

# DB2 9 DBA certification exam 731 prep, Part 1: Server management

Dwaine Snow

June 27, 2006

Learn skills that help you to properly manage your DB2® database servers. This is the first in a [series of seven tutorials](#) to help you prepare for the DB2 9 for Linux®, UNIX®, and Windows™ Database Administration (Exam 731).

[View more content in this series](#)

## Before you start

### About this series

If you are preparing to take the DB2 DBA certification exam 731, you've come to the right place -- a study hall, of sorts. This [series of seven DB2 certification preparation tutorials](#) covers the major concepts you'll need to know for the test. Do your homework here and ease the stress on test day.

### About this tutorial

This tutorial introduces skills you must have to properly manage a DB2 server. This is the first tutorial in a series of seven tutorials to help you prepare for the DB2 9 for Linux, UNIX, and Windows Database Administration Certification (Exam 731). The material in this tutorial primarily covers the objectives in Section 1 of the exam, Server Management. You can view these objectives at: [http://www-03.ibm.com/certify/tests/test\\_index.shtml](http://www-03.ibm.com/certify/tests/test_index.shtml).

Topics covered in this tutorial include:

- What data consistency is
- What transactions are and how they are initiated and terminated
- How transactions are isolated from each other in a multi-user environment
- How DB2 9 provides concurrency control through the use of locks
- What types of locks are available and how locks are acquired
- What factors influence locking

You should also review the [Resources](#) at the end of this tutorial for more information about DB2 server management.

## Objectives

After completing this tutorial, you should be able to:

- Understand the basic skills of managing DB2 servers, including how to create, drop, start, stop, list, migrate, and update instances.
- Use three methods of configuring DB2 client and server connectivity.
- Manage a DB2 server's access and security.
- Use the Task Center to create tasks that are coded in DB2 commands, OS commands, or MVS shell commands.
- Troubleshoot errors encountered in DB2.

## Prerequisites

To understand some of the material presented in this tutorial, you should be familiar with the following terms:

- **Object:** Anything in a database that can be created or manipulated with SQL (for example, tables, views, indexes, packages).
- **Table:** A logical structure that is used to present data as a collection of unordered rows with a fixed number of columns. Each column contains a set of values, each value of the same data type (or a subtype of the column's data type); the definitions of the columns make up the table structure, and the rows contain the actual table data.
- **Record:** The storage representation of a row in a table.
- **Field:** The storage representation of a column in a table.
- **Value:** A specific data item that can be found at each intersection of a row and column in a database table.
- **Structured Query Language (SQL):** A standardized language used to define objects and manipulate data in a relational database. (For more on SQL, see the fourth tutorial in this series.
- **DB2 optimizer:** A component of the SQL precompiler that chooses an access plan for a Data Manipulation Language (DML) SQL statement by modeling the execution cost of several alternative access plans and choosing the one with the minimal estimated cost.

To take the DB2 9 DBA exam, you must have already passed the [DB2 9 Fundamentals exam 730](#). If it's available, we recommend that you take the [DB2 Fundamentals tutorial series](#) before starting this series.

Although not all materials discussed in the Fundamentals tutorial series are required to understand the concepts described in this tutorial, you should at least have a basic knowledge of:

- DB2 products
- DB2 tools
- DB2 instances
- Databases
- Database objects

## System requirements

You do not need a copy of DB2 to complete this tutorial. However, you will get more out of the tutorial if you download the free trial version of [IBM DB2 9](#) to work along with this tutorial.

## DB2 instances

### Creating and dropping an instance

A DB2 *instance* is a logical context in which DB2 commands and functions are executed. You can think of an instance as a service or a daemon process that manages access to database files. More than one instance can be defined on a server machine. Each instance is independent of the others, meaning that all instances can be managed, manipulated, and tuned separately.

To create an instance in Windows simply issue this command:

```
db2icrt instance_name
```

In Linux and UNIX you must also provide a user ID that will be used to create fenced user-defined function and stored procedure processes, like so:

```
db2icrt -u fenced_user_ID instance_name
```

User-defined functions and stored procedures, by default, are created in fenced mode so that these processes run in a different address space than the DB2 engine, also known as the system controller process, *db2sysc*. This protects the database manager from being accidentally or maliciously damaged by any user-defined routines.

To drop an instance, disconnect all database connections and stop the instance by issuing this command:

```
db2idrop -f instance_name
```

### Listing, migrating, and updating a DB2 instance

To list DB2 instances that exist on a server, use the command:

```
db2ilist
```

Instance migration is required if you decide to move up to a newer version of the DB2 software than is installed on your server, or if an instance is to be migrated from a 32-bit to a 64-bit instance. On Windows, instance migration is done implicitly during the necessary migration process. On Linux and UNIX, use the following command to migrate an existing instance explicitly:

```
db2imigr instance_name
```

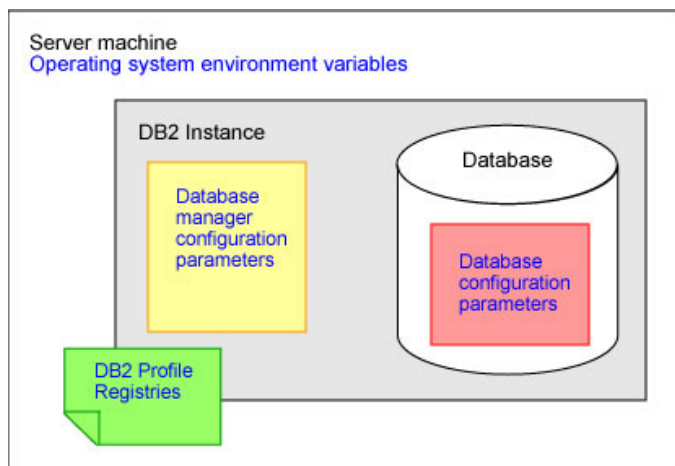
When you install fixpacks or patches to DB2, once the fixpack has been installed on the server you need to update the existing instances to link to the new fixpack files. Use the following command to update an instance:

```
db2iupdt instance_name
```

## Setting up the DB2 environment

Proper setup of the DB2 environment is very important because it controls how DB2 operates and functions. The DB2 environment is made up of:

- DB2 profile registries
- Operating system environment variables
- DB2 database manager configuration parameters
- DB2 database configuration parameters



## Setting profile registries

DB2 profile registries are DB2-specific variables that affect the management, configuration, and performance of the DB2 system. You usually need to stop and restart an instance for changes made to the DB2 profile registries to take effect.

To list all supported DB2 profile registries:

```
db2set -lr
```

To set a DB2 profile registry:

```
db2set registry_variable = value
```

Note that there are no spaces between the variable name, the equals sign, and the variable value. Here is an example setting the DB2COMM registry variable to a single value:

```
db2set DB2COMM=TCPIP
```

Here is an example setting the DB2COMM registry variable to multiple values:

```
db2set DB2COMM=TCPIP,NPIPE,LOCAL
```

To reset a DB2 profile registry to its default value, simply use the same command as above but do not specify any value:

```
db2set registry_variable =
```

To display all the DB2 profile registries currently set on the server, issue this command:

```
db2set -all
```

You will get output similar to the following:

```
[e] DB2PATH=C:\Program Files\IBM\SQLLIB_01
[i] DB2ACCOUNTNAME=IBM-TP101\dwsnow
[i] DB2INSTOWNER=IBM-SB2QTSR5RSN
[i] DB2PORTRANGE=60001:60004
[i] DB2INSTPROF=C:\PROGRA~1\IBM\SQLLIB~1
[i] DB2COMM=TCPIP,NPIPE,LOCAL
[g] DB2_EXTSECURITY=YES
[g] DB2SYSTEM=IBM-TP101
[g] DB2PATH=C:\Program Files\IBM\SQLLIB_01
[g] DB2INSTDEF=DB2V
```

Indicators surrounded by the square brackets ( `[ ]` ) represent the scope of the registry profile, as follows:

- `[e]` represents a registry setting for the current session or environment
- `[u]` represents a user-level registry
- `[n]` represents a node-level registry
- `[i]` represents an instance-level registry
- `[g]` represents a global-level registry

## Setting system environment variables

Most DB2 environment settings are controlled by the DB2 profile registry. Those that are not stored in the profile registry are called the *operating system environment variables*. Commands for setting the system variables will vary depending on the platforms and UNIX shells you are using.

Here are some examples:

- On Windows: `set DB2INSTANCE=PROD`
- In the Korn shell on Linux and UNIX: `export DB2INSTANCE=PROD`

`DB2INSTANCE` is an important system variable to know about. It specifies the current application's sessions, or window's default DB2 instance. Once this variable is set, all subsequent DB2 commands are executed within the scope of that instance.

To find out which DB2 instance you are working in, run the DB2 command:

```
get instance
```

For example, you can do this by just running:

```
db2 get instance
```

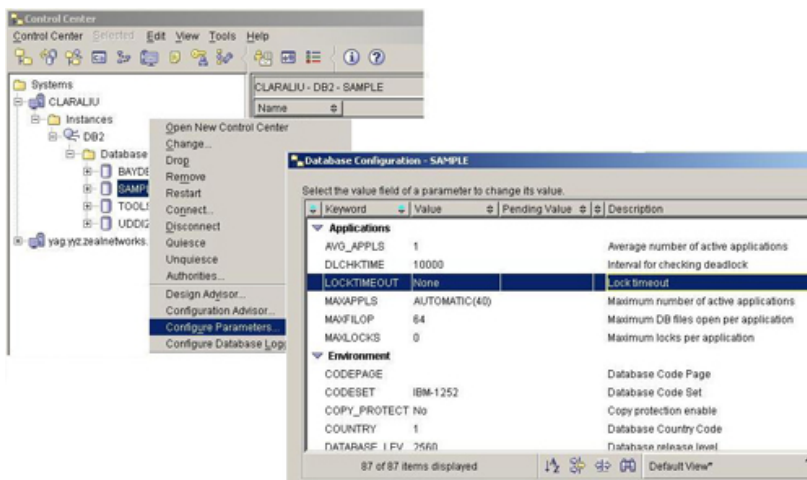
The output from this command will look something like the following:

```
The current database manager instance is: DB2V
```

## Setting configuration parameters

In DB2, there are two "levels" of configuration. At the *instance* (or database manager) level you configure the entire DB2 environment for that instance, and this impacts all databases in the instance and all applications using databases in the instance. You can configure parameters at the *database* level, which effects the behavior of all applications accessing that particular database. Refer to the [Monitoring DB2 activity](#) tutorial for more information about the parameters. The database manager, database configuration parameters, and their values can be viewed and set using either the DB2 Control Center or using DB2 commands.

At the Control Center, right click on the instance or database you would like to configure or change and select **Configure Parameters**. You will get a list of configuration parameters with short descriptions and their current and pending values, as shown below.



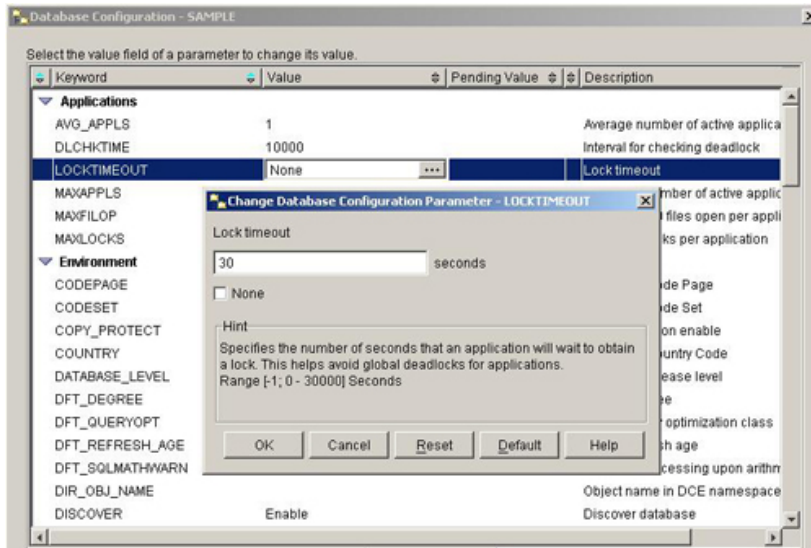
You can also get the same output by using the DB2 commands:

```
db2 get database manager configuration
db2 get database configuration for database_name
```

To update the values of the database manager or database configuration parameters at the Control Center, open the DBM or DB Configuration window. Find the parameter you want to change and double click on its value. The '...' icon will open a form that contains hints on how to set the variable, and explanations of what ranges are allowed.

If the parameter value can change immediately and dynamically, this will happen as soon as you select **OK**. Some of the less important parameters cannot change dynamically, so in this case

you will see that their current value and pending values will be different. The pending value is the new value that will be used the next time the instance or database is stopped and restarted. The **Pending Value Effective** column tells you when the new value will take effect.



The following commands can also be used to set the values of your database manager or database configuration parameters:

```
db2 update database manager configuration using parameter new_value
db2 update database configuration for database_name using parameter new_value
```

If the parameter change you specified cannot take effect immediately, a warning message like the following will be returned after you run the update db/dbm configuration command:

```
SQL1362W One or more of the parameters submitted for immediate modification
were not changed dynamically. Client changes will not be effective until the next time
the application is started or the TERMINATE command has been issued. Server changes
will not be effective until the next DB2START command.
```

## Setting configuration parameters online

Most configuration parameters can be set online while an instance or a database is still running. By default, changes to these online configurable parameters will take effect immediately where possible. For example, if the value of `sortheap` is changed, all new SQL requests will use the new value. To specify this immediate behavior explicitly, append the `immediate` keyword to the `update` command:

```
db2 update database manager configuration using parameter new_value immediate
db2 update database configuration for database_name using
parameter new_value immediate
```

If you choose to defer the changes until the instance is restarted or until the database is activated, specify the `deferred` keyword instead:

```
db2 update database manager configuration using parameter new_value deferred
db2 update database configuration for database_name using
parameter new_value deferred
```

You'll sometimes want to find out what changes have been made and deferred. To show the current and pending values of the database manager configuration parameters, attach to the instance first, then specify the `show detail` option in the `get database manager configuration` command, like so (note that `instance_name` is the value set by the system environment variable `DB2INSTANCE`):

```
db2 attach to instance_name
db2 get database manager configuration show detail
```

Similarly, to list the current and pending values of the database configuration parameters, connect to the database first and then use the `show detail` option:

```
db2 connect to database_name
db2 get database configuration for database_name show detail
```

Pending values are listed under the **Delayed Value** column, as shown below.

```
C:\>db2 connect to sample

Database Connection Information
Database server      = DB2/NT 8.1.0
SQL authorization ID = CLARALIU
Local database alias = SAMPLE

C:\>db2 get database configuration show detail

Database Configuration for Database

Description              Parameter      Current Value      Delayed Value
-----
Database configuration release level  = 8x8a88
Database release level    = 8x8a88
Database territory        = US
Database code page        = 1252
Database code set         = IBM-1252
Database country/region code = 1
Dynamic SQL Query management <DYN_QUERY_MGMT> = DISABLE      DISABLE
Discovery support for this database <DISCOVER_DB> = ENABLE       ENABLE
Default query optimization class <DFT_QUERYOPT> = 5             5
Degree of parallelism <DFT_DEGREE> = 1             1
Continue upon arithmetic exceptions <DFT_SQLMATHWARN> = NO          NO
Default refresh age <DFT_REFRESH_AGE> = 0             0
Number of frequent values retained <NUM_FREQUENTVALUES> = 10          10
Number of quantiles retained <NUM_QUANTILES> = 20          20
Backup pending            = NO
```

## Forcing an instance and a database

If you need to make a database or database manager configuration change take effect immediately, and the parameter you changed is not dynamic, you will need to stop and restart the database or the whole instance. If there are applications connected to and using the database, or databases in the instance, you cannot stop and restart the database or instance. In this case, you can *kick the users off* of the database using the DB2 command:

```
force application all
```

Or, you can stop the instance and kick off all users at the same time using the command:

```
db2stop force
```



If you only want to force off a specific application, you will need to know the application's *handle*. To find the handle use the command:

```
list applications
```

You will get output similar to the following:

| Auth Id | Application Name | Appl. Handle | Application Id          | DB Name | # of Agents |
|---------|------------------|--------------|-------------------------|---------|-------------|
| -----   | -----            | -----        | -----                   | -----   | -----       |
| DSNOW   | db2bp.exe        | 8            | *LOCAL.DB2.020205193946 | SAMPLE  | 1           |

To force off only the command line processor (or command window), in this case the db2bp.exe application, use the DB2 command:

```
force application (8)
```

## DB2 client/server connectivity

### DB2 client/server environment

Due to changes in the overall use of communication protocols among DB2 users, DB2 currently supports the following protocols for DB2 client/server connectivity:

- TCP/IP
- NPIPE

DB2 Connect, which uses Distributed Relational Database Architecture (DRDA), is required for connections to host databases such as DB2 for z/OS and/or DB2 for iSeries.

### Preparing your DB2 server for remote application connections

Before DB2 clients (applications) can connect to a database, you must ensure that the server-side communications are set up properly to accept the connection requests. To prepare a server for TCP/IP connections you need to set up a TCP/IP listener, as follows.

1. Set the DB2 profile registry, DB2COMM, to enable the instance to listen for connections from TCP/IP using the command:

```
db2set DB2COMM=TCPIP
```

2. Set the required information for the TCP/IP protocol in the database manager configuration file.

You need to assign a port number to each DB2 instance that is TCP/IP connection enabled. A file called "services" contains entries for every service defined on the system and their associated port numbers. The location of the file will depend on your operating system. For example, on Linux and UNIX it is usually stored in the directory /etc.

Since a port number can be used by only a single service at a time, it is highly recommended that you use the services file as a central place to maintain a list of all services (and DB2

instances) and their associated port numbers. For example, to reserve TCP/IP port number 50000 for the DB2 instance *db2icdb2*, add the following line to the `services` file:

```
db2icdb2      50000/tcp
```

Update the database manager configuration so that DB2 will use the port number associated with the service *db2icdb2* for the instance you are working on:

```
db2 update database manager configuration using svcename db2icdb2
```

You can also code the port number directly in the database manager configuration, rather than adding the port number to the `services` file. In this case, update the database manager configuration parameter `svcename` with the correct port number, as follows:

```
db2 update database manager configuration using svcename 50000
```

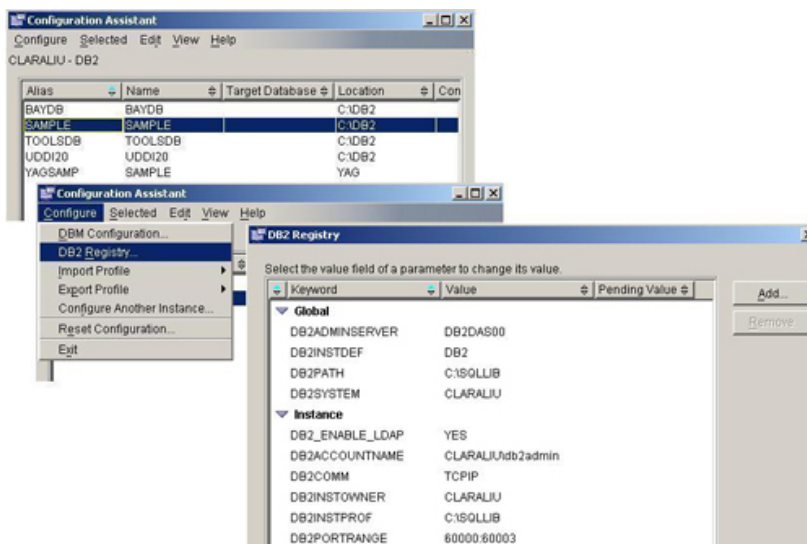
3. The database manager configuration parameter `svcename` is not dynamic, so you must stop and restart the instance so that the TCP/IP listener is started, as follows:

```
db2stop
db2start
```

## Using the DB2 Configuration Assistant

The DB2 Configuration Assistant provides user-friendly wizards and a graphical interface for configuring the environment you or your applications will be using. From the Configuration Assistant, shown below, you can:

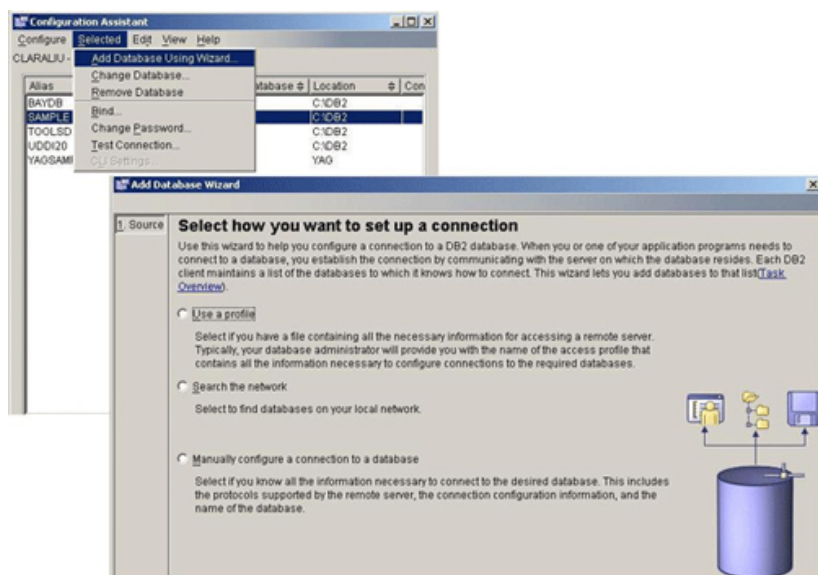
- Add new database connections
- Update database connectivity information
- View and update database manager configuration parameters
- View and update DB2 profile registries
- Bind applications to a database
- Update the Call Level Interface (CLI) settings



## Three ways to configure database connectivity

There are three options available in the DB2 Configuration Assistant for setting up a database connection. You can:

- Search the network for DB2 databases
- Use DB2 access profiles
- Configure the connection manually

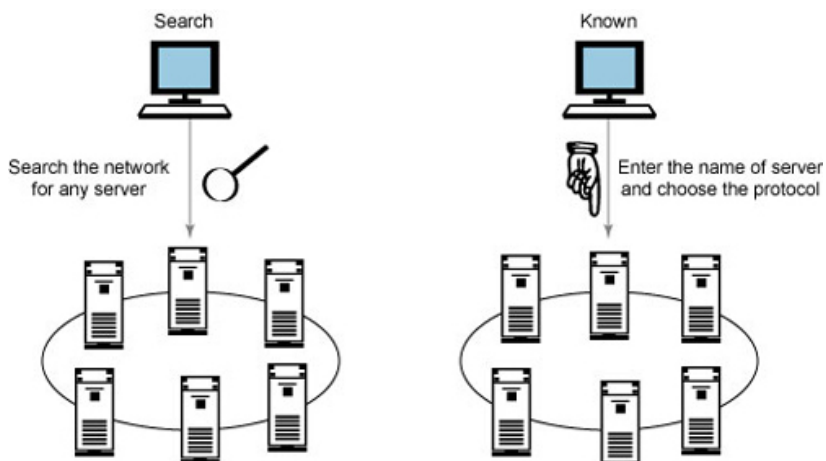


## Configuring database connectivity automatically by searching the network

DB2 Discovery searches and locates DB2 servers on your network. You can choose to use either the *search* or *known* discovery method.

The search method searches the network for any DB2 servers. This method may take some time to return a result.

If you know some information about the DB2 server you want to locate, you can use the known method and provide information such as the database or server name to limit the search.



Sometimes you might not want certain DB2 servers, instances, or databases to be *discoverable*. For example, imagine a DB2 server containing a production instance and a development instance. In the development instance, two databases, ACCT and HUMRES, are defined. You might want to keep the production instance from being discovered and allow only the ACCT database in the development instance to be discovered. DB2 allows you to configure this so that you do not make confidential databases easily available on your servers.

A DB2 server is searchable only if the Administration Server (DAS) service is running on that server, and the `discover` configuration parameter is set to `search`, as follows:

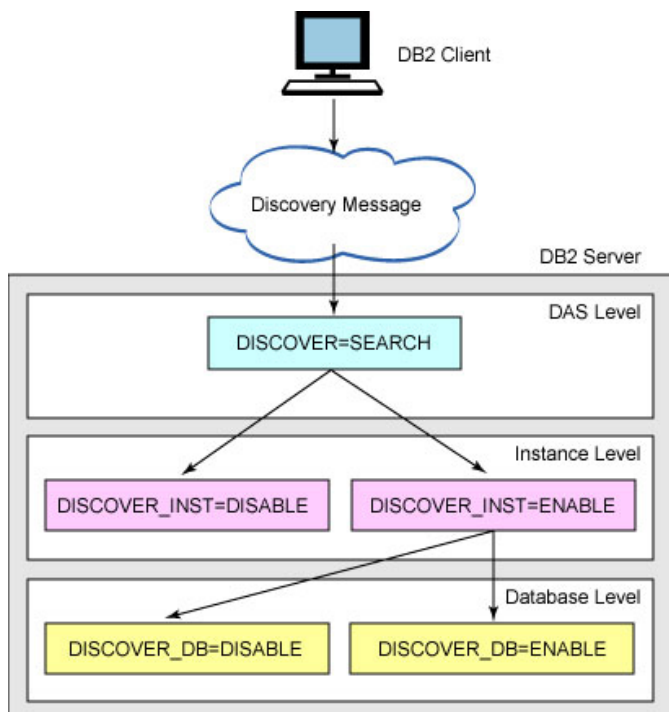
```
db2admin start
db2 update admin configuration using discover search
```

You can also control which instances are discoverable by setting the `discover_inst` database manager configuration parameter as follows:

```
db2 update database manager configuration using discover_inst enable
```

Each database has a similar configuration parameter, `discover_db`, that can enable or disable database discovery, as follows:

```
db2 update database configuration for database_name using discover_db enable
```



It is important to point out that disabling discovery at the DAS, instance, or database level does not restrict DB2 clients from setting up database connectivity through other methods (which will be discussed next). DB2 clients can still connect to a remote database even though its database configuration `discover_db` is disabled.

## Configuring database connectivity automatically with DB2 access profiles

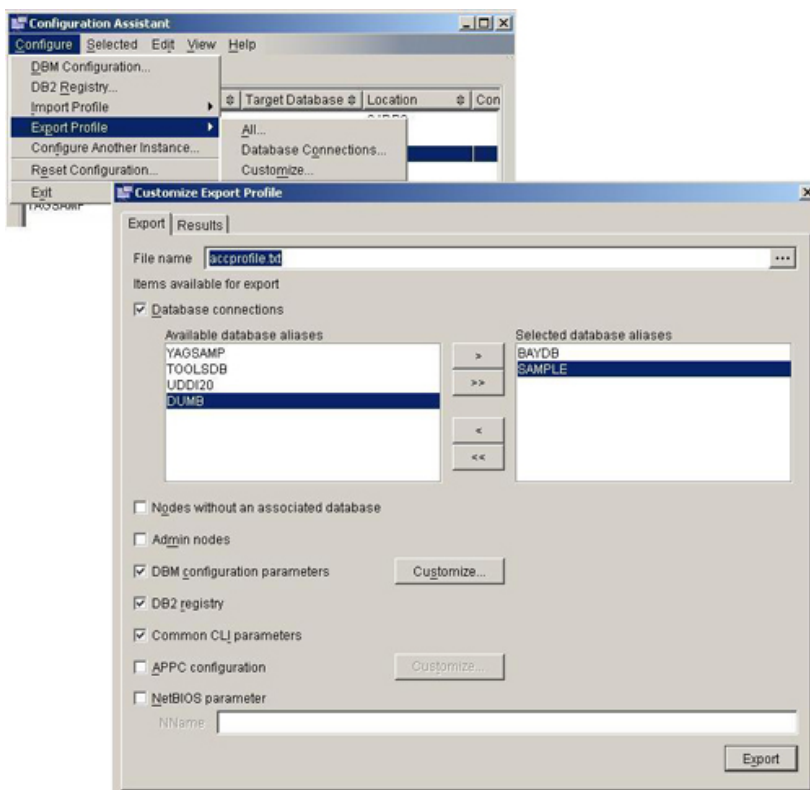
What would you do if you needed to set up DB2 client/server connectivity for 1,000 or more workstations? You could go to each workstation and use the discovery method from the Configuration Assistant, but that task would probably take you a long time to complete. In such a situation, you should consider using a *DB2 access profile*.

An access profile contains information that a client needs for configuring connectivity with a DB2 server. There are two types of access profiles:

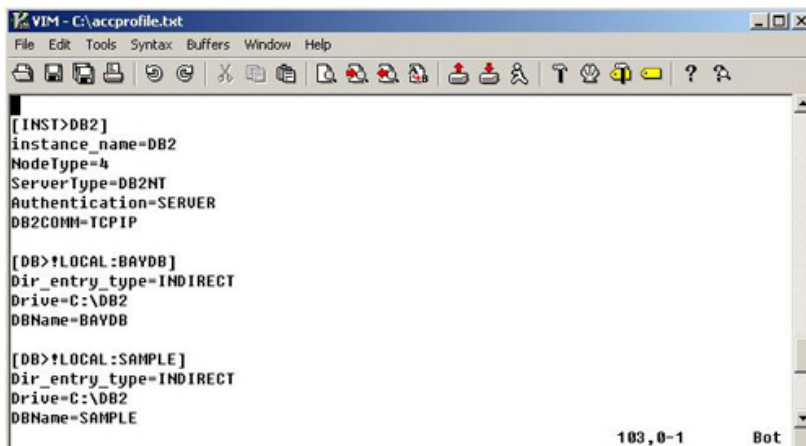
- A *server access profile* is generated on a DB2 server. It contains information about all or selective instances and databases that are defined on the server.
- A *client access profile* is generated on a DB2 client. It contains information about instances (also called *nodes*) and databases already cataloged on the client.

Let's walk through the DB2 access profile approach step by step.

1. Use the Configuration Assistant to export information to an access profile (which is just an ASCII file). In the figure below, notice that some DB2 environment settings, such as database manager configuration and DB2 profile registries, can also be exported.



2. Send the exported file to the client.

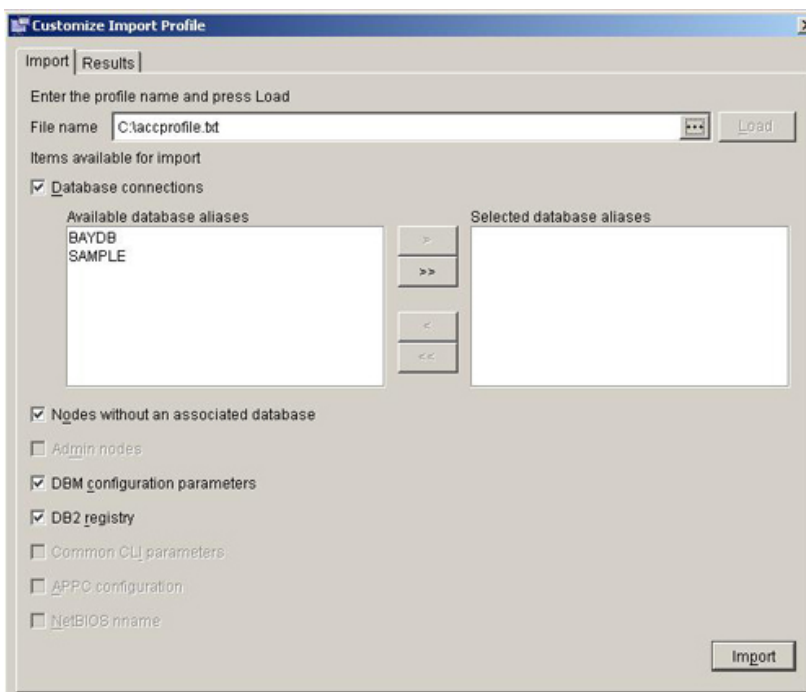


```
[INST>DB2]
instance_name=DB2
NodeType=4
ServerType=DB2NT
Authentication=SERVER
DB2COMM=TCPIP

[DB>!LOCAL:BAYDB]
Dir_entry_type=INDIRECT
Drive=C:\DB2
DBName=BAYDB

[DB>!LOCAL:SAMPLE]
Dir_entry_type=INDIRECT
Drive=C:\DB2
DBName=SAMPLE
```

3. Use the Configuration Assistant to import the file into the DB2 client.

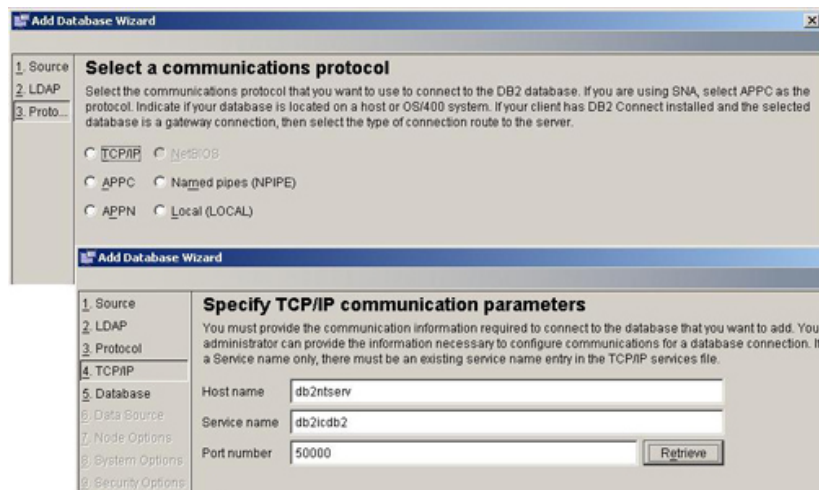


You may find it burdensome to explain to all your users how to import the profile using the Configuration Assistant. And, some users may have a version of the DB2 runtime client installed that does not include Configuration Assistant. In such cases, you can use the following command to perform the same import as described above:

```
db2cfimp access_profile_name
```

## Configuring database connectivity manually

If you know all of the information required to configure your connectivity, you can use the Add Database wizard from the Configuration Assistant, as shown below.



Alternatively, you can use the `catalog` commands with the DB2 Command Line Processor (CLP) or a DB2 Command Window, as follows.

1. You must first catalog the node, or DB2 server, and instance combination.  
Each instance you want to attach to needs to be cataloged as a node. Use the `catalog` command with varying keywords for each supported communication protocol. Some examples:

```
db2 catalog tcpip node mynode remote db2server.mycompany.com server db2icdb
db2 catalog netbios node jeremy remote N01FCBE3 adapter 0
```

2. Catalog the database.  
Catalog one or more databases that belong to the cataloged instance. Some examples:

```
db2 catalog database sample as mysamp at node mynode
db2 catalog database baydb as newbaydb at node mynode
```

## Listing node and database directories

The information for nodes (DB2 servers) and databases that you successfully catalogued are stored in the DB2 NODE directory and in the DATABASE directory. These provide an abstract mapping of the DB2 instances and databases used by the clients.

To list the server and instance combinations in the node directory, use the command:

```
db2 list node directory
```

You should see output similar to the following:





To list the database directory, use the command:

```
db2 list database directory
```

You should see output similar to the following:

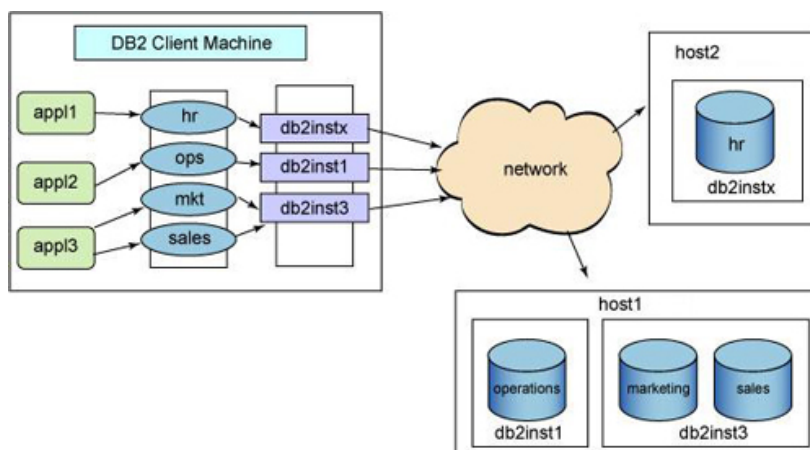
```
DB2 CLP
C:\>db2 list database directory

System Database Directory
Number of entries in the directory = 6
Database 1 entry:
Database alias           = VAGSAMP
Database name            = SAMPLE
Node name                = VAG
Database release level   = a.00
Comment                  =
Directory entry type     = Remote
Catalog database partition number = -1
Database 2 entry:
Database alias           = TOOLSDB
Database name            = TOOLSDB
Database drive           = C:\DB2
Database release level   = a.00
Comment                  =
Directory entry type     = Indirect
Catalog database partition number = 0
```

Let's use an example to get a complete picture of the DB2 client/server environment and how to catalog the DB2 node and database.

In the diagram below, the first DB2 server (host1) has two DB2 instances and three databases defined on it. The second DB2 server (host2) contains only one instance and one database.

For a client machine to connect to all four databases in this scenario, each remote instance must be cataloged and stored in the client's node directory, and each database must be catalogued on its respective node (DB2 server and instance).



## Attaching to an instance and connecting to a database

Once you've set up the client/server connectivity by cataloging the nodes and databases, you can either attach to an instance to do instance level administration tasks, or connect to a database to read or write data from and to the database.



To attach to a DB2 instance use the DB2 command `attach`, as follows:

```
attach to nodename user username using password
```

After attaching to an instance you can perform administrative tasks on the instance, such as:

- Creating and dropping databases
- Retrieving, updating, and resetting database manager and database configuration parameters
- Managing database monitors
- Backing up, restoring, and rolling forward a database
- Forcing users and applications off the databases defined in the instance

To connect to a database to be able to select, insert, update or delete data, you must first connect to the database as follows:

```
connect to database_name user username using password  
[new new_password confirm new_password ]
```

Notice that the `connect` statement also allows you to specify a new password for the specified user if you wish.

Once you are connected to a database you can perform Data Manipulation Language (DML) operations, such as:

- SELECT
- INSERT
- UPDATE
- DELETE

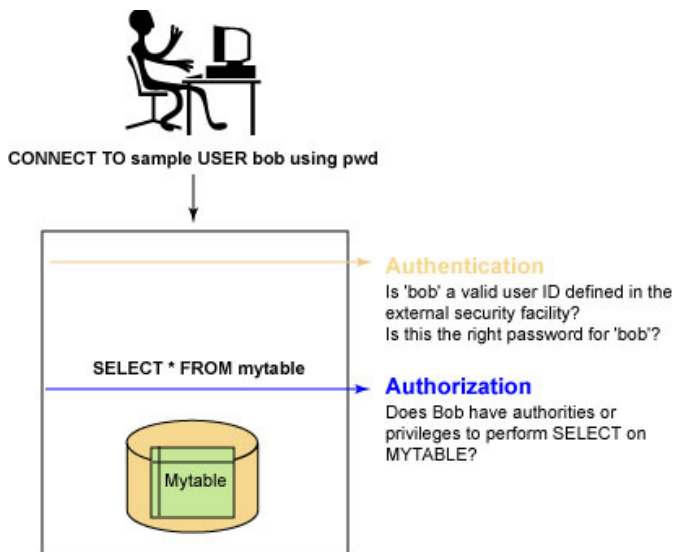
You can also perform:

- Database definition language (DDL) to `CREATE` or `ALTER` database objects.
- Database control language (DCL) to `GRANT` or `REVOKE` object privileges.
- Precompile and bind package operations to the database.
- Move data into or out of the database using the `EXPORT`, `IMPORT`, and `LOAD` utilities.

## DB2 security

### DB2 security overview

DB2 security is handled by a combination of external security services and an internal DB2 authorization mechanism. The external security service authenticates users who want to access a DB2 server; security software outside of DB2 takes care of the authentication. This software can be a security facility of the operating system, or a separate product such as Kerberos. Once the user ID and password have been successfully verified, an internal DB2 process takes over and makes sure that the user is authorized to perform the requested operations.



## Authentication types

The authentication type determines where the user ID/password pair is verified. The supported authentication types are:

- SERVER (default)
- SERVER\_ENCRYPT
- KERBEROS
- KRB\_SERVER\_ENCRYPT
- CLIENT

Authentication types are set at both the server and client.

### At server

Only one authentication type is allowed per instance, meaning that the setting applies to all databases defined under that instance. Specify it in the database manager configuration file with the `AUTHENTICATION` parameter.

```
db2 update database manager configuration authentication auth_type
```

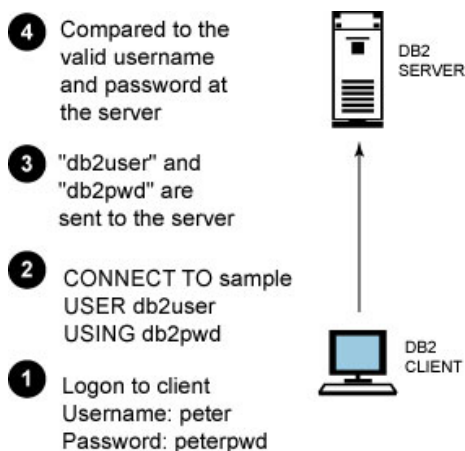
### At client

Each database cataloged at the client has its own authentication type specified with the `CATALOG DATABASE` command.

```
db2 catalog database db_name at node node_name authentication auth_type
```

## Authenticating with the SERVER option

With the `SERVER` option, the user ID and password are sent to the server for validation. Consider the example below.

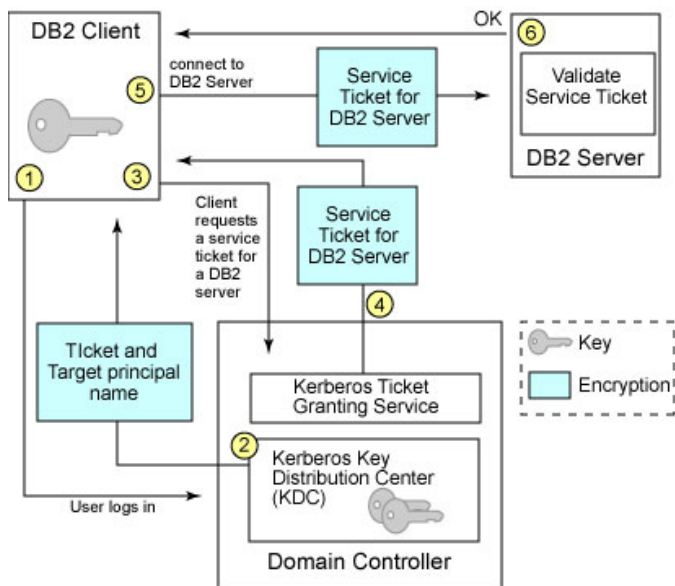


1. A user logs into a workstation with the username *peter* and password *peterpwd*.
2. *peter* then connects to the SAMPLE database with the user ID *db2user* and password *db2pwd*, which are defined at the remote DB2 server.
3. *db2user* and *db2pwd* are sent to the server through the network.
4. *db2user* and *db2pwd* are validated at the DB2 server.

If you want to protect the user ID and password from eavesdropping, use authentication type `SERVER_ENCRYPT` so that both user ID and password are encrypted.

## Authenticating with Kerberos

Kerberos is an external security facility that uses conventional cryptography to create a shared encrypted key. It offers a secured authentication mechanism because the user ID and password no longer need to be transferred across the network in clear text. By using the encrypted key, it also makes single sign-on to a remote DB2 server possible. The diagram below shows how Kerberos authentication works with DB2.

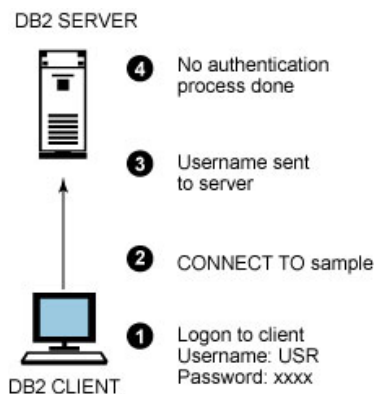


Authentication type `KERBEROS` can be used when both the DB2 client and server support the Kerberos security protocol. However, some clients may not support Kerberos, but still need to access the DB2 server. To ensure that both types of clients are able to connect securely, set the authentication type at the DB2 server as `KRB_SERVER_ENCRYPT`. This allows all Kerberos-enabled clients to authenticate with Kerberos, while other clients use `SERVER_ENCRYPT` authentication instead. Below is a summary of different client and server authentication settings related to Kerberos.

| Client Specification | Server Specification | Client/Server Resolution |
|----------------------|----------------------|--------------------------|
| KERBEROS             | KRB_SERVER_ENCRYPT   | KERBEROS                 |
| Any other setting    | KRB_SERVER_ENCRYPT   | SERVER_ENCRYPT           |

## Authenticating on the client

This option allows authentication to occur at the client machines. When a user successfully logs in to the client machine, a connection can be made to the database without challenging the user for a password.



It is important to understand that there are client systems that do not have a reliable security facility, such as Windows 9x and Classic Mac OS. They are called *untrusted clients*. Anyone who has access to these systems can also connect to the DB2 server without any authentication. Who knows what kind of destructive operations they will perform (for example, dropping a database)? To provide the flexibility of allowing trusted clients to perform authentication on their own and, at the same time, forcing untrusted clients to be authenticated at the server, two other database manager configuration parameters are introduced:

- `TRUST_ALLCLNTS`
- `TRUST_CLNTAUTH`

These two parameters will be evaluated only when authentication is set to `CLIENT`.

## Trusting clients

`TRUST_ALLCLNTS` determines which types of clients are trusted. The parameter has the following possible values:

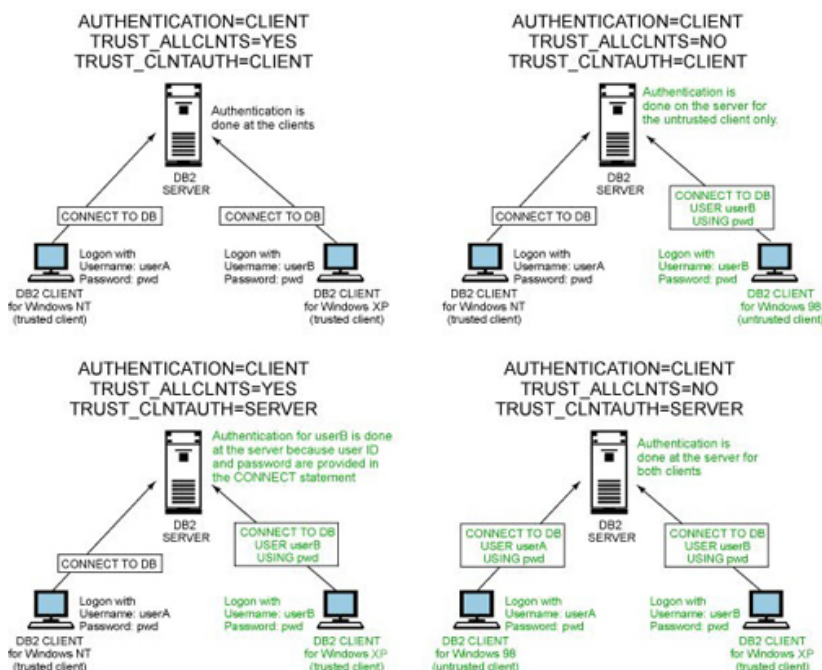
- **YES** -- Trust all clients. This is the default setting. Authentication will take place at the client. There is an exception, which we'll discuss in more detail about `TRUST_CLNTAUTH` below.
- **NO** -- Trust only clients with reliable security facilities (trusted clients). For untrusted clients to connect, user ID and password must be provided for authentication to take place at the server.
- **DRDAONLY** -- Trust only clients that are running on iSeries or zSeries platforms (for example, DRDA clients). Any other clients must provide user ID and password.

Consider a scenario in which a DB2 server has set authentication to `CLIENT` and `TRUST_ALLCLNTS` to **YES**. You log into a Windows 2000 machine as *localuser* and connect to the remote database without specifying a user ID and password. *localuser* will be the connected authorization ID at the database. What if you want to connect to the database with a different user ID -- as *poweruser*, who has the authority to perform a database backup, for instance?

To allow such behavior, use `TRUST_CLNTAUTH` to specify where authentication will take place if a user ID and password are supplied in a `connect` statement or `attach` command. Two values are allowed:

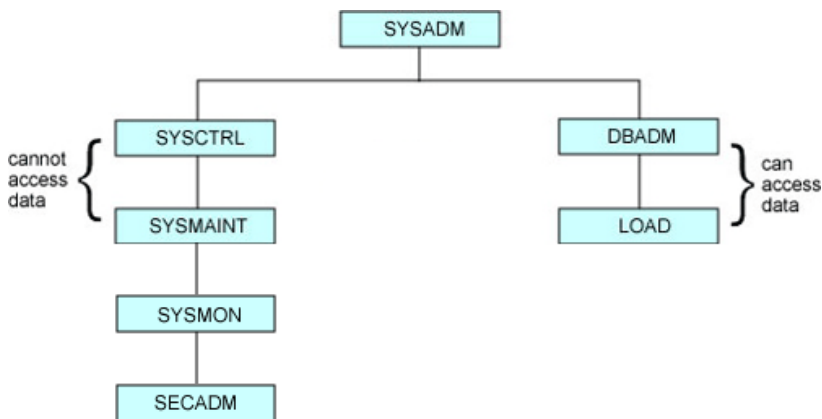
- **CLIENT** -- Authentication is performed at client; user ID and password are not required.
- **SERVER** -- Authentication is done at the server when user ID and password are supplied.

Let's look at some examples to illustrate usage of the parameters:



## Setting authority levels

Authority levels control the ability to perform database manager maintenance operations and manage database objects. There are seven authorities in DB2:



### **SYSADM**

Has full privileges for managing the instance and also has access to data in the underlying databases.

### **SYSCTRL and SYSMAINT**

Have certain privileges in managing the instance, its databases, and database objects. These authorities do *not* have access to the data. For example, statements such as `'SELECT * FROM mytable'` or `'DELETE FROM mytable'` are not allowed.

### **DBADM**

Has privileges to perform administrative tasks on the specified database. It also has full data access to the database.

### **LOAD**

Has privileges to run the load utility against the specified database.

### **SYSMON**

Has the privilege to reset monitor switches and capture database monitor snapshots.

### **SECADM**

Has the privilege to transfer the control ownership of virtually any object in the database from one user to another, grant or revoke set session user privilege, and create, drop, grant and revoke labels and permissions on labels.

The following table summarizes the functions that each authority can perform.

| Function                         | SYSADM | SYSCTRL | SYSMAINT | DBADM | LOAD | SYSMON | SECADM |
|----------------------------------|--------|---------|----------|-------|------|--------|--------|
| Update DBM CFG                   | YES    |         |          |       |      |        |        |
| Grant / Revoke DBADM             | YES    |         |          |       |      |        |        |
| Establish / Change SYSCTRL       | YES    |         |          |       |      |        |        |
| Establish / Change SYSMAINT      | YES    |         |          |       |      |        |        |
| Establish / Change SECADM        | YES    |         |          |       |      |        |        |
| Force users                      | YES    | YES     |          |       |      |        |        |
| Create / Drop databases          | YES    | YES     |          |       |      |        |        |
| Restor to a new database         | YES    | YES     |          |       |      |        |        |
| Update DB CFG                    | YES    | YES     | YES      |       |      |        |        |
| Backup database / table space    | YES    | YES     | YES      |       |      |        |        |
| Restore to an existing database  | YES    | YES     | YES      |       |      |        |        |
| Start / Stop an Instance         | YES    | YES     | YES      |       |      |        |        |
| Restore a table space            | YES    | YES     | YES      |       |      |        |        |
| Run Trace                        | YES    | YES     | YES      |       |      |        |        |
| Obtain Monitor Snapshots         | YES    | YES     | YES      | YES   |      | YES    |        |
| Query Table Space                | YES    | YES     | YES      | YES   |      |        |        |
| Prune History files              | YES    | YES     | YES      | YES   |      |        |        |
| Quiesce Table Space              | YES    | YES     | YES      | YES   |      |        |        |
| Load tables                      | YES    |         |          | YES   | YES  |        |        |
| Set / unset check pending state  | YES    |         |          | YES   |      |        |        |
| Create / Drop Event Monitors     | YES    |         |          |       |      |        |        |
| Transfer Ownership of an Object  |        |         |          |       |      |        | YES    |
| Grant / Revoke SETSESSIONUSER    |        |         |          |       |      |        | YES    |
| Create / Drop labels             |        |         |          |       |      |        | YES    |
| Grant / Revoke label permissions |        |         |          |       |      |        | YES    |

## Managing DB2 authorities

The SYS\* authorities are set in the database manager configuration by assigning a user group defined in the operating system or security facility to the associated parameters. It must be a group name with a maximum length of 8 characters, as shown below.

### Database Manager Configuration

SYSADM group name (SYSADM\_GROUP) = ADM1  
 SYSCTRL group name (SYSCTRL\_GROUP) = CTRL1  
 SYSMAINT group name (SYSMAINT\_GROUP) = MAINT1

db2 update dbm cfg using sysadm\_group adm1  
 db2 update dbm cfg using sysctrl1\_group ctr11  
 db2 update dbm cfg using sysmaint\_group maint1

DBADM and LOAD are database-level authorities. They are granted to a user or a group of users with a `grant` statement and revoked with a `revoke` statement:

```
connect to sample;
grant dbadm on database to user john;
grant load on database to group dbagrp;
revoke load on database from group dbagrp;
```

Note that users with LOAD authority also require INSERT privilege on the table before data can be loaded. The next section discusses privileges.

## Setting privileges

Privileges give users the right to access database objects in a specific way. The following lists offer a summary of privileges for different database objects.

### Database privileges:

- CONNECT allows users to connect the database.
- BINDADD allows users to create new packages in the database.
- CREATETAB allows users to create new tables in the database.
- CREATE\_NOT\_FENCED allows users to create non-fenced user-defined functions or stored procedures.
- IMPLICIT\_SCHEMA allows users to create objects in a schema that do not already exist.
- QUIESCE\_CONNECT allows users to access the database while it is quiesced.
- CREATE\_EXTERNAL\_ROUTINE allows users to create stored procedures written in C, the Java™ language, OLE, and COBOL.

### Schema privileges:

- CREATEIN allows users to create objects within the schema.
- ALTERIN allows users to alter objects within the schema.
- DROPIN allows users to drop objects within the schema.

To explicitly create a new schema, use the `create schema` command:

```
connect to sample user dbowner;  
create schema dev authorization devuser;
```

### Table space privileges:

- USE OF TABLESPACE allows users to create tables within the specified table space. This privilege cannot be used on the SYSCATSPACE or any system temporary table spaces.

### Table and view privileges:

- CONTROL provides the user with all privileges for a table or view, and the ability to grant those privileges (except CONTROL) to others.
- ALTER allows users to alter a table or view.
- DELETE allows users to delete records from a table or view.
- INDEX allows users to create indexes on a table.
- INSERT allows users to insert an entry into a table or view.
- REFERENCES allows users to create and drop a foreign key, specifying the table as the parent in a relationship.
- SELECT allows users to retrieve rows from a table or view.
- UPDATE allows users to update entries in a table or view. This privilege can also limit users in updating specific columns only: `grant update (workdept, job) on table employee to devuser;`



- ALL PRIVILEGES grants all the above privileges except CONTROL on a table or view.

### Package privileges:

- CONTROL provides users the ability to rebind, drop, or execute a package, and the ability to grant these privileges (except CONTROL) to others.
- BIND allows users to rebind an existing package.
- EXECUTE allows users to execute a package.

### Index privileges:

- CONTROL allows users to drop an index.

### Routine privileges:

- EXECUTE allows users to execute the user-defined function.

### Sequence privileges:

- USAGE allows users to use NEXTVAL and PREVVAL expressions for a sequence object.

## Granting explicit privileges

Granting a privilege `with grant option` allows the authorization ID to extend the specified privilege to others. This option is only available to package, routine, schema, table, table space, and view.

Although the grant privilege is extended, the revoke privilege is not. If privileges are received through the `with grant option`, a user will not be able to revoke the privileges from others. Here are some examples.

This statement allows *john* to perform `select`, `update`, or `delete` operations on the table *employee* and to grant any of these privileges to others:

```
grant select, update, delete on table employee to user john with grant option
```

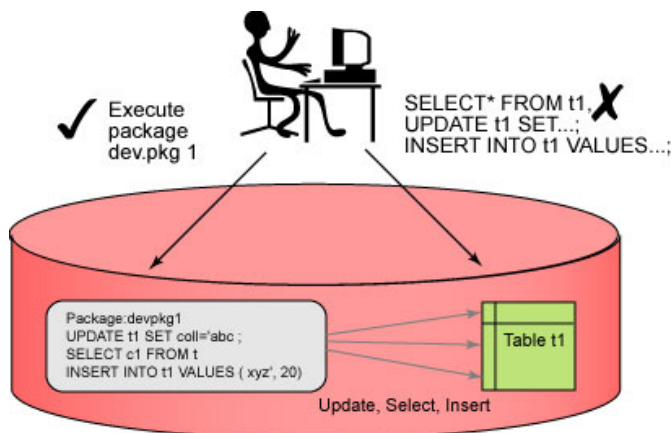
This statement allows users in the *devusers* group to rebind, drop, and execute the package *dev.pkg1*. The same group of users can also grant BIND and EXECUTE (but not CONTROL) privileges to others.

```
grant control on package dev.pkg1 to group devusers with grant option
```

## Granting implicit and indirect privileges

Typically, DB2 privileges are granted explicitly with `grant` statements as discussed previously. Sometimes users may also obtain privileges implicitly or indirectly from certain operations performed. Let's look at some scenarios.

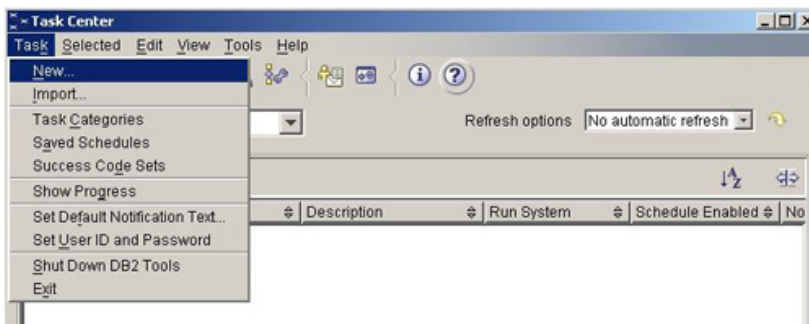
- A user granted DBADM authority is also implicitly granted BINDADD, CONNECT, CREATETAB, CREATE\_NOT\_FENCED, and IMPLICIT\_SCHEMA.
- When a user creates a database:
  - DBADM authority is granted to the database creator.
  - CONNECT, CREATETAB, BINDADD, and IMPLICIT\_SCHEMA privileges are granted to PUBLIC.
  - USE OF TABLESPACE privilege on the table space USERSPACE1 is granted to PUBLIC.
  - BIND and EXECUTE privileges on each successfully bound utility are granted to PUBLIC.
  - EXECUTE privileges with grant option on all functions in the SYSFUN schema are granted to PUBLIC.
- A user who creates a table, a view, an index, a schema, or a package automatically receives CONTROL privilege on the database object he or she creates.
- When a user executes a package that contains static SQL statements, explicit privileges for database objects referenced in the statements are not required. The user only needs EXECUTE privilege on the package to execute the statements. However, this does not mean that the user has direct access to the underlying database objects. Consider the following example:



## Scheduling jobs

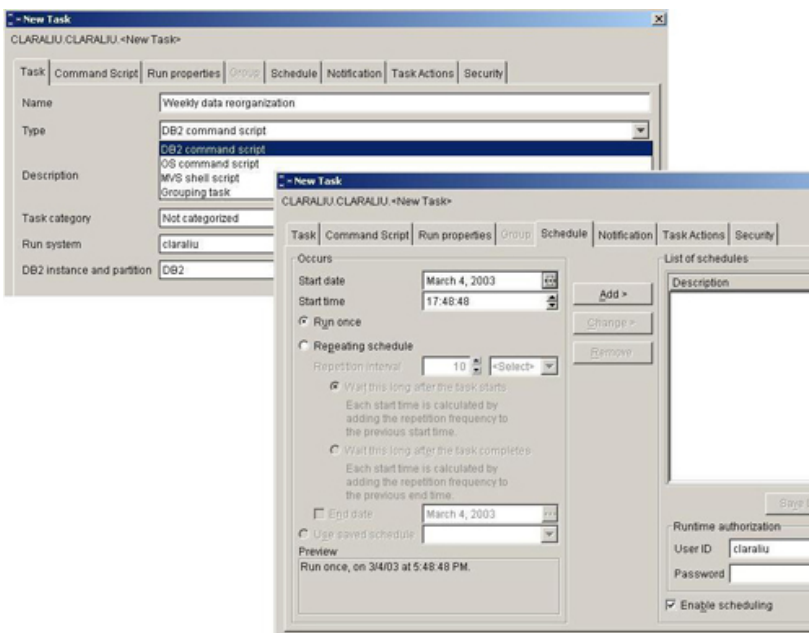
### Using the DB2 Task Center

Managing a DB2 server does not just involve initial implementation of the instance and database; it also entails performing regular maintenance tasks such as REORG and RUNSTATS, as well as loading or unloading data as needed. DB2 has an integrated set of graphical tools to help administrators implement, manipulate, and maintain DB2 instances and databases efficiently. The DB2 Task Center provides an easy-to-use graphical interface for creating and organizing tasks, managing the task flow, scheduling tasks, and distributing notifications about the status of tasks that have run.



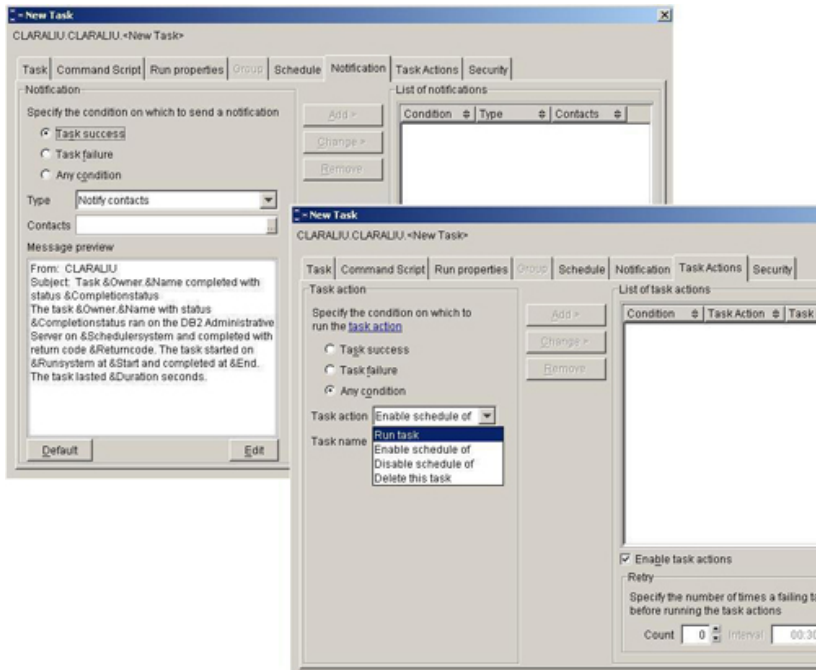
## Creating a new task

While the Task Center can create, edit, and manage DB2 database scripts, it can also create, edit, and manage Operating System command or shell scripts. To begin using the Task Center, enter or import a script under the **Command Script** tab. You are now ready to schedule this task, set its frequency (if it is repeated), and set the authorization ID used to execute the task.



## Setting notifications and task actions

The **Notification** tab lets you specify if and when to send out a notification that the task has completed. Here you specify who to send the notification to, and what the message text should be. In some cases you may want to use such notifications as *triggers* to run other tasks, depending upon the result of the task. You can use the **Task Actions** tab to run, schedule, or disable scheduling of another task.



## Creating a tools catalog database

The DB2 tools catalog is used to store task information created by the Task Center, and it must exist in order to use the DB2 Task Center to create and schedule tasks. The DB2 tools catalog can be created within any existing database or in a separate database using the `create tools catalog` command. The tools catalog requires both regular and system temporary table spaces, with a 32K page size. If these table spaces are not specified in the command, they will be created.

The following command creates a new database for the tools catalog, and within this database the tools catalog tables are created in the database schema `toolscat`:

```
db2 create tools catalog toolscat create new database toolsdb
```

The following command creates the tools catalog tables in the database schema `toolscat` in an existing database `toolsdb`. Within this database the tools catalog tables are created in the `tbbsp32k` table space:

```
db2 create tools catalog toolscat user existing tablespace tbbsp32k in database toolsdb
```

## Using notification logs

### Capturing diagnostic information

DB2 uses a first failure data capture (FFDC) mechanism to automatically capture information about errors and warnings as they are happening, rather than having to go back and reproduce the error to capture the diagnostic information. This diagnostic information is recorded in several places, such as: the administration notification log, DB2 diagnostic log, dump files, trap files, and (for Linux and UNIX) core files.

The most important for DBAs is the administration notification log because, as the name implies, this log is designed to contain information useful to DB2 database and system administrators. The DB2 diagnostic file (also called the `db2diag.log`) contains detailed information that is mainly used by DB2 customer support. The dump files capture information in a binary format named after the failing process ID when a process encounters a severe error. Trap and core files are generated when DB2 terminates abnormally and cannot continue processing. These files are also binary files and sometimes contain a memory dump for the process that was terminated.

Understanding how to interpret the notification logs is a skill that will grow over time, so do not feel overwhelmed the first time you look at one.

Each instance has one DB2 notification log, and on Linux and UNIX it is a file called `instance_name.nfy`. On Windows the notification log information is written into the Windows Event Log, and is part of the application log, with its application being DB2. You need to use the Windows Event Viewer to look at this on Windows; on Linux and UNIX you can use an ASCII editor or simply use the `more` command to view the log file. On Linux and UNIX, the `instance_name.nfy` file can be found in the instance owner's home directory, under the `sqllib/db2dump` directory.

## Setting the notification level

Information recorded in the administration logs can be written by DB2, the Health Monitor, and user applications. The `NOTIFYLEVEL` database manager configuration parameter determines what level of information, and ultimately how much information, will be captured. There are five levels of information possible:

- 0: No administration notification messages will be captured. This setting is not recommended.
- 1: Only fatal or unrecoverable errors are logged.
- 2: Any conditions that require immediate attention are logged. This level also captures Health Monitor alarms.
- 3: This is the default setting. It captures Health Monitor alarms, Health Monitor warnings, and Health Monitor attentions.
- 4: All error and informational messages are captured.

Note that DB2 captures all the levels of information up to and including the value set in `NOTIFYLEVEL`. For example, if `NOTIFYLEVEL` is set to 3, information of levels 1, 2, and 3 is logged.

## Interpreting the DB2 notification log

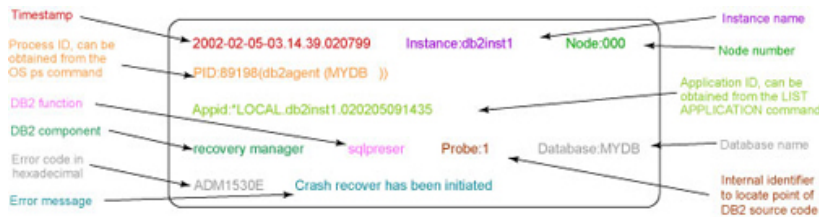
DB2 appends new errors, warnings, or informational events to the end of the DB2 notification log, so the administration log grows continuously on Linux and UNIX. Windows manages the size of the system event log, and rolls older events out of the log.

A good use of the Task Center would be to back up and then delete (or simply rename) these logs regularly.

Each event log entry is made up of different pieces:

- A timestamp indicating when the event occurred.
- The instance name, node ID, database name, process ID, application ID, or name of the DB2 or user application function encountering the error.
- The error type and unique identifier (a number in hex); this usually starts with DIA or ADM.
- A message explaining the error.

The figure below shows a typical administration notification log entry.



## Installing and configuring DB2

### Installing and configuring DB2

In Version 9, DB2 can be configured while you are installing DB2 or creating databases. By default the configuration advisor will automatically run when you create a DB2 instance or database. This will:

- Examine the server resources -- memory, CPUs, disks, and information about the application workload.
- Determine a good set of database manager or database configuration parameters so that this workload will run well.

By default, databases are also created with automatic maintenance enabled so you can tell DB2 to automatically reorganize tables, gather statistics (runstats), or take backups.

You can also use the `AUTOCONFIGURE` option on the command line when creating a database, and specify some or all of the server and workload characteristics, as follows:

```
db2 create database db_name autoconfigure using config-keyword value,config-keyword
value, ...
```

In DB2 V8, you could set the database configuration parameter `database_memory` to `AUTOMATIC` to allow DB2 to calculate the amount of memory needed. In V9, when you enable Self Tuning Memory (`SELF_TUNING_MEM`) DB2 will continuously examine the database workload and memory usage, and will automatically change certain database configuration parameters to keep the system running optimally. The key parameters that self tune memory changes are:

- Sort Heap
- Buffer Pools
- Lock List
- Package Cache
- Database Memory

Self Tuning Memory operates in two ways, based on the `database_memory` configuration parameter. If `database_memory` is set to `AUTOMATIC`, DB2 will allocate and release memory to or from the operating system as needed to increase and reduce the configuration parameters above to react to changes in the database workload. If `database_memory` is set to a specific value, DB2 will still perform self tuning but will allocate the specified amount of database shared memory, and will redistribute that memory between the configuration parameters above. It will not acquire more memory, or return memory to the operating system.

## Enabling Self Tuning Memory

Self Tuning Memory for the database is enabled by setting the `SELF_TUNING_MEM` parameter to `YES` or `ON`, as follows:

```
db2 update db cfg for dbname using self_tuning_mem yes
or
db2 update db cfg for dbname using self_tuning_mem on
```

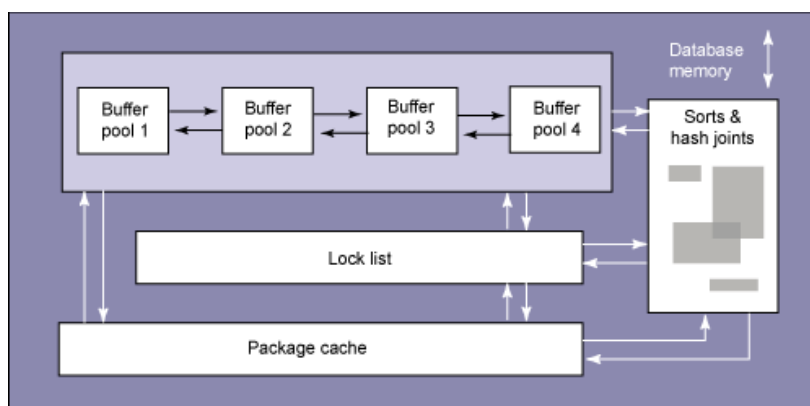
Whether DB2 uses and frees available memory as needed, or uses a set amount of memory is specified by the `DATABASE_MEMORY` parameter. To allow DB2 to use and free available memory as needed, use the command:

```
db2 update db cfg for dbname using database_memory automatic
```

To specify the amount of memory available to DB2, use the command:

```
db2 update db cfg for dbname using database memory 1000000
```

The figure below shows the components of self tuning memory, and how the `database_memory` configuration parameter sets the limit for this memory.



## Conclusion

### Summary

So that's the story on server management in DB2 9. This tutorial introduced you to the basic knowledge and skills of managing DB2 servers. Every server has one or more DB2 instances. An

instance provides a logical environment in which DB2 commands and functions can execute. You learned how to create, drop, start, stop, list, migrate, and update instances. The DB2 environment plays an important role in influencing how DB2 behaves. It is made up of the operating system environment settings, DB2 profile registries, database manager, and database configuration parameters. They can be easily configured with commands as illustrated in the tutorial.

You learned three methods of configuring DB2 client and server connectivity. You can search the network with DB2 discovery, use DB2 access profiles, or specify communication information manually. The Configuration Assistant is available to use any of the methods to set up the node and database catalog directories. With proper connectivity configuration, you can attach to an instance to perform remote administration tasks and connect to a database for data access.

Managing a DB2 server also includes managing its access. DB2 security is composed of authentication and authorization. Authentication is performed externally by a security facility. There are different authentication types that allow you to control where authentication should take place. Once a user is authenticated, DB2 will check to make sure the user is allowed to perform the requested operations. Different levels of authorities and privileges are available to support granular security control.

You were also introduced to task creation and automation. Use the Task Center to create tasks that are coded in DB2 commands, OS commands, or MVS shell commands. The tool has options to enter instructions for notification purposes, and to specify other tasks to perform upon completion.

The DB2 administration notification log records errors and warnings raised by DB2, the Health Monitor, and user applications. Each event log entry contains information such as timestamp, database name, application ID, DB2 function and component that raise the message. This is definitely a good place to start in troubleshooting errors encountered in DB2.

[Part 2](#), which discusses the creation of DB2 databases, as well as the various methods used for placing and storing objects within a database, provides an understanding of system calls, teaches you how to make your own modules, and, finally, shows you how to create, apply, and submit patches.

To keep an eye on this series, bookmark the series page [DB2 9 DBA certification \(Exam 731\) preparation tutorials](#).

© Copyright IBM Corporation 2006

([www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml))

Trademarks

([www.ibm.com/developerworks/ibm/trademarks/](http://www.ibm.com/developerworks/ibm/trademarks/))