Authorization, privileges, and object ownership

Last Updated: 2024-01-12

Users (identified by an authorization ID) can successfully execute operations only if they have the authority to perform the specified function. To create a table, a user must be authorized to create tables; to alter a table, a user must be authorized to alter the table; and so forth.

The database manager requires that each user be specifically authorized to use each database function needed to perform a specific task. A user can acquire the necessary authorization through a grant of that authorization to their user ID or through membership in a role or a group that holds that authorization.

There are three forms of authorization, *administrative authority*, *privileges,* and *LBAC credentials*. In addition, ownership of objects brings with it a degree of authorization on the objects created. These forms of authorization are discussed in the following section.

Administrative authority 🔗

The person or persons holding administrative authority are charged with the task of controlling the database manager and are responsible for the safety and integrity of the data.

System-level authorization

The system-level authorities provide varying degrees of control over instance-level functions:

– SYSADM (system administrator) authority

The SYSADM (system administrator) authority provides control over all the resources created and maintained by the database manager. The system administrator possesses all the authorities of SYSCTRL, SYSMAINT, and SYSMON authority. The user who has SYSADM authority is responsible both for controlling the database manager, and for ensuring the safety and integrity of the data.

– SYSCTRL authority

The SYSCTRL authority provides control over operations that affect system resources. For example, a user with SYSCTRL authority can create, update, start, stop, or drop a database. This user can also start or stop an instance, but cannot access table data. Users with SYSCTRL authority also have the SYSMAINT and SYSMON authorities.

– SYSMAINT authority

The SYSMAINT authority provides the authority required to perform maintenance operations on all databases associated with an instance. ==A user with SYSMAINT authority can update the== ==database configuration, backup a database or table space, restore an existing database, and== ==monitor a database==. Like SYSCTRL, SYSMAINT does not provide access to table data. Users with SYSMAINT authority also have SYSMON authority.

9.4

– SYSMON (system monitor) authority

9.4 | The SYSMON (system monitor) authority provides the authority required to use the database system monitor.

Database-level authorization

The database level authorities provide control within the database:

– DBADM (database administrator)

The DBADM authority level provides administrative authority over a single database. This database administrator possesses the privileges required to create objects and issue database commands.

The DBADM authority can be granted only by a user with SECADM authority. The DBADM authority cannot be granted to PUBLIC.

9.3

– SECADM (security administrator)

The SECADM authority level provides administrative authority for security over a single database. The security administrator authority possesses the ability to manage database security objects (database roles, audit policies, trusted contexts, security label components, and security labels) and grant and revoke all database privileges and authorities. A user with SECADM authority can transfer the ownership of objects that they do not own. They can also use the AUDIT statement to associate an audit policy with a particular database or database object at the server.

The SECADM authority has no inherent privilege to access data stored in tables. It can only be granted by a user with SECADM authority. The SECADM authority cannot be granted to PUBLIC.

– SQLADM (SQL administrator)

The SQLADM authority level provides administrative authority to monitor and tune SQL statements within a single database. It can be granted by a user with ACCESSCTRL or SECADM authority.

– WLMADM (workload management administrator)

The WLMADM authority provides administrative authority to manage workload management objects, such as service classes, work action sets, work class sets, and workloads. It can be granted by a user with ACCESSCTRL or SECADM authority.

– EXPLAIN (explain authority)

The EXPLAIN authority level provides administrative authority to explain query plans without gaining access to data. It can only be granted by a user with ACCESSCTRL or SECADM authority.

– ACCESSCTRL (database access control authority) | 9.3

The ACCESSCTRL authority level provides administrative authority to issue the following GRANT (and REVOKE) statements.

▪ GRANT (Database Authorities)

ACCESSCTRL authority does not give the holder the ability to grant ACCESSCTRL, DATAACCESS, DBADM, or SECADM authority. Only a user who has SECADM authority can grant these authorities.

- GRANT (Global Variable Privileges)
- GRANT (Index Privileges)
- GRANT (Module Privileges)
- GRANT (Package Privileges)
- GRANT (Routine Privileges)
- GRANT (Schema Privileges)
- GRANT (Sequence Privileges)
- GRANT (Server Privileges)
- GRANT (Table, View, or Nickname Privileges)
- GRANT (Table Space Privileges)
- GRANT (Workload Privileges)
- GRANT (XSR Object Privileges)

ACCESSCTRL authority can only be granted by a user with SECADM authority. The ACCESSCTRL authority cannot be granted to PUBLIC.

- DATAACCESS (database data access authority)

The DATAACCESS authority level provides the following privileges and authorities.

- LOAD authority
- SELECT, INSERT, UPDATE, DELETE privilege on tables, views, nicknames, and materialized query tables
- EXECUTE privilege on packages
- EXECUTE privilege on modules
- EXECUTE privilege on routines
  Except on the audit routines: AUDIT_ARCHIVE, AUDIT_LIST_LOGS, AUDIT_DELIM_EXTRACT.

- READ privilege on all global variables and WRITE privilege on all global variables except variables which are read-only
- USAGE privilege on all XSR objects
- USAGE privilege on all sequences

It can be granted only by a user who holds SECADM authority. The DATAACCESS authority cannot be granted to PUBLIC.

- Database authorities (non-administrative)

To perform activities such as creating a table or a routine, or for loading data into a table, specific database authorities are required. For example, the LOAD database authority is required for use of the `load` utility to load data into tables (a user must also have the privilege to insert data into the table).

Schema-level authorization

The schema-level authorities provide control over a schema. They cannot be granted to PUBLIC or be granted with the WITH GRANT OPTION.

- SCHEMAADM (Schema administrator)

The SCHEMADM authority provides administrative authority over a single schema. The schema administrator has the privilege to create and manage objects in the schema. It implicitly has the schema LOAD authority.

- ACCESSCTRL (Schema access control authority)
  The schema ACCESSCTRL authority provides the ability to grant and revoke all privileges on objects defined in the schema. The schema ACCESSCTRL authority can also grant and revoke all schema-level authorities and privileges on the schema except schema ACCESSCTRL itself.

- DATAACCESS (Schema data access authority)
  The schema DATAACCESS authority allows users to read and write data on all objects existing in the schema. The authority also gives users the permission to reference sequences and xsrobjects, execute routines, modules, and packages defined the schema, and implicit schema LOAD authority.

- LOAD (Schema load authority)
  The schema LOAD authority allows users to use the load utility to load data into tables defined in the schema (the user must also have the privilege to insert data into the table).

Privileges 🔗

A privilege is a permission to perform an action or a task. Authorized users can create objects, have access to objects they own, and can pass on privileges on their own objects to other users by using the GRANT statement.

Privileges may be granted to individual users, to groups, or to PUBLIC. PUBLIC is a special group that consists of all users, including future users. Users that are members of a group will indirectly take advantage of the privileges granted to the group, where groups are supported.

The CONTROL privilege: Possessing the CONTROL privilege on an object allows a user to access that database object, and to grant and revoke privileges to or from other users on that object.

> ℹ️ **Note:** The CONTROL privilege only applies to tables, views, nicknames, indexes, and packages.

If a different user requires the CONTROL privilege to that object, a user with SECADM or ACCESSCTRL authority could grant the CONTROL privilege to that object. The CONTROL privilege cannot be revoked from the object owner, however, the object owner can be changed by using the TRANSFER OWNERSHIP statement.

Individual privileges: Individual privileges can be granted to allow a user to carry out specific tasks on specific objects. Users with the administrative authorities ACCESSCTRL or SECADM, or with the CONTROL privilege, can grant and revoke privileges to and from users.

Individual privileges and database authorities allow a specific function, but do not include the right to grant the same privileges or authorities to other users. The right to grant table, view, schema, package, routine, and sequence privileges to others can be extended to other users through the WITH GRANT OPTION on the GRANT statement. However, the WITH GRANT OPTION does not allow the person granting the privilege to revoke the privilege once granted. You must have SECADM authority, ACCESSCTRL authority, or the CONTROL privilege to revoke the privilege.

Privileges on objects in a package or routine: When a user has the privilege to execute a package or routine, they do not necessarily require specific privileges on the objects used in the package or routine. If the package or routine contains static SQL or XQuery statements, the privileges of the owner of the package are used for those statements. If the package or routine contains dynamic SQL or XQuery statements, the authorization ID used for privilege checking depends on the setting of the **DYNAMICRULES BIND** option of the package issuing the dynamic query statements, and whether those statements are issued when the package is being used in the context of a routine (except on the audit routines: AUDIT_ARCHIVE, AUDIT_LIST_LOGS, AUDIT_DELIM_EXTRACT).

A user or group can be authorized for any combination of individual privileges or authorities. When a privilege is associated with an object, that object must exist. For example, a user cannot be given the SELECT privilege on a table unless that table has previously been created.

> ℹ️ **Note:** Care must be taken when an authorization name representing a user or a group is granted authorities and privileges and there is no user, or group created with that name. At some later time, a user or a group can be created with that name and automatically receive all of the authorities and privileges associated with that authorization name.

The REVOKE statement is used to revoke previously granted privileges. The revoking of a privilege from an authorization name revokes the privilege granted by all authorization names.

Revoking a privilege from an authorization name does not revoke that same privilege from any other authorization names that were granted the privilege by that authorization name. For example, assume that CLAIRE grants SELECT WITH GRANT OPTION to RICK, then RICK grants SELECT to BOBBY and CHRIS. If CLAIRE revokes the SELECT privilege from RICK, BOBBY and CHRIS still retain the SELECT privilege.

LBAC credentials 🔗

Label-based access control (LBAC) lets the security administrator decide exactly who has write access and who has read access to individual rows and individual columns. The security administrator configures the LBAC system by creating security policies. A security policy describes the criteria used to decide who has access to what data. Only one security policy can be used to protect any one table but different tables can be protected by different security policies.

After creating a security policy, the security administrator creates database objects, called security labels and exemptions that are part of that policy. A security label describes a certain set of security criteria. An exemption allows a rule for comparing security labels not to be enforced for the user who holds the exemption, when they access data protected by that security policy.

Once created, a security label can be associated with individual columns and rows in a table to protect the data held there. Data that is protected by a security label is called protected data. A security administrator allows users access to protected data by granting them security labels. When a user tries to access protected data, that user's security label is compared to the security label protecting the data. The protecting label blocks some security labels and does not block others.

Object ownership 🔗

When an object is created, one authorization ID is assigned *ownership* of the object. Ownership means the user is authorized to reference the object in any applicable SQL or XQuery statement.

When an object is created within a schema, the authorization ID of the statement must have the required privilege to create objects in the implicitly or explicitly specified schema. That is, the authorization name must either be the owner of the schema, or possess the CREATEIN privilege on the schema.

> ℹ **Note:** This requirement is not applicable when creating table spaces, buffer pools or database partition groups. These objects are not created in schemas.

When an object is created, the authorization ID of the statement is the definer of that object and by default becomes the owner of the object after it is created.

> ℹ **Note:** One exception exists. If the AUTHORIZATION option is specified for the CREATE SCHEMA statement, any other object that is created as part of the CREATE SCHEMA operation is owned by the authorization ID specified by the AUTHORIZATION option. Any objects that are created in the schema after the initial CREATE SCHEMA operation, however, are owned by the authorization ID associated with the specific CREATE statement.
> For example, the statement `CREATE SCHEMA SCOTTSTUFF AUTHORIZATION SCOTT CREATE TABLE T1 (C1 INT)` creates the schema SCOTTSTUFF and the table `SCOTTSTUFF.T1`, which are both owned by SCOTT. Assume that the user BOBBY is granted the CREATEIN privilege on the SCOTTSTUFF schema and creates an index on the `SCOTTSTUFF.T1` table. Because the index is created after the schema, BOBBY owns the index on `SCOTTSTUFF.T1`.

Privileges are assigned to the object owner based on the type of object being created:
– The CONTROL privilege is implicitly granted on newly created tables, indexes, and packages. This privilege allows the object creator to access the database object, and to grant and revoke privileges to or from other users on that object. If a different user requires the CONTROL privilege to that object, a user with ACCESSCTRL or SECADM authority must grant the CONTROL privilege to that object. The CONTROL privilege cannot be revoked by the object owner.
– The CONTROL privilege is implicitly granted on newly created views if the object owner has the CONTROL privilege on all the tables, views, and nicknames referenced by the view definition.
– Other objects like triggers, routines, sequences, table spaces, and buffer pools do not have a CONTROL privilege associated with them. The object owner does, however, automatically receive each of the privileges associated with the object and those privileges are with the WITH GRANT OPTION, where supported. Therefore the object owner can provide these privileges to other users by using the GRANT statement. For example, if USER1 creates a table space, USER1 automatically has the USEAUTH privilege with the WITH GRANT OPTION on this table space and can grant the

USEAUTH privilege to other users. In addition, the object owner can alter, add a comment on, or drop the object. These authorizations are implicit for the object owner and cannot be revoked. Certain privileges on the object, such as altering a table, can be granted by the owner, and can be revoked from the owner by a user who has ACCESSCTRL or SECADM authority. Certain privileges on the object, such as commenting on a table, cannot be granted by the owner and cannot be revoked from the owner. Use the TRANSFER OWNERSHIP statement to move these privileges to another user. When an object is created, the authorization ID of the statement is the definer of that object and by default becomes the owner of the object after it is created. However, when you use the **BIND** command to create a package and you specify the **OWNER** *authorization id* option, the owner of objects created by the static SQL statements in the package is the value of *authorization id*. In addition, if the AUTHORIZATION clause is specified on a CREATE SCHEMA statement, the authorization name specified after the AUTHORIZATION keyword is the owner of the schema.

A security administrator or the object owner can use the TRANSFER OWNERSHIP statement to change the ownership of a database object. An administrator can therefore create an object on behalf of an authorization ID, by creating the object using the authorization ID as the qualifier, and then using the TRANSFER OWNERSHIP statement to transfer the ownership that the administrator has on the object to the authorization ID.

- **Authorities overview**
  Various administrative authorities exist at the instance level, at the database level, and at the schema level. These administrative authorities group together certain privileges and authorities so that you ca grant them to the users who are responsible for these tasks in your database installation.
- **Internal system-defined routine**
- **Database authorities**
  Each database authority allows the authorization ID holding it to perform some particular type of action on the database as a whole. Database authorities are different from privileges, which allow a certain action to be taken on a particular database object, such as a table or an index.
- **Schema authorities**
  Schema authorities are designed to allow an authorization ID to perform certain duties on a schema, i.e., schema administrator, schema access control manager, and schema data administrator. Schema authorities have been designed on the same principle as the database authorities but are different from them as their scope is limited only to the schema on which they are granted. Schema authorities are also different from privileges, which allow a certain action to be taken on a particular schema or schema object, such as a table or an index.
- **Authorization IDs in different contexts**
  An authorization ID is used for two purposes: identification and authorization checking. For example, the session authorization ID is used for initial authorization checking.
- **Default privileges granted on creating a database**
  When you create a database, default database level authorities and default object level privileges are granted to you within that database.
- **Default PUBLIC privilege for built-in routines**
  When a non-restrictive database is created, the special group PUBLIC is granted EXECUTE with GRAN to the majority of built-in routines.

– **Controlling access for database administrators (DBAs)**
  You may want to monitor, control, or prevent access to data by database administrators (users holdir DBADM authority).

– **Gaining access to data through indirect means**
  To successfully manage security, you need to be aware of indirect ways that users can gain access to data.

Parent topic: **Related reference**

→ Db2 security model overview   → GRANT (database authorities) statement