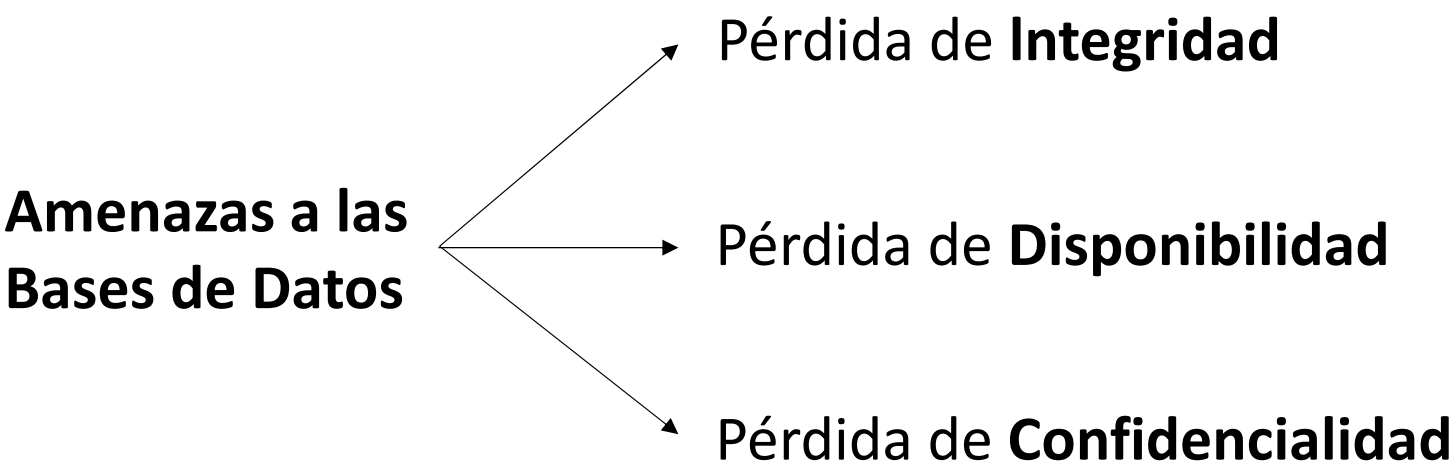
Universidad Nacional de La Matanza

Catedra de Base de Datos

Clase Teórica de Seguridad

Lunes 29/06/2020

Seguridad en Bases de Datos



Seguridad en Bases de Datos

Pérdida de **INTEGRIDAD**

Por ejemplo: un usuario elimina involuntariamente una tabla.

¿Como se puede evitar que ocurra algo así?

- Restringir permisos y que solo puedan eliminar los usuarios administradores
- Tener varias instancias/entornos de bases de datos (desarrollo, test, producción, etc.)

Todo esto disminuye las probabilidades de que suceda pero igualmente puede ocurrir.

Seguridad en Bases de Datos

Pérdida de **INTEGRIDAD**

Por ejemplo: un usuario elimina involuntariamente una tabla.

Para recuperar la información perdida (o modificada) tenemos que restaurar un Backup.

Se puede tomar Backup con una sentencia SQL muy simple, así que no hay excusa para no hacerlo periódicamente.

Seguridad en Bases de Datos

BACKUP y RESTORE

Comandos para SQL Server:

BACKUP DATABASE <Nombre de la Base de Datos>
TO DISK = '<Path completo del archivo>'

Ejemplo:
BACKUP DATABASE Universidad
TO DISK = 'C:\Universidad_backup.bak'

Seguridad en Bases de Datos

BACKUP y RESTORE

Comandos para SQL Server:

RESTORE DATABASE <Nombre de la Base de Datos>
FROM DISK = '<Path completo del archivo>'

Ejemplo:
BACKUP DATABASE Universidad
FROM DISK = 'C:\Universidad_backup.bak'

Seguridad en Bases de Datos

BACKUP y RESTORE

- Este comando se puede ejecutar “en caliente”, es decir, no es necesario que todos los usuarios salgan de la base para poder ejecutarlo.
- Mostramos solo la versión más simple del comando, pero en realidad tiene muchos otros parámetros (que no vamos a ver), los cuales permiten hacer copias incrementales, comprimidas, con clave, etc.
- No sirve copiar los dos archivos de la base de datos (datafiles) directo desde el sistema operativo:
  Universidad.mdf y Universidad.ldf
  Si hacemos esto y después los pisamos, toda la base de datos dará error y quedará en estado inconsistente porque parte de la información está en la base Master y ambas quedarán desincronizadas.

Seguridad en Bases de Datos

Pérdida de **DISPONIBILIDAD**

Por ejemplo1: la base de datos deja de funcionar por una falla en el disco rígido del servidor.

Para evitar que esto suceda se deben tomar medidas de seguridad a nivel del hardware, específicamente focalizado en el disco rígido.
A continuación veremos algunas alternativas.

Seguridad en Bases de Datos

Pérdida de **DISPONIBILIDAD**

Seguridad de los datos en disco:

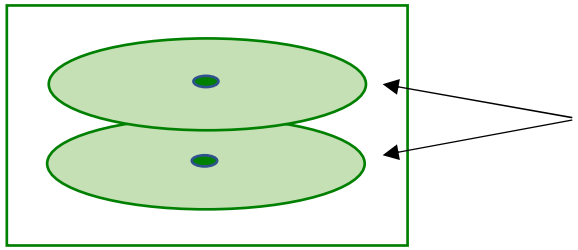
a) Utilizar un sistema RAID
(Redundant Array of Inexpensive Disks)
Consiste en utilizar varios discos rígidos de forma tal de que si falla uno de ellos no se pierda información (y pueda ser reemplazado sin problemas)
Existen diversas categorías de RAID, las más usadas son:
RAID 1 y RAID 5

Seguridad en Bases de Datos

Pérdida de **DISPONIBILIDAD**

Seguridad de los datos en disco:

a) Utilizar un sistema RAID
   RAID 1: Disco espejado 100%. Dos discos se usan como uno solo.



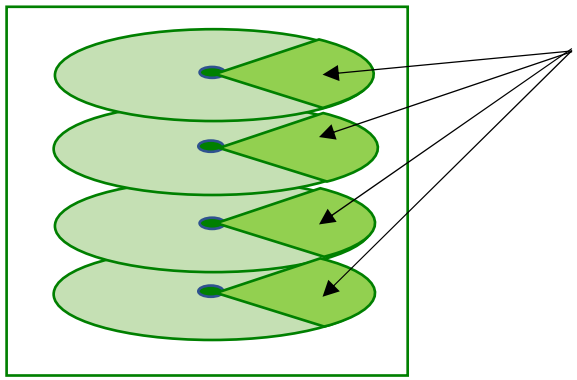
Ambos discos tienen exactamente la misma información.
Si falla uno de ellos, no hay problema porque se sigue usando el otro y tenemos tiempo para luego reemplazar el disco dañado.

Seguridad en Bases de Datos

Pérdida de **DISPONIBILIDAD**

Seguridad de los datos en disco:

a) Utilizar un sistema RAID
RAID 5: Los datos se distribuyen en varios discos (por ejemplo 4) con redundancia suficiente como para tolerar que un disco falle.



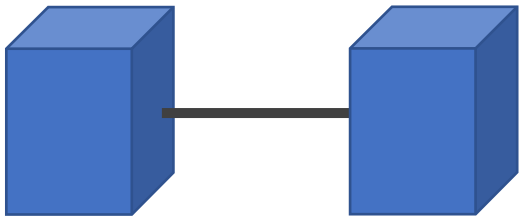
Cada disco tiene una porción de información redundante con los otros.
Si un disco falla, con las porciones de los otros se cubre su ausencia.
Obviamente, luego hay que reemplazar el disco dañado tan pronto como sea posible y al reemplazarlo se debe reconstruir todo el RAID.

Seguridad en Bases de Datos

Pérdida de **DISPONIBILIDAD**

Seguridad de los datos en disco:

b) Utilizar un sistema almacenamiento independiente (Storage)
Son servidores especialmente diseñados para almacenar información. Por ejemplo, tienen 80 discos rígidos conectados con fibra óptica y múltiples buffers de memoria. Con un sistema operativo creado especialmente para ellos (un Linux compilado para controlar ese hardware especifico).

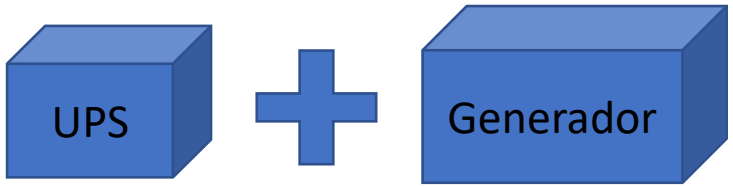


Seguridad en Bases de Datos

Pérdida de **DISPONIBILIDAD**

Por ejemplo2: la base de datos deja de funcionar por un corte del suministro eléctrico.

Para evitar que esto suceda se deben tomar medidas de seguridad con respecto al suministro eléctrico.



Funciona como una gran batería
Se activa en forma instantánea.

Tarda en encender (no es instantáneo)

Seguridad en Bases de Datos

Pérdida de **CONFIDENCIALIDAD**

Por ejemplo: alguien no autorizado logra acceder a información confidencial.

Existen 3 tipos de mecanismo de seguridad:

- Discrecional (DAC: Discretionary Access Control)
- Basado en Roles (RBAC: Role Based Access Control)
- Mandatorio (MAC: Mandatory Access Control)

Seguridad en Bases de Datos

DAC

Consiste en otorgar y revocar privilegios a los distintos usuarios.
Para ello se utilizan los comandos GRANT y REVOKE.

GRANT

Permite otorgar permisos a un usuario o un rol.

GRANT <permiso> ON <tipo de objeto> <nombre del objeto>
TO <usuario/rol> [WITH GRANT OPTION]

Permisos:
SELECT / INSERT / UPDATE / DELETE / ALTER / DROP / EXECUTE / ALL

Tipos de Objetos:
[TABLE] / COLUMN / FUNCTION / PROCEDURE / DATABASE ALIAS

El tipo de objeto TABLE es opcional y se usa en el caso de las Tablas o Vistas.

Seguridad en Bases de Datos

GRANT

Ejemplos:

GRANT ALL ON Tabla2 TO Usuario2 WITH GRANT OPTION

GRANT UPDATE ON COLUMN Tabla1.Columna5 TO Usuario3

GRANT SELECT ON DATABASE ALIAS Base1 TO Usuario1

Seguridad en Bases de Datos

DAC

REVOKE

Permite revocar (quitar) permisos a un usuario o un rol.

REVOKE [GRANT OPTION FOR] <permiso>
ON <tipo de objeto> <nombre del objeto> FROM <usuario/rol>

Ejemplos:

REVOKE GRANT OPTION FOR ALL ON Tabla2 FROM Usuario2

REVOKE UPDATE ON COLUMN Tabla1.Columna5 FROM Usuario3

Notas:
En cada motor de base de datos hay particularidades para estas sentencias, por ejemplo dar permisos sobre columnas.

Seguridad en Bases de Datos

RBAC

Consiste en primero definir Roles con sus permisos y luego asignar esos Roles a los Usuarios.

La idea es no otorgar permisos directo sobre los usuarios.

La ventaja es fácilmente se pueden dar los mismos permisos a varios usuarios asignándoles el mismo rol.

Creación de un Rol

CREATE ROLE <Nombre del Rol>

Ejemplo:
CREATE ROLE OperadorBasico

Seguridad en Bases de Datos

Asignar un Usuario a un Rol

ALTER ROLE <Nombre del Rol>
  ADD MEMBER <Nombre Usuario>

Ejemplo:
ALTER ROLE OperadorBasico

  ADD MEMBER jperez

Desasignar un Usuario a un Rol

ALTER ROLE <Nombre del Rol>
  DROP MEMBER <Nombre Usuario>

Seguridad en Bases de Datos

MAC (también llamada Seguridad Multinivel)

Consiste en clasificar todos los objetos de la base de datos (tablas, vistas, etc.) según determinados niveles de seguridad.

Por ejemplo, los niveles podrían ser:

TS (Top Secret)
S (Secret)
C (Classified)
U (Unclassified)

TS > S > C > U

Luego a los usuarios se les asigna un nivel de seguridad y pueden acceder a todos los objetos de nivel igual o inferior.

No todos los RDBMS soportan este mecanismo de seguridad. Todos soportan DAC y RBAC, pero muy pocos MAC.

Por ejemplo: Oracle tiene una funcionalidad llamada Oracle Label Security que es una implementación del mecanismo MAC. Esta funcionalidad debe instalarse como algo adicional y se usa solo para bases de datos gubernamentales o de organismos de seguridad

Seguridad en Bases de Datos

SQL Injection

Es una técnica que aprovecha las vulnerabilidades de una aplicación que tiene sentencias SQL embebidas en su código fuente.

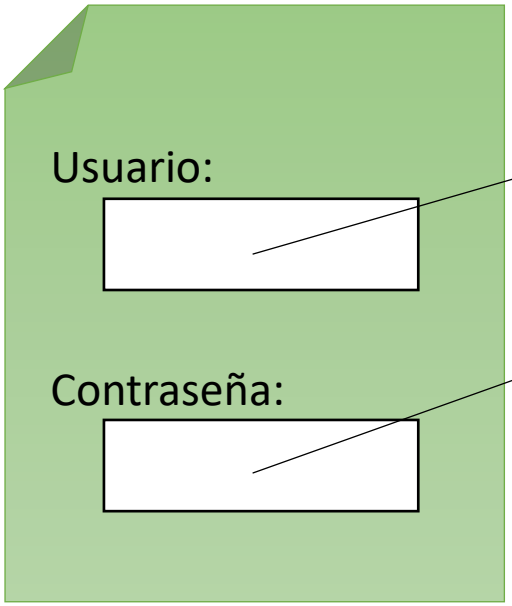
Básicamente, consiste en inyectar fragmentos de código SQL en el lugar donde el usuario debería ingresar valores de parámetros.

Seguridad en Bases de Datos

SQL Injection

Ejemplo:

Supongamos una página PHP con el siguiente código.



SQL = “SELECT * FROM Usuarios WHERE username=‘” + pUser + “’ AND password=‘” + pPwd + “’;”

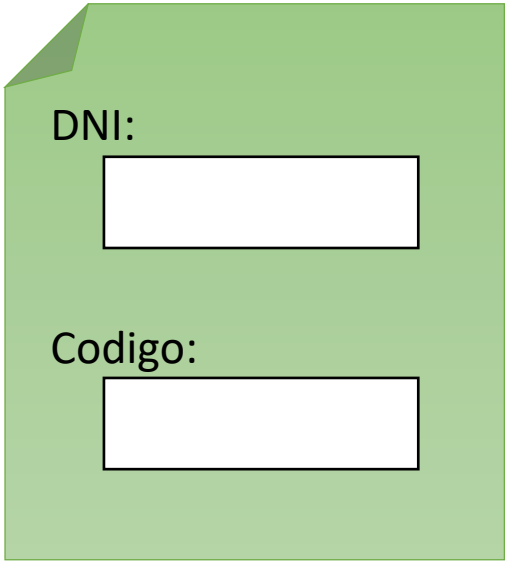
¿Qué pasaría si en el lugar donde va la contraseña ingresan lo siguiente?

blabla’ OR ‘a’= ‘a

SQL Injection

Ejemplo2:

Si los parámetros son valores numéricos, es aun mas fácil porque no hay que tener en cuenta las comillas.



SQL = “SELECT * FROM Usuarios WHERE dni=” + pDni + “ AND code=” + pCodigo + “;”

¿Qué pasaría si en el lugar donde va el DNI ingresan lo siguiente?

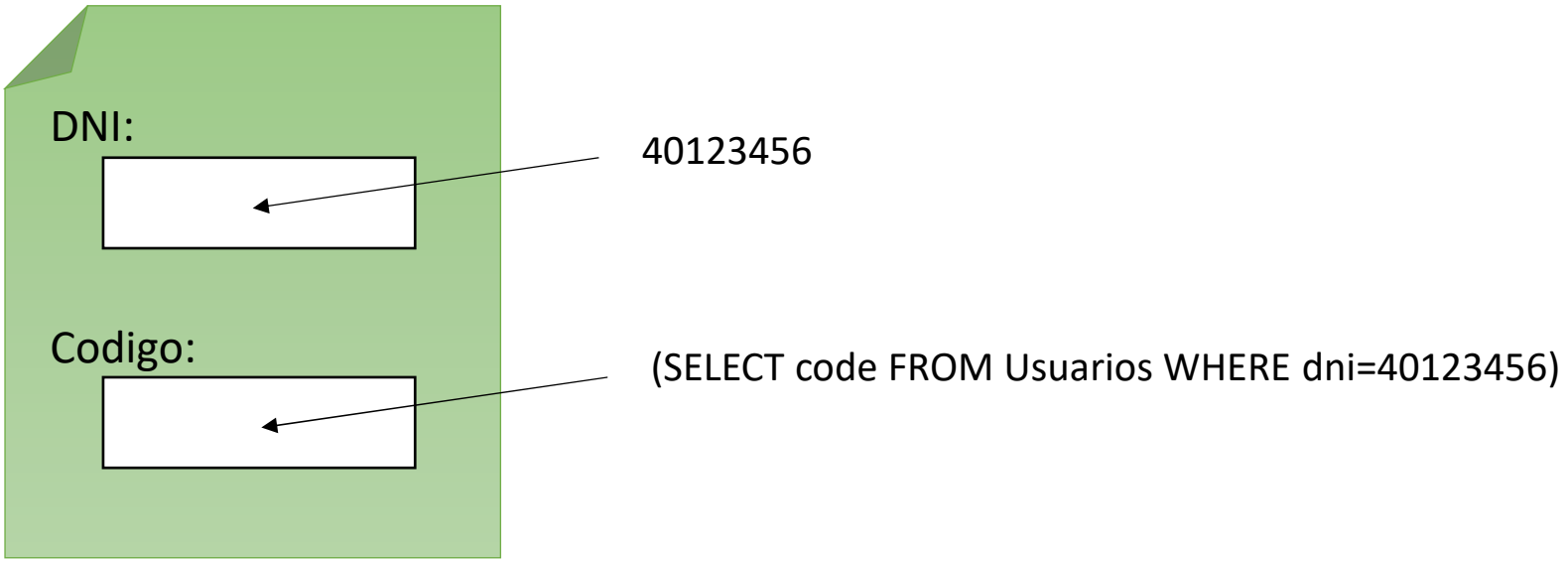
40123456 --

Los dos guiones después del DNI tal vez podrían comentar todo lo que sigue de ahí en adelante y dejarían sin efecto la otra parte de la consulta SQL.

SQL Injection

Ejemplo3:

¿Qué pasaría si ingresan lo siguiente?



Seguridad en Bases de Datos

SQL Injection

Las formas de evitar el SQL Injection son diversas:

- Validar los valores ingresados por el usuario y no permitir símbolos especiales como ser: comillas, espacios, paréntesis, etc.
- Evitar armar consultas SQL en el código fuente y en su lugar usar procedimientos almacenados
- Usar Binding de variables para los parámetros (depende del lenguaje de programación).