Gestión de la seguridad de los datos.

Índice

1 Gestión de la seguridad de los datos

1.1 Tipos de copias de seguridad

Podemos establecer diferentes clasificaciones:

1. Frío - Caliente
2. Física - Lógica
3. Completa - Incremental
4. Manual - Automática

1.1.1 Frío - Caliente

- **En frío:** Este tipo de copias de seguridad se realizan parando el servicio de base de datos, para evitar que los usuarios puedan acceder a los datos mientras se realiza la copia. Tiene el inconveniente de que el servicio no estará disponible durante el tiempo que dure el proceso de copia.
- **En caliente:** En este caso no es necesario detener el servicio de base de datos, de modo que los usuarios pueden acceder a la base de datos mientras se realiza la copia de seguridad.

1.1.2 Física o Binaria - Lógica o Textual

- **Física o Binaria:** Este tipo de copias consiste en copiar desde el disco los archivos binarios que utiliza el sistema gestor de bases de datos para almacenar las bases de datos. Este tipo de copias tienen la ventaja de que son muy rápidas y se pueden realizar tanto en frío como en caliente.

  La restauración de estas copias de seguridad consiste en copiar los archivos a sus ubicaciones originales. Para realizar copias físicas podemos hacer uso de comandos de copia de archivos (`cp`, `scp`, `tar`, `rsync`), `mysqlhotcopy` o `InnoDB Hot Backup`.
- **Lógica o Textual:** Una copia lógica consiste en exportar los datos y los diferentes objetos de las bases de datos en archivos de texto. Este tipo de copias tienen el inconveniente de que son más lentas, pero tienen la ventaja de que nos permite importarlas en otros sistemas gestores de bases de datos.

  La restauración de estas copias de seguridad consiste en volver a cargar el contenido de los archivos de texto en las bases de datos del servidor. Las técnicas que podemos utilizar para realizar este tipo de copias son la sentencia de SQL `SELECT ... INTO OUTFILE`, `mysqldump`, `MySQL Workbench` o `phpMyAdmin`.

1.1.3 Completa - Incremental

- **Completa:** Una copia completa es una copia de todos los datos de la base de datos (o de varias bases de datos).
- **Incremental:** Es una copia que contiene sólo los aquellos datos que han cambiado respecto a la última copia.

1.1.4 Manual - Automática

- **Manual:** Son las copias que ejecutamos nosotros de forma manual.
- **Automática:** Lo normal es que el proceso de copias de seguridad esté automatizado y que las copias de seguridad se realicen de forma automática.

1.2 ¿Dónde se guardan los datos de una base de datos MySQL?

Los datos de las bases de datos de MySQL se almacenan en el directorio:

```
/var/lib/mysql
```

Dentro de este directorio se crea **un directorio para cada una de las bases de datos**.

Por ejemplo, si en hemos creado una base de datos llamada `empleados` estará almacenada en:

```
/var/lib/mysql/empleados
```

El contenido de este directorio podría ser el siguiente:

```
.
|-- departamento.frm
|-- departamento.ibd
|-- empleado.frm
`-- empleado.ibd
```

1.3 ¿Qué almacena cada uno de los archivos?

Los archivos que utiliza MySQL para almacenar las bases de datos pueden tener las siguientes extensiones:

- `.frm`
- `.ibd`
- `.myd`
- `.myi`

1.3.1 `.frm`

Los archivos con extensión `.frm` contienen la estructura de las tablas. Hay tantos archivos `.frm` como tablas tenga nuestra base de datos. Estos archivos se usan para las **tablas de tipo InnoDB y MyISAM**.

1.3.2 `.ibd`

Los archivos con extensión `.ibd` almacenan los datos y los índices de las **tablas de tipo InnoDB**. Hay tantos archivos `.ibd` como tablas tenga nuestra base de datos.

1.3.3 `.myd`

Los archivos con extensión `.myd` almacenan los datos de las **tablas de tipo MyISAM**. Hay tantos archivos `.myd` como tablas tenga nuestra base de datos.

1.3.4 `.myi`

Los archivos con extensión `.myi` almacenan los índices de las **tablas de tipo MyISAM**. Hay tantos archivos `.myi` como tablas tenga nuestra base de datos.

1.4 Copias de seguridad físicas o binarias

1.4.1 Cómo realizar y restaurar una copia binaria `MyISAM`

Consultar la sección: **32.3.1 Making Binary MyISAM Backups** del documento MySQL 5.0 Certification Study Guide.

1.4.2 Cómo realizar y restaurar una copia binaria `InnoDB`

Consultar la sección: **32.3.2 Making Binary InnoDB Backups** del documento MySQL 5.0 Certification Study Guide.

```
1  -- Paso 1
2  USE tienda;
3
4  -- Paso 2
5  SHOW TABLES;
6
7  -- Paso 3
8  SELECT *
9  FROM producto;
10
```

```
11 -- This forces the server to close the table and provides your connection with
       a read lock on the table.
12 FLUSH TABLES producto FOR EXPORT;
13
14 -- Paso 4
15 -- Hacemos una copia del archivo producto.ibd
16
17 -- Then, once you've copied the files, you can release the lock with UNLOCK
       TABLES:
18 UNLOCK TABLES;
19
20 -- Paso 5
21 DELETE
22 FROM producto;
23
24 -- Paso 6
25 SELECT *
26 FROM producto;
27
28 -- Paso 7
29 ALTER TABLE tienda.producto DISCARD TABLESPACE;
30
31 -- Paso 8
32 -- Restauramos el archivo producto.idb
33 -- Recuerda que en Linux el propietario y el grupo del archivo tiene que ser
       mysql.
34
35 -- Paso 9
36 ALTER TABLE tienda.producto IMPORT TABLESPACE;
37
38 -- Paso 10
39 SELECT *
40 FROM producto;
```

Referencias:

- InnoDB File-Per-Table Tablespaces.
- Importing InnoDB Tables.
- InnoDB Backup.

1.5 Copias de seguridad lógicas o textuales

1.5.1 `SELECT ... INTO OUTFILE`

La sentencia `SELECT ... INTO OUTFILE` nos permite realizar copias de seguridad lógicas o textuales, exportando los datos en diferentes formatos de texto.

Ejemplo

```
1 -- Paso 1. Seleccionamos una base de datos
2 USE tienda;
3
```

```
-- Paso 2. Mostramos las tablas de la base de datos
SHOW TABLES;

-- Paso 3. Intentamos exportar todos los datos de la tabla fabricante
SELECT *
INTO OUTFILE 'fabricante.txt'
FROM fabricante;

-- Paso 4. Obtendremos el siguiente mensaje de error
-- Error Code: 1290. The MySQL server is running with the --secure-file-priv
    option so it cannot execute this statement

-- Paso 5. Consultamos la variable secure_file_priv para obtener la ruta donde
    se guardarán los archivos que generamos con SELECT .. INT OUTFILE
SHOW VARIABLES LIKE 'secure_file_priv';

-- Paso 6. Añadimos la ruta al nombre del archivo
SELECT *
INTO OUTFILE '/var/lib/mysql-files/fabricante.txt'
FROM fabricante;

-- Paso 7. Añadimos algunos modificadores para mejorar la salida
SELECT *
INTO OUTFILE '/var/lib/mysql-files/fabricante.txt'
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
FROM fabricante;

-- Paso 8.
-- ?¿Qué error obtenemos en el paso anterior?
-- ?¿Es posible reescribir los archivos con SELECT .. INTO OUTFILE?
-- ?¿Cómo podemos resolverlo?
```

Puede encontrar más información en la documentación oficial.

1.5.2 LOAD DATA

La sentencia LOAD DATA nos permite restaurar copias de seguridad lógicas o textuales, importando los datos desde un archivo de texto a nuestra base de datos.

```
LOAD DATA
    [LOW_PRIORITY | CONCURRENT] [LOCAL]
    INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
```

```
        [TERMINATED BY 'string']
    ]
    [IGNORE number {LINES | ROWS}]
    [(col_name_or_user_var
        [, col_name_or_user_var] ...)]
    [SET col_name={expr | DEFAULT}
        [, col_name={expr | DEFAULT}] ...]
```

Puede encontrar más información en la documentación oficial.

Ejemplo:

```
-- Paso 1. Seleccionamos la base de datos tienda
USE tienda;

-- Paso 2. Creamos una tabla nueva llamada fabricante_backup
CREATE TABLE fabricante_backup (
  codigo INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL
);

-- Paso 3. Importamos los datos del archivo /var/lib/mysql-files/fabricante.txt
    en la nueva tabla
LOAD DATA INFILE '/var/lib/mysql-files/fabricante.txt'
INTO TABLE fabricante_backup
FIELDS TERMINATED BY ',';

-- Paso 4. Comprobamos que los datos se han insertado correctamente
SELECT *
FROM fabricante_backup;
```

1.5.3 `mysqldump`

La utilidad `mysqldump` permite realizar copias de seguridad lógicas o textuales de una base de datos MySQL.

Existen tres formas de usar `mysqldump`:

1) Exportar una o varias tablas de una base de datos,

```
mysqldump [options] db_name [tbl_name ...]
```

2) Exportar una o varias bases de datos completas,

```
mysqldump [options] --databases db_name ...
```

3) Exportar todas las bases de datos completas.

```
mysqldump [options] --all-databases
```

Referencia:

- Documentación en la página oficial.

1.5.3.1 Exportar una o varias tablas de una base de datos

Para exportar una o varias tablas de una base de datos podemos usar este comando:

```
mysqldump -u [username] -p [database_name] [tbl_name ...] > [backup_name].sql
```

Ejemplo:

```
mysqldump -u root -p wordpress > backup.sql
```

En este ejemplo estamos exportando todas las tablas de la base de datos `wordpress` y estamos guardando la salida con las sentencias `SQL` en un archivo llamado `backup.sql`.

Nota importante: En este caso **no se incluye** la sentencia `CREATE DATABASE` en el archivo de *backup*. Sólo se generan sentencias de tipo `CREATE TABLE` y `INSERT`.

1.5.3.2 Exportar una o varias bases de datos completas

```
mysqldump -u [username] -p --databases db_name [...] > [backup_name].sql
```

Ejemplo:

```
mysqldump -u root -p --databases wordpress mediawiki > backup.sql
```

En este ejemplo estamos exportando dos bases de datos completas llamadas `wordpress` y `mediawiki`, y estamos guardando la salida con las sentencias `SQL` en un archivo llamado `backup.sql`.

Nota importante: En este caso **sí se incluye** la sentencia `CREATE DATABASE` en el archivo de *backup*.

1.5.3.3 Exportar todas las bases de datos completas

```
mysqldump -u [username] -p --all-databases > [backup_name].sql
```

Ejemplo:

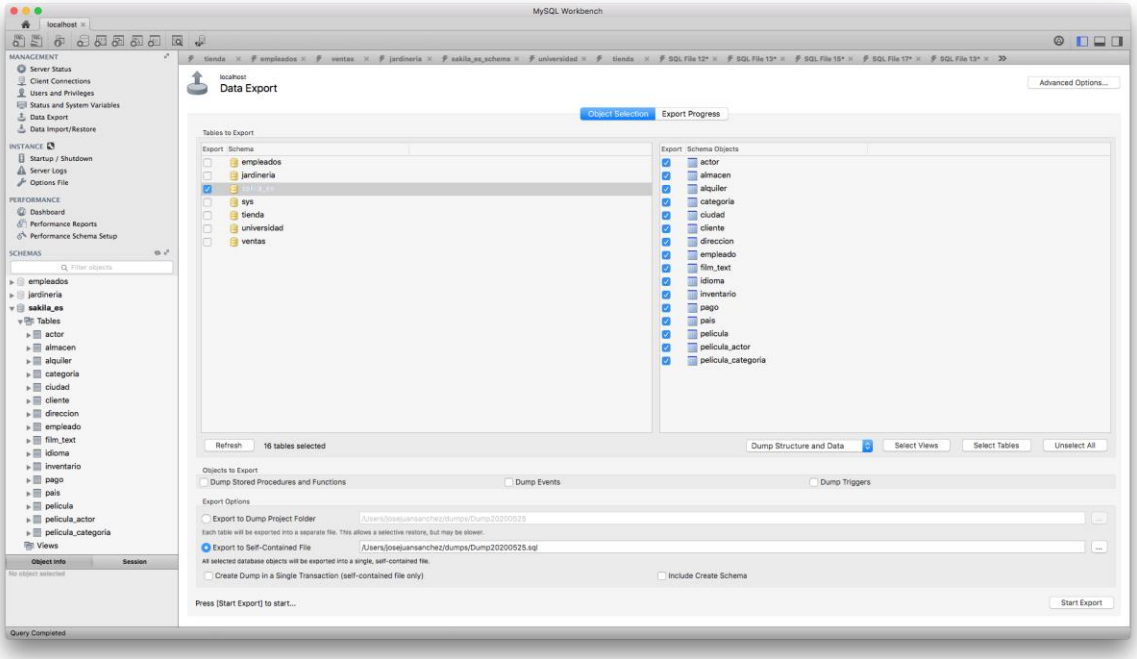
```
mysqldump -u root -p --all-databases > backup.sql
```

En este ejemplo estamos exportando todas las bases completas que existen en el MySQL Server al que nos estamos conectando. La salida con las sentencias `SQL` se guarda en un archivo llamado `backup.sql`.

Nota importante: En este caso **sí se incluye** la sentencia `CREATE DATABASE` en el archivo de *backup*.

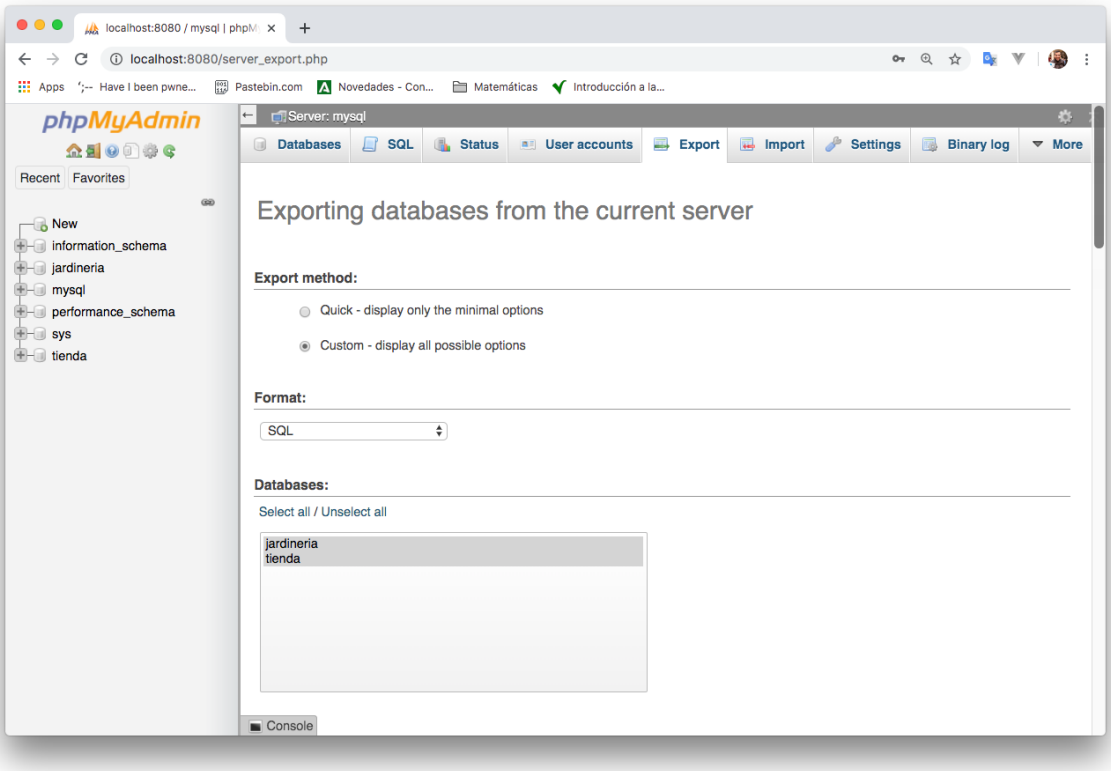
1.5.4 Realizar copias de seguridad con MySQL Workbench

También es posible realizar copias de seguridad haciendo uso de herramientas gráficas como MySQL Workbench.



1.5.5 Realizar copias de seguridad con phpMyAdmin

También es posible realizar copias de seguridad haciendo uso de herramientas gráficas como phpMyAdmin.



1.6 Restaurar una copia de seguridad lógica o textual

Para restaurar una copia de seguridad lógica o textual existen varias opciones. Podemos hacerlo **haciendo uso de aplicaciones gráficas** como MySQL Workbench o phpMyAdmin, o **desde línea de comandos**.

Para restaurar una copia de seguridad lógica o textual desde la línea de comandos tenemos las siguientes posibilidades:

- `mysql < backup.sql`
- `source backup.sql`
- `cat backup.sql | mysql`
- `copy/paste`

1.6.1 `mysql < backup.sql`

Con el cliente de línea de comandos de MySQL (`mysql`) podemos hacer uso del operador de redirección de entrada < para ejecutar sentencias SQL desde un script e importar una copia de seguridad lógica o textual.

Dependiendo de la opción que hayamos elegido para generar el *backup*, será necesario indicar previamente el nombre de la base de datos donde vamos a restaurar la copia.

Recuerda que la sentencia `CREATE DATABASE` sólo se incluye en el *backup* cuando usamos las opciones `--databases` y `--all-databases`. En estos casos podemos restaurar el *backup* con el siguiente comando:

```
mysql -u [username] -p < [backup_name].sql
```

Ejemplo:

```
mysql -u root -p < backup.sql
```

En este caso nos estamos conectando con el usuario `root` y estamos restaurando todas las sentencias `SQL` que están incluidas en el archivo `backup.sql`.

Si hemos realizado el *backup* sin usar las opciones `--databases` y `--all-databases` entonces la base de datos sobre la que vamos a restaurar los datos debe existir. Si no estuviese creada la podemos crear con la siguiente sentencia `SQL`:

```
CREATE DATABASE db_name CHARACTER SET ut8mb4;
```

Una vez que tengamos creada la base de datos podemos restaurar el *backup* con el siguiente comando:

```
mysql -u [username] -p [db_name] < [backup_name].sql
```

Ejemplo:

```
mysql -u root -p wordpress < backup.sql
```

1.6.2 `source backup.sql`

Desde el cliente de línea de comandos de MySQL (`mysql`) podemos hacer uso del comando `source` para ejecutar sentencias SQL e importar una copia de seguridad lógica o textual.

Ejemplo:

```
# Desde la línea de comandos del sistema operativo
mysqldump -u root -p --databases tienda > tienda.sql
mysql -u root -p

# Desde la línea de comandos de MySQL
source tienda.sql
```

También es posible utilizar el comando `\.` que es equivalente al comando `source`.

Ejemplo:

```
# Desde la línea de comandos del sistema operativo
mysqldump -u root -p --databases tienda > tienda.sql
mysql -u root -p

# Desde la línea de comandos de MySQL
\. tienda.sql
```