IBM® DB2® for Linux®, UNIX®, and Windows®

Best practices

A practical guide to restrictive databases

Mihai Iacob
DB2 Security Development
IBM Canada Lab

Paolo Cirone
DB2 Information Development
IBM Canada Lab

Walid Rjaibi
IBM Senior Technical Staff Member
DB2 Security Architect
IBM Canada Lab

Mang Tang
DB2 Security Development
IBM Canada Lab

Issued: June 2012

Executive summary

Tools such as IBM® InfoSphere® Guardium® provide vulnerability assessment capabilities to security administrators. These tools typically provide reports that identify compliance failures with security best practices and current vulnerabilities.

Starting with DB2® for Linux®, UNIX®, and Windows® Version 10.1, DB2 and InfoSphere Guardium software are tuned for better out of the box integration. As a result, restrictive DB2 databases achieve a better compliance score on vulnerability assessment tests when compared to restrictive DB2 databases from previous releases and non-restrictive DB2 databases in general.

Restrictive DB2 databases provide an out of the box *least privilege* approach to databases and database objects. This least privilege approach makes it much easier for you to secure your databases.

While restrictive databases provide better out of the box, security compliance scores, their use has been limited because administrators did not know what privileges to grant their end users to make them more usable. For example, administrators might not understand why they needed to grant EXCUTE privilege on some CLP package that, on the surface, has nothing to do with the statement their user issued. This paper provides you with all the information you need to more effectively make use of restrictive databases.

Introduction

This paper provides a practical guide to getting started with restrictive DB2 databases. The paper includes examples that detail the required steps to set up database authorizations and privileges for typical categories of users: with CONNECT authority, with SECADM authority, with DBADM authority, and with DBADM, DATAACCESS, and ACCESSCTRL authority. Also included are details on the different requirements in terms of privileges on various system objects when the command line processor, CLP, is used to connect to restrictive DB2 databases.

A follow-on paper will cover details on the different requirements in terms of privileges on various system objects when CLI, Java, and Perl clients are used to connect to restrictive DB2 databases.

Restrictive DB2 databases

Restrictive databases were first introduced in DB2 for Linux, UNIX, and Windows Version 9.1. Restrictive databases provide a solution to customers for whom database security is a major concern. This capability allows users to create a database where a very limited number of privileges are granted to the special group PUBLIC.

Starting with DB2 for Linux, UNIX, and Windows Version 10.1, security is further enhanced. As of Version 10.1, no privileges are granted to the special group PUBLIC by the database manager when a restrictive database is created and after it is created. This change allows for greater control over who has what privileges and authorities in the database. This change implies that the security administrator must take time and set up customized authorization profiles for various types of users that require access to the database based on their job description.

To secure a database prior to the introduction of restrictive databases, security administrators had to take the opposite approach. Security administrators had to selectively, or completely, revoke authorities and privileges from the special group PUBLIC.

To create a restrictive database, you need to specify the RESTRICTIVE keyword with the CREATE DATABASE command. Here is an example:

```
db2 CREATE DATABASE testdb RESTRICTIVE;
```

This sets the restrict_access database configuration parameter to YES. After the restrict_access parameter is set when the database is created, it cannot be modified.

Non-restrictive versus restrictive databases

This section highlights the differences between non-restrictive and restrictive databases.

In order to try the examples in this section, you must first create the required databases. It is assumed that all commands are run as user NEWTON, which has SYSADM authority.

```
db2start;

db2 create db testdbnr;

db2 create db testdbr restrictive;
```

Restrict_access database configuration parameter

In a non-restrictive database, restrict_access is set to NO. In a restrictive database, restrict_access is set to YES. This database configuration parameter cannot be modified after the database is created.

You can use the following command to check the value of restrict_access on the non-restrictive database, testdbnr, we created:

```
db2 get db cfg for testdbnr | grep -i restrict

Restrict access                                        = NO
```

You can use the following command to check the value of restrict_access on the restrictive database, testdbr, we created:

```
db2 get db cfg for testdbr | grep -i restrict

Restrict access                                        = YES
```

Database authorities

In a non-restrictive database, some database authorities such as BINDADD, CONNECT, CREATETAB, and IMPLICIT_SCHEMA are granted to the special group PUBLIC. In a restrictive database no database authorities are granted to the special group PUBLIC.

You can use the following commands to check which database authorities are granted to PUBLIC on the non-restrictive database, testdbnr, we created:

```
db2 connect to testdbnr user newton using <password> ;

db2 "select substr(authority,1,25) as authority, d_user,
d_group, d_public, role_user, role_group, role_public, d_role
from table
(sysproc.auth_list_authorities_for_authid ('PUBLIC', 'G') )
as t order by authority";

AUTHORITY                    D_USER D_GROUP D_PUBLIC ROLE_USER ROLE_GROUP ROLE_PUBLIC D_ROLE

-------------------------    ------ ------- -------- --------- ---------- ----------- ------

ACCESSCTRL                   *      *       N        *         *          N           *

BINDADD                      *      *       Y        *         *          N           *

CONNECT                      *      *       Y        *         *          N           *

CREATETAB                    *      *       Y        *         *          N           *

CREATE_EXTERNAL_ROUTINE      *      *       N        *         *          N           *

CREATE_NOT_FENCED_ROUTINE    *      *       N        *         *          N           *

CREATE_SECURE_OBJECT         *      *       N        *         *          N           *
```

```
DATAACCESS                *     *      N       *        *          N          *

DBADM                     *     *      N       *        *          N          *

EXPLAIN                   *     *      N       *        *          N          *

IMPLICIT_SCHEMA           *     *      Y       *        *          N          *

LOAD                      *     *      N       *        *          N          *

QUIESCE_CONNECT           *     *      N       *        *          N          *

SECADM                    *     *      N       *        *          N          *

SQLADM                    *     *      N       *        *          N          *

SYSADM                    *     *      *       *        *          *          *

SYSCTRL                   *     *      *       *        *          *          *

SYSMAINT                  *     *      *       *        *          *          *

SYSMON                    *     *      *       *        *          *          *

WLMADM                    *     *      N       *        *          N          *

 20 record(s) selected.

db2 connect reset;
```

You can use the following commands to check that no database authorities are granted to PUBLIC on the restrictive database, testdbr, we created:

```
db2 connect to testdbr user newton using <password> ;

db2 "select substr(authority,1,25) as authority, d_user,
d_group, d_public, role_user, role_group, role_public, d_role
from table
(sysproc.auth_list_authorities_for_authid ('PUBLIC', 'G') )
as t order by authority";

AUTHORITY                 D_USER D_GROUP D_PUBLIC ROLE_USER ROLE_GROUP ROLE_PUBLIC D_ROLE

------------------------- ------ ------- -------- --------- ---------- ----------- ------

ACCESSCTRL                *     *      N       *        *          N          *

BINDADD                   *     *      N       *        *          N          *

CONNECT                   *     *      N       *        *          N          *

CREATETAB                 *     *      N       *        *          N          *

CREATE_EXTERNAL_ROUTINE   *     *      N       *        *          N          *

CREATE_NOT_FENCED_ROUTINE *     *      N       *        *          N          *

CREATE_SECURE_OBJECT      *     *      N       *        *          N          *

DATAACCESS                *     *      N       *        *          N          *

DBADM                     *     *      N       *        *          N          *

EXPLAIN                   *     *      N       *        *          N          *
```

```
IMPLICIT_SCHEMA              *      *       N        *         *          N           *

LOAD                         *      *       N        *         *          N           *

QUIESCE_CONNECT              *      *       N        *         *          N           *

SECADM                       *      *       N        *         *          N           *

SQLADM                       *      *       N        *         *          N           *

SYSADM                       *      *       *        *         *          *           *

SYSCTRL                      *      *       *        *         *          *           *

SYSMAINT                     *      *       *        *         *          *           *

SYSMON                       *      *       *        *         *          *           *

WLMADM                       *      *       N        *         *          N           *

 20 record(s) selected.

db2 connect reset;
```

IMPLICIT_SCHEMA database authority

When you create a non-restrictive database, IMPLICIT_SCHEMA is automatically granted to the special group PUBLIC. When a user with IMPLICIT_SCHEMA authority on a non-restrictive database implicitly creates a schema, CREATEIN on that schema is automatically granted to both the user and to PUBLIC.

In a restrictive database environment, IMPLICIT_SCHEMA is not granted to PUBLIC. When a user with IMPLICIT_SCHEMA authority creates a schema, only that user receives CREATEIN on that schema.

You can use the following commands to check how the IMPLICIT_SCHEMA and CREATIN authorities are granted to PUBLIC on the non-restrictive database, testdbnr, we created:

```
db2 connect to testdbnr user newton using <password>;

db2 “select substr(grantor,1,10) as grantor,
grantortype, substr(grantee,1,10) as grantee,
granteetype, implschemaauth from syscat.dbauth where
grantee='PUBLIC'”

GRANTOR    GRANTORTYPE GRANTEE    GRANTEETYPE IMPLSCHEMAAUTH

---------- ----------- ---------- ----------- --------------

SYSIBM     S           PUBLIC     G           Y

 1 record(s) selected.

db2 “create table test_schema.test_table(col1 int)”;

db2 “select substr(grantor,1,10) as grantor,
grantortype, substr(grantee,1,10) as grantee,
granteetype, substr(schemaname,1,20) as schemaname,
alterinauth, createinauth, dropinauth from
syscat.schemaauth where schemaname='TEST_SCHEMA'" ;

GRANTOR    GRANTORTYPE GRANTEE    GRANTEETYPE SCHEMANAME           ALTERINAUTH CREATEINAUTH DROPINAUTH

---------- ----------- ---------- ----------- -------------------- ----------- ------------ ----------

SYSIBM     S           PUBLIC     G           TEST_SCHEMA          N           Y            N

 1 record(s) selected.

db2 connect reset;
```

You can use the following commands to check how the IMPLICIT_SCHEMA and CREATIN authorities are granted on the restrictive database, testdbr, we created:

```
db2 connect to testdbr user newton using <password>;

db2 "select substr(grantor,1,10) as grantor,
grantortype, substr(grantee,1,10) as grantee,
granteetype, implschemaauth from syscat.dbauth" ;

GRANTOR    GRANTORTYPE GRANTEE    GRANTEETYPE IMPLSCHEMAAUTH

---------- ----------- ---------- ----------- --------------

SYSIBM     S           NEWTON     U           N

 1 record(s) selected.

db2 "create table test_schema.test_table(col1 int)";

db2 "select substr(grantor,1,10) as grantor,
grantortype, substr(grantee,1,10) as grantee,
granteetype, substr(schemaname,1,20) as schemaname,
alterinauth, createinauth, dropinauth from
syscat.schemaauth where schemaname='TEST_SCHEMA' ";

GRANTOR    GRANTORTYPE GRANTEE    GRANTEETYPE SCHEMANAME           ALTERINAUTH CREATEINAUTH DROPINAUTH

---------- ----------- ---------- ----------- -------------------- ----------- ------------ ----------

SYSIBM     S           NEWTON     U           TEST_SCHEMA          N           Y            N

 1 record(s) selected.

db2 connect reset;
```

System objects

In non-restrictive databases a large number of privileges on system objects are granted to the special group PUBLIC. Detailed information is available by running the following SQL statement:

```
db2 "select * from sysibmadm.privileges where authid='PUBLIC"
```

The output gives a comprehensive report of privileges on database objects made to the special group PUBLIC.

In restrictive databases no privileges on system objects are granted to the special group PUBLIC by default.

You can use the following commands to check the number of privileges granted to PUBLIC on the non-restrictive database, testdbnr, we created:

```
db2 connect to testdbnr user newton using <password>;

db2 “select count(*) as number_of_grants_to_public from
sysibmadm.privileges where authid='PUBLIC'”;

NUMBER_OF_GRANTS_TO_PUBLIC

-------------------------

                      980

 1 record(s) selected.

db2 connect reset;
```

You can use the following commands to check the number of privileges granted to PUBLIC on the restrictive database, testdbr, we created:

```
db2 connect to testdbr user Newton using <password>;

db2 “select count(*) as number_of_grants_to_public from
sysibmadm.privileges where authid='PUBLIC'”;

NUMBER_OF_GRANTS_TO_PUBLIC

-------------------------

                         0

 1 record(s) selected.

db2 connect reset;
```

If you want a detailed report of privileges on database objects granted to the special group PUBLIC, run the following query:

```
db2 “select * from sysibmadm.privileges where authid='PUBLIC'”
```

User defined types

DB2 databases do not have a specific privilege associated with user defined types. Rather, eight EXECUTE privileges on cast and comparison functions are granted to specific users directly or indirectly through a role, trusted context role, group, or the special group PUBLIC. In a non-restrictive DB2 database, the database manager grants EXECUTE privileges on these system functions by default to the special group PUBLIC.

In a restrictive database, no such privilege is granted automatically.
For more information on this topic, see the Common usage scenarios section in this paper.

The following commands highlight how the EXECUTE privileges are granted in the non-restrictive database, testdbnr:

```
db2 connect to testdbnr user newton using <password>;

db2 create distinct type myint as integer with comparisons;

db2 "select substr(a.routineschema,1,10) as routineschema,
substr(a.routinename,1,10) as routinename,
substr(b.grantor,1,10) as grantor, b.grantortype,
substr(b.grantee,1,10) as grantee, b.granteetype,
b.executeauth from syscat.routines a,
syscat.routineauth b where a.routineschema='NEWTON'
and a.specificname=b.specificname order by b.grantee" ;

ROUTINESCHEMA ROUTINENAME GRANTOR    GRANTORTYPE GRANTEE    GRANTEETYPE EXECUTEAUTH

------------ ----------- ---------- ----------- ---------- ----------- -----------

NEWTON       MYINT       SYSIBM     S           NEWTON     U           G

NEWTON       INTEGER     SYSIBM     S           NEWTON     U           G

NEWTON       =           SYSIBM     S           NEWTON     U           G

NEWTON       <           SYSIBM     S           NEWTON     U           G

NEWTON       >           SYSIBM     S           NEWTON     U           G

NEWTON       <=          SYSIBM     S           NEWTON     U           G

NEWTON       >=          SYSIBM     S           NEWTON     U           G

NEWTON       <>          SYSIBM     S           NEWTON     U           G

NEWTON       MYINT       SYSIBM     S           PUBLIC     G           Y

NEWTON       INTEGER     SYSIBM     S           PUBLIC     G           Y

NEWTON       =           SYSIBM     S           PUBLIC     G           Y

NEWTON       <           SYSIBM     S           PUBLIC     G           Y

NEWTON       >           SYSIBM     S           PUBLIC     G           Y

NEWTON       <=          SYSIBM     S           PUBLIC     G           Y

NEWTON       >=          SYSIBM     S           PUBLIC     G           Y

NEWTON       <>          SYSIBM     S           PUBLIC     G           Y

 16 record(s) selected.

db2 connect reset;
```

The following commands highlight how the EXECUTE privileges are granted in the restrictive database, testdbr:

```
db2 connect to testdbr user newton using <password>;

db2 create distinct type myint as integer with comparisons;

db2 "select substr(a.routineschema,1,10) as routineschema,
substr(a.routinename,1,10) as routinename,
substr(b.grantor,1,10) as grantor, b.grantortype,
substr(b.grantee,1,10) as grantee, b.granteetype,
b.executeauth from syscat.routines a,
syscat.routineauth b where a.routineschema='NEWTON'
and a.specificname=b.specificname order by b.grantee" ;

ROUTINESCHEMA  ROUTINENAME GRANTOR    GRANTORTYPE GRANTEE    GRANTEETYPE EXECUTEAUTH

-------------  ----------- ---------- ----------- ---------- ----------- -----------

NEWTON         MYINT       SYSIBM     S           NEWTON     U           G

NEWTON         INTEGER     SYSIBM     S           NEWTON     U           G

NEWTON         =           SYSIBM     S           NEWTON     U           G

NEWTON         <           SYSIBM     S           NEWTON     U           G

NEWTON         >           SYSIBM     S           NEWTON     U           G

NEWTON         <=          SYSIBM     S           NEWTON     U           G

NEWTON         >=          SYSIBM     S           NEWTON     U           G

NEWTON         <>          SYSIBM     S           NEWTON     U           G

 8 record(s) selected.

db2 connect reset;
```

Common usage scenarios

This section highlights errors that users, with differing authorities, might encounter when performing tasks in a restrictive database environment. Suggestions are provided on how to overcome these errors.

The following user cases are highlighted:

- a user with no direct or indirect authorities or privileges on the database,
- a user with SECADM authority,
- a user with DBADM authority,
- a user with DBDM, DATAACCESS and ACCESSCTRL authority,
- a user with DBADM, DATAACCESS, ACCESSCTRL and SECADM authority.

These cases assume user NEWTON is the database creator. As a result, NEWTON has DBADM, DATAACCESS, ACCESSCTRL, and SECADM authorities on the database, testdbr. The database, testdbr, was created with the RESTRICTIVE option.

A user with no direct or indirect authorities or privileges on testdbr

This case assumes a user, JOE, is a regular user with no authorities on testdbr and no privileges on any objects in testdbr.

To check that JOE does not have any authorities on the database and no privileges on any objects within the database directly or indirectly, you can use the following commands:

```
db2 connect to testdbr user newton using <password>

db2 "select substr(authority,1,25) as authority,
d_user, d_group, d_public, role_user, role_group,
role_public, d_role from table
(sysproc.auth_list_authorities_for_authid ('JOE', 'U') )
as t order by authority";

AUTHORITY                 D_USER D_GROUP D_PUBLIC ROLE_USER ROLE_GROUP ROLE_PUBLIC D_ROLE

------------------------- ------ ------- -------- --------- ---------- ----------- ------

ACCESSCTRL                N      N       N        N         N          N           *

BINDADD                   N      N       N        N         N          N           *

CONNECT                   N      N       N        N         N          N           *

CREATETAB                 N      N       N        N         N          N           *

CREATE_EXTERNAL_ROUTINE   N      N       N        N         N          N           *

CREATE_NOT_FENCED_ROUTINE N      N       N        N         N          N           *

CREATE_SECURE_OBJECT      N      N       N        N         N          N           *

DATAACCESS                N      N       N        N         N          N           *

DBADM                     N      N       N        N         N          N           *

EXPLAIN                   N      N       N        N         N          N           *

IMPLICIT_SCHEMA           N      N       N        N         N          N           *

LOAD                      N      N       N        N         N          N           *

QUIESCE_CONNECT           N      N       N        N         N          N           *

SECADM                    N      N       N        N         N          N           *

SQLADM                    N      N       N        N         N          N           *

SYSADM                    *      N       *        *         *          *           *

SYSCTRL                   *      N       *        *         *          *           *

SYSMAINT                  *      N       *        *         *          *           *
```

```
SYSMON                    *      N       *        *         *          *           *

WLMADM                    N      N       N        N         N          N           *

 20 record(s) selected.

db2 “select count(*) as number_of_grants_to_joe from
sysibmadm.privileges where authid='JOE'” ;

NUMBER_OF_GRANTS_TO_JOE

-----------------------

                     0

 1 record(s) selected.

db2 “select count(*) as number_of_grants_to_public
from sysibmadm.privileges where authid='PUBLIC'” ;

NUMBER_OF_GRANTS_TO_PUBLIC

--------------------------

                        0

 1 record(s) selected.
```

Next, JOE tries to connect to testdbr. In this case an error indicating that JOE does not have CONNECT authority on testdbr is expected.

```
db2 connect to testdbr user joe using <password>;

SQL1060N  User "JOE     " does not have the CONNECT privilege.  SQLSTATE=08004
```

The solution to this problem is to grant CONNECT authority to JOE.

```
db2 connect to testdbr user newton using <password>;

db2 grant connect on database to user joe;
```

After JOE has been granted CONNECT authority, JOE can try to connect to testdbr again. In this case a successful connection to testdbr is expected.

```
db2 connect to testdbr user joe using <password>;

  Database Connection Information

Database server        = DB2/LINUXX8664 10.1.0

SQL authorization ID   = JOE

Local database alias   = TESTDBR
```

After JOE is connected to testdbr, he tries to create a schema.

```
db2 create schema joe;

DB21034E  The command was processed as an SQL statement because it was not a

valid Command Line Processor command.  During SQL processing it returned:

SQL5193N  The current session user does not have usage privilege on any

enabled workloads.  SQLSTATE=42524
```

To solve this particular problem, there are a few options:

- grant USAGE on SYSDEFAULTUSERWORKLOAD to JOE
- Create a database role or group for regular users, make JOE a member of this role or group and grant USAGE on SYSDEFAULTUSERWORKLOAD to this role or group. This might be the best solution because a database is likely to have more than one regular user. Therefore, creating a general role or group to hold privileges that are required by everyone is worth the effort.

The following commands illustrate how to create database roles for regular users to manage such cases.

```
db2 connect to testdbr user newton using password;

db2 create role ROLE_REGULAR_USERS;

db2 grant role ROLE_REGULAR_USERS to user JOE;

db2 grant USAGE on workload SYSDEFAULTUSERWORKLOAD to role
ROLE_REGULAR_USERS;

db2 connect reset;
```

Now JOE can again try to create a schema:

```
db2 connect to testdbr user joe using <password>;

db2 create schema joe;

DB21034E  The command was processed as an SQL statement because it was not a

valid Command Line Processor command.  During SQL processing it returned:

SQL0551N  "USER2" does not have the required authorization or privilege to

perform operation "EXECUTE" on object "NULLID.SQLC2J23".  SQLSTATE=42501
```

The client we are using, CLP, relies on some packages to do work behind the scenes. There are five packages, one for each isolation level. Each user that wants to use the CLP client requires EXECUTE privilege on these packages in order to run any statements.

To solve this problem, there are a few options:

- grant EXECUTE on each package to JOE,
- grant EXECUTE on each package to a role or group and make JOE a member of this role or group. An even better solution is to create a database role for the sole purpose of grouping these packages. Then grant membership in this role to the ROLE_REGULAR_USERS role created in the previous step. The following commands show this solution:

```
db2 connect to testdbr user newton using <password> ;

db2 "select substr(grantor,1,8) as grantor,
grantortype, substr(grantee,1,8) as grantee,
granteetype, substr(pkgschema,1,8) as pkgschema,
substr(pkgname,1,8) as pkgname,controlauth,bindauth,executeauth
from syscat.packageauth where pkgschema='NULLID' and
pkgname LIKE 'SQLC%'" ;

GRANTOR  GRANTORTYPE GRANTEE  GRANTEETYPE PKGSCHEMA PKGNAME  CONTROLAUTH BINDAUTH EXECUTEAUTH

-------- ----------- -------- ----------- --------- -------- ----------- -------- -----------

SYSIBM   S           NEWTON   U           NULLID    SQLC2J23 Y           G        G

SYSIBM   S           NEWTON   U           NULLID    SQLC3J22 Y           G        G

SYSIBM   S           NEWTON   U           NULLID    SQLC4J22 Y           G        G

SYSIBM   S           NEWTON   U           NULLID    SQLC5J22 Y           G        G

SYSIBM   S           NEWTON   U           NULLID    SQLC6J22 Y           G        G

 5 record(s) selected.
```

NEWTON has full access (CONTROL, BIND WITH GRANT, EXECUTE WITH GRANT) on these packages because he is the database creator.

```
db2 create role ROLE_CLP_PACKAGES ;

db2 grant EXECUTE on package NULLID.SQLC2J23 to
role ROLE_CLP_PACKAGES;

db2 grant EXECUTE on package NULLID.SQLC3J22 to
role ROLE_CLP_PACKAGES;

db2 grant EXECUTE on package NULLID.SQLC4J22 to
role ROLE_CLP_PACKAGES;

db2 grant EXECUTE on package NULLID.SQLC5J22 to
role ROLE_CLP_PACKAGES;

db2 grant EXECUTE on package NULLID.SQLC6J22 to
role ROLE_CLP_PACKAGES;
```

```
db2 grant role ROLE_CLP_PACKAGES to role ROLE_REGULAR_USERS ;
```

Now JOE can create a schema:

```
db2 connect to testdbr user joe using <password> ;

db2 create schema joe ;

DB20000I  The SQL command completed successfully.
```

Note: The CLP package names might change between different fix pack levels of a DB2 product version. Supporting a down-level CLP client requires the server to have the down-level CLP packages available for the specific client level.

JOE now tries to create a table in schema JOE.

```
db2 connect to testdbr user joe using <password> ;

db2 create table joe.mytable(col integer);

DB21034E  The command was processed as an SQL statement because it was not a

valid Command Line Processor command.  During SQL processing it returned:

SQL0552N  "USER2" does not have the privilege to perform operation "CREATE

TABLE".  SQLSTATE=42502
```

To solve this problem, grant CREATETAB authority on the database testdbr to JOE.

```
db2 connect to testdbr user newton using <password>;

db2 grant createtab on database to user joe;

db2 connect to testdbr user joe using <password>;

db2 create table joe.mytable(col integer);

DB21034E  The command was processed as an SQL statement because it was not a

valid Command Line Processor command.  During SQL processing it returned:

SQL0286N  A table space could not be found with a page size of at least "4096"

that authorization ID "USER2" is authorized to use.  SQLSTATE=42727
```

JOE does not have USE privilege on any table space in the testdbr database. When a database is created, one of the initial table spaces created is USERSPACE1. USERSPACE1 is intended for user-defined tables and indexes. For more information on table spaces, see the Further reading section of this paper.

The solution is to grant USE of table space USERSPACE1 to user JOE.

```
db2 connect to testdbr user newton using <password>;

db2 grant use of tablespace USERSPACE1 to user JOE ;

db2 connect to testdbr user joe using <password>;

db2 create table joe.mytable(col integer);

DB20000I  The SQL command completed successfully.
```

When a database is created, there are two user-defined types created automatically: DB2SECURITYLABEL and SQLSTATE. As previously mentioned, when a user-defined type is created, eight cast and comparison functions are created automatically. In a restrictive database, EXECUTE authority on these functions is not granted to PUBLIC. This portion of the scenario tries to make use of the DB2SECURITYLABEL type.

The purpose of this paper is not to explain label based access control (LBAC). Detailed information about LBAC is available in other technical papers. For the purposes of this portion of the scenario the SECADM user, NEWTON, creates the necessary LBAC objects so that JOE can create an LBAC protected table.

```
db2 connect to testdbr user newton using <password>;

db2 "create security label component DEPARTMENTS
set {'Collection', 'Research', 'Analysis'}";

db2 "create security policy p components
DEPARTMENTS with db2lbacrules ";

db2 "create security label p.analysis_label component
DEPARTMENTS 'Analysis'";

db2 grant security label p.analysis_label to user joe;

db2 connect to testdbr user joe using <password>;

db2 "create table joe.departments(id int, label
DB2SECURITYLABEL) security policy p" ;

DB21034E  The command was processed as an SQL statement because it was not a

valid Command Line Processor command.  During SQL processing it returned:

SQL0574N  DEFAULT value or IDENTITY attribute value is not valid for column

"LABEL" in table "JOE.DEPARTMENTS".  Reason code: "1".  SQLSTATE=42894
```

This error could be misleading. The reason JOE cannot create this protected table is he does not have EXECUTE authority on the function SYSPROC.DB2SECURITYLABEL.

There are a few options to solve this problem:

- grant EXECUTE authority to JOE only on function SYSPROC.DB2SECURITYLABEL,

- grant EXECUTE authority to JOE on all cast and comparison functions for the DB2SECURITYLABEL system type,

- create a database role for each system type (DB2SECURITYLABEL, SQLSTATE) and group the cast and comparison functions in that role. Then grant the roles the ROLE_REGULAR_USERS role, so that anyone can make use of the default system types. This is the preferred solution.

```
db2 connect to testdbr user newton using <password>;

db2 create role ROLE_DB2SECURITYLABEL_TYPE ;

db2 create role ROLE_SQLSTATE_TYPE;

db2 "select substr(a.routineschema,1,10) as routineschema,
substr(a.routinename,1,16) as routinename,
substr(a.specificname,1,18) as
specificname,substr(b.grantor,1,10) as grantor,
b.grantortype, substr(b.grantee,1,10) as grantee,
b.granteetype, b.executeauth from syscat.routines a,
syscat.routineauth b where a.routineschema='SYSPROC'
and a.definer='NEWTON' and a.specificname=b.specificname
order by b.grantee" ;

ROUTINESCHEMA ROUTINENAME      SPECIFICNAME       GRANTOR    GRANTORTYPE GRANTEE    GRANTEETYPE EXECUTEAUTH

------------- ---------------- ------------------ ---------- ----------- ---------- ----------- -----------

SYSPROC       DB2SECURITYLABEL SQL120516152816900 SYSIBM     S           NEWTON     U           G

SYSPROC       VARCHAR          SQL120516152816901 SYSIBM     S           NEWTON     U           G

SYSPROC       =                SQL120516152817000 SYSIBM     S           NEWTON     U           G

SYSPROC       <                SQL120516152817038 SYSIBM     S           NEWTON     U           G

SYSPROC       >                SQL120516152817039 SYSIBM     S           NEWTON     U           G

SYSPROC       <=               SQL120516152817040 SYSIBM     S           NEWTON     U           G

SYSPROC       >=               SQL120516152817041 SYSIBM     S           NEWTON     U           G

SYSPROC       <>               SQL120516152817042 SYSIBM     S           NEWTON     U           G

SYSPROC       DB2SQLSTAT       SQL120516152817001 SYSIBM     S           NEWTON     U           G

SYSPROC       CHAR             SQL120516152817002 SYSIBM     S           NEWTON     U           G

SYSPROC       DB2SQLSTAT       SQL120516152817003 SYSIBM     S           NEWTON     U           G

SYSPROC       =                SQL120516152817100 SYSIBM     S           NEWTON     U           G

SYSPROC       <                SQL120516152817147 SYSIBM     S           NEWTON     U           G

SYSPROC       >                SQL120516152817148 SYSIBM     S           NEWTON     U           G

SYSPROC       <=               SQL120516152817149 SYSIBM     S           NEWTON     U           G
SYSPROC       >=              SQL120516152817150 SYSIBM     S           NEWTON     U           G

SYSPROC       <>              SQL120516152817151 SYSIBM     S           NEWTON     U           G

 17 record(s) selected.

db2 grant execute on specific function
SYSPROC.SQL120516152816900 to role role_db2securitylabel_type;

db2 grant execute on specific function
SYSPROC.SQL120516152816901 to role role_db2securitylabel_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817000 to role role_db2securitylabel_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817038 to role role_db2securitylabel_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817039 to role role_db2securitylabel_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817040 to role role_db2securitylabel_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817041 to role role_db2securitylabel_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817042 to role role_db2securitylabel_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817001 to role role_sqlstate_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817002 to role role_sqlstate_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817003 to role role_sqlstate_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817100 to role role_sqlstate_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817147 to role role_sqlstate_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817148 to role role_sqlstate_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817149 to role role_sqlstate_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817150 to role role_sqlstate_type;

db2 grant execute on specific function
SYSPROC.SQL120516152817151 to role role_sqlstate_type;
```

```
db2 grant role role_db2securitylabel_type to role
ROLE_REGULAR_USERS ;

db2 grant role role_sqlstate_type to role ROLE_REGULAR_USERS ;

db2 connect to testdbr user joe using <password>;

db2 "create table joe.departments(id int, label
DB2SECURITYLABEL) security policy p" ;

DB20000I  The SQL command completed successfully.
```

In the Non-restrictive versus restrictive databases section of this paper, using a user-defined type was discussed. This part of the scenario continues from that section.

```
db2 connect to testdbr user newton using <password> ;
```

If you did not already create a distinct type `myint` in a previous section, issue the following command:

```
db2 create distinct type myint as integer with comparisons ;
```

Continue with the following commands:

```
db2 connect to testdbr user joe using <password> ;

db2 “values newton.myint(7)” ;

SQL0551N  "JOE" does not have the required authorization or privilege to

perform operation "EXECUTE" on object "NEWTON.MYINT".  SQLSTATE=42501
```

To solve this problem, you can:

- grant EXECUTE on cast function NEWTON.MYINT to user JOE,
- a better solution is to create a database role to group the cast and comparison functions for type myint and grant this role to the ROLE_REGULAR_USERS role.

```
db2 connect to testdbr user newton using <password> ;

db2 "select substr(a.routineschema,1,10) as routineschema,
substr(a.routinename,1,10) as routinename,
substr(a.specificname,1,18) as
specificname,substr(b.grantor,1,10) as grantor,
b.grantortype, substr(b.grantee,1,10) as grantee,
b.granteetype, b.executeauth from syscat.routines a,
syscat.routineauth b where a.routineschema='NEWTON'
and a.definer='NEWTON' and a.specificname=b.specificname
order by b.grantee" ;

ROUTINESCHEMA ROUTINENAME SPECIFICNAME       GRANTOR    GRANTORTYPE GRANTEE    GRANTEETYPE EXECUTEAUTH

------------- ----------- ------------------ ---------- ----------- ---------- ----------- -----------

NEWTON        MYINT       SQL120520141108300 SYSIBM     S           NEWTON     U           G

NEWTON        INTEGER     SQL120520141108500 SYSIBM     S           NEWTON     U           G

NEWTON        =           SQL120520141108600 SYSIBM     S           NEWTON     U           G

NEWTON        <           SQL120520141108607 SYSIBM     S           NEWTON     U           G

NEWTON        >           SQL120520141108608 SYSIBM     S           NEWTON     U           G

NEWTON        <=          SQL120520141108709 SYSIBM     S           NEWTON     U           G

NEWTON        >=          SQL120520141108710 SYSIBM     S           NEWTON     U           G

NEWTON        <>          SQL120520141108711 SYSIBM     S           NEWTON     U           G

 8 record(s) selected.

db2 create role role_myint_type;

db2 grant execute on specific function
NEWTON.SQL120520141108300 to role role_myint_type;

db2 grant execute on specific function
NEWTON.SQL120520141108500 to role role_myint_type;

db2 grant execute on specific function
NEWTON.SQL120520141108600 to role role_myint_type;

db2 grant execute on specific function
NEWTON.SQL120520141108607 to role role_myint_type;

db2 grant execute on specific function
NEWTON.SQL120520141108608 to role role_myint_type;

db2 grant execute on specific function
NEWTON.SQL120520141108709 to role role_myint_type;

db2 grant execute on specific function
NEWTON.SQL120520141108710 to role role_myint_type;

db2 grant execute on specific function
NEWTON.SQL120520141108711 to role role_myint_type;

db2 grant role role_myint_type to role ROLE_REGULAR_USERS;

db2 connect to testdbr user joe using <password> ;

db2 “values newton.myint(7)” ;

1
```

```
-----------

        7

 1 record(s) selected.
```

In this part of the scenario, JOE tries to call a procedure created by another user.

```
db2 connect to testdbr user newton using <password> ;

db2 “create table newton.t1(c1 int)”;

db2 “create procedure p1 begin insert into
newton.t1 values(7);end”;

db2 grant EXECUTE on procedure p1 to user joe;

db2 connect to testdbr user joe using <password> ;

db2 “call newton.p1()” ;

SQL0551N  "JOE" does not have the required authorization or privilege to

perform operation "EXECUTE" on object "NULLID.SYSSH200".  SQLSTATE=42501
```

To solve this problem, you can:

- grant EXECUTE authority on package SYSSH200 to user JOE,
- a better solution is to create a role ROLE_CLI_PACKEGES and group together all the CLI packages, one for each isolation level, and grant to ROLE_CLI_PACKAGES the ROLE_REGULAR_USERS role.

```
db2 connect to testdbr user newton using <password> ;

db2 "select substr(grantor,1,8) as grantor,
grantortype, substr(grantee,1,8) as grantee,
granteetype, substr(pkgschema,1,8) as pkgschema,
substr(pkgname,1,8) as pkgname,controlauth,bindauth,executeauth
from syscat.packageauth where pkgschema='NULLID' and
pkgname LIKE 'SYSSH%'" ;

GRANTOR  GRANTORTYPE GRANTEE  GRANTEETYPE PKGSCHEMA PKGNAME  CONTROLAUTH BINDAUTH EXECUTEAUTH

-------- ----------- -------- ----------- --------- -------- ----------- -------- -----------

SYSIBM   S           NEWTON   U           NULLID    SYSSH100 Y           G        G

SYSIBM   S           NEWTON   U           NULLID    SYSSH101 Y           G        G

SYSIBM   S           NEWTON   U           NULLID    SYSSH102 Y           G        G

SYSIBM   S           NEWTON   U           NULLID    SYSSH200 Y           G        G

SYSIBM   S           NEWTON   U           NULLID    SYSSH201 Y           G        G
SYSIBM   S          NEWTON   U          NULLID    SYSSH202 Y          G        G

SYSIBM   S          NEWTON   U          NULLID    SYSSH300 Y          G        G

SYSIBM   S          NEWTON   U          NULLID    SYSSH301 Y          G        G

SYSIBM   S          NEWTON   U          NULLID    SYSSH302 Y          G        G

SYSIBM   S          NEWTON   U          NULLID    SYSSH400 Y          G        G

SYSIBM   S          NEWTON   U          NULLID    SYSSH401 Y          G        G

SYSIBM   S          NEWTON   U          NULLID    SYSSH402 Y          G        G

 12 record(s) selected.

db2 create role ROLE_CLI_PACKAGES;

db2 grant EXECUTE on package NULLID.SYSSH100 to
role ROLE_CLI_PACKAGES;

db2 grant EXECUTE on package NULLID.SYSSH101 to
role ROLE_CLI_PACKAGES;

db2 grant EXECUTE on package NULLID.SYSSH102 to
role ROLE_CLI_PACKAGES;

db2 grant EXECUTE on package NULLID.SYSSH200 to
role ROLE_CLI_PACKAGES;

db2 grant EXECUTE on package NULLID.SYSSH201 to
role ROLE_CLI_PACKAGES;

db2 grant EXECUTE on package NULLID.SYSSH202 to
role ROLE_CLI_PACKAGES;

db2 grant EXECUTE on package NULLID.SYSSH300 to
role ROLE_CLI_PACKAGES;

db2 grant EXECUTE on package NULLID.SYSSH301 to
role ROLE_CLI_PACKAGES;

db2 grant EXECUTE on package NULLID.SYSSH302 to
role ROLE_CLI_PACKAGES;

db2 grant EXECUTE on package NULLID.SYSSH400 to
role ROLE_CLI_PACKAGES;

db2 grant EXECUTE on package NULLID.SYSSH401 to
role ROLE_CLI_PACKAGES;

db2 grant EXECUTE on package NULLID.SYSSH402 to
role ROLE_CLI_PACKAGES;

db2 grant role ROLE_CLI_PACKAGES to role ROLE_REGULAR_USERS;

db2 connect to testdbr user joe using <password> ;
```

```
db2 “call newton.p1()” ;



Return Status = 0
```

Starting with DB2 for Linux, UNIX, and Windows Version 9.7, you can create a database in Oracle compatibility mode by setting the DB2_COMPATIBILITY_VECTOR to ORA. For the purposes of this part of the scenario a cursor variable is created.

As for the previous databases, the oradbr database is created by NEWTON.

```
db2set DB2_COMPATIBILITY_VECTOR=ORA;

db2stop;

db2start;

db2 create db oradbr restrictive;
```

Assume that the roles created thus far for testdbr have been created and setup similarly for oradbr.

```
db2 connect to oradbr user joe using <password>;

db2 “create table joe.t2(c1 int)”;

db2 "create variable vcur cursor constant
(cursor for select c1 from t2)";

DB21034E  The command was processed as an SQL statement because it was not a

valid Command Line Processor command.  During SQL processing it returned:

SQL0551N  "JOE" does not have the required authorization or privilege to

perform operation "EXECUTE" on object "SYSIBMINTERNAL.SQLEXC_COMPILE_PL".

SQLSTATE=42501
```

To solve this problem, you can:

- grant EXECUTE authority on procedures SYSIBMINTERNAL.SQLEXC_XCOMP1 and SYSIBMINTERNAL.SQLEXC_XCOMP2 to JOE,

- a better solution, as shown previously, is to create a database role, ROLE_ORA_MODE, and group these procedures in this role. Then grant membership in it to the ROLE_RELGULAR_USERS.

```
db2 connect to oradbr user newton using <password>;

db2 create role ROLE_ORA_MODE;

db2 grant EXECUTE on specific procedure
SYSIBMINTERNAL.SQLEXC_XCOMP1 to role  ROLE_ORA_MODE;

db2 grant EXECUTE on specific procedure
SYSIBMINTERNAL.SQLEXC_XCOMP2 to role  ROLE_ORA_MODE;

db2 grant role ROLE_ORA_MODE to role ROLE_REGULAR_USERS;

db2 connect to oradbr user joe using <password>;

db2 "create variable vcur cursor constant
(cursor for select c1 from t2)";
```

A user with SECADM authority

The SECADM authority is responsible for administering authorities and privileges within the database. The SECADM authority has some implicit authorities and privileges. Unlike DATAACCESS authority, it does not have access to data within a database. Of the cases covered in the section of this paper, A
user with no direct or indirect authorities or privileges on testdbr
the SECADM is susceptible to the following issues:

- workload authorization errors,
- CLP packages authorization errors,
- table space authorization errors,
- system and user-defined types authorization errors,
- oracle mode specific authorization errors,
- CLI packages authorization errors.

A user with DBADM authority

The DBADM authority is responsible for managing objects within a database. The DBADM authority has some implicit authorities and privileges.

Starting with DB2 Version 9.7, access to data has been removed from the DBADM authority and vested in the DATAACCESS authority. Also, the ability to grant and revoke privileges on objects has been removed from DBADM and vested in the ACCESSCTRL authority.

For more information, see the Authorization model topic in the Information Center referenced in the Further reading section of this paper.

Of the cases covered in the section of this paper, A
user with no direct or indirect authorities or privileges on testdbr, the DBADM
is susceptible to the following issues:

- CLP packages authorization errors,
- system and user defined-types authorization errors,
- CLI packages authorization errors.

A user with DBADM, DATAACCESS, and ACCESSCTRL authority

This is the equivalent of the DB2 Version 9.5 DBADM authority. Because of the authorities and privileges vested in these three authorities, this user will not experience any authorization errors pointed out in the section of this paper, A
user with no direct or indirect authorities or privileges on testdbr.

A user with DBADM, DATAACCESS, ACCESSCTRL and SECADM authority.

The database creator is granted these four authorities automatically when they create the database. In the examples covered in this paper, the database creator would be NEWTON. These four authorities together form what one might call a super-user.
This user will not experience any of the authorization errors pointed out in the section of this paper, A user with no direct or indirect authorities or privileges on testdbr.

Conclusion

This paper provides an overview of the restrictive DB2 database. It also compares the restrictive database with a standard database. The various common scenarios covered provide some background into typical authorization errors and pitfalls when using a restrictive database along with suggested solutions on how to avoid those errors.

Further reading

- Information Management best practices:
  http://www.ibm.com/developerworks/data/bestpractices/

- DB2 for Linux, UNIX, and Windows best practices:
  http://www.ibm.com/developerworks/data/bestpractices/db2luw/

- DB2 Version 10.1 Information Center:
  http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp

- CREATE DATABASE command:
  http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0001941.html

- restrict_access database configuration parameter:
  http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.config.doc/doc/r0022605.html

- Authorities overview:
  http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.sec.doc/doc/c0055206.html

- IMPLICIT_SCHEMA authority:
  http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.sec.doc/doc/c0005525.html

- DBADM authority:
  http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.sec.doc/doc/c0005521.html

- SECADM authority:
  http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.sec.doc/doc/c0021054.html

- DATAACCESS authority:
  http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.sec.doc/doc/c0053934.html

- ACCESSCTRL authority:
  http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.sec.doc/doc/c0053933.html

- AUTH_LIST_AUTHORITIES_FOR_AUTHID table function:
  http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.sql.rtn.doc/doc/r0052898.html

- SYSCAT.DBAUTH catalog view: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.sql.ref.doc/doc/r0001041.html

- SYSCAT.SCHEMAAUTH catalog view: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.sql.ref.doc/doc/r0001058.html

- SYSCAT.ROUTINEAUTH catalog view: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.sql.ref.doc/doc/r0007491.html

- SYSCAT.ROUTINES catalog view: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.sql.ref.doc/doc/r0001045.html

- PRIVILEGES administrative view – Retrieve privilege information: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.sql.rtn.doc/doc/r0021978.html

- DB2_COMPATIBILITY_VECTOR registry variable: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.apdv.porting.doc/doc/r0052867.html

- Default workloads: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.wlm.doc/doc/c0051473.html

- Usage privilege on workloads: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.sec.doc/doc/c0051935.html

- InfoSphere Guardium Database Vulnerability Assessment: http://www-01.ibm.com/software/data/guardium/database-vulnerability-assessment/

- Defining initial table spaces on database creation: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.dbobj.doc/doc/t0004922.html

Contributors

Serge Boivin
DB2 Information Development
IBM Canada Lab

Mokhtar Kandil
DB2 Security Development
IBM Canada Lab

Notices

Trademarks

Contacting IBM

To provide feedback about this paper, write to db2docs@ca.ibm.com

To contact IBM in your country or region, check the IBM Directory of Worldwide Contacts at http://www.ibm.com/planetwide

To learn more about IBM Information Management products, go to http://www.ibm.com/software/data/