DB2 9 Fundamentals exam 730 prep, Part 2: Security

Skill Level: Intermediate

Graham G. Milne (gmilne@ca.ibm.com)
I/T Specialist DB2 UDB
IBM Canada

20 Jul 2006

This tutorial introduces the concepts of authentication, authorization, and privileges as they relate to DB2® 9. It is the second in a series of seven tutorials designed to help you prepare for the DB2 9 Fundamentals Certification Exam (730). You should have basic knowledge of database concepts and operating system security. This is the second in a series of seven tutorials to help you prepare for the DB2® 9 for Linux®, UNIX®, and Windows™ Fundamentals exam 730.

Section 1. Before you start

About this series

Thinking about seeking certification on DB2 fundamentals (Exam 730)? If so, you've landed in the right spot. This series of seven DB2 certification preparation tutorials covers all the basics -- the topics you'll need to understand before you read the first exam question. Even if you're not planning to seek certification right away, this set of tutorials is a great place to start getting to learn what's new in DB2 9.

About this tutorial

In this tutorial, you'll learn about DB2 9 security features, including DB2 9 authentication, authorization, and privileges.

This is the second in a series of seven tutorials you can use to help prepare for the DB2 9 Fundamentals exam 730. The material in this tutorial primarily covers the objectives in Section 2 of the test, which is entitled "Security". You can view these objectives at: http://www.ibm.com/certify/tests/.

Prerequisites

To understand the concepts described in this tutorial, you should already have a basic knowledge of database concepts and an understanding of operating system security features.

System requirements

The examples in this tutorial are specific to DB2 9 running on a Windows™ operating system (with native security features). However, the concepts and information provided are relevant to DB2 running on any distributed platform.

You do not need a copy of DB2 9 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.

Setup

To complete the steps in this tutorial, you should have:

1. Logged into a Windows machine as a user who is a member of the Administrators group. In the examples in this tutorial, we will be logged in with the user ID *gmilne*.
2. Installed DB2 9.
3. Created a new group on the machine on which DB2 was installed. In this tutorial, the group ID *db2grp1* is used.
4. Created a second user ID on the machine on which DB2 was installed. In this tutorial, for this purpose we will use the user ID *test1*. Note that the *test1* user is not a member of the Administrators group.

Section 2. DB2 security

Aspects of database security

Database security is of utmost importance today. Your database might allow customers to purchase products over the Internet, or it can contain historical data used to predict business trends; either way, your company needs a sound database security plan.

A database security plan should define:

- Who is allowed access to the instance and/or database
- Where and how a user's password will be verified
- Authority level that a user is granted
- Commands that a user is allowed to run
- Data that a user is allowed to read and/or alter
- Database objects a user is allowed to create, alter, and/or drop

DB2 security mechanisms

There are three main mechanisms within DB2 that allow a DBA to implement a database security plan: *authentication*, *authorization*, and *privileges*.

Authentication is the first security feature you'll encounter when you attempt to access a DB2 instance or database. DB2 authentication works closely with the security features of the underlying operating system to verify user IDs and passwords. DB2 can also work with security protocols like Kerberos to authenticate users.

Authorization involves determining the operations that users and/or groups can perform, and the data objects that they may access. A user's ability to perform high-level database and instance management operations is determined by the authorities that they have been assigned. The five different authority levels within DB2 are SYSADM, SYSCTRL, SYSMAINT, DBADM, and LOAD.

Privileges are a bit more granular than authorities, and can be assigned to users and/or groups. Privileges help define the objects that a user can create or drop. They also define the commands that a user can use to access objects like tables, views, indexes, and packages. New to DB2 9 is the concept of label-based access control (LBAC), which allows more granular control of who can access individual rows and/or columns.

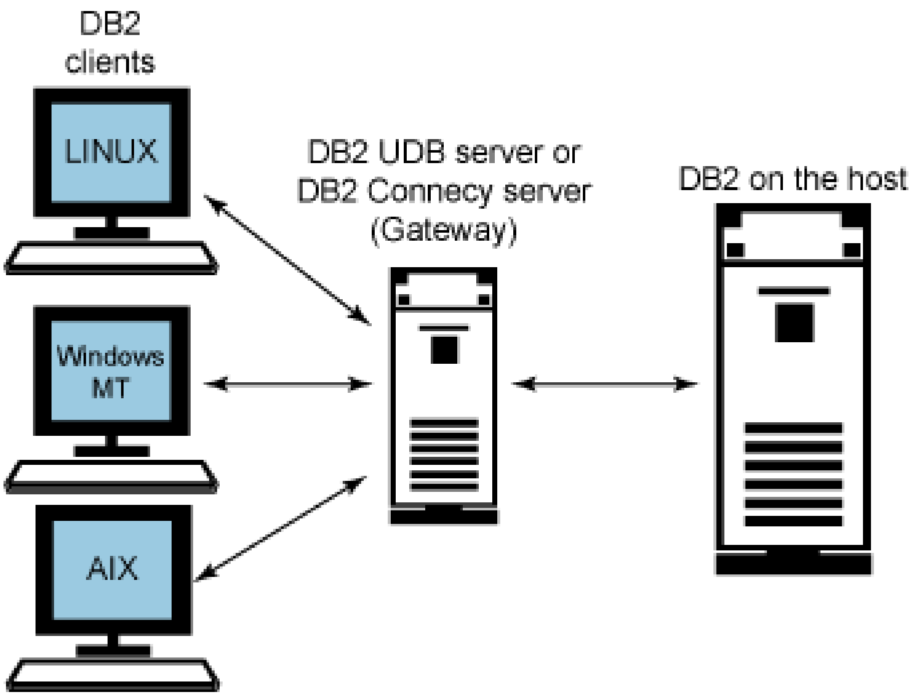
To prepare for the next section of the tutorial, you will need to create a database within the DB2 instance. Make sure that the `%DB2INSTANCE%` variable is still set to DB2, and then create the sample database using the command `db2sampl` *`drive`* , using the name of the drive where you want to create the sample. For the examples in this tutorial, you'll create the sample database on your D: drive, as follows:

```
D:\SQLLIB\BIN> db2sampl d:
```

Clients, servers, gateways, and hosts

It is particularly important that you understand the terms *client, server, gateway*, and *host* when considering the security of the entire database environment. A database environment often consists of several different machines; you must safeguard the database at any potential data access point. The concepts of clients, servers, gateways, and hosts are particularly important when dealing with DB2 authentication.

The diagram below illustrates a basic client-server-host configuration.



The *database server* is the machine (or machines in a partitioned database system) on which the database physically resides. The DB2 database *clients* are machines that are configured to run queries against the database on the server. These clients can be local (reside on the same physical machine as the database server) or they can be remote (reside on separate machines).

If the database resides on a mainframe machine running an operating system like AS/400 (iSeries) or OS/390 (zSeries), it's called a *host* or *host server.* A *gateway* is a machine running the DB2 Connect product. Through the gateway, DB2 client machines can connect to a DB2 database that resides on a host machine. The gateway is also referred to as the DB2 Connect Server. Systems with the Enterprise Server Edition product installed also have the DB2 Connect functionality built in.

Section 3. DB2 authentication

When DB2 authenticates

DB2 authentication controls the following aspects of a database security plan:

- Who is allowed access to the instance and/or database
- Where and how a user's password will be verified

It does this with the help of the underlying operating system security features whenever an *attach* or *connect* command is issued. An attach command is used to connect to the DB2 instance, whereas a connect command is used to connect to a database within a DB2 instance. The examples below walk you through the different ways that DB2 will authenticate a user issuing these commands. These examples use the default authentication type of SERVER in the database manager configuration file. Example 3 below illustrates how DB2 can be used to change the password on the OS of the server.

Log on to the machine where DB2 is installed with the user ID you used to create the DB2 instance. Issue the following commands:

```
db2 attach to DB2
```

Here, authentication is done implicitly. The user ID used to log onto the machine is used and is assumed to be already verified by the operating system.

```
db2 connect to sample user test1 using password
Database Connection Information
Database server         = DB2/NT 9.1.0
SQL authorization ID    = TEST1
Local database alias    = SAMPLE
```

Here, authentication is done explicitly. The user *test1* with the password *password* is verified by the operating system. User *test1* is successfully connected to the sample database.

```
db2 connect to sample user test1 using password new chgpass confirm chgpass
```

The user ID *test1* with password *password* is verified by the operating system as in example 2. The password for *test1* is then changed by the operating system from *password* to *chgpass*. As a result, the command in example 2 will fail if you reissue it.

DB2 authentication types

Authentication types are used by DB2 to determine where authentication is to take place. For example, in a client-server environment, will the client or the server machine verify the user's ID and password? In a client-gateway-host environment, will the client or host machine verify the ID and password?

DB2 9 has the ability to specify different authentication mechanisms depending on whether the user is attempting to connect to the database, or perform instance attachments and instance level operations. By default, the instance is set up to use one type of authentication for all instance level and connection level requests. This is specified by the Database Manager Configuration parameter AUTHENTICATION. Introduced in V9.1 is the Database Manager Configuration parameter SRVCON_AUTH. This parameter specifically deals with connections to databases. So, for example, if you have the following set in your DBM CFG:

```
DB2 GET DBM CFG
Server Connection Authentication          (SRVCON_AUTH) = KERBEROS
Database manager authentication        (AUTHENTICATION) = SERVER_ENCRYPT
```

Then attachments to the instance would use SERVER_ENCRYPT. Connections to the database however would use KERBEROS authentication. if KERBEROS was not properly initialized for the server but a valid userid / password was supplied then the user would be allowed to attach to the instance, but not allowed to connect to the database.

The following table summarizes the available DB2 authentication types. In a client-gateway-host environment, these authentication options are set on the client and gateway, not on the host machine. Setting these options is discussed in more detail throughout this section. See Clients, servers, gateways, and hosts for a refresher.

Type	**Description**
SERVER	Authentication takes place on the server.
SERVER_ENCRYPT	Authentication takes place on the server. Passwords are encrypted at the client machine before being sent to the server.
CLIENT	Authentication takes place on the client machine (see Dealing with untrusted clients for exceptions).
*KERBEROS	Authentication is performed by the Kerberos security software.
*KRB_SERVER_ENCRYPT	Authentication is performed by Kerberos security software if the client setting is KERBEROS. Otherwise, SERVER_ENCRYPT is used.
DATA_ENCRYPT	Authentication takes place on the server. The server accepts encrypted userids and

	passwords, and will encrypt the data. This operates the same way as SERVER_ENCRYPT, except the data is encrypted as well.
DATA_ENCRYPT_CMP	Authentication is the same as for DATA_ENCRYPT, except that this scheme allows older clients that don't support the DATA_ENCRYPT scheme to connect using SERVER_ENCRYPT authentication. The data in this case will not be encrypted. if the client connecting supports DATA_ENCRYPT, it is forced to encrypt the data, and can not downgrade to SERVER_ENCRYPT authentication. This authentication type is only valid in the server's database manager configuration file and is not valid when used on the CATALOG DATABASE command on a client or gateway instance.
GSSPLUGIN	Authentication is controlled by an external GSS-API plugin.
GSS_SERVER_ENCRYPT	Authentication is controlled by an external GSS-API plugin. In the case where the client doesn't support one of the server's GSS-API plugins, SERVER_ENCRYPT authentication is used.

*These settings are valid only for Windows 2000, AIX, Solaris and Linux® operating systems.

Setting authentication on the server

Authentication is set on the database server within the Database Manager Configuration (DBM CFG) file using the AUTHENTICATION parameter. Remember, the DBM CFG file is an instance-level configuration file. Thus, the AUTHENTICATION parameter affects all databases within the instance. The following commands illustrate how this parameter can be altered.

To view the authentication parameter in the configuration file:

```
db2 get dbm cfg
```

To alter the authentication parameter to `server_encrypt`:

```
C:\PROGRA~1\SQLLIB\BIN> db2 update dbm cfg using authentication server_encrypt
C:\PROGRA~1\SQLLIB\BIN> db2stop
C:\PROGRA~1\SQLLIB\BIN> db2start
```

Certain authentication types, like GSSPLUGIN, KERBEROS, and CLIENT require the setting of other Database Manager Configuration parameters such as TRUST_ALLCLNTS, SRV_PLUGIN_MODE, and SRVCON_PW_PLUGIN. More details on these settings below.

Setting authentication on the gateway

Authentication is set on the gateway using the catalog database command. For the examples here, we'll use a host database named *myhostdb*.

To set the gateway authentication type to SERVER, you would issue the following command on the gateway machine:

```
db2 catalog database myhostdb at node nd1 authentication SERVER
db2 terminate
```

Note that authentication is never performed on the gateway itself. In DB2 Version 8, authentication must always occur at either the client or the host database server.

Setting authentication on the client

Let's consider two scenarios on two separate client machines. We'll configure one to connect to a database on a server machine (DB2 UDB LUW distributed platform), and the other to connect to a database on a host machine (DB2 for zSeries, for example).

- **Client connecting to a server database:** The client authentication setting in the database directory entry for the database being connected to must match that of the database server (with the exception of KRB_SERVER_ENCRYPT, DATA_ENCRYPT_CMP, and GSS_SERVER_ENCRYPT).
  Let's assume the server authentication type is set to SERVER. The following command would then be issued on the client to catalog the server database named *sample*:

```
db2 catalog database sample at node nd1 authentication SERVER
```

  If the authentication type is not specified, the client will try to use SERVER_ENCRYPT by default.

- **Client connecting to a host database:** Let's assume that the authentication type on the gateway is set to SERVER. If an authentication type is not specified, SERVER_ENCRYPT authentication is assumed when accessing a database through DB2 Connect. Authentication will take place on the host database server. The following command issued from the client will cause the client to send unencrypted userids and passwords to the gateway:

```
db2 catalog database myhostdb at node nd1 authentication SERVER
```

Now let's assume authentication is set to SERVER_ENCRYPT on the gateway. Authentication will once again take place on the host database server. The userid and password is encrypted on the client before being sent to the gateway, and encrypted on the gateway before being sent to the host machine. This is the default behaviour.

Dealing with untrusted clients

If the server or gateway machine has authentication set to CLIENT, this implies that the client is expected to authenticate a user's ID and password. However, some client machines may not have operating systems with native security features. Such *untrusted* clients include DB2 clients running on Windows 98 and Windows ME. DB2 V9.1 does not support Windows 98 or Windows ME, but it does support downlevel clients and so may still have to deal with untrusted V8 clients.

There are two additional parameters in the DBM CFG file used to determine where authentication should take place when the server or gateway authentication method is set to CLIENT and untrusted clients are attempting to connect to the database or attach to the DB2 instance. These are the TRUST_ALLCLNTS and TRUST_CLNTAUTH parameters.

When the server or gateway authentication type is CLIENT, there are two other factors that come into play in addition to the TRUST_ALLCLNTS and TRUST_CLNTAUTH parameters. The first is whether a user ID and password were explicitly supplied and the second is the type of client connecting. The three DB2 clients are:

- **Untrusted clients**: As described above
- **Host clients**: Clients running on host operating systems like zSeries
- **Trusted clients**: Clients running non-host operating systems that have native security features such as Windows NT, 2000, 2003, XP and all forms of Unix / Linux.

When authentication is set to CLIENT

The table below summarizes where authentication will take place when a connect or attach command is issued by each type of client to a server whose authentication type is set to CLIENT.

User ID/Password Supplied?	TRUST_ALLCLNTS	TRUST_CLNTAUTH	Untrusted Client	Trusted Client	Host Client
No	Yes	CLIENT	CLIENT	CLIENT	CLIENT
No	Yes	SERVER	CLIENT	CLIENT	CLIENT

No	No	CLIENT	SERVER	CLIENT	CLIENT
No	No	SERVER	SERVER	CLIENT	CLIENT
No	DRDAONLY	CLIENT	SERVER	SERVER	CLIENT
No	DRDAONLY	SERVER	SERVER	SERVER	CLIENT
Yes	Yes	CLIENT	CLIENT	CLIENT	CLIENT
Yes	Yes	SERVER	SERVER	SERVER	SERVER
Yes	No	CLIENT	SERVER	CLIENT	CLIENT
Yes	No	SERVER	SERVER	SERVER	SERVER
Yes	DRDAONLY	CLIENT	SERVER	SERVER	CLIENT
Yes	DRDAONLY	SERVER	SERVER	SERVER	SERVER

DRDAONLY refers to host clients only, despite the fact that DB2 Version 8 clients connect using DRDA as well.

The examples below illustrate setting authentication types and parameters on the server and client:

Setting authentication on the server:

```
db2 update dbm cfg using authentication client
db2 update dbm cfg using trust_allclnts yes
db2 update dbm cfg using trust_clntauth server
db2stop
db2start
```

Setting authentication on the client:

```
db2 catalog database sample at node nd1 authentication client
```

In the above example, if the command

```
db2 connect to sample
```

is issued from any client, authentication takes place on the client. If the command

```
db2 connect to sample user test1 using password
```

is issued from any client, authentication takes place on the server.

DB2's security plugin architecture

DB2 V8.2 introduced the concept of security plugins for DB2. This concept has been further enhanced in DB2 V9.1. Using standard GSS-API calls, a user can write a security plugin and pass the job of authenticating the userid to an external security program. An example of this is DB2's own KERBEROS authentication. When you install DB2 ESE, or the application development client on a machine part of that install places sample application code in your instance directory. if you look in the **samples\security\plugins** directory you will see in there examples of how to code security plugins. This section will outline the use of plugins in the DB2 security architecture, but does not cover how to code or compile the plugins themselves. for a detailed description of how this is done, please refer to DB2 UDB Security Part 2: Understand the DB2 Universal Database Security plug-ins.

Kerberos authentication

Kerberos authentication provides DB2 a way to authenticate users without having to flow userids or passwords over the network. The Kerberos security protocol performs authentication as a third party authentication service by using conventional cryptography to create a shared secret key. This key becomes a user's credential and is used to verify the identity of users during all occasions when local or network services are requested. Using the Kerberos security protocol enables the use of a single sign-on to a remote DB2 database server.

First, lets review the setup of DB2 to use Kerberos authentication. As mentioned above, Kerberos authentication is implemented in DB2 using the plugin architecture. The source code for the default kerberos plugin is provided in the `samples/security/plugins` directory, called `IBMkrb5.c`. Before the Kerberos authentication will work for DB2, Kerberos has to be enabled and supported on both client and server. In order for this to work, the following conditions must be met:

1. The client and server machines must belong to the same realm (trusted domains in Windows terminology
2. The appropriate Principals (userids in Kerberos) must be set up.
3. The server's keytab file must be created and readable by the instance owner.
4. All machines must have synchronised clocks.

You can find more information on setting up Kerberos in the documentation accompanying the Kerberos product installed.

To enable DB2 to use KERBEROS authentication you must first tell the client where to find the kerberos plugin you are using. On the client, run the following command:

```
DB2 UPDATE DBM CFG USING CLNT_KRB_PLUGIN IBMkrb5
DB2 TERMINATE
```

In this example, the default KERBEROS plugin is used. This could have been modified by the DBA to perform special functions if they were required by the

Kerberos implementation being used.

There is also the ability to tell the client exactly which server principal it is authenticating against. This option bypasses the first step of Kerberos authentication where the client has to discover the server principal of the instance it is connecting to. The AUTHENTICATION parameter can be specified when cataloggin the database on the client. Its format is:

```
DB2 CATALOG DB dbname AT NODE node name AUTHENTICATION KERBEROS TARGET PRINCIPAL
   service/host@REALM
```

This step is optional.

```
DB2 CATALOG DB sample AT NODE testnd AUTHENTICATION KERBEROS TARGET PRINCIPAL
   gmilne/gmilne02.torolab.ibm.com@TOROLAB.IBM.COM
```

The next step to set up Kerberos authentication is to set up the server. The `srvcon_gssplugin_list`. This parameter can be set up with a list of different supported GSS-API plugins, but you are only allowed one Kerberos plugin. if no Kerberos plugin is in the list, the default IBMkrb5 plugin is automatically used. if you intend to allow all authentication (instance attachments as well as database connections) to use Kerberos, then perform the following:

```
DB2 UPDATE DBM CFG USING AUTHENTICATION KERBEROS
or
DB2 UPDATE DBM CFG USING AUTHENTICATION KRB_SERVER_ENCRYPT
```

if you only want DB2 to use Kerberos to authenticate incoming database connections (and use SERVER for incoming instance attachments), then perform the following:

```
DB2 UPDATE DBM CFG USING SVRCON_AUTH KERBEROS
or
DB2 UPDATE DBM CFG USING SVRCON_AUTH KRB_SERVER_ENCRYPT
```

Depending on the bit width (32 or 64 bit) of the instance, DB2 will automatically load the IBMkrb5 plugin when the instance is started.

Other Authentication Settings

if you look in the DBM CFG for a V9.1 instance, you will see various settings that can affect the way that DB2 will authenticate userids. As mentioned above, there are settings for standard OS userid authentication (CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT, DATA_ENCRYPT_CMP), as well as plugins for passing authentication to external programs (KERBEROS, KRB_SERVER_ENCRYPT, GSSPLUGIN, GSS_SERVER_ENCRYPT). This section deals specifically with some of the other configuration variables that can have an impact on how a user is authenticated.

```
Client Userid-Password Plugin          (CLNT_PW_PLUGIN) =
Group Plugin                             (GROUP_PLUGIN) =
GSS Plugin for Local Authorization    (LOCAL_GSSPLUGIN) =
Server Plugin Mode                    (SRV_PLUGIN_MODE) = UNFENCED
Server List of GSS Plugins      (SRVCON_GSSPLUGIN_LIST) =
Server Userid-Password Plugin        (SRVCON_PW_PLUGIN) =
Cataloging allowed without authority   (CATALOG_NOAUTH) = NO
Bypass federated authentication            (FED_NOAUTH) = NO
```

In the above list, the parameters already discussed have been removed.

CLNT_PW_PLUGIN	This parameter is specified on the client side DBM CFG. It specifies the name of the client plugin used for client and local authentication.
GROUP_PLUGIN	The default of this value is blank (NULL). Setting this to the name of a user defined plugin will invoke that plugin for all group enumeration instead of relying on the operating system group lookup. This is tied in to the authorization sections discussed later.
LOCAL_GSSPLUGIN	This parameter specifies the name of the default GSS API plug-in library to be used for instance level local authorization when the value of the authentication database manager configuration parameter is set to GSSPLUGIN or GSS_SERVER_ENCRYPT.
SRV_PLUGIN_MODE	(YES/NO) The default setting for this parameter is NO. When changed to YES, the GSS-API Plugins used are launched in a FENCED mode, similar to the way that FENCED stored procedures work. A FENCED plugin that crashes can not cause the DB2 instance to crash. while the plugins are being developed, it is recommended to run them in a fenced mode so that logic problems and memory leaks in those plugins will not crash the instance. Once the plugin is determined to be safe, it should be run unfenced for performance reasons.
SRVCON_GSSPLUGIN_LIST	A list of plugins that the database manager on the server will use during authentication when either KERBEROS, KRB_SERVER_ENCRYPT, GSSPLUGIN or GSS_SERVER_ENCRYPT are used.. Each plugin in the list should be separated by a comma (',') with no spaces in between. The plugins are listed in order of preference, with the first one in the list being used first to attempt to authenticate the userid / password sent. Only when all the plugins listed have returned an error will DB2 return an authentication error to the user.
SRVCON_PW_PLUGIN	This parameter allows the user to change the default authentication DB2 uses to verify userids

	and passwords when either CLIENT, SERVER, or SERVER_ENCRYPT authentication is specified. By default, its value is NULL and the default DB2 methods are used.
CATALOG_NOAUTH	(YES/NO) Default NO. Changing this parameter to YES allows users that are not verified to be members of the SYSADM, SYSCTRL, or SYSMAINT groups to change the Database, Node, Admin and DCS catalogs on the machine. This is only useful in client scenarios where the user logged into the machine is either using an untrusted client (defined above) or are logged on with a userid that is not allowed to connect to the database or attach to the instance but must catalog entries on the client machine.
FED_NOAUTH	When fed_noauth is set to yes, authentication is set to server or server_encrypt, and federated is set to yes, then authentication at the instance is bypassed. It is assumed that authentication will happen at the data source. Exercise caution when fed_noauth is set to yes. Authentication is done at neither the client nor at DB2. Any user who knows the SYSADM authentication name can assume SYSADM authority for the federated server.

Section 4. DB2 authorities

Introduction to authorities

DB2 authorities control the following aspects of a database security plan:

- The authority level that a user is granted
- The commands that a user is allowed to run
- The data that a user is allowed to read and/or alter
- The database objects a user is allowed to create, alter, and/or drop

Authorities are made up of groups of privileges and higher-level database manager (instance-level) maintenance and utility operations. Of the five authorities available in DB2, SYSADM, SYSCTRL, SYSMAINT, and SYSMON are *instance-level authorities*. That means that their scope includes instance-level commands as well as commands against all the databases within the instance. These authorities can only be assigned to a group; you can do so through the DBM CFG file.

The DBADM, LOAD, and SECADM authorities are assigned to a user or group for a particular database. This can be done explicitly using the `GRANT` command.

The following sections describe how each authority is assigned and what commands users with that authority are allowed to perform. Note that any reference to group membership implies that the user and group names have already been defined at the operating system level.

Users can determine what authorities and database-level privileges they have by issuing the following command:

```
db2 get authorizations
```

Obtaining SYSADM authority

SYSADM authority in DB2 is comparable to root authority on UNIX or Administrator authority on Windows. Users with SYSADM authority for a DB2 instance are able to issue any DB2 commands against that instance, any databases within the instance, and any objects within those databases. They also have the ability to access data within the databases and grant or revoke privileges and authorities. SYSADM users are the only users allowed to update the DBM CFG file.

SYSADM authority is controlled in the DBM CFG file via the SYSADM_GROUP parameter. When the instance is created, this parameter is set to Administrator on Windows (although it appears blank if you issue the command `db2 get dbm cfg`). On UNIX, it is set to the primary group of the user who created the instance.

Since SYSADM users are the only users allowed to update the DBM CFG, they are also the only ones allowed to grant any of the SYS* authorities to other groups. The following example illustrates how to grant SYSADM authority to the group *db2grp1*:

```
  db2 update dbm cfg using SYSADM_GROUP db2grp1
```

Remember, this change will not take effect until the instance is stopped and then restarted. Also, keep in mind that if you are not currently logged in as a member of *db2grp1*, you may not have authority to restart the instance! You would have to log out and log back in with an ID in the correct group, or add your current ID to *db2grp1*.

Obtaining SYSCTRL authority

Users with SYSCTRL authority can perform all administrative and maintenance commands within the instance. However, unlike SYSADM users, they cannot access any data within the databases unless they are granted the privileges required to do so. Examples of commands a SYSCTRL user can perform against any database in the instance are:

- `db2start/db2stop`

- `db2 create/drop database`
- `db2 create/drop tablespace`
- `db2 backup/restore/rollforward database`
- `db2 runstats` (against any table)
- `db2 update db cfg for database` *`dbname`*

A user with SYSADM authority can assign SYSCTRL to a group using the following command:

```
db2 update dbm cfg using SYSCTRL_GROUP group name
```

Obtaining SYSMAINT authority

The commands that a user with SYSMAINT authority can issue are a subset of those allowed to users with SYSCTRL authority. SYSMAINT users can only perform tasks related to maintenance, such as:

- `db2start/db2stop`
- `db2 backup/restore/rollforward database`
- `db2 runstats` (against any table)
- `db2 update db cfg for database` *`dbname`*

Notice that users with SYSMAINT cannot create or drop databases or tablespaces. They also cannot access any data within the databases unless they are granted the explicit privileges required to do so.

If you have SYSADM authority, you can assign SYSMAINT authority to a group using the following command:

```
db2 update dbm cfg using SYSMAINT_GROUP group name
```

Obtaining SYSMON authority

SYSMON authority provides the ability to take database system monitor snapshots of a database manager instance or its databases. SYSMON authority is assigned to the group specified by the `sysmon_group` configuration parameter. If a group is specified, membership in that group is controlled outside the database manager through the security facility used on your platform.

SYSMON authority enables the user to run the following commands:

- `GET DATABASE MANAGER MONITOR SWITCHES`
- `GET MONITOR SWITCHES`
- `GET SNAPSHOT`
- `LIST ACTIVE DATABASES`
- `LIST APPLICATIONS`
- `LIST DCS APPLICATIONS`
- `RESET MONITOR`
- `UPDATE MONITOR SWITCHES`

SYSMON authority enables the user to use the following APIs:

- `db2GetSnapshot` - Get snapshot
- `db2GetSnapshotSize` - Estimate size required for `db2GetSnapshot()` output buffer
- `db2MonitorSwitches` - Get/update monitor switches
- `db2ResetMonitor` - Reset monitor

SYSMON authority enables the user to use all snapshot SQL table functions without previously running `SYSPROC.SNAP_WRITE_FILE`. `SYSPROC.SNAP_WRITE_FILE` takes a snapshot and saves its content into a file. If any snapshot table functions are called with null input parameters, the file content is returned, instead of a real-time system snapshot.

Users with the SYSADM, SYSCTRL, or SYSMAINT authority level also possess SYSMON authority.

A user with SYSADM authority can assign SYSMON to a group using the following command:

```
db2 update dbm cfg using SYSMON_GROUP group name
```

Obtaining DBADM authority

DBADM authority is a database-level authority rather than an instance-level authority. In summary, DBADM users have complete control over a database -- almost. DBADM users cannot perform such maintenance or administrative tasks as:

- `drop database`
- `drop/create tablespace`
- `backup/restore database`

- `update db cfg for database` *`db name`*

However, they can perform the following tasks:

- `db2 create/drop table`
- `db2 grant/revoke` (any privilege)
- `db2 runstats` (any table)

DBADM users are also automatically granted all privileges to the database objects and their contents. Since DBADM authority is a database-level authority, it can be assigned to both users and groups. The following commands illustrate different ways in which you can give DBADM authority.

- `db2 create database test`
  This command gives implicit DBADM authority on the database named *test* to the user who issued the command.

- `db2 connect to sample`
  `db2 grant dbadm on database to user tst1`
  This command can only be issued by SYSADM users; it issues DBADM authority to the user *tst1* on the sample database. Note that the issuing user must be connected to the sample database before granting DBADM authority.

- `db2 grant dbadm on database to group db2grp1`
  This command grants DBADM authority to everyone in the group *db2grp1*. Again, only SYSADM users can issue this command.

Obtaining LOAD authority

LOAD authority is also considered a database-level authority, and can therefore be granted to both users and groups. As the name implies, LOAD authority allows users to issue the LOAD command against a table. The LOAD command is typically used as a faster alternative to insert or import commands when populating a table with large amounts of data. Depending on the type of LOAD you wish to perform, having LOAD authority alone may not be sufficient. Specific privileges on the table may also be required.

The following commands can be run by users with LOAD authority:

- `db2 quiesce tablespaces for table`
- `db2 list tablespaces`
- `db2 runstats` (any table)

- `db2 load insert` (must have insert privilege on table)
- `db2 load restart/terminate after load insert` (must have insert privilege on table)
- `db2 load replace` (must have insert and delete privilege on table)
- `db2 load restart/terminate after load replace` (must have insert and delete privilege on table)

Only users with either SYSADM or DBADM authority are permitted to grant or revoke LOAD authority to users or groups. The following examples illustrate how LOAD authority can allow our user to `LOAD` data into a table called *sales.* Assume that the command `db2 connect to sample` has already been issued.

-
  ```
  db2 grant load on database to user tst1
  db2 grant insert on table sales to user tst1
  ```
  With LOAD authority and insert privilege, *tst1* could issue a `LOAD INSERT` or a `LOAD RESTART`, or `TERMINATE` after a `LOAD INSERT` against the sales table.

-
  ```
  db2 grant load on database to group grp1
  db2 grant delete on table sales to group grp1
  db2 grant insert on table sales to group grp1
  ```
  With LOAD authority, as well as delete and insert privileges, any member of *grp1* could issue a `LOAD REPLACE` or a `LOAD RESTART`, or `TERMINATE` after a `LOAD REPLACE` against the sales table.

Obtaining SECADM authority

SECADM authority is considered a database-level authority, but can only be granted to a specific user by a SYSADM user. A user with SECADM can perform the following:

- Create and drop security label components
- Create and drop security policies
- Create and drop security labels
- Grant and revoke security labels
- Grant and revoke LBAC rule exemptions
- Grant and revoke setsessionuser privileges
- Execute the SQL statement `TRANSFER OWNERSHIP` on objects that you do not own

No other user can perform these functions, not even the SYSADM, unless SECADM was explicitly granted to that SYSADM user. This is important because these security abilities are very powerful and should only be granted to a user who is defined as a security administrator. See the "Label-based access control" section for more information on this security feature new to DB2 V9.

Section 5. DB2 privileges

Database and object privileges

In the preceding section, the concept of privileges was briefly touched. Privileges can generally be placed into two main categories: database-level privileges, which span all objects within the database, and object-level privileges, which are associated with a specific object.

The database-level privileges that a user might be given are:

- CREATETAB: Users can create tables within the database.
- BINDADD: Users can create packages in the database using the BIND command.
- CONNECT: Users can connect to the database.
- CREATE_NOT_FENCED: Users can create unfenced user-defined functions (UDFs).
- IMPLICIT_SCHEMA: Users can implicitly create schemas within the database without using the CREATE SCHEMA command.
- LOAD: Users can load data into a table
- QUIESCE_CONNECT: Users can access a database while it is in a quiesced state.
- CREATE_EXTERNAL_ROUTINE: Users can create a procedure for use by applications and other users of the database.

Database *objects* include tables, views, indexes, schemas, and packages. Fortunately, most of the object-level privileges are self explanatory. The following table summarizes these privileges.

Privilege name	**Relevant object(s)**	**Description**
CONTROL	Table, View, Index, Package, Alias, Distinct Type, User Defined function, Sequence	Provides full authority on the object. Users with this privilege can also grant or revoke privileges on the object to other users.

DELETE	Table, View	Allows users to delete records from the object.
INSERT	Table, View	Allows users to insert records into the object via the INSERT or the IMPORT commands.
SELECT	Table, View	Provides the ability to view the contents of the object using the select statement.
UPDATE	Table, View	Allows users to modify records within the object using the update statement.
ALTER	Table	Allows users to alter the object definition using the alter statement.
INDEX	Table	Allows users to create indexes on the object using the create index statement.
REFERENCES	Table	Provides the ability to create or drop foreign key constraints on the object.
BIND	Package	Allows users to rebind existing packages.
EXECUTE	Package, Procedure, Function, Method	Allows users to execute packages and routines.
ALTERIN	Schema	Allows users to modify definitions of objects within the schema.
CREATEIN	Schema	Allows users to create objects within the schema.
DROPIN	Schema	Allows users to drop objects within the schema.

Information on object-level privileges is stored in the system catalog views. The view names are `syscat.tabauth`, `syscat.colauth`, `syscat.indexauth`, `syscat.schemaauth`, `syscat.routineauth`, and `syscat.packageauth`.

Explicit privileges

Privileges can be *explicitly* granted and revoked to users or groups using the GRANT and REVOKE commands. Let's take a look at how you can use these commands on various objects.

While logged in as a user with Administrator authority on Windows, bring up two DB2 command windows. Make sure that the `db2instance` variable is set to `DB2` in both windows!

From Window 1, issue the following command:

```
db2 connect to sample
```

Now, from Window 2, issue this command:

```
db2 connect to sample user test1 using password
```

Remember, the commands in Window 1 are being issued by a user with SYSADM authority. The commands in Window 2 are being issued by *tst1*, a user with no specific authority or privileges on the sample database. Note that the schema name associated with the tables in your sample database will be the name of the user that issued the `db2sampl` command. In these examples, that user is *GMILNE*.

Now, from Window 2, issue the following command:

```
db2 select * from gmilne.org
```

You should see this response:

```
SQL0551N  "TEST1" does not have the privilege to perform operation "SELECT"
on object "GMILNE.ORG".
```

To correct the situation, issue the following command from Window 1:

```
db2 grant select on table gmilne.org to user test1
```

Now the earlier command will succeed! Next, let's issue a more ambitious command from Window 2:

```
db2 insert into gmilne.org values (100, 'Tutorial', 1, 'Eastern', 'Toronto')
```

Again, you'll see an error message:

```
SQL0551N  "TEST1" does not have the privilege to perform operation  "INSERT"
on object "GMILNE.ORG"
```

So, enter the following command from Window 1:

```
db2 grant insert on table gmilne.org to group db2grp1
```

The earlier failed INSERT command should now complete successfully, because *test1* is a member of group *db2grp1*.

Now, enter the following command in Window 2:

```
db2 drop table gmilne.emp_photo
```

Again, you'll see an error message:

```
SQL0551N  "TEST1" does not have the privilege to perform operation "DROP TABLE"
on object "GMILNE.EMP_PHOTO".
```

So, we'll have the grant that privilege. Enter the following from Window 1:

```
db2 grant dropin on schema gmilne to all
```

The `DROP TABLE` command should now complete successfully.

Now that we've finished our example, let's revoke all the privileges you just granted. Do so by issuing the following commands from Window 1:

```
db2 revoke select on table gmilne.org from user test1
db2 revoke insert on table gmilne.org from group db2grp1
db2 revoke dropin on schema gmilne from all
```

Note that revoking privileges from a group does not necessarily revoke it from all members of that group. For example, the following command could have been used to revoke all privileges (except CONTROL) from *db2grp1* on the table gmilne.org:

```
db2 revoke all on table gmilne.org from group db2grp1
```

However, the user *test1* (who is a member of *db2grp1*) would have kept the select privileges on that table, since he or she had been granted that privilege directly.

Implicit privileges

DB2 may grant privileges automatically when certain commands are issued, without the need for an explicit GRANT statement to be issued, as you saw previously. The table below summarizes some commands that result in privileges being implicitly granted by the database manager. Note that these privileges are implicitly revoked when the object created is dropped. They are not, however, revoked when higher-level privileges are explicitly revoked.

Command issued	**Privilege granted**	**To whom it is granted**
`CREATE TABLE mytable`	CONTROL on *mytable*	User issuing command
`CREATE SCHEMA myschema`	CREATEIN, ALTERIN, DROPIN on *myschema*, plus the ability to grant these to others	User issuing command
`CREATE VIEW myview`	CONTROL on *myview* only if CONTROL is held on all tables and views referenced in the definition of myview	User issuing command
`CREATE DATABASE mydb`	SELECT on *mydb* 's system catalog tables, IMPLICIT_SCHEMA on mydb *	PUBLIC**

*When a user creates a database, that user is implicitly granted DBADM authority on that database. With DBADM authority comes implicit CONNECT, CREATETAB, BINDADD, IMPLICIT_SCHEMA, and CREATE_NOT_FENCED privileges. These privileges will remain with the user even if the DBADM authority is revoked.

**PUBLIC is a special DB2 group that includes all users of a particular database. Unlike the other groups we've discussed thus far, PUBLIC does not have to be defined at the operating system level. There are some privileges granted to PUBLIC by default. For example, this group receives CONNECT privilege on the database and SELECT privilege on the catalog tables automatically. GRANT and REVOKE commands can be issued against the PUBLIC group, like so:

```
db2 grant select on table sysibm.systables to public
db2 revoke select on table sysibm.systables from public
```

Indirect privileges

Privileges can be obtained indirectly when *packages* are executed by the database manager. A package contains one or more SQL statements that have been converted into a format that DB2 uses internally to execute them. In other words, a package contains multiple SQL statements in an executable format. If all the statements in the package are static, a user would only require EXECUTE privilege on the package to successfully execute the statements in the package.

For example, assume *db2package1* executes the following static SQL statements:

```
db2 select * from org
db2 insert into test values (1, 2, 3)
```

In this case, a user with EXECUTE privilege on *db2package1* would indirectly be granted SELECT privilege on the table org and INSERT privilege on the table test.

Label-based access control

New to DB2 9 is the concept of label-based access control (LBAC). What LBAC provides the DBA is the ability to restrict read / write privileges on the row or column level of a table.

Previously, the only way to introduce these restrictions was to create a view, authorize the view's use by the user in question, and remove access to the base table.

This tutorial will only demonstrate one example of a LBAC security scenario. For a more detailed explanation of LBAC, please refer to DB2 Label-Based Access Control, a practical guide, Part 1: Understand the basics of LBAC in DB2 on developerWorks.

LBAC is set up by the *security administrator* by creating Security Policies. Each table may only be subscribed to one security policy, but the system may have as many security policies as you'd like. There are several steps required to set up LBAC. The first thing you must do is determine the type of access control you require for your data.

Lets assume the following. In your organization there are three sets of people.

Name	Organizational Role
Jane	Human Resources Executive
Joe	Manager of Department D11 and E21
Frank	Team Lead - Department A00

Now, in the organization's database there is a table that defines Employee information. This will be based off of the EMP table in the SAMPLE database. It contains data on employees and the departments they belong to. It's existing definition is as follows:

```
                                db2 => describe select * from emp

SQLDA Information

 sqldaid : SQLDA     sqldabc: 896  sqln: 20  sqld: 14

 Column Information

 sqltype               sqllen  sqlname.data                    sqlname.length
 --------------------  ------  ------------------------------  --------------
 452   CHARACTER            6  EMPNO                                        5
 448   VARCHAR             12  FIRSTNME                                     8
 453   CHARACTER            1  MIDINIT                                      7
 448   VARCHAR             15  LASTNAME                                     8
 453   CHARACTER            3  WORKDEPT                                     8
 453   CHARACTER            4  PHONENO                                      7
 385   DATE                10  HIREDATE                                     8
 453   CHARACTER            8  JOB                                          3
 500   SMALLINT             2  EDLEVEL                                      7
 453   CHARACTER            1  SEX                                          3
 385   DATE                10  BIRTHDATE                                    9
 485   DECIMAL           9, 2  SALARY                                       6
 485   DECIMAL           9, 2  BONUS                                        5
 485   DECIMAL           9, 2  COMM                                         4
```

The organization has rules in place that are audited on a regular basis. Part of this audit indicates that the employees should not have access to data that is considered confidential. The rules stipulate that executives have full read / write access to all employee records, Managers have read / write access to anyone in their department, and team leads have read access to anyone in the department they lead.

To set up LBAC security to enable these rules.

1. Defining the security policies and labels and granting the security labels to the users

2. Modification of the EMP table including the security label column and attaching the security policy to it

Defining the security policies and labels

To define the security policies and labels, SECADM authority is required.

Step 1a. Create the security label component

The first thing you need to do is to determine the best type of security component to define for this policy. In this particular case, the best fit is a policy type of "TREE". A Tree policy means that you can define a set of labels such that the children have a subset of the rights that their parent does. In this example, create a security component named "J_DEPT".

```
                CREATE SECURITY LABEL COMPONENT J_DEPT
TREE ('HR_EXECUTIVE' ROOT,
      'MAN_D11_E21' UNDER 'HR_EXECUTIVE'
      'A00' UNDER 'HR_EXECUTIVE',
      'B01' UNDER 'HR_EXECUTIVE',
      'C01' UNDER 'HR_EXECUTIVE',
      'D11' UNDER 'MAN_D11_E21',
      'D21' UNDER 'HR_EXECUTIVE',
      'E01' UNDER 'HR_EXECUTIVE',
      'E11' UNDER 'HR_EXECUTIVE',
      'E21' UNDER 'MAN_D11_E21'
)
```

The above layout indicates that the root is HR_EXECUTIVE, and all the departments are children under that executive.

Step 1b. Define the security policy

The next step required to use LBAC security in the above example is to define the policy associated with the security label component above. A security policy can use more than one components.

```
 CREATE SECURITY POLICY J_DEPT_POLICY
COMPONENTS J_DEPT
WITH DB2LBACRULES
RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL
```

Step 1c. Create the security labels

The third step in setting up the security policy is to create the security labels. This is where you will specify the different roles that each user has. In this case, since the example is fairly simple, there will only be three labels, Executive, Manager and Team Lead.

```
CREATE SECURITY LABEL J_DEPT_POLICY.EXECUTIVE
       COMPONENT J_DEPT 'HR_EXECUTIVE'
```

```
        CREATE SECURITY LABEL J_DEPT_POLICY.MANAGE_D11_E21
               COMPONENT J_DEPT 'MAN_D11_E21'

        CREATE SECURITY LABEL J_DEPT_POLICY.A00
               COMPONENT J_DEPT 'A00'

        CREATE SECURITY LABEL J_DEPT_POLICY.B01
               COMPONENT J_DEPT 'B01'

        CREATE SECURITY LABEL J_DEPT_POLICY.C01
               COMPONENT J_DEPT 'C01'

        CREATE SECURITY LABEL J_DEPT_POLICY.D11
               COMPONENT J_DEPT 'D11'

        CREATE SECURITY LABEL J_DEPT_POLICY.D21
               COMPONENT J_DEPT 'D21'

        CREATE SECURITY LABEL J_DEPT_POLICY.E01
               COMPONENT J_DEPT 'E01'

        CREATE SECURITY LABEL J_DEPT_POLICY.E11
               COMPONENT J_DEPT 'E11'

        CREATE SECURITY LABEL J_DEPT_POLICY.E21
               COMPONENT J_DEPT 'E21'
```

In the next step you'll define the actual permissions associated with these labels.

Step 1d. Grant rights based on labels

The following steps outline the procedures for granting the rights to the table data. Rights are either ALL ACCESS, WRITE ACCESS or READ ACCESS. if none of these rights are granted to a user, then that user doesn't have the ability to access any of the table data. Remember that executives have full access, managers have full access to their departments and team leads have read access to members of the departments they lead.

```
db2 grant security label J_DEPT_POLICY.A00 to user Frank for read access
db2 grant security label J_DEPT_POLICY.MANAGE_D11_E21 to user Joe for all access
db2 grant security label J_DEPT_POLICY.EXECUTIVE to user Jane for all access
```

Setting the above labels on the users will cascade rights based on the tree definitions in step 1a. Because user Joe is labeled as MANAGE_D11_E21, and is given all rights, he will be able to read and write rows that have a security tag of J_DEPT_POLICY.D11 or J_DEPT_POLICY.E21 (since they are his children).

Step 2: Modify the EMP table

When modifying the EMP table, you must create an extra column to store the security label. This is of type "DB2SECURITYLABEL". You are going to modify the existing EMP table in the SAMPLE database. To do this, you must user a user that has been granted root level privilege in the policy, so in this case the user Jane. You must also first drop the MQT table ADEFUSR from the sample database.

```
CONNECT TO SAMPLE

   Database Connection Information

 Database server        = DB2/NT 9.1.0
 SQL authorization ID   = GMILNE
 Local database alias   = SAMPLE

DROP TABLE ADEFUSR

CONNECT RESET

CONNECT TO SAMPLE USER Jane USING password

ALTER TABLE EMP
                ADD COLUMN DEPT_TAG DB2SECURITYLABEL
                ADD SECURITY POLICY J_DEPT_POLICY
```

If you select from the EMP table, you will see the additional column defined. Because you performed the alter with a user defined on the EXECUTIVE level, all the security tags will have been added as EXECUTIVE. To change this, you need to update the table.

```
db2 => select EMPNO, FIRSTNME, LASTNAME, WORKDEPT, SALARY,
varchar(SECLABEL_TO_CHAR('J_DEPT_POLICY',DEPT_TAG),30) from gmilne.emp

EMPNO  FIRSTNME     LASTNAME        WORKDEPT SALARY      6
------ ------------ --------------- -------- ----------- ------------------------------
000010 CHRISTINE    HAAS            A00        152750.00 HR_EXECUTIVE
000020 MICHAEL      THOMPSON        B01         94250.00 HR_EXECUTIVE
000030 SALLY        KWAN            C01         98250.00 HR_EXECUTIVE
000050 JOHN         GEYER           E01         80175.00 HR_EXECUTIVE
000060 IRVING       STERN           D11         72250.00 HR_EXECUTIVE
000070 EVA          PULASKI         D21         96170.00 HR_EXECUTIVE
000090 EILEEN       HENDERSON       E11         89750.00 HR_EXECUTIVE
000100 THEODORE     SPENSER         E21         86150.00 HR_EXECUTIVE
000110 VINCENZO     LUCCHESSI       A00         66500.00 HR_EXECUTIVE
000120 SEAN         O'CONNELL       A00         49250.00 HR_EXECUTIVE
000130 DELORES      QUINTANA        C01         73800.00 HR_EXECUTIVE
000140 HEATHER      NICHOLLS        C01         68420.00 HR_EXECUTIVE
000150 BRUCE        ADAMSON         D11         55280.00 HR_EXECUTIVE
000160 ELIZABETH    PIANKA          D11         62250.00 HR_EXECUTIVE
000170 MASATOSHI    YOSHIMURA       D11         44680.00 HR_EXECUTIVE
000180 MARILYN      SCOUTTEN        D11         51340.00 HR_EXECUTIVE
000190 JAMES        WALKER          D11         50450.00 HR_EXECUTIVE
000200 DAVID        BROWN           D11         57740.00 HR_EXECUTIVE
000210 WILLIAM      JONES           D11         68270.00 HR_EXECUTIVE
000220 JENNIFER     LUTZ            D11         49840.00 HR_EXECUTIVE
000230 JAMES        JEFFERSON       D21         42180.00 HR_EXECUTIVE
000240 SALVATORE    MARINO          D21         48760.00 HR_EXECUTIVE
000250 DANIEL       SMITH           D21         49180.00 HR_EXECUTIVE
000260 SYBIL        JOHNSON         D21         47250.00 HR_EXECUTIVE
000270 MARIA        PEREZ           D21         37380.00 HR_EXECUTIVE
000280 ETHEL        SCHNEIDER       E11         36250.00 HR_EXECUTIVE
000290 JOHN         PARKER          E11         35340.00 HR_EXECUTIVE
000300 PHILIP       SMITH           E11         37750.00 HR_EXECUTIVE
000310 MAUDE        SETRIGHT        E11         35900.00 HR_EXECUTIVE
000320 RAMLAL       MEHTA           E21         39950.00 HR_EXECUTIVE
000330 WING         LEE             E21         45370.00 HR_EXECUTIVE
000340 JASON        GOUNOT          E21         43840.00 HR_EXECUTIVE
200010 DIAN         HEMMINGER       A00         46500.00 HR_EXECUTIVE
200120 GREG         ORLANDO         A00         39250.00 HR_EXECUTIVE
200140 KIM          NATZ            C01         68420.00 HR_EXECUTIVE
200170 KIYOSHI      YAMAMOTO        D11         64680.00 HR_EXECUTIVE
200220 REBA         JOHN            D11         69840.00 HR_EXECUTIVE
200240 ROBERT       MONTEVERDE      D21         37760.00 HR_EXECUTIVE
200280 EILEEN       SCHWARTZ        E11         46250.00 HR_EXECUTIVE
```

```
200310 MICHELLE     SPRINGER        E11         35900.00 HR_EXECUTIVE
200330 HELENA       WONG            E21         35370.00 HR_EXECUTIVE
200340 ROY          ALONZO          E21         31840.00 HR_EXECUTIVE

  42 record(s) selected.

update emp set DEPT_TAG=(SECLABEL_BY_NAME('J_DEPT_POLICY','A00')) where WORKDEPT='A00'

update emp set DEPT_TAG=(SECLABEL_BY_NAME('J_DEPT_POLICY','B01')) where WORKDEPT='B01'

update emp set DEPT_TAG=(SECLABEL_BY_NAME('J_DEPT_POLICY','C01')) where WORKDEPT='C01'

update emp set DEPT_TAG=(SECLABEL_BY_NAME('J_DEPT_POLICY','D11')) where WORKDEPT='D11'

update emp set DEPT_TAG=(SECLABEL_BY_NAME('J_DEPT_POLICY','D21')) where WORKDEPT='D21'

update emp set DEPT_TAG=(SECLABEL_BY_NAME('J_DEPT_POLICY','E01')) where WORKDEPT='E01'

update emp set DEPT_TAG=(SECLABEL_BY_NAME('J_DEPT_POLICY','E11')) where WORKDEPT='E11'

update emp set DEPT_TAG=(SECLABEL_BY_NAME('J_DEPT_POLICY','E21')) where WORKDEPT='E21'

db2 => select EMPNO, FIRSTNME, LASTNAME, WORKDEPT, SALARY,
varchar(SECLABEL_TO_CHAR('J_DEPT_POLICY',DEPT_TAG),30) from emp

EMPNO  FIRSTNME     LASTNAME        WORKDEPT SALARY      6
------ ------------ --------------- -------- ----------- ------------------------------
000010 CHRISTINE    HAAS            A00        152750.00 A00
000020 MICHAEL      THOMPSON        B01         94250.00 B01
000030 SALLY        KWAN            C01         98250.00 C01
000050 JOHN         GEYER           E01         80175.00 E01
000060 IRVING       STERN           D11         72250.00 D11
000070 EVA          PULASKI         D21         96170.00 D21
000090 EILEEN       HENDERSON       E11         89750.00 E11
000100 THEODORE     SPENSER         E21         86150.00 E21
000110 VINCENZO     LUCCHESSI       A00         66500.00 A00
000120 SEAN         O'CONNELL       A00         49250.00 A00
000130 DELORES      QUINTANA        C01         73800.00 C01
000140 HEATHER      NICHOLLS        C01         68420.00 C01
000150 BRUCE        ADAMSON         D11         55280.00 D11
000160 ELIZABETH    PIANKA          D11         62250.00 D11
000170 MASATOSHI    YOSHIMURA       D11         44680.00 D11
000180 MARILYN      SCOUTTEN        D11         51340.00 D11
000190 JAMES        WALKER          D11         50450.00 D11
000200 DAVID        BROWN           D11         57740.00 D11
000210 WILLIAM      JONES           D11         68270.00 D11
000220 JENNIFER     LUTZ            D11         49840.00 D11
000230 JAMES        JEFFERSON       D21         42180.00 D21
000240 SALVATORE    MARINO          D21         48760.00 D21
000250 DANIEL       SMITH           D21         49180.00 D21
000260 SYBIL        JOHNSON         D21         47250.00 D21
000270 MARIA        PEREZ           D21         37380.00 D21
000280 ETHEL        SCHNEIDER       E11         36250.00 E11
000290 JOHN         PARKER          E11         35340.00 E11
000300 PHILIP       SMITH           E11         37750.00 E11
000310 MAUDE        SETRIGHT        E11         35900.00 E11
000320 RAMLAL       MEHTA           E21         39950.00 E21
000330 WING         LEE             E21         45370.00 E21
000340 JASON        GOUNOT          E21         43840.00 E21
200010 DIAN         HEMMINGER       A00         46500.00 A00
200120 GREG         ORLANDO         A00         39250.00 A00
200140 KIM          NATZ            C01         68420.00 C01
200170 KIYOSHI      YAMAMOTO        D11         64680.00 D11
200220 REBA         JOHN            D11         69840.00 D11
200240 ROBERT       MONTEVERDE      D21         37760.00 D21
200280 EILEEN       SCHWARTZ        E11         46250.00 E11
200310 MICHELLE     SPRINGER        E11         35900.00 E11
200330 HELENA       WONG            E21         35370.00 E21
200340 ROY          ALONZO          E21         31840.00 E21

  42 record(s) selected.
```

After the update, lets see what the individual users can do. You'll connect to the database using the Executive userid Jane. Start with the same select statement performed before:

```
db2 => select EMPNO, FIRSTNME, LASTNAME, WORKDEPT, SALARY,
varchar(SECLABEL_TO_CHAR('J_DEPT_POLICY',DEPT_TAG),30) from gmilne.emp

EMPNO  FIRSTNME     LASTNAME        WORKDEPT SALARY      6
------ ------------ --------------- -------- ----------- ------------------------------
000010 CHRISTINE    HAAS            A00        152750.00 A00
000020 MICHAEL      THOMPSON        B01         94250.00 B01
000030 SALLY        KWAN            C01         98250.00 C01
000050 JOHN         GEYER           E01         80175.00 E01
000060 IRVING       STERN           D11         72250.00 D11
000070 EVA          PULASKI         D21         96170.00 D21
000090 EILEEN       HENDERSON       E11         89750.00 E11
000100 THEODORE     SPENSER         E21         86150.00 E21
000110 VINCENZO     LUCCHESSI       A00         66500.00 A00
000120 SEAN         O'CONNELL       A00         49250.00 A00
000130 DELORES      QUINTANA        C01         73800.00 C01
000140 HEATHER      NICHOLLS        C01         68420.00 C01
000150 BRUCE        ADAMSON         D11         55280.00 D11
000160 ELIZABETH    PIANKA          D11         62250.00 D11
000170 MASATOSHI    YOSHIMURA       D11         44680.00 D11
000180 MARILYN      SCOUTTEN        D11         51340.00 D11
000190 JAMES        WALKER          D11         50450.00 D11
000200 DAVID        BROWN           D11         57740.00 D11
000210 WILLIAM      JONES           D11         68270.00 D11
000220 JENNIFER     LUTZ            D11         49840.00 D11
000230 JAMES        JEFFERSON       D21         42180.00 D21
000240 SALVATORE    MARINO          D21         48760.00 D21
000250 DANIEL       SMITH           D21         49180.00 D21
000260 SYBIL        JOHNSON         D21         47250.00 D21
000270 MARIA        PEREZ           D21         37380.00 D21
000280 ETHEL        SCHNEIDER       E11         36250.00 E11
000290 JOHN         PARKER          E11         35340.00 E11
000300 PHILIP       SMITH           E11         37750.00 E11
000310 MAUDE        SETRIGHT        E11         35900.00 E11
000320 RAMLAL       MEHTA           E21         39950.00 E21
000330 WING         LEE             E21         45370.00 E21
000340 JASON        GOUNOT          E21         43840.00 E21
200010 DIAN         HEMMINGER       A00         46500.00 A00
200120 GREG         ORLANDO         A00         39250.00 A00
200140 KIM          NATZ            C01         68420.00 C01
200170 KIYOSHI      YAMAMOTO        D11         64680.00 D11
200220 REBA         JOHN            D11         69840.00 D11
200240 ROBERT       MONTEVERDE      D21         37760.00 D21
200280 EILEEN       SCHWARTZ        E11         46250.00 E11
200310 MICHELLE     SPRINGER        E11         35900.00 E11
200330 HELENA       WONG            E21         35370.00 E21
200340 ROY          ALONZO          E21         31840.00 E21

  42 record(s) selected.
```

And the update command:

```
db2 => update gmilne.emp set DEPT_TAG=(SECLABEL_BY_NAME('J_DEPT_POLICY','E01'))
where WORKDEPT='E01' DB20000I  The SQL command completed successfully.
```

As you can see, Jane has full access to all the data in the table. Now lets look at what Joe can see. First, the select again.

```
db2 => select EMPNO, FIRSTNME, LASTNAME, WORKDEPT, SALARY,
varchar(SECLABEL_TO_CHAR('J_DEPT_POLICY',DEPT_TAG),30) from gmilne.emp

EMPNO  FIRSTNME     LASTNAME        WORKDEPT SALARY      6
------ ------------ --------------- -------- ----------- ------------------------------
000060 IRVING       STERN           D11         72250.00 D11
000100 THEODORE     SPENSER         E21         86150.00 E21
000150 BRUCE        ADAMSON         D11         55280.00 D11
000160 ELIZABETH    PIANKA          D11         62250.00 D11
000170 MASATOSHI    YOSHIMURA       D11         44680.00 D11
000180 MARILYN      SCOUTTEN        D11         51340.00 D11
000190 JAMES        WALKER          D11         50450.00 D11
000200 DAVID        BROWN           D11         57740.00 D11
000210 WILLIAM      JONES           D11         68270.00 D11
000220 JENNIFER     LUTZ            D11         49840.00 D11
000320 RAMLAL       MEHTA           E21         39950.00 E21
000330 WING         LEE             E21         45370.00 E21
000340 JASON        GOUNOT          E21         43840.00 E21
200170 KIYOSHI      YAMAMOTO        D11         64680.00 D11
200220 REBA         JOHN            D11         69840.00 D11
200330 HELENA       WONG            E21         35370.00 E21
200340 ROY          ALONZO          E21         31840.00 E21

  17 record(s) selected.
```

See how he can only see information from departments D11 and E21? Lets see what happens when he tries to select data that is in the table, but he is not allowed to see:

```
db2 => select EMPNO, FIRSTNME, LASTNAME, WORKDEPT, SALARY,
varchar(SECLABEL_TO_CHAR('J_DEPT_POLICY',DEPT_TAG),30)
from gmilne.emp where empno='000130'

EMPNO  FIRSTNME     LASTNAME        WORKDEPT SALARY      6
------ ------------ --------------- -------- ----------- ------------------------------

  0 record(s) selected.
```

You know from the previous select with Jane that there is an employee in there with empno 000130, but Joe is not allowed to see it.

Now, one last test, with Frank.

First, the same select the other two users have run:

```
db2 => select EMPNO, FIRSTNME, LASTNAME, WORKDEPT, SALARY,
varchar(SECLABEL_TO_CHAR('J_DEPT_POLICY',DEPT_TAG),30) from gmilne.emp

EMPNO  FIRSTNME     LASTNAME        WORKDEPT SALARY      6
------ ------------ --------------- -------- ----------- ------------------------------
000010 CHRISTINE    HAAS            A00        152750.00 A00
000110 VINCENZO     LUCCHESSI       A00         66500.00 A00
000120 SEAN         O'CONNELL       A00         49250.00 A00
200010 DIAN         HEMMINGER       A00         46500.00 A00
200120 GREG         ORLANDO         A00         39250.00 A00

  5 record(s) selected.
```

In this case you can see that Frank can only see information about users from the department he leads. Lets see what happens when he tries to update:

```
db2 => update gmilne.emp set DEPT_TAG=(SECLABEL_BY_NAME('J_DEPT_POLICY','A00'))
where WORKDEPT='A00'DB21034E  The command was processed as an SQL statement
because it was not a valid Command Line Processor command.  During SQL processing it
returned:
SQL20402N Authorization ID "FRANK" does not have the LBAC credentials to
perform the "UPDATE" operation on table "EMPLOYEE".  SQLSTATE=42519
```

Even though he is trying to update a record that is in his own department, you created his access security to only allow read access to the table. Our business requirements have been satisfied.

Section 6. Summary

Now that you've completed this tutorial, you should have a fundamental understanding of the following topics:

Elements of a DB2 security plan: You should understand the structure of the entire DB2 environment, which includes client, servers, gateways, and hosts. You should also understand authentication, authorization, and privileges.

DB2 authentication types: You should know how to set authentication types using the `db2 update dbm cfg using authentication` *`type`* command on the server, and using the `db2 catalog database` command on the gateway and client.

DB2 authorities: You should understand the basics of the SYSADM, SYSCTRL, SYSMAINT, and SYSMON authorities, which are set in the DBM CFG file. You should also understand the basics of the DBADM, LOAD, and SECADM authorities, which are set using the `GRANT` command and revoked using the `REVOKE` command. Additionally, you should know what command each authority is allowed to run.

DB2 privileges: You should have an understanding of the different types of privileges and what they allow a user to do. Examples are CONTROL, INSERT, DELETE, CREATEIN, DROPIN, REFERENCES, and SELECT. You should also know how a privilege is obtained/revoked explicitly (`GRANT`/`REVOKE` commands), implicitly, or (for packages only) indirectly. In addition to this you should have a basic understanding of label-based access control, and how to define different types of policies based on this new security concept.

To access other tutorials in this series, bookmark the series page, DB2 9 DBA exam 731 prep tutorials.

Resources

Learn

- Check out the other parts of the DB2 9 Fundamentals exam 730 prep tutorial series.
- Certification exam site. Click the exam number to see more information about Exams 730 and 731.
- DB2 9 overview. Find information about the new data server that includes patented pureXML technology.
- The DB2 9 Information Center. Learn more about DB2 security.
- DB2 Label-Based Access Control, a practical guide, Part 1: Understand the basics of LBAC in DB2 (developerWorks) provides more information on label-based access control.
- DB2 UDB Security Part 2: Understand the DB2 Universal Database Security plug-ins (developerWorks, December 2005) outlines DB2's implementation of external security plug-ins, and how to implement them in your DB2 environment.

Get products and technologies

- DB2 9 is available as a free download.
- DB2 Express-C. Download a no-charge version of DB2 Express Edition for the community that offers the same core data features as DB2 Express Edition and provides a solid base to build and deploy applications.

About the author

Graham G. Milne
Graham Milne, HBSc. - Computer Science, is a DB2 Certified Advance Technical Expert and has been working with DB2 since 1998. Currently Graham is a Premium Support Manager for DB2 supporting large premium customers. Previous to this, he was the senior advanced service consultant for DB2 support based out of the IBM Toronto Software Lab.