
      db2 ? change database xxx comment with \"text\"

C:\>
```

Control Center

The Control Center (CC) is a GUI-based tool, commonly used to manage and administer DB2 systems, databases, and objects. Using the Control Center, users can create and manage instances, databases and database objects, as well as configure and tune these instances and databases. The Control Center provides wizards to help users create or modify different objects. The Control Center provides various object views, which can be modified to suit users needs. For example, users may choose to filter only a set of actions that can be performed on an object, such as a table, which are relevant to their daily activities. Following is the Control Center, using a default view to see all objects in all databases:

Figure 3. Control Center with default view



Name	Schema	Table space	Comment	Index table space	Large data table space	Type
AUTHOR	HWLD	TBSP_DATA1	Author of H...	TBSP_INDEX1		T
MODULE	HWLD	TBSP_DATA1		TBSP_INDEX1		T
PARTIICPANT	HWLD	TBSP_DATA1				T
SYSATTRIB...	SYSIBM	SYSCATSPACE				T
SYSBUFFER...	SYSIBM	SYSCATSPACE				T

Schema	: HWLD
Creator	: RENATAR
Columns	: 4

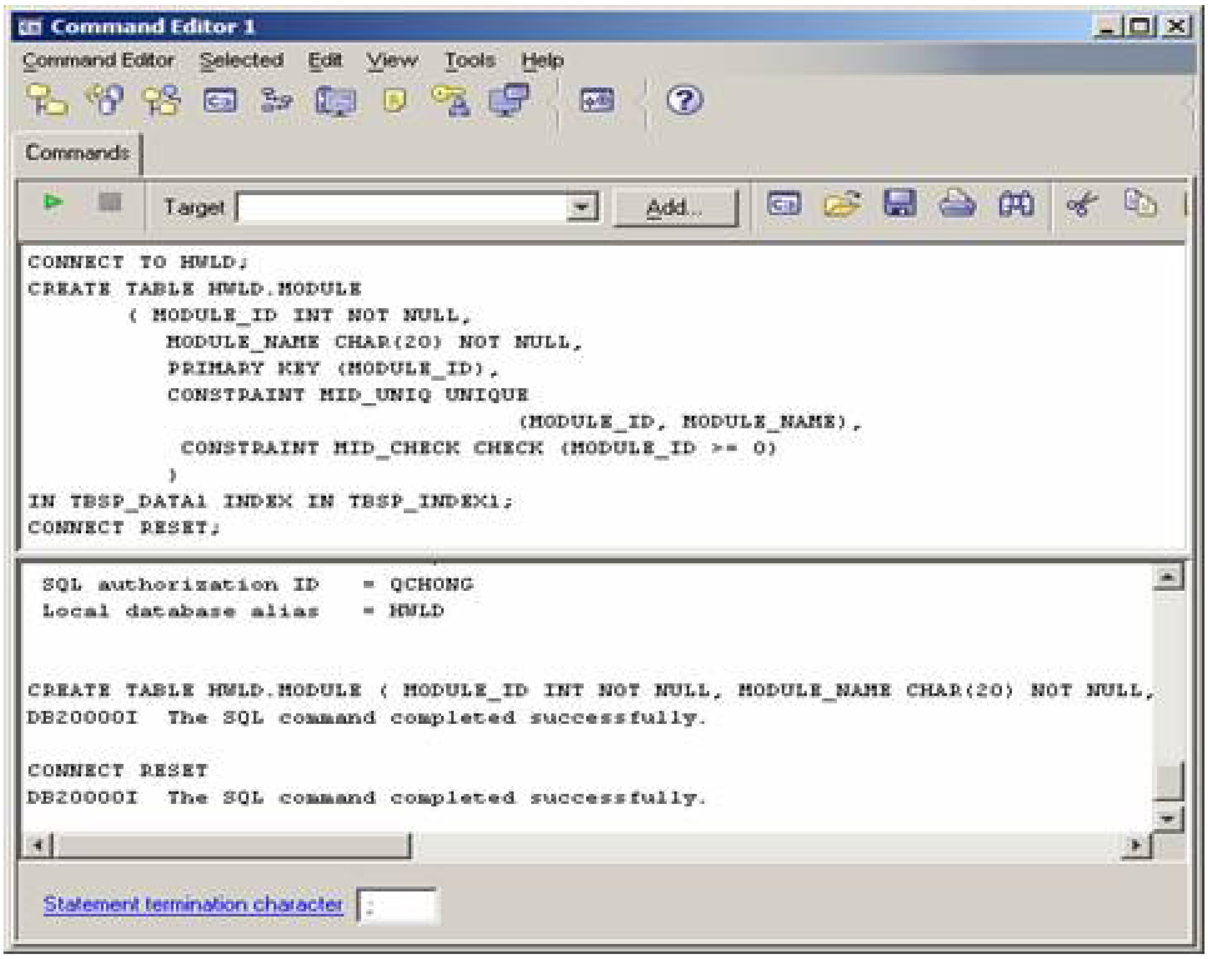
Columns

Key	Name	Data type	Length	Nullable
	AUTHOR_NAME	VARCHAR	50	No
	AUTHOR_ID	INTEGER	4	No
	MODULE_NAME	CHARACTER	20	No
	MODULE_ID	INTEGER	4	Yes

Command Editor

The Command Editor is very similar to the CLP in that it allows for administration of DB2 databases using text based commands. It also accepts DB2 commands, SQL statements and operating system commands. Unlike the CLP, the Command Editor allows users to cut and paste into the clipboard for easy scripting and the previous commands can be retrieved easily using a GUI. Following is an example of using the command editor to create a table:

Figure 4. DB2 Command Editor


Command Editor 1
Command Editor Selected Edit View Tools Help
Commands
Target
Add...
Statement termination character
;

```
CONNECT TO HWLD;
CREATE TABLE HWLD.MODULE
        ( MODULE_ID INT NOT NULL,
          MODULE_NAME CHAR(20) NOT NULL,
          PRIMARY KEY (MODULE_ID),
          CONSTRAINT MID_UNIQ UNIQUE
                                   (MODULE_ID, MODULE_NAME),
           CONSTRAINT MID_CHECK CHECK (MODULE_ID >= 0)
          )
IN TBSP_DATA1 INDEX IN TBSP_INDEX1;
CONNECT RESET;
```

```
 SQL authorization ID   = QCHONG
 Local database alias   = HWLD


CREATE TABLE HWLD.MODULE ( MODULE_ID INT NOT NULL, MODULE_NAME CHAR(20) NOT NULL,
DB20000I  The SQL command completed successfully.

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

Section 3. How does DB2 fit into SOA?

Service-Oriented Architecture (SOA) is an architectural style of software where service is used to support the requirements of the user. It allows the extension and reuse of existing services, tasks and applications for that purpose, by making resources available as independent components. These components can then be used to build complex applications to address the needs of the user.

IBM has defined the three starting points for SOA: people, process and information. With a growing number of Web-based businesses, the need for exposing and consuming information from databases is increasing. DB2 is the backbone that provides this service. The pureXML technology in DB2 9 lets you create XML applications that extend the existing technology.

Section 4. Creating a DB2 database

After installing DB2, you are ready to create your own database in DB2 environment. First you need to consider which instance your database should use. An *instance* provides a logical layer where you can group multiple databases together, controlled by a database manager configuration (DBM CFG) file. The DBM CFG file contains a list of DBM CFG parameters which you can use to tune the instance. You can create multiple instances on each workstation, and in each instance, you can create multiple databases. For this tutorial, use the default instance `DB2`, which was created during installation. More detail on the DBM CFG file can be found in the next tutorial, "DB2 for Linux, UNIX and Windows – Introduction to some advanced features and concepts."

As the following exercise shows, there are several factors to consider when you create a database. There are multiple ways to create a database. You can use the GUI Create Database Wizard from the Control Center, which guides you through the steps to create a database. Or, if you know the actual Create Database command, you can issue the command directly from a DB2 command window or a command editor or from the command line.

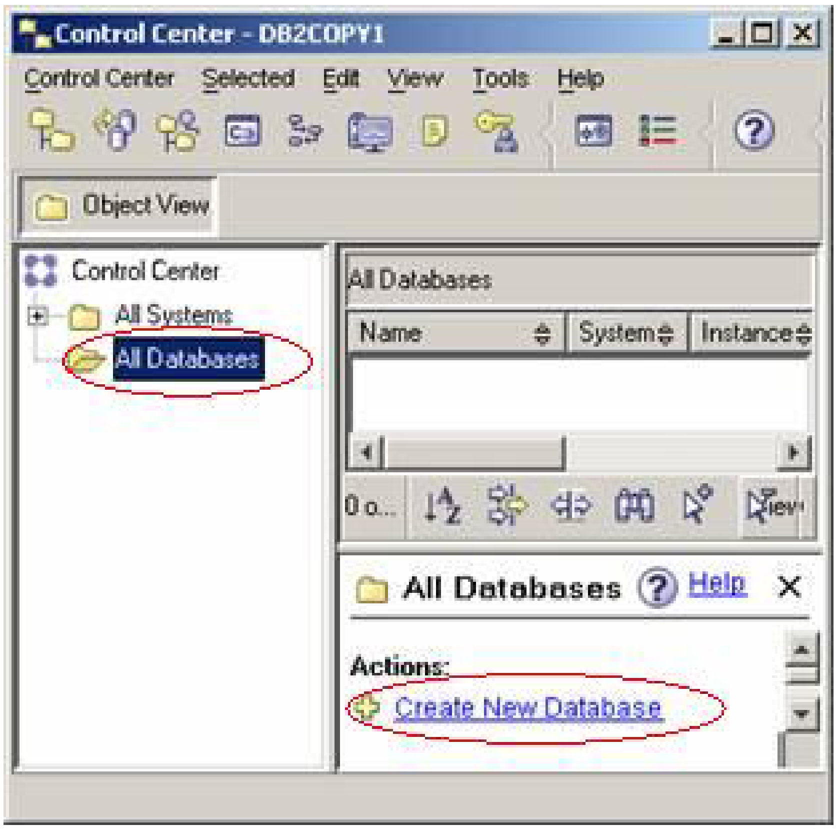
In this example, use the Create Database wizard from the DB2 First Steps Launch pad:

Would you like to see these steps demonstrated for you?

Show me

1. Launch the DB2 Control Center. Click **Start > All Programs > IBM DB2 > General Administration Tools** and select **Control Center**. When prompted to select the Control Center View, click **OK** to select the default (Advanced).

2. Click **All Databases** as shown in Figure 5, and then click **Create New Database**. A Create Your Own Database Wizard opens.
   Figure 5. Create New Database from Control Center



3. In the Create Your Own Database Wizard, specify the database name and where the database directories should go. Type `HelloWld` as the database name. Specify `C:\` as the default path. Enter `hwld` as the database alias. Type in any descriptive comment in the Comment field.

4. Check **Enable Database for XML**. XML columns are available only in UTF-8 code set (Unicode) database. Enabling Database for XML creates a Unicode database.

5. You are not going to specify any other database options, so select the Summary panel, then click **Show Command** to view the DB2 commands that are used to create this database.

6. Because you enabled XML in this database, the correct CODESET UTF-8 is used. Click **Close**. To save a set of commands to a file for future reference or to reuse, click **Save** here instead of Close.

7. From Create New Database window, click **Finish** and wait until the create database wizard finishes.

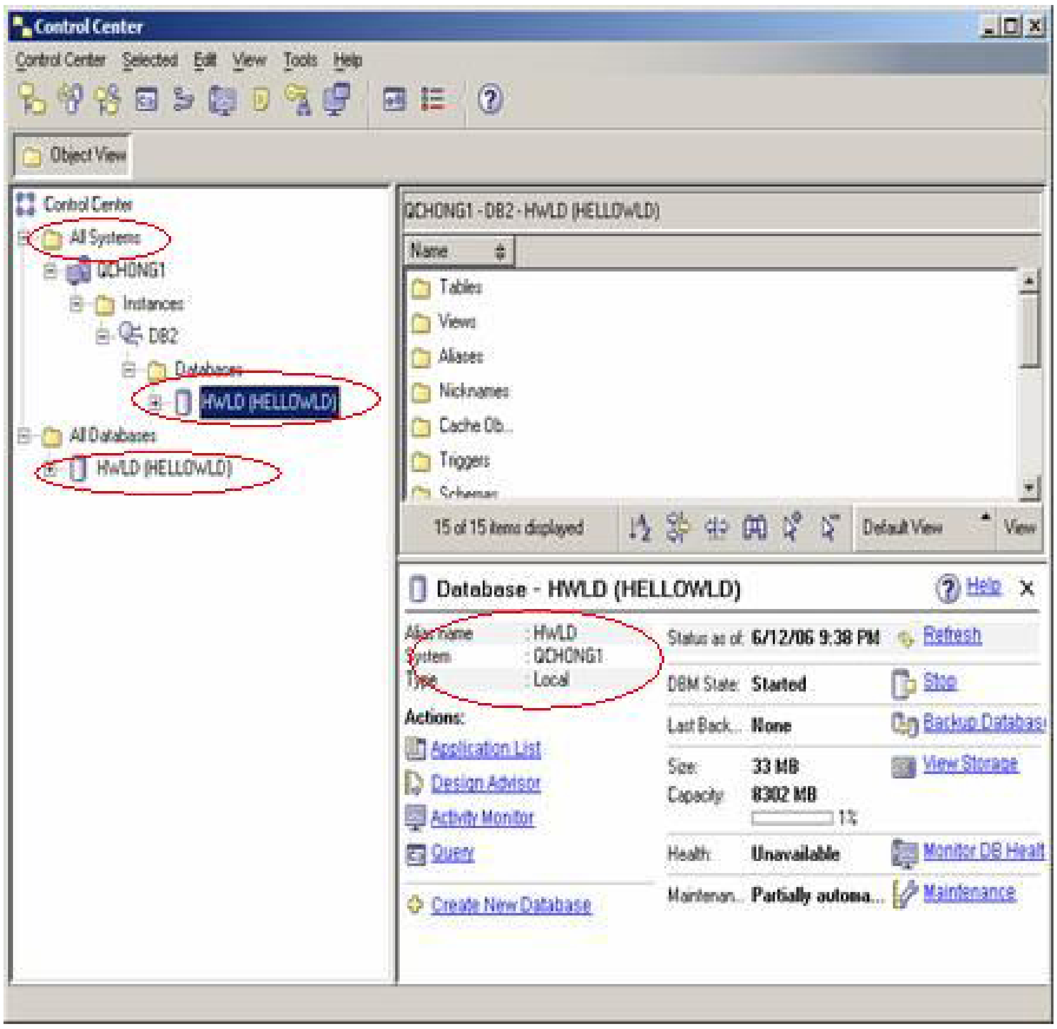
8. When DB2 creates a database, it also creates the following:
   - Necessary database directories on the drive or path you specified
   - A set of default tablespaces including SYSCAT, TEMPORARY and USER tablespaces

- A set of system catalog tables and views in the catalog tablespace
- The database configuration (DB CFG) file and default values are set
- The space for the database recovery log files and allocates it
- Several applications for the database such as CLI and the command line processor

Tablespaces and system catalog tables are discussed in detail in later sections. More detail on the DBM CFG file can be found in the next tutorial, "DB2 for Linux, UNIX and Windows – Introduction to some advanced features and concepts."

9. Use the DB2 Control Center to check the database you created. From the left panel in Control Center, locate and expand **All Systems** and **All Databases**. See if you can locate the HELLOWLD database you just created. All Systems provides the database object view, first grouped according to machine, instance, and then database. There is only one instance, DB2, on this machine. All Databases provides a quick view of all databases located on this machine.

Figure 6. View your new Database



Alias Name : HWLD
System : QCHONG1
Type : Local

Status as of: 6/12/06 9:38 PM
DBM State: Started
Last Back... None
Size: 33 MB
Capacity: 8302 MB
1%
Health: Unavailable
Maintenan... Partially automa...

10. If you know the exact DB2 command to create a database, you can use the CREATE DATABASE command you have saved in the previous Create New Database wizard window and modify it correspondingly, and then from DB2 command line processor, issue the CREATE DATABASE command to create a database.

11. From **Start > Run**, type `db2cmd` and press **Enter**. A DB2 command line processor window pops up. To get help for the `create database` command, from the command line window type `db2 ? create database`.

12. Type `db2 CREATE DATABASE MYTESTDB ON C:` After the command completes successfully, click **Selected** from the Control Center main menu bar and click **Refresh**. Then see if you can locate your new database MYTESTDB from the Control Center.

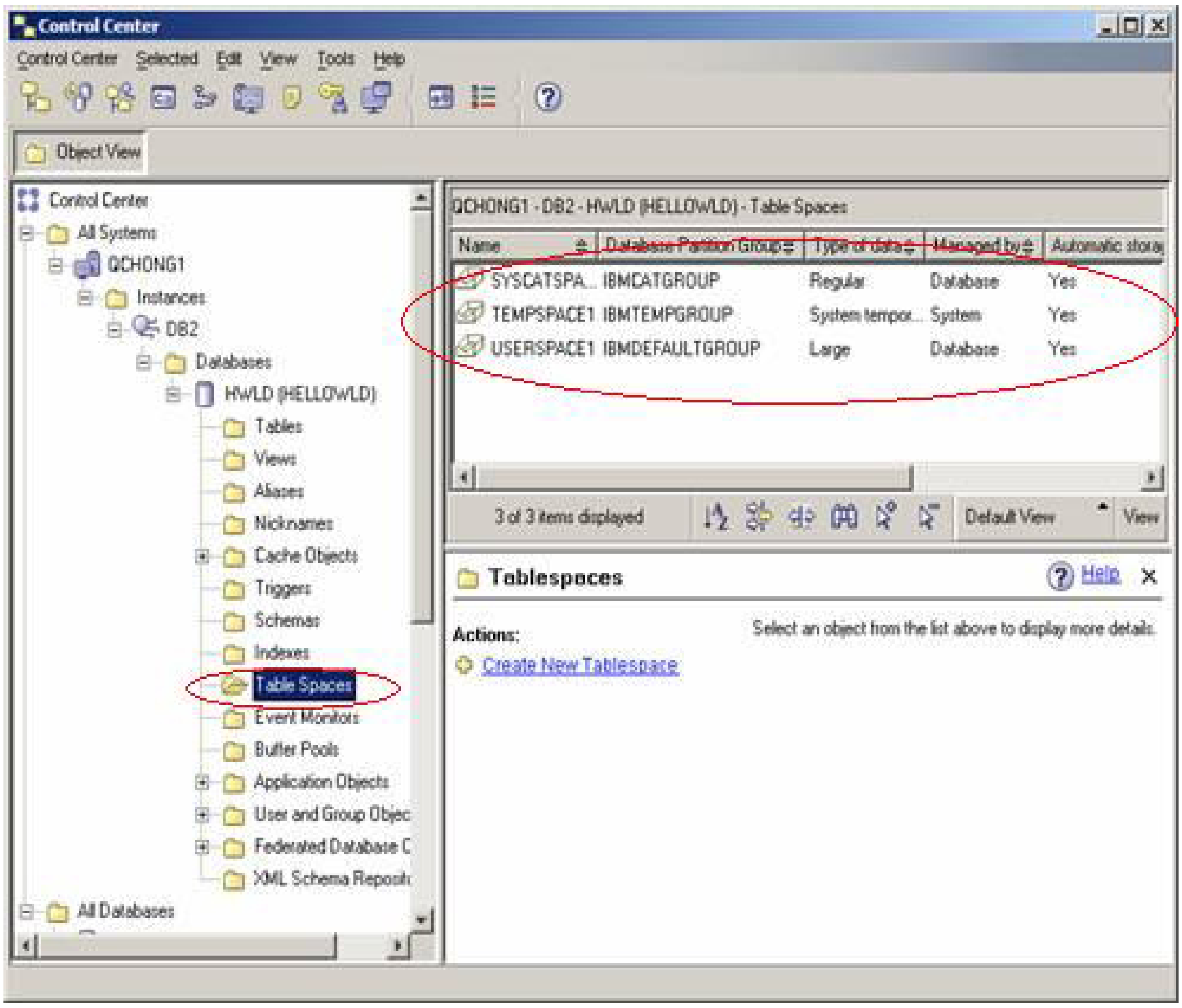
13. Enter the command `db2 LIST DATABASE DIRECTORY` from DB2 command window. You should see the database entries for HELLOWLD and MYTESTDB databases.

Section 5. Creating a tablespace

A *tablespace* is a logical layer in which some database objects, such as tables, views, and indexes, reside. A database can have multiple tablespaces. When a database is first created, a set of tablespaces is created automatically by DB2.

From the Control Center, expand **HELLOWLD** and click **Table Spaces**. You should see three different tablespaces listed on the right display window in the Control Center as shown in Figure 7:

Figure 7. Control Center Table Spaces view



QCHONG1 - DB2 - HWLD (HELLOWLD) - Table Spaces

Name	Database Partition Group	Type of data	Managed by	Automatic storag
SYSCATSPA...	IBMCATGROUP	Regular	Database	Yes
TEMPSPACE1	IBMTEMPGROUP	System tempor...	System	Yes
USERSPACE1	IBMDEFAULTGROUP	Large	Database	Yes

These tablespaces are created by DB2 when the database HWLD is created. SYSCATSPACE and TEMPSPACE1 are both system tablespaces and cannot be dropped or recreated. SYSCATSPACE is the system catalog tablespace, which is used to hold meta information about the database. TEMPSPACE1 is used to hold temporary results during certain database operations. USERSPACE1 is the default user tablespace to hold user data such as tables, views and indexes. Any DB2 user who has enough authority can drop and recreate USERSPACE1 or create more user tablespaces. There is more on authorities in later sections.

SMS or DMS tablespace

When you create a tablespace, you can choose whether the tablespace is an SMS tablespace or a DMS tablespace. SMS stands for System Managed Space, and DMS stands for Database Managed Space. In an SMS tablespace, space is managed by the operating system's file system, and space is allocated on demand. An SMS tablespace requires less initial setup to create, fewer management considerations, and is generally easier to create and use because space is allocated automatically when needed. In a DMS tablespace, space is managed by DB2. A DMS tablespace requires space to be preallocated at its creation time and so requires more initial work to set up, and has more management considerations. However, it provides a lot more flexibility to the user to control the data placement and improve performance for data accessing. In general, a well-designed and tuned DMS tablespace achieves better performance than a similar SMS tablespace.

From the Table Spaces view, click on each tablespace. Take a look at the detailed information displayed in the right bottom window. SYSCATSPACE and USERSPACE1 are both DMS tablespaces and TEMPSPACE1 is an SMS tablespace. You can also create your own tablespace.

Bufferpool

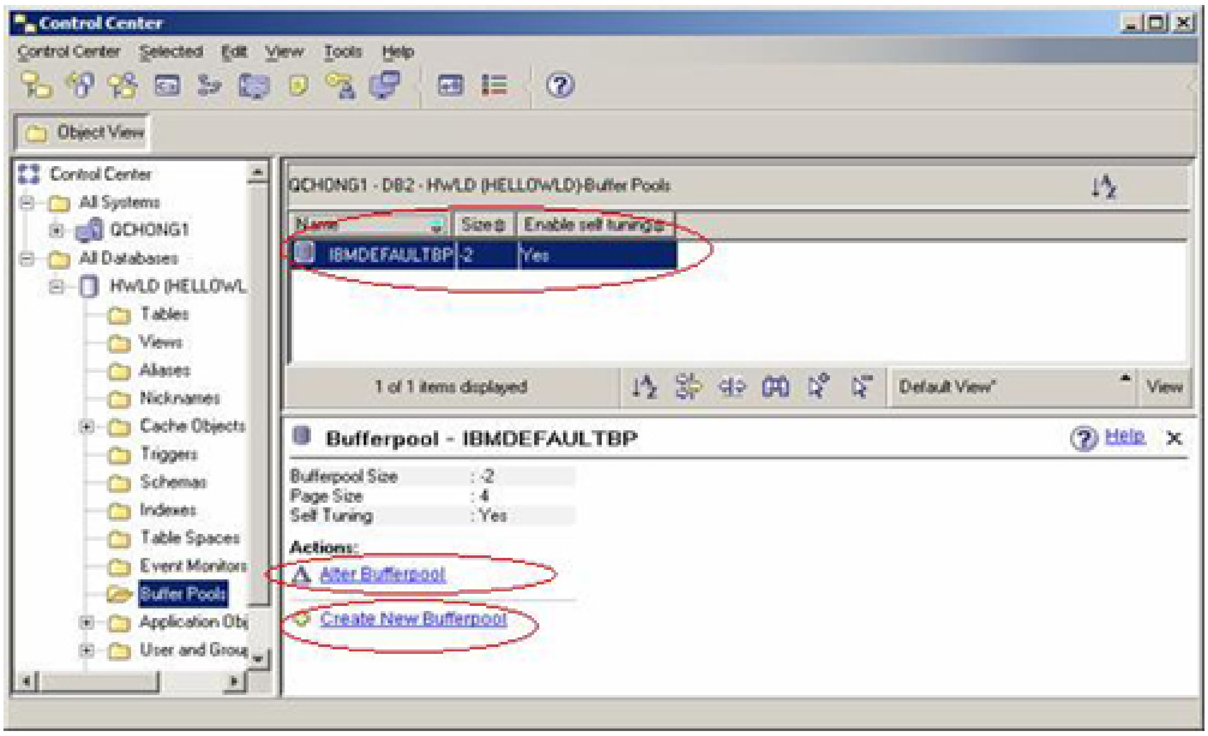
A *bufferpool* is a collection of blocks of memory in the form of pages. When a database is first created, a default bufferpool, IBMDEFAULTBP, is also created. You can create a new bufferpool when you create a tablespace or use the Create New Bufferpool wizard from the Buffer Pools view.

The most important role that a bufferpool plays is to help reduce the I/O cost when the database is reading and writing data to and from the disk. This is achieved by I/O pre-fetching and page cleaners. Pre-fetching reduces the I/O cost of reading a page by predetermining what possible pages may be needed for a particular query and then reading them into the bufferpool so that when a query needs them, they are ready to be used. Page cleaners ensure that any pages no longer needed by transactions are written to disk first if the pages have been updated, and then purged from the bufferpool. This is to ensure there is enough clean space to read in pages in the bufferpool.

DB2 supports different page sizes: 2K, 4K, 8K, 16K, and 32K. IBMDEFAULTBP uses a 4K page size. If you would like to create a tablespace that uses a page size other than 4K, first make sure you have a bufferpool in that page size in the database. Multiple tablespaces can use the same bufferpool. You can specify which bufferpool to use when you create or alter the tablespace. Choosing and creating proper bufferpools is important for database performance. You can create a new bufferpool or alter an existing bufferpool to suit your needs.

From the Control Center, click **Buffer Pools**. You can view all the bufferpools in this database. Choose to create a new bufferpool or alter an existing bufferpool.

Figure 8. Control Center Bufferpool view



Name	Size	Enable self tuning
IBMDEFAULTBP	-2	Yes

Bufferpool - IBMDEFAULTBP

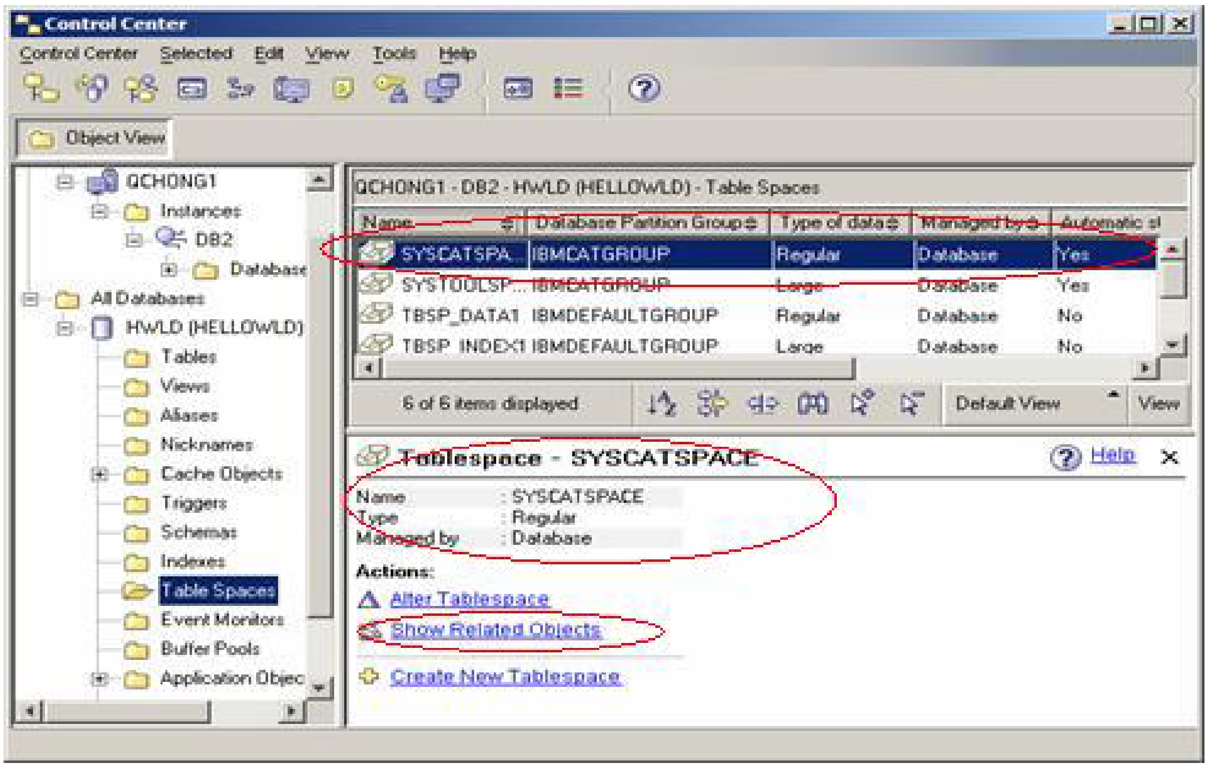
Bufferpool Size	: -2
Page Size	: 4
Self Tuning	: Yes

DB2 system catalog tables and views

DB2 creates multiple system catalog tables and views in the catalog tablespace when it creates a new database. These system catalog tables and views are used to keep track of important information that the database manager needs to know about its own database objects, access control information, and tools. The system catalog views are based on the base system catalog tables. The system catalog views are normally queried by users interested in system catalog table data.

From the Control Center, under the All Database view, expand **HELLOWLD**, and then select **Table Spaces**. Select **SYSCATSPACE** displayed in the top right window. You can see the summary of this tablespace in the bottom right window:

Figure 9. Control Center -- Tablespaces view



QCHONG1 - DB2 - HWLD (HELLOWLD) - Table Spaces

Name	Database Partition Group	Type of data	Managed by	Automatic st
SYSCATSPA...	IBMCATGROUP	Regular	Database	Yes
SYSTOOLSP...	IBMCATGROUP	Large	Database	Yes
TBSP_DATA1	IBMDEFAULTGROUP	Regular	Database	No
TBSP_INDEX1	IBMDEFAULTGROUP	Large	Database	No

Tablespace - SYSCATSPACE

Name	: SYSCATSPACE
Type	: Regular
Managed by	: Database

Click **Show Related Objects**. A new page appears showing all tables and indexes along with their schema SYSIBM in tablespace SYSCATSPACE. Click **Indexes** and all indexes in SYSCATSPACE are listed. **Show SQL** shows you the select command to get the list of tables or indexes in this tablespace.

Tables, indexes, schemas and views are all important DB2 objects and are discussed in detail later in the tutorial. But first, try your hand at creating a tablespace:

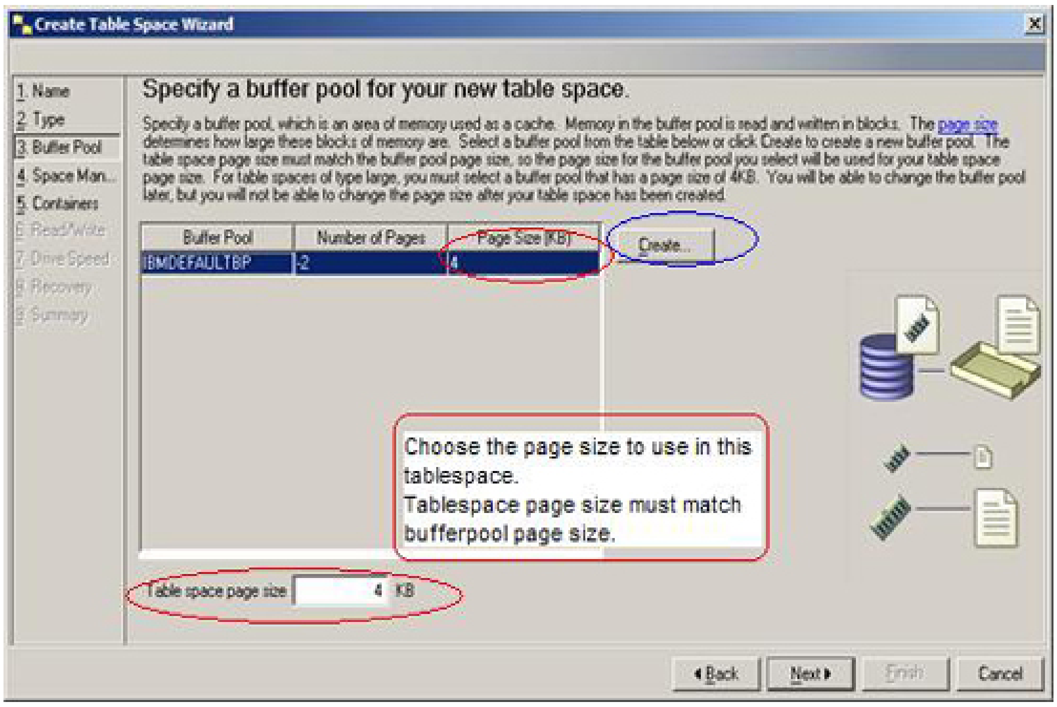
Would you like to see these steps demonstrated for you?

Show me

1. Click **Create New Tablespace** from the Control Center to start the Create Table Space Wizard. Type `TBSP_DATA1` as the tablespace name, and type `DMS tablespace to hold data` in the comment field.

2. Select **Regular** to choose the type of tablespace to use and click **Next**. The Specify a buffer pool panel appears, on which you select the bufferpool to be used by this tablespace.

3. Use the default IBM bufferpool, but you can also create a new bufferpool using the Create button. Click **Next**.
   Figure 10. Create New Tablespace -- Specify a Bufferpool



Specify a buffer pool for your new table space.

Specify a buffer pool, which is an area of memory used as a cache. Memory in the buffer pool is read and written in blocks. The page size determines how large these blocks of memory are. Select a buffer pool from the table below or click Create to create a new buffer pool. The table space page size must match the buffer pool page size, so the page size for the buffer pool you select will be used for your table space page size. For table spaces of type large, you must select a buffer pool that has a page size of 4KB. You will be able to change the buffer pool later, but you will not be able to change the page size after your table space has been created.

Buffer Pool	Number of Pages	Page Size (KB)
IBMDEFAULTBP	-2	4

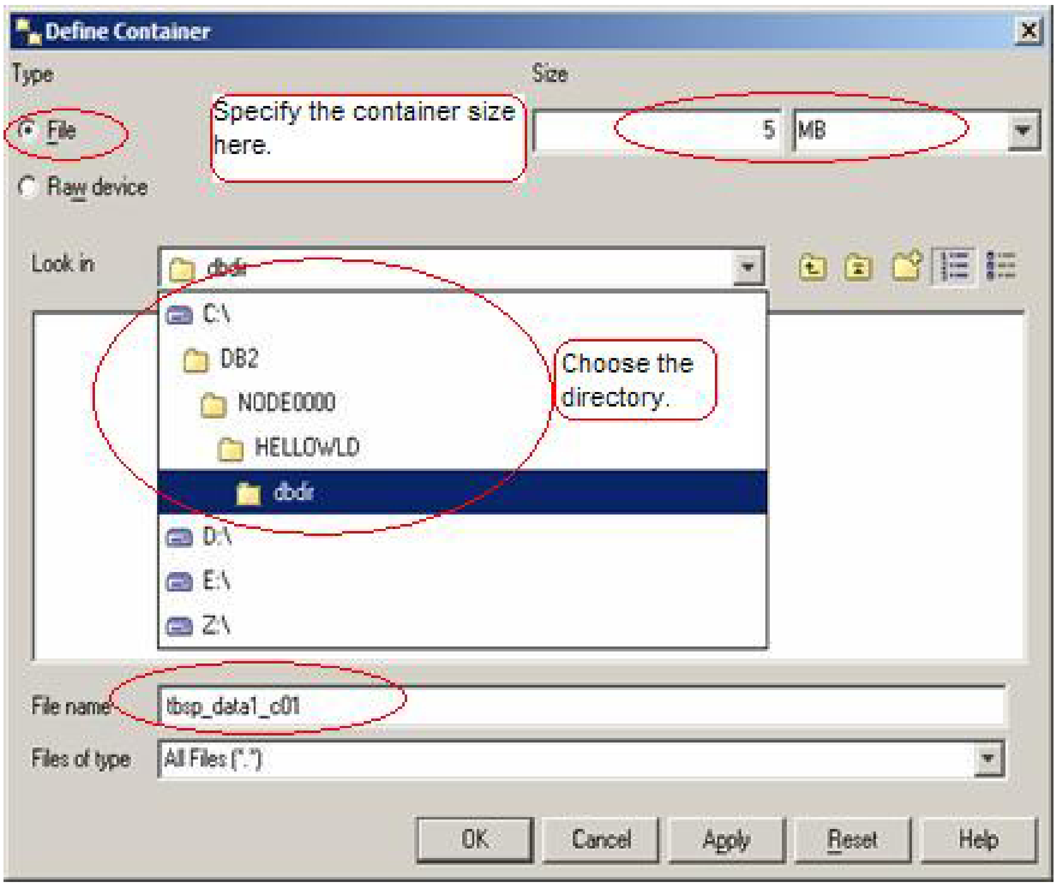
Choose the page size to use in this tablespace.
Tablespace page size must match bufferpool page size.

Table space page size 4 KB

4. This panel allows you to specify whether your tablespace is to be managed by the database (DMS) or the system (SMS). Select **Database-managed space (high performance)**. Click **Next**. The Containers page appears.

5. From the Containers page, you can specify the number of containers. A *Container* can be a directory, a file or a device where tablespace data is to be stored. You can create multiple containers for each tablespace, and you can drop existing or add more containers to a tablespace after the tablespace is created. Click **Add** and the Define Container page appears, allowing you to define containers for your new tablespace.

6. You can define multiple containers for the new tablespace. Different containers can be located in different physical nodes or paths. Follow the instructions as shown in Figure 11 to add container and then click **OK**.
Figure 11. Create New Tablespace -- Add Container



Specify the container size here.

Size: 5 MB

Choose the directory.

Look in: dbdir

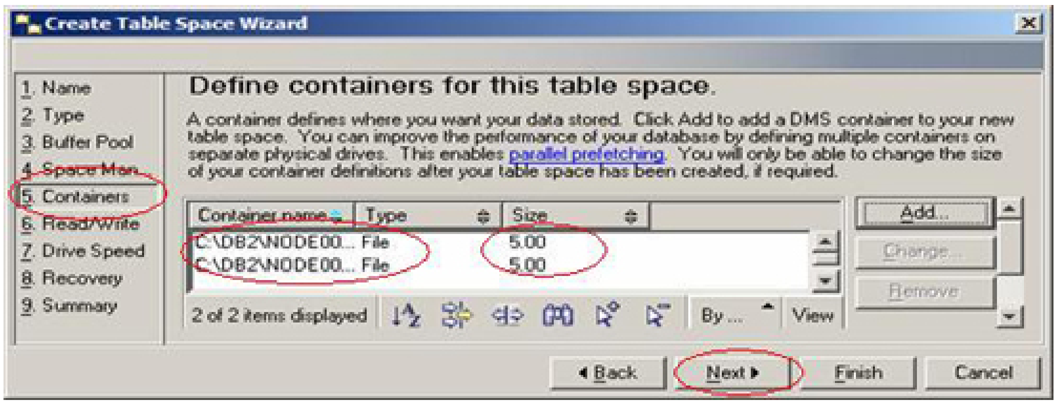
C:\
DB2
NODE0000
HELLOWLD
dbdir
D:\
E:\
Z:\

File name: tbsp_data1_c01

Files of type: All Files (*.*)

7. Add another container tbsp_data1_c02 in the same path using the same actions.

8. You should see two containers added as shown in Figure 12. Each is 5MB in size, for a total of 10MB. If you are unhappy with the containers you have defined, click **Change** or **Remove**. Click **Next** to continue.
Figure 12. Create New Tablespace -- Container Page



Define containers for this table space.

A container defines where you want your data stored. Click Add to add a DMS container to your new table space. You can improve the performance of your database by defining multiple containers on separate physical drives. This enables parallel prefetching. You will only be able to change the size of your container definitions after your table space has been created, if required.

Container name	Type	Size
C:\DB2\NODE00...	File	5.00
C:\DB2\NODE00...	File	5.00

2 of 2 items displayed

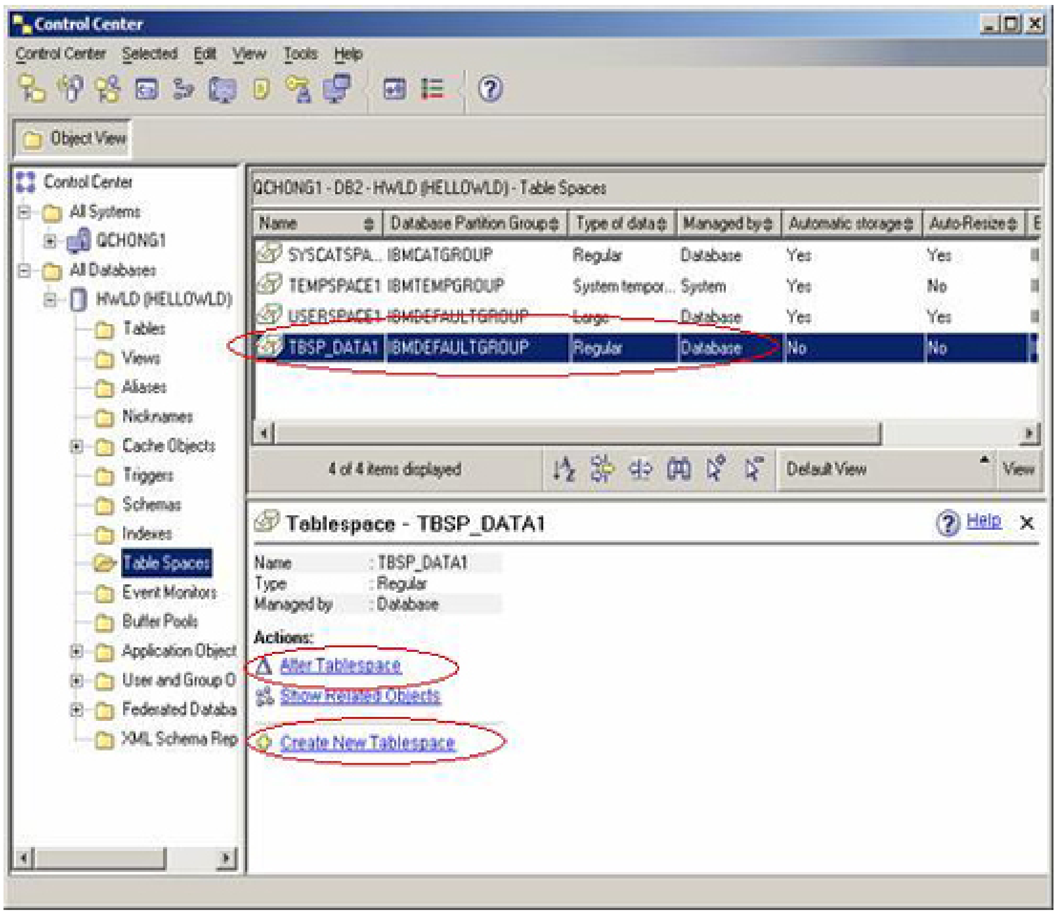
9. This next page helps you to determine the extent and prefetch sizes that should be used for this tablespace. An extent contains a consecutive number of pages. The number of pages is referred to as the extent size. In general, it's more desirable to allocate multiple pages at a time to reduce the overhead of space allocation. Prefetch size refers to the

number of pages the prefetcher reads in when prefetching is enabled. The concept of prefetching is discussed later. Select **Less than 25MB** to change the average size of a table and the recommended extent size automatically changes to the appropriate setting.

10. Click **Summary** to display the Summary page. Click **Show SQL** and save the SQL under C:\DB2\hellowld\crt_tbsp.clp. Click **Finish** to let the Create Tablespace process complete.

11. From the Control Center Table Spaces view, you should be able to find the new tablespace TBSP_DATA1. If you are not completely satisfied with the newly created tablespace TBSP_DATA1, click **Alter Tablespace** wizard to change the tablespace. Click **Create New Tablespace** if you want to create an SMS tablespace.

Figure 13. Control Center -- Tablespace view



Name	Database Partition Group	Type of data	Managed by	Automatic storage	Auto-Resize
SYSCATSPA...	IBMCATGROUP	Regular	Database	Yes	Yes
TEMPSPACE1	IBMTEMPGROUP	System tempor...	System	Yes	No
USERSPACE1	IBMDEFAULTGROUP	Large	Database	Yes	Yes
TBSP_DATA1	IBMDEFAULTGROUP	Regular	Database	No	No

Tablespace - TBSP_DATA1

Name	: TBSP_DATA1
Type	: Regular
Managed by	: Database

Actions:
Alter Tablespace
Show Related Objects
Create New Tablespace

It is beneficial to store your index, Large Objects (LOB) and Long Field (LF) or XML data separately from table data. However there is no LOB/LF data in this tutorial so you do not need to create tablespaces for them. The XML data is stored in the same tablespace as the rest of the data, TBSP_DATA1, so you do not need to create a tablespace for that either. Later sections discuss indexes.

12. From the DB2 command line, enter `cd C:\DB2\HELLOWLD\` to create a tablespace. You might need to create the directory HELLOWLD first if it

does not exist. Then run the following command `notepad crt_tbsp.clp`. In the notepad, edit the Create Tablespace statement by replacing it with the following line:

```
CREATE LARGE TABLESPACE TBSP_INDEX1 PAGESIZE 4 K MANAGED BY DATABASE USING
(FILE 'C:\DB2\NODE0000\hellowld\dbdir\tbsp_index1_c01' 5120,
FILE 'C:\DB2\NODE0000\hellowld\dbdir\tbsp_index1_c02' 5120)
EXTENTSIZE 8 OVERHEAD 10.5 PREFETCHSIZE 8 TRANSFERRATE 0.14 BUFFERPOOL IBMDEFAULTBP;
```

13. Replace the COMMENT statement with the following:

```
COMMENT ON TABLESPACE TBSP_INDEX1 IS 'tablespace to hold index data';
```

14. The following commands should be in crt_tbsp.clp now:

```
CONNECT TO HELLOWLD;

CREATE LARGE TABLESPACE TBSP_INDEX1 PAGESIZE 4 K MANAGED BY DATABASE USING
(FILE 'C:\DB2\NODE0000\hellowld\dbdir\tbsp_index1_c01' 5120,
 FILE 'C:\DB2\NODE0000\hellowld\dbdir\tbsp_index1_c02' 5120 )
EXTENTSIZE 8 OVERHEAD 10.5 PREFETCHSIZE 16 TRANSFERRATE 0.14 BUFFERPOOL IBMDEFAULTBP ;

COMMENT ON TABLESPACE TBSP_INDEX1 IS 'tablespace to hold index data';

CONNECT RESET;
```

15. Save it and exit the notepad.
16. Run the `db2 -tvf crt_tbsp.clp` command from the DB2 command line window, and you should see the tablespace completed successfully. The `-tvf` options are DB2 command line options that allow you to run DB2 commands from a file.
17. Go to the Control Center, click **Selected** from the Control Center main menu, then click **Refresh** from the drop down menu to view all the tablespaces you have created so far.

Section 6. Using DB2 objects: Creating schemas, tables, and views

In general, data is stored in relational tables in DB2. Each table consists of a number of columns and rows. Table columns are defined during creation of a table. They can also be added or dropped later after the table is created. Data stored in a table needs to be consistent with the data column definitions. Each table can have multiple indexes and views.

A *view* is like a logical table, composed of the result set of a SELECT statement from one or multiple tables or views. Unlike the relational table, data in a view does not need to be stored physically on disk. The data is made available when the view is queried. A view does not use physical space other than storing its definitions in the system catalogs. You can use Data Manipulation Language (DML) to query a view or even update a view after it is created. Views provide flexible data access to a subset of a table or a join of a result set from multiple tables, while hiding the complexity of the data in the base tables.

To provide a logical view of database objects, such as tables, indexes, and views, classify them using one or more schemas. A *schema* is a logical categorization of database objects. You can create multiple database objects using the same or different schema. For instance, in the tablespace SYSCATSPACE, all the base system tables and indexes are grouped into the same schema, SYSIBM. All the views on the base catalog tables and indexes are grouped into a schema, SYSCAT or SYSSTAT.

This exercise demonstrates how to create schemas, tables, and views:

Would you like to see these steps demonstrated for you?

Show me

1. From the Control Center, click **All Databases > HELLOWLD > Schemas**. Take a look at the existing schemas that are created by DB2 when a database is created. Click **Create New Schema** from the display window located at the bottom of the right side window.

2. From the Create Schema wizard, type the new schema name as `HWLD`. Use the default Authorization Name. Click **OK** to run the `CREATE SCHEMA` command. Again, click **Show SQL** to see the actual DB2 command. After the command completes, check if HWLD is displayed in the schema view.

3. When creating an object, specify the schema name to which you want it to belong. If a schema name is not explicitly specified, then by default the user ID is used as the schema, as long as the user has the IMPLICIT_SCHEMA privilege (privileges are discussed in detail in the next tutorial in this series). See Resources for a link to the entire Hello World series.

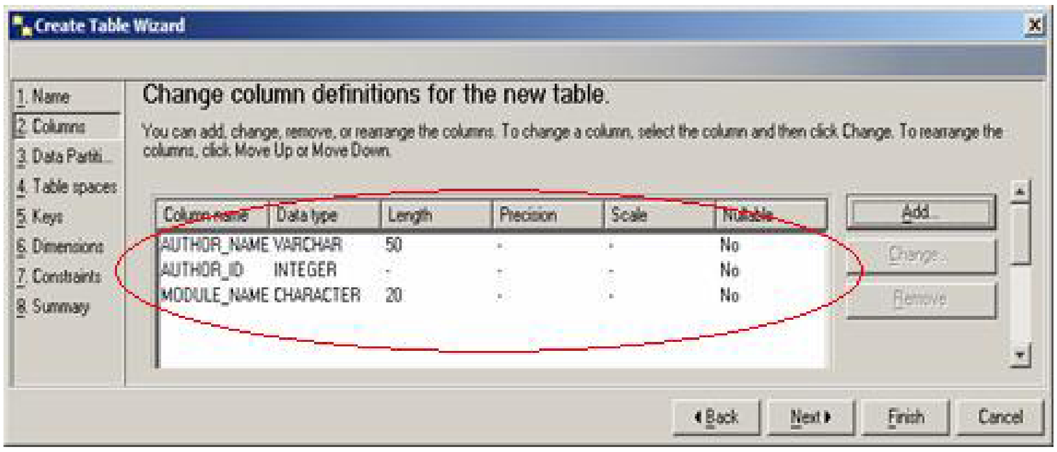
4. To create a table, select **All Databases > HELLOWLD > Tables** from the left side of the Control Center display window. Click **Create New Table** from the bottom right display window. The Create New Table wizard window launches to guide you through the steps to create a table.

5. For Create Table Wizard, first choose **HWLD** from the drop-down menu as the schema name. Enter `AUTHOR` as the Table name. Type a

descriptive comment. Click **Next**.

6. From the Columns page, click **Add** to add columns for the table, AUTHOR. From the Add Column Page, specify `AUTHOR_NAME` as Column name, `VARCHAR` as Data type, and Length as `50=`. Click **OK**.

7. Click **Add** to add a second column, with AUTHOR_ID as the Column name and Integer for the Data type. Click OK. Add a third column with MODULE_NAME as the Column name, the Data type is CHARACTER, and length is **20**. Click **OK**.

8. You should see three columns listed in the Columns page.
Figure 14. Create Table -- Columns


Change column definitions for the new table.

You can add, change, remove, or rearrange the columns. To change a column, select the column and then click Change. To rearrange the columns, click Move Up or Move Down.

Column name	Data type	Length	Precision	Scale	Nullable
AUTHOR_NAME	VARCHAR	50	-	-	No
AUTHOR_ID	INTEGER	-	-	-	No
MODULE_NAME	CHARACTER	20	-	-	No

Click **Next**. The Data Partitions page displays.

9. *Data partitioning*, a new feature in DB2 9, allows users to partition a large table across multiple tablespaces. Click **Next** to skip this step because the AUTHOR table is not going to be a partitioned table.

10. From the Table spaces page, select **TBSP_DATA1** as the tablespace. Select **Use Separate Index Space**, and choose **TBSP_INDEX1** as the index tablespace. Optionally, you may also specify a separate large tablespace for any large objects, but you are not doing that here. Click **Next**.

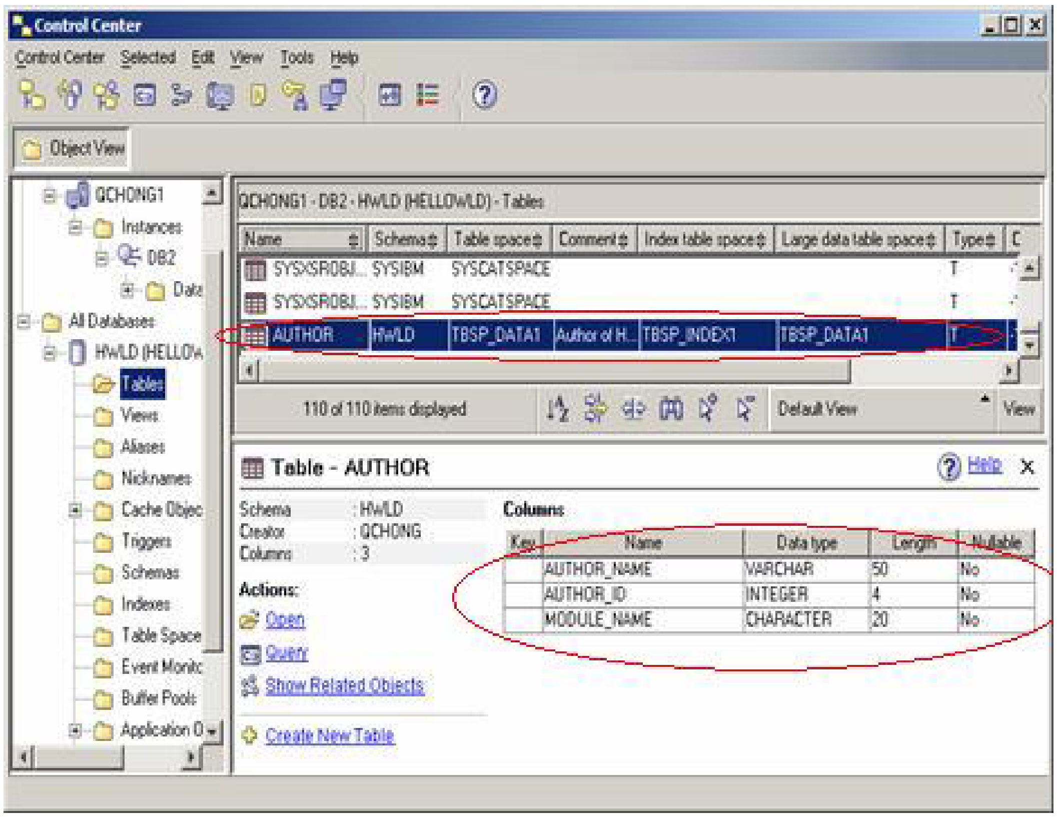
11. If you do not see TBSP_DATA1 or TBSP_INDEX1, cancel the Create Table Wizard and make sure to refresh the tablespace view from Control Center.

12. Do not define any primary or unique keys, dimensions, or constraints at this time. Continue to click **Next** on the subsequent pages until you reach the Summary page.

13. Use **Show SQL** from Summary page to view the actual Create Table command. Click **Finish** to create the HWLD.AUTHOR table.

14. You should see the DB2 Message window (DB20000), which indicates the command has completed without errors. Close the message window.

15. From the Control Center Tables view, click the AUTHOR table you have just created, and you should see the table columns definition, schema, and creator in the display window.
Figure 15. Control Center -- Table view



QCHONG1 - DB2 - HWLD (HELLOWLD) - Tables

Name	Schema	Table space	Comment	Index table space	Large data table space	Type
SYSXSROBJ...	SYSIBM	SYSCATSPACE				T
SYSXSROBJ...	SYSIBM	SYSCATSPACE				T
AUTHOR	HWLD	TBSP_DATA1	Author of H...	TBSP_INDEX1	TBSP_DATA1	T

Table - AUTHOR

Schema	: HWLD
Creator	: QCHONG
Columns	: 3

Columns

Key	Name	Data type	Length	Nullable
	AUTHOR_NAME	VARCHAR	50	No
	AUTHOR_ID	INTEGER	4	No
	MODULE_NAME	CHARACTER	20	No

16. After examining the HWLD.AUTHOR table, add another column, AUTHOR_DOC. Make it an XML column because you store the XML document directly in this XML column.

17. Right-click **AUTHOR**, select **Alter** from the menu, and an Alter Table wizard opens. Click **Add** to start the Add Column wizard.

18. From the Add Column Wizard type `AUTHOR_DOC` as the Column name and select **XML** as the Data type. Select **Nullable** to indicate this column can contain NULL value. Click **OK**.

19. Check the Alter Table wizard to verify a new XML column was added. You have the option to store this table data in a compressed format, especially if space is a big concern and your table data is large. Here, the data remains in uncompressed format. Click **OK** to let ALTER TABLE finish. A message, DB20000, should be returned. Close the message window.
From the Control Center, Table > Author, you should see the fourth column with a Data type of XML.

20. To create a view based on the Author table, select **Control Center > All Databases > HWLD > Views > Create New View**. A Create View wizard guides you through steps to create a view.

21. From the Create View Wizard, choose **HWLD** as the View schema, type in `DB2_AUTHOR` as the View name.

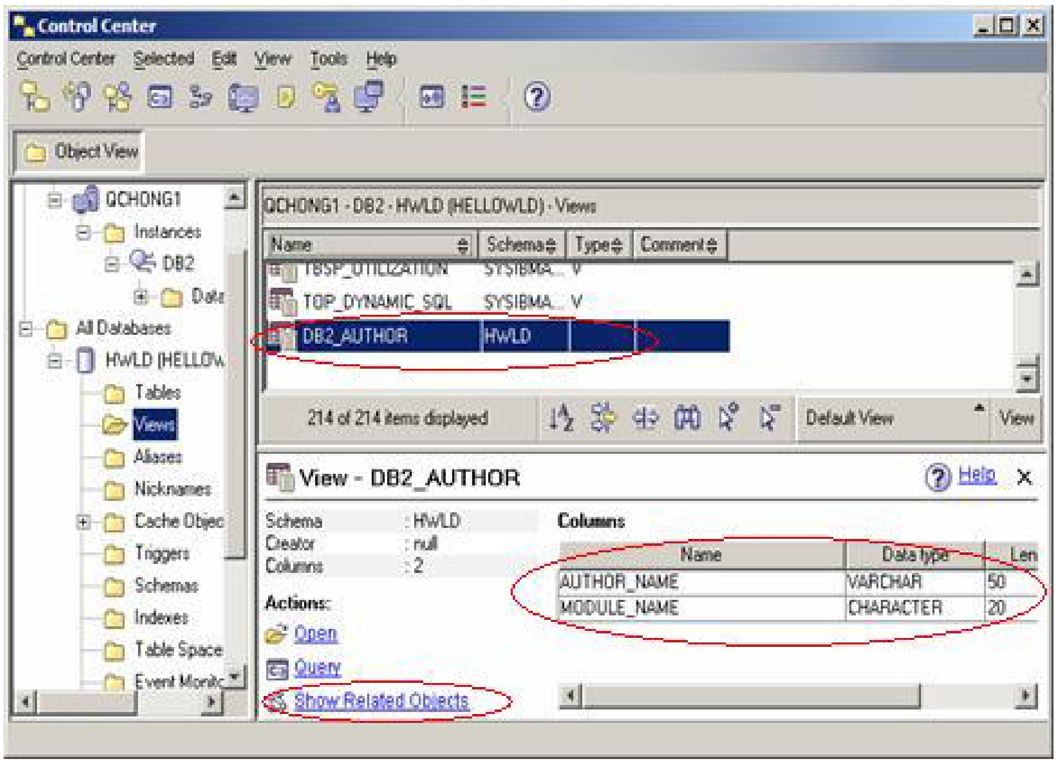
22. Replace the SQL statement with the following:

```
(AUTHOR_NAME, MODULE_NAME)
AS
SELECT AUTHOR_NAME, MODULE_NAME FROM HWLD.AUTHOR WHERE MODULE_NAME='DB2 UDB'
```

Check options should have **None** selected to keep the view read-only. Click **OK** to create the view DB2_AUTHOR.

23. From the Control Center, select the DB2_AUTHOR view and look at the view definition in the panel on the lower right half of the screen.

Figure 16. Views -- Show Related Objects



QCHONG1 - DB2 - HWLD (HELLOWLD) - Views

Name	Schema	Type	Comment
TBSP_UTILIZATION	SYSIBMA...	V	
TOP_DYNAMIC_SQL	SYSIBMA...	V	
DB2_AUTHOR	HWLD		

View - DB2_AUTHOR

Schema : HWLD
Creator : null
Columns : 2

Columns

Name	Data type	Len
AUTHOR_NAME	VARCHAR	50
MODULE_NAME	CHARACTER	20

24. Click **Show Related Objects**. From the Show Related page, click on the Tables tab. Information on the base table for this view is displayed. Click **Close**.

Section 7. Using DB2 objects: Creating indexes

An *index* is an auxiliary object in the database, but it is often critical to the database application performance. Indexes allow queries to potentially access data more efficiently. Unique indexes are also used to "enforce the uniqueness" of certain data columns.

An index is defined on a table, on a subset of table columns that are used as index keys. An index stores the index keys in a sorted order. In addition to storing the index key, each index entry contains a logical pointer called a record ID (RID), which points to the location of a data row in a table. You can specify whether you want to store the index entries in ascending or descending order. Indexes are stored separately from the table data.

You can choose whether the index is clustering or not. A *clustering index* allows data in the table to be clustered in the same order as the clustering index. A clustering index further improves performance when a query accesses the table data in the same sequence as the clustering index.

You can define multiple indexes on a table, but you can have only one clustering index for each table, with the exception of Multiple Dimensional Clustering (MDC) tables. You cannot create indexes on a view.

Indexes can be created using the CREATE INDEX statement. The following exercise demonstrates how to create an index using the GUI tool:

Would you like to see these steps demonstrated for you?

Show me

1. From the Control Center, select the Indexes view. Click **Create New Index** to launch the Create Index wizard.

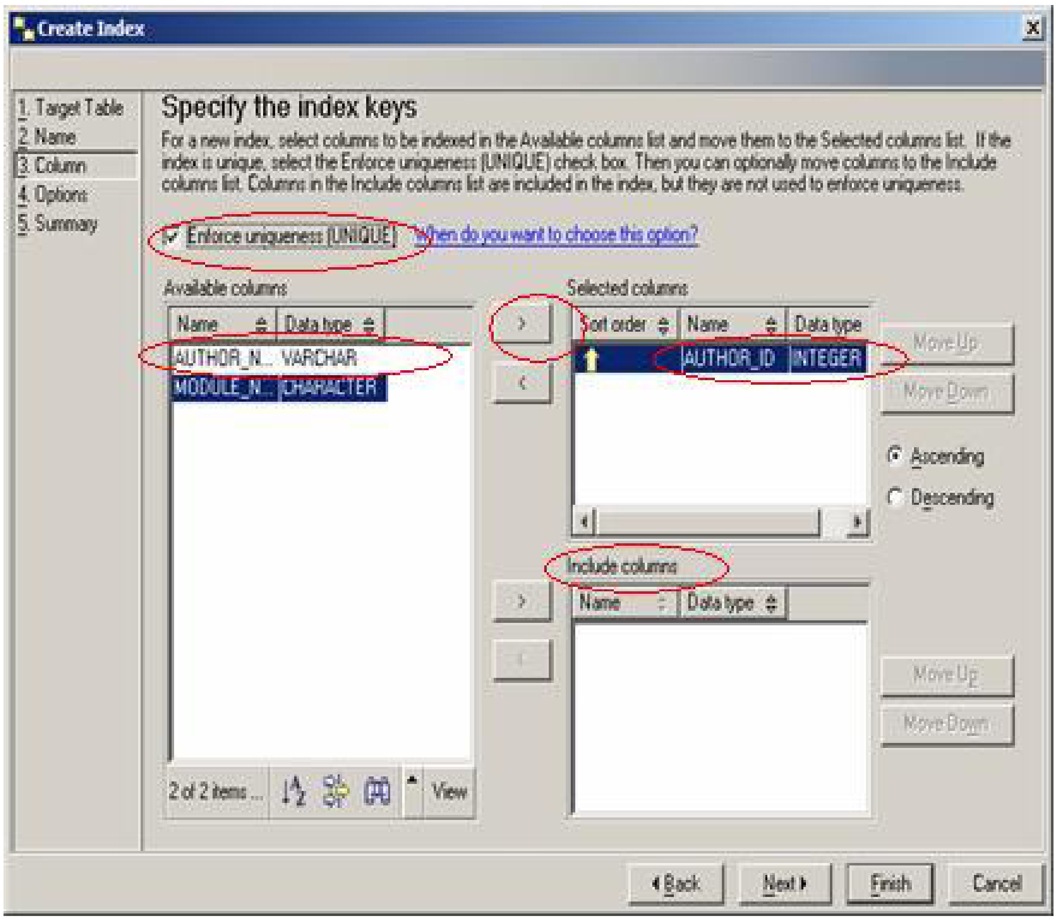
2. From the Create Index Wizard, select the table where the new index is created. Select **HWLD** as the Table schema and **AUTHOR** as the Table Name. Specify **No** for XML column options. Click **Next**.

3. From the Name page, choose **HWLD** as the Index schema and specify `AUTHORID` as Index Name. Click **Next** to go to the Column page to specify index columns.

4. Select **AUTHOR_ID**, and click the **>** button to add it to the right as selected index columns. Check **Enforce uniqueness**, to keep the data in AUTHOR_ID unique. Include columns is a feature that DB2 provides to store additional data columns as part of the index for fast data access,

especially during an index only scan. Here there are no include columns. Click **Next**.

Figure 17. Create Index Wizard



Specify the index keys

For a new index, select columns to be indexed in the Available columns list and move them to the Selected columns list. If the index is unique, select the Enforce uniqueness (UNIQUE) check box. Then you can optionally move columns to the Include columns list. Columns in the Include columns list are included in the index, but they are not used to enforce uniqueness.

☑ Enforce uniqueness (UNIQUE) When do you want to choose this option?

Available columns

Name	Data type
AUTHOR_N...	VARCHAR
MODULE_N...	CHARACTER

2 of 2 items...

Selected columns

Sort order	Name	Data type
↑	AUTHOR_ID	INTEGER

[✓ Ascending] Descending

Include columns

Name	Data type

5. The Options page allows you to customize some index performance options. Check **Customize performance options**.

6. Choose the default values. PCTFREE and LEVEL 2 PCTFREE specify the free space to be left on the leaf pages and level 2 index pages of the index tree during index creation. If you know your table is going to have many inserts later on, you may want to increase the 10% default value, to avoid index page splits during those inserts. If MINPCTUSED is specified, DB2 uses the threshold value to determine when two neighbouring index pages, if both are near empty, can and will be merged together. This feature, if enabled, can help reduce empty space on index pages caused by deletes. Proper tuning of these parameters helps improve index access performance.

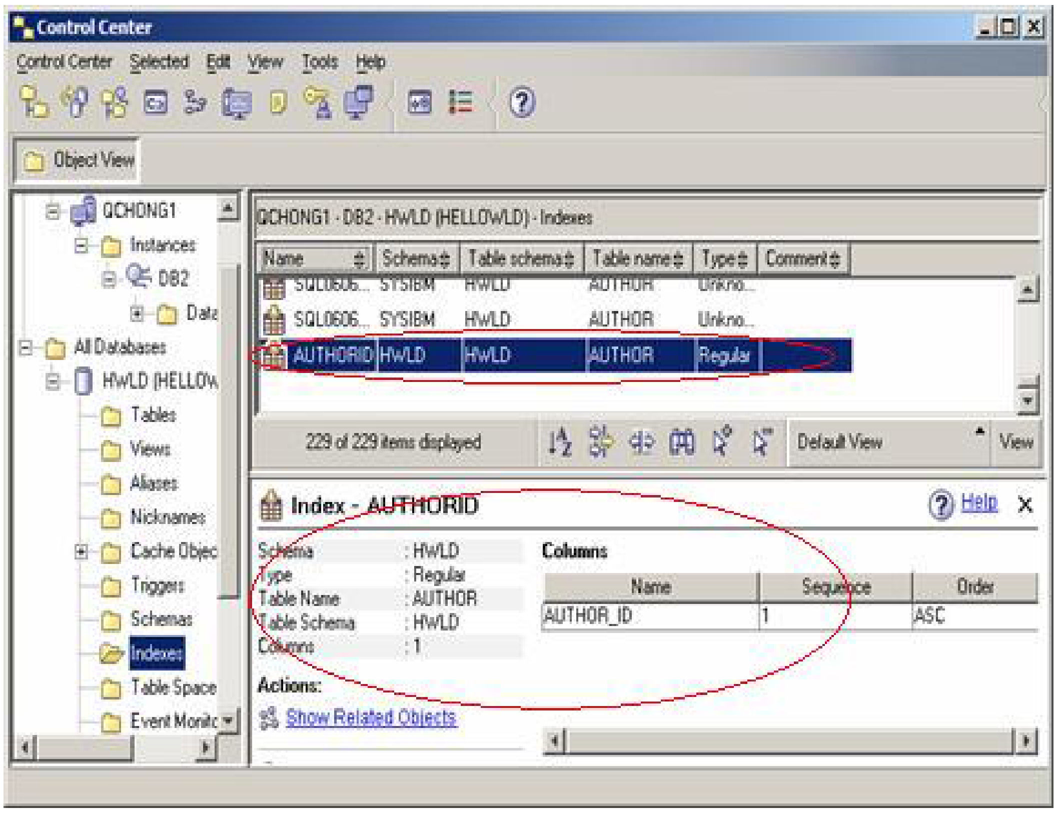
7. Click **Next** to go to the Summary page.

8. From the Summary page, use **Show SQL** to see the CREATE INDEX statement. If your table has data, use **Estimate Size** to check the space usage for the new index. Click **Finish** to let index creation complete. Message DB20000 should be returned to indicate the wizard completed

successfully. Close the message window.

9. From the Control Center Indexes view, select **AUTHORID** from index name. Take a look at the details for this index. Click **Show Related Objects** to view the related object information such as the base table HWLD.AUTHOR for this index, as well as the tablespace TBSP_INDEX1 where the index data resides.

Figure 18. Control Center -- Indexes



Name	Schema	Table schema	Table name	Type	Comment
SQL0606...	SYSIBM	HWLD	AUTHOR	Unkno...	
SQL0606...	SYSIBM	HWLD	AUTHOR	Unkno...	
AUTHORID	HWLD	HWLD	AUTHOR	Regular	

Index - AUTHORID

Schema	: HWLD
Type	: Regular
Table Name	: AUTHOR
Table Schema	: HWLD
Columns	: 1

Columns

Name	Sequence	Order
AUTHOR_ID	1	ASC

Section 8. Using DB2 objects: Creating constraints

Constraints allow the DBMS to enforce certain integrity of data in its tables by preventing incorrect or unexpected data from being entered into the table. The following constraints are used in DB2 to enforce data integrity:

Primary keys
Used to enforce uniqueness on one or a set of column values. Only one primary key can exist for each table.

Foreign keys (referential constraints)
Used to establish a referential relationship between two tables, typically referred to as the child and parent table, with action defined in the REFERENCE clause. The action is executed when data

inserted/deleted/updated in the parent or child table meets the predefined criteria. A foreign key should always reference a primary key or unique keys in the parent table.

Unique keys
Defined to enforce uniqueness on one or a set of column values. In contrast to the primary key, a table can have multiple unique keys. Unique keys also allow foreign keys to reference different data columns other than the primary key. A unique index is created whenever a primary key or a unique key is defined.

Check constraints
Defined at table level against one or more table columns, they are used to enforce rules specified for those table columns so that the data inserted or updated in those columns meets the conditions predefined in the check constraint. Unlike other constraints, check constraint may be temporarily turned off during certain operations, such as LOAD, and the table is placed into a check pending state until the constraint is checked.

Create constraints when creating a table, or add them later using the ALTER TABLE statement or a GUI interface.

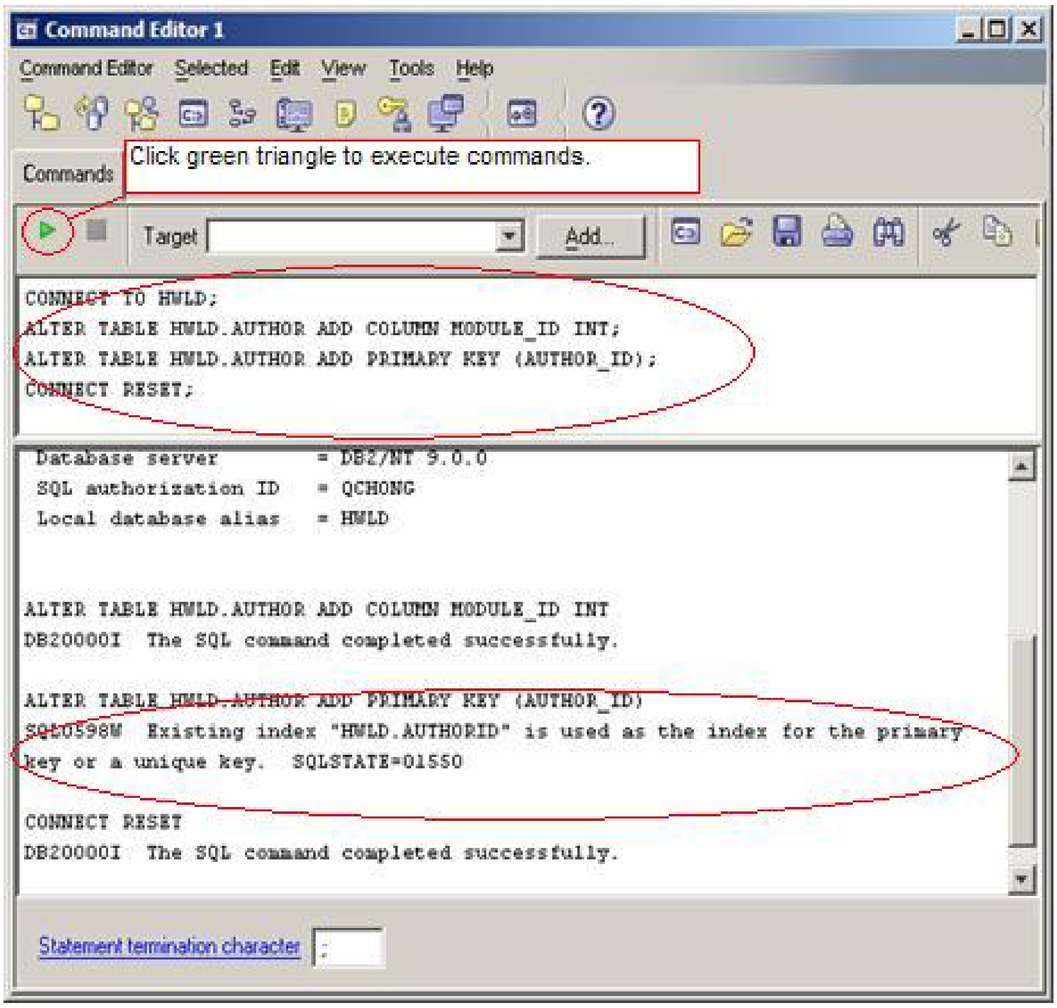
The following exercise gives you the chance to practice creating and using these constraints. It also demonstrates how to use the Command Editor to perform some of the daily administration tasks:

Would you like to see these steps demonstrated for you?

Show me

1. From the Control Center main menu bar, click **Tools > Command Editor** from the drop-down menu. A Command Editor window appears. Use this editor to edit your DB2 commands, execute them, and then view the returned DB2 or SQL message.

2. First add an additional column to the table AUTHOR and then add a primary key on this table. Add the following commands and click the green execute button as shown in Figure 35:
Figure 19. Command Editor -- Alter Table


Command Editor 1
Command Editor Selected Edit View Tools Help
Commands
Click green triangle to execute commands.
Target
Add...

```
CONNECT TO HWLD;
ALTER TABLE HWLD.AUTHOR ADD COLUMN MODULE_ID INT;
ALTER TABLE HWLD.AUTHOR ADD PRIMARY KEY (AUTHOR_ID);
CONNECT RESET;
```

```
 Database server        = DB2/NT 9.0.0
 SQL authorization ID   = QCHONG
 Local database alias   = HWLD


ALTER TABLE HWLD.AUTHOR ADD COLUMN MODULE_ID INT
DB20000I  The SQL command completed successfully.

ALTER TABLE HWLD.AUTHOR ADD PRIMARY KEY (AUTHOR_ID)
SQL0598W  Existing index "HWLD.AUTHORID" is used as the index for the primary
key or a unique key.  SQLSTATE=01550

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

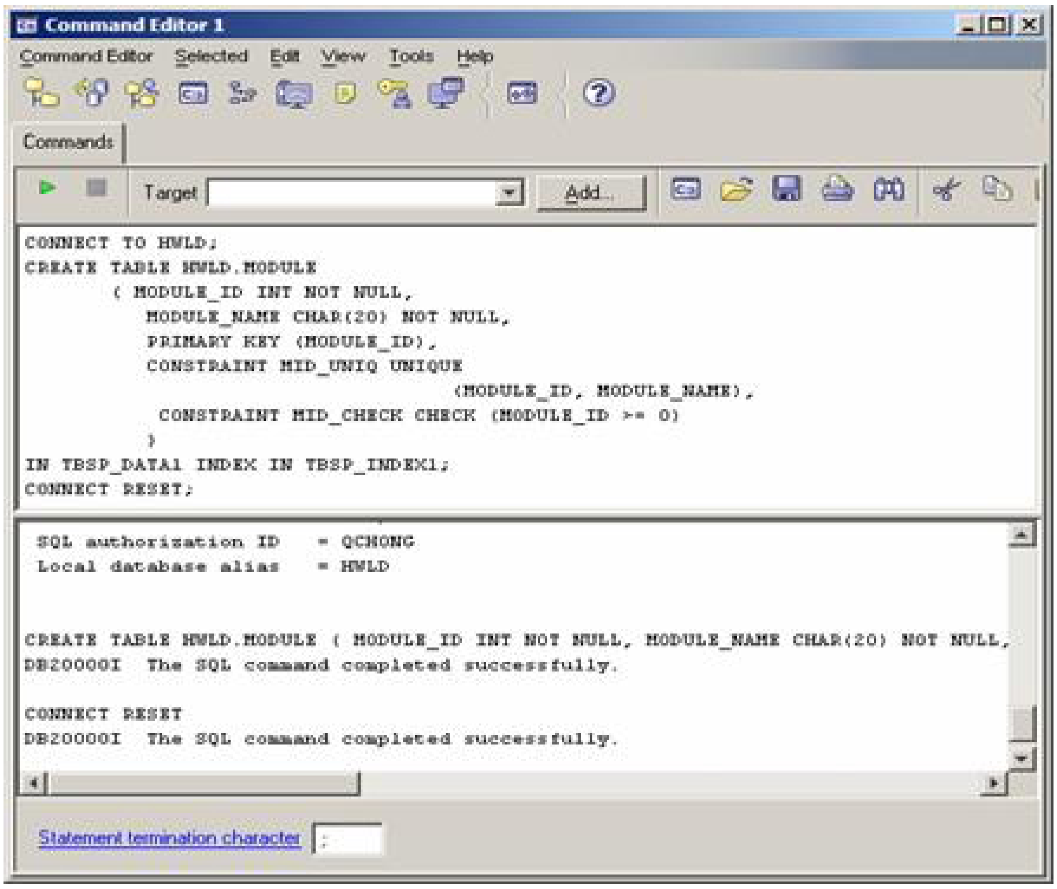
Statement termination character ;

When you add a primary key, DB2 creates a unique index underneath because there is already a unique index on AUTHOR_ID. DB2 uses an existing index for the primary key, so SQL0598W is returned.

3. From Command Editor, you can also choose to save the existing commands to a file or replace them with new commands you want to edit or run next.

4. Replace the commands in the Command Editor window with the following:

```
CONNECT TO HWLD;
CREATE TABLE HWLD.MODULE
( MODULE_ID INT NOT NULL,
MODULE_NAME CHAR(20) NOT NULL,
PRIMARY KEY (MODULE_ID),
CONSTRAINT MID_UNIQ UNIQUE
(MODULE_ID, MODULE_NAME),
CONSTRAINT MID_CHECK CHECK (MODULE_ID >= 0)
)
IN TBSP_DATA1 INDEX IN TBSP_INDEX1;
CONNECT RESET;
```

5. This `CREATE TABLE` command creates one primary key constraint, one unique key constraint, and one check constraint during table creation. The check constraint enforces positive values in data column MODULE_ID. Figure 20 shows the expected run results:
   Figure 20. Command Editor -- Create Table
   

```
CONNECT TO HWLD;
CREATE TABLE HWLD.MODULE
       ( MODULE_ID INT NOT NULL,
          MODULE_NAME CHAR(20) NOT NULL,
          PRIMARY KEY (MODULE_ID),
          CONSTRAINT MID_UNIQ UNIQUE
                                  (MODULE_ID, MODULE_NAME),
           CONSTRAINT MID_CHECK CHECK (MODULE_ID >= 0)
          )
IN TBSP_DATA1 INDEX IN TBSP_INDEX1;
CONNECT RESET;
```

```
 SQL authorization ID    = QCHONG
 Local database alias    = HWLD


CREATE TABLE HWLD.MODULE ( MODULE_ID INT NOT NULL, MODULE_NAME CHAR(20) NOT NULL,
DB20000I  The SQL command completed successfully.

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

   You can use the CREATE TABLE wizard to accomplish the same thing.

6. Now you have created two tables. Table HWLD.AUTHOR contains all the author information on the Hello World modules. Table HWLD.MODULE contains all the Hello World series module information. Now create another table, HWLD.PARTICIPANT, to contain all the participants' information including name, department (optional), the module taken, and the quiz score.
   Table HWLD.PARTICIPANT should have the following table definitions:

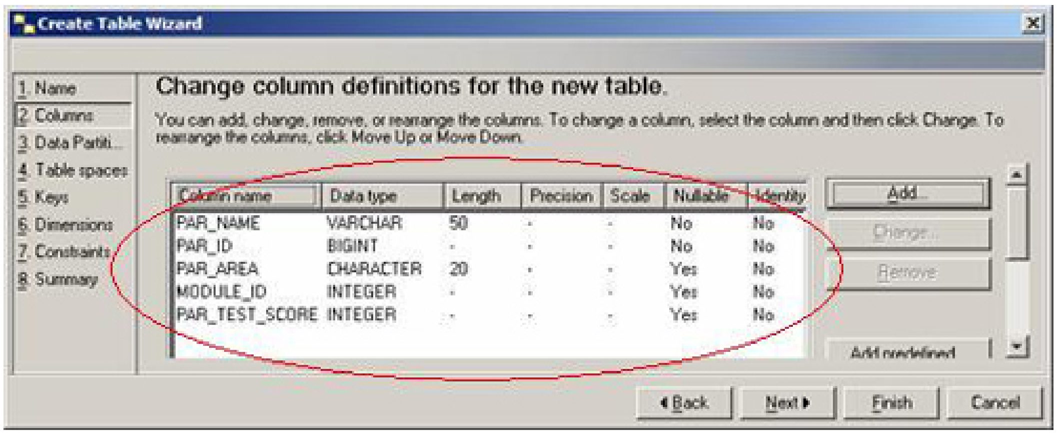
```
(PAR_NAME VARCHAR(50) NOT NULL,
PAR_ID BIGINT NOT NULL,
PAR_AREA CHAR(20),

MODULE_ID INT,

PAR_TEST_SCORE INT)
```

7. Note the following conditions:
   - Each person can attempt multiple modules multiple times
   - Test score should be between 0 and 100
   - The MODULE_ID in HWLD.PARTICIPANT should refer to the MODULE_ID in HWLD.MODULE. When a module is deleted from table HWLD.MODULE, set the corresponding MODULE_ID to NULL in PARTICIPANT table, so the MODULE_ID here is never an invalid value.
   - When a row is inserted into HWLD.PARTICIPANT table, if the MODULE_ID does not exist in HWLD.MODULE, fail the insert.
   - When a module is updated in table HWLD.MODULE, if there is a record existing in HWLD.PARTICIPANT table referring to that MODULE_ID, fail the update.
8. Add the following constraints to enforce the preceding conditions:
   - Check constraint on column PAR_TEST_SCORE to ensure its values are between 0 and 100
   - Foreign key on MODULE_ID to reference to MODULE_ID in HWLD.MODULE, ON DELETE SET NULL, ON UPDATE RESTRICT (Note, the insert rule 4 is already forced by default).

   These constraints can be created when table HWLD. PARTICIPANT is created.

9. From the Control Center, launch the Create New Table wizard from the Tables view. Select **HWLD** as Table schema, type `PARTICIPANT` as Table name, and add a comment. Click **Next**.
10. From the Columns page, click **Add** to add the first column. Specify `PAR_NAME VARCHAR(50) NOT NULL` and choose **VARCHAR** as the data type.
11. Click **OK**. From the Columns page, add a second column, `PAR_ID BIGINT NOT NULL`. Click **OK**. Add a third column, `PAR_AREA CHAR(20)`. Notice, this column allows a NULL value.
12. Similarly, add the fourth column, `MODULE_ID INT` and then the fifth column, `PAR_TEST_SCORE INT`. Both have Data type as INTEGER and both allow NULL values.

13. Double-check the Columns you have added.
Figure 21. Create Table Wizard -- Columns



Change column definitions for the new table.

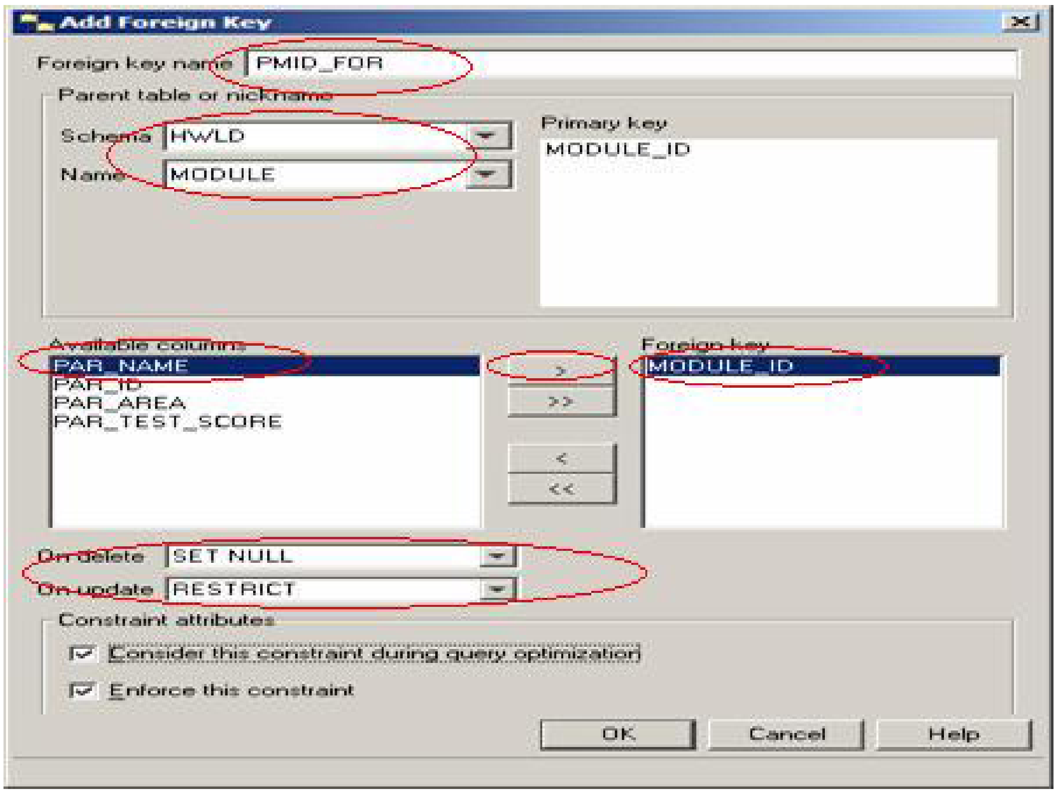
You can add, change, remove, or rearrange the columns. To change a column, select the column and then click Change. To rearrange the columns, click Move Up or Move Down.

Column name	Data type	Length	Precision	Scale	Nullable	Identity
PAR_NAME	VARCHAR	50	-	-	No	No
PAR_ID	BIGINT	-	-	-	No	No
PAR_AREA	CHARACTER	20	-	-	Yes	No
MODULE_ID	INTEGER	-	-	-	Yes	No
PAR_TEST_SCORE	INTEGER	-	-	-	Yes	No

14. Click **Next**. Skip the Data Partitions page by clicking **Next** again.

15. Similar to the previous two tables created, use separate tablespaces for data and index for the table, HWLD.PARTICIPANT. Select **TBSP_DATA1** for Table space and select **TBSP_INDEX1** as index space. Click **Next**.

16. Click **Add Foreign** from Keys page. Enter `PMID_FOR` as the Foreign key name.
Figure 22. Create Table Wizard -- Add Foreign Key



Foreign key name: PMID_FOR

Schema: HWLD

Name: MODULE

Primary key: MODULE_ID

Available columns: PAR_NAME, PAR_ID, PAR_AREA, PAR_TEST_SCORE

Foreign key: MODULE_ID

On delete: SET NULL

On update: RESTRICT

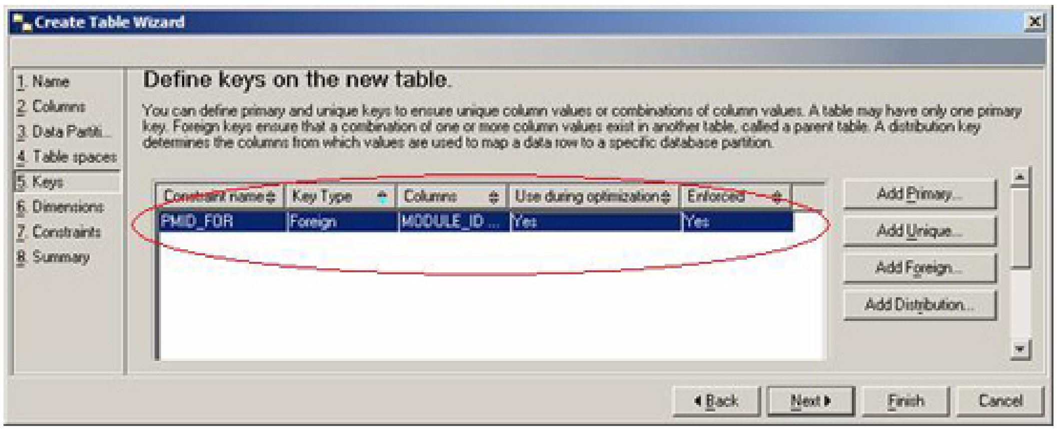
☑ Consider this constraint during query optimization

☑ Enforce this constraint

17. Select **HWLD.MODULE** as the parent table. Select **MODULE_ID** from Available columns, and click > to add it as Foreign key. Select **SET NULL**

for On delete, and select **RESTRICT** for On update. Click **OK**. You should see the following constraint defined:

Figure 23. Create Table Wizard -- Keys



Create Table Wizard

Define keys on the new table.

You can define primary and unique keys to ensure unique column values or combinations of column values. A table may have only one primary key. Foreign keys ensure that a combination of one or more column values exist in another table, called a parent table. A distribution key determines the columns from which values are used to map a data row to a specific database partition.

Constraint name	Key Type	Columns	Use during optimization	Enforced
PMID_FOR	Foreign	MODULE_ID ...	Yes	Yes

Click **Next**. (You can create different type of keys such as unique, primary keys here. In this exercise, you just create a foreign key.)

18. Click **Next** again to skip the Dimensions page.

19. From Define Check Constraints page, click **Add** to start the Add Check Constraint wizard.

20. Enter `PAR_SCORE_CHECK` as the Check Constraint name. Add the following as Check condition:

```
PAR_TEST_SCORE BETWEEN 0 and 100
```

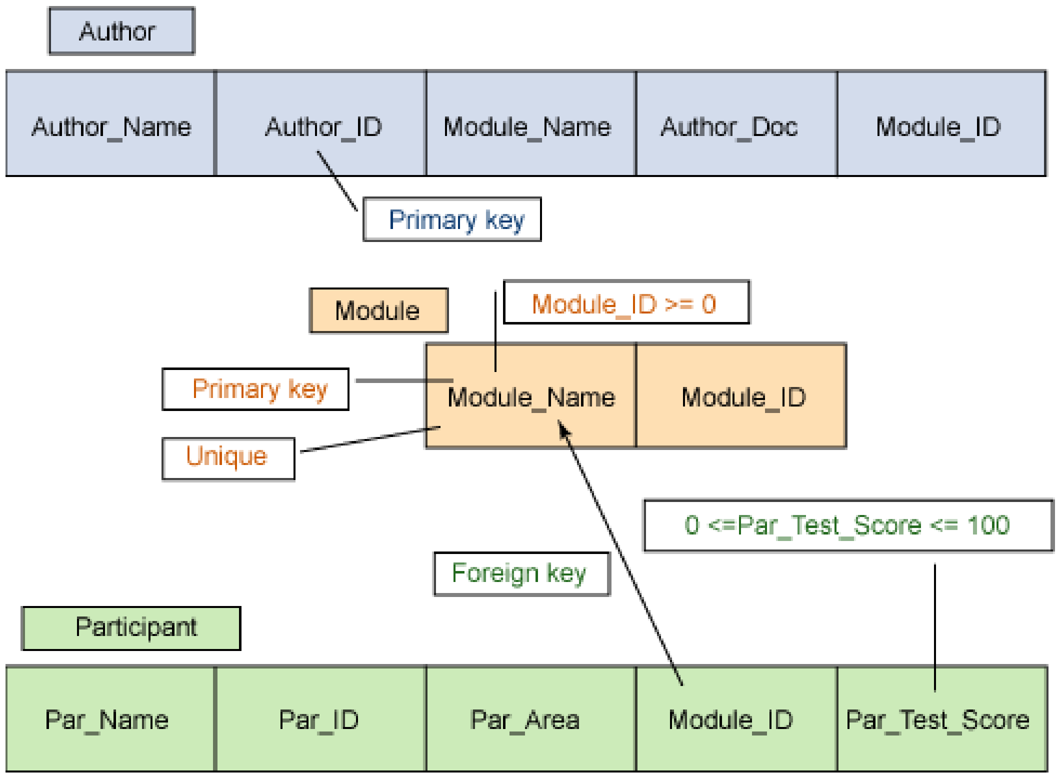
    Select both Constraint attributes.

21. Type in a comment. Click **OK** to add this constraint. Click **Finish**. DB20000 should be returned in the DB2 Message window. Close it.

22. Check to see if the newly created PARTICIPANT table is displayed in the Control Center.
    Review the HWLD.PARTICIPANT and see if you can think of any other conditions which you can use constraints to implement. (Hint: the current implementation does not enforce the uniqueness of participants and their assigned participant IDs; there are a couple of ways to implement this. One way is to break up the current HWLD.PARTICIPANT table into two tables, force uniqueness on the participant ID, and establish a foreign key to reference participant ID. Can you think of any other ways?)

    The following figure summarizes the three tables created so far and the constraints you have added to these tables.

    Figure 24. Tables just created and their Constraints



Section 9. Using DB2 Objects: Triggers and UDTs/UDFs

The following concepts are related to tables and may help you better understand the subsequent exercise.

Triggers

Triggers provide a means for the Database Management System (DBMS) to enforce certain business requirements, such as data validation, and enforcing rules and referential integrity of the data. Triggers can be defined against a base table. When one or a set of predefined conditions are met, one or a set of actions can be triggered to perform against the table. A trigger can be fired either before or after an insert, delete or update.

Triggers are not covered in detail here. Refer to the *DB2 Administration Guide* for additional information.

UDTs and UDFs

In addition to the native data types that DB2 provides, users can create User-Defined Types (UDTs) using native DB2 data types. Users can also create their own User-Defined Functions (UDFs). Both UDTs and UDFs provide additional flexibility to the database application developer to implement their applications.

For details on User-Defined Types and User-Defined Functions, refer to the *DB2 Administration Guide*.

Section 10. Inserting and deleting data from the table

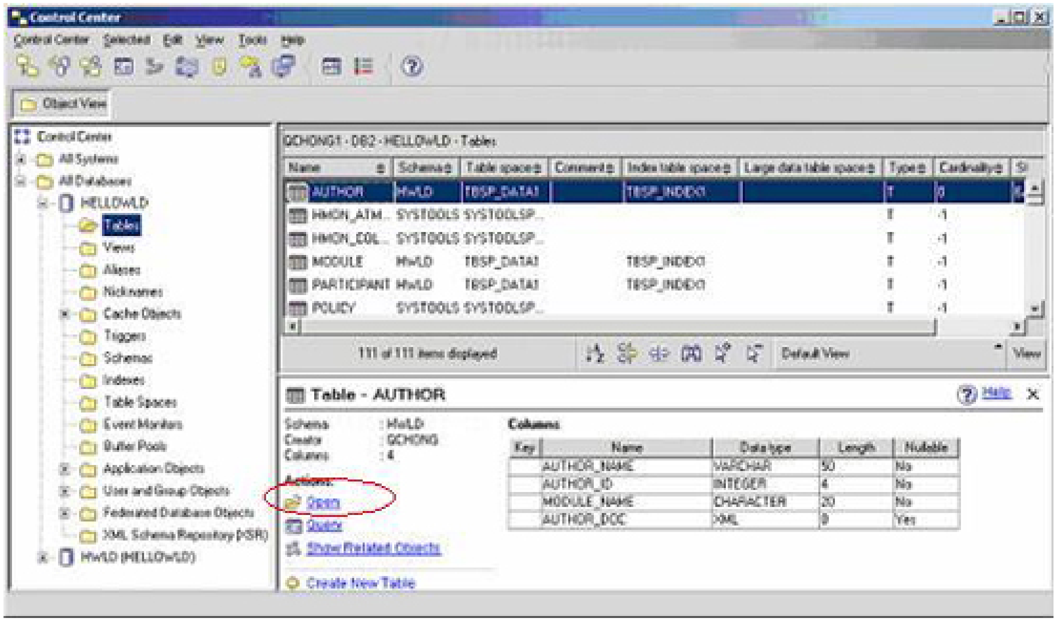
Once you have created a table to store your data, you need to be able to get the data in and out of the table to make use of it. The simplest way to get data into the table is using the INSERT command. To insert data, provide the table and columns you want to insert into and the data itself. Once the data is inserted, you select, update, or delete it.

This exercise demonstrates how to move data in and out of a database.

Would you like to see these steps demonstrated for you?

Show me

1. From the Control Center Object view, select the AUTHOR table from the database list and click **Open**.
   Figure 25. Control Center -- Open table



QCHONG1 - DB2 - HELLOWLD - Tables

Name	Schema	Table space	Comment	Index table space	Large data table space	Type	Cardinality
AUTHOR	HWLD	TBSP_DATA1		TBSP_INDEX1		T	0
HMON_ATM...	SYSTOOLS	SYSTOOLSP...				T	-1
HMON_COL...	SYSTOOLS	SYSTOOLSP...				T	-1
MODULE	HWLD	TBSP_DATA1		TBSP_INDEX1		T	-1
PARTICIPANT	HWLD	TBSP_DATA1		TBSP_INDEX1		T	-1
POLICY	SYSTOOLS	SYSTOOLSP...				T	-1

Table - AUTHOR

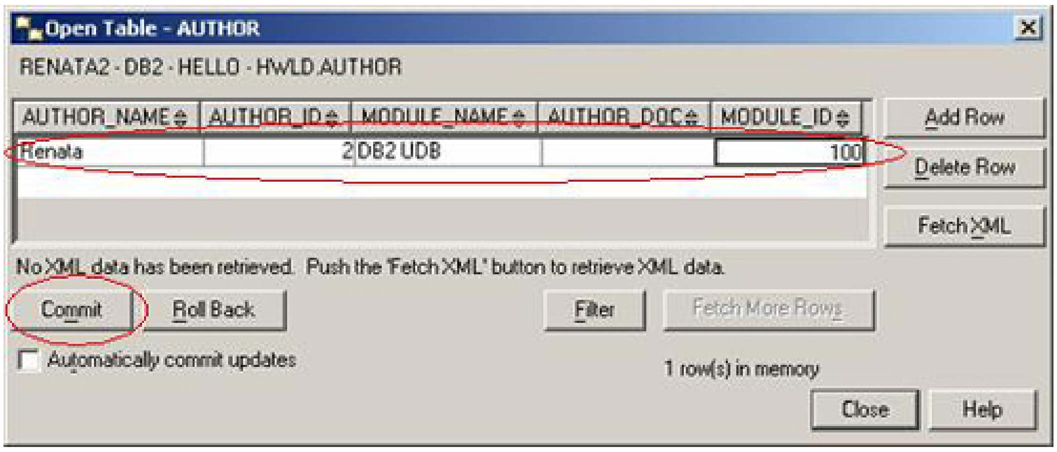
Schema : HWLD
Creator : QCHONG
Columns : 4

Columns

Key	Name	Data type	Length	Nullable
	AUTHOR_NAME	VARCHAR	50	No
	AUTHOR_ID	INTEGER	4	No
	MODULE_NAME	CHARACTER	20	No
	AUTHOR_DOC	XML	0	Yes

2. Click **Add Row** as shown in the following figure. Enter data into the columns and commit the transaction.

Figure 26. Add Row



AUTHOR_NAME	AUTHOR_ID	MODULE_NAME	AUTHOR_DOC	MODULE_ID
Renata	2	DB2 UDB		100

No XML data has been retrieved. Push the 'Fetch XML' button to retrieve XML data.

1 row(s) in memory

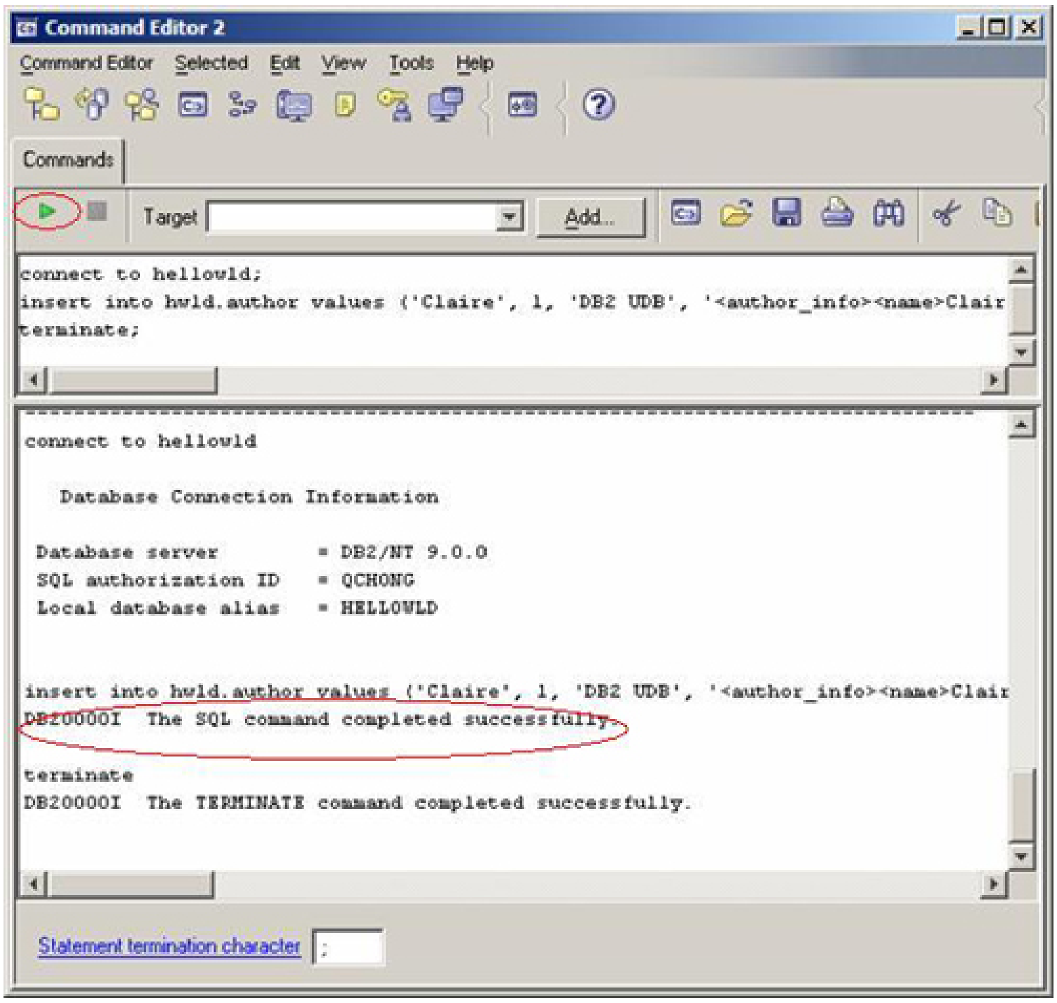
3. If no errors were generated, there is no feedback. Click **Close** to close the Open Table window.

4. Open the Command Editor with **Tools > Command Editor**. Enter the INSERT command manually, including the statement to connect to the database:

```
connect to hellowld;
insert into hwld.author values ('Claire', 1, 'DB2 UDB',
'<author_info>
<name>Claire</name>
<ID>1</ID>
<department>
<department_title>DB2 CE</department_title>
<responsibility>Supporting customer problems</responsibility>
<start_year>2005</start_year>
</department>
<department>
<department_title>DB2 Index Manager</department_title>
<responsibility>Develop new features for DB2 indexes</responsibility>
<start_year/>
</department>
</author_info>', 100);
terminate;
```

(You can also perform the same action using the command editor view.)

5. Click the green triangle to run the command. Figure 47 shows the expected run results:
Figure 27. Command Editor -- Insert



```
connect to hellowld;
insert into hwld.author values ('Claire', 1, 'DB2 UDB', '<author_info><name>Clair
terminate;
```

```
------------------------------------------------------------------------------
connect to hellowld

   Database Connection Information

 Database server        = DB2/NT 9.0.0
 SQL authorization ID   = QCHONG
 Local database alias   = HELLOWLD


insert into hwld.author values ('Claire', 1, 'DB2 UDB', '<author_info><name>Clair
DB20000I  The SQL command completed successfully.

terminate
DB20000I  The TERMINATE command completed successfully.
```

6. If you entered some data that you no longer want to have in the table, you can delete it using the same window you used to insert. Click the table **AUTHOR** and **Open**.

7. Select the first row and click **Delete Row**. Click **Commit** to commit the delete operation. If you decide to abort the delete operation, click **Roll Back** instead of **Commit**.

8. Click on the remaining row and delete it to empty the table, then click **Close**.

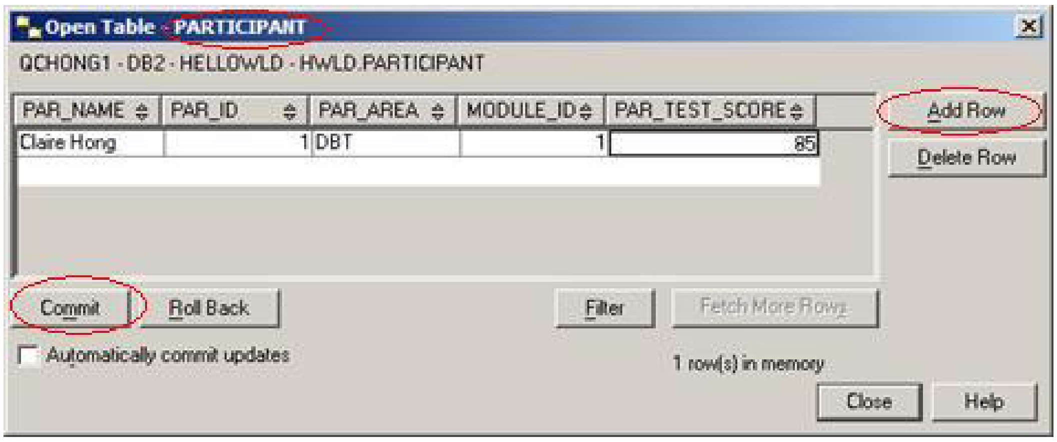
Section 11. Validating constraints

As you learned earlier, constraints are used to ensure invalid data is not used in a table. Rows that don't meet the defined constraints are rejected. In this exercise, you are going to validate constraints to see what happens when invalid data is entered.

The table PARTICIPANT was created with a foreign key, which means that for each value in the column PAR_ID in PARTICIPANT, there must be a row in the MODULE table with the same value in MODULE_ID column. Because the MODULE table is currently empty, you do not satisfy this condition if you enter data into PARTICIPANT table. Follow the steps to validate the constraints:

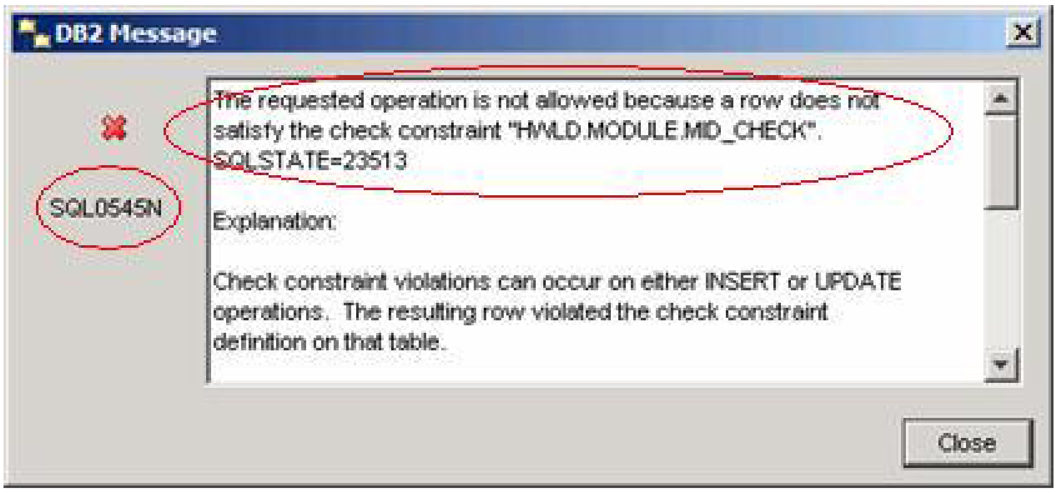
Would you like to see these steps demonstrated for you?

Show me

1. From the Control Center click the PARTICIPANT table and then click **Open**. In the Open Table panel, click **Add Row**. Enter the following row as shown into the PARTICIPANT table, and click **Commit**.
Figure 28. Control Center -- Insert Row

Open Table - PARTICIPANT
QCHONG1 - DB2 - HELLOWLD - HWLD.PARTICIPANT

PAR_NAME	PAR_ID	PAR_AREA	MODULE_ID	PAR_TEST_SCORE
Claire Hong	1	DBT	1	85

2. The message SQL0530N is returned, stating you do not meet the requirements of referential integrity enforced with foreign key constraint.
3. Click **Close** once to close the message and a second time to close the Open Table window.
4. Validate the check constraint you have defined on the MODULE table. Because the MODULE table was created with a check constraint ("CONSTRAINT MID_CHECK CHECK (MODULE_ID >= 0)"), MODULE_ID cannot contain a negative value.
5. From the Control Center click the MODULE table and then click **Open**. In the Open Table panel, enter a row into the MODULE table with a negative value for MODULE_ID and assign it a name for MODULE_NAME. Try it out and see what happens. The following error tells you that you should not be doing that:
Figure 29. Validate check constraint



The requested operation is not allowed because a row does not satisfy the check constraint "HWLD.MODULE.MID_CHECK". SQLSTATE=23513

Explanation:

Check constraint violations can occur on either INSERT or UPDATE operations. The resulting row violated the check constraint definition on that table.

6. Click **Close** to close the message and **Close** again to close Open Table window.

Section 12. Importing, loading, and exporting data

Inserting data one row at a time can be very time consuming, especially for companies with thousands of employees or millions of customer transactions. Import and load utilities simplify this task. Also, sometimes it is useful to place some of the data from the tables into a data file for further analysis and processing. The export utility is helpful in this case. Import, load and export utilities can also be used to move data from one table to another or from one database to another.

In the following exercise you are going to import data from a file on disk and then export it back from the table into a different file. The files were created for this tutorial and can be found in the Download section. Then you are going to use the load utility to populate a table with data stored in a file.

Though import, export, and load utilities have numerous options available, this exercise, focuses on the following:

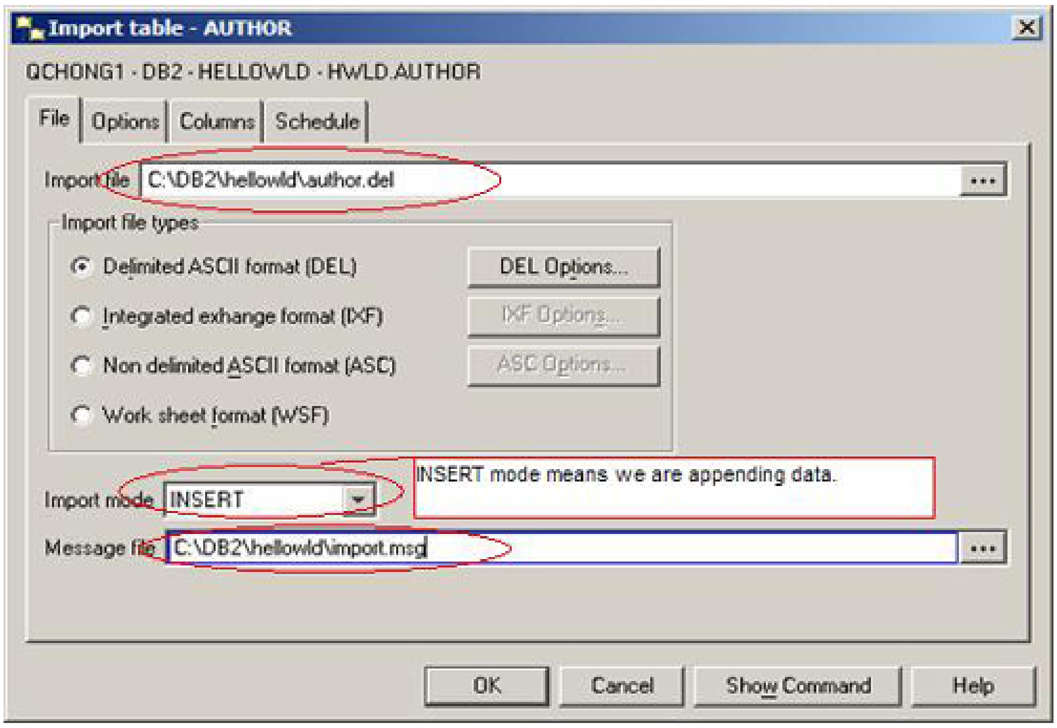
- *File type* -- Specifies the format of the file to store the data. In this example, DEL means delimited ASCII file
- *Select statement* -- Indicates you have a choice of selecting all columns from a table during Export (select *) or specific columns (select <column list>). In this example, you select all columns.
- *Import/Load operation* -- Indicates you have a choice during Import to insert the data (append), replace it, update duplicates, or create a table using the data file. In this example, you insert the data.

For a complete description of these utilities, refer to the *DB2 Data Movement Utilities Guide and Reference* (see Resources).

Would you like to see these steps demonstrated for you?

Show me

1. Use import to populate the HWLD.AUTHOR table. From the Control Center, right-click the AUTHOR table and choose **Import**.

2. Select options as shown:
Figure 30. Import Table -- Author


Import file: C:\DB2\hellowld\author.del

(●) Delimited ASCII format (DEL)
( ) Integrated exhange format (IXF)
( ) Non delimited ASCII format (ASC)
( ) Work sheet format (WSF)

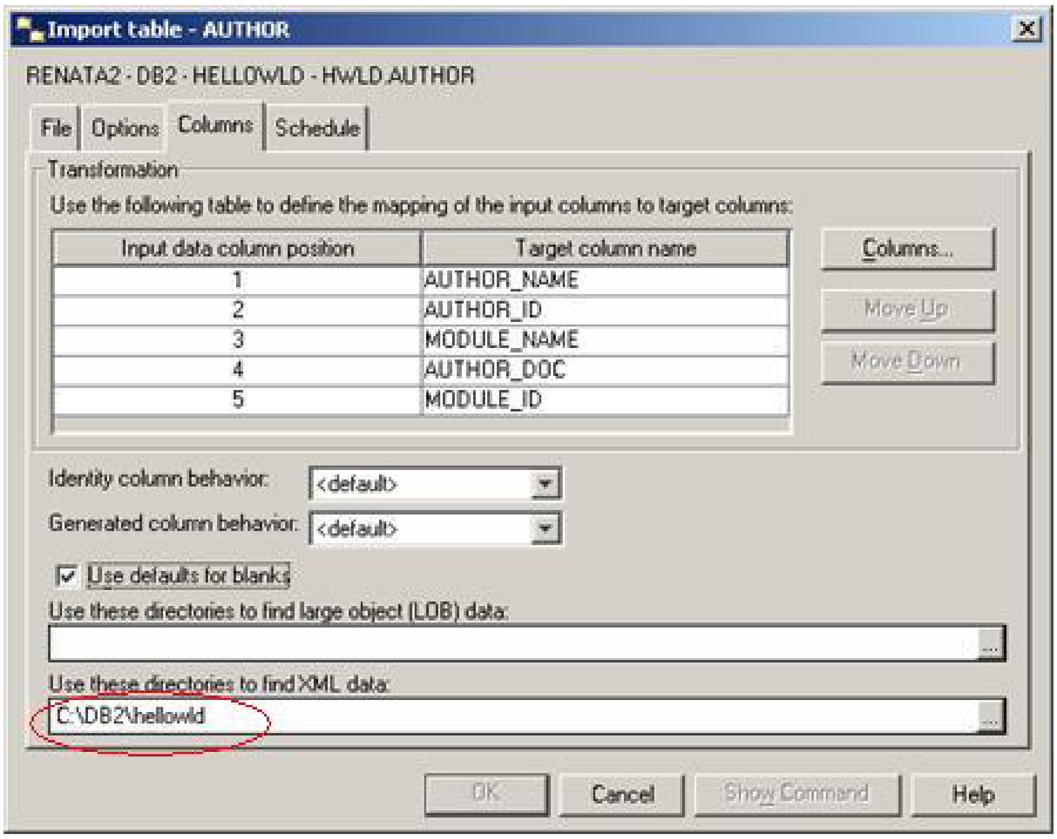
Import mode: INSERT

INSERT mode means we are appending data.

Message file: C:\DB2\hellowld\import.msg

Previously you told DB2 to use the file author.del (included in the zip file that you downloaded) and use it to populate the table HWLD.AUTHOR. Any messages related to import are placed in the message file C:\DB2\hellowld\import.msg.

3. From the Columns tab, enter the path to find XML columns.
Figure 31. Import table -- Author


Import table - AUTHOR

RENATA2 - DB2 - HELLOWLD - HWLD.AUTHOR

Transformation

Use the following table to define the mapping of the input columns to target columns:

Input data column position	Target column name
1	AUTHOR_NAME
2	AUTHOR_ID
3	MODULE_NAME
4	AUTHOR_DOC
5	MODULE_ID

Identity column behavior: <default>

Generated column behavior: <default>

☑ Use defaults for blanks

Use these directories to find large object (LOB) data:

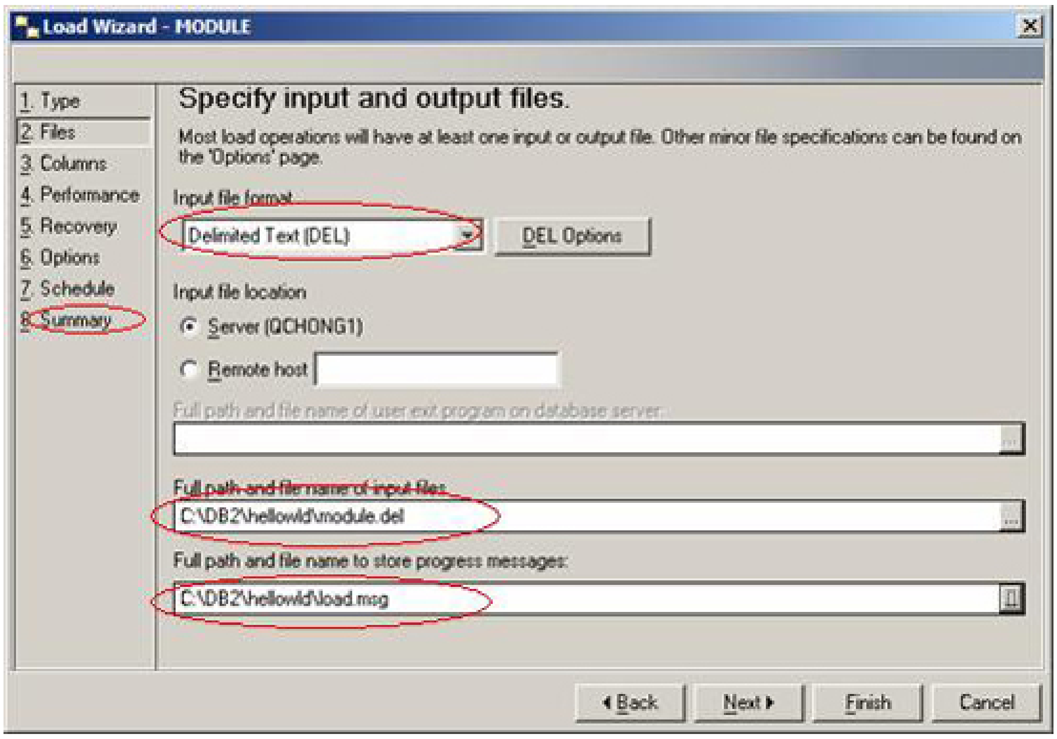
Use these directories to find XML data: C:\DB2\hellowld

You can click **Show Command** to see the full import statement and save it if you want. The message DB20000 should indicate that everything went OK.

4. Use the load command to populate HWLD.MODULE table. Right-click MODULE table and select **Load**.

5. Select **Append data to table** from the Load Wizard. Click **Next**.

6. For Full path and file name of input file, locate the file `module.del` (included in the zip file that you downloaded) and specify the message file.
   Figure 32. Load Wizard -- File


Load Wizard - MODULE

Specify input and output files.

Most load operations will have at least one input or output file. Other minor file specifications can be found on the 'Options' page.

Input file format: Delimited Text (DEL)

Input file location: Server (QCHONG1)

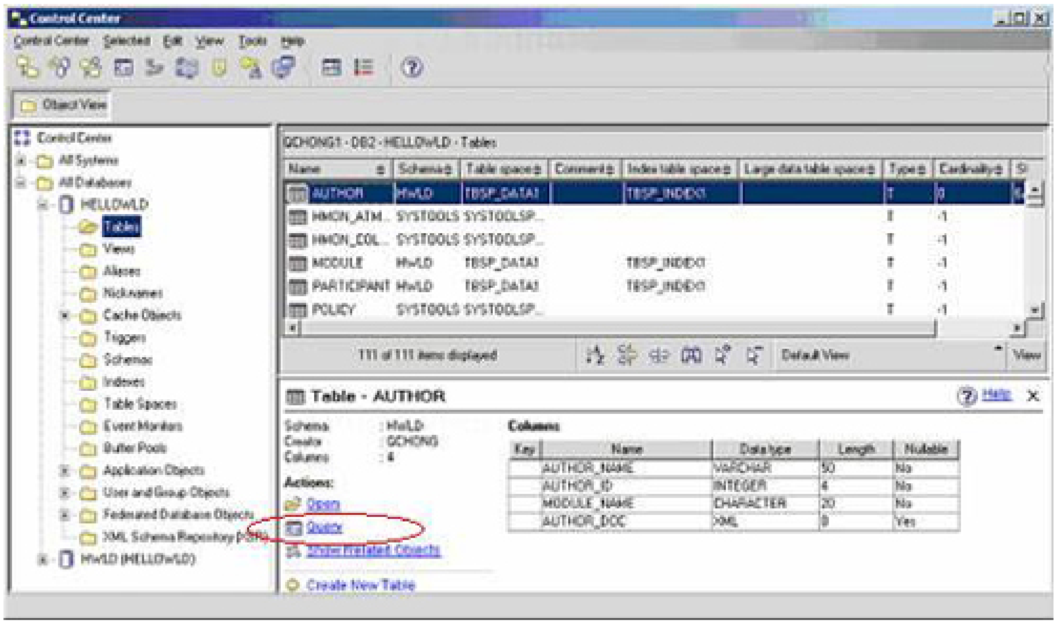
Full path and file name of input files: C:\DB2\hellowld\module.del

Full path and file name to store progress messages: C:\DB2\hellowld\load.msg

7. Click **Summary**. You see a load statement that you can save to run on the command line. If you click **Finish**, the command should execute successfully, and you should get a dialog window with DB20000 message.

8. Validate the integrity of the data and query data.
Because the module table has integrity constraints defined, after the load utility is run, the integrity must be validated for the table can't be accessed. To do that, run the SET INTEGRITY statement from the command editor. Set integrity checks to ensure that each row satisfies the constraints condition and remove any row that does not meet the condition.

9. Select the MODULE table and click **Query**.
Figure 33. Query Table HWLD.MODULE



QCHONG1 - DB2 - HELLOWLD - Tables

Name	Schema	Table space	Comment	Index table space	Large data table space	Type	Cardinality
AUTHOR	HWLD	TBSP_DATA1		TBSP_INDEX1		T	0
HMON_ATM...	SYSTOOLS	SYSTOOLSP...				T	-1
HMON_COL...	SYSTOOLS	SYSTOOLSP...				T	-1
MODULE	HWLD	TBSP_DATA1		TBSP_INDEX1		T	-1
PARTICIPANT	HWLD	TBSP_DATA1		TBSP_INDEX1		T	-1
POLICY	SYSTOOLS	SYSTOOLSP...				T	-1

Table - AUTHOR

Schema : HWLD
Creator : QCHONG
Columns : 4

Columns

Key	Name	Data type	Length	Nullable
	AUTHOR_NAME	VARCHAR	50	No
	AUTHOR_ID	INTEGER	4	No
	MODULE_NAME	CHARACTER	20	No
	AUTHOR_DOC	XML	0	Yes

This runs the default statement SELECT * from the table.

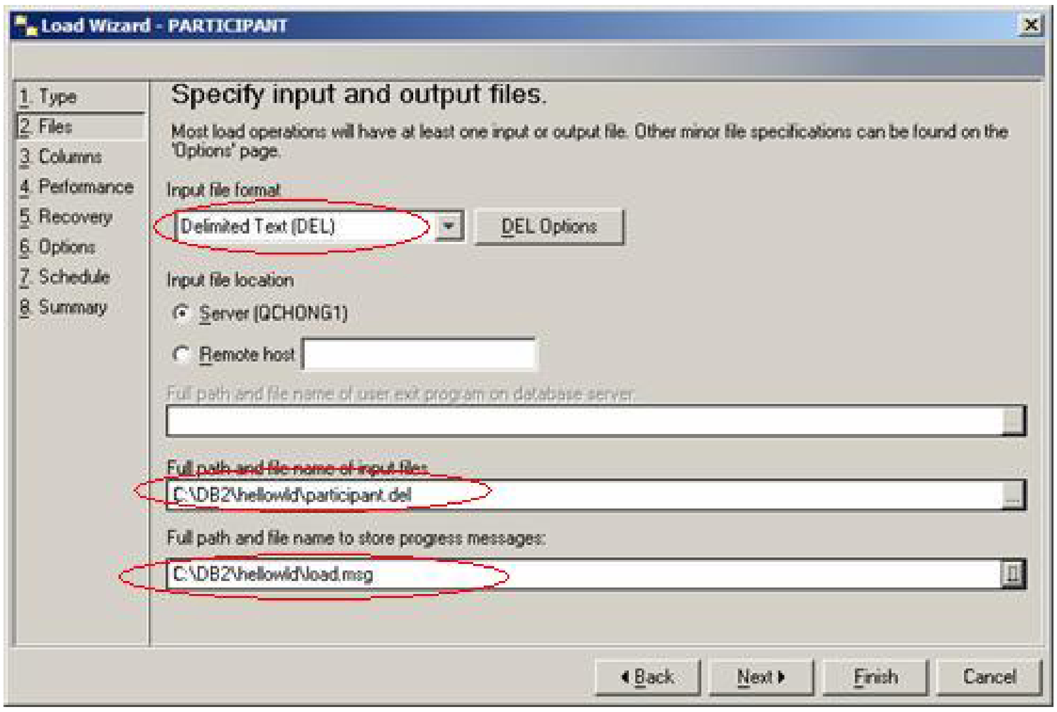
10. Before running select, add the following SET INTEGRITY statement:

```
------------------------------ Commands Entered ------------------------------
set integrity for hwld.module immediate checked;
SELECT * FROM HWLD.MODULE;
------------------------------------------------------------------------------
set integrity for hwld.module immediate checked
DB20000I The SQL command completed successfully.

SELECT * FROM HWLD.MODULE

MODULE_ID MODULE_NAME
----------- --------------------
1 DB2 UDB
2 WebSphere
3 Tivoli
4 Eclipse
5 Rational
6 Lotus Notes
7 Web Commerce

7 record(s) selected.
```

11. Load the HWLD.PARTICIPANT table. From the Control Center, right-click **PARTICIPANT** table and select **Load**. Select the load options as

specified in Figure 58, similar to what was done for the HWLD.MODULE table. The file `participant.del` is included in the zip file that you downloaded.

Figure 34. Load wizard



Specify input and output files.

Most load operations will have at least one input or output file. Other minor file specifications can be found on the 'Options' page.

Input file format: Delimited Text (DEL)

Input file location: [✓ Server (QCHONG1)] Remote host

Full path and file name of user exit program on database server

Full path and file name of input files: C:\DB2\hellowld\participant.del

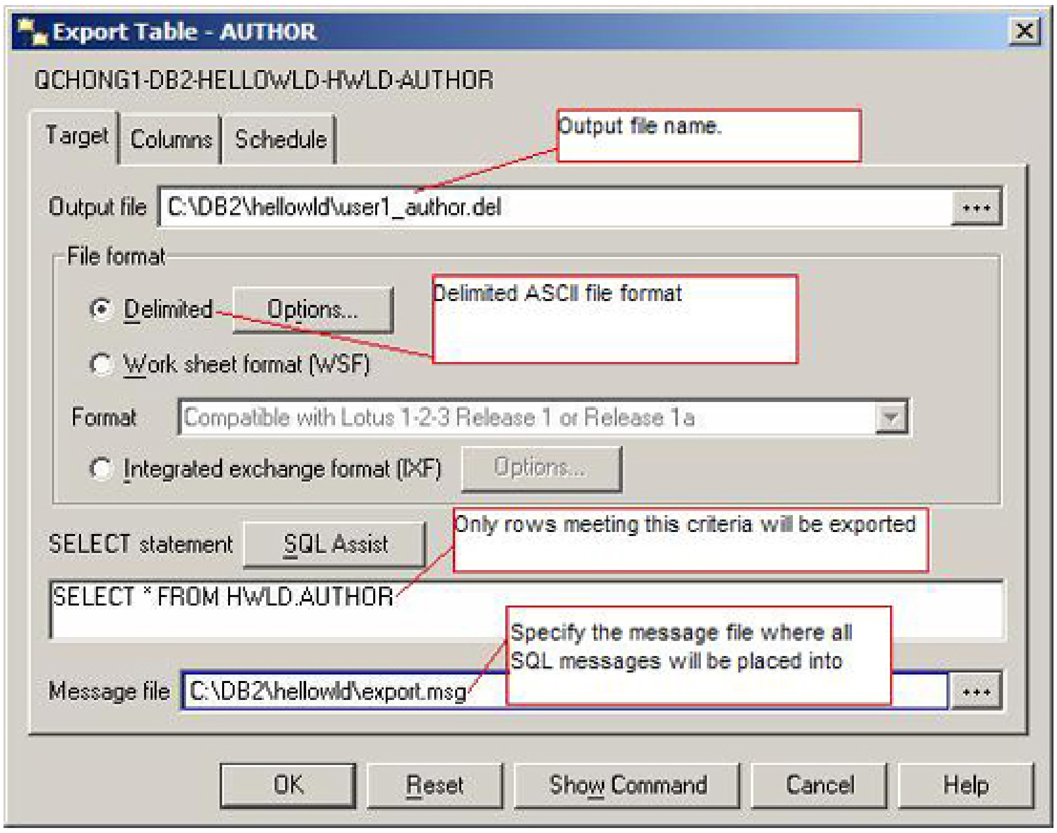
Full path and file name to store progress messages: C:\DB2\hellowld\load.msg

Again, because the participant table has referential constraints, load puts it into check pending state, so it cannot be accessed, until all rows are validated.

12. Click the **PARTICIPANT** table and select **Query**. Before the select statement, enter the SET INTEGRITY statement. Change the select statement to include only the participant name, ID, and test score for those participants whose ID is 4:

```
------------------------------ Commands Entered -----------------------------
set integrity for hwld.participant immediate checked;
SELECT PAR_NAME, PAR_ID, PAR_TEST_SCORE FROM HWLD.PARTICIPANT where PAR_ID=4;
-----------------------------------------------------------------------------
set integrity for hwld.participant immediate checked
DB20000I The SQL command completed successfully.

SELECT PAR_NAME, PAR_ID, PAR_TEST_SCORE FROM HWLD.PARTICIPANT where PAR_ID=4

PAR_NAME PAR_ID PAR_TEST_SCORE
----------------------------------------- -------------------- --------------
mike 4 60
mike 4 80
```

```
2 record(s) selected.
```

13. Now that the tables are populated, export the data into a data file. To select what table to export from, right-click the **AUTHOR** table and choose **Export**. The following window opens:
Figure 35. Export table

Export Table - AUTHOR
QCHONG1-DB2-HELLOWLD-HWLD-AUTHOR
Output file: C:\DB2\hellowld\user1_author.del
File format: (●) Delimited
Format: Compatible with Lotus 1-2-3 Release 1 or Release 1a
SELECT statement: SELECT * FROM HWLD.AUTHOR
Message file: C:\DB2\hellowld\export.msg
Target
Columns
Schedule
Output file name.
Work sheet format (WSF)
Integrated exchange format (IXF)
Options...
Delimited ASCII file format
SQL Assist
Only rows meeting this criteria will be exported
Specify the message file where all SQL messages will be placed into
OK
Reset
Show Command
Cancel
Help

14. Enter required information as shown above. From the Columns tab, select the option to place each XML document in a separate file. You can click Show Command to see the command and save it or run it later on the command line.
You should see `DB20000` to indicate everything is OK.

The output file is C:\DB2\hellowld\user1_author.del. Contents of the file user1_author.del are:

```
"Claire",1,"DB2 UDB ","<XDS FIL='author.del.001.xml'/>", 100
"Renata",2,"DB2 UDB ","<XDS FIL='author.del.002.xml'/>", 98
"Jane",3,"WEB COMMERCE ","<XDS FIL='author.del.003.xml'/>", 89
```

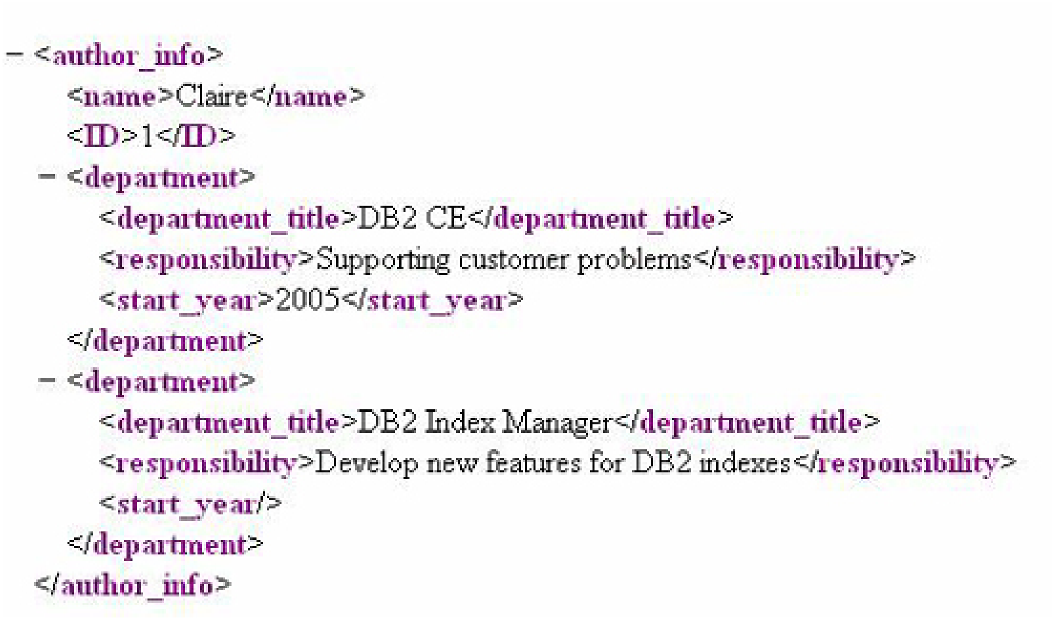
Notice that the column MODULE_NAME is padded to 20 characters. Also the XML documents are not actually in the DEL file, only a reference (XDS) to the file containing XML data exists.

Following are the contents of author.del.001.xml formatted for readability:

```
<?xml version="1.0" encoding="UTF-8" ?>
<author_info>
<name>Claire</name>
<ID>1</ID>
<department>
<department_title>DB2 CE</department_title>
<responsibility>Supporting customer problems</responsibility>
<start_year>2005</start_year>
</department>
<department>
<department_title>DB2 Index Manager</department_title>
<responsibility>Develop new features for DB2 indexes</responsibility>
<start_year/>
</department>
</author_info>
```

If you use a Web browser to view the XML document, you see the following:

Figure 36. XML Data



```
- <author_info>
    <name>Claire</name>
    <ID>1</ID>
  - <department>
      <department_title>DB2 CE</department_title>
      <responsibility>Supporting customer problems</responsibility>
      <start_year>2005</start_year>
    </department>
  - <department>
      <department_title>DB2 Index Manager</department_title>
      <responsibility>Develop new features for DB2 indexes</responsibility>
      <start_year/>
    </department>
  </author_info>
```

Section 13. Querying the data from a table and from a view

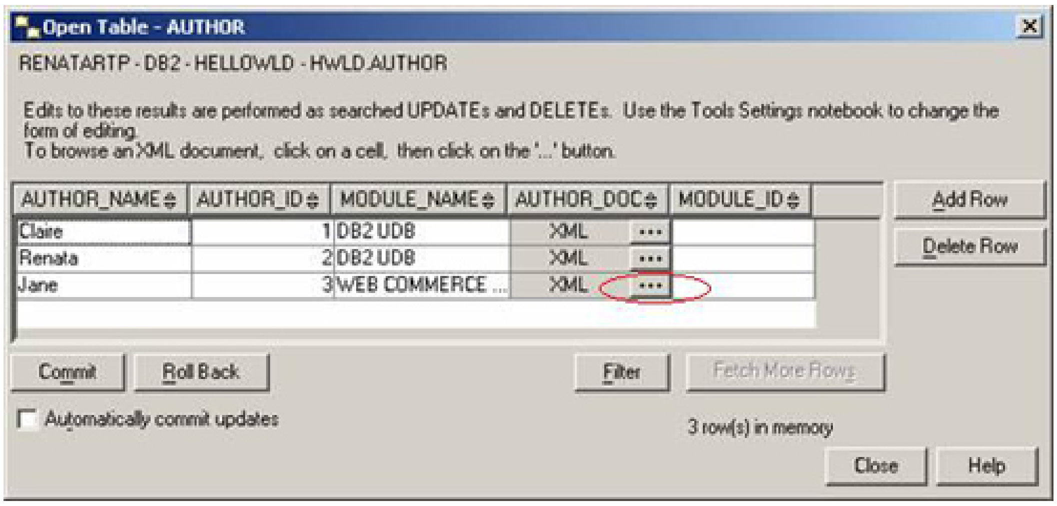
Once data is inserted into a table, it can be queried using SELECT statements or the Query operation in the Control Center. By providing predicates in the SELECT statement, users can retrieve only the information which is useful or relevant to them. You have seen examples of SELECT statements using the Command Editor in the previous exercise for the load operation. The next exercise shows you a different way to query the data.

Would you like to see these steps demonstrated for you?

Show me

1. Look at the contents of the table. Select the HWLD.AUTHOR table and click **Open**.

2. Figure 37 displays all the data, except that XML columns data. To view XML data, click the **...** button on the row you want to view in AUTHOR_DOC column.
Figure 37. View a Table



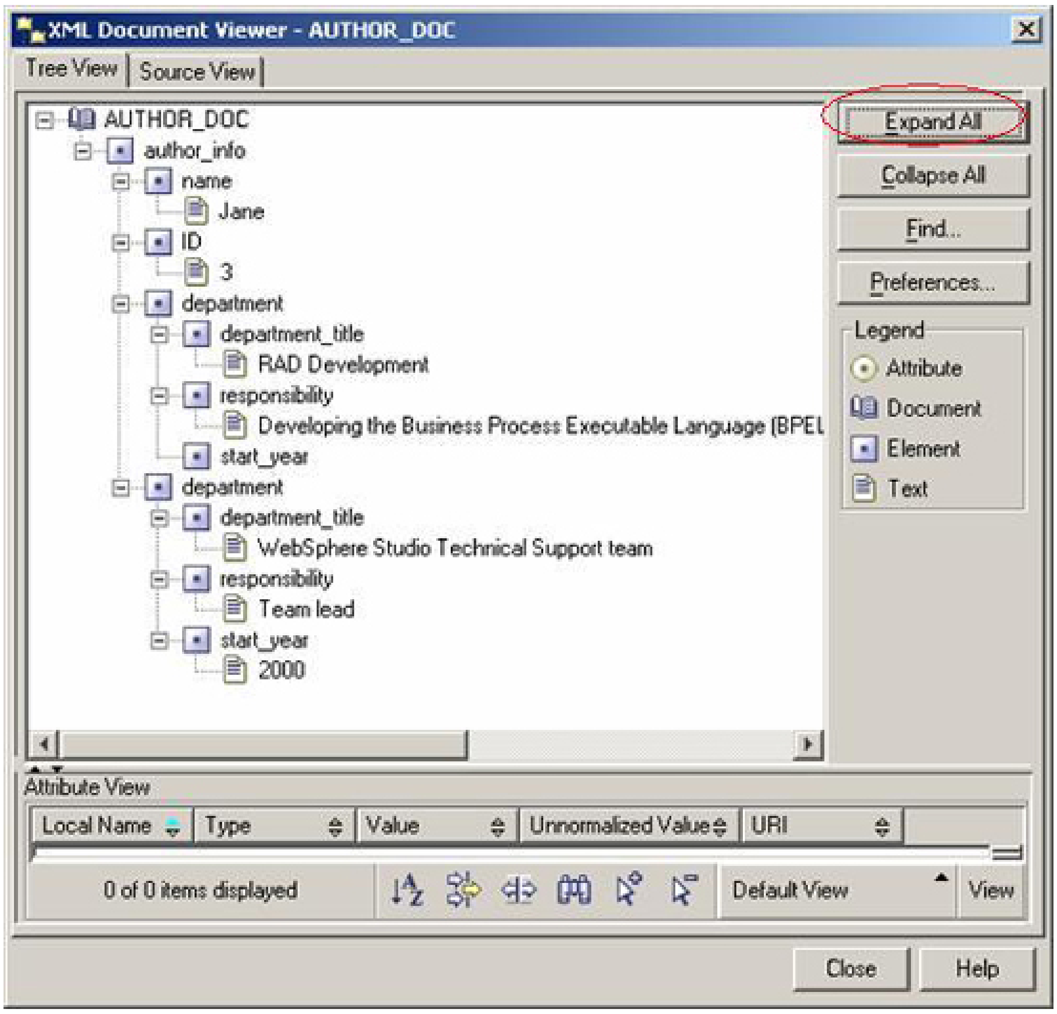
Open Table - AUTHOR

RENATARTP - DB2 - HELLOWLD - HWLD.AUTHOR

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.
To browse an XML document, click on a cell, then click on the '...' button.

AUTHOR_NAME	AUTHOR_ID	MODULE_NAME	AUTHOR_DOC	MODULE_ID
Claire	1	DB2 UDB	XML ...	
Renata	2	DB2 UDB	XML ...	
Jane	3	WEB COMMERCE ...	XML ...	

3. The following window should open:
Figure 38. View XML data in a table



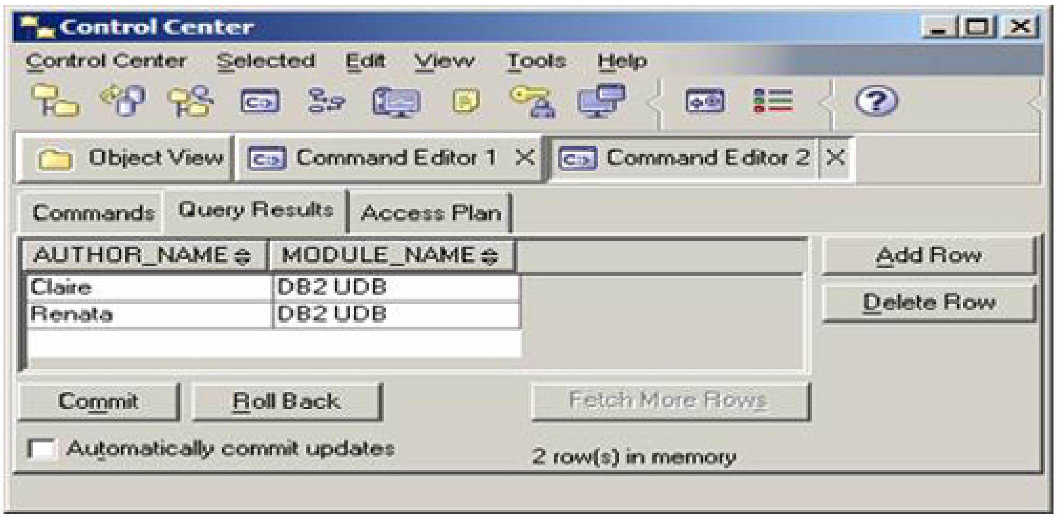
```
AUTHOR_DOC
  author_info
    name
      Jane
    ID
      3
    department
      department_title
        RAD Development
      responsibility
        Developing the Business Process Executable Language (BPEL
      start_year
    department
      department_title
        WebSphere Studio Technical Support team
      responsibility
        Team lead
      start_year
        2000
```

0 of 0 items displayed

Click the Expand All button and you see the whole structure of the XML document.

4. You can also query a view. Use the simple view created HWLD.DB2_AUTHOR. Click **Control Center > View** and select **DB2_AUTHOR**. Click **Query**.
A Command Editor is launched with the appropriate SELECT statement to query DB2_AUTHOR.

5. Click the green execution button from the Command Editor. The Query Results panel opens, where the select results from view HWLD. DB2_AUTHOR are displayed:
Figure 39. Query a view



AUTHOR_NAME	MODULE_NAME
Claire	DB2 UDB
Renata	DB2 UDB

Section 14. Running XQuery

XQuery is a language for querying and formatting XML data. The pureXML feature allows XML documents to be stored in their native hierarchical structure within the table, as opposed to being converted to text or other formats. Because XML has a non-linear and unpredictable layout, XQuery can be used to quickly parse through and find the path that matches your criteria. XQuery is based on expressions that are evaluated to return a value. There are many kinds of expressions that can be built from the XQuery keywords, symbols and operands. DB2 accepts the keyword XQUERY as an indication that it is followed by an XQuery expression. With XQuery in DB2 you can call one of two functions to obtain XML data: db2-fn:sqlquery and db2-fn:xmlcolumn. An example of each is presented in this exercise. Note that the values returned are not necessarily complete XML documents.

DB2 9 also offers Developer Workbench with XQuery builder to help you build queries against the XML columns if you are not familiar with the XQuery syntax or the structure of your XML document. See Resources to find links to information about Developer Workbench.

Run the following either from the command editor or from the command line:

Would you like to see these steps demonstrated for you?

Show me

1. Use the db2-fn:xmlcolumn function, which searches the specified path in a given XML column and returns a sequence of XML values. The argument of the function and the column name are case sensitive, and

the column name must be a qualified name of the column. In the first example, search the AUTHOR_INFO column and extract the names of the authors:
`xquery db2-fn:xmlcolumn('HWLD.AUTHOR.AUTHOR_DOC')/author_info/name`

You'll see the following:

```
1
-------------------------------
<name>Claire</name>
<name>Renata</name>
<name>Jane</name>

3 record(s) selected.
```

2. Use the db2-fn:xmlcolumn function to search AUTHOR_INFO column and extract all the information about the department.

```
xquery db2-fn:xmlcolumn('HWLD.AUTHOR.AUTHOR_DOC')/author_info/department
```

You'll see the following:

```
1
------------------------------------------------------------------------
<department>
<department_title>DB2 CE</department_title>
<responsibility>Supporting customer problems</responsibility>
<start_year>2005</start_year>
</department>
<department>
<department_title>DB2 Index Manager</department_title>
<responsibility>Develop new features for DB2 indexes</responsibility>
<start_year/>
</department>
<department>
<department_title>FVT kernel</department_title>
<responsibility>Testing the newest solutions for DB2</responsibility>
<start_year>2006</start_year>
</department>
```

```
<department>

<department_title>DB2 Load</department_title>

<responsibility>Develop new features for data movement utilities</responsibility>

<start_year>2003</start_year>

</department>

<department>

<department_title>RAD Development</department_title>

<responsibility>Developing the Business Process Executable Language (BPEL) and Business
Rules debugger in WebSphere Integration Developer</responsibility>

<start_year/>

</department>

<department>

<department_title>WebSphere Studio Technical Support team</department_title>

<responsibility>Team lead</responsibility>

<start_year>2000</start_year>

</department>

6 record(s) selected.
```

3. Use a combination XQuery that invokes SQL to retrieve the department name only of the author with ID value 1.

```
xquery db2-fn:sqlquery('select AUTHOR_DOC from HWLD.AUTHOR
 where author_id=1')/author_info/department/department_title
```

You'll see the following:

```
1
--------------------------------------------------------------

<department_title>DB2 CE</department_title>

<department_title>DB2 Index Manager</department_title>

2 record(s) selected.
```

Section 15. Conclusion

Congratulations! You have completed the seventh tutorial in the Hello, World! series, DB2 for Linux, UNIX, and Windows: Introduction to basic features and concepts. The next tutorial in the series, "DB2 for Linux, UNIX and Windows – Introduction to some advanced features and concepts.", will discuss the following:

- Database and Database Manager Configuration
- Database Concurrency
- Database Recovery
- Controlling Data and Database Access

The Information Center provides a more detailed overview of DB2 and help on specific tools and commands. See the Resources if you are interesting in learning more.

Downloads

Description	Name	Size	Download method
Files used in this tutorial	samplefiles.zip	2KB	HTTP

Information about download methods

Resources

Learn

- More tutorials in this Hello World series
- DB2 Information Center
- IBM developerWorks DB2 Zone
- IBM developerWorks DB2 basic series presents articles geared toward beginning users.
- New to DB2 for Linux, Unix, and Window is designed for people new to DB2 for Linux, UNIX, and Windows.
- DB2 9 Fundamentals certification 730 prep series
- DB2 Developer Workbench, Part 1: Developer Workbench concepts and basic tasks tutorial
- DB2 Developer Workbench, Part 2: Developer Workbench and stored procedures
- DB2 Data Movement Utilities Guide and Reference
- DB2 9 Database administration 731 certification prep series
- IBM Service Oriented Architecture
- Get documentation on XQuery builder in Developer Workbench
- Check out the Workbench resources at developerWorks

Get products and technologies

- Build your next development project with IBM trial software, available for download directly from developerWorks
- Download Developer Workbench

Discuss

- Participate in the discussion forum for this content.
- Participate in Global WebSphere® Community blogs and get involved in the community.

About the authors

Claire Hong
Claire Hong is a Software Developer with eight years experience at IBM Toronto Software Development Lab, working on the design, implementation, performance tuning and L3 advanced support of DB2 database kernel areas including index, recovery, and locking. From DB2 Version 6 to Version 9, she has delivered features

such as Async Index Cleanup, Online Index Creation, Online Index Reorg, HADR Index Support, as well as numerous index performance tuning features. Currently, Claire is working for the DB2 Continuing Engineering Team, Kernel area, and is responsible for Index Manager as well as Data Protection Service components. Her most recent interests are to transfer her DB2 knowledge to external DB2 user communities and help improve the DB2 user experience.

Renata Kupresak
Renata Kupresak is a Software Developer for DB2 currently working in the FVT Kernel team where she is responsible for Quality Assurance of different product components. Prior to that she worked on the Data Movement team where she was the component owner of Import and Export Utilities. She was also involved in introducing XML support for these utilities. Renata graduated from York University with a Bachelor in Science, Combined Honours in Computer Science and Applied Mathematics as well as Bachelor of Education in Intermediate/Senior Division.