ON TOUR | NEW YORK

PostgreSQL vs. SQL Server: Security Model Difference

Taiob Ali

He/Him/His

Database Solutions Manager

GMO LLC

Taiob Ali

He/Him

Database Solutions Manager
GMO LLC

I am a Microsoft Data Platform MVP with over 19 years of experience designing and implementing data solutions across finance, e-commerce, and healthcare. My expertise encompasses the Microsoft Data Platform, MongoDB, Azure AI, and Python, enabling data-driven innovation.

As a dedicated community advocate, I've presented at over 100 events worldwide, including SQL Saturdays, Data Saturdays, and international conferences. I founded the Database Professionals Virtual Meetup Group, serve on the New England SQL Server User Group, and the SQL Saturday boards.

@sqlworldwide

sqlworldwide

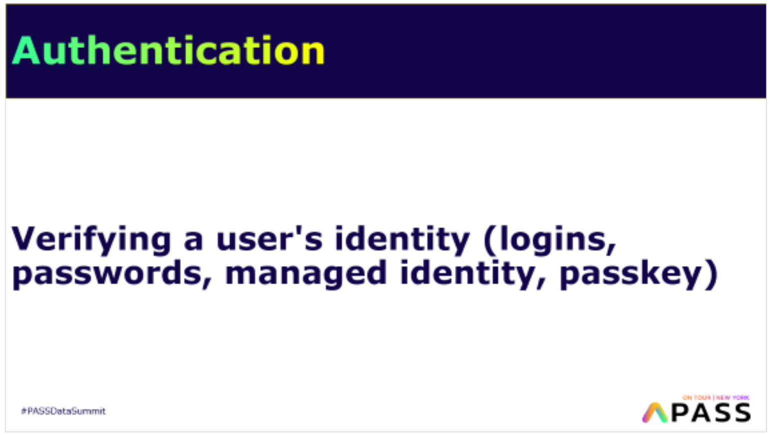
http://www.sqlworldwide.com/

Your feedback is important to us

Evaluate this session at:

passdatacommunitysummit.com/evalutations-nyc

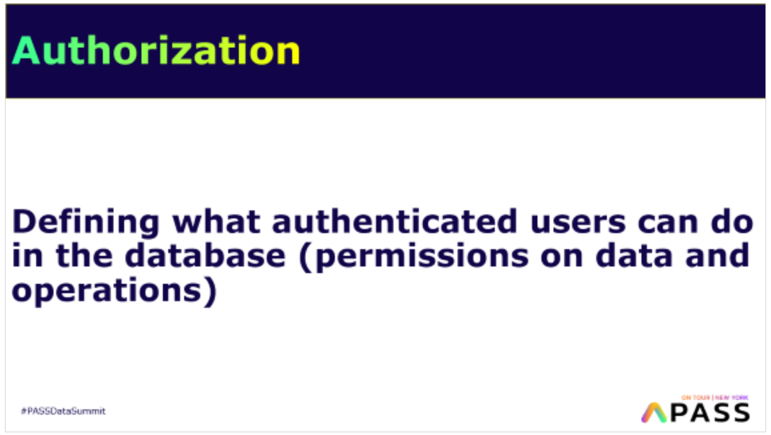
Are we on the same page?



Authentication

Verifying a user's identity (logins, passwords, managed identity, passkey)

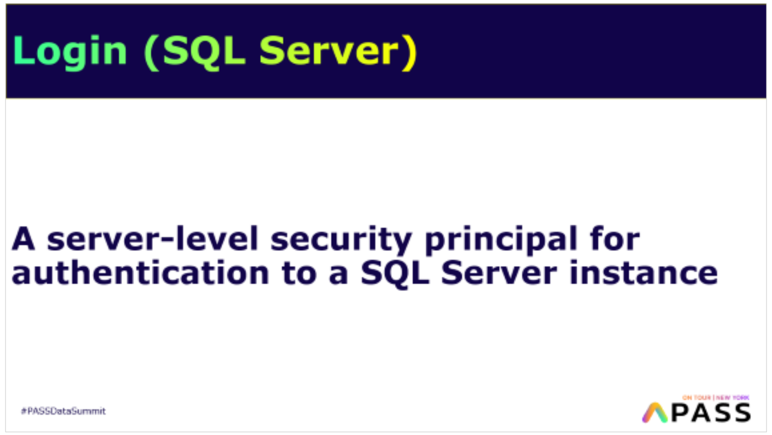
#PASSDataSummit



Authorization

Defining what authenticated users can do in the database (permissions on data and operations)

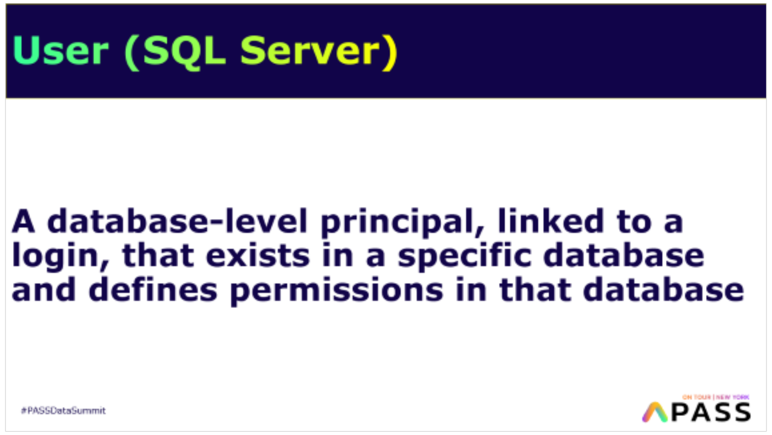
#PASSDataSummit



Login (SQL Server)

A server-level security principal for authentication to a SQL Server instance

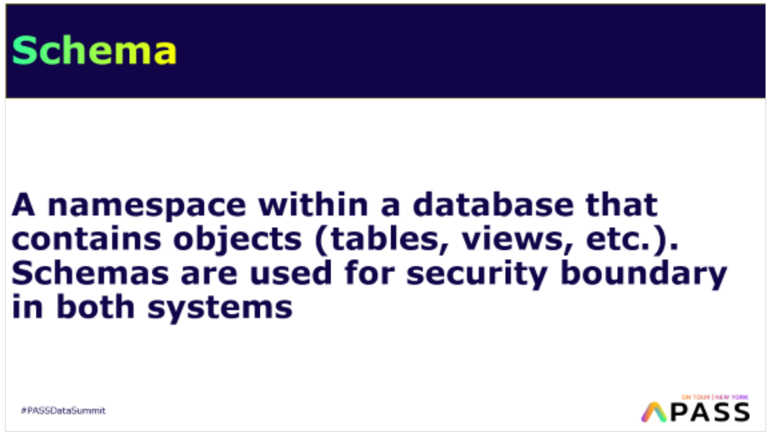
#PASSDataSummit



User (SQL Server)

A database-level principal, linked to a login, that exists in a specific database and defines permissions in that database

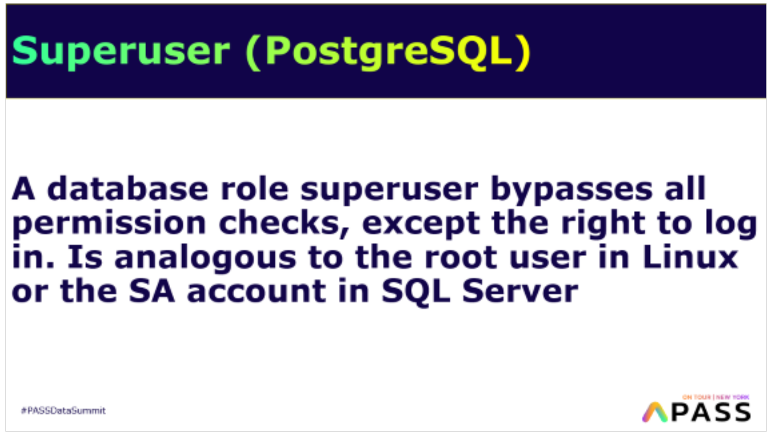
#PASSDataSummit



Schema

A namespace within a database that contains objects (tables, views, etc.). Schemas are used for security boundary in both systems

#PASSDataSummit



Superuser (PostgreSQL)

A database role superuser bypasses all permission checks, except the right to log in. Is analogous to the root user in Linux or the SA account in SQL Server

#PASSDataSummit

Authentication

Verifying a user's identity (logins, passwords, managed identity, passkey)

Authorization

Defining what authenticated users can do in the database (permissions on data and operations)

Login (SQL Server)

A server-level security principal for authentication to a SQL Server instance

User (SQL Server)

A database-level principal, linked to a login, that exists in a specific database and defines permissions in that database

Schema

A namespace within a database that contains objects (tables, views, etc.). Schemas are used for security boundary in both systems

Superuser (PostgreSQL)

A database role superuser bypasses all permission checks, except the right to log in. Is analogous to the root user in Linux or the SA account in SQL Server

SQL Server Roles

- A collection of privileges or permissions
- Fixed Server Roles
- Fixed Database Roles
- User-defined server roles (since SQL 2012)
- User-defined database roles
- Application roles at the Database level

PostgreSQL Cluster

- PostgreSQL, a cluster is not a group of separate servers working together (like in some other databases)
- PostgreSQL cluster = one running PostgreSQL server + the collection of databases it manages, all stored in one data directory

PostgreSQL Roles

- Roles exist at the **cluster** level
- A ROLE that allows login is considered a user
- A role that is not allowed to log in is a group
- Roles can own database objects (For example, tables and functions)

PostgreSQL Roles Permission

- Sum of privileges
  - Granted directly to it
  - Privileges granted to any role it is presently a member of
  - Privileges granted to PUBLIC.

SQL Server: Two-level Principal

- SQL Server employs a two-level principal system for security
- It distinguishes between login and user for access control
- Server and database scopes are separately managed
- Authentication and authorization are clearly separated

PostgreSQL: Unified Role-based System

- PostgreSQL utilizes a unified role-based system
- Roles in PostgreSQL handle both authentication and authorization

Authentication: on Windows

Feature	SQL Server (Windows)	PostgreSQL (Windows)
Windows Authentication	✅ Native via SSPI	⚠️ Possible via SSPI (complex)
SQL Authentication	✅ Supported	✅ Supported
Active Directory Integration	✅ Easy with AD	⚠️ Possible via Kerberos
Ease of Setup for Windows Auth	✅ Very easy	⚠️ Complex

Authentication: on Linux

Feature	SQL Server (Linux)	PostgreSQL (Linux)
Linux System User Authentication	❌ Not supported	✅ Via peer, ident, PAM
SQL Authentication	✅ Supported	✅ Supported
Active Directory Integration	⚠️ Possible via Kerberos	✅ With Kerberos setup
PAM / OS-level Authentication	❌ Not supported	✅ Supported
Ease of Setup for OS-Level Authentication	❌ Not applicable	✅ Easy with peer

Demo-I

PostgreSQL: Role Attributes

- LOGIN / **NOLOGIN**
- SUPERUSER / **NOSUPERUSER**
- CREATDB / **NOCREATEDB**
- CREATEROLE / **NOCREATEROLE**
- **INHERIT** / NOINHERIT
- BYPASSRLS / **NOBYPASSRLS**

Full List

PostgreSQL: Role Scope

- PostgreSQL roles exist globally and are shared across databases in the cluster
- When a role connects to a database, it operates within that database’s context

SQL Server: Public Role

- Every SQL Server login belongs to the public server role
- If a user has no set permissions, they get the public ones
- You can't change membership in public
- Permissions can be granted, denied, or revoked from the public fixed server roles

SQL Database: Public Role

- Every SQL database has a public role
- Default permissions
- Permissions can be granted, denied, or revoked from the public fixed database roles

PostgreSQL: Public Role

- Built-in Role for All Users: Every PostgreSQL role is automatically a member
- Cluster-Wide & Permanent: Exists in every database and cannot be dropped
- Grants Apply to Everyone: Permissions given to the public apply to all current and future users.
- Revoke unwanted default grants from public to tighten access.

PostgreSQL: Public Schema

- Every database has a schema name PUBLIC
- Pre-v15 Public role (all users) had CREATE privilege
- Pos- v15 Removed Create privilege

PostgreSQL: Object Ownership

- Object creator becomes the owner by default
- Ownership grants exclusive privileges
- Ownership can be changed post-creation

PostgreSQL: Default Privileges

- Helps manage access consistently
- Everything to Object Owner (creator)
- On some objects, PUBLIC (everybody) has certain privileges
    - Database: CONNECT
    - Function & Procedures: EXECUTE
    - Languages & Data type: USAGE

PostgreSQL: Alter Default Privileges

- Modify the default privileges for objects that get created in the future
- You can create a custom set of privileges based on the requirement for all objects (similar to the predefined role pg_read_all_data)
- Remove default privilege and alter default privilege before **DROP ROLE**

PostgreSQL: Alter Default Privileges

- **Use** group roles to centralize ownership
- **Avoid** granting CREATE to public roles
- **Maintain** a single set of default privileges

Demo-II

Lesson Learned

Decide how many unique sets of privileges you need based on the project

Carefully create groups that can become the object owner

Use the default privilege to assign permissions to the groups

Further Reading

- PostgreSQL Documentation maintained by Global Development Group
- PostgreSQL Basics: Roles and Privileges by Ryan Booz
- PostgreSQL Basics: Object Ownership and Default Privileges by Ryan Booz
- PostgreSQL Basics: A Template for Managing Database Privileges by Ryan Booz
- PostgreSQL ALTER DEFAULT PRIVILEGES - permissions explained by Laurenz Albe

Your feedback is important to us

Evaluate this session at:

passdatacommunitysummit.com/evalutations-nyc

Thank you

Reach out to me with questions/comments.
You are guaranteed an answer!

Taiob Ali

@sqlworldwide

Taiob at sqlworldwide dot com

https://sqlworldwide.com/