5.7. Privileges

When an object is created, it is assigned an owner. The owner is normally the role that executed the creation statement. For most kinds of objects, the initial state is that only the owner (or a superuser) can do anything with the object. To allow other roles to use it, *privileges* must be granted.

There are different kinds of privileges: `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE`, `REFERENCES`, `TRIGGER`, `CREATE`, `CONNECT`, `TEMPORARY`, `EXECUTE`, `USAGE`, `SET` and `ALTER SYSTEM`. The privileges applicable to a particular object vary depending on the object's type (table, function, etc.). More detail about the meanings of these privileges appears below. The following sections and chapters will also show you how these privileges are used.

The right to modify or destroy an object is inherent in being the object's owner, and cannot be granted or revoked in itself. (However, like all privileges, that right can be inherited by members of the owning role; see Section 22.3.)

An object can be assigned to a new owner with an `ALTER` command of the appropriate kind for the object, for example

```
ALTER TABLE table_name OWNER TO new_owner;
```

Superusers can always do this; ordinary roles can only do it if they are both the current owner of the object (or a member of the owning role) and a member of the new owning role.

To assign privileges, the GRANT command is used. For example, if `joe` is an existing role, and `accounts` is an existing table, the privilege to update the table can be granted with:

```
GRANT UPDATE ON accounts TO joe;
```

Writing `ALL` in place of a specific privilege grants all privileges that are relevant for the object type.

The special “role” name `PUBLIC` can be used to grant a privilege to every role on the system. Also, “group” roles can be set up to help manage privileges when there are many users of a database — for details see Chapter 22.

To revoke a previously-granted privilege, use the fittingly named REVOKE command:

```
REVOKE ALL ON accounts FROM PUBLIC;
```

Ordinarily, only the object's owner (or a superuser) can grant or revoke privileges on an object. However, it is possible to grant a privilege “with grant option”, which gives the recipient the right to grant it in turn to others. If the grant option is subsequently revoked then all who received the privilege from that recipient (directly or through a chain of grants) will lose the privilege. For details see the GRANT and REVOKE reference pages.

An object's owner can choose to revoke their own ordinary privileges, for example to make a table read-only for themselves as well as others. But owners are always treated as holding all grant options, so they can always re-grant their own privileges.

The available privileges are:

`SELECT`

> Allows `SELECT` from any column, or specific column(s), of a table, view, materialized view, or other table-like object. Also allows use of `COPY TO`. This privilege is also needed to reference existing column values in `UPDATE` or `DELETE`. For sequences, this privilege also allows use of the `currval` function. For large objects, this privilege allows the object to be read.

`INSERT`

> Allows `INSERT` of a new row into a table, view, etc. Can be granted on specific column(s), in which case only those columns may be assigned to in the `INSERT` command (other columns will therefore receive default values). Also allows use of `COPY FROM`.

`UPDATE`

> Allows `UPDATE` of any column, or specific column(s), of a table, view, etc. (In practice, any nontrivial `UPDATE` command will require `SELECT` privilege as well, since it must reference table columns to determine which rows to update, and/or to compute new values for columns.) `SELECT ... FOR UPDATE` and `SELECT ... FOR SHARE` also require this privilege on at least one column, in addition to the `SELECT` privilege. For sequences, this privilege allows use of the `nextval` and `setval` functions. For large objects, this privilege allows writing or truncating the object.

`DELETE`

> Allows `DELETE` of a row from a table, view, etc. (In practice, any nontrivial `DELETE` command will require `SELECT` privilege as well, since it must reference table columns to determine which rows to delete.)

`TRUNCATE`

> Allows `TRUNCATE` on a table.

`REFERENCES`

> Allows creation of a foreign key constraint referencing a table, or specific column(s) of a table.

`TRIGGER`

> Allows creation of a trigger on a table, view, etc.

`CREATE`

> For databases, allows new schemas and publications to be created within the database, and allows trusted extensions to be installed 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.
>
> For tablespaces, allows tables, indexes, and temporary files to be created within the tablespace, and allows databases to be created that have the tablespace as their default tablespace.
>
> Note that revoking this privilege will not alter the existence or location of existing objects.

`CONNECT`

> Allows the grantee to connect to the database. This privilege is checked at connection startup (in addition to checking any restrictions imposed by `pg_hba.conf`).

`TEMPORARY`

> Allows temporary tables to be created while using the database.

`EXECUTE`

> Allows calling a function or procedure, including use of any operators that are implemented on top of the function. This is the only type of privilege that is applicable to functions and procedures.

`USAGE`

> For procedural languages, allows use of the 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 schema (assuming that the objects' own privilege requirements are also met). Essentially this allows the grantee to “look up” objects within the schema. Without this permission, it is still possible to see the object names, e.g., by querying system catalogs. Also, after revoking this permission, existing sessions might have statements that have previously performed this lookup, so this is not a completely secure way to prevent object access.
>
> For sequences, allows use of the `currval` and `nextval` functions.
>
> For types and domains, allows use of the type or domain in the creation of tables, functions, and other schema objects. (Note that this privilege does not control all “usage” of the type, such as values of the type appearing in queries. It only prevents objects from being created that depend on the type. The main purpose of this privilege is controlling which users can create dependencies on a type, which could prevent the owner from changing the type later.)
>
> For foreign-data wrappers, allows creation of new servers using the foreign-data wrapper.
>
> For foreign servers, allows creation of foreign tables using the server. Grantees may also create, alter, or drop their own user mappings associated with that server.

`SET`

> Allows a server configuration parameter to be set to a new value within the current session. (While this privilege can be granted on any parameter, it is meaningless except for parameters that would normally require superuser privilege to set.)

`ALTER SYSTEM`

> Allows a server configuration parameter to be configured to a new value using the ALTER SYSTEM command.

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

PostgreSQL grants privileges on some types of objects to `PUBLIC` by default when the objects are created. No privileges are granted to `PUBLIC` by default on tables, table columns, sequences, foreign data wrappers, foreign servers, large objects, schemas, tablespaces, or configuration parameters. For other types of objects, the default privileges granted to `PUBLIC` are as follows: `CONNECT` and `TEMPORARY` (create temporary tables) privileges for databases; `EXECUTE` privilege for functions and procedures; and `USAGE` privilege for languages and data types (including domains). The object owner can, of course, `REVOKE` both default and expressly granted privileges. (For maximum security, issue the `REVOKE` in the same transaction that creates the object; then there is no window in which another user can use the object.) Also, these default privilege settings can be overridden using the ALTER DEFAULT PRIVILEGES command.

Table 5.1 shows the one-letter abbreviations that are used for these privilege types in *ACL* (Access Control List) values. You will see these letters in the output of the psql commands listed below, or when looking at ACL columns of system catalogs.

Table 5.1. ACL Privilege Abbreviations

Privilege	Abbreviation	Applicable Object Types
`SELECT`	`r` (“read”)	`LARGE OBJECT`, `SEQUENCE`, `TABLE` (and table-like objects), table column
`INSERT`	`a` (“append”)	`TABLE`, table column
`UPDATE`	`w` (“write”)	`LARGE OBJECT`, `SEQUENCE`, `TABLE`, table column
`DELETE`	`d`	`TABLE`
`TRUNCATE`	`D`	`TABLE`
`REFERENCES`	`x`	`TABLE`, table column
`TRIGGER`	`t`	`TABLE`
`CREATE`	`C`	`DATABASE`, `SCHEMA`, `TABLESPACE`
`CONNECT`	`c`	`DATABASE`
`TEMPORARY`	`T`	`DATABASE`
`EXECUTE`	`X`	`FUNCTION`, `PROCEDURE`
`USAGE`	`U`	`DOMAIN`, `FOREIGN DATA WRAPPER`, `FOREIGN SERVER`, `LANGUAGE`, `SCHEMA`, `SEQUENCE`, `TYPE`
`SET`	`s`	`PARAMETER`
`ALTER SYSTEM`	`A`	`PARAMETER`

Table 5.2 summarizes the privileges available for each type of SQL object, using the abbreviations shown above. It also shows the psql command that can be used to examine privilege settings for each object type.

Table 5.2. Summary of Access Privileges

Object Type	All Privileges	Default `PUBLIC` Privileges	psql Command
`DATABASE`	`CTc`	`Tc`	`\l`
`DOMAIN`	`U`	`U`	`\dD+`
`FUNCTION` or `PROCEDURE`	`X`	`X`	`\df+`
`FOREIGN DATA WRAPPER`	`U`	none	`\dew+`
`FOREIGN SERVER`	`U`	none	`\des+`
`LANGUAGE`	`U`	`U`	`\dL+`
`LARGE OBJECT`	`rw`	none	`\dl+`
`PARAMETER`	`sA`	none	`\dconfig+`
`SCHEMA`	`UC`	none	`\dn+`
`SEQUENCE`	`rwU`	none	`\dp`
`TABLE` (and table-like objects)	`arwdDxt`	none	`\dp`
Table column	`arwx`	none	`\dp`
`TABLESPACE`	`C`	none	`\db+`
`TYPE`	`U`	`U`	`\dT+`

The privileges that have been granted for a particular object are displayed as a list of `aclitem` entries, where each `aclitem` describes the permissions of one grantee that have been granted by a particular grantor. For example, `calvin=r*w/hobbes` specifies that the role `calvin` has the privilege `SELECT` (`r`) with grant option (`*`) as well as the non-grantable privilege `UPDATE` (`w`), both granted by the role `hobbes`. If `calvin` also has some privileges on the same object granted by a different grantor, those would appear as a separate `aclitem` entry. An empty grantee field in an `aclitem` stands for `PUBLIC`.

As an example, suppose that user `miriam` creates table `mytable` and does:

```
GRANT SELECT ON mytable TO PUBLIC;
GRANT SELECT, UPDATE, INSERT ON mytable TO admin;
GRANT SELECT (col1), UPDATE (col1) ON mytable TO miriam_rw;
```

Then psql's `\dp` command would show:

```
=> \dp mytable
                                  Access privileges
 Schema |  Name   | Type  |   Access privileges   |   Column privileges   | Polic
--------+---------+-------+-----------------------+-----------------------+------
 public | mytable | table | miriam=arwdDxt/miriam+| col1:                +|
        |         |       | =r/miriam            +|   miriam_rw=rw/miriam |
        |         |       | admin=arw/miriam      |                       |
(1 row)
```

If the “Access privileges” column is empty for a given object, it means the object has default privileges (that is, its privileges entry in the relevant system catalog is null). Default privileges always include all privileges for the owner, and can 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=arwdDxt/miriam`) and then modify them per the specified request. Similarly, entries are shown in “Column privileges” only for columns with nondefault privileges. (Note: for this purpose, “default privileges” always means the built-in default privileges for the object's type. An object whose privileges have been affected by an `ALTER DEFAULT PRIVILEGES` command will always be shown with an explicit privilege entry that includes the effects of the `ALTER`.)

Notice that the owner's implicit grant options are not marked in the access privileges display. A `*` will appear only when grant options have been explicitly granted to someone.

Submit correction

If you see anything in the documentation that is not correct, does not match your experience with the particular feature or requires further clarification, please use this form to report a documentation issue.