1. **How can one restrict access to certain columns of a database table?**

In PostgreSQL, you can restrict access to certain columns of a table using GRANT and REVOKE statements or by leveraging views.

* Grant Privileges for Specific Columns: PostgreSQL allows column-level privileges. This ensures the role can only access the specified columns. For example:

GRANT SELECT (column1, column2) ON table\_name TO role\_name;

REVOKE SELECT (column3) ON table\_name FROM role\_name;

* Use Views: Create a view that exposes only the desired columns and grant access to the view, not the table. This method hides restricted columns and simplifies user access management.

CREATE VIEW allowed\_columns\_view AS

SELECT column1, column2 FROM table\_name;

GRANT SELECT ON allowed\_columns\_view TO role\_name;

1. **What is the difference between user identification and user authentication?**

**User Identification:** Identification is the process of determining the identity of a user. It involves providing an identifier such as a username, user ID, or email. It answers the question: “Who are you?”

**User Authentication:** Authentication is the process of verifying the claimed identity, usually by validating credentials such as a password, token, biometric data, or other methods. It answers the question: “Are you who you say you are?”

1. **What are the recommended authentication protocols for PostgreSQL?**

PostgreSQL supports several authentication methods, and the choice depends on the use case. Recommended protocols include:

* **SCRAM-SHA-256 (default):** A modern, secure password-based authentication method. It ensures that passwords are hashed and salted, reducing risks of password exposure.
* **Certificate Authentication:** Uses SSL/TLS certificates to verify a user's identity. This is highly secure for environments that require encrypted communication.
* **Kerberos/GSSAPI:** Recommended for large organizations with centralized authentication (e.g., Active Directory).
* **LDAP Authentication:** Suitable for environments already using an LDAP server for authentication.
* **Peer Authentication:** Recommended for local connections where the database trusts the operating system user identity.

Each method can be configured in the pg\_hba.conf file based on network location and the required security level.

1. **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* allows a database client to authenticate as one user (the proxy) but perform actions as another user (the proxied user). This is typically achieved through SET ROLE or SESSION AUTHORIZATION commands. This allows a single, trusted application or service account (proxy) to manage access on behalf of end-users without requiring direct database roles for every user. Proxy authentication is used to centralize user management, especially in environments where application-layer user management is preferred.

**Advantages for Role-Based Access Control (RBAC):**

* Simplified Management: Instead of creating and managing individual database roles for each user, a proxy role can dynamically assume user-specific privileges.
* Security: Minimizes the number of direct user accounts in the database, reducing exposure to security risks.
* Flexibility: Makes it easier to implement dynamic and fine-grained access controls in applications where user permissions vary.

**Example: Simplified Access Control Using Proxy Roles**

Without Proxy Authentication: Every user requires a dedicated database role and permissions. This approach becomes unmanageable in large systems.

CREATE ROLE alice LOGIN;

GRANT SELECT ON table1 TO alice;

With Proxy Authentication: Use a single proxy role for application access. The application ensures the user (e.g., Alice) only has access to permitted resources, delegating actual user management to the application layer.

SET ROLE alice;