IBM PRESS

DB2 Developer's Guide

A Solutions-Oriented Approach to Learning the Foundation and Capabilities of DB2 for z/OS

Sixth Edition

Craig S. Mullins

Common DB2 SQLCODE Values

SQLCODE	SQLSTATE	Description
+000	00000	The SQL statement finished successfully.
+100	02000	No rows found to satisfy the SQL statement.
+117	01525	Number of values being inserted does not equal number of columns in the table.
-101	54001	SQL statement is too complex.
-104	42601	Illegal symbol encountered in SQL statement. Usually, this means you have a syntax error somewhere in your SQL statement.
-122	42803	Column function used illegally; all columns not applied to the column function must be in the GROUP BY.
-150	42807	Invalid view UPDATE requested; or an invalid INSERT, UPDATE, or DELETE was requested on a transition table during a triggered action.
-305	22002	A null was returned but no indicator variable is available to assign null to the host variable.
-501	24501	Must open a cursor before attempting to fetch from it or close it.
-502	24502	Cannot open a cursor twice without first closing it.
-510	42828	The table specified by the cursor of the UPDATE or DELETE statement cannot be modified as requested.
-530	23503	Invalid foreign key value supplied for the specified constraint name.
-532	23504	Deletion violates the named referential constraint.
-545	23513	INSERT or UPDATE caused a check constraint violation.
-552	42502	User is attempting to perform an operation for which he or she is not authorized.
-803	23505	Insert violates uniqueness constraint.
-805	51002	The DBRM or package name was not found in the plan.
-811	21000	Must use a cursor when more than one row is returned as the result of an embedded SELECT statement.
-818	51003	Plan/Package vs. load module timestamp mismatch. The DBRM in the executing plan or package was not created from the same precompilation as the load module.
-904	57011	The specified resource is unavailable. Determine why, and retry the request.
-911	40001	The current unit of work has been rolled back.
-913	57033	Unsuccessful execution caused by deadlock or timeout.
-922	42505	The user is not authorized to perform the task

```
CREATE MASK SSNMASK
ON     EMP
FOR COLUMN SSN RETURN
  CASE
    WHEN (VERIFY_GROUP_FOR_USER(SESSION_USER, 'PAYROLL') = 1)
    THEN SSN
    WHEN (VERIFY_GROUP_FOR_USER(SESSION_USER, 'HR') = 1)
    THEN 'XXX-XX-' || SUBSTR(SSN,8,4)
    ELSE NULL
  END
ENABLE;

COMMIT;
```

This mask returns the actual data when accessed by a user in accounting, a version with the first 5 digits masked when access by human resources, and null for anyone else. Of course, column access control must be activated for the table before any mask will be enforced:

```
ALTER TABLE EMP
 ACTIVATE COLUMN ACCESS CONTROL;

COMMIT;
```

Row and Column Access Control Guidelines

Using row and column access control, a security administrator can enforce detailed security policies for the databases under their control. Keep the following issues and guidelines in mind as you implement row and column access control.

Views Access control is not needed at the view level because a view automatically receives row and column access control from its underlining base tables. All access restrictions are based on individual user permissions and masks specified by policies associated with the base tables.

Authority Required `SECADM` authority is required to activate or deactivate row and column access control for a table, to grant or revoke the `CREATE_SECURE_OBJECT` system privilege, and to create, alter, or drop row permissions and column masks.

If the `SEPARATE SECURITY` system parameter (on `DSNTIPP1`) is set to `NO` during installation, `SYSADM` can be used to perform these tasks. However, as of DB2 V10, `SECADM` should be viewed as an industry best practice instead of relying on `SYSADM`, which is more powerful, and therefore more prone to compliance issues.

Activation When a table is activated for row or column access control, all users, including the table owner and `SECADM`, `SYSDM`, and `DBADM` authorities, are subject to the same security condition.

You can activate row and column access control either before or after the row permissions or column masks have been created. If they already exist, activation begins enforcing the permissions and masks become effective. If they do not exist, activating row access control for a table causes DB2 to generate a default row permission that prevents any

access to the table by SQL; activating column access control waits for the column masks to be created.

In general, it is a wiser course of action to create row permissions and masks before activation unless you want to stop all activity until the row and access control is built.

Row and Access Control Versus Multi-Level Security (MLS) Although similar in effect, row access control and MLS cannot be used together. If a table is activated for row access control, it cannot be altered to include a security label column, and vice versa.

Column access control can coexist with MLS. If a table is activated for column level access control, it can be altered to include a security label column, and vice versa.

Session Variables

Session variables, set by DB2 or by the user, offer another way to provide additional information to applications. Session variables are set by DB2, and are accessible to application programs and end users. By accessing session variables, an application program can determine many aspects of its environment and the DB2 features in effect.

Table 12.1 outlines the session variables available to DB2 users.

TABLE 12.1 DB2 Session Variables

Session Variable	Description
APPLICATION_ENCODING_SCHEME	The application encoding scheme specified for the subsystem. Value is EBCDIC, ASCII, UNICODE, or 1–65533.
COBOL_STRING_DELIMITER	The string delimiter. Value will be DEFAULT, ", or '.
DATA_SHARING_GROUP_NAME	Name of the data sharing group.
DATE_FORMAT	The date format in use. Value will be ISO, JIS, USA, EUR, or LOCAL.
DATE_LENGTH	The LOCAL DATE LENGTH install parameter. Value is 10-254, or 0 for no exit.
DECIMAL_ARITHMETIC	The DECIMAL ARITHMETIC install parameter. Value is DEC15, DEC31, 15, or 31.
DECIMAL_POINT	The DECIMAL POINT install parameter. Value is '.' or ','.
DEFAULT_DECFLOAT_ROUND_MODE	The DECFLOAT ROUNDING MODE install parameter.
DEFAULT_DEFAULT_SSID	The value of GROUP ATTACH field on the DSNTIPK installation panel or the SUBSYSTEM NAME field on the DSNTIPM installation panel.
DEFAULT_LANGUAGE	The LANGUAGE DEFAULT install parameter. Value is ASM, C, CPP, IBMCOB, FORTRAN, or PL/I.
DEFAULT_LOCALE_LC_CTYPE	The value of LOCALE LC_CTYPE install parameter.
DSNHDECP_NAME	The fully qualified data set name of the data set from which the DSNHDECP or a user-specified application defaults module was loaded.
DYNAMIC_RULES	The USE FOR DYNAMICRULES install parameter. Value is YES or NO.
ENCODING_SCHEME	The DEF ENCODING SCHEME install parameter. Value is EBCDIC, ASCII, or UNICODE.

TABLE 12.1 Continued

Session Variable	Description
MIXED_DATA	The MIXED DATA install parameter. Value is YES or NO.
NEWFUN	The INSTALL TYPE parameter. Value is INSTALL, UPDATE, MIGRATE, or ENFM; reflects the setting of the DSNHDECP variable NEWFUN.
PACKAGE_NAME	Name of the package currently in use.
PACKAGE_SCHEMA	Schema name of the current package.
PACKAGE_VERSION	Version of the current package.
PAD_NUL_TERMINATED	The PAD NUL-TERMINATED install parameter. Value is YES or NO.
PLAN_NAME	Name of the plan currently being run.
SECLABEL	The user's security label (if any); null if not defined.
SQL_STRING_DELIMITER	The SQL STRING DELIMITER install parameter. Value is DEFAULT, ", or '.
SSID	DB2 subsystem identifier.
STANDARD_SQL	The STD SQL LANGUAGE install parameter. Value is YES or NO.
SYSTEM_NAME	Name of the system, as defined in field SUBSYSTEM NAME on installation panel DSNTIPM.
SYSTEM_ASCII_CCSID	A comma-delimited string of the ASCII CCSIDs in use on this system.
SYSTEM_EBCDIC_CCSID	A comma-delimited string of the EBCDIC CCSIDs in use on this system.
SYSTEM_UNICODE_CCSID	A comma-delimited string of the UNICODE CCSIDs in use on this system.
TIME_FORMAT	The TIME FORMAT install parameter. Value is ISO, JIS, USA, EUR, or LOCAL.
TIME_LENGTH	The LOCAL TIME LENGTH install parameter. Value is 8-254, or 0 for no exit.
VERSION	Version of the DB2 subsystem. This value is a string, formatted as *pppvvrrm* where: *ppp* is a product string set to the value 'DSN'. *vv* is a two-digit version identifier such as '09'. *rr* is a two-digit release identifier such as '01'. *m* is a one-digit maintenance level identifier.

Each session variable must be qualified by SYSIBM. A built-in function named GETVARIABLE can retrieve session variable values. So, you could create a view based on a security label, for example:

```
CREAT VIEW VSECLBL AS
  SELECT column-list
  FROM   table-name
  WHERE  SECLABEL_COL = GETVARIABLE(SYSIBM.SECLABEL);
```

The built-in function can be used in views, triggers, stored procedures, and constraints to enforce a security policy.

Users can add up to ten session variables by setting the name and value in their connection or sign-on exits. User-created session variables are qualified by SESSION. For example, the customer might have a connection or sign-on exit that examines the SQL user's IP address, and maps the IP address to the user's site within the company. This is recorded in a session variable, named say, USER_SITE. This session variable is then accessible using the built-in function, for example:

```
GETVARIABLE(SESSION.USER_SITE)
```

Using session variables much more information is available to application programs as they execute, and more control and security is provided, as well. Additionally, session variables can be trusted. They are set by DB2 and an application cannot modify them.

Data Definition Control

DB2 data definition control support provides a mechanism for controlling how and who can issue DDL statements. It requires installing and setting up special tables for managing data definition control. These are the application registration table (ART) and the object registration table (ORT).

Data definition control enables you to control DDL by application name, by application name with exceptions, by object name, or by object name with exception.

Consider installing and using data definition control if you have applications that must CREATE, DROP, or ALTER database objects. Data definition control can be set up for controlling DDL on aliases, comments, databases, indexes, labels, storage groups, synonyms, tables, table spaces, and views.

Trusted Context and Roles

V9 Introduced in DB2 V9, a trusted context is a database object that identifies a connection as trusted when connecting to DB2 for z/OS from a specific location. A TRUSTED CONTEXT is a database object that identifies a specific location and is owned by a single system authid.

After a TRUSTED CONTEXT is established, you can define a set of interactions between DB2 and the external entity so that the existing database connection can be used by a different user without requiring authentication.

What Problem Does Trusted Context Solve?

Trusted context addresses the problem of establishing a trusted relationship between DB2 and an external entity, such as a middleware server. Examples include WebSphere® and ERP servers such as SAP NetWeaver, Oracle Siebel, and Oracle PeopleSoft.

The typical behavior for connecting to these tiered or middleware platforms before the advent of trusted context was to use one system userid for establishing the connection, as well as for performing all transactions on behalf of every end user. But this clearly was not an ideal setup from a security- and compliance-perspective. This lack of visibility also masked which transactions at the middle tier were performed on behalf of the user and which were performed solely for the benefit of the application.

A security hole exists in this type of setup because even though individual users authenticate at the application server, the application itself uses a generic authid and

password, perhaps hard-coded into programs. For dynamic SQL, which is typical for client/server applications, the generic authid has the authority to access and modify data in application tables. It becomes a trivial exercise for a user intent on malicious activity to use the exposed authid and password of the application for other access.

Trusted context can solve this problem by granting the privileges for dynamic SQL activity to a ROLE, instead of to a general authid. A ROLE provides context dependent privileges in that the role is available only within the trusted context, and the privileges granted to the role can be exercised only through the trusted connection. The trusted context thereby can be used to limit the exercise of a role's privileges to users connecting to DB2 from a particular application server identified by an IP address or domain name.

In this manner a trusted context plugs this common DB2 security hole. The generic authid used by the application is not useful unless it can be associated with the ROLE that has been granted the privileges. But remember, the ROLE can be used only when the connection to DB2 is from the identified server within the trusted context.

Roles

A ROLE is a database object to which DB2 privileges can be granted or revoked. Roles enable the delivery of context-specific privileges. Within a trusted connection you can switch to another authid (with or without authentication) and exercise the use of ROLEs. This means that within a trusted context, privileges can be assigned via roles, instead of directly to a user's authid. Furthermore, you can audit database transactions based on the actual switched user instead of on the middleware ID.

Basically, roles offer a more flexible approach to authorization than groups or users. A ROLE can be thought of as a virtual authid assigned to an actual authid through a trusted connection. Within a trusted connection, a single ROLE can be associated with a thread at any point in time.

An Example of Roles and Trusted Context

Now walk through a quick example. In this case, assume you want to set up the authorization in a PeopleSoft environment. You can CREATE a ROLE for the PeopleSoft application as follows:

```
CREATE ROLE PSFT_ROLE;
```

> **NOTE**
>
> The name of the ROLE cannot begin with the characters SYS nor can it be DBADM, NONE, NULL, PUBLIC, or SECADM.

The next step is to GRANT the requisite privileges to the ROLE. This would require authorizing the PSFT_ROLE to access the PeopleSoft table, as well as any other activity required by the vendor. Then you need to define the TRUSTED CONTEXT within which the ROLE will be used.

```
CREATE TRUSTED CONTEXT PSFTCTXT
BASED UPON CONNECTION USING SYSTEM AUTHID PSFTADM
ATTRIBUTES (ADDRESS '1.22.33.444')
DEFAULT ROLE PSFT_ROLE
ENABLE;
```

Of course, you need to substitute an accurate IP address.

Trusted Context Guidelines

Consider the following additional guidelines and tips as you prepare to use trusted context and roles in your shop.

Consider Local Trusted Context Even though you concentrated on the most common use for a trusted context—for securing an a distributed connection to DB2—a trusted context also can be defined for a local connection to DB2 through a batch job or a started task.

Restricting a `ROLE` to an Individual When you create a `TRUSTED CONTEXT`, you can use the `WITH USE FOR` clause to restrict the use of a `ROLE` to individual IDs using a trusted connection.

If no `WITH USE FOR` clause is specified on the `TRUSTED CONTEXT`, the privileges of the role associated with the trusted context can be used by anybody that connects to DB2 through that trusted connection. This may be sufficient if there is a logon and authentication process required to use the application that uses the connection.

`ROLE`s as Object Owner When you create a `TRUSTED CONTEXT`, you can indicate whether the `ROLE` is to be used as the owners of objects created using a trusted connection based on the specified trusted context. By coding the `WITH ROLE AS OBJECT OWNER AND QUALIFIER`, you indicate that the context assigned role is the owner of the objects created using a trusted connection based on this trusted context and that role must possess all the privileges necessary to create the object. The associated `ROLE` is used as the default for the `CURRENT SCHEMA` special register.

Encryption

Another security feature for data protection is encryption. Data encryption is a process whereby data is transformed using an algorithm to make it unreadable to anyone without the decryption key. The general idea is to make the effort of decrypting so difficult as to outweigh the advantage to a hacker of accessing the unauthorized data. Figure 12.3 illustrates the general flow of encryption and decryption.

Encryption has been used by governments and military organizations for years to enable secret data transmission and communication. With the added data protection and regulatory requirements of late, encryption is more commonly used in many business systems and applications.

There are two types of situations in which data encryption can be deployed: data in transit and data at rest. In a database context, data “at rest” encryption protects data stored in the database, whereas data “in transit” encryption is used for data transferred over a network.