

Seguridad en las Bases de Datos

Bases de Datos 2022

Motivación

- Hasta el momento, solo hemos *jugado* con bases de datos
- En una situación real, se maneja información sensible

Ejemplos:

- Contraseñas
- Números de tarjetas de crédito o débito
- Pedidos realizados por un cliente
- Historia clínica de una persona

Hay que proteger esos datos del acceso y/o de la modificación malintencionados.

Seguridad

Protección contra:

- Revelación no autorizada (Confidencialidad)
- Alteración no autorizada (Integridad)
- Destrucción intencional o involuntaria

Protección dirigida a dos tipos de usuarios:

- Los que no tienen derecho de acceso
- Los que tienen derecho limitado a ciertas acciones

Sistema de Seguridad del DBMS

Objetivos:

- **Integridad**
Sólo los usuarios autorizados deberían tener acceso para modificar datos.
- **Disponibilidad**
Los datos deben estar disponibles para usuarios y programas de actualización autorizados.
- **Confidencialidad**
Protección de los datos de su revelación no autorizada.

Elementos que pueden ser protegidos

Granularidad:

- Un atributo de una tupla
- Un conjunto de columnas
- Una tupla individual
- Un conjunto de tuplas de una relación
- Una relación en particular
- Un conjunto de relaciones
- La base de datos completa

Métodos para el Control de Accesos

Control de Acceso Discrecional:

Garantiza privilegios a usuarios, incluyendo la capacidad para acceder archivos de datos específicos, registros o campos para operar de una manera determinada (select, insert, delete, o update).

Seguridad a cargo del DBMS

Encriptado de Datos

Los datos son ilegibles a menos que se tenga conocimiento del código.

Seguimiento del 'rastro' (Audit Trail)

Si alguien entra a la base de datos, entonces la idea es poder saber a qué datos accedió y que hizo con ellos.

Ataque por Inyección en SQL

(SQL Injection Attack)

- Ocurre cuando cuando algunos datos del usuario están destinados a modificar una sentencia SQL
- El input de un usuario que puede modificar una sentencia SQL debe ser cuidadosamente editado para asegurar que sólo se reciben inputs válidos y que no se ha ingresado código adicional.

Ataque por Inyección en SQL (SQL Injection Attack)

- Ejemplo: en un campo de texto de un formulario en la WEB se requiere a los usuarios que ingresen su nombre.
- El usuario ingresa: ' Juan Durán ' OR TRUE

```
select * from instructor  
where name = 'Juan Durán' OR TRUE;
```

- **Resultado:** puede llegar a obtener toda la tabla instructor

Ataque por Inyección en SQL (SQL Injection Attack)

Otro Ejemplo:

```
select * from instructor  
where name = 'X';
```

El usuario ingresa en lugar de X:

```
a'; drop table instructor;--
```

```
select * from instructor  
where name = 'a'; drop table instructor;--';
```

La tabla instructor es eliminada

Ataque por Inyección en SQL (SQL Injection Attack)



Seguridad a Nivel de Usuario en SQL

- Cada usuario tiene ciertos derechos sobre ciertos objetos.
- Distintos usuarios pueden tener los mismos o distintos derechos sobre los mismos objetos.
- Para controlar la granularidad de los derechos de acceso, los usuarios pueden tener derechos (autorización / privilegios) sobre

Tabla
Vista

Control de Acceso Discrecional

- En SQL standard se incluyen los privilegios **SELECT**, **INSERT**, **UPDATE**, y **DELETE**.
- En el lenguaje de definición de SQL se incluyen los comandos **GRANT** y **REVOKE** para administrar privilegios.

El comando **GRANT** es usado para conceder autorización. La forma básica de esta sentencia es la siguiente:

```
grant          <privilege list>  
on  <relation name or view name>  
to <user/role list>;
```

GRANT - Ejemplo

```
grant select  
on department  
to Mafalda, Guille;
```

```
grant update (budget)  
on department  
to Mafalda, Guille;
```

Revoke

Para anular una autorización, se use la sentencia **REVOKE**. La estructura es muy similar a **GRANT**:

```
revoke <privilege list>  
on <relation name or view name>  
from <user/role list>;
```

Revoke - Ejemplo

```
revoke select  
on department  
from Mafalda, Guille;
```

```
revoke update (budget)  
on department  
from Mafalda, Guille;
```


Roles

Problema:

Si hay muchos usuarios con muchos privilegios diferentes, resultará difícil adicionar nuevos privilegios a cada individuo.

Surge la idea de Roles:

- Grupos de privilegios relacionados que se otorgan a usuarios.
- Si se cambian los privilegios encapsulados en un rol, los privilegios de todos los usuarios que tienen ese rol también cambian.

Roles en SQL

Ejemplo de creación:

```
create role instructor;
```

Ejemplo de asignación de privilegios a roles:

```
grant select on takes  
to instructor;
```

Ejemplo de asignación de privilegios de roles a roles y a usuarios:

```
create role dean;  
grant instructor to dean;  
grant dean to Mafalda;
```

Algunos consejos de seguridad

- El usuario **root** debe estar protegido con una contraseña
- Únicamente el usuario **root** debe tener acceso a la tabla `mysql.user`
- No dar más privilegios que los necesarios
- No dar permisos a todos los hosts (comodín `'%'`)
- No almacenar contraseñas (como datos) en texto plano