Control de acceso

Administración de Bases de Datos

Departamento de Lenguajes y Sistemas Informáticos

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Contenido

1. Introducción
2. Gestión de privilegios en SQL
3. Roles
4. Vistas y procedimientos almacenados
5. Otras técnicas

Introducción

- Ley Orgánica de Protección de Datos de Carácter Personal (LOPD)
  - Garantiza y protege las libertades públicas y los derechos fundamentales de las personas físicas.
    - En lo referente al tratamiento de los datos personales.
  - Información sensible protegida por ley: raza, religión, datos médicos, condenas penales, morosidad, ...
  - La Agencia Española de Protección de Datos (AEPD) es el organismo público que vela por el cumplimiento de la LOPD.

Introducción

- Ley Orgánica de Protección de Datos de Carácter Personal (LOPD)
  - Artículo 9, primer punto:

"El responsable del fichero, y, en su caso, el encargado del tratamiento, deberán adoptar las medidas de índole técnica y organizativas necesarias que garanticen la seguridad de los datos de carácter personal y eviten su alteración, pérdida, tratamiento o acceso no autorizado, habida cuenta del estado de la tecnología, la naturaleza de los datos almacenados y los riesgos a que están expuestos, ya provengan de la acción humana o del medio físico o natural."

Introducción

- Ley Orgánica de Protección de Datos de Carácter Personal (LOPD)
  - *Ejemplos de incumplimientos y sanciones asociadas:*


EL PAÍS

Comunidad Valenciana

GENERALITAT VALENCIANA · VALENCIA · CAST

PROTECCIÓN DE DATOS >

Mercadona paga una sanción de 2,5 millones de euros a Protección de Datos

La cadena testó un proyecto piloto que permitía detectar personas con orden de alejamiento de sus tiendas


MENÚ

ANÁLISIS XATAKA SELECCIÓN MÓVILES CIENCIA GUÍAS DE COMPRA

PARTNERS Innovación LG Tecnología Midea Tecnología Vodafone Planeta Fantasy Espacio Intel

El reconocimiento facial de Mercadona acaba en multa de 2,5 millones de euros: qué dice la Agencia de Protección de Datos y qué lecciones se pueden extraer

Enlaces:

- Noticia en El País: https://elpais.com/economia/2021-07-22/mercadona-paga-una-sancion-de-25-millones-de-euros-a-proteccion-de-datos.html
- Noticia en Xataka: https://www.xataka.com/privacidad/reconocimiento-facial-mercadona-acaba-multa-2-5-millones-euros-que-dice-agencia-proteccion-datos-que-lecciones-se-pueden-extraer
- Resolución de la AEPD: https://www.aepd.es/es/documento/ps-00120-2021.pdf

Introducción

- Ley Orgánica de Protección de Datos de Carácter Personal (LOPD)
  - *Ejemplos de incumplimientos y sanciones asociadas:*

epSevilla | europa press

europapress / andalucía / sevilla

Protección de Datos expedienta a la Agencia Tributaria y abre expediente a Tecnocom por el agujero informático

Europa Press Andalucía

Publicado: jueves, 28 abril 2016 14:45
@epandalucia

Newsletter

SEVILLA, 28 Abr. (EUROPA PRESS) -

Facua Sevilla ha informado de que la Agencia Española de Protección de Datos (AEPD) ha iniciado un procedimiento de declaración de infracción de administraciones públicas a la

Diario de Sevilla SEVILLA

SEVILLA PROVINCIA ANDALUCÍA ESPAÑA ECONOMÍA SOCIEDAD DEPORTES CULTURA COFRADÍAS OPINIÓN TODAS LAS SECCIONES

SEVILLA VIVIR JUZGADO DE GUARDIA RUTAS DE SENDERISMO

GOYA ¿Dónde están las nueve esculturas de Goya distribuidas por Sevilla?

SEVILLA

La multa por los datos publicados puede llegar a los 300.000 euros

- La sanción recaería sobre la empresa encargada de gestionar la web de la Hacienda local y no sobre el ente público

DIEGO J. GENIZ

Enlaces:

- Noticia en EuropaPress: https://www.europapress.es/andalucia/sevilla-00357/noticia-proteccion-datos-expedienta-agencia-tributaria-abre-expediente-tecnocom-agujero-informatico-20160428144556.html
- Noticia en Diario de Sevilla: https://www.diariodesevilla.es/sevilla/multa-publicados-puede-llegar-euros_0_950605434.html

Introducción

- El administrador de base de datos debe velar por la seguridad e integridad de los datos.

- *Seguridad*: protección de datos contra revelación, alteración o destrucción no autorizados.
  - Vigilar que los usuarios de la BBDD realizan operaciones para las que están autorizados.

- *Integridad*: exactitud o validez de los datos.
  - Vigilar que los usuarios usan datos consistentes.

Introducción

- Generalmente, el administrador de base de datos es el responsable principal de la seguridad del SGBD.

- Debe tener una cuenta de super-usuario.
  - La forma de acceso debe estar bien protegida.
    - P.e. contraseña, certificado, ...

- Debe especificar:
  - Qué usuarios tienen acceso y a qué datos.
  - Qué operaciones pueden realizar sobre dichos datos.

Introducción

- En la mayoría de SGBDs, el acceso se controla mediante privilegios.
  - Permisos para realizar determinadas operaciones sobre ciertos datos.
- Diferentes usuarios pueden tener privilegios distintos sobre un mismo objeto.
- Clases de privilegios.
  - A nivel de cuenta.
  - A nivel de objeto (BD, tabla, columna).

Gestión de privilegios en SQL

- Crear una cuenta de usuario
  - En su forma más simple:

```
CREATE USER <nombre-usuario> IDENTIFIED BY 'contraseña';
```

  - Tras su ejecución, este comando no aparece en el histórico de comandos por motivos de seguridad.
  - Manual del comando:
    - https://dev.mysql.com/doc/refman/8.0/en/create-user.html

- Borrar una cuenta de usuario:

```
DROP USER <nombre-usuario>;
```

Gestión de privilegios en SQL

- Conceder privilegios:

```
GRANT <privilegio> ON <objeto>
TO <sujeto> [WITH GRANT OPTION]
```

  - donde:
    - ‹privilegio›: Permiso o privilegio a conceder
    - ‹objeto›: Objeto involucrado, p.e. BD, tabla, ...
    - ‹sujeto›: Usuario/rol receptor del privilegio
    - [With Grant Option]: Permite la propagación del permiso (Opcional)

  - Ejemplo:
    - Permitir que el usuario “Unai” realice operaciones Select sobre la tabla “miTabla” de la BBDD “miBD”.

```
GRANT select ON miBD.miTabla TO Unai;
```

Gestión de privilegios en SQL

- Retirar privilegios:

```
REVOKE [GRANT OPTION FOR] <privilegio>
ON <objeto> FROM <sujeto>
```

  - donde:
    - [Grant Option For]: Quita la capacidad de propagar el permiso (Opcional)
    - ‹privilegio›: Permiso o privilegio a retirar
    - ‹objeto›: Objeto involucrado, p.e. BD, tabla, ...
    - ‹sujeto›: Usuario/rol poseedor del privilegio

  - Ejemplo:
    - Retirar permiso para hacer operaciones Select sobre la tabla “miTabla” de la BBDD “miBD” al usuario “Unai”.

```
REVOKE select ON miBD.miTabla FROM Unai;
```

Gestión de privilegios en SQL

- Privilegios más habituales:

Privilegio	Descripción	Contexto
CREATE	Crear nuevas BDs o tablas	BD, tabla o índice
RELOAD	Re-cargar tablas del sistemas	Global
INSERT	Inserción de filas en tablas	Tabla o columna
SELECT	Lectura de filas de tablas	Tabla o columna
DROP	Eliminar BDs, tablas o vistas	BD, tabla o vista

  - Listado completo para MySQL:
    - https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#privileges-provided-summary
    - Otros SGBDs pueden organizar los privilegios de forma diferente.

Gestión de privilegios en SQL

- Tipos de objetos al usar Grant/Revoke.

Objeto	Contexto	Ejemplos de permisos posibles
.	Global (nivel de cuenta)	SHOW DATABASES, CREATE (con *.*, crear nuevas BDs y tablas)
‹BD›.*	BD	CREATE (con ‹BD›.*, crear tablas sólo en ‹BD›) SELECT (con ‹BD›.*, leer cualquier tabla de ‹BD›)
‹BD›.‹tabla›	Tabla	SELECT (Leer ‹tabla› en ‹BD›) DELETE
‹BD›.‹tabla›(columna)	Columna	INSERT, SELECT, UPDATE

- Ejemplo:
  - Dar permiso para crear nuevas tablas en la BBDD “tienda” al usuario “Unai”

```
GRANT create ON tienda.* TO Unai;
```

Gestión de privilegios en SQL

- Permisos especiales

Permiso	Descripción
ALL	Todos los permisos disponibles en el nivel actual
REFERENCES	Permite referenciar una tabla desde otra.
RELOAD	Permite el uso del comando "Flush privileges"*
SUPER	Permisos de super-usuario *(uso desaconsejado)*

* El comando *Flush privileges* re-carga la tabla de privilegios. Útil para activar permisos recién otorgados.

- Mostrar permisos otorgados:
  - Para nuestro usuario actual:
  - Para otro usuario:

```
SHOW GRANTS;
```

```
SHOW GRANTS FOR <usuario>;
```

Gestión de privilegios en SQL

- Propagación de privilegios.
  - La cláusula "With grant option" permite, al usuario receptor de un nuevo permiso, propagarlo a otros usuarios.
  - Ejemplo:
    - Permitir que el usuario Unai actualice filas en la tabla "empleados" de la BD "tienda" y propague ese permiso a otros usuarios:

```
GRANT update ON tienda.empleados TO Unai WITH GRANT OPTION;
```

  - Se puede revocar la capacidad de propagación para un permiso.
  - Ejemplo:
    - Retirar la capacidad de propagación del permiso anterior.

```
REVOKE GRANT OPTION FOR update ON tienda.empleados FROM Unai;
```

Ejercicio 1

- Como super-usuario:
  - Crear un BD llamada “ej1DB”
  - Crear un usuario “ej1user” y otorgar permisos para:
    - Crear tablas en ej1DB.
    - Leer, insertar y borrar filas en las tablas de ej1DB.
  - *Se deben otorgar permisos de la forma más precisa posible.*

- Iniciar sesión como “ej1user” y, en “ej1DB”:
  - Verificar los permisos asignados.
  - Crear 1 tabla “ejTabla” con los siguientes campos y datos:
  - Borrar las filas id 2 y 3.

id (Entero)	nombre (Texto)
1	Hola
2	Mundo
3	Adiós

Roles

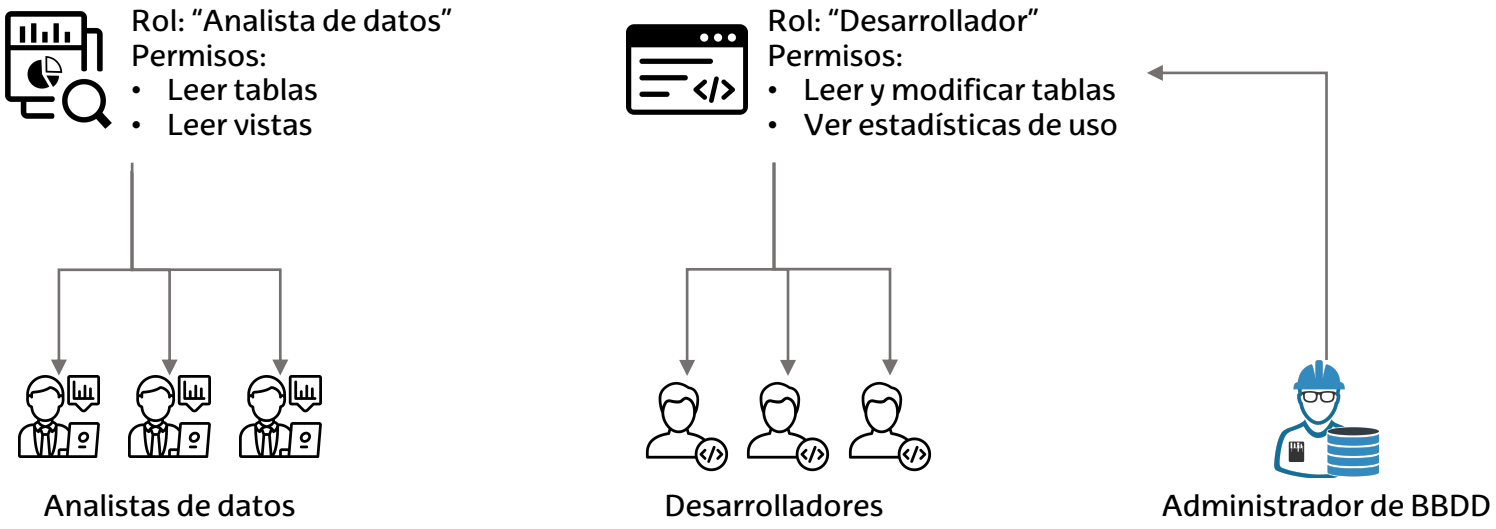
- En un SGBD con muchos usuarios, varios de ellos pueden necesitar el mismo conjunto de permisos.
  - P.e. todos los desarrolladores de un mismo proyecto.

- En esas situaciones, los permisos se pueden gestionar de forma unificada utilizando roles.

Roles

- Un rol es una colección de privilegios con un nombre.

- Ejemplo para la BD de un proyecto:



Iconos de: https://www.flaticon.com/

Roles

- Crear un rol:
  - Requiere ser super-usuario o tener el permiso “Create role”.

```
CREATE ROLE <nombre-rol>;
```

- Asignar privilegios a un rol:
  - Requiere ser super-usuario o tener asignado el privilegio que se quiere asignar.

```
GRANT <privilegio> ON <objeto> TO <nombre-rol>;
```

- Asignar rol a un usuario:
  - Requiere ser super-usuario o tener asignado el rol.

```
GRANT <nombre-rol> TO <usuario>;
```

Roles

- Tras asignar un rol a un usuario, este debe activarlo para poder usar sus privilegios.
  - Motivo: poder distinguir entre diferentes roles.

- Como usuario de un rol:
  - Activar un rol concreto:
  - Mostrar el rol en uso:

```
SET ROLE <nombre-rol>;
```

```
SELECT current_role();
```

- Alternativamente, se puede definir un rol por defecto que se activa cuando el usuario inicia sesión.
  - Como super-usuario:

```
SET DEFAULT ROLE <nombre-rol> TO <usuario>;
```

Roles

- Mostrar privilegios de un usuario cuando activa un rol:

```
SHOW GRANTS FOR <usuario> USING <nombre-rol>;
```

- Retirar privilegios de un rol:

```
REVOKE <privilegio> ON <objeto> FROM <nombre-rol>;
```

- Eliminar un rol:

```
DROP <nombre-rol>;
```

- Más información sobre la gestión de roles:
  - https://dev.mysql.com/doc/refman/8.0/en/roles.html

Ejercicio 2

- Como super-usuario:
  - Crear una BD “ej2DB” y un usuario “ej2user”.
  - Crear un rol “ej2rol” con los siguientes permisos:
    - Crear tablas en “ej2DB”.
    - Insertar y leer filas de las tablas en “ej2DB”.
  - Asignar el rol ”ej2rol” a “ej2user”.

- Iniciar sesión como “ej2user” y, en “ej2DB”:
  - Crear una tabla “ejTabla” con los siguientes campos y datos:
  - Intentar borrar la fila con “id” = 3. ¿Qué sucede?

id (Entero)	word (texto)
1	Hello
2	World
3	Goodbye

Control de acceso

- En ocasiones, el sistema de privilegios del SGBD no permite controlar las operaciones de los usuarios a un nivel detallado.

- P.e., para prevenir que los usuarios lean o modifiquen filas concretas de un tabla.

Vistas

- Son consultas almacenadas que producen un resultado al ser invocadas.
  - Permiten limitar la visibilidad de los datos.

- Ejemplo:
  - Un desarrollador D necesita consultar la tabla T, pero sólo debe ver las filas en las que la columna C es mayor que 40.

  - *Solución*:
  - El super-usuario crea una vista V que recupera las filas de T cuyo valor C es mayor que 40.
  - El super-usuario asigna a D privilegios para realizar Select sobre V.

Vistas

- Creación de una vista:

```
CREATE VIEW <BD>.<nombre-vista> AS <sentencia-select>;
```

- Borrar una vista:

```
DROP VIEW <nombre-vista>;
```

- Más información:
  - https://dev.mysql.com/doc/refman/8.0/en/views.html

Rutinas almacenadas

- Porciones de código SQL reutilizables con un nombre.
  - Más flexibles que las vistas, pero más complejas de mantener.
- Hay 2 tipos de rutinas almacenadas:
  - Procedimientos almacenados
    - Se invocan utilizando la función CALL.
  - Funciones almacenadas
    - Se invocan usando el nombre de la función.
- No se deben utilizar para implementar lógica de negocio.
  - Hacerlo complica el mantenimiento y depuración de la aplicación.

Procedimientos almacenados

- Crear procedimiento almacenado en MySQL:
  - *Paso previo*: cambiar el delimitador de fin de sentencia.
    - Si no hacemos este cambio, MySQL interpretaría el primer ";" en el código del procedimiento como el fin del mismo.

```
DELIMITER //
```

  - Crear procedimiento:

```
CREATE PROCEDURE <nombre> (<parámetros>)
BEGIN
  <sentencias>
END //
```

  - Paso posterior: Re-establecer el delimitador.

```
DELIMITER ;
```

Procedimientos almacenados

- Crear procedimiento almacenado en MySQL:
  - *Parámetros de entrada:*
  - Se debe indicar si son de entrada, salida o entrada/salida.
    - Se indica con IN, OUT e INOUT, p.e.:

```
CREATE PROCEDURE miProc (IN nombre CHAR(3), OUT valor INT)
  BEGIN …
```

  - En el cuerpo del procedimiento, se les asignan valores:
    - Mediante una consulta Select, p.e.:

```
SELECT columna INTO valor FROM … WHERE …
```

    - Usando una sentencia Set, p.e.:

```
SET valor = 3;
```

Procedimientos almacenados

- Crear procedimiento almacenado en MySQL:
  - *Cuerpo del procedimiento:*
  - Declaración de variables locales:

```
DECLARE variable <tipo>;
```

  - Se pueden usar estructuras de control.
    - P.e., condicionales:

```
IF <condición> THEN
  <sentencias>
ELSE …
```

    - P.e. bucles:

```
WHILE <condición> DO
  <sentencias>
END WHILE
```

Procedimientos almacenados

- Todo procedimiento almacenado está asociado a una BD.
  - Si la BD se borra, también todos sus procedimientos.
  - Mostrar procedimientos existentes en una BD:

```
SHOW PROCEDURE STATUS WHERE DB = '<nombre-BD>';
```

- Llamar a un procedimiento almacenado:

```
CALL miProc(…);
```

- Son necesarios los privilegios *Create Routine* y *Execute* para crear y ejecutar procedimientos, respectivamente.

- Más información:
  - https://dev.mysql.com/doc/refman/8.0/en/create-procedure.html

Ejercicio 3

- Crear una BD “BDvalores” con una tabla “valores” que contenga los siguientes datos:

id (Entero)	valor (Entero)
1	3
2	2
3	4

- Crear y ejecutar un procedimiento almacenado que:
  - Sume todos los valores de la columna “valor”.
  - Si la suma es menor que 10, muestre un mensaje “Total < 10”.
  - Si la suma es mayor o igual que 10, muestre “Total >= 10”.
    - Para mostrar un mensaje, usar: SELECT “Mi mensaje”;
- En la fila de la tabla con id = 2, modificar valor = 6.
  - Verificar que el procedimiento muestra un mensaje correcto.

Otras técnicas

- Ataques de denegación de servicio.
  - Creación de excesivas conexiones al SGBD, afectando a su rendimiento o directamente bloqueando su ejecución.
  - *Prevención*: limitar los recursos que usa cada cuenta de usuario.
    - P.e. nº de conexiones simultáneas, nº de conexiones por hora, ...
  - Ejemplo:
    - Permitir que el usuario Unai realice 500 consultas/hora como máximo.

```
ALTER USER Unai WITH MAX_QUERIES_PER_HOUR 500;
```

  - Más información:
    - https://dev.mysql.com/doc/refman/8.0/en/user-resources.html

Otras técnicas

- Ataques *SQL Injection*
  - Inyección de código malicioso en una consulta a la BD.
    - Explicación detallada: https://www.hacksplaining.com/exercises/sql-injection
  - *Prevención*: Usar parametrización en las consultas del código de la aplicación.
    - Ejemplo de consulta vulnerable en Java[1]:

```
String query = "Select * from USERS where name = '" + request.getParam("username")
               + "' and password = '" + request.getParam("password") + "'" ;
```

No hay validación del contenido

    - Versión mejorada usando parametrización[1]:

```
String custname = request.getParameter("customerName");
String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname);
ResultSet results = pstmt.executeQuery( );
```

[1]Fuente: https://github.blog/2022-01-27-beyond-sql-injection-owasp-tips-secure-database-access/