System versus SQL name, Part 1: Object authority and privileges for SQL database objects

Birgitta Hauser

January 23, 2012

When executing SQL statements you can run them either using System or SQL Naming. The System Naming conventions follow the traditional methods used on IBM i systems such as library support. The SQL Naming mode on the other hand is defined in the SQL Standard and used by all other databases. When asking what is the difference between SQL and System Naming, you will normally get the answer, schema and object are either separated with a slash or a period. However there are many more differences especially with regard to the access authority and ownership of database objects created either using SQL or System Naming. This article will show the differences between SQL and System Naming primarily focusing on how the ownerships and access authorities vary when creating DB2 for i database objects (such as tables, stored procedures or triggers) with the two naming conventions.

View more content in this series

Naming Conventions for creating Database Objects

When creating database objects developers can choose a naming method that follows the **traditional IBM i behavior** with the System Naming mode (*SYS) or that conforms to the **SQL Standard rules** with the SQL Naming convention (*SQL).

The main difference between DB2 for i and other database management systems (DBMS) is that DB2 for i is integrated into the operating system. This integrated aspect allows IBM i users to directly access DB2 for i databases with their operating system user profile and the associated access authorities. Other databases are not integrated in an operating system, therefore specific database users with individual access authorities must be defined.

The default naming used to create database objects with SQL depends on the environment where the SQL DDL (Data Definition Language) commands are executed.

The default naming for all **server-side SQL environments**, such as STRSQL (Start SQL Interactive Session) or RUNSQLSTM (Run SQL Statements), but also embedded SQL in an HLL (High Level Language) program such as RPG or COBOL is **System Naming**.

The default naming value that is used on **client-based SQL environments**, such as System i Navigator, IBM Rational Developer for Power Systems Software (RDp), middleware (ODBC, JDBC, etc.) or third-party SQL tools is typically **SQL Naming**.

To avoid a mismatch in object authorities and access methods, you need to decide whether using System Naming or SQL Naming will work best in your application environment. You may need to change the default naming in some environments to match the naming conventions you are using in your application environment.

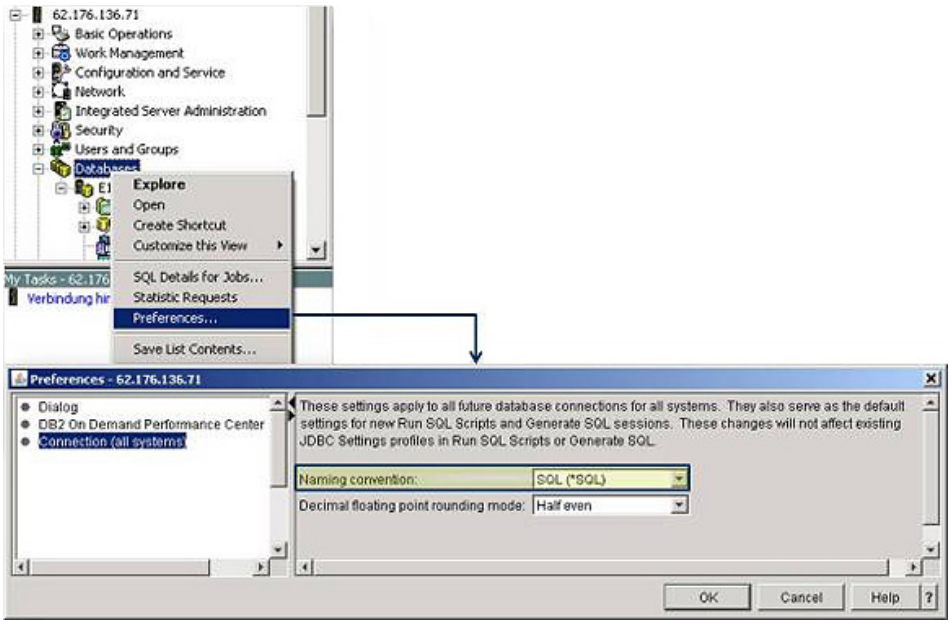
System i Navigator Interface

If you want to use the **System i Navigator Interface** to create your database objects, you can predefine the naming to be used as follows:

Open your connection, right-click on the Database icon and select the Preferences task as shown in Figure 1. System i Navigator – Set Preferences

The **Preferences** window offers 3 options. The **Connection (all Systems)** option allows you to predefine the naming convention to be used for future connections. This setting will also be used as the default naming value for future Run SQL Scripts and Generate SQL executions, but will NOT affect any existing windows.

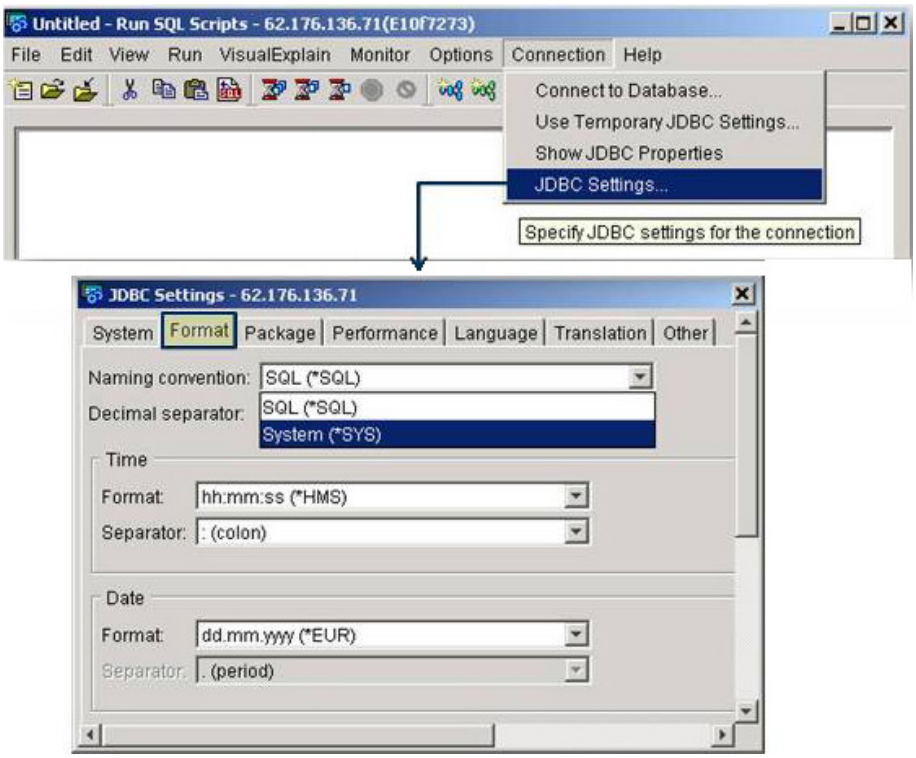
Figure 1. System i Navigator – Set Preferences



System i Navigator's Run SQL Scripts Tool

If you want to use the **Run SQL Scripts Tool** to execute a SQL script stored in a file or entered interactively, the naming convention is controlled by clicking on the Connection pull-down menu and selecting the JDBC Setup task. The naming convention can be set on the Format tab.

Figure 2. System i Navigator – Run SQL Scripts – Set Naming Conventions



RUNSQLSTM – Run SQL Statements

If you want to execute SQL statements stored in either a source physical file member or an IFS (Integrated File System) file through **RUNSQLSTM** (Run SQL Statements), the naming convention can be specified on the RUNSQLSTM command with the Naming parameter as shown in the following example. The specified SQL script will be executed by using SQL Naming.

Listing 1. RUNSQLSTM setting Naming Conventions

```
RUNSQLSTM SRCFILE(MYSCHEMA/QSQLSRC) SRCMBR(MYSCRIPT) NAMING(*SQL)
```

Embedded SQL in an HLL Program

If you want to use **embedded SQL** in an HLL program such as RPG or COBOL to either process data in your tables or to create new database objects, the default naming setting is System Naming.

If you want to use SQL Naming, you can predefine the naming convention in the compile command (CRTSQLRPGI, CRTSQLCBL1 or CRTSQLCI depending on the programming language) as shown in the following example:

Listing 2. Create an embedded SQL program using SQL Naming

```
CRTSQLRPGI OBJ(MYPGMLIB/MYSQLPGM) SRCFILE(MYSRCLIB/QRPGLESRC) SRCMBR(MYMBR) OPTION(*SQL)
```

Instead of specifying the naming method in the compile command, it is also possible to include it in your source code by adding a **SET OPTION** statement (as shown in the next example). The SET OPTION statement must be the first SQL statement in your source code and include all options you want to set.

Listing 3. SET OPTION Statement for setting Naming Conventions

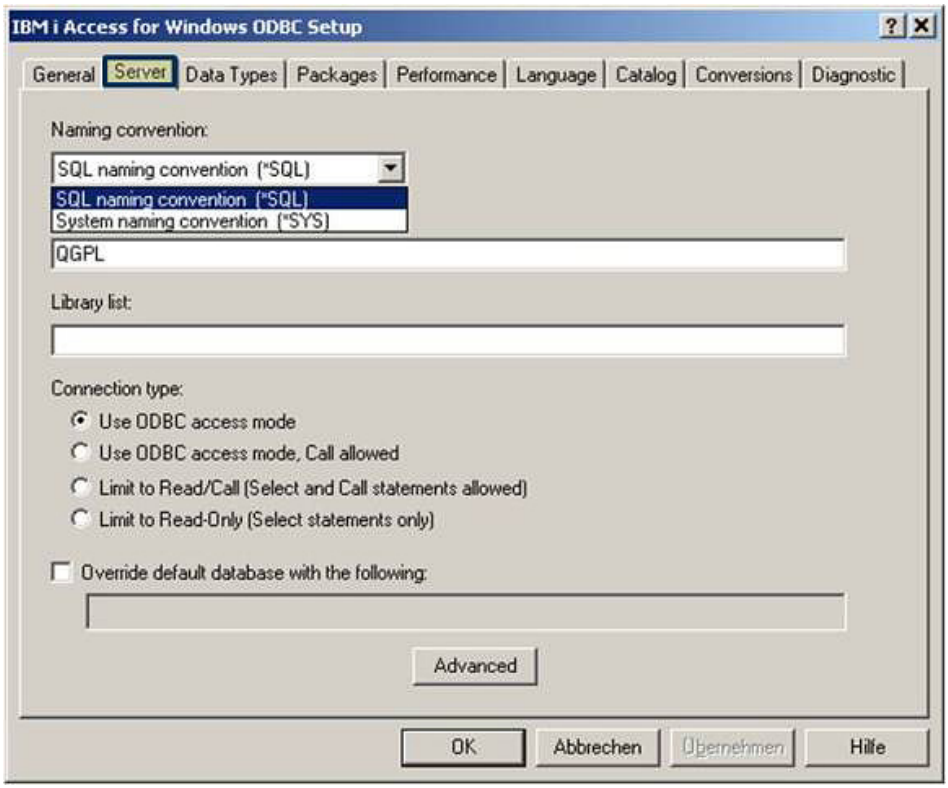
```
/Free Exec SQL Set Option Commit=*NONE, Naming=*SQL DatFmt=*ISO, CloSQLCsr=*ENDACTGRP; //All other source
 code including embedded SQL statements /End-Free
```

IBM i Access for Windows ODBC Driver

The naming convention can be specified for ODBC connections by using the IBM i Access for Windows - ODBC Administration interface or connection keywords.

The next figure shows the Naming convention being controlled on the ODBC Administration interface on the Server tab.

Figure 3. ODBC Setup



JDBC Access

For JDBC access naming convention can be controlled by specifying JDBC driver connection properties in the connection URL.

The **naming property** connection property supports the values of sql and system for naming convention. SQL Naming is the default.

When using System Naming a library list can be predefined with the **libraries property** as shown in the following example.

Listing 4. JDBC setting Naming Conventions

```
conn = DriverManager.getConnection("jdbc:db2:*local: ... ... naming=system;libraries=MYLIBA,MYLIBB,MYLIBX");
```

IBM i Access for Windows ADO.NET Provider

When using ADO.NET the naming convention and the library list for System Naming can be set when establishing the connection. The iDB2Connection object connects to DB2 for i. The naming convention is provided as Connection String property.

The following code shows how System Naming and the library list can be set with the iDB2Connection object:

Listing 5. ADO.NET setting Naming Conventions

```
iDB2Connection conn = new iDB2Connection("DataSource=abc; userid=XXX;password=YYY; Naming=System;
 LibraryList=*USRLIBL,MYLIB");
```

SQL CLI - Call Level Interface

When using the SQL CLI function, the naming convention is an attribute that can be set by executing the SQLSetConnectAttr function. To set the naming convention to System Naming, the SQL_ATTR_DBC_SYS_NAMING constant must be passed for the attribute parameter and the SQL_TRUE constant for the attribute value parameter as shown in the next example.

Listing 6. Set Connection Attributes using SQLCLI

```
rc = SQLSetConnectAttr(ConnHandle: SQL_ATTR_DBC_SYS_NAMING: SQL_TRUE: 4);
```

STRSQL – Start SQL Interactive Session

If you want to change the naming used to run your SQL statements in interactive SQL, execute the **STRSQL** CL command, press function key F13=Services and select option 1 (Change Session Attributes) after.

Schema – Container to hold Database Objects

A schema is a container to be used to store database objects. On IBM i the term schema is used as being an equivalent of a library.

Schemas or libraries can be created with either the CRTLIB (Create Library) CL command or the CREATE SCHEMA SQL statement. While the CRTLIB command only creates an empty container, the SQL statement automatically adds a journal, a journal receiver and a couple of catalog views with information about all database objects that are located in this schema.

When creating a library with the CRTLIB command, the owner of the library will be either the user profile that creates the library, or the group profile.

Whether the user or a group profile becomes the owner depends on the OWNER option setting for the user profile. If the OWNER option is set to *GRPPRF, the user profile specified in the GRPPRF option will become the owner of all objects created by this user, otherwise the user profile becomes the object owner.

The following example shows the CHGUSRPRF (Change User Profile) command to be used to set the owner of all objects in future created by the PGMRGRP2 user profile to the QPGMR group profile.

Listing 7. Change User Profile Command setting Owner = Group Profile

```
CHGUSRPRF USRPRF(PGMRGRP2) GRPPRF(QPGMR) OWNER(*GRPPRF)
```

All example database objects that are created in this article will be created by the user profile named PGMRGRP2. This user profile is associated with the QPGMR group profile. Based on this the QPGMR group profile will be the owner of all the database objects created by PGMRGRP2.

Creating a Schema using System Naming

When creating a schema with the CREATE SCHEMA statement using System Naming the following rules apply:

- The owner of the schema is the user profile or the group profile depending on the OWNER option setting in the user profile definition.
- The owner has *ALL object authority while the *PUBLIC object authority is based on the QCRTAUT (Create Default Public Authority) system value whose default value is *CHANGE.

Creating a schema or library with either the CRTLIB command or the CREATE SCHEMA statement with System Naming results in the same ownership and identical object authorities.

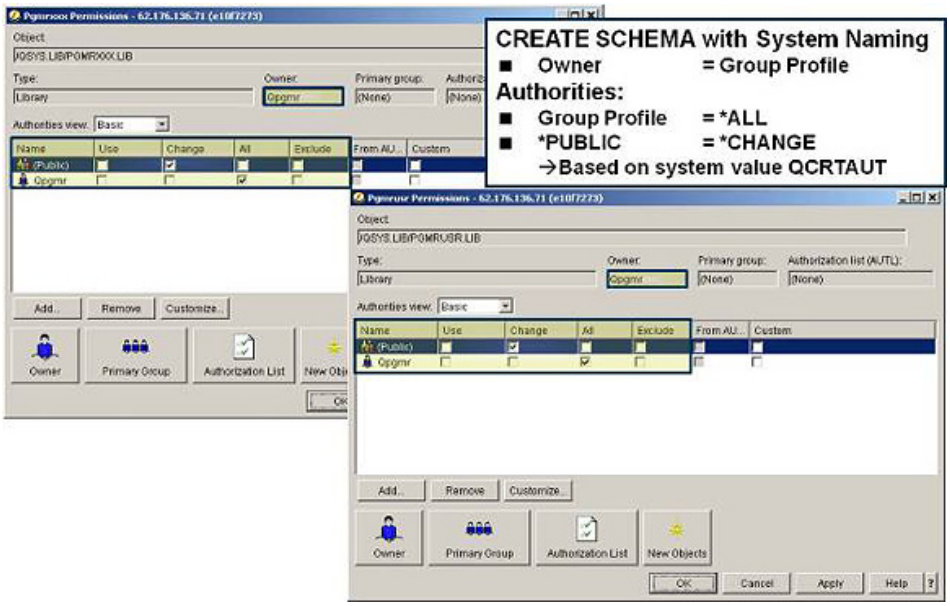
The PGMRGRP2 user profile creates two schemas (PGMRUSR2 and PGMRXXX2) with the following SQL statements using System Naming:

Listing 8. Create Schema Example

```
CREATE SCHEMA PGMRXXX2; CREATE SCHEMA PGMRUSR2;
```

The owner for both schemas is the QPGMR group profile. The group profile has *ALL object authority while the *PUBLIC authority is set to *CHANGE depending on the QCRTAUT system value.

Figure 4. CREATE SCHEMA with System Naming



The object owner and the assigned object authorities can be displayed, set or removed with either the EDTOBJAUT (Edit Object Authority) command or the System i Navigator Permission interface. With System i Navigator this interface can be accessed by right-clicking on a database object and selecting the Permissions task.

The object ownership can be changed with the CHGOBJOWN (Change Object Owner) CL command. There is no SQL statement or System i Navigator interface to change the object owner with SQL.

Creating a Schema using SQL Naming

When creating a schema with SQL Naming in effect the rules are more complicated:

- If a user profile with the same name as the schema exists, the owner of the schema and all objects created into this schema is that user profile. For example, a developer creates the schema WEBERP for a new web-based application. There happens to be an employee named Weber Peter whose user profile is also WEBERP. The user profile WEBERP becomes the owner of the WEBERP schema.
- If the schema name does not match a user profile name, the owner of the schema is the user profile of the job executing the CREATE SCHEMA statement. When creating a schema with SQL Naming, the OWNER option setting of the user profile definition is ignored.

The owner is the only user profile having any authority to the schema. If other users require object authority to the schema, the owner or a user profile with security administration authority (*SECADM) or all object authority (*ALLOBJ) can grant authority to the schema using the GRTOBJAUT (Grant Object Authority) CL command.

There is no SQL statement available to grant object authority for a schema.

- For database objects created with SQL Naming the *PUBLIC object authority is always set to *EXCLUDE. The QCRTAUT system value is ignored.

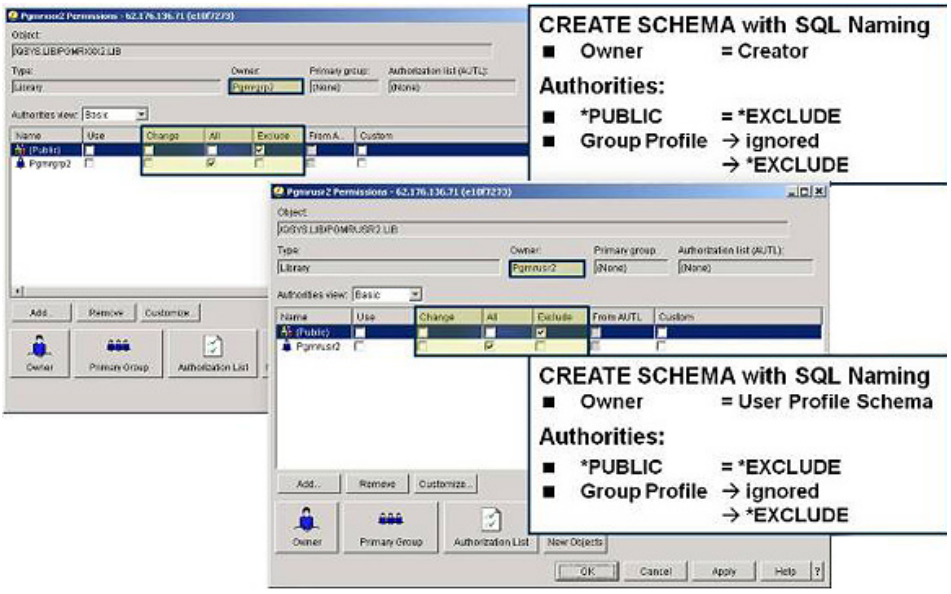
To compare the differences between system and SQL Naming, the schemas previously created with System Naming are dropped and recreated by the same user with SQL Naming.

When comparing the schemas with regard to the ownership and object authorities we will discover several differences:

- The owner of schema PGMRXXX2 is PGMRGRP2, the creator of the schema. The owner setting for the PGMRGRP2 user profile is ignored.
- The owner PGMRGRP2 gets *ALL object authority, while the *PUBLIC object authority is set to *EXCLUDE. Contrary to System Naming where the *PUBLIC object authority depends on the QCRTAUT system value. Consequently, a different developer who is also member of the QPGMR group profile is not allowed to modify the schema or to create an object in this schema. This behavior may be problematic for a company that works intensively with group profiles and where the owner of all objects created by any developer has to become the group profile.
- The owner of schema PGMRUSR2 is PGMRUSR2, because there is an existing user profile with this name. Previously, when creating the schemas using System Naming the owner of both schemas was the QPGMR group profile.
- The owner of the PGMRUSR2 schema, PGMRUSR2, gets *ALL object authority, while the *PUBLIC authority is set to *EXCLUDE. Even though the user PGMRGRP2 was able to create the schema PGMRUSR2, that user does not have any authority on the schema. PGMRGRP2 cannot modify the schema nor create or change any object within this schema.

The following screen shots show the permissions (also known as authorities) for the schemas created with SQL Naming.

Figure 5. CREATE SCHEMA with SQL Naming



Table, Views and Indexes – Objects to maintain Data

Tables are objects to store persistent user data in multiple columns and rows.

Views and indexes are database objects associated with a table but do not contain any data.

Creating Tables, Views and Indexes with System Naming

The rules for determining the ownership and applying object authorities match the rules that are used for creating schemas. The owner is either the creator of the object or the group profile and the *PUBLIC object authority is set to the QCRTAUT system value.

For the next example (Figure 6. CREATE TABLE with System Naming) the table EMPLOYEE is created with System Naming in two different schemas, PGMRUSR and PGMRXXX, using the following SQL statement:

Listing 9. Create Table EMPLOYEE

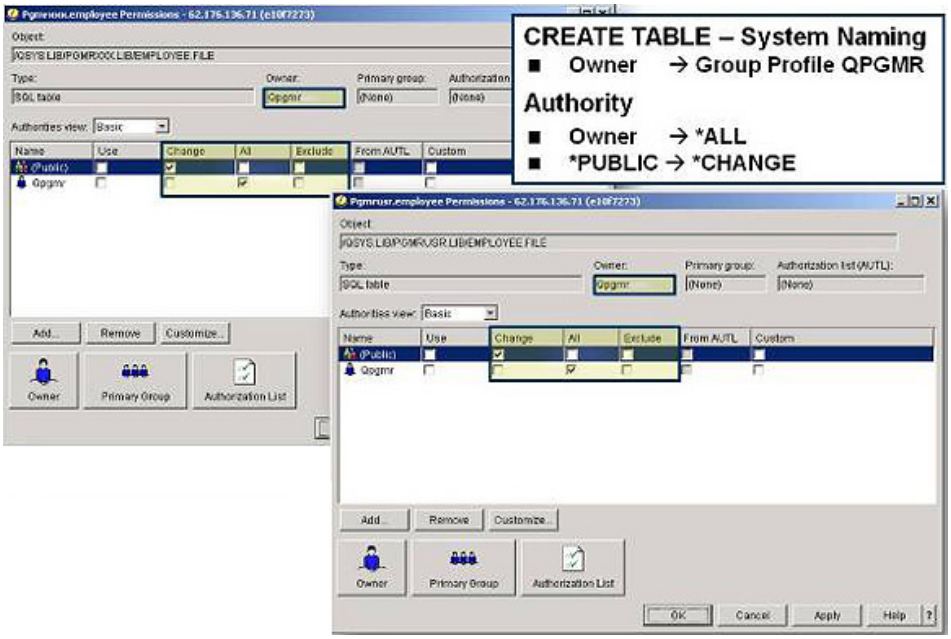
```
Create Table MySchema/Employee (FirstName VarChar(50) Not NULL Default '', Name VarChar(50) Not NULL Default
 '', Street VarChar(50) Not NULL Default '', ZipCode VarChar(15) Not NULL Default '', City VarChar(50) Not
 NULL Default '', Country Char(3) Not NULL Default '', Birthday Date Not NULL);
```

Both schemas were previously created with the CREATE SCHEMA statement using System Naming by the user profile PGMRGRP2. The owner of both schemas is the QPGMR group profile, based on the OWNER setting of the PGMRGRP2 user profile. The group profile is the owner of the table created in schema PGMRUSR even though there is a user profile PGMRUSR.

The owning user profile, QPGMR, has *ALL object authority while the *PUBLIC authority is set to *CHANGE (based on QCRTAUT system value).

Consequently, all users that are associated with the QPGMR group profile are allowed to access, modify and even delete the EMPLOYEE table in both schemas, PGMRXXX and PGMRUSR.

Figure 6. CREATE TABLE with System Naming



Creating Tables, Views and Indexes with SQL Naming

When using SQL Naming different rules apply:

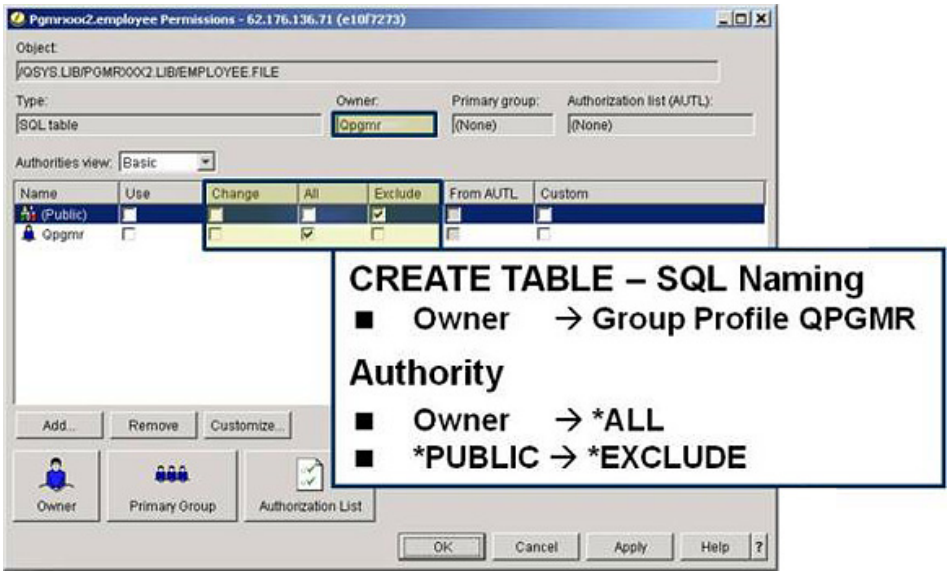
- If a user profile with the same name as the schema into which the table, view or index is created exists, the owner of the table is that user profile.

- If there is no user profile with the same name of the schema, the owner will be either the user profile or group profile depending on the OWNER option setting in the user profile definition.
- When creating database objects other than schemas with SQL Naming, the OWNER option setting in the user profile definition is considered and the group profile will become the owner of the database object.

Figure 7. CREATE TABLE with SQL Naming in a Schema that does not match a User Profile displays the authority results for the EMPLOYEE table created in the schema PGMRXXX2 by user PGMRGRP2.

The owner of the EMPLOYEE table is the QPGMR group profile. The owner QPGMR has a value of *ALL for object authority while the *PUBLIC authority is set to *EXCLUDE. As a result, all users that are associated with the QPGMR group profile are not only allowed to access the EMPLOYEE table, but are also allowed to modify or delete the table.

Figure 7. CREATE TABLE with SQL Naming in a Schema that does not match a User Profile



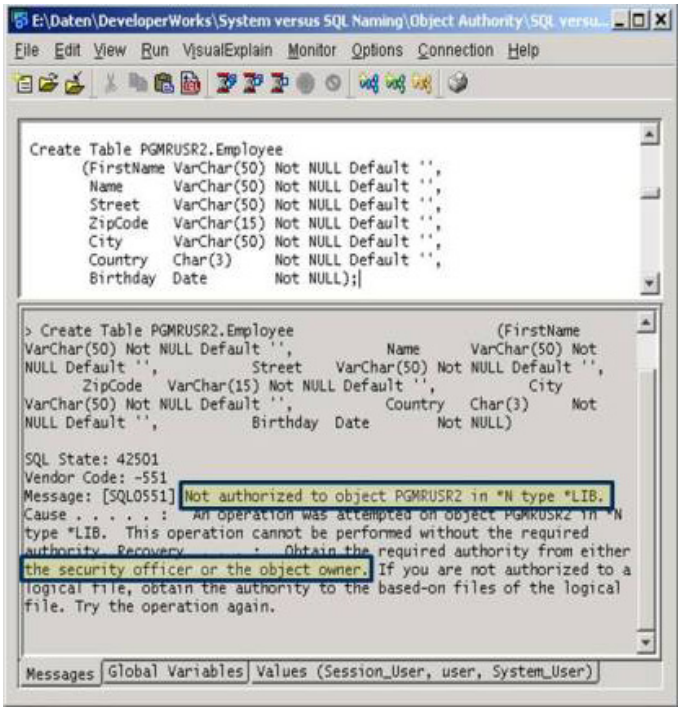
In the next example (Figure 8. CREATE TABLE with SQL Naming in a Schema that does match a User Profile) the user PGMRGRP2 tries to create the EMPLOYEE table in the PGMRUSR2 schema.

The schema was previously created with the CREATE SCHEMA statement with SQL Naming by the user PGMRGRP2. Because there is a user profile named PGMRUSR2 this user profile became the owner of the schema and got *ALL object authority for the schema while *PUBLIC authority was set to *EXCLUDE.

The execution of the CREATE TABLE statement fails with an SQL State value of 24501, because PGMRGRP2 is not authorized to the PGMRUSR2 schema, even though that user created the schema. (Figure 5. CREATE SCHEMA with SQL Naming demonstrates the lack of authority that user PGMRGRP2 has on the PGMRUSR2 schema).

To allow PGMRGRP2 to create a table or any object in the PGMRUSR2 schema with SQL Naming, that user profile or the associated QPGMR group profile must be explicitly authorized to the schema, by executing either the GRTOBJAUT or EDTOBJAUT command.

Figure 8. CREATE TABLE with SQL Naming in a Schema that does match a User Profile

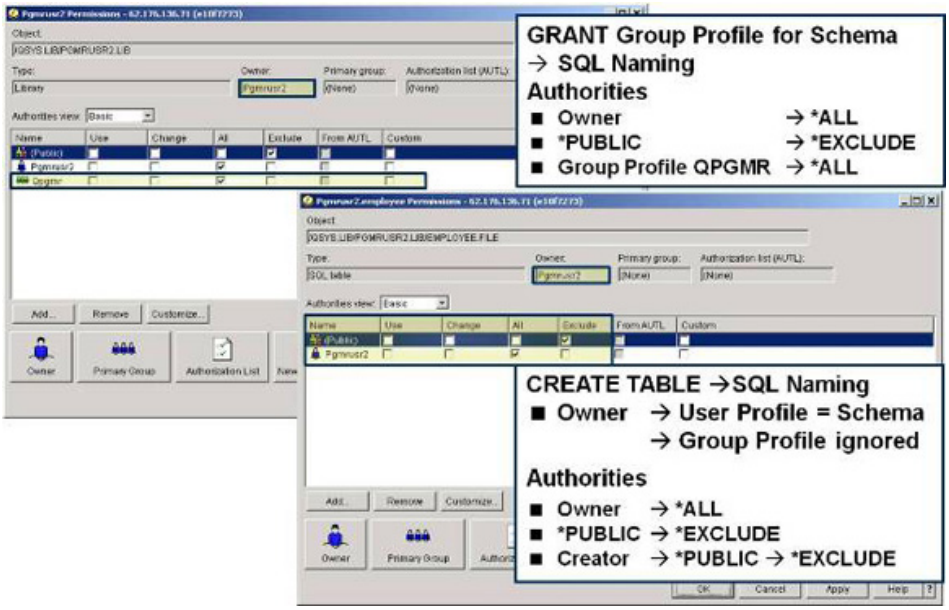


Assuming that the QPGMR group profile was explicitly authorized to the PGMRUSR2 schema, the PGMRGRP2 user would be able to create the EMPLOYEE table in this schema.

Because the table is created using SQL Naming and PGMRUSR2 is an existing user profile, this user profile again becomes the owner of the EMPLOYEE table, with *ALL object authority while the *PUBLIC object authority is set to *EXCLUDE as shown in Figure 9. Permissions for Schema = User Profile and Table.

In this situation, the PGMRGRP2 user is able create the table, but is not allowed to use it in any way. The PGMRGRP2 user or the QPGMR group profile must be explicitly authorized to get access to the object they previously created.

Figure 9. Permissions for Schema = User Profile and Table



Potential Problem Situation

Creating database objects for existing applications using SQL Naming may cause unexpected problems on the IBM i.

Assume that all of the database objects for the existing material management application are stored in a library called MAWI. This library was created long ago with the CRTLIB command. The owner of library MAWI is the QPGMR group profile. *PUBLIC object authority for library MAWI is *CHANGE.

On this system all user profile names are combined of the first 2 characters of the last name and the first 2 characters of the first name. A data entry clerk in the HR department named Willy Maier, so accordingly his user profile is MAWI.

If a developer creates a new table or view in library MAWI using SQL Naming, the owner of this new table will be Willy Maier, because he has a user profile matching the name of the library. Only the MAWI user profile will have access authorities for the new table or view. The developer and any other user are excluded, due to SQL Naming forcing the *PUBLIC access authority to be set to *EXCLUDE by default.

SQL Routines

SQL Routines are executable SQL objects similar to high-level language (HLL) programs. The term SQL Routine is used to refer to a stored procedure, a trigger or a user-defined function (UDF).

These routines are written in either SQL or with a high-level language such as RPG or Cobol. In either case a Stored Procedure or an UDF is created with one of the following SQL statements:

- CREATE PROCEDURE

- CREATE FUNCTION

Ownership and Object Authorities for SQL Routines

The rules for determining the object ownership and authority match the rules that are used for creating tables, views or indexes with either System or SQL Naming.

The SQL statements embedded in the routine are executed based on the naming convention used when **creating the routine**, even though the SQL routine might be invoked in a runtime environment that uses a different naming mode.

For example, a stored procedure was created from an interface where SQL Naming was used. If this stored procedure is called from a RPG program with Embedded SQL where System Naming is used by default, the embedded SQL statements in the RPG program will use System Naming while the SQL statements within the stored procedure are executed with SQL Naming.

The ownership and access authorities of the routine object are only used for calling this routine. The object ownership and authority values may or may not be applied to the SQL statement executed by the routine itself. The authorization-ID (or user profile) that is applied to the SQL requests run by the routine depends on the naming convention used at the time the **routine was created** and whether the SQL statements executed by the routine are **static** or **prepared dynamically**.

When using System Naming DB2 utilizes the user profile who calls the routine. When using SQL Naming to execute static SQL statements within the routine, DB2 uses the routine's owner by default to perform its authorization processing on the static SQL statements.

The user profile that called the routine is always applied by default to the execution of dynamic SQL statements by the routine independent of whether System or SQL Naming is used.

The user profiles applied to the security validation and execution of static and dynamic SQL statements can be manually controlled in the SET OPTION statement by specifying the USRPRF (User Profile for static SQL statements) and DYNUSRPRF (User Profile for dynamic SQL statements) options.

The USERPRF Option can be set to one of the following values:

- *NAMING: *USER is used for System Naming and *OWNER for SQL Naming
- *OWNER: Static SQL Statements are executed with the owner's authorities
- *USER: Static SQL Statements are executed with the user's authorities

Option DYNUSRPRF can be set to:

- *USER: Default value for both System and SQL Naming. Dynamic SQL statements are executed with the user's authorities
- *OWNER: Dynamic SQL statements are executed with the owner's authorities

If you are using SQL Naming and want your dynamic SQL statements being executed by the same user profile as your static SQL statements, you would need to set both of these options, USRPRF and DYNUSRPRF, to either *OWNER or *USER.

The following SQL statement shows the abridged source code for a SQL stored procedure that will be created using SQL Naming. At runtime all of the static and dynamic SQL statements embedded in the procedure will be executed by the *OWNER based on the values specified on the SET OPTION clause.

Listing 10. CREATE PROCEDURE

```
Create Procedure PGMRUSR2.HSINFO (In Parm1 Integer) Dynamic Result Sets 1 Language SQL Set Option DYNUSRPRF =
 *OWNER,USRPRF = *NAMING Begin /* Routine code goes here */ End ;
```

Trigger

Triggers are a special kind of SQL routine. Trigger programs are linked to either a table or a SQL view and are activated by the database manager for a specified event (Insert, Update or Delete).

The ownership of trigger programs is determined in the same way as for all other SQL routines, but the object and execution authorities are set differently by the CREATE TRIGGER statement.

*PUBLIC object authority is set to *EXCLUDE, independent of whether System or SQL Naming is used. For all other SQL objects created with System Naming the *PUBLIC object authority is set to the QCRTAUT system value.

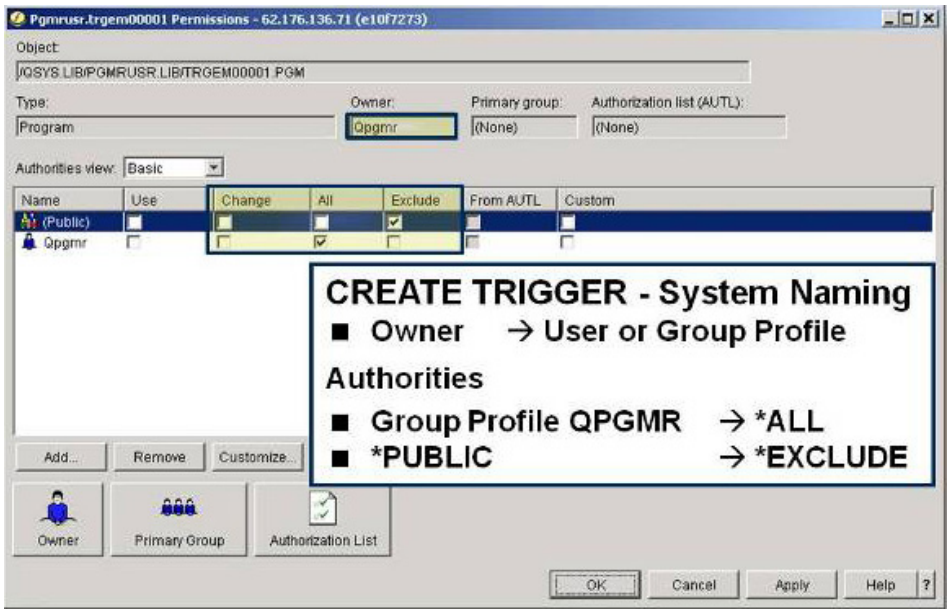
The next example shows the source code for a SQL Trigger to be created with System Naming.

Listing 11. CREATE TRIGGER

```
CREATE TRIGGER PGMRUSR/TRGEMPLOYEE BEFORE INSERT ON PGMRUSR/EMPLOYEE REFERENCING NEW ROW AS N FOR EACH ROW
 MODE DB2ROW BEGIN ATOMIC /* Source code goes here */ END;
```

Figure 10 displays the permission chart for this trigger program created with System Naming. The owner is the QPGMR group profile and the owner has all object authorities while the *PUBLIC object authority is set to *EXCLUDE.

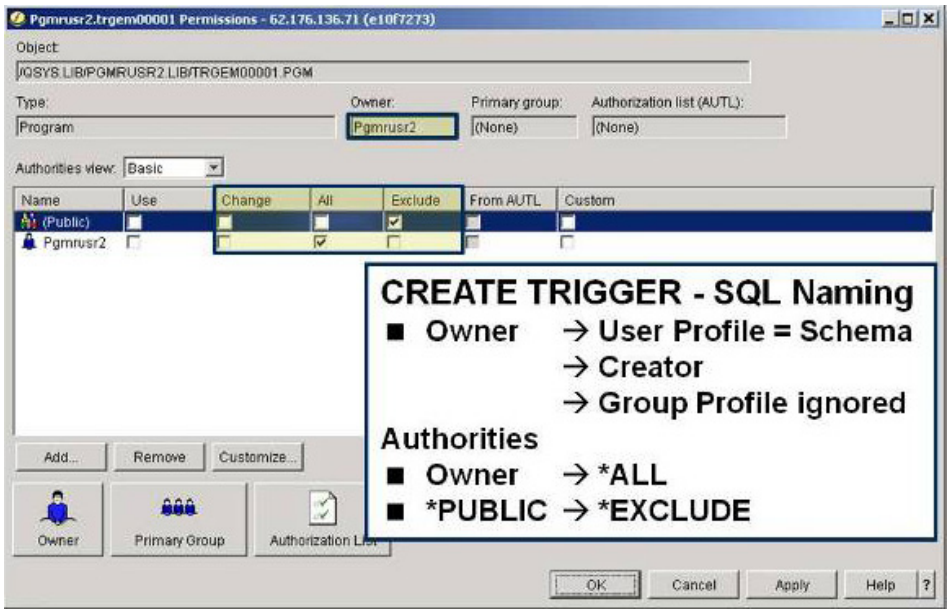
Figure 10. Trigger created with System Naming



To create a trigger program with SQL Naming in a schema with the same name of an existing user profile, the creator must be either explicitly granted to the table or view or have one of the the special authorities *ALLOBJ or *SECADM. The owner of the trigger program will be either the user with the same name as the schema or the creator's user profile or its associated group profile depending on the OWNER option setting in the creator's user profile definition.

The following Figure 11. Trigger created with SQL Naming shows the permission chart for the trigger program created by user PGMRGRP2 into schema PGMRUSR2 using SQL Naming. Because PGMRUSR2 is also an existing user profile, this user profile becomes the owner of the trigger program. The owner PGMRUSR2 has *ALL object authority while *PUBLIC object authority is set to *EXCLUDE.

Figure 11. Trigger created with SQL Naming



The trigger will always be activated with the adopted authority of the owner of the trigger program, independent of the naming convention used to create the trigger.

GRANT / REVOKE Authorities

Whether your database objects are created with System or SQL Naming, the ownership and object authorities must be checked carefully. If the default behavior does not meet your security requirements, the GRANT or REVOKE SQL statements can be used to adjust settings.

Object authorities for any user or group profile and even for *PUBLIC use can be set with the GRANT statement. If object authorities must be removed, the REVOKE statement can be used. The GRANT and REVOKE statements can be used in composition with all database objects that can be accessed or executed with the exception of schemas and triggers.

It is also possible to use the GRTOBJAUT (Grant Object Authority) and EDTOBJAUT (Edit Object Authority) CL commands to modify object authorities for database objects. However, there are some differences in providing authorities with either CL commands or SQL statements.

SET SESSION AUTHORIZATION

The SET SESSION AUTHORIZATION and SET SESSION USER statements can impact object ownership and authorities when working with SQL Naming.

After a connection has been established, the user profile can be switched to a different user profile (authorization id) to adopt the access authorities of this user profile by using the SET SESSION AUTHORIZATION or SET SESSION statements. You have already learned how the user profile value is applied to object ownership and authorities when creating objects with SQL Naming.

Conclusion

You should now have a good understanding of why DB2 objects created with System or SQL Naming have ownership and access authorities assigned differently.

Because of these different behaviors, you should decide on a single naming convention method for all your database objects created with SQL (or at least for all database objects located in a single schema).

- If you intend to design an application that has the ability to run on different database systems, SQL Naming is the right method to achieve maximum portability.
- If you are working only with DB2 for i and have to maintain older applications with a mix of DDS based objects and SQL database objects and use IBM i specific object authorities (such as group profiles), System Naming is the better solution.

And now have fun in planning, designing, creating and maintaining database objects with either system or SQL Naming.

Resources

IBM i - DB2 for i SQL Reference - 7.1

IBM developerWorks DB2 for i - Forum

IBM developerWorks - IBM i Technology Updates