



**EDUCACIÓN**  
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO  
NACIONAL DE MÉXICO®

Instituto Tecnológico de Tlaxiaco

# **TECNOLÓGICO NACIONAL DE MÉXICO INSTITUTO TECNOLÓGICO DE TLAXIACO**

## **SEGURIDAD Y VIRTUALIZACIÓN**

### **Práctica 3 - Bases de datos seguras**

#### **CARRERA:**

**INGENIERIA EN SISTEMAS COMPUTACIONALES**

#### **INTEGRANTES:**

**SANDRA YOLOTZIN REYES GARCÍA – 19620079**

**LUZ KARINA REYES LÓPEZ – 21620184**

**JERONIMA ROQUE CABALLERO – 21620206**

#### **DOCENTE**

**OSORIO SALINAS EDWARD**

**Tlaxiaco, Oax., Agosto de 2024.**

## TABLA DE ILUSTRACIONES

Ilustración 1.....	3
Ilustración 2.....	3
Ilustración 3.....	4
Ilustración 4.....	4
Ilustración 5.....	4
Ilustración 6.....	5
Ilustración 7.....	5
Ilustración 8.....	6
Ilustración 9.....	6
Ilustración 10.....	7
Ilustración 11.....	7
Ilustración 12.....	8
Ilustración 13.....	8
Ilustración 14.....	9
Ilustración 15.....	9
Ilustración 16.....	10
Ilustración 17.....	10
Ilustración 18.....	10
Ilustración 19.....	11
Ilustración 20.....	11
Ilustración 21.....	12
Ilustración 22.....	15
Ilustración 23.....	16
Ilustración 24.....	17

Primero instalamos MySQL, después ingresamos con nuestra contraseña que asignamos durante el proceso de instalación.

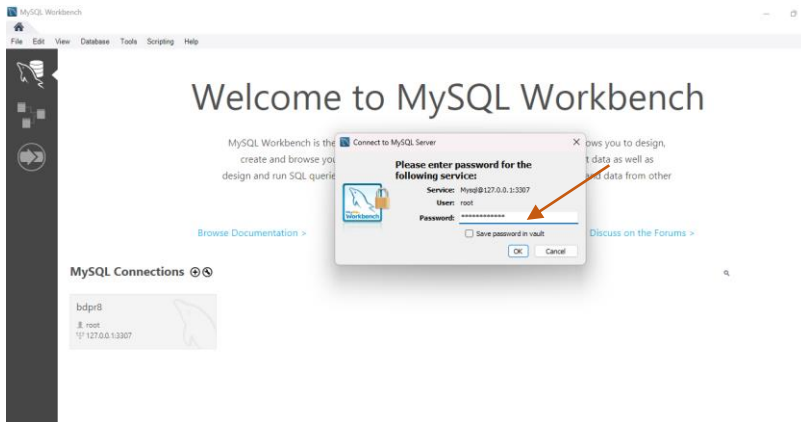


Ilustración 1

Luego en la parte de Schemas damos clic derecho y seleccionamos la opción que dice Create Schema para crear una base de datos con las siguientes tres tablas diferentes: users, address y customers.

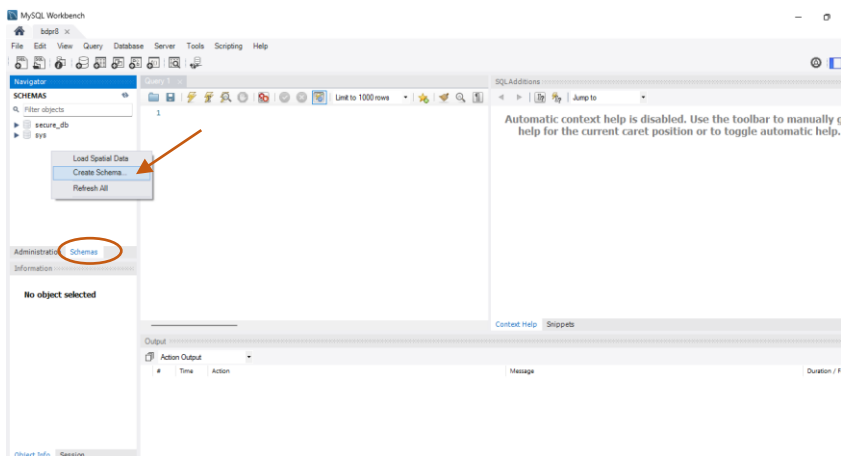


Ilustración 2

En esta parte le asignamos el nombre a la base de datos y damos clic en apply para que se cree la base de datos.

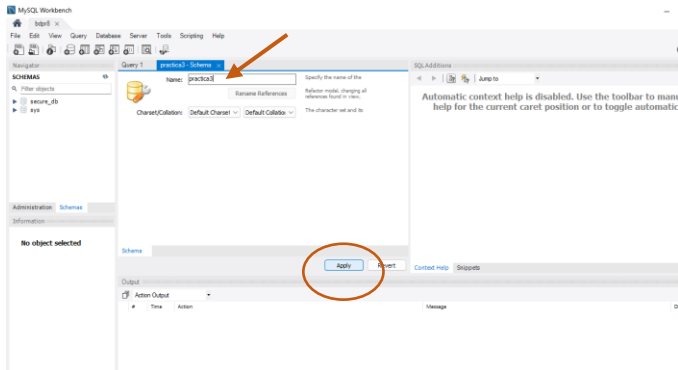


Ilustración 3

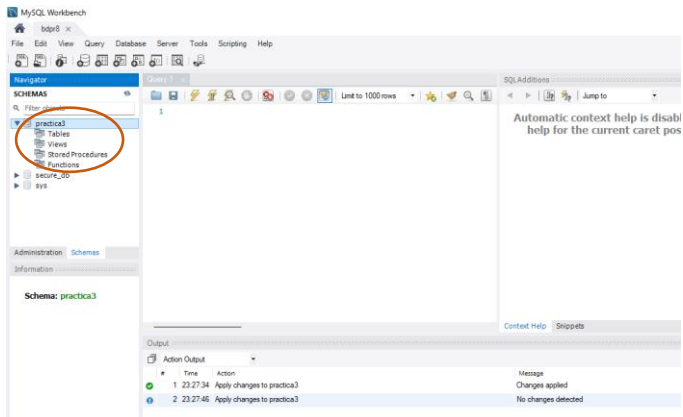


Ilustración 4

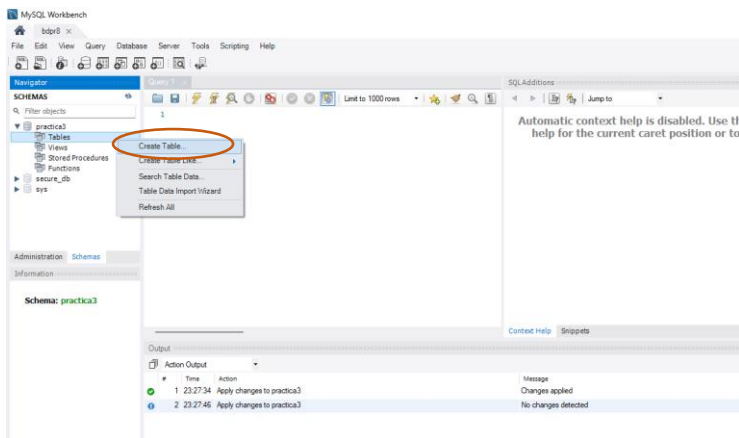


Ilustración 5

En esta parte se muestra la creación de las 3 tablas, cada una con sus columnas correspondientes.

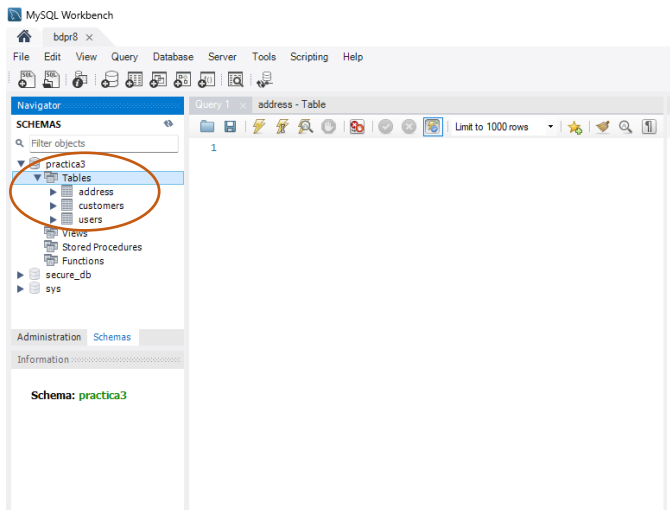


Ilustración 6

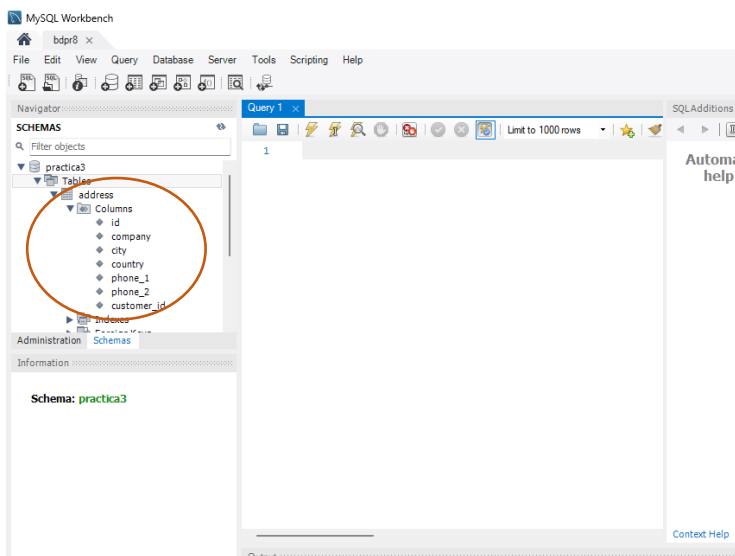


Ilustración 7

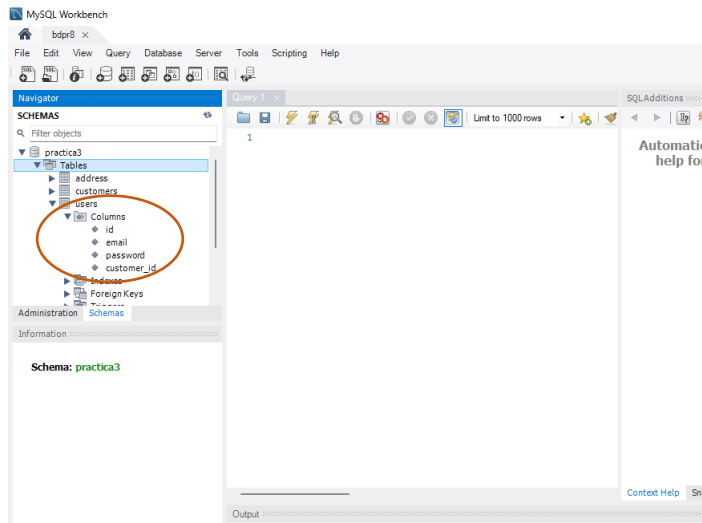


Ilustración 8

Luego creamos tres usuarios en MySQL con los siguientes permisos:

- Usuario 1: Permisos de lectura en la tabla `customers`
- Usuario 2: Permisos de lectura y escritura en la tabla `address`
- Usuario 3: Permisos de lectura, escritura y eliminación en la tabla `users`

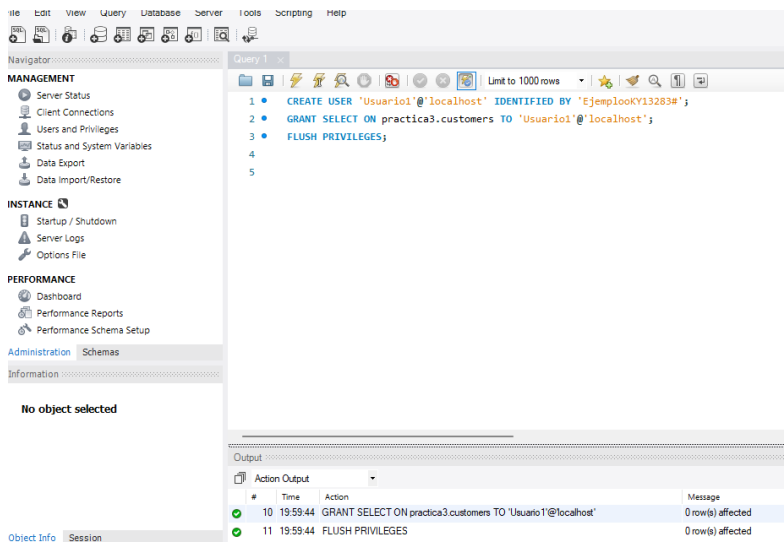
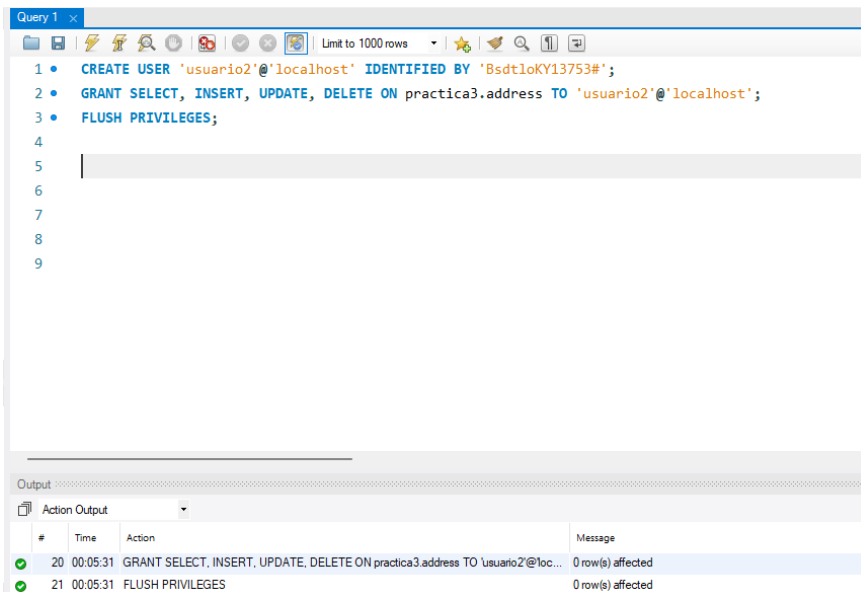


Ilustración 9

Primero hay que asignar el nombre para el usuario y una contraseña segura que desees asignar.

Con el comando concede los privilegios SELECT, que permiten al usuario leer la tabla.



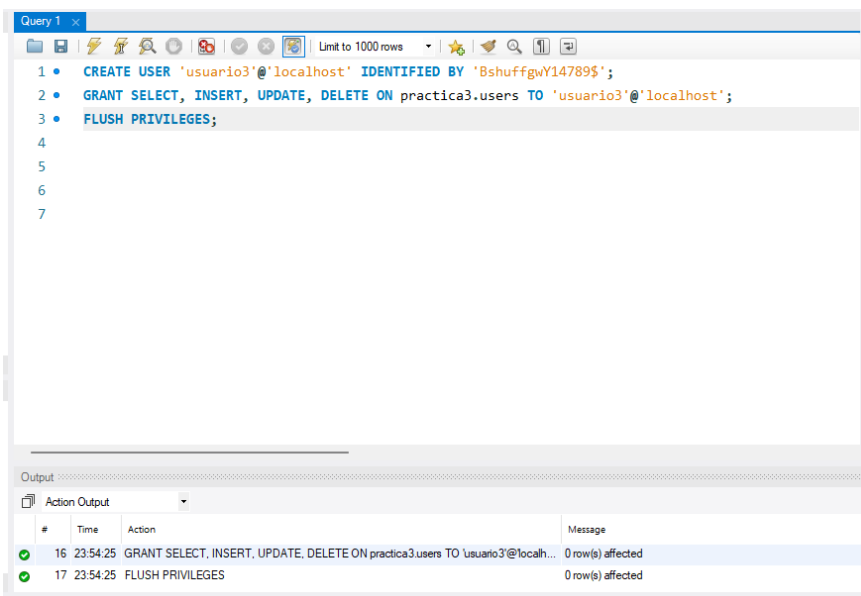
```
Query 1
1 • CREATE USER 'usuario2'@'localhost' IDENTIFIED BY 'BsdtdloKY13753#';
2 • GRANT SELECT, INSERT, UPDATE, DELETE ON practica3.address TO 'usuario2'@'localhost';
3 • FLUSH PRIVILEGES;
4
5
6
7
8
9
```

Output

#	Time	Action	Message
20	00:05:31	GRANT SELECT, INSERT, UPDATE, DELETE ON practica3.address TO 'usuario2'@'loc...	0 row(s) affected
21	00:05:31	FLUSH PRIVILEGES	0 row(s) affected

Ilustración 10

Con los comandos SELECT, INSERT, UPDATE concede los Permisos de lectura y escritura en la tabla `address`.



```
Query 1
1 • CREATE USER 'usuario3'@'localhost' IDENTIFIED BY 'BshuffgwY14789$';
2 • GRANT SELECT, INSERT, UPDATE, DELETE ON practica3.users TO 'usuario3'@'localhost';
3 • FLUSH PRIVILEGES;
4
5
6
7
```

Output

#	Time	Action	Message
16	23:54:25	GRANT SELECT, INSERT, UPDATE, DELETE ON practica3.users TO 'usuario3'@'localh...	0 row(s) affected
17	23:54:25	FLUSH PRIVILEGES	0 row(s) affected

Ilustración 11

Con los comandos SELECT, INSERT, UPDATE, DELETE ON se concede los permisos de lectura, escritura y eliminación en la tabla `users`.

Posteriormente creamos un backup de la base de datos `secure\_db` para restaurar la base de datos en un servidor diferente.

Para crear un backup de la base de datos `secure\_db` primero la seleccionamos y también seleccionamos donde queremos que se guarde.

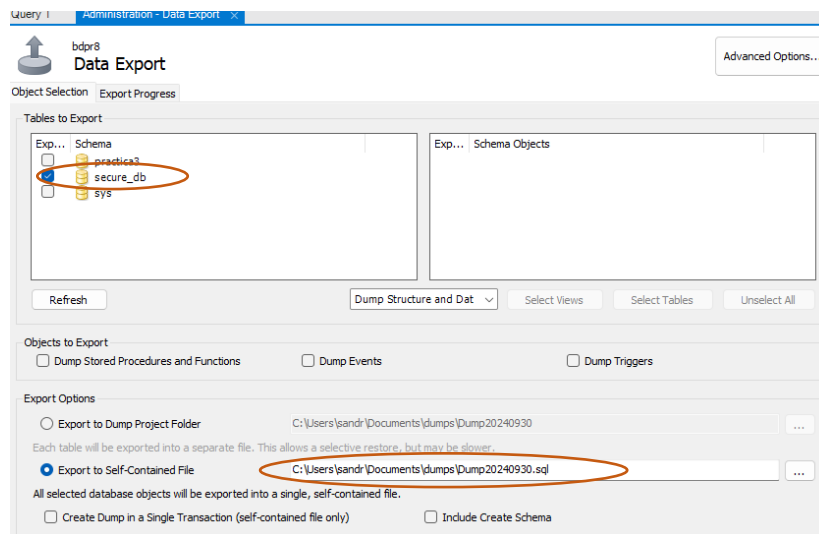


Ilustración 12

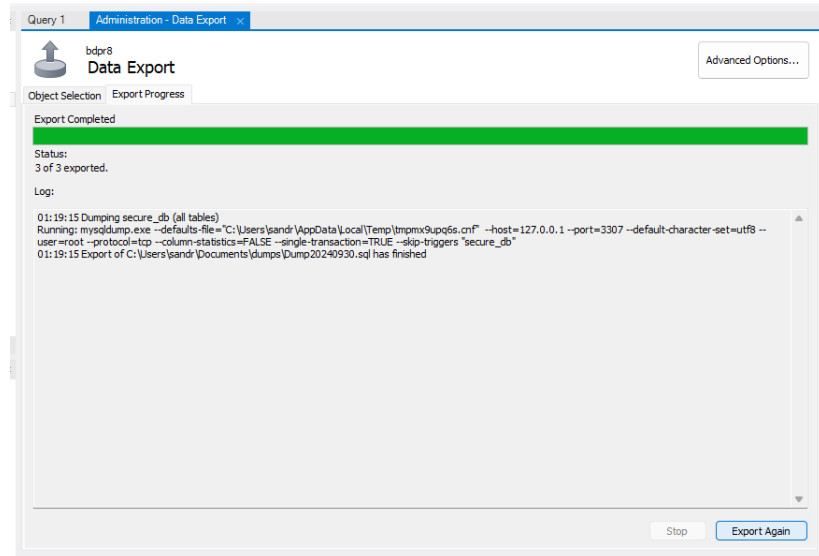


Ilustración 13



Ya que se creo exitosamente el backup, nos dirigimos a la ubicación que asignamos y se puede observar que ya se encuentra el backup.

Luego lo compartimos a otra integrante del equipo para que lo restaure en su servidor.

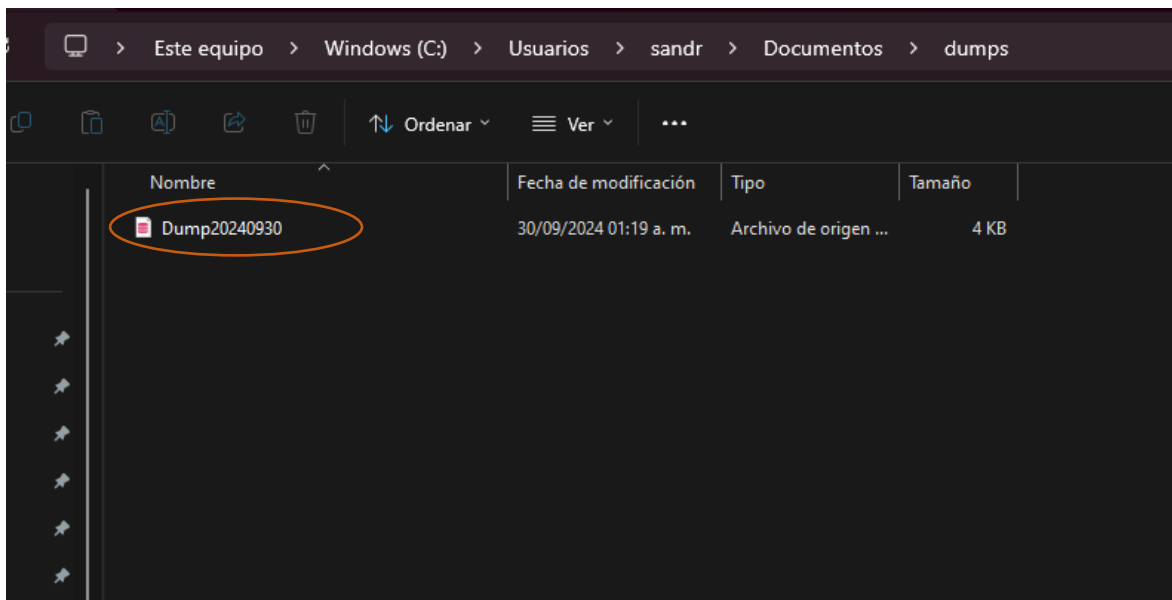


Ilustración 14

Primero se tuvo que crear una base de datos vacía para realizar el respaldo, en nuestro caso con el mismo nombre.

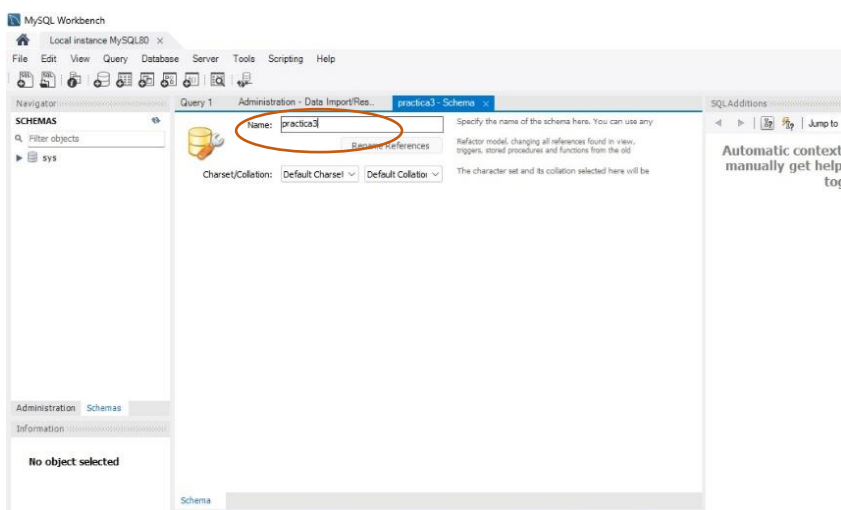


Ilustración 15

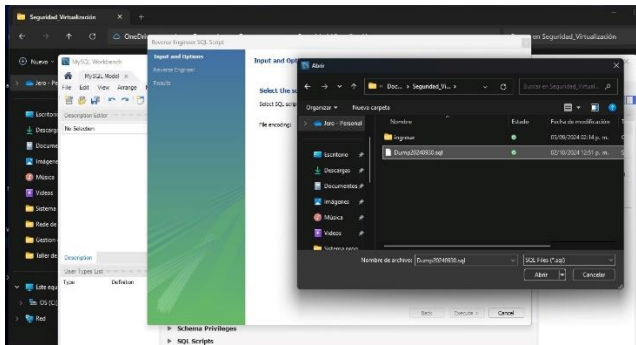


Ilustración 16

Luego importamos el archivo anterior, también seleccionamos en que base de datos queremos que se importe y seleccionamos la base de datos vacia creada que se llama practica3.

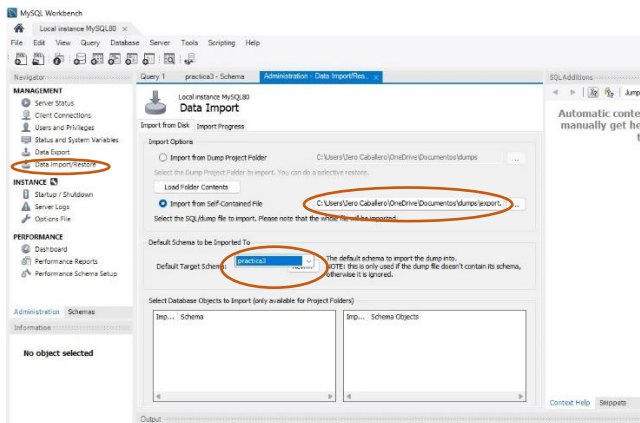


Ilustración 17

Finalmente se puede observar que se a importado y restaurado con éxito el backup a la base de datos.

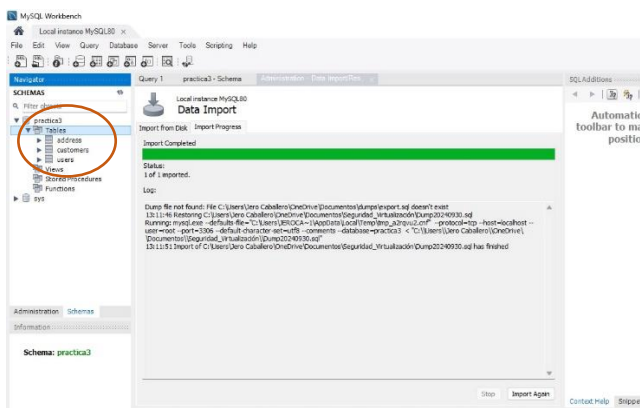


Ilustración 18

## INVESTIGA Y DESCRIBE LOS CONCEPTOS DE SQL INJECTION Y CÓMO SE PUEDEN PREVENIR

¿Qué es la inyección de SQLy cómo funciona?

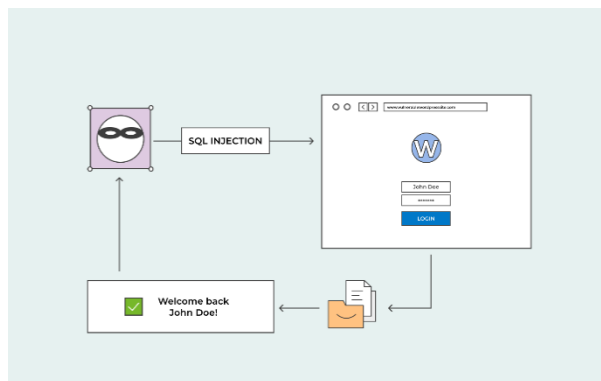


Ilustración 19

La inyección de SQL es un tipo de ciberataque encubierto en el cual un hacker inserta código propio en un sitio web con el fin de quebrantar las medidas de seguridad y acceder a datos protegidos. Una vez dentro, puede controlar la base de datos del sitio web y secuestrar la información de los usuarios.

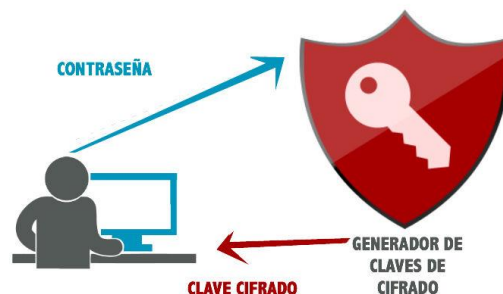
### Cómo prevenir la inyección de código SQL



Ilustración 20

Aunque la inyección de código SQL es una de las amenazas más frecuentes de API, puede evitarse eficazmente con las estrategias de prevención adecuadas. Algunos métodos útiles para evitar la inyección de código SQL incluyen restringir los procedimientos de la base de datos, sanear las entradas de la base de datos y aplicar el acceso con menos privilegios.

## Restringir los procedimientos y código de la base de datos



*Ilustración 21*

La inyección de código SQL depende en gran medida de la capacidad de un atacante para manipular las entradas de datos y funciones de la base de datos. Al restringir estas entradas y limitar el tipo de procedimientos de base de datos que se pueden realizar, las organizaciones pueden minimizar el riesgo de consultas no autorizadas o maliciosas. Las formas de hacerlo incluyen:

**Aplicación de declaraciones preparadas y consultas parametrizadas:** las declaraciones preparadas definen el código SQL aceptable, y luego establecen parámetros específicos para las consultas entrantes. Las declaraciones de SQL maliciosas se clasifican como entradas de datos no válidas y no como comandos ejecutables.

**Utilizar procedimientos almacenados:** al igual que las declaraciones preparadas, los procedimientos almacenados son declaraciones de SQL preparadas y reutilizables que pueden recuperarse de una base de datos y evitan que las partes maliciosas ejecuten código directamente en la base de datos.

### ¿Qué efecto tienen los ataques de inyección de SQL?

Los hackers recurren a los ataques de inyección de SQL con el fin de introducirse en la base de datos de un sitio web. A veces solo quieren eliminar datos para provocar el caos y, en otras ocasiones, lo que buscan es editar la base de datos, especialmente en el caso de sitios web financieros. En el momento en que el hacker ha logrado el control de la base de datos, ya es fácil interferir en los saldos de las cuentas de los clientes y mandarse dinero a su propia cuenta.

**SQL Injection** (Inyección SQL) es un tipo de ataque en el que un atacante manipula las consultas SQL que una aplicación envía a una base de datos, con el fin de ejecutar comandos maliciosos. El objetivo de este ataque es obtener acceso no autorizado a datos, manipular o eliminar información, o incluso tomar control del servidor de base de datos.

### Conceptos clave de SQL Injection:

1. **Vulnerabilidad en las consultas:** Las aplicaciones vulnerables permiten que los datos de entrada proporcionados por el usuario se incluyan directamente en las consultas SQL sin una validación o limpieza adecuada. Los atacantes pueden inyectar código malicioso en estos puntos de entrada para alterar la consulta prevista.
2. **Modificación de consultas:** Al enviar caracteres especiales o código SQL directamente en campos de entrada (como formularios, URL o cookies), un atacante puede modificar la estructura de una consulta SQL. Por ejemplo:
  - Entrada válida: usuario = 'Sandra'
  - Entrada maliciosa: usuario = 'Sandra' OR '1'='1' (esto convierte la consulta en verdadera siempre y da acceso a los datos).
3. **Ejecución de comandos no deseados:** Los atacantes pueden utilizar este tipo de ataques para realizar diferentes acciones maliciosas:
  - **Acceso no autorizado:** Iniciar sesión como un usuario sin conocer la contraseña.
  - **Exfiltración de datos:** Obtener datos sensibles como contraseñas, correos, información financiera, etc.
  - **Alteración de datos:** Modificar, insertar o eliminar datos en la base de datos.
  - **Control del servidor:** En algunos casos, es posible ejecutar comandos en el servidor de base de datos, lo que puede comprometer todo el sistema.

### Ejemplo de ataque SQL Injection:

Supongamos que una aplicación web tiene una consulta SQL mal construida:

```
SELECT * FROM usuarios WHERE nombre = 'usuario' AND password = 'password';
```

Un atacante puede ingresar usuario' OR '1'='1' como nombre y dejar el campo de contraseña en blanco.

Esto convierte la consulta en:

```
SELECT * FROM usuarios WHERE nombre = 'usuario' OR '1'='1' AND password = '';
```

La condición OR '1'='1' es siempre verdadera, lo que da acceso a la base de datos sin necesidad de autenticación.

### **Cómo prevenir SQL Injection:**

#### **1. Uso de consultas preparadas (Prepared Statements):**

También conocidas como consultas parametrizadas, separan la lógica de la consulta de los datos proporcionados por el usuario. Con este enfoque, los datos ingresados se tratan como texto, evitando que se interpreten como comandos SQL.

#### **2. Validación y saneamiento de entradas:** Siempre se deben validar y limpiar los datos ingresados por los usuarios para asegurarse de que solo contengan los caracteres permitidos y no tengan secuencias maliciosas.

- **Escapar caracteres especiales:** Asegurarse de que caracteres como ', ", -- (comentario), ; (fin de consulta) no sean malinterpretados por la base de datos.
- **Tipos de datos apropiados:** Validar que los campos solo acepten los tipos de datos que corresponden. Por ejemplo, en un campo numérico, verificar que la entrada sea un número válido.

#### **3. Uso de ORM (Object-Relational Mapping):** Utilizar frameworks ORM, como Hibernate o Entity Framework, abstrae las consultas SQL, lo que reduce la posibilidad de exposición a inyecciones SQL, ya que estos frameworks manejan las consultas de manera segura.

#### **4. Límite de permisos de base de datos:** Los usuarios de la base de datos deben tener solo los permisos mínimos necesarios para realizar las acciones requeridas. Si un atacante explota una inyección SQL, las consecuencias serán menores si los permisos están limitados.

#### **5. Gestión de errores adecuada:** No mostrar mensajes de error detallados al usuario final que puedan revelar la estructura de las consultas SQL o la base de datos. Estos mensajes pueden dar pistas al atacante sobre cómo estructurar su ataque.

6. **Cifrado de información sensible:** Cifrar datos críticos como contraseñas o información financiera para que, incluso si los atacantes obtienen acceso, los datos estén protegidos.
7. **Actualizaciones y parches:** Mantener el sistema de gestión de bases de datos (DBMS) y el software actualizado es crucial para corregir vulnerabilidades conocidas que puedan ser explotadas para realizar inyecciones SQL.

Implementando estas medidas, se pueden prevenir los ataques de SQL Injection y proteger las bases de datos de accesos no autorizados y manipulaciones maliciosas.

## INVESTIGA Y DESCRIBE LOS CONCEPTOS DE BASES DE DATOS SEGURAS Y CÓMO SE PUEDEN IMPLEMENTAR.

¿Qué es la seguridad de las bases de datos?



*Ilustración 22*

La seguridad de las bases de datos se refiere al conjunto de herramientas, medidas y controles diseñados para establecer y mantener la confidencialidad, la integridad y la disponibilidad de las bases de datos. Este artículo se va a centrar principalmente en la confidencialidad, ya que es el elemento que se ve comprometido en la mayoría de las infracciones de datos.

La seguridad de las bases de datos debe tratar y proteger lo siguiente:

- Los datos de la base de datos

- El sistema de gestión de bases de datos (DBMS)
- Cualquier aplicación asociada

El servidor de base de datos físico y/o el servidor de base de datos virtual, y el hardware subyacente

La infraestructura informática y/o de red utilizada para acceder a la base de datos

La seguridad de las bases de datos es una iniciativa compleja que implica todos los aspectos de las tecnologías y las prácticas de seguridad de la información. Además, se enfrenta a la usabilidad de la base de datos. Cuanto más accesible y utilizable sea la base de datos, más vulnerable será ante las amenazas de seguridad; cuanto más protegida esté la base de datos ante las amenazas, más difícil será acceder a ella y utilizarla. En ocasiones, esta paradoja se denomina regla de Anderson (enlace externo a IBM).

### ¿Por qué es importante?



Ilustración 23

Por definición, una infracción de datos es la incapacidad de mantener la confidencialidad de los datos en una base de datos. La cantidad de daño que las infracciones de datos infligen a su empresa depende de varios factores o consecuencias:

**Daño a la reputación de la marca:** los clientes o los socios pueden no estar dispuestos a comprar sus productos o servicios (o a hacer negocios con su empresa) si no sienten que pueden confiar en usted para proteger los datos.

**Multas o sanciones por falta de conformidad:** el impacto financiero por no cumplir con las normativas globales, como la Sarbannes-Oxley Act (SAO) o eñ Payment



Card Industry Data Security Standard (PCI DSS); las normativas de privacidad de datos específicas del sector, como la HIPAA, o las normativas regionales de privacidad de datos, como el Reglamento General de Protección de Datos (RGPD) de Europa, puede ser devastador, con sanciones superiores, en el peor de los casos, a varios millones de dólares por violación.

**Costes de reparación de infracciones y notificación a los clientes:** además del coste de comunicar una infracción al cliente, la organización que sufre la infracción debe abonar las actividades forenses y de investigación, de gestión de crisis, triaje, reparación de los sistemas afectados, etc.

## AMENAZAS Y DIFICULTADES HABITUALES



*Ilustración 24*

Son muchas las configuraciones erróneas de software, vulnerabilidades o patrones de descuido o mal uso que pueden dar lugar a una infracción. Los siguientes son los tipos o causas más habituales de los ataques de seguridad de base de datos y sus causas.

### Amenazas internas

Las amenazas internas son amenazas de seguridad de una de las tres fuentes que tienen acceso con privilegios a la base de datos:

Un usuario interno malicioso que tiene la intención de hacer daño.

Un usuario interno negligente que comete errores que provocan que la base de datos sea vulnerable a los ataques.

Un infiltrado un usuario externo que, de alguna manera, obtiene las credenciales a través de una estrategia de phishing u obtiene acceso a la propia base de datos de credenciales.

Una **base de datos segura** es aquella que implementa mecanismos y técnicas para proteger la confidencialidad, integridad y disponibilidad de los datos almacenados, evitando accesos no autorizados, manipulaciones indebidas o pérdida de información. Aquí están algunos conceptos clave y formas de implementación:

## 1. Confidencialidad

Se refiere a la protección de los datos para que solo personas autorizadas puedan acceder a ellos. La confidencialidad se puede implementar con las siguientes técnicas:

- **Cifrado de datos:** Tanto los datos almacenados como los datos en tránsito deben estar cifrados, de manera que, incluso si son interceptados, no puedan ser leídos sin la clave adecuada.
- **Control de acceso:** Se establecen políticas para determinar quién puede acceder a qué información. Por ejemplo, mediante la gestión de permisos por roles o listas de control de acceso (ACL).

## 2. Integridad

La integridad garantiza que los datos no sean modificados o manipulados de manera no autorizada. Se puede asegurar mediante:

- **Firmas digitales:** Ayudan a verificar que los datos no han sido alterados desde que fueron generados.
- **Checksums o Hashing:** El uso de funciones hash permite verificar que los datos no han sido modificados. Los valores hash pueden compararse para detectar cambios indebidos en los datos.

## 3. Disponibilidad

La disponibilidad asegura que los datos estén accesibles cuando los usuarios autorizados lo requieran. Esto puede lograrse mediante:

- **Backups y recuperación ante desastres:** Copias de seguridad periódicas y planes de recuperación en caso de fallos del sistema.
- **Redundancia:** Sistemas distribuidos y redundantes que eviten puntos únicos de fallo, garantizando que la base de datos siga disponible incluso si un componente falla.

#### 4. Auditoría y monitoreo

El monitoreo y auditoría de los accesos y modificaciones a la base de datos permiten detectar actividades inusuales o sospechosas. Los logs de auditoría proporcionan un registro de todas las acciones, facilitando la investigación en caso de un incidente de seguridad.

#### 5. Autenticación

Asegura que solo usuarios legítimos puedan acceder a la base de datos. Los mecanismos de autenticación incluyen:

- **Autenticación multifactor (MFA):** Requiere múltiples formas de verificación (contraseña, token, biometría) para aumentar la seguridad.
- **Autenticación basada en certificados:** Verifica la identidad de los usuarios o sistemas mediante certificados digitales.

#### 6. Actualizaciones de software

Mantener el sistema de gestión de bases de datos (DBMS) y sus componentes actualizados es crucial para cerrar vulnerabilidades conocidas. Las actualizaciones y parches de seguridad deben aplicarse regularmente para proteger contra nuevas amenazas.

#### 7. Segmentación de red y seguridad perimetral

Limitar el acceso a la base de datos desde la red mediante firewalls, VPNs y políticas de segmentación de red ayuda a minimizar los riesgos de ataques externos. Además, solo los servicios esenciales deberían tener acceso a la base de datos.

#### 8. Manejo de privilegios mínimos

La aplicación de este principio significa otorgar a cada usuario solo los permisos mínimos necesarios para realizar su trabajo. Esto reduce el riesgo de abuso o error accidental que pueda comprometer la seguridad.

#### Implementación de bases de datos seguras

- **Cifrado completo (Full Disk Encryption o FDE):** Cifra toda la base de datos, asegurando que los datos no se puedan leer sin la clave.
- **Políticas de gestión de contraseñas:** Implica el uso de contraseñas fuertes y periódicamente renovadas para usuarios y servicios.
- **Bases de datos aisladas:** Aislar entornos de desarrollo, prueba y producción para que los datos sensibles no estén expuestos innecesariamente.

- **Políticas de expiración de sesiones:** Para minimizar los riesgos de sesiones de usuarios no cerradas correctamente.

Al implementar estos conceptos, se puede garantizar que las bases de datos sean más resistentes ante ataques y accesos indebidos, manteniendo su seguridad y funcionalidad.

## REFERENCIAS

<https://www.avast.com/es-es/c-sql-injection#:~:text=La%20inyecci%C3%B3n%20de%20SQL%20es,la%20informaci%C3%B3n%20de%20los%20usuarios.>

<https://www.ibm.com/es-es/topics/database-security#:~:text=%C2%BFQu%C3%A9%20es%20la%20seguridad%20de,de%20las%20bases%20de%20datos.>

<https://www.one.com/es/seguridad-de-su-web/que-es-sql-injection#:~:text=SQL%20injection%20es%20un%20tipo,que%20la%20expone%20a%20ataques.>