This tutorial introduces the concepts of authentication, authorities, privileges, audit facility, trusted context, RCAC, and LBAC as they relate to DB2® 10. It is the seventh in a series of tutorials designed to help you prepare for the DB2 10.1 for Linux®, UNIX®, and Windows® Database Administration (exam 611). You should have basic knowledge of database concepts and operating system security.

Before you start

About this series

If you are preparing to take the DB2 10.1 for Linux, UNIX, and Windows DBA certification exam (exam 611), you've come to the right place — a study hall of sorts. This series of "DB2 10.1 DBA certification exam 611" tutorials covers the major concepts you'll need to know for the test.

About this tutorial

You'll learn about DB2 10.1 security features, including label-based access control, and the new row and column access control.

This is the seventh in a series of tutorials you can use to help prepare for the DB2 10.1 DBA for Linux, UNIX and Windows exam 611. The material here primarily covers the objectives in Section 7 of the test, titled "Security." You can view the objectives.

Prerequisites

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

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 10.1.
The examples in this tutorial are specific to DB2 10.1 running Linux with native security features. However, the concepts and information provided are relevant to DB2 running on any distributed platform.

**DB2 security**

**Aspects of database security**

Database security is of utmost importance today. Your database might allow customers to purchase products online, 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
- The Authority level a user is granted
- Commands a user is allowed to run
- Data 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 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 they have been assigned. DB2 has two groups of authorities: system-level authorities (or instance-level authorities) and database-level authorities.

*Privileges* are a bit more granular than authorities and can be assigned to users, groups, or roles. Privileges help define the objects a user can create or drop. They also define the commands a user can use to access objects like tables, views, indices, and packages. New to DB2 10.1 is the concept of row and column access control (RCAC), an additional layer of security that can be used to complement the table privileges model.

**DB2 authentication**

*DB2 authentication* controls the following aspects of a database security plan: who is allowed access to the instance or database, and where and how a user's password will be verified.
It does this with the help of the underlying OS 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.

**Introduction to DB2 authorities**

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

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

Authorities are made up of groups of privileges and higher-level database manager (instance-level) maintenance and utility operations.

Of the many authorities available in DB2, SYSADM, SYSCTRL, SYMAINT, and SYSMON are *instance-level authorities*. These authorities can only be assigned to a group; you can do so through the DBM CFG file.
Figure 1. Instance-level authorities

The SECADM, ACCESSCTRL, DATAACCESS, DBADB, SQLADM, WLMADM, and EXPLAIN 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 OS level.

### System-level authorities

**System-level (or instance-level) authorities** enable you to perform instance-wide functions, such as creating and upgrading databases, managing table spaces, and monitoring activity and performance on your instance. No instance-level authority provides access to data in database tables.

We will see the abilities given by each instance-level administrative authority.
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 DB2 commands against that instance, databases within the instance, and objects within those databases. Examples of commands a SYSADM user can perform against any database in the instance:

- Update and restore a database manager configuration file (DBM CFG)
- Grant and revoke table space privileges
- Update and restore a database

SYSADM users don't have the ability to access data within the tables unless they are granted the DATAACCESS authority, or privileges required to do so.

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:

```
update dbm cfg using SYSADM_GROUP db2grp1.
```

Remember that this change will not take effect until the instance is stopped and 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.

SYSCTRL authority

Users with SYSCTRL authority can perform all administrative and maintenance commands within the instance. Users possessing SYSCTRL authority cannot access any data stored in tables, unless they are granted the DATAACCESS authority, or privileges required to do so. Examples of commands a SYSCTRL user can perform against any database in the instance:

- Update a database node or distributed connection services (DCS)
- Restore to a new or existing database
- Force users off the system
- Create or drop a database
- Create, drop, or alter a table space
- Use any table space

Users with SYSADM authority can assign SYSCTRL to a group using the following command: `db2 update dbm cfg using SYSCTRL_GROUP group name`.

SYSMAINT authority

The commands that a user with SYSMAINT authority can issue are a subset of those allowed to users with SYSCTRL authority. Users possessing SYSMAINT authority cannot access any data
stored in tables, unless they are granted the DATAACCESS authority, or privileges required to do so. SYSMAINT users can only perform tasks related to maintenance, such as:

- Backup a database or table space
- Restore an existing database
- Roll forward recovery
- Start or stop an instance
- Restore or quiesce a table space, and query its state
- Run tracing
- Take database system monitor snapshots
- Reorganize tables
- Use RUNSTATS to update history log files

Notice that users with SYSMAINT cannot create or drop databases or table spaces.

If you have SYSADM authority, use the following command: `db2 update dbm cfg using SYSMAINT_GROUP group name`.

**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. Users possessing SYSMAINT cannot access any data stored in tables, unless they are granted the DATAACCESS authority, or privileges required to do so.

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 and LIST DCS APPLICATIONS`
- `LIST PACKAGES, LIST TABLES, LIST TABLESPACES`
- `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
- `db2mrtk` — Memory tracker
- `db2ResetMonitor` — Reset monitor

SYSMON authority enables users 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`.

**Database-level administrative authorities**

*Database-level administrative authorities* provide control within a specific database in the instance. We will see the abilities given by each database-level administrative authority.

**SECADM authority**

SECADM authority is able to manage database security objects (such as database roles, audit policies, trusted contexts, and security label components), and grant and revoke all database privileges and authorities. Users possessing SECADM authority cannot access any data stored in tables, unless they are granted the DATAACCESS authority or privileges required to do so. The SECADM authority can be granted to a user, a role, or a group, but not to PUBLIC.

A user with SECADM can perform the following:

- Create, alter, drop, and comment on security label objects
- Create and drop security policies and security labels
- Grant and revoke all database authorities and privileges
- Create and drop column masks and row permissions
- Activate and deactivate row and column access control feature
- Use and manage AUDIT features

A user with SECADM authority can assign SECADM authority to another user using the following command: `db2 grant SECADM on database to user username`. ACCESSCTRL authority and DATAACCESS authority are subsets of the SECADM authority.

**ACCESSCTRL authority**

ACCESSCTRL is the authority required to grant and revoke privileges on objects within a specific database. Like SECADM authority, users possessing ACCESSCTRL authority cannot access any data stored in the database (except catalog tables and views), unless they are granted the DATAACCESS authority, or privileges required to do so. The ACCESSCTRL authority can be granted to a user, a role, or a group, but not to PUBLIC.

A user with ACCESSCTRL can perform the following:

- Query system catalog tables and views
- Grant and revoke SQLADM, WLMADM, EXPLAIN, BINDADD, CONNECT, QUIESCE_CONNECT, CREATETAB, CREATE_EXTERNAL_ROUTINE, CREATE_NOT_FENCED_ROUTINE, IMPLICIT_SCHEMA, and LOAD authorities
• Grant and revoke all privileges on global variables, indices, nicknames, packages, routines (except system-defined audit routines), schemas, sequences, servers, tables, table spaces, views, and XSR objects

A user with SECADM authority can assign ACCESSCTRL authority to another user using the following command: `db2 grant ACCESSCTRL on database to user username`.

**DATAACCESS authority**

DATAACCESS authority allows access to data within a specific database. The DATAACCESS authority can be granted to a user, a role, or a group, but not to PUBLIC.

For all tables, views, materialized query tables, and nicknames, a user with ACCESSCTRL possesses the following authorities and privileges:

- LOAD authority on the database
- SELECT privilege (including system catalog tables and views)
- INSERT privilege
- UPDATE privilege
- DELETE privilege

In addition, DATAACCESS authority provides the following privileges:

- EXECUTE on all packages, on all routines (except audit routines), and on all modules
- READ on all global variables and WRITE on all global variables except read-only variables
- USAGE on all XSR objects and all sequences

A user with SECADM authority can assign DATAACCESS authority to another user using the following command: `db2 grant DATAACCESS on database to user username`.

**DBADM authority**

DBADM authority provides complete control over a database. Among other things, the database administrator possesses the privileges required to create objects and issue database commands. DBADM authority by itself cannot access any data stored in the database, unless they are granted DATAACCESS authority, or privileges required to do so. This authority can be granted to a user, a role, or a group, but not to PUBLIC.

**NOTE:** When a SECADM grants DBADM on a database to a user, the user implicitly receives DATAACCESS authority on the database, unless the SECADM uses the `WITHOUT DATA ACCESS` tag when granting the DBADM authority.

A user with DBADM can perform such maintenance or administrative tasks as:

- Create, alter, and drop non-security related database objects
- Read log files
- Create, activate, and drop event monitors
- Query the state of a table space
- Update log history files
- Quiesce a table space
• Reorganize a table
• Collect catalog statistics using the RUNSTATS utility

A user with SECADM authority can assign DBADM authority to another user using the following command: `db2 grant DBADM on database to user username`.

SQLADM authority, EXPLAIN authority, and WLMADM authority are subsets of the DBADM authority. This authority can be granted to a user, a role, or a group, including PUBLIC.

SQLADM authority
SQLADM authority provides the ability to monitor and tune SQL statements. This authority can be granted to a user, a role, or a group, including PUBLIC.

A user with SQLADM can perform such maintenance or administrative tasks as:

• Creating, dropping, and flushing an event monitor
• Setting an event monitor state
• Using EXPLAIN to capture access plan information
• Reorganizing tables and indices
• Executing RUNSTATS

A user with SECADM authority can assign SECADM authority to another user using the following command: `db2 grant SECADM on database to user username`.

EXPLAIN authority
The EXPLAIN authority provides the ability to explain query plans without gaining access to data for a specific database. This authority can be granted to a user, a role, or a group, including PUBLIC.

A user with EXPLAIN can perform such maintenance or administrative tasks as:

• Use EXPLAIN to capture access plan information
• Use PREPARE to dynamically prepare an SQL statement for execution
• Use DESCRIBE on output of a SELECT statement or of an XQuery statement

The EXPLAIN authority also provides EXECUTE privilege on the built-in explain routines.

A user with SECADM authority can assign EXPLAIN authority to another user using the following command: `db2 grant EXPLAIN on database to user username`.

WLMADM authority
The WLDMADM authority provides the ability to manage workload objects on a specific database. This authority can be granted to a user, a role, or a group, including PUBLIC.

A user with WLMADM can perform such maintenance or administrative tasks as:

• Create, alter, comment on, and drop the following workload manager objects:
  • Histogram templates
• Service classes
• Thresholds
• Work action sets
• Work class sets
• Workloads
• Grant and revoke workload privileges
• Execute the built-in workload management routines

There are other database authorities capable of performing some particular type of action on the database.

• `CONNECT` allows the holder to connect to the database.
• `QUIESCE_CONNECT` allows the holder to access the database while it is quiesced.
• `IMPLICIT_SCHEMA` allows any user to create a schema implicitly.
• `CREATE_EXTERNAL_ROUTINE` allows the holder to create a procedure for use by applications and other users.
• `CREATE_NOT_FENCED_ROUTINE` allows the holder to create a user-defined function or procedure that is not fenced.
• `CREATETAB` allows the holder to create new tables in the database.
• `BINDADD` allows the holder to create new packages in the database.
• `LOAD` allows the holder to use the LOAD utility.

**DB2 privileges**

**Database and object privileges**

In the preceding sections, 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, indices, schemas, and packages. Fortunately, most of the object-level privileges are self-explanatory. The following table summarizes these privileges.

**Table 1. Summary of privileges**

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Relevant objects</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CREATETAB</code></td>
<td></td>
<td>Users can create tables within the database.</td>
</tr>
<tr>
<td><code>BINDADD</code></td>
<td></td>
<td>Users can create packages in the database using the <code>BIND</code> command.</td>
</tr>
<tr>
<td><code>CONNECT</code></td>
<td></td>
<td>Users can connect to the database.</td>
</tr>
<tr>
<td><code>CREATE_NOT_FENCED</code></td>
<td></td>
<td>Users can create unfenced user-defined functions (UDFs).</td>
</tr>
<tr>
<td><code>IMPLICIT_SCHEMA</code></td>
<td></td>
<td>Users can implicitly create schemas within the database without using the CREATE SCHEMA command.</td>
</tr>
<tr>
<td><code>LOAD</code></td>
<td></td>
<td>Users can load data into a table.</td>
</tr>
<tr>
<td><code>QUIESCE_CONNECT</code></td>
<td></td>
<td>Users can access a database while it is in a quiesced state.</td>
</tr>
<tr>
<td><code>CREATE_EXTERNAL_ROUTINE</code></td>
<td></td>
<td>Users can create a procedure for use by applications and other users of the database.</td>
</tr>
<tr>
<td>PRIVILEGE</td>
<td>OBJECT</td>
<td>DESCRIPITION</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CONTROL</td>
<td>Table, View, Index, Package, Alias, Distinct Type, User Defined function, Sequence</td>
<td>Provides full authority on the object. Users with this privilege can also grant or revoke privileges on the object to other users.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Table, View</td>
<td>Allows users to delete records from the object.</td>
</tr>
<tr>
<td>INSERT</td>
<td>Table, View</td>
<td>Allows users to insert records into the object via the INSERT or IMPORT commands.</td>
</tr>
<tr>
<td>SELECT</td>
<td>Table, View</td>
<td>Provides the ability to view the contents of the object using the SELECT statement.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>Table, View</td>
<td>Allows users to modify records within the object using the UPDATE statement.</td>
</tr>
<tr>
<td>ALTER</td>
<td>Table</td>
<td>Allows users to alter the object definition using the ALTER statement.</td>
</tr>
<tr>
<td>INDEX</td>
<td>Table</td>
<td>Allows users to create indices on the object using the create INDEX statement.</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>Table</td>
<td>Provides the ability to create or drop foreign key constraints on the object.</td>
</tr>
<tr>
<td>BIND</td>
<td>Package</td>
<td>Allows users to rebind existing packages.</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>Package, Procedure, Function, Method</td>
<td>Allows users to execute packages and routines.</td>
</tr>
<tr>
<td>ALTERIN</td>
<td>Schema</td>
<td>Allows users to modify definitions of objects within the schema.</td>
</tr>
<tr>
<td>CREATEIN</td>
<td>Schema</td>
<td>Allows users to create objects within the schema.</td>
</tr>
<tr>
<td>DROPIN</td>
<td>Schema</td>
<td>Allows users to drop objects within the schema.</td>
</tr>
</tbody>
</table>

Information about 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:

1. Grant SELECT privilege on CUSTOMERS table to user user1: `db2 grant select on table CUSTOMERS to user user1`
2. Grant SELECT, INSERT, and UPDATE privileges on CUSTOMERS table to user user2: `db2 grant select, insert, update on table CUSTOMERS to user user1`
3. Grant all the appropriate privileges, except CONTROL, on CUSTOMERS table to user user3: `db2 grant all on table CUSTOMERS to user user3`
4. Grant all the appropriate privileges, except CONTROL, on CUSTOMERS table to user user4 and the ability to extend all privileges to others: `db2 grant all on table CUSTOMERS to user user4 WITH GRANT OPTION`

Audit facility

DB2 includes an audit facility that allows you to monitor data access and provides information needed for subsequent analysis. Auditing can help discover unwanted, unknown, and
acceptable access to the data as well as keep history records of the activities on the database system.

The audit facility provides the ability to audit at both the instance and the individual database level, independently recording all instance and database level activities with separate logs for each. The system administrator (who holds SYSADM authority) can use the db2audit tool to configure audit at the instance level as well as to control when such audit information is collected. The system administrator can use the db2audit tool to archive both instance and database audit logs as well as to extract audit data from archived logs of either type.

The security administrator (who holds SECADM authority within a database) can create audit policies to control what is audited within an individual database. The following objects can have an audit policy associated with them:

- The whole database
- Tables
- Trusted contexts
- Authorization IDs representing users, groups, or roles
- Authorities (SYSADM, SECADM, DBADM, SQLADM, WLMADM, ACCESSCTRL, DATAACCESS, SYSCTRL, SYSMaint, SYSMON)

The security administrator can use audit policies in conjunction with the SQL statement, AUDIT, to configure and control the audit requirements for an individual database. The security administrator can use the following audit routines to perform the specified tasks:

- The SYSPROC.AUDIT_ARCHIVE stored procedure archives audit logs.
- The SYSPROC.AUDIT_LIST_LOGS table function allows you to locate logs of interest.
- The SYSPROC.AUDIT_DELIM_EXTRACT stored procedure extracts data into delimited files for analysis.

The security administrator can grant EXECUTE privilege on these routines to another user, enabling the security administrator to delegate these tasks, if required.

**Trusted context**

A **trusted context** is a database object that contains the connection attributes used to define a trusted relationship between the database and an external entity, such as an application server. The trusted relationship is based off on the following attributes:

- System authorization ID — Represents the user that establishes a database connection
- IP address (or domain name) — Represents the host from which a database connection is established
- Data stream encryption — Represents the encryption setting (if any) for the data communication between the database server and the database client

When a user establishes a database connection, DB2 checks whether the connection attributes match the trusted context requirements in the database. When a match occurs, the database connection is said to be trusted.
Trusted context allows connections to be shared among users without the needing to authenticate IDs. In addition, data access control is enforced by allowing the initiator to acquire additional capabilities that may not be available outside the scope of the trusted connection.

The additional capabilities might be limited depending on the type of trusted connection being requested. If an explicit trusted connection is established, the initiator is able to switch the user ID on the connection to a different user ID with or without authentication. In addition, the initiator of an explicit trusted connection may also acquire additional privileges via role inheritance. On the other hand, an implicit trusted connection results from a regular connection request, so it can only acquire additional privileges via role inheritance. They cannot switch the user ID.

As an example, in a tree-tier environment, the user establishes a connection with the middle tier, then the middle tier can then establish an explicit trusted connection to the database, giving the middle tier the ability to switch the current user ID on the connection to a different user ID, with or without authentication.

Suppose that the security administrator creates the following trusted context object.

**Listing 1. Trusted context object syntax**

```
CREATE TRUSTED CONTEXT CTX1 BASED UPON CONNECTION USING SYSTEM AUTHID USER2 ATTRIBUTES (ADDRESS '192.0.2.1')
DEFAULT ROLE managerRole ENABLE
```

If user `user1` requests a trusted connection from IP address 192.0.2.1, DB2 returns a warning indicating that a trusted connection could not be established. Consequently, user `user1` simply got a non-trusted connection.

If user `user2` requests a trusted connection from IP address 192.0.2.1, the request is granted because the connection attributes matches the trusted context CTX1. As soon as user `user2` establishes a trusted connection, user2 can now acquire all the privileges and authorities associated with the trusted context role managerRole. Note that these privileges and authorities may not be available to user `user2` outside the scope of this trusted connection.

**Row and column access control**

In order to comply with various government regulations, companies need to implement rules to ensure that information is adequately secured. As a result, individuals should only be allowed to access the subset of data needed to perform their tasks.

DB2 10 introduces a new feature called **row and column access control (RCAC)**. RCAC acts as an additional layer of data security by controlling access to a table at the row level, column level, or both, complementing the table privileges model.

RCAC has several advantages over existing mechanisms. For example, no database user is exempted from RCAC rules. This means that even higher-level authorities, such as users with DATAACCESS authority, are not exempt. RCAC also ensures that table data is protected regardless of how that table is accessed. Regardless of whether the data is accessed via direct
SQL, application logic, ad-hoc query tools, report generation tools, or even browsing through IBM Data Studio, all users are subject to RCAC rules. A further advantage of the RCAC solution is that no application changes are required to take full advantage of this additional layer of data security. Consequently, the feature is transparent to existing applications.

The RCAC feature represents an important shift in paradigm: It is no longer about what data is being asked but rather who is asking for the data. Depending on who requested data, result sets for the same query can differ immensely.

**RCAC rules**

RCAC comprises two sets of rules: row permissions and column masks.

**Row permissions** are defined by an SQL search condition used to define what set of rows a user has access to. If the reference of the table is for a fetch operation such as `SELECT`, the application of the row permission determines what set of rows can be retrieved by the user who requested the fetch operation. If the reference of the table is for a data change operation such as `INSERT`, the application of the row permission determines whether all rows to be changed can be inserted or updated by the user who requested the data change operation.

Column masks are defined by an SQL `CASE` expression that describes what column values a user is permitted to see and under what conditions.

While multiple columns in a table may have column masks, only one column mask can be created for a single column. When column access control is activated for the table, the `CASE` expression in the column mask definition is applied to an output column to determine the masked values that are returned to an application. The application of column masks affects the final output only; it does not impact the operations, such as predicated and ordering, in an SQL statement.

**Implementing row permissions**

There are several steps involved in creating row permissions. Row permission must be defined via `CREATE PERMISSION` SQL statement. Suppose a bank stores its customers' information in the CUSTOMERS table. By default, the managers (Eric and Pamela) can see all users' information as shown below.

<table>
<thead>
<tr>
<th>SSN</th>
<th>NAME</th>
<th>ADDRESS</th>
<th>PHONE</th>
<th>CARD_NUMBER</th>
<th>BALANCE</th>
<th>MGR_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-55-1234</td>
<td>Max</td>
<td>21 John Street</td>
<td>905-413-2311</td>
<td>1234-5678-9012-3456</td>
<td>89.70</td>
<td>ERIC</td>
</tr>
<tr>
<td>123-58-9812</td>
<td>Mike</td>
<td>98 Yonge Street</td>
<td>905-232-5633</td>
<td>3459-5344-6452-1234</td>
<td>8.30</td>
<td>PAMELA</td>
</tr>
<tr>
<td>123-11-9856</td>
<td>Sam</td>
<td>50 Rogers Street</td>
<td>905-413-2457</td>
<td>4335-5633-5673-9634</td>
<td>40.00</td>
<td>ERIC</td>
</tr>
<tr>
<td>123-19-1454</td>
<td>Dug</td>
<td>76 Dallas Street</td>
<td>905-331-1445</td>
<td>7647-9246-2345-3452</td>
<td>76.00</td>
<td>PAMELA</td>
</tr>
</tbody>
</table>

The CEO defined that the managers in a bank can only access customers for which they are responsible. All managers are members in role MANAGER.

The SECADM can use the `CREATE PERMISSION` statement as shown below to ensure that each manager can only access his own customers.

<table>
<thead>
<tr>
<th>SSN</th>
<th>NAME</th>
<th>ADDRESS</th>
<th>PHONE</th>
<th>CARD_NUMBER</th>
<th>BALANCE</th>
<th>MGR_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-55-1234</td>
<td>Max</td>
<td>21 John Street</td>
<td>905-413-2311</td>
<td>1234-5678-9012-3456</td>
<td>89.70</td>
<td>ERIC</td>
</tr>
<tr>
<td>123-58-9812</td>
<td>Mike</td>
<td>98 Yonge Street</td>
<td>905-232-5633</td>
<td>3459-5344-6452-1234</td>
<td>8.30</td>
<td>PAMELA</td>
</tr>
<tr>
<td>123-11-9856</td>
<td>Sam</td>
<td>50 Rogers Street</td>
<td>905-413-2457</td>
<td>4335-5633-5673-9634</td>
<td>40.00</td>
<td>ERIC</td>
</tr>
<tr>
<td>123-19-1454</td>
<td>Dug</td>
<td>76 Dallas Street</td>
<td>905-331-1445</td>
<td>7647-9246-2345-3452</td>
<td>76.00</td>
<td>PAMELA</td>
</tr>
</tbody>
</table>

DB2 10.1 DBA for Linux, UNIX, and Windows certification exam

611 prep, Part 7: Security
Listing 3. CREATE PERMISSION statement syntax

CREATE PERMISSION MANAGER_ROW_ACCESS ON CUSTOMERS FOR ROWS WHERE VERIFY_ROLE_FOR_USER(SESSION_USER,'MANAGER') = 1 AND CUSTOMER.MGR_ID = SESSION_USER) ENFORCED FOR ALL ACCESS ENABLE;

Finally, the row access control feature must be enabled: ALTER TABLE CUSTOMERS ACTIVATE ROW ACCESS CONTROL;

After creating the row permission and activating the row access control feature, each manager belonging to the MANAGER role is permitted to work with the data of the customers under his responsibility. If Pamela queries the CUSTOMERS table, she will be able to view the data of the customers under her responsibility.

Listing 4. Pamela's new view on the CUSTOMERS table

<table>
<thead>
<tr>
<th>SSN</th>
<th>NAME</th>
<th>ADDRESS</th>
<th>PHONE</th>
<th>CARD_NUMBER</th>
<th>BALANCE</th>
<th>MGR_ID</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>123-58-9812</td>
<td>Mike</td>
<td>98 Yonge Street</td>
<td>905-232-5633</td>
<td>3459-5344-6452-1234</td>
<td>8.30 PAMELA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>123-19-1454</td>
<td>Dug</td>
<td>76 Dallas Street</td>
<td>905-331-1445</td>
<td>7647-9246-2345-3452</td>
<td>76.00 PAMELA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Using the same rule, Pamela can only update the customers under her responsibility.

Implementing column masks

As mentioned, column masks are defined by an SQL CASE expression that describes what column values a user is permitted to see and under what conditions. Only an authorization ID or role with the SECADM or SYSADM authority can manage column masks.

Continuing the previous example, the security administrator can protect the customers' balance information to protect the BALANCE column, so that it can be accessed by users in the ACCOUNTING role.

Listing 5. CREATE (column) MASK statement syntax

CREATE MASK BALANCE_MASK ON CUSTOMERS FOR COLUMN BALANCE RETURN CASE WHEN VERIFY_ROLE_FOR_USER(SESSION_USER,'ACCOUNTING') = 1 THEN BALANCE ELSE 0.00 END ENABLE;

And the column access control feature must be enabled: ALTER TABLE CUSTOMERS ACTIVATE COLUMN ACCESS CONTROL;

After creating the column mask and activating the row access control feature, each accountant belonging to the ACCOUNTING role is permitted to visualize the customers' balance. If Pamela queries the CUSTOMERS table, she won't be able to view the customers' balance.

Listing 6. Pamela's new view on the CUSTOMERS table

<table>
<thead>
<tr>
<th>SSN</th>
<th>NAME</th>
<th>ADDRESS</th>
<th>PHONE</th>
<th>CARD_NUMBER</th>
<th>BALANCE</th>
<th>MGR_ID</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>123-58-9812</td>
<td>Mike</td>
<td>98 Yonge Street</td>
<td>905-232-5633</td>
<td>3459-5344-6452-1234</td>
<td>0.00 PAMELA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>123-19-1454</td>
<td>Dug</td>
<td>76 Dallas Street</td>
<td>905-331-1445</td>
<td>7647-9246-2345-3452</td>
<td>0.00 PAMELA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As expected, PAMELA is not able to retrieve customer balance information. She simply sees a balance of 0.00.
DB2 RCAC is a powerful feature that can help your business secure critical information. By acting as an additional layer of data security, it can allow access control at the row level, column level, or both without the need to modify your existing application.

**Label-based access control**

Since DB2 9, *label-based access control (LBAC)* has provided the ability to granularly control who can access your data. LBAC lets you decide exactly who has write access, and who has read access to individual rows and individual columns of a table.

LBAC is similar to RCAC except it uses security labels attached to table objects to control who has access. Users attempting to access an object must have its security label granted to them. When there's a match, access is permitted. Without a match, access is denied and data is hidden. Unlike RCAC, LBAC is targeted for applications where control must be structured around a rigid hierarchical representation of a business entity, such as government security clearance levels.

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

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. Let's assume that in your organization there are three sets of people.

<table>
<thead>
<tr>
<th>Name</th>
<th>Organizational role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jane</td>
<td>Human resources executive</td>
</tr>
<tr>
<td>Joe</td>
<td>Manager of Department D11 and E21</td>
</tr>
<tr>
<td>Frank</td>
<td>Team lead — Department A00</td>
</tr>
</tbody>
</table>

Now, in the organization's database there is a table that defines employee information. This will be based on the EMP table in the SAMPLE database. It contains data on employees and the departments they belong to. Its existing definition is shown below.

### Listing 7. EMP table definition

```
db2 => describe select * from emp SQLDA Information sqldaid : SQLDA sqldabc: 896 sqln: 20
sqlid: 14 Column Information sqltype sqlllen sqlname.data sqlname.length
------------------------------------ -----
               CHARACTER 6 EMPNO 5 448 VARCHAR 12 FIRSTNAME 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 regularly audited. Part of this audit indicates that the employees should not have access to data 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.

Set up LBAC security to enable these rules:

1. Define the security policies and labels, and grant the security labels to the users.
2. Modify the EMP table, including the security label column, and attach the security policy to it.

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

**Listing 8. Create the security label component syntax**

```
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 component.

**Listing 9. Create security policy syntax**

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

**Step 1c. Create security labels**
The third step in setting up the security policy is to create the security labels. This is where you will specify the roles each user has. In this case, since the example is fairly simple, there will only be three labels: Executive, Manager, and Team Lead.

**Listing 10. Create security labels syntax**

```
```
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, that user doesn't have the capability 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.

**Listing 11. Grant security labels syntax**

```sql
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. You must use a user that has been granted root level privilege in the policy — in this case, the user Jane. You must also drop the MQT table ADEFUSR from the sample database.

**Listing 12. Adding DB2SECURITYLABEL to table**

```sql
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 changes 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.
Listing 13. View newly added security tags on EMP table

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

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>FIRSTNME</th>
<th>LASTNAME</th>
<th>WORKDEPT</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>000010</td>
<td>CHRISTINE</td>
<td>HAAS</td>
<td>A00</td>
<td>152750.00</td>
</tr>
<tr>
<td>000020</td>
<td>MICHAEL</td>
<td>THOMPSON</td>
<td>B01</td>
<td>94250.00</td>
</tr>
<tr>
<td>000030</td>
<td>JOHN</td>
<td>GEYER</td>
<td>E01</td>
<td>80175.00</td>
</tr>
<tr>
<td>000040</td>
<td>IRVING</td>
<td>STERN</td>
<td>D11</td>
<td>72250.00</td>
</tr>
<tr>
<td>000050</td>
<td>EILEEN</td>
<td>HENDERSON</td>
<td>E11</td>
<td>89750.00</td>
</tr>
<tr>
<td>000060</td>
<td>SEAN</td>
<td>O'CONNELL</td>
<td>A00</td>
<td>49250.00</td>
</tr>
<tr>
<td>000070</td>
<td>CHRISTINE</td>
<td>HAAS</td>
<td>A00</td>
<td>152750.00</td>
</tr>
<tr>
<td>000080</td>
<td>MICHAEL</td>
<td>THOMPSON</td>
<td>B01</td>
<td>94250.00</td>
</tr>
<tr>
<td>000090</td>
<td>JOHN</td>
<td>GEYER</td>
<td>E01</td>
<td>80175.00</td>
</tr>
<tr>
<td>000100</td>
<td>IRVING</td>
<td>STERN</td>
<td>D11</td>
<td>72250.00</td>
</tr>
<tr>
<td>000110</td>
<td>EILEEN</td>
<td>HENDERSON</td>
<td>E11</td>
<td>89750.00</td>
</tr>
<tr>
<td>000120</td>
<td>SEAN</td>
<td>O'CONNELL</td>
<td>A00</td>
<td>49250.00</td>
</tr>
<tr>
<td>000130</td>
<td>CHRISTINE</td>
<td>HARRIS</td>
<td>A00</td>
<td>152750.00</td>
</tr>
<tr>
<td>000140</td>
<td>MICHAEL</td>
<td>THOMPSON</td>
<td>B01</td>
<td>94250.00</td>
</tr>
<tr>
<td>000150</td>
<td>JOHN</td>
<td>GEYER</td>
<td>E01</td>
<td>80175.00</td>
</tr>
<tr>
<td>000160</td>
<td>IRVING</td>
<td>STERN</td>
<td>D11</td>
<td>72250.00</td>
</tr>
<tr>
<td>000170</td>
<td>EILEEN</td>
<td>HENDERSON</td>
<td>E11</td>
<td>89750.00</td>
</tr>
<tr>
<td>000180</td>
<td>SEAN</td>
<td>O'CONNELL</td>
<td>A00</td>
<td>49250.00</td>
</tr>
<tr>
<td>000190</td>
<td>JOHN</td>
<td>GEYER</td>
<td>E01</td>
<td>80175.00</td>
</tr>
<tr>
<td>000200</td>
<td>IRVING</td>
<td>STERN</td>
<td>D11</td>
<td>72250.00</td>
</tr>
<tr>
<td>000210</td>
<td>EILEEN</td>
<td>HENDERSON</td>
<td>E11</td>
<td>89750.00</td>
</tr>
</tbody>
</table>

Listing 14. Update security tags on EMP table

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','D01')) where WORKDEPT='D01'
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'
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','D01')) where WORKDEPT='D01'
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'

After the update, let's see what the individual users can do. You'll connect to the database using the Executive user ID Jane. Start with the same select statement performed before.
Listing 15. View recently updated security tags on EMP 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
------------------------------ 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 LUCCHESI
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 58450.00 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 49100.00 D21 000260 SYBILL JOHNSON D21 47250.00 D21 000270 MARIA PEREZ D21 37390.00 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 200100 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
17 record(s) selected.
```
And the update command:
```
db2 => update gmilne.emp set DEPT_TAG=(SECLABEL_BY_NAME('J_DEPT_POLICY','E01')) where WORKDEPT='E01'
```

The SQL command completed successfully.

As you can see, Jane has full access to all the data in the table. Now let's look at what Joe can see. Look at the select command again.

Listing 16. Joe's view on the EMP table

```
select EMPNO, FIRSTNME, LASTNAME, WORKDEPT, SALARY, 
varchar(SECLABEL_TO_CHAR('J_DEPT_POLICY',DEPT_TAG),30) 
from gmilne.emp 
  EMPNO FIRSTNME LASTNAME WORKDEPT SALARY
------------------------------ 000101
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 58450.00 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 49100.00 D21 000260 SYBILL JOHNSON D21 47250.00 D21 000270 MARIA PEREZ D21 37390.00 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 200100 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
17 record(s) selected.
```

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

Listing 17. Joe's attempt to access data he is not allowed to

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

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. The same select the other two users have run.
Listing 18. Franks's view on the EMP table

```
db2 =\gt; select EMPNO, FIRSTNAME, LASTNAME, WORKDEPT, SALARY,
   varchar(SECLABEL_TO_CHAR('J_DEPT_POLICY',DEPT_TAG),30) from gmilne.emp

        EMPNO  FIRSTNAME      LASTNAME   WORKDEPT  SALARY     DEPT_TAG
          ------  -------------  ----------  ---------  --------
          000010  CHRISTINE      HAAS      A00     152750.00  A00
          000010  VINCENZO       LUCCHESSI A00      66500.00  A00
          000010  SEAN           O'CONNELL A00  49250.00  A00
          000010  LAVERNE        CASPER OFFICE A00  200010  A00
          000010  DIAN HEMMINGER A00      46500.00  A00
          000010  GREG ORLANDO OFFICE A00  500120  A00
          000010  GREG ORLANDO OFFICE A00  200120  A00
          000010  GREG ORLANDO OFFICE A00  39250.00  A00
          000010  GREG ORLANDO OFFICE A00  200120  A00
          000010  GREG ORLANDO OFFICE A00  39250.00  A00
          000010  GREG ORLANDO OFFICE A00  200120  A00

5 record(s) selected.
```

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

```
db2 =\gt; update gmilne.emp set DEPT_TAG=(SECLABEL_BY_NAME('J_DEPT_POLICY','A00')) where WORKDEPT='A00'
```

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 in his own department, you created his access security to only allow read access to the table. Our business requirements have been satisfied.

**Summary**

You should have a fundamental understanding of the following topics:

- DB2 authorizations and privileges (including data access/access control)
- The security administrator role
- The audit facility
- Trusted contexts
- Row and column access control (fine-grain access control) (hidden columns)
- Label-based access control
Related topics

- Read the Preparation Guide for DB2 10.1 DBA for Linux, UNIX, and Windows Exam 611 to learn in-depth information about each of the concepts presented in this tutorial. This guide is compilation of topics from the DB2 10.1 documentation.
- Use the DB2 documentation in IBM Knowledge Center to find more details about each of the concepts presented in this tutorial.
- Learn more about Exam 611.
- You can learn more about monitoring DB2 from the DB2 10.1 Information Center.
- Use an RSS feed to request notification for the upcoming tutorials in this series. (Find out more about RSS feeds of developerWorks content.)
- Attend or download the material for Information Management bootcamps.
- Now you can use DB2 for free. Download DB2 Express-C, 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.

© Copyright IBM Corporation 2012

Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)