Universidad Nacional Autonoma de México
Facultad de Ingeniería

Conexión y permisos en bases de datos

Maria Fernanda Ordoñez Figueroa

Tarea 2

Fecha: 24 de agosto de 2025

¿Qué requiero para conectarme a la base de datos?

Para conectarse a una base de datos, es necesario contar con varios elementos y cumplir ciertos requisitos técnicos y de seguridad: [1]

- **Usuario y contraseña:** Son las credenciales que identifican y autentican a quien intenta acceder. Cada usuario puede tener diferentes permisos y restricciones.
- **Dirección del servidor:** Es la IP o el nombre del host donde se encuentra el motor de base de datos. En sistemas distribuidos, puede haber varios servidores.
- **Puerto:** Es el número de puerto por el que el servicio de base de datos escucha las conexiones. Por ejemplo, MySQL usa el puerto 3306, PostgreSQL el 5432.
- **Permisos:** El usuario debe tener los privilegios necesarios para realizar las operaciones requeridas (lectura, escritura, administración, etc.).
- **Cliente o driver:** Es el software o librería que permite establecer la conexión. Ejemplos: psql (PostgreSQL), mysql (MySQL), mongo shell (MongoDB), JDBC/ODBC (Java, Excel, etc.).
- **Nombre de la base de datos:** En muchos sistemas, se debe especificar a qué base de datos se desea conectar, ya que puede haber varias en el mismo servidor.

Ejemplo de conexión:

```
psql -h 192.168.1.100 -p 5432 -U usuario -d basedatos
```

En este ejemplo, se conecta a PostgreSQL usando la IP, puerto, usuario y nombre de la base de datos.

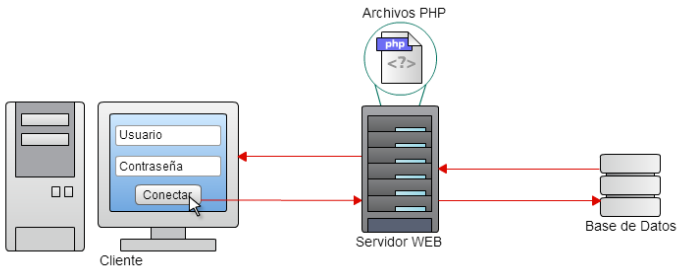

Figura 1: Ejemplo de conexión a una base de datos desde PHP [2]

Permisos a nivel sistema y objeto

Los permisos determinan qué acciones puede realizar un usuario o rol dentro del sistema de gestión de bases de datos (DBMS). Se dividen principalmente en dos niveles:

- **Permisos a nivel sistema:** Permiten ejecutar acciones globales que afectan al sistema completo, como:[3]
  - Crear o eliminar bases de datos.
  - Crear, modificar o eliminar usuarios y roles.
  - Configurar parámetros del servidor.
  - Realizar backups y restauraciones.
  - Supervisar el rendimiento y estado del sistema.

  Estos permisos suelen estar reservados para administradores (DBA).

- **Permisos a nivel objeto:** Se refieren a acciones sobre objetos específicos dentro de una base de datos, como:[4]
  - Tablas: SELECT, INSERT, UPDATE, DELETE, ALTER, DROP.
  - Vistas: SELECT, UPDATE.
  - Funciones y procedimientos: EXECUTE.
  - Esquemas, secuencias, índices, etc.

  Estos permisos permiten controlar el acceso granular a los datos y objetos.

Ejemplo: Un usuario puede tener permiso para consultar (`SELECT`) una tabla, pero no para modificarla (`UPDATE`).

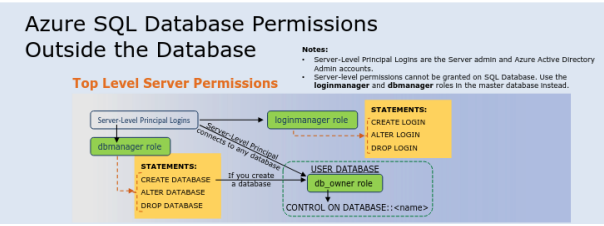

Figura 2: Jerarquía de permisos en un sistema de gestión de bases de datos

¿Cómo dar y quitar permisos?

La gestión de permisos varía según el DBMS, pero en sistemas SQL se utilizan los comandos `GRANT` y `REVOKE`:

- **Dar permisos:**

```
GRANT SELECT, INSERT ON empleados TO usuario;
```

Este comando permite al usuario consultar e insertar datos en la tabla `empleados`.

- **Quitar permisos:**

```
REVOKE INSERT ON empleados FROM usuario;
```

Aquí se revoca el permiso de insertar datos en la tabla.[5]

Permisos avanzados: Se pueden otorgar permisos con la opción `WITH GRANT OPTION`, permitiendo que el usuario a su vez otorgue ese permiso a otros.

```
GRANT SELECT ON empleados TO usuario WITH GRANT OPTION;
```

En sistemas NoSQL, los permisos suelen gestionarse mediante roles y configuraciones específicas. Por ejemplo, en MongoDB:

```
db.createUser({
  user: "usuario",
  pwd: "contraseña",
  roles: [ { role: "readWrite", db: "miBase" } ]
})
```

Diferencia entre role y usuario

- **Usuario:** Es una cuenta individual que representa a una persona, aplicación o servicio que accede a la base de datos. Cada usuario tiene su propio nombre y contraseña.
- **Role (Rol):** Es un conjunto de permisos agrupados bajo un nombre. Los roles facilitan la administración, ya que se pueden asignar a varios usuarios y modificar los permisos de todos ellos de manera centralizada.

Ventajas de los roles:

- Simplifican la gestión de permisos en sistemas con muchos usuarios.
- Permiten aplicar cambios de privilegios de forma masiva y consistente.
- Mejoran la seguridad y el control de acceso.

Ejemplo en PostgreSQL:

```
CREATE ROLE analista;
GRANT SELECT ON todas_las_tablas TO analista;
GRANT analista TO usuario1;
GRANT analista TO usuario2;
```

Aquí, el rol `analista` tiene permiso de consulta sobre todas las tablas, y se asigna a dos usuarios diferentes. [6]

Referencias

[1] Microsoft. "SQL Server 2017 and Azure SQL Database Permissions Infographic." Accessed: 2024-06-10. (2017), dirección: https://raw.githubusercontent.com/Microsoft/sql-server-samples/master/samples/features/security/permissions-posters/Microsoft_SQL_Server_2017_and_Azure_SQL_Database_permissions_infographic.pdf.

[2] Microsoft. "Getting Started with Database Engine Permissions." Accessed: 2024-06-10. (2024), dirección: https://learn.microsoft.com/es-es/sql/relational-databases/security/authentication-access/getting-started-with-database-engine-permissions?view=sql-server-ver17.

[3] Sucerman. "Gestión de permisos y seguridad." Accessed: 2024-06-10. (2024), dirección: https://contenidos.sucerman.com/nivel2/web1/unidad4/leccion2.html.

[4] FasterCapital. "DCL para administradores de bases de datos: gestión de permisos y seguridad." Accessed: 2024-06-10. (2024), dirección: https://fastercapital.com/es/contenido/DCL-para-administradores-de-bases-de-datos--gestion-de-permisos-y-seguridad.html.

[5] IBM. "System Object Permissions." Accessed: 2024-06-10. (2024), dirección: https://www.ibm.com/docs/es/netcoolomnibus/8.1.0?topic=roles-system-object-permissions.

[6] IBM. "SQL Server Database Privileges." Accessed: 2024-06-10. (2024), dirección: https://www.ibm.com/docs/es/bpm/8.5.6?topic=privileges-sql-server-database.