SQL\_DCL\_Gabriela\_Cretu\_HW\_Task4

Gabriela Creţu

April 2025

* Task 4. Prepare answers to the following questions

1.1 How can one restrict access to certain columns of a database table?

In PostgreSQL, there are two main types of privileges:

* Table-level privileges: Grants the privilege on all columns in the table.
* Column-level privileges: Grants the privilege on a specific column in the table.

To restrict access to only certain columns, the process is similar to table-level privileges. However, the key difference is that we first specify the columns we want to restrict access to before applying the usual privilege grants.

The following code snippet demonstrates how to first revoke the UPDATE privilege on an entire table and then grant it only on specific columns. In this case, the privilege is restricted to the title and content columns of the public.posts table.

-- Revoke update p ri vi le ge on the entire table

REVOKE UPDATE ON TABLE public . posts FROM a u t h e n t i c a t e d ;

-- Grant update p ri vi le ge only on the title and content columns

GRANT UPDATE ( title , content ) ON TABLE public . posts TO a u t h e n t i c a t e d ;

1.2 What is the difference between user identification and user authentication?

When comparing the two, the difference might not seem obvious at first, but there’s an important distinc-tion. Identification is simply telling the system who you are, like your username or role in PostgreSQL. Authentication, on the other hand, is the process of proving who you are by verifying your credentials, like entering a password.

1.3 What are the recommended authentication protocols for PostgreSQL?

Peer authentication-which relies on operating system facilities to identify the process at the other end of a local connection, not supported for remote connections-is usually recommendable for local connections, though trust authentication(the platform trust that the peron is who he says he is) might be sufficient in some circumstances. Password authentication(the user has to send a password) is the easiest choice for remote connections. All the other options require some kind of external security infrastructure (usually an authentication server or a certificate authority for issuing SSL certificates), or are platform-specific.

1

1.4 What is proxy authentication in PostgreSQL and what is it for? Why does it make the previously discussed role-based access control easier to implement?

Proxy authentication in PostgreSQL is a mechanism where a middle tier server facilitates user authentication between a client and the database server. It acts as an intermediary, allowing clients to authenticate indirectly through the proxy instead of directly with the database. This can enhance security and simplify user management.

There are three primary forms of proxy authentication:

* Middle-tier server authenticates both the database and the client: In this case, the middle tier server authenticates itself with the database and verifies the client identity. This allows the client’s identity to be maintained and passed through to the database, ensuring that the database server can apply role-based access control based on the authenticated client’s identity.
* Client is not authenticated by the middle-tier server: The client (which could be a database user) provides their identity and password, which are passed through the middle tier server to the database for authentication. This form ensures that the client’s authentication remains intact, with the middle tier simply relaying the credentials.
* Global user authentication via middle tier: Here, the middle tier authenticates the client (for example, using a certificate or Distinguished Name (DN)) and passes this information to the database. The database then uses this identity for role-based access control.

How proxy authentication aids role-based access control:

Proxy authentication can simplify the implementation of role-based access control (RBAC) in several ways:

* By acting as an intermediary, the middle tier server can maintain a consistent mapping of client identities across different systems and databases. This makes it easier to apply roles and permissions based on authenticated identities, as the middle tier can pass these details directly to the database.
* It also allows for centralized authentication and role management at the middle tier level. Instead of each client managing authentication directly with the database, the middle tier can streamline the process, ensuring that the database can focus on enforcing role-based access policies without being concerned with client authentication.
* The proxy connection provides an abstraction layer that can isolate client authentication mechanisms from the database itself, allowing for easier integration with external authentication systems (e.g., LDAP, certificates) while still leveraging PostgreSQL’s role-based access control.

2