GRANT

Name

GRANT -- define access privileges

Synopsis

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | RULE | REFERENCES | TRIGGER }
    [,...] | ALL [ PRIVILEGES ] }
    ON [ TABLE ] tablename [, ...]
    TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | TEMPORARY | TEMP } [,...] | ALL [ PRIVILEGES ] }
    ON DATABASE dbname [, ...]
    TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON FUNCTION funcname ([type, ...]) [, ...]
    TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON LANGUAGE langname [, ...]
    TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | USAGE } [,...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schemaname [, ...]
    TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

Description

The `GRANT` command gives specific privileges on an object (table, view, sequence, database, function, procedural language, or schema) to one or more users or groups of users. These privileges are added to those already granted, if any.

The key word `PUBLIC` indicates that the privileges are to be granted to all users, including those that may be created later. `PUBLIC` may be thought of as an implicitly defined group that always includes all users. Any particular user will have the sum of privileges granted directly to him, privileges granted to any group he is presently a member of, and privileges granted to `PUBLIC`.

If `WITH GRANT OPTION` is specified, the recipient of the privilege may in turn grant it to others. By default this is not allowed. Grant options can only be granted to individual users, not to groups or `PUBLIC`.

There is no need to grant privileges to the owner of an object (usually the user that created it), as the owner has all privileges by default. (The owner could, however, choose to revoke some of his own privileges for safety.) The right to drop an object, or to alter its definition in any way is not described by a grantable

privilege; it is inherent in the owner, and cannot be granted or revoked. It is not possible for the owner's grant options to be revoked, either.

Depending on the type of object, the initial default privileges may include granting some privileges to `PUBLIC`. The default is no public access for tables and schemas; `TEMP` table creation privilege for databases; `EXECUTE` privilege for functions; and `USAGE` privilege for languages. The object owner may of course revoke these privileges. (For maximum security, issue the `REVOKE` in the same transaction that creates the object; then there is no window in which another user may use the object.)

The possible privileges are:

SELECT

Allows *SELECT* from any column of the specified table, view, or sequence. Also allows the use of *COPY* TO. For sequences, this privilege also allows the use of the `currval` function.

INSERT

Allows *INSERT* of a new row into the specified table. Also allows *COPY* FROM.

UPDATE

Allows *UPDATE* of any column of the specified table. `SELECT ... FOR UPDATE` also requires this privilege (besides the `SELECT` privilege). For sequences, this privilege allows the use of the `nextval` and `setval` functions.

DELETE

Allows *DELETE* of a row from the specified table.

RULE

Allows the creation of a rule on the table/view. (See *CREATE RULE* statement.)

REFERENCES

To create a foreign key constraint, it is necessary to have this privilege on both the referencing and referenced tables.

TRIGGER

Allows the creation of a trigger on the specified table. (See *CREATE TRIGGER* statement.)

CREATE

For databases, allows new schemas to be created within the database.

For schemas, allows new objects to be created within the schema. To rename an existing object, you must own the object *and* have this privilege for the containing schema.

TEMPORARY
TEMP

Allows temporary tables to be created while using the database.

EXECUTE

Allows the use of the specified function and the use of any operators that are implemented on top of the function. This is the only type of privilege that is applicable to functions. (This syntax works for aggregate functions, as well.)

USAGE

For procedural languages, allows the use of the specified language for the creation of functions in that language. This is the only type of privilege that is applicable to procedural languages.

For schemas, allows access to objects contained in the specified schema (assuming that the objects' own privilege requirements are also met). Essentially this allows the grantee to "look up" objects within the schema.

ALL PRIVILEGES

Grant all of the privileges applicable to the object at once. The `PRIVILEGES` key word is optional in PostgreSQL, though it is required by strict SQL.

The privileges required by other commands are listed on the reference page of the respective command.

Notes

The *REVOKE* command is used to revoke access privileges.

It should be noted that database superusers can access all objects regardless of object privilege settings. This is comparable to the rights of `root` in a Unix system. As with `root`, it's unwise to operate as a superuser except when absolutely necessary.

If a superuser chooses to issue a `GRANT` or `REVOKE` command, the command is performed as though it were issued by the owner of the affected object. In particular, privileges granted via such a command will appear to have been granted by the object owner.

Currently, to grant privileges in PostgreSQL to only a few columns, you must create a view having the desired columns and then grant privileges to that view.

Use psql's `\z` command to obtain information about existing privileges, for example:

```
=> \z mytable

        Access privileges for database "lusitania"
 Schema |  Table  |           Access privileges
--------+---------+---------------------------------------
 public | mytable | {=r/postgres,miriam=arwdRxt/postgres,"group todos=arw/postgres"}
(1 row)
```

The entries shown by `\z` are interpreted thus:

```
              =xxxx -- privileges granted to PUBLIC
           uname=xxxx -- privileges granted to a user
```

```
group gname=xxxx -- privileges granted to a group

                r -- SELECT ("read")
                w -- UPDATE ("write")
                a -- INSERT ("append")
                d -- DELETE
                R -- RULE
                x -- REFERENCES
                t -- TRIGGER
                X -- EXECUTE
                U -- USAGE
                C -- CREATE
                T -- TEMPORARY
           arwdRxt -- ALL PRIVILEGES (for tables)
                * -- grant option for preceding privilege

             /yyyy -- user who granted this privilege
```

The above example display would be seen by user `miriam` after creating table `mytable` and doing

```
GRANT SELECT ON mytable TO PUBLIC;
GRANT SELECT, UPDATE, INSERT ON mytable TO GROUP todos;
```

If the "Access privileges" column is empty for a given object, it means the object has default privileges (that is, its privileges column is null). Default privileges always include all privileges for the owner, and may include some privileges for `PUBLIC` depending on the object type, as explained above. The first `GRANT` or `REVOKE` on an object will instantiate the default privileges (producing, for example, `{=,miriam=arwdRxt}`) and then modify them per the specified request.

Examples

Grant insert privilege to all users on table `films`:

```
GRANT INSERT ON films TO PUBLIC;
```

Grant all privileges to user `manuel` on view `kinds`:

```
GRANT ALL PRIVILEGES ON kinds TO manuel;
```

Compatibility

According to the SQL standard, the `PRIVILEGES` key word in `ALL PRIVILEGES` is required. The SQL standard does not support setting the privileges on more than one object per command.

The SQL standard allows setting privileges for individual columns within a table:

```
GRANT privileges
    ON table [ ( column [, ...] ) ] [, ...]
    TO { PUBLIC | username [, ...] } [ WITH GRANT OPTION ]
```

The SQL standard provides for a `USAGE` privilege on other kinds of objects: character sets, collations, translations, domains.

The `RULE` privilege, and privileges on databases, schemas, languages, and sequences are PostgreSQL

extensions.

See Also

REVOKE
