Certificación y controles de seguridad en Bases de Datos: Sistemas de gestión de bases de datos

Miguel Expósito Martín

Universidad de Cantabria

miguel.exposito@unican.es

27/11/2018

Visión general

Definición

Definición

Un sistema de gestión de bases de datos (SGBD) es una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, a la que se conoce normalmente como base de datos, contiene información relevante para una organización. El objetivo primordial de un SGBD es ofrecer un medio conveniente y eficiente para almacenar y recuperar información de la base de datos.[Silberschatz, 2010]

Un SGBD es un paquete de software utilizado para definir, crear, utilizar y mantener una base de datos. Típicamente consiste de varios módulos software, cada uno con su propia funcionalidad.[Lemahieu, 2018]

Valor de mercado

Según Gartner, el valor de mercado total de los SGBD en 2015 era de 35.9 billones de dólares, un 8,7 % más comparado con el año anterior.

Arquitectura

Arquitectura

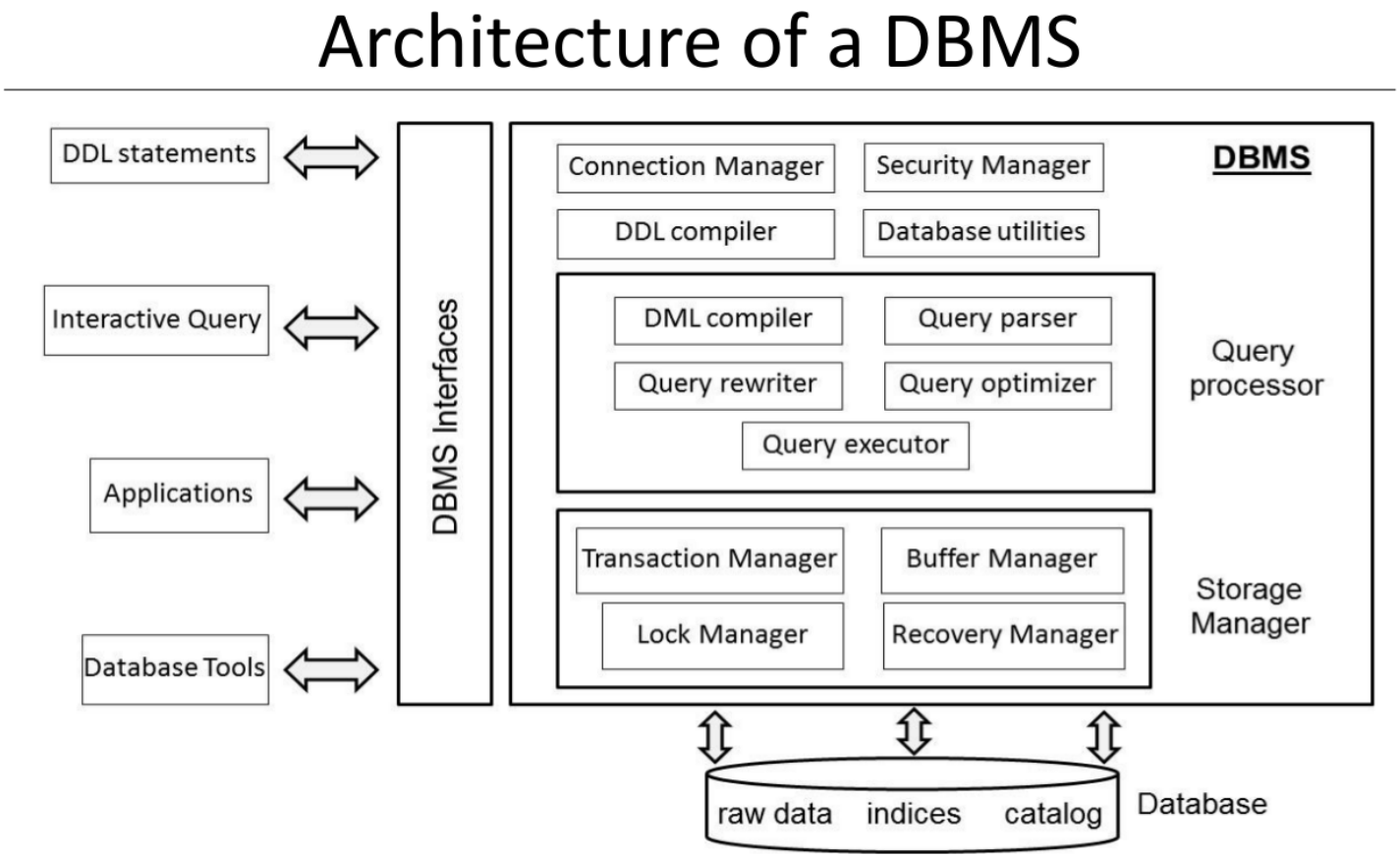

Figura: Arquitectura de SGBDs.[1]

[1] Fuente: [Lemahieu, 2018]

Arquitectura: gestor de conexiones y seguridad

- Proporciona facilidades para iniciar una conexión (remota) a una base de datos.
- Verifica credenciales de inicio de sesión (usuario/contraseña) y devuelve un manejador de conexión.
- Verifica si un usuario tiene los privilegios suficientes para ejecutar las acciones requeridas.

Arquitectura: compilador de DDL

- Compila las definiciones de datos especificadas en DDL(*Data Definition Language*)
- Parsea las definiciones de DDL, comprueba su corrección sintáctica y lo traduce a un formato interno.
- Una vez compilado con éxito, se registran las definiciones de datos en el catálogo.

Arquitectura: procesador de consultas

Una de las partes más importantes de un SGBD, asiste en la ejecución de consultas a la base de datos tales como recuperación, inserción, actualización o borrado de información. Generalmente consta de los siguientes componentes:

- **Compilador DML** (*Data Manipulation Language*): compila las sentencias de manipulación de datos. SQL es un DML declarativo.
- **Parser y reescritor de consultas**: parsea la consulta a un formato de representación interno. Puede reescribir la consulta para optimizarla independientemente del estado de la BD, utilizando un conjunto de reglas predefinidas específicas del SGBD.
- **Optimizador de consultas**: optimiza la consulta en función del estado de la BD, evaluando los distintos planes de ejecución. Estas estimaciones se llevan a cabo utilizando el catálogo y procedimientos de inferencia estadística.
- **Ejecutor de consultas**: se encarga de la ejecución de la consulta devuelta por el optimizador recuperando los datos solicitados desde el gestor de almacenamiento.

Arquitectura: procesador de consultas

- **Gestor de almacenamiento**: gobierna el acceso físico a los archivos y supervisa el correcto almacenamiento de datos. Consta de:
  - **Gestor de transacciones**: supervisa la ejecución de transacciones, creando una planificación con operaciones de lectura y escritura mejorando la eficiencia y rendimiento globales y garantizando atomicidad, consistencia, aislamiento y durabilidad (ACID).
  - **Gestor de buffer**: gestiona la memoria de buffer del SGBD. Es responsable de su cacheo y monitorización.
  - **Gestor de bloqueos**: juega un importante papel en el control de concurrencia, asegurando la integridad de los datos en todo momento. Evita conflictos entre transacciones previniendo actualizaciones simultáneas sobre un mismo elemento de datos.
  - **Gestor de recuperación**: recoge todas las operaciones de base de datos en un fichero de histórico, pudiendo ser utilizado para deshacer acciones o transacciones abortadas, así como para recuperación de caídas.

Arquitectura: utilidades e interfaces

- Un SGBD cuenta con diversas utilidades:
  - Utilidad de **carga**: soporta la importación de datos en la BD desde diferentes fuentes.
  - Utilidad de **reorganización**: reorganiza automáticamente la BD para mejorar su rendimiento.
  - Utilidades de **monitorización**: tiempos de respuesta, espacio de almacenamiento, etc.
  - Utilidades de **gestión de usuarios**: usuarios, roles y privilegios.
  - Utilidades de **copia de seguridad y recuperación**.
- Un SGBD necesita interactuar con diversos actores: diseñadores, administradores, aplicaciones, usuarios finales... Para facilitar dicha comunicación, se proporcionan varios interfaces de usuario (web, línea de comandos, GUI de escritorio, etc.). Ejemplo: MySQL Workbench.

Categorización

Categorización

- Basada en el modelo de datos
  - **Jerárquica**: modelo basado en árbol. Procedural y orientada a registro. No tiene procesador de consultas. Ejemplo: el registro de *Windows*.
  - **De red**: modelo basado en red, más flexible que el jerárquico. Procedural y orientada a registro, tampoco tiene procesador de consultas. CODASYL es uno de los tipos de BD jerárquica más populares.
  - **Relacional**: utiliza el modelo relacional, el más usado en la industria.SQL se utiliza tanto para DML como para DDL. Es declarativo y orientado a conjuntos. Sí tiene procesador de consultas y presenta separación entre modelo lógico y de datos internos. Ejemplos: *Oracle, MySQL, SQL Server...*

Categorización

- Basada en el modelo de datos
  - **Orientada a objetos**: basada en el modelo del mismo nombre. Ejemplo: *db4o*. No son muy populares en la industria debido a su complejidad.
  - **Relacional extendida**: utiliza el modelo relacional extendido con conceptos de orientación a objetos. La mayor parte de las BBDD relacionales incorporan este tipo de extensiones.
  - **XML**: utiliza el modelo XML para almacenar datos. Las BBDD relacionales también incorporan extensiones para XML. Ejemplo: *eXist*.
  - **NoSQL** (documentos, grafos): almacenes clave-valor, de documentos (JSON), de grafos (RDF)... Ejemplos: *MongoDB, Neo4j*.
- Basada en el grado de acceso simultáneo
  - **Monousuario**: solo un usuario a la vez. Ej: *MS Access*.
  - **Multiusuario**: múltiples usuarios interactuando simultáneamente.

Categorización

- Basada en la arquitectura
  - **Centralizada**: mainframes que procesan todas las consultas en un único host.
  - **Cliente-servidor**: clientes activos solicitan servicios a servidores pasivos.
  - **N-capas**: GUI + servidor de aplicaciones + BD.
  - **En la nube**: alojada por terceros, pueden distribuir los datos entre múltiples máquinas. Ej: *Cassandra*.
  - **Federada**: proporciona una interfaz uniforme a múltiples fuentes de datos subyacentes.
  - **En memoria**: almacena los datos en memoria. Se utilizan para propósitos de tiempo real.

Categorización

- Basada en el tipo de uso
  - **Transaccional (OLTP)**: enfocada en gestión operacional o transaccional (ej: TPV). Deben tener buen soporte para grandes volúmenes de consultas pequeñas.
  - **Analítica (OLAP)**: enfocada en utilizar datos operativos con enfoques tácticos o estratégicos. En este paragidma, un número limitado de usuarios lanza consultas complejas para analizar altos volúmenes de datos.
  - **Multimedia**: permite almacenar datos en forma de texto, imágenes, audio, vídeo...Proporcionan motores de consulta basados en el contenido.
  - **Espacial**: soporta el almacenamiento y consulta de datos espaciales. Es la base de los GIS. Ej: PostgreSQL.
  - **Móvil**: se ejecuta sobre *smartphones*. Presenta una baja huella de memoria, ideal para potencia de cálculo limitada. Ejemplos: *SQLite, Oracle Lite, SQL Server Compact*, etc.

Ejercicio 2.1: seleccione la respuesta correcta

1. ¿Cuál de los siguientes componentes **no es parte** del gestor de almacenamiento en una arquitectura de SGBD?
   a. Gestor de conexiones
   b. Gestor de transacciones
   c. Gestor de buffer
   d. Gestor de recuperación

2. CODASYL es un ejemplo de...
   a. SGBD jerárquico
   b. SGBD en red
   c. SGBD relacional
   d. SGBD orientado a objetos

Ejercicio 2.1: seleccione la respuesta correcta

3 ¿Cuál de los siguientes tipos de SGBD **no pertenece** a una clasificación basada en modelo de datos?

a. SGBD jerárquico
b. SGBD en red
c. SGBD en la nube
d. SGBD objeto-relacional

4 Si necesita utilizar una arquitectura de SGBD que permita el acceso a múltiples fuentes de datos y proporcione una interfaz uniforme que oculte los detalles de bajo nivel, el SGBD más apropiado sería...

a. SGBD de n-capas
b. SGBD en la nube
c. SGBD cliente-servidor
d. SGBD federado

Ejercicio 2.1: seleccione la respuesta correcta

5 ¿Qué afirmación **no es correcta**?

a. Un SGBD relacional presenta una separación completa entre el modelo de datos interno y el lógico.
b. Una arquitectura de SGBD en la nube es aconsejable para empresas pequeñas.
c. El procesador/re-escritor de consultas optimiza una consulta basándose en el estado actual de la BD.
d. Un SGBD en memoria almacena todos los datos en la memoria interna en lugar de utilizar almacenamiento externo más lento, como el basado en disco.

Ejercicio 2.1: seleccione la respuesta correcta

6 ¿Qué afirmación **es correcta**?

a. El compilador de DDL procesa sentencias DDL, comprueba su corrección semántica, las traslada a un formato interno y registra las definiciones de datos en el catálogo.
b. El procesador/re-escritor de consultas optimiza las consultas basándose en el estado actual de la base de datos.
c. Un SGBD OLTP se utiliza principalmente para toma de decisiones estratégicas y tácticas.
d. Un SGBD con capacidades XML permite almacenar datos en este formato; sin embargo, no permite lanzarle consultas.

Seguridad en SGBDs

Seguridad en SGBDs

En general, todos los SGBDs disponen de mecanismos más o menos sofisticados para garantizar la seguridad en sus distintas dimensiones. Algunos de estos mecanismos son:

- Configuración segura a nivel de red y de sistema operativo.
- Conexiones seguras cliente-servidor.
- Funcionalidades de auditoría.
- Encriptación de datos.
- Anonimización o enmascarado de datos.
- Gestión de usuarios y roles.

En las siguientes secciones se revisarán estos mecanismos a través de un caso de uso: MariaDB, un clon de MySQL.

Caso de uso: MySQL

Instalación

Ejercicio 2.2: instalación de un SGBD

Para poder realizar este y otros ejercicios, será necesario disponer de una instancia de base de datos de pruebas. En este caso, se instalará MariaDB. MariaDB es un clon de MySQL, un SGBD opensource muy popular en la industria que puede descargarse de forma gratuita desde su web.

Para este ejercicio, se utilizará Vagrant para desplegar una VM con MariaDB. Se trata de una herramienta que permite la creación y configuración de entornos de desarrollo ligeros, reproducibles y portables.

¿Qué es Vagrant en realidad?

No es más que un wrapper sobre Virtualbox o VMware. Un archivo de configuración define la máquina, el software a instalar y la forma de acceso.

ADIOS A LA FRASE "EN MI MÁQUINA FUNCIONA"

Ejercicio 2.2: instalación de un SGBD

Como paso previo, será necesario instalar Virtualbox. Para ello:

1. Descargar el archivo de instalación para la plataforma adecuada.
2. Seguir el proceso de instalación.

Para instalar Vagrant:

1. Descargar el archivo de instalación para la plataforma adecuada
2. Seguir el proceso de instalación.
3. Una vez finalizado, crear un directorio de trabajo (p.ej., *seguridad*)
4. En dicho directorio, crear un fichero *Vagrantfile* con el siguiente contenido:

Fichero Vagrantfile

```
Vagrant.configure("2") do |config|
config.vm.box = "ubuntu/trusty64"
end
```

Ejercicio 2.2: instalación de un SGBD

Para iniciar sesión en la VM:

- Arrancar la máquina de vagrant ejecutando desde la shell en el directorio donde reside el Vagrantfile:

Arranque de la VM con Ubuntu

```
vagrant up
vagrant ssh
```

Importante: sobre Vagrant

- La contraseña de superusuario es **vagrant**
- El directorio donde se ubica el *Vagrantfile* se comparte en la VM invitada (*guest*) en **/vagrant/**

Ejercicio 2.2: instalación de un SGBD

Para instalar MariaDB:

Instalación de MariaDB en Ubuntu

```
sudo apt-get install software-properties-common
sudo apt-key adv --recv-keys --keyserver  \
hkp://keyserver.ubuntu.com:80 0xcbcb082a1bb943db
sudo add-apt-repository 'deb  \
http://tedeco.fi.upm.es/mirror/mariadb/repo/10.3/ubuntu \
trusty main'
sudo apt-get update
sudo apt-get install mariadb-server
```

Durante el proceso de instalación, se solicitará una contraseña para el usuario **root**.

Ejercicio 2.2: instalación de un SGBD

Para comprobar que la instalación es correcta:

Arranque de la VM con Ubuntu

```
# verificar que la BD escucha en el puerto correspondiente
netstat -anp | grep 3306
# iniciar sesión con el usuario root
mysql -u root -p
# mostrar los schemas existentes
show databases
```

¿qué contraseña habéis puesto?

root, prueba, admin, blank, mysql...

mejor algo así...

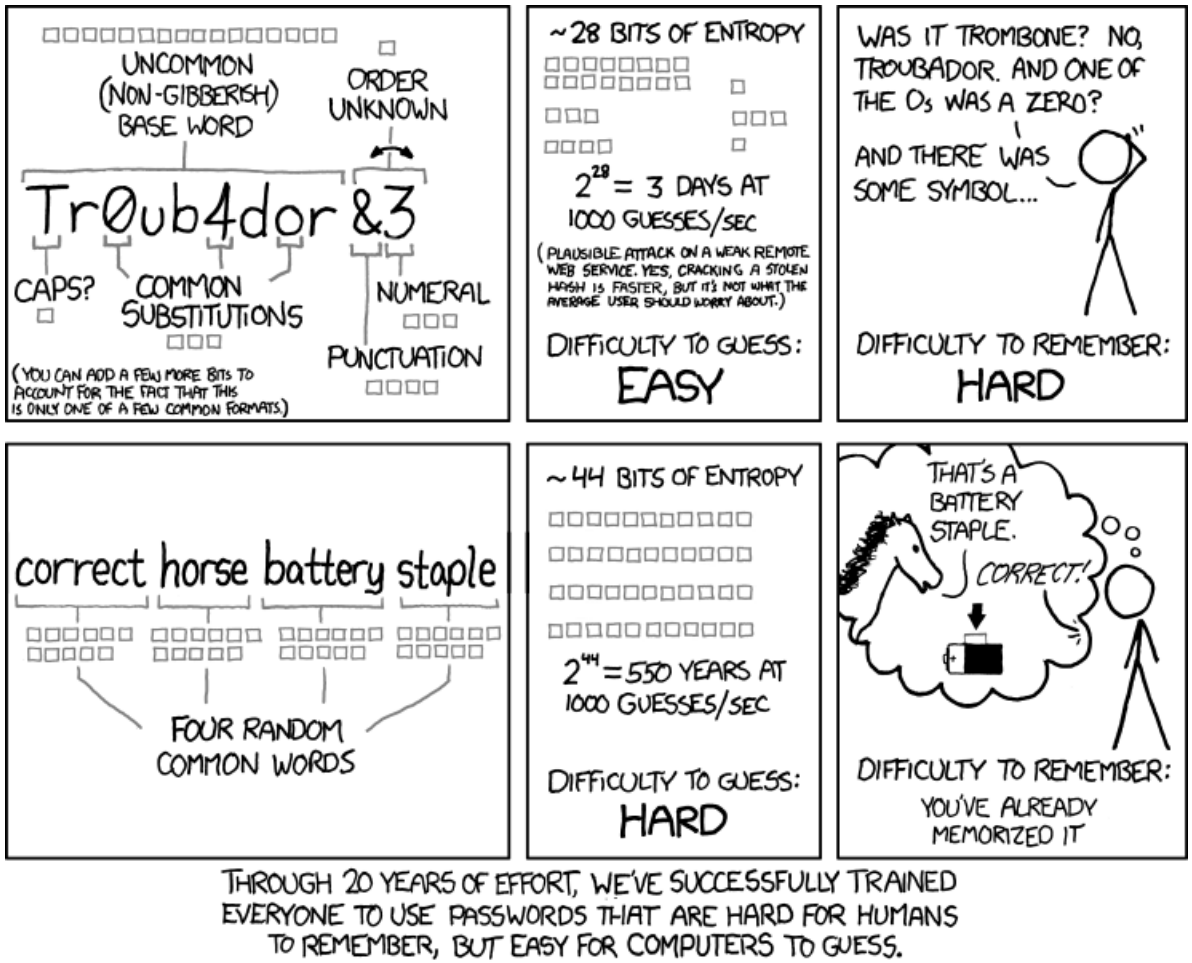

Figura: Password strength.[2]

[2] Fuente: https://xkcd.com/936/

Seguridad

Seguridad en MariaDB: contraseñas

MariaDB 10.1 introduce una API de validación de contraseñas (ver Password Validation Plugin API). Esto permite crear validadores de contraseñas personalizados así como implementar autenticación de dos factores (por ejemplo, con *Google Authenticator*).

Esta versión también ofrece extensiones listas para usar, como `simple_password_check`, para el establecimiento de políticas de contraseñas, o cracklib_password_check, que comprueba la robustez de la contraseña según la biblioteca *CrackLib*.

Seguridad en MariaDB: autenticación

Desde su versión 5.2, MariaDB cuenta con la extensión de autenticación PAM, que permite a los DBAs establecer un entorno de base de datos en el que los usuarios puedan utilizar las mismas contraseñas de los inicios de sesión a través de shell y otros servicios.

Además, la biblioteca compartida pam_ldap permite que los usuarios se autentiquen contra un servidor LDAP.

Finalmente, para la versión 10.2 de la BD estaba planificada la puesta en producción de una extensión con soporte para Kerberos.

Seguridad en MariaDB: roles y encriptación

La versión 10.0 ya introdujo los roles de usuario, funcionalidad que ha sido mejorada en la versión 10.1. Los DBAs pueden empaquetar un conjunto de privilegios y asociarlos a un rol, pudiendo conceder un rol y sus privilegios asociados a uno o varios usuarios. Asimismo, se puede asignar un rol por defecto a cada usuario creado.

Por último, el soporte para encriptación es quizá el aspecto relacionado con la seguridad de la información más interesante de la versión 10.1. Permite encriptar *tablespaces*, tablas y logs.

Securización

Securización de MariaDB: líneas generales

- Solo el usuario *root* debe tener acceso a la tabla user.
- Nunca conceder privilegios innecesarios o a todos los hosts.
- Nunca almacenar contraseñas en texto plano (almacenar un hash SHA2).
- Securizar la red con un cortafuegos. Bloquear el puerto 3306 excepto a hosts autorizados.
- Nunca enviar datos sin encriptar a través de la red (usar SSH, TLS).
- Requerir que todas las cuentas de usuario tengan contraseña.
- Garantizar que la única cuenta de usuario de Linux con permisos de lectura o escritura en los directorios de la base de datos es la utilizada para ejecutar *mysqld*.
- Nunca ejecutar el proceso servidor como usuario *root*.
- Eliminar usuarios anónimos y *database test*.
- No permitir que *root* acceda remotamente.

Securización de MariaDB: opciones mysqld

- `--local-infile=0`: hará que el comando `LOAD DATA LOCAL` falle. Este comando carga un archivo de texto en una tabla con gran rapidez; `LOCAL` hace que el archivo se lea en el cliente.
- `--safe-show-database`: sustituido en MariaDB 5.5 por el privilegio `SHOW DATABASES`. Permite al usuario listar las bases de datos existentes.
- `--safe-user-create`: impide la creación de nuevos usuarios por un usuario que no tiene privilegios de escritura en la tabla `mysql.user`
- `--secure-auth`: impide la autenticación con cuentas que tengan contraseñas en formato anterior a MySQL 4.1.
- `--skip-name-resolve`: si vale 1, solo se utilizarán IPs para las conexiones (no DNS).
- `--skip-symbolic-links`: usando enlaces simbólicos, mysqld podría usarse para eliminar o renombrar un archivo fuera del directorio de datos. Versiones anteriores a 5.0.67 contienen una vulnerabilidad relacionada (CVE-2008-4097).

Securización de MariaDB: otras recomendaciones

- No usar `--skip-grant-tables`: este flag permite conectarse a cualquiera sin usuario, contraseña y privilegios. Deshabilita la gestión de cuentas de usuario.
- En Windows, no usar `--enabled-named-pipe`. Usar TCP.
- No conceder los privilegios `PROCESS`, `FILE`, `SUPER` a usuarios no administradores.
- No ejecutar MariaDB en el mismo servidor que un backend de una aplicación. Muchos servidores web tienen vulnerabilidades conocidas que permitirían a un atacante acceder a la base de datos.
- Deshabilitar el control total de la base de datos a usuarios locales y los permisos por defecto para usuarios remotos.
- No utilizar versiones previas a 4.1.X. (En general, usar siempre las últimas versiones).
- No permitir a los desarrolladores el acceso a entornos de producción.
- Activar las funciones de auditoría.

Ejercicio 2.3: securizar MariaDB

`mysql_secure_installation` es un *shell script* disponible para sistemas Unix que permite mejorar la seguridad de una instalación de MariaDB de las siguientes formas:

- Asignando una contraseña a las cuentas de *root*.
- Eliminando cuentas de *root* con acceso desde fuera del host local.
- Eliminando cuentas de usuarios anónimos
- Eliminando la base de datos de *test*, que puede ser accedida por defecto por usuarios anónimos

Ejecute el *script* y responda a las preguntas interactivamente. Por otra parte, compruebe qué usuario del sistema operativo está ejecutando el proceso `mysqld`. ¿Lo considera seguro?

Conexiones seguras

Conexiones seguras en MariaDB

Por defecto, MariaDB transmite los datos en claro entre el servidor y los clientes. Esto puede ser aceptable si el servidor y el cliente se ejecutan en la misma máquina, pero no lo es cuando no se da esta circunstancia.

Para mitigar este problema, MariaDB permite encriptar los datos entre el cliente y el servidor usando TLS (*Transport Layer Security*).

Comprobar soporte para TLS

```
SHOW VARIABLES LIKE 'have_ssl';
```

Si el valor de `have_ssl` es DISABLED implica que el SGBD tiene soporte para TLS pero este está desactivado. Antes de activarlo, será necesario generar los certificados de servidor y cliente.

Ejercicio 2.4: creación de certificados SSL

Paso previo: directorio de trabajo

```
$ su
$ cd /etc/mysql
$ mkdir ssl
$ cd ssl
```

Creación de certificado de CA

```
# openssl genrsa 2048 > ca-key.pem
# sudo openssl req -new -x509 -nodes -days
365000 -key ca-key.pem -out ca-cert.pem
```

Ejercicio 2.4: creación de certificados SSL

Creación de certificado de servidor

```
# openssl req -newkey rsa:2048 -days 365000 -nodes
 -keyout server-key.pem -out server-req.pem
# openssl rsa -in server-key.pem -out server-key.pem
# openssl x509 -req -in server-req.pem -days 365000
 -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 -out
  server-cert.pem
```

Ejercicio 2.4: creación de certificados SSL

Creación de certificado de cliente

```
# openssl req -newkey rsa:2048 -days 365000 -nodes
 -keyout client-key.pem -out client-req.pem
# openssl rsa -in client-key.pem -out client-key.pem
# openssl x509 -req -in client-req.pem -days 365000
 -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 -out
  client-cert.pem
```

Verificación de certificados

```
# openssl verify -CAfile ca-cert.pem server-cert.pem
 client-cert.pem
```

Ejercicio 2.4: conexiones seguras

Activar SSL en el servidor

```
# vi /etc/mysql/mariadb.conf.d/50-server.cnf
```

Contenido a añadir

```
[mysqld]
ssl
ssl-ca=/etc/mysql/ssl/ca-cert.pem
ssl-cert=/etc/mysql/ssl/server-cert.pem
ssl-key=/etc/mysql/ssl/server-key.pem
```

Reiniciar servidor

```
# /etc/init.d/mysql restart
```

Ejercicio 2.4: conexiones seguras

Configurar SSL en cliente

```
# vi /etc/mysql/mariadb.conf.d/50-mysql-clients.cnf
```

Contenido a añadir

```
[mysql]
ssl-ca=/etc/mysql/ssl/ca-cert.pem
ssl-cert=/etc/mysql/ssl/client-cert.pem
ssl-key=/etc/mysql/ssl/client-key.pem
```

Los certificados de CA y cliente así como la clave privada del cliente deben copiarse a todos los clientes desde donde se desee conectar.

Ejercicio 2.4: conexiones seguras

Conexión con el cliente

```
# mysql -u root -p
```

Comandos a ejecutar en la BD

```
MariaDB [(none)]> SHOW VARIABLES LIKE '%ssl%';
MariaDB [(none)]> status;
```

Comprobar el contenido del parámetro SSL.

Ejercicio 2.5: conexión segura desde Python

Crear un pequeño *script* en *Python* que, usando el paquete `MySQLdb`, utilice los certificados de cliente para conectarse a la base de datos y ejecutar el comando `"SHOW STATUS LIKE 'Ssl_cipher'"`, cuyo resultado deberá mostrar por consola.

Encriptado de tablas y tablespaces

Data at Rest Encryption

También conocida como *"Transparent Data Encryption"*, esta tecnología permite encriptar los datos de las tablas almacenadas en disco. Está soportada en varios motores de almacenamiento (*XtraDB, InnoDB, Aria*). Entre sus limitaciones:

- No se encriptan metadatos ni datos enviados al cliente.
- El *log* de error no se encripta.
- El complemento de auditoría no puede generar salida encriptada.

Para utilizar esta funcionalidad, es necesario instalar un complemento que gestione las claves de encriptación (ej: File Key Management Plugin).

El encriptado ocurre al escribir páginas a disco. También es posible encriptar archivos temporales y archivos de log binarios.

Ejercicio 2.6: Encriptación de tabla (1/6)

Se va a proceder a encriptar una tabla con la funcionalidad de Data Rest Encryption de MariaDB. En primer lugar, será necesario habilitar el File Key Management Plugin. Para ello, habrá que preparar un archivo con las claves. Con el usuario **root**:

Archivo de claves

```
# openssl rand -hex 16 >> /etc/mysql/keys
# openssl rand -hex 16 >> /etc/mysql/keys
# openssl rand -hex 16 >> /etc/mysql/keys
```

Ejercicio 2.6: Encriptación de tabla (2/6)

Una vez creado y poblado el archivo, será necesario editarlo y añadir los identificadores a cada línea, de esta forma:

Identificadores de clave

```
# Keys
1;a3c93624f4968eb95056b6902de874ef
2;04e478eefe15b03c836282464b0e94a2
3;8c8ada2dfb4542b8e2673703f0364079
```

Ejercicio 2.6: Encriptación de tabla (3/6)

A continuación, se deberá encriptar el archivo:

Encriptado de archivo de claves

```
# openssl enc -aes-256-cbc -md sha1 -k mi_password \
-in /etc/mysql/keys -out /etc/mysql/keys.enc
```

En donde *mi_password* será un *password* robusto a elegir por el alumno. Una vez terminado, se puede borrar el archivo de claves en texto plano.

Ejercicio 2.6: Encriptación de tabla (4/6)

Una vez preparados los distintos archivos, se puede configurar MariaDB para usar el *File Key Management Plugin*. Para ello, modificar el archivo de configuración `/etc/my.cnf`

Configuración del plugin

```
[mysqld]
...
# File Key Management
plugin_load_add = file_key_management
file_key_management_filename = /etc/mysql/keys.enc
file_key_management_filekey = FILE:/etc/mysql/.key
file_key_management_encryption_algorithm = aes_cbc
```

Una vez modificada la configuración, reiniciar el servidor MariaDB. El archivo `.key` contendrá el *password* utilizado para encriptar el archivo de claves. **¡Cuidado con los permisos!**

Ejercicio 2.6: Encriptación de tabla (5/6)

Comprobar que el plugin está activado iniciando sesión con el usuario root en MariaDB y ejecutando el comando `SHOW plugins;`. A continuación, crear una base de datos de prueba y una tabla encriptada:

Creación de tabla encriptada

```
create table encrypt_test1 (
id bigint,
str varchar(256)
) engine = InnoDB
encrypted = YES
encryption_key_id = 1;
```

La información sobre las tablas encriptadas puede mostrarse así:

```
SELECT * FROM information_schema.innodb_tablespaces_encryption\G;
```

Ejercicio 2.6: Encriptación de tabla (6/6)

¿Cómo comprobar en el sistema de archivos que los datos de la tabla han sido, en efecto, encriptados? Utilice la tabla staff y el nombre *'Mike'* para probarlo.

¿Dónde debería realizarse la encriptación?

Dependiendo del problema a resolver, la encriptación puede tener lugar en varios puntos del sistema completo:

- **Disco completo:** muy sencillo, pero sólo protege de la pérdida física del almacenamiento. Útil para dispositivos móviles.
- **Nivel de archivo:** utiliza agentes de encriptación específicos según el sistema operativo. Suele utilizarse para proteger archivos sensibles de configuración.
- **Base de datos:** cada proveedor tiene sus soluciones específicas. Algunas partes del SGBD quedan sin encriptar. Han de complementarse con seguridad de las conexiones cliente-servidor.
- **Aplicación:** requiere de esfuerzos de desarrollo, pero es la más utilizada. Ejemplo: **nunca almacenar contraseñas en claro en una BD, sino su hash SHA2 (por ejemplo).**

Auditoría

Auditoría

MariaDB presenta un complemento o *plugin* de auditoría cuyo propósito fundamental es garantizar la trazabilidad de la actividad del servidor. Almacena quién se conecta al servidor (nombre y *host*), que consultas se lanzan, qué tablas se consultan y qué variables se cambian. Esta información se almacena en un archivo de *log* rotatorio o puede ser enviada al *syslogd* local.

Para instalar el *plugin*, iniciar sesión como **root** y ejecutar el comando `SHOW GLOBAL VARIABLES LIKE 'plugin_dir';`, que mostrará la ruta en la que se almacenan los plugins. A continuación, se puede comprobar la existencia en dicho directorio del fichero `server\_audit.so`.

Ejercicio 2.7: instalación del plugin server_audit

Para instalar el *plugin*, iniciar sesión como **root** y ejecutar:

Activación del plugin server audit

```
INSTALL SONAME 'server_audit';
SHOW GLOBAL VARIABLES LIKE 'server_audit%';
SET GLOBAL server_audit_events='CONNECT,QUERY,TABLE';
SET GLOBAL server_audit_logging=ON;
```

El *plugin* tan solo tiene tres tipos de eventos a registrar. Para comprobar su funcionamiento, se puede hacer un seguimiento al fichero generado mientras se realizan consultas y operaciones en la BD con distintos usuarios:

Comprobación de la trazabilidad

```
# tail /var/lib/mysql/server_audit.log
```

Anonimización y enmascarado

Anonimización y enmascarado

La anonimización es el proceso mediante el cual se transforman datos sensibles de forma que el valor real de los mismos no pueda ser recuperado por otros usuarios, como un analista de datos o un desarrollador. Pueden utilizarse distintas técnicas:

- Agregación: resúmenes de datos que no comprometen microdatos o datos de individuos (medias, desviaciones, etc.)
- Discretización: en vez de proporcionar el dato exacto, se particiona en clases disjuntas y mutuamente exclusivas. Ejemplo: grupos de edad.
- Distorsión: se añade ruido a los valores reales.
- Generalización: por ejemplo, no utilizar direcciones completas sino sólo a nivel de localidad.

MariaDB no presenta funcionalidades de anonimización de forma nativa, pero ofrece otro producto: *MaxScale*, un *proxy* o pasarela inteligente que hace *forwarding* o reenvío de sentencias SQL a distintos SGBDs y que sí dispone de funcionalidades de enmascarado.

Resumen

Referencias

Abraham Silberschatz et al. (2010)

Database System Concepts, 6th edition

Wilfried Lemahieu et al. (2018)

Principles of Database Management