SISTEMAS DE BASES DE DATOS

CONTROL Y SEGURIDAD

DCL

DATA CONTROL LANGUAGE

Introducción

- SQL provee mecanismos de seguridad limitados.
- Se confía en el software para implementar un framework de seguridad más robusto.
- Los vendedores de Bases de Datos implementan varios aspectos de seguridad en distintas maneras.

Mecanismos de Seguridad Básica

Hay tres niveles de seguridad comunes para todos los RDBMS (Relational DataBase Management Systems):

Autenticación **A**utorización **A**uditar

En lo que difieren es en la manera en que cada uno de estos RDBMS implementan estos niveles.

Mecanismos de Seguridad Básica

✔✔ **Autenticación**: El usuario se conecta al RDBMS.

✔✔ **Autorización**: El usuario obtiene acceso a la base de datos o a los objetos del esquema de la base de datos para realizar ciertas acciones, basados en el set de privilegios asignados al usuario.

✔✔ **Auditar**: Para monitorear actividad sospechosa y realizar un análisis.

Autenticación

✔✔ La primera defensa es la autenticación.

✔✔ Antes de acceder al RDBMS se debe proveer suficiente información validada ya sea por el **RDBMS** o por el **sistema operativo** en el cual esta base de datos se encuentra instalada.

✔✔ Una vez que la identidad es autenticada, se puede proceder con el intento de acceder a los recursos, objetos y datos de la base de datos.

Autorización

✔✔ Una vez que el usuario es autenticado y se le concede acceso a la base de datos, el RDBMS emplea un complejo sistema de privilegios (permisos) para objetos particulares de la base de datos.

✔✔ Estos privilegios incluyen permisos para acceder, modificar, destruir o ejecutar objetos relevantes de la base de datos, así como añadir, modificar y borrar datos.

Auditar

✔ Provee medios para monitorear actividad de base de datos, ambas legítima y no autorizadas.
✔ Preserva la bitácora de intentos de acceso a la base de datos, ya sean exitosos o fallidos.
✔ Tambien los intentos de delete e inserts.
✔ Es un componente necesario para ser considerado para una certificación de seguridad.

Encriptación

✔✔ Provee una capa adicional de seguridad, protegiendo los datos de personas no autorizadas.

✔✔ Aún si se obtiene acceso a la base de datos, no será fácil descifrar datos encriptados.

https://dev.mysql.com/doc/refman/5.5/en/encryption-functions.html

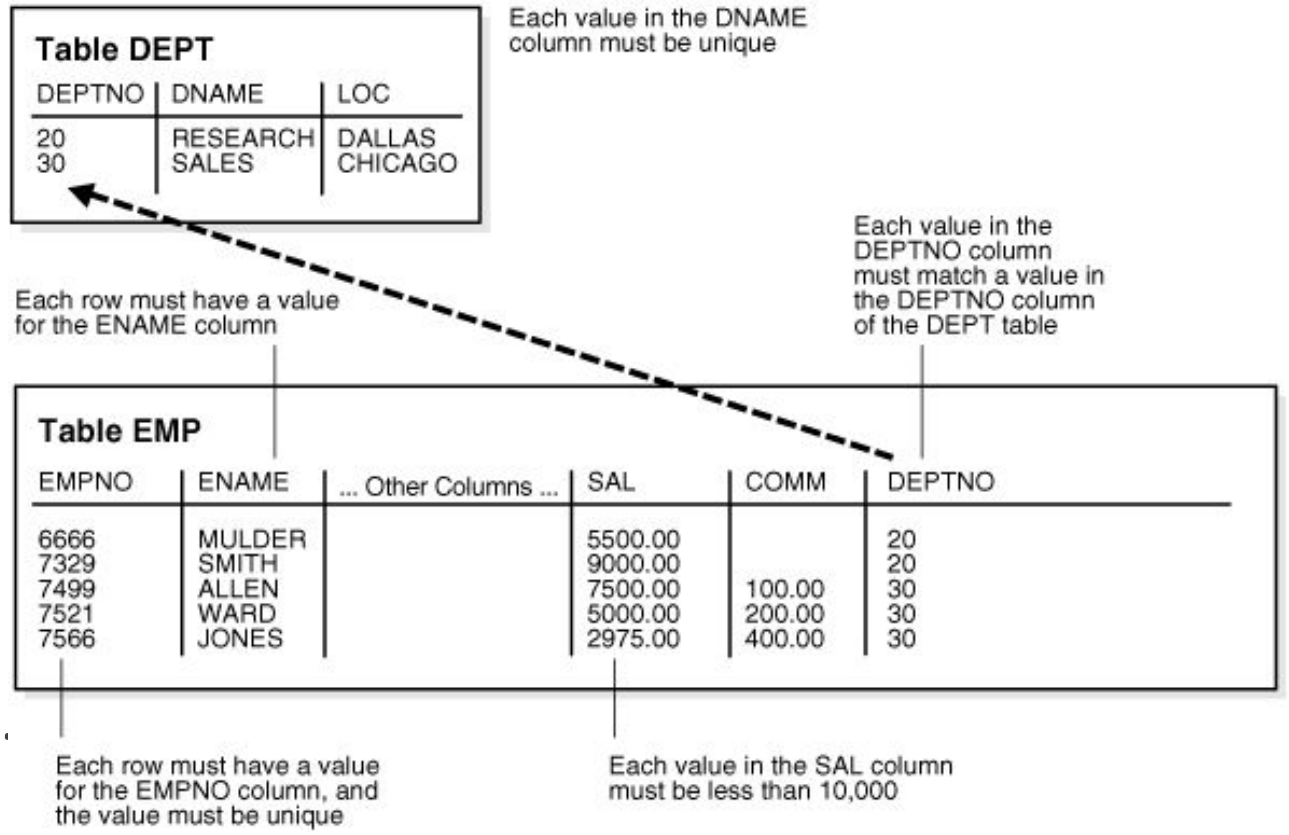
Integridad y Consistencia

✔✔ Mientras la seguridad se basa en la autenticación y autorización de procedimientos, la integridad de los datos juega el rol de proteger los datos de manipulación no intencional o maliciosa.

✔✔ Por ejemplo, aun si un usuario obtiene acceso a la base de datos (robando la clave), aun debe seguir reglas relacionales para manipular datos.

Integridad y Consistencia

a. Regla: No se permite registros huérfanos. Es decir no se puede borrar registros de una tabla de un padre sin entender las relaciones de la base de datos.

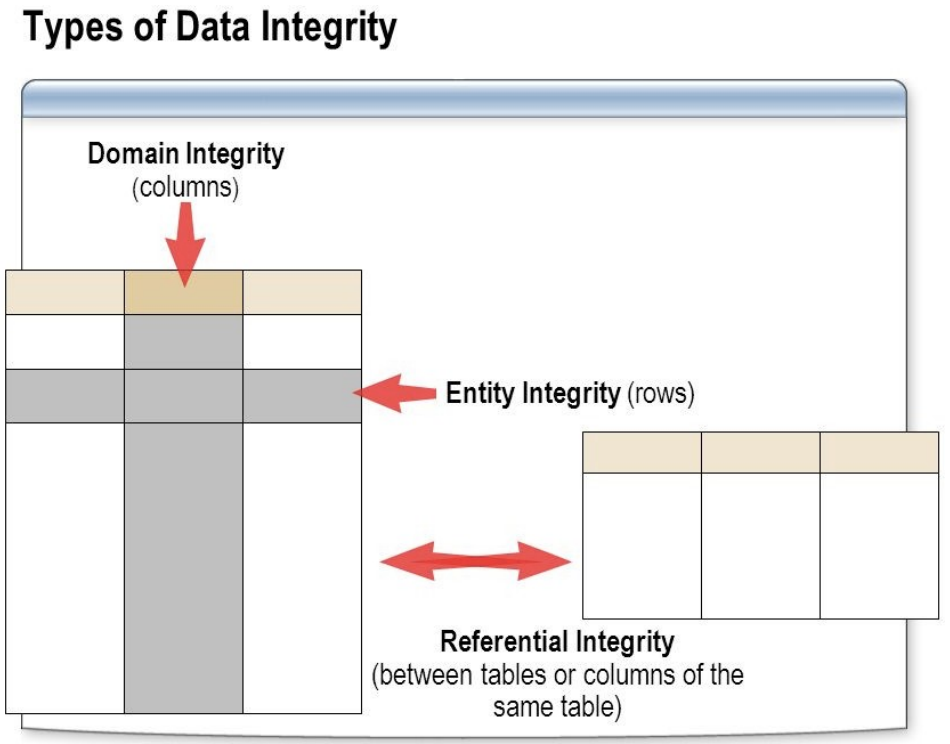


✔✔ Existen ciertos RDBMS que incluyen la característica ¨CASCADE¨ por defecto (remover registros hijos cuando se borrar el del padre).

Integridad y Consistencia

b. Regla: No insertar un registro duplicado en una columna protegida con un UNIQUE constraint.

c. Regla: No ingresar datos inválidos que violen un CHECK constraint.



Definiendo un Usuario:

Database User

✔✔ Es alguien que hace uso de los servicios que el servidor del RDBMS provee. Puede ser una aplicación, un administrador, o cualquiera que acceda a la base de datos en algún momento.

✔✔ Esta es la primera línea de defensa cuando hablamos de los aspectos de seguridad.

CREATE USER

```
CREATE USER <user> [IDENTIFIED BY [PASSWORD] 'password']
            [, <user2> [IDENTIFIED BY [PASSWORD]
            'password']];
```

```
CREATE USER alumno IDENTIFIED BY 'clavealumno';
```

```
CREATE USER 'gvsaltos'@'localhost' IDENTIFIED BY 'P@ssw0rd';
```

```
SHOW CREATE USER <user>;
```

DROP USER

```
DROP USER <username>;
```

En MySQL el usuario está separado de la base de datos

```
DROP USER 'gvsaltos'@'localhost';
```

ALTER USER

Provee una manera conveniente de modificar las propiedades del usuario sin la necesidad de hacerle DROP al usuario y luego recrearlo desde el principio.

```
ALTER USER <user_specification> [, user_specification]...
```

```
ALTER USER 'gvsaltos'@'localhost' PASSWORD EXPIRE;
```

```
SET PASSWORD FOR 'gvsaltos'@'localhost' = PASSWORD('pass2');
```

Administrando la seguridad con Prilivegios

✔✔ Un RDBMS es una colección de objetos, esquemas, tablas, vistas, procedimientos, etc. Así como los procesos que administran estos objetos.

✔✔ Restringir el acceso a estos objetos es un mecanismo de seguridad esencial implementado en el nivel de SQL a través del sistema de privilegios.

Privilegios

✔✔ Representan el derecho de un usuario en particular para acceder, crear, manipular y destruir varios objetos dentro de una base de datos, así como realizar tareas administrativas.

✔✔ Se los puede conceder a un usuario o ROLE.

✔✔ Se los puede dividir en dos categorías:

1. Privilegios de Sistema
2. Privilegios de Objetos

Privilegios a nivel de Sistema

✔Permite a los usuarios realizar tareas administrativa en un RDBMS.

✔Crear una base de datos.

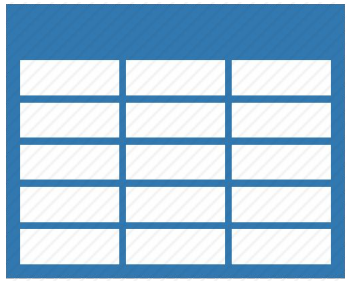
✔Crear y Borrar Usuario.

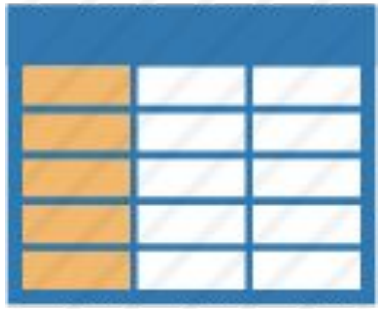
✔Crear, alterar y destruir objetos de bases de datos, etc.

✔Se necesita un alto nivel de autoridad en el RDBMS para conceder o tener privilegios de sistema.

Privilegios a nivel de Objetos

✔✔ Estos privilegios pueden llegar hasta el nivel de columna (si el objeto de la base de datos es una tabla o vista), o a cualquier otro objeto en la base de datos como stored procedures, funciones y triggers.





GRANT

```
GRANT [privilegio]
ON [<database>]. [<table>]
TO '<user>'@'localhost';
```

Nota:

Cada vez que se actualiza o se cambia permisos de usuario, hay que "refrescar" estos permisos utilizando el comando:

FLUSH PRIVILEGES;

```
SHOW GRANTS FOR '<user>'@'localhost';
```

GRANT en Mysql

(Todas las DBs)

```
CREATE USER ’admin'@'localhost' IDENTIFIED BY 'some_pass';
GRANT ALL PRIVILEGES ON *.* TO ’admin'@'localhost’ WITH GRANT OPTION;
```

```
GRANT SELECT, INSERT ON *.* TO ’admin'@’localhost’;
```

```
CREATE USER 'admin'@'localhost';
GRANT RELOAD,PROCESS ON *.* TO 'admin'@'localhost’;
```

```
SHOW PRIVILEGES;
SHOW PROCESSLIST;
```

GRANT en Mysql

(1 BD)

```
CREATE USER ’sistema'@'localhost';
GRANT EXECUTE ON PROCEDURE db1.sp1 TO ’sistema'@'localhost’;
```

```
CREATE USER 'monty'@'localhost' IDENTIFIED BY 'some_pass';
GRANT ALL PRIVILEGES ON db1.* TO 'monty'@'localhost’ WITH GRANT OPTION;
```

```
GRANT SELECT, INSERT ON db1.* TO 'monty'@’localhost’;
```

GRANT en Mysql

(1 Tabla)

```
CREATE USER ’ monty'@'localhost' IDENTIFIED BY 'some_pass';
GRANT ALL PRIVILEGES ON db1.tb1 TO ’ monty'@'localhost’ WITH GRANT OPTION;
```

```
GRANT SELECT, INSERT ON db1.tb1 TO ’monty'@’localhost’;
```

```
CREATE USER 'monty'@'localhost' IDENTIFIED BY 'some_pass';
GRANT SELECT (col1), INSERT (col1, col2) ON db1.tbl1 TO 'monty'@'localhost’;
```

RESUMEN (GRANT)

PRIVILEGIOS	DESCRIPCION
ALL PRIVILEGES	Un usuario puede acceder a todas las bases de datos del sistema.
CREATE	Permite crear objetos en la base de datos
DROP	Perimite eliminar objetos de la base de datos
DELETE	Permite eliminar registros de las tablas
INSERT	Permite insertar registros en las tablas
SELECT	Permite leer los registros de las tablas
UPDATE	Permite actualizar los datos de las tablas
GRANT OPTION	Permite agregar o remover privilegios de los usuarios.

REVOKE

```
REVOKE [privilegio]
ON [<database>]. [<table>]
FROM ‘<user>’@’localhost’;
```

```
> REVOKE INSERT ON *.* FROM 'jeffrey'@'localhost';
> REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...
```