Profesora: Olga Cuervo Miguélez

TEMA 6

ADMINISTRACIÓN DE BASES DE DATOS. COPIAS DE SEGURIDAD

Módulo: **Bases de Datos**

Ciclo: **DAM**

6.1. Copias de seguridad

Definición

De forma genérica, las copias de seguridad, copias de respaldo o backups (por su nombre en inglés) son una medida preventiva en seguridad informática que consiste en realizar copias de los datos originales, o incluso programas, para tener una forma de recuperarlos en caso de pérdida en cualesquiera que sean las circunstancias que lo provoquen. En el caso específico de las bases de datos, las copias de seguridad se aplican a los datos almacenados en ellas.

Las copias de seguridad deben asegurar dos características de la información:

- ✓ **Integridad**: se refiere a la calidad de la información, ya que debe ser exacta y completa y no sufrir manipulación o alteración alguna respecto del original sin la debida autorización.
- ✓ **Disponibilidad**: la información debe ser accesible en todo momento para cualquier usuario que tenga permisos para acceder a ella.

Planificación de las copias de seguridad

Para estar preparados ante cualquier desastre que elimine o dañe los datos almacenados en una base de datos, se debe planificar una política de realizar copias de seguridad periódicas. Esta adecuada planificación de copias de seguridad es parte del plan de contingencia de una empresa, ya que la pérdida de datos puede poner en peligro la continuidad del negocio.
Algunos de los requisitos que debe cumplir la planificación de copias de seguridad son:

- ✓ **Identificar las bases de datos**, o datos dentro de ellas, que requieren preservación.
- ✓ **Establecer la frecuencia** con la que se realizarán los procesos de copia, así como el tipo de copia. Esta frecuencia influye en la cantidad de información que se puede perder con respecto a la fuente original. Este parámetro es extremadamente importante y requiere un análisis exhaustivo.

Por ejemplo, supongamos una empresa en la que se realizan copias de seguridad todas las noches a las 3:00 a.m. Si la base de datos falla a las 12:00 am, toda la información generada desde la noche anterior hasta esa hora no se guardará en la copia de seguridad.

- ✓ **Establecer esquema de rotación**: La rotación se refiere a cómo se almacenan y protegen las copias de seguridad. Un esquema de rotación indica cuántas cintas (u otros tipos de medios de almacenamiento) se utilizan para realizar la copia de seguridad.

Se puede encontrar información adicional sobre los esquemas de rotación de copias de seguridad en el siguiente enlace.

Cómo optimizar su estrategia de rotación de cintas de respaldo | Computer Weekly

- ✓ **Disponer del almacén físico para las copias**. Este almacén se determina en función de la seguridad que requiere la información, entre almacenes en un mismo edificio o remotos en edificios externos. Por ejemplo, cuando ocurre un incendio en el edificio de la empresa, la información almacenada en un edificio externo aún está disponible.
- ✓ **Buscar una mínima probabilidad de error**, asegurándose de que los datos se copian en soportes fiables y en buen estado. No se deben utilizar soportes próximos a su vida útil, para evitar que fallen a la hora de recuperar la información que contienen.
- ✓ **Controlar los soportes que contienen las copias**, guardándolas en lugar seguro y restringiendo su acceso únicamente a las personas autorizadas.
- ✓ **Planificar la restauración de los ejemplares:**
    - Formación del personal técnico encargado de su realización.

- Disponer de soportes para restaurar la copia, distintos a los de producción.
- Establecer los medios para disponer de dicha copia en el menor tiempo posible.

✓ **Probar el sistema de forma exhaustiva,** para comprobar su correcta planificación y la eficacia de los medios disponibles.

✓ **Definir la vigencia de las copias**, estableciendo un plazo en el cual dejan de ser válidas y podrán ser sustituidos por otros más actuales.

✓ **Controlar la obsolescencia de los dispositivos de almacenamiento**. Para el caso de las copias que se almacenen informaicon historica de la organización, por ejemplo proyectos ya cerrados, se debe tener en cuenta el tipo de dispositivo en el que se realiza la copia, para evitar que cuando se requiera la restauración de dicha información, dejen de existir lectores apropiados para el dispositivo.

Cuandose retiran los medios de almacenamiento, porque llega a su límite vida útil establecida en la política de respaldo, es importante realizar un proceso de eliminación o destrucción segura, para garantizar que la información que contienen no pueda recuperarse más tarde.

✓ **Especificar sistemas de respaldo**, especialmente cuando se requiera copias de seguridad en sistemas que no se pueden detener para realizar la copia en frío, por lo que deberá indicar la estrategia a utilizar para realizar la copia en caliente.

Tipos de copias de seguridad

A grandes rasgos, existen tres tipos de copias de seguridad: completas, diferenciales e incrementales.

✓ **Copia completa**. En este tipo de copia, se crea una copia de todos los datos existentes. A la primera vez que se realiza una copia sobre la información suele ser de este tipo.

✓ **Copia diferencial**. La copia de seguridad diferencial contendrá todos los datos que se crearon o modificaron desde la última copia de seguridad completa. Para restaurar datos se requiere la última copia completa y la última copia diferencial.

✓ **Copia incremental**. En este caso solo una copia de los datos que se crearon o modificaron desde la última copia completa o diferencial realizada. Para restaurar los datos, se requiere la última copia completa y todas las copias incrementales realizadas desde entonces.

Independientemente de estas definiciones genéricas, cada SGBD tiene su forma particular y sus propias herramientas para proporcionar al administrador la flexibilidad suficiente para tiempo para realizar copias de seguridad.

Copias de seguridad con MySQL

mysqldump

MySQL dispone de una utilidad llamada **mysqldump**, que se instala con el propio MySQL y que permite crear un script con el esquema y los datos de la base de datos. Este script se puede usar como copia de seguridad, para mover bases de datos de un servidor a otro, o para crear una base de datos de prueba basada en una existente. La sintaxis es la siguiente:

```
mysqldump [opciones] nombre_base_datos [nombre_tabla]
```

Algunas opciones que se pueden utilizar son:

`--help`. Mostra unha mensaxe de axuda con todas as opcións (`-?`).

`--add-locks`. Añade LOCK TABLES, y UNLOCK TABLE para copiar cada tabla.

`--databases`. Permite indicar el nombre de varias bases de datos `(-B)`.

`--all-databases`. Volcar el contenido de todas las bases de datos del servidor `(-A)`.

--flush-logs. Vuelca los ficheros de registro de log antes de comenzar el volcado (`-F`).

--lock-tables. Bloquea todas las tablas antes de comezar el volcado (`-l`).

--no-create-db. No incluye la sentencia CREATE DATABASE (`-n`).

--no-create-info. Copia solamente los datos. No incluye las sentencias de creación (base de datos ni tablas).

--non-data. No incluirá ninguna información sobre las filas de la tablas. Esta opción sirve para crear una copia del esquema de la base de datos.

--opt. Hace una operación de volcado rápida. Es lo mismo que especificar: `-- adddrop-table --add-locks -?create-` **options --disable-keys --extended-insert --lock-tables --quick --set-charset**.

--quick. Acelera el proceso en tablas grandes ya que no carga en memoria los resultados, sino que recupera y grava fila a fila `(-q)`.

--result-file= Guarda la salida en el fichero indicado (`-r fichero`).

Para tablas transaccionales (InnoDB, BDB):

--single-transaction. Vuelca las tablas transaccionales en un estado consistente, sin bloquear ninguna aplicación. Es incompatible con **--lock-tables**. Para tablas grandes es recomendable combinar con **--quick**.

Nota: Estas opciones se pueden gravar en un fichero de configuración, dentro de la sección [`mysqldump`].

Se verán a continuación algunos ejemplos de uso:

Copia de seguridad básica

El siguiente comando realiza una copia de seguridad completa de una base de datos llamada NOMBRE_BASE_DE_DATOS utilizando el usuario de nombre USUARIO con la contraseña CONTRASINAL:

```
$ mysqldump --user=USUARIO --password=CONTRASINAL NOMBRE_BASE_DE_DATOS > copia_seguridad.sql
```

Si por ejemplo el usuario es *root*, la contraseña también es *root* y la base de datos se llama *acme*, el comando que se debe ejecutar es el siguiente:

```
$ mysqldump --user=root --password=root acme > copia_seguridad.sql
```

Si por motivos de seguridad no se quiere escribir la contraseña como parte del comando, se puede cambiar la opción `--password=XX` por `-p`. Al hacerlo, MySQL pedirá que se escriba la contraseña a mano cada vez que se realice una copia de seguridad:

```
$ mysqldump --user=root -p acme > copia_seguridad.sql
Enter password: *********
```

https://uniwebsidad.com/tutoriales/copias-de-seguridad-avanzadas-para-bases-de-datos-mysql?from=librosweb
https://uniwebsidad.com/tutoriales/como-hacer-copias-de-seguridad-de-una-base-de-datos-mysql?from=librosweb

Recuperando una copia de seguridad

Suponiendo que los datos a recuperar están en el archivo *copia_seguridad.sql*, el comando que se debe ejecutar para recuperar la información de la base de datos es el siguiente:

```
$ mysql --user=USUARIO --password=CONTRASEÑA NOMBRE_BASE DE DATOS < copia_seguridad.sql
```

En este caso se ejecuta el comando **mysql** y no el comando **mysqldump**. Utilizando los mismos datos que en el ejemplo anterior, el comando a ejecutar sería:

```
$ mysql --user=root --password=root NOMBRE_BASE DE DATOS < copia_seguridad.sql
```

En este comando no es necesario indicar el nombre de la base de datos que se está recuperando, porque los archivos generados por **mysqldump** ya contienen esa información. De hecho, al ejecutar este comando de recuperación elimina la base de datos original y toda la información de sus tablas, para después insertar toda la información contenida en el archivo *copia_seguridad.sql*.

Si la copia de seguridad se realiza con una versión moderna de MySQL y la recuperación desde el la información se realiza con una versión un poco antigua, es mejor agregar la opción **--skip-opt** al realizar la copia de seguridad, para deshabilitar algunas opciones modernas e incompatibles:

```
$ mysql --user=USUARIO --password=CONTRASEÑA --skip-opt NOMBRE_BASE DE DATOS < copia_seguridad.sql
```

Copias de seguridad de más de una base de datos

Normalmente, el comando **mysqldump** se usa para hacer una copia de seguridad de un única base de datos. Sin embargo, a veces es necesario copiar varias bases de datos. En algunas ocasiones es necesario copiar varias bases de datos. Para esto, se usa la opción **--databases** y se especifican los nombres de todas las bases de datos separadas por un espacio en blanco:

```
$ mysqldump --user=USUARIO --password=CONTRASEÑA --databases NOMBRE_BASE_DATOS_1 NOMBRE_BASE_DATOS_2 NOMBRE_BASE_DATOS_3 > copia_seguridad.sql
```

Si desea hacer una copia de seguridad de todas las bases de datos, utilice la opción **--all-databases** en su lugar:

```
$ mysqldump --user=USUARIO --password=CONTRASEÑA --all-databases > copia_seguridad.sql
```

Incluyendo solamente algunas tablas en la copia de seguridad

De forma predeterminada, el comando **mysqldump** vuelca todas las tablas en la base de datos especificada. Para incluir solo una o más tablas, deberá indicar sus nombres después del nombre de la base de datos:

```
# volcado de una sola tabla
$ mysqldump --user=USUARIO --password=CONTRASINAL NOMBRE_BASE_DE_DATOS NOMBRE_TABLA > copia_seguridad.sql
# volcado de tres tablas
$ mysqldump --user=USUARIO --password=CONTRASINAL NOMBRE_BASE_DE_DATOS NOMBRE_TABLA_1 NOMBRE_TABLA_2 NOMBRE_TABLA_3 > copia_seguridad.sql
```

Excluyendo algunas tablas de la copia de seguridad

Para no incluir una tabla específica en la copia de seguridad, deberá indicarlo con el opción **--ignore-table**, cuyo valor se debe dar como NOMBRE_BASE_DE_DATOS.TABLE_NAME (cuando solo el nombre del tabla se produce un error):

```
$ mysqldump --user=USUARIO --password=CONTRASINAL --ignore-table=NOMBRE_BASE_DE_DATOS.NOMBRE_TABLA NOMBE_BASE_DE_DATOS > copia_seguridad.sql
```

Por ejemplo, para hacer una copia de seguridad de una base de datos llamada *store* sin que se incluya una tabla llamada *ventas* se debe ejecutar el siguiente comando:

```
$ mysqldump --user=USUARIO --password=CONTRASINAL --ignore-table=store.ventas store > copia_seguridad.sql
```

Para excluir varias tablas, sus nombres se especificarán con tantas opciones **--ignore-table**:

```
$ mysqldump --user=USUARIO --password=CONTRASINAL
--ignore-table=NOMBRE_BASE_DE_DATOS.NOMBRE_TABLA_1
--ignore-table=NOMBRE_BASE_DE_DATOS.NOMBRE_TABLA_2
--ignore-table=NOMBRE_BASE_DE_DATOS.NOMBRE_TABLA_3
```

```
NOMBRE_BASE_DE_DATOS > copia_seguridad.sql
```

Limitando el número de registros de cada tabla

De forma predeterminada, el comando mysqldump vuelca todos los registros de todas las tablas. Cuando si desea prefiltrar los registros, se agregará la opción --where, que le permite especificar una condición WHERE igual a la de las consultas SELECT:

```
$ mysqldump --user=USUARIO --password=CONTRASINAL NOMBRE_BASE_DE_DATOS NOMBRE_TABLA --where="edad > 18 AND edad < 65" > copia_seguridad.sql
```

Si lo que le interesa es simplemente limitar la cantidad de registros volcados para cada tabla, la opción **--where** se puede usar junto con el siguiente truco tomado del sitio *StackOverflow*:

```
$ mysqldump --user=USUARIO --password=CONTRASINAL --where="1 limit 1000" NOMBRE_BASE_DE_DATOS > copia_seguridad.sql
```

La opción **--where="1 limit 1000"** hace que solo se extraigan los primeros 1000 registros de cada tabla. Para ajustar este valor solo hay que cambiar el valor 1000 pero no cambiar la primera parte (*1 límit*).

Volcado de la estructura de las tablas pero no sus datos

Por defecto, el comando **mysqldump** vuelca tanto la estructura de las tablas como toda información Si lo que te interesa es volcar la estructura de las tablas y columnas, tienes que utilice la opción **--non-data**. De esta manera puedes crear otra base de datos exactamente igual pero vacía:

```
$ mysqldump --user=USUARIO --password=CONTRASINAL --no-data NOMBRE_BASE_DE_DATOS > copia_seguridad.sql
```

Creando archivos con líneas más cortas

De forma predeterminada, el comando **mysqldump** combina cientos de instrucciones INSERT individuales en una sola declaración INSERT para insertar muchos registros a la vez:

```
INSERT INTO `NOMBRE_DE_TABLA` VALUES
(1, '...', '...', '...'), (2, '...', '...', '...'),
(3, '...', '...', '...'), (4, '...', '...', '...'),
(5, '...', '...', '...'), (6, '...', '...', '...');
```

Este es el comportamiento recomendado en la mayoría de las situaciones, pero puede generar errores con sistemas antiguos incapaces de procesar líneas de miles de bytes de longitud.

Igualmente, puede tener problemas con el editor de texto al intentar abrir un archivo de copia seguridad que contiene esas líneas tan largas.

Si este es el caso, agregue la opción **--extended-insert=false** para hacer que cada INSERT se ejecuta con su propia declaración:

```
$ mysqldump --user=USUARIO --password=CONTRASINAL --extended-insert=false NOMBRE_BASE_DE_DATOS > copia_seguridad.sql
```

Cuando abra el archivo de copia de seguridad ahora, verá que las instrucciones ya no existen instrucciones INSERT múltiples:

```
INSERT INTO `NOMBRE_DE_TABLA` VALUES (1, '...', '...', '...');
INSERT INTO `NOMBRE_DE_TABLA` VALUES (2, '...', '...', '...');
INSERT INTO `NOMBRE_DE_TABLA` VALUES (3, '...', '...', '...');
INSERT INTO `NOMBRE_DE_TABLA` VALUES (4, '...', '...', '...');
INSERT INTO `NOMBRE_DE_TABLA` VALUES (5, '...', '...', '...');
INSERT INTO `NOMBRE_DE_TABLA` VALUES (6, '...', '...', '...');
```

El principal problema de esta opción es que la recuperación de datos es varias órdenes de magnitud más lenta que cuando se combinan múltiples registros en una sola instrucción INSERT.

Evitando el bloqueo de las tablas

De forma predeterminada, el comando **mysqldump** bloquea las tablas de la base de datos antes de hacer el volcado de la información. Este es el comportamiento recomendado para evitar inconsistencias al recuperar la información.

Sin embargo, cuando realiza una copia de seguridad del servidor de producción y éste tiene mucha actividad, este comportamiento puede ser inaceptable. Para evitar esto, agregue la opción **--add-locks=false**:

```
$ mysqldump --user=USUARIO --password=CONTRASINAL --add-locks=false NOMBRE_BASE_DE_DATOS > copia_seguridad.sql
```

Creando bases de datos con datos binarios

De forma predeterminada, el comando **mysqldump** vuelca toda la información tal como está almacenada en el base de datos. Cuando se guardan archivos binarios (como imágenes o archivos PDF) en algunas tablas, pueden ocurrir errores al procesar después la información binaria o al recuperarla.

Para evitar estos problemas, puede obligar a MySQL a convertir la información binario a un formato hexadecimal más seguro para la transmisión y recuperación. Para eso, se agrega la opción **--hex-blob**:

```
$ mysqldump --user=USUARIO --password=CONTRASINAL --hex-blob NOMBRE_BASE_DE_DATOS > copia_seguridad.sql
```

El principal problema de esta opción es que puede aumentar considerablemente el tamaño del archivo de la copia de seguridad.

Cambiando el formato de salida

De forma predeterminada, el comando **mysqldump** vuelca la información en formato SQL. Para usar el formato XML, se puede agregar la opción **--xml**:

```
$ mysqldump --user=USUARIO --password=CONTRASINAL --xml NOMBRE_BASE_DE_DATOS > copia_seguridad.sql
```

El contenido del archivo generado será algo como el siguiente:

copia_seguridad_test_XML.sql

```xml
<?xml version="1.0"?>
<mysqldump xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<database name="test">
    <table_structure name="prueba">
        <field Field="idprueba" Type="int unsigned" Null="NO" Key="PRI" Extra="" Comment="" />
        <key Table="prueba" Non_unique="0" Key_name="PRIMARY" Seq_in_index="1"
        Column_name="idprueba" Collation="A" Cardinality="2" Null="" Index_type="BTREE"
        Comment="" Index_comment="" Visible="YES" />
        <options Name="prueba" Engine="InnoDB" Version="10" Row_format="Dynamic"
        Rows="2" Avg_row_length="8192" Data_length="16384" Max_data_length="0" Index_length="0"
        Data_free="0" Create_time="2022-06-05 20:50:10" Collation="utf8mb4_0900_ai_ci"
        Create_options="" Comment="" />
    </table_structure>
    <table_data name="prueba">
    <row>
        <field name="idprueba">10</field>
    </row>
    <row>
        <field name="idprueba">125</field>
    </row>
    </table_data>
</database>
</mysqldump>
```

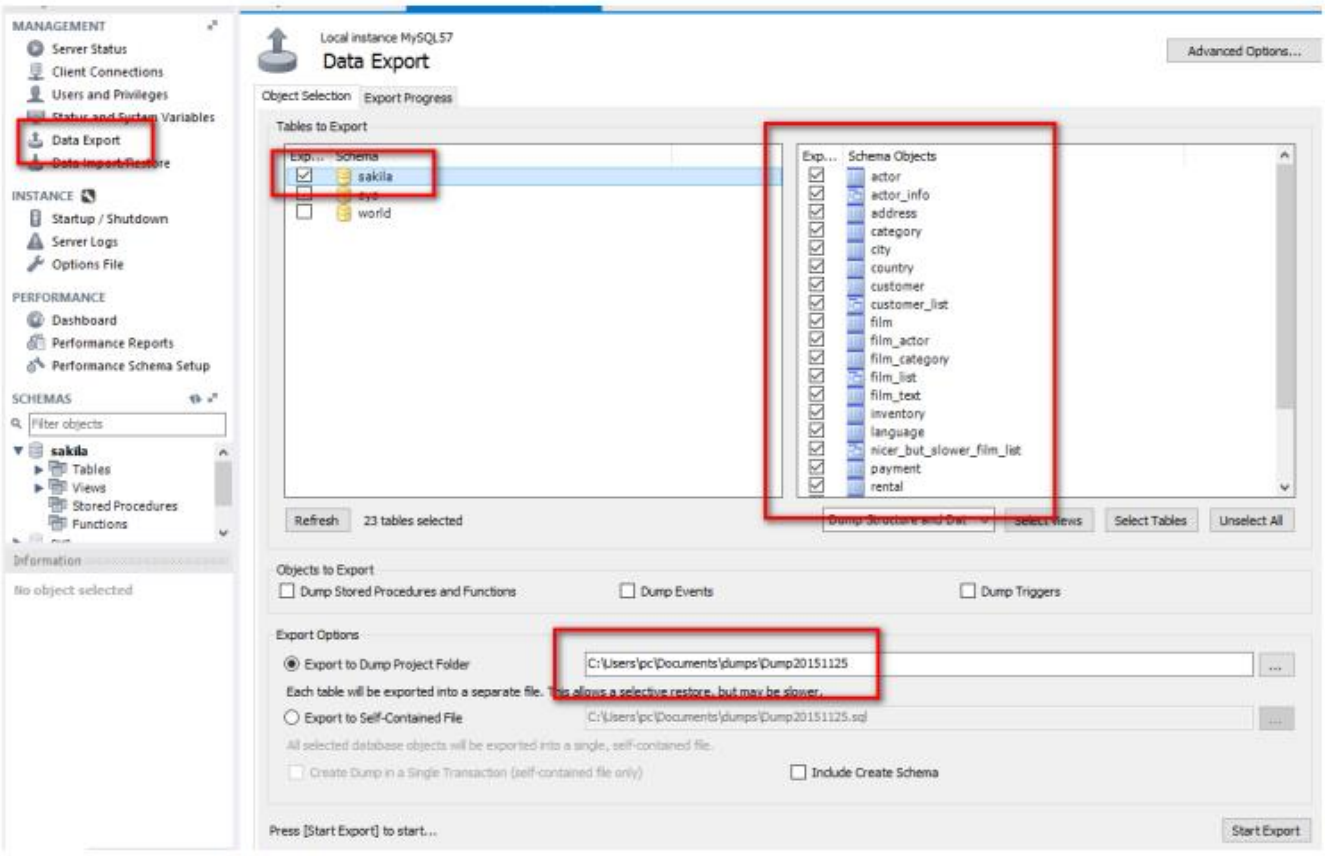
Manual oficial de ***mysqldump***: https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html

MySQL Workbench

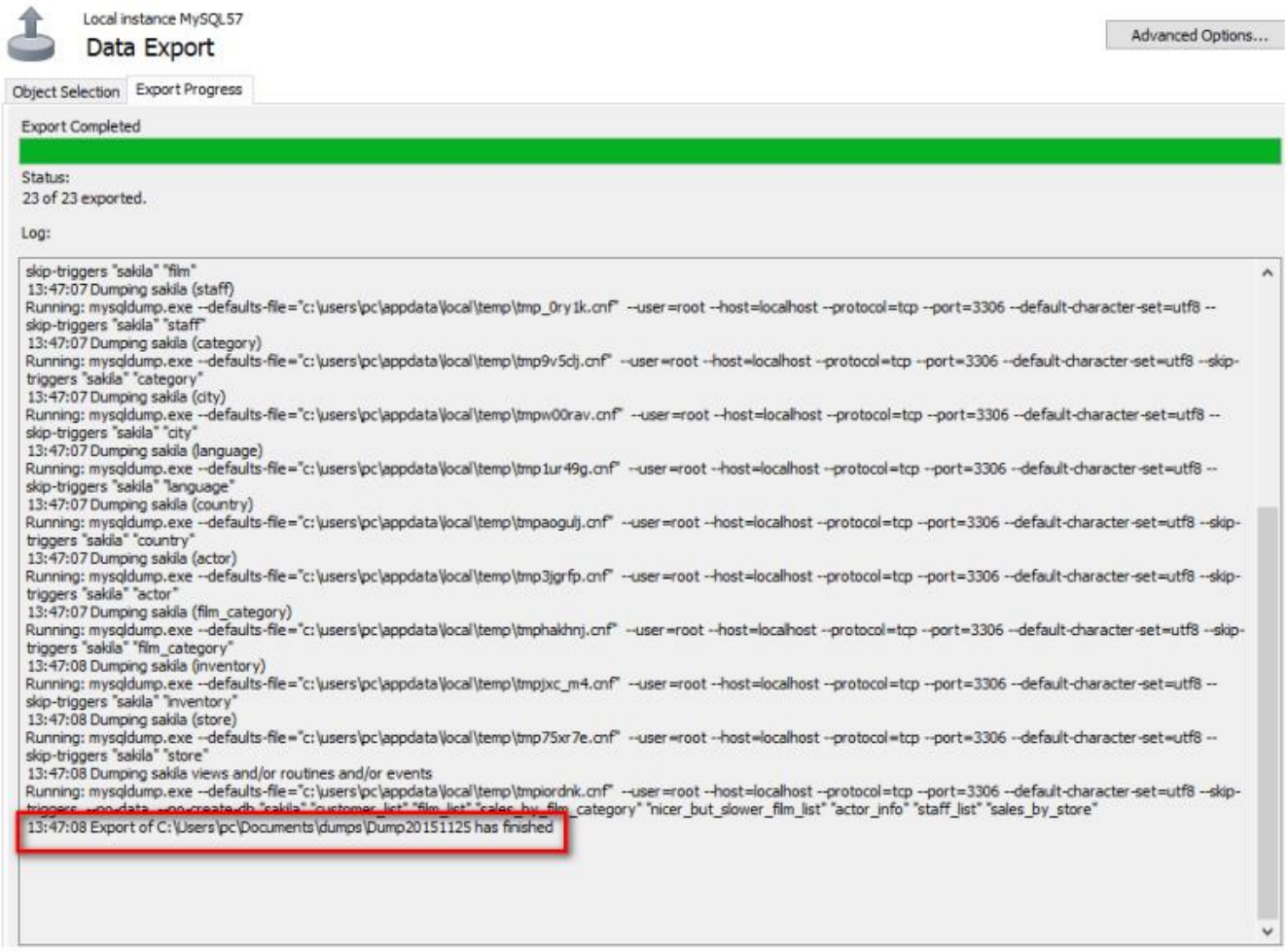
Desde la herramienta gráfica *MySQL Workbench*, también existe la posibilidad de hacer copias de seguridad de las bases de datos existentes. Para ello se utilizan las opciones exportación e importación de bases de datos.

Exportar

Mediante la opción *Data Export* se seleccionará la base de datos y las tablas de las que desea realizar la copia de seguridad, así como la ubicación del archivo de copia de seguridad.

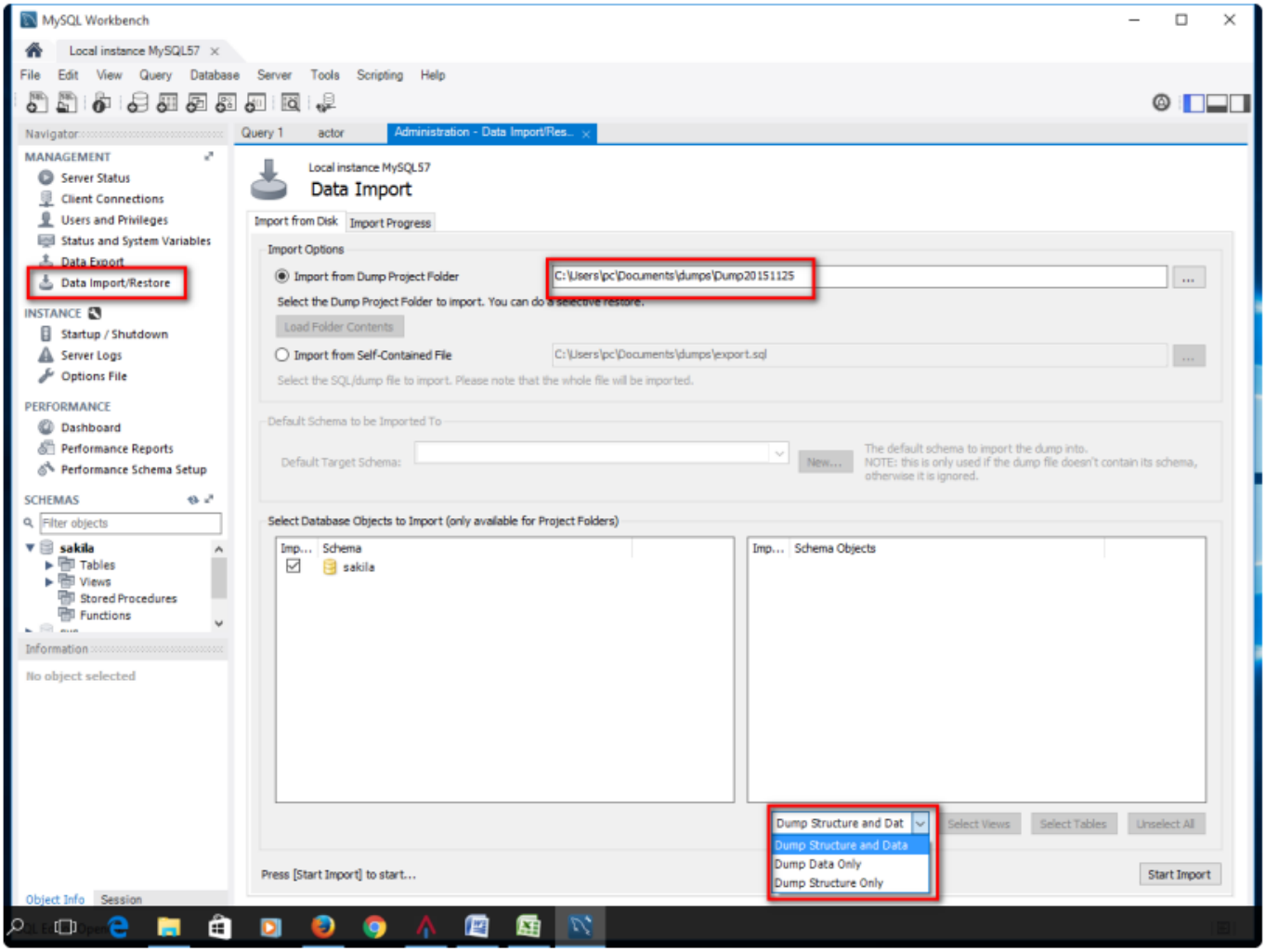


Una vez seleccionadas las opciones deseadas, se pulsará la opción "*Data Export*" y, sino hubo errores, se concluirá la copia de seguridad:

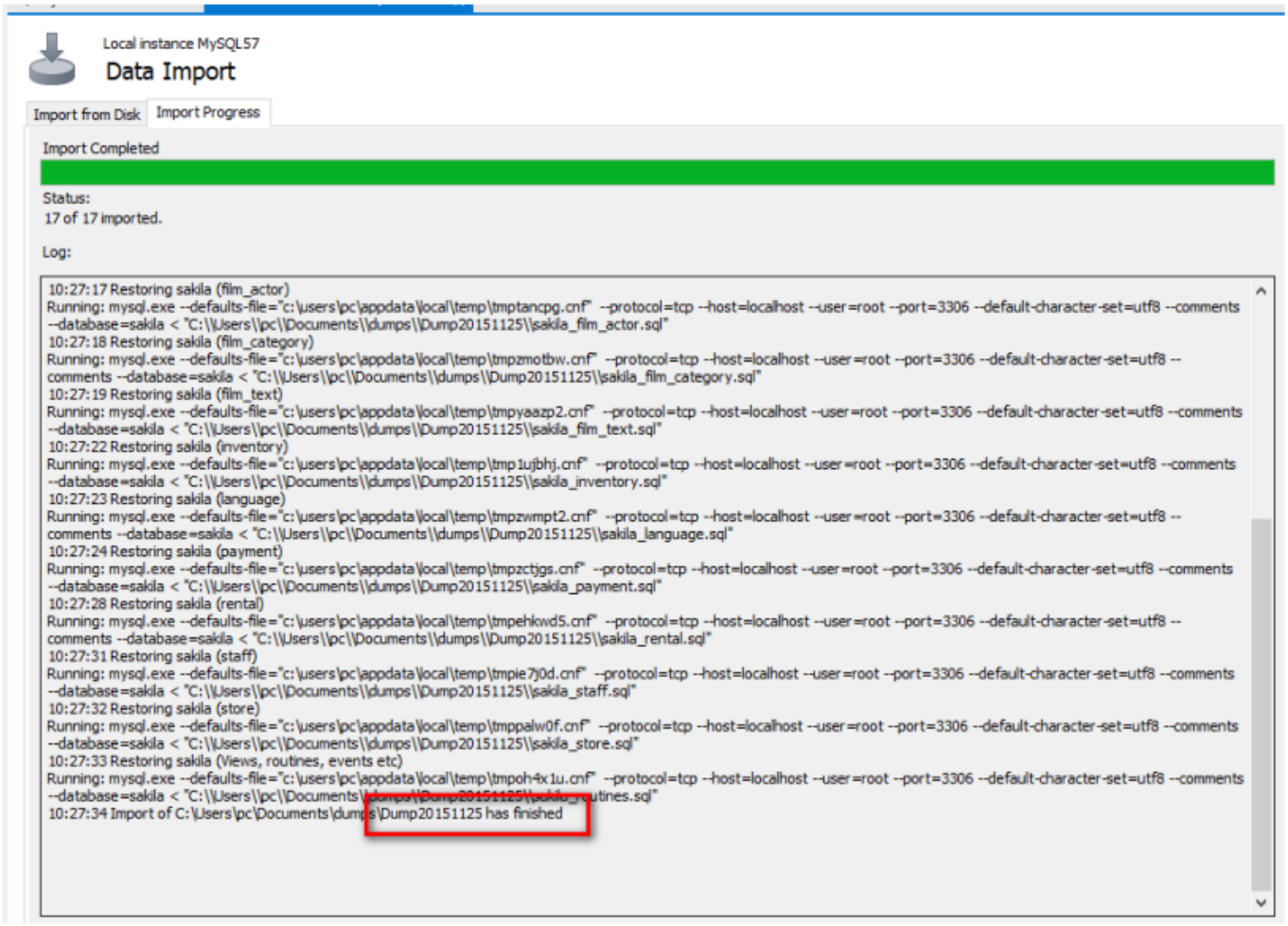


Importar

Para restaurar unha copia de seguridad hecha mediante a opción *DataExport* de *MySQL Workbench*, se utilizará la opción *Data Import/Restore*. Una vez dentro, se selecciona el directorio donde se guardo una copia de seguridad. Es importante saber que se puede seleccionar entre restaurar solo el esquema, los datos o las dos cosas:



Una vez seleccionadas las opciones deseadas, se pulsará el botón "*Start Import*" y, si no hubo errores, se concluirá la restauración de la copia de seguridad:



6.2. Intercambio de datos entre SGBDs

En algunos casos es necesario realizar un intercambio de datos, total o parcial, entre diferentes SGBD. Para facilitar esta tarea existen herramientas externas o integradas con el propio SGBD, que permiten la exportación e importación de datos entre esquemas de bases de datos, utilizando diferentes estrategias.

Intercambio de datos mediante scripts SQL

Una manera fácil de intercambiar datos entre SGBD, ya sean iguales o diferentes fabricante, es a través de scripts SQL.

Como ya se vio en la sección anterior, existen herramientas y comandos que permiten crear copias de seguridad que dan como resultado archivos con sentencias SQL de creación de estructura de la base de datos y de inserción de datos en las tablas. Importando estos los archivos en otro SGBD también podrían replicar toda o parte de la base de datos original así como los datos almacenados en él.

En caso de que quisiera importar/exportar entre modelos SGBD diferentes, sería necesario asegurarse de que existe una compatibilidad completa entre la versión de SQL utilizada por los dos sistemas gestores. Por ejemplo, es común que haya problemas con los diferentes tipos de datos utilizados para almacenar fechas.

Importar

Es posible importar datos a las tablas de una base de datos desde archivos de texto plano.

Prácticamente todas las aplicaciones diseñadas para procesar datos (hojas de cálculo, sistemas de gestión de bases de datos, ...), tienen la capacidad de importar y exportar datos desde/hacia archivos de texto (ASCII) de campos delimitados.

Un ejemplo de un archivo de texto de campos delimitados podría ser el siguiente (*personal.txt*):

```
1112345;"Martínez Salas, Fernando";"PROFESOR";2200.00;10
4123005;"Bueno Zarco, Elisa";"PROFESOR";2200.00;10
4122025;"Montes García, M.Pilar";"PROFESOR";2200.00;10
1112346;"Rivera Silvestre, Ana";"PROFESOR";2050.00;15
9800990;"Ramos Ruiz, Luis";"PROFESOR";2050.00;15
8660990;"De Lucas Fdez, M.Angel";"PROFESOR";2050.00;15
7650000;"Ruiz Lafuente, Manuel";"PROFESOR";2200.00;22
43526789;"Serrano Laguía, María";"PROFESOR";2050.00;45
4480099;"Ruano Cerezo, Manuel";"ADMINISTRATIVO";1800.00;10
1002345;"Albarrán Serrano, Alicia";"ADMINISTRATIVO";1800.00;15
7002660;"Muñoz Rey, Felicia";"ADMINISTRATIVO";1800.00;15
5502678;"Marín Marín, Pedro";"ADMINISTRATIVO";1800.00;22
```

Estos archivos se caracterizan por estar almacenados en formato ASCII, normalmente con las extensiones: *.txt* o *.csv*. Tienen los registros de longitud variable separados por dos puntos carácter de nueva línea: **\n** (ASII 10) en Unix y Mac y **\r\n** (ASCII 10 y 13) en Windows, los campos están delimitados por un carácter especial (en este caso el ";") y normalmente los datos de tipo de texto se encierran entre comillas.

MySQL proporciona dos métodos para importar datos directamente desde este tipo de archivos en tablas. Uno es la utilidad *mysqlimport*, y el otro método consiste en ejecutar la instrucción **LOAD DATA INFILE**. Ambos métodos se discutirán a continuación.

LOAD DATA INFILE

La sintaxis de la sentencia **LOAD FILE** es la siguiente:

```
LOAD DATA [LOCAL] INFILE 'path/nome_ficheiro.txt'
  [REPLACE | IGNORE]
  INTO TABLE nome_táboa
[FIELDS
   [TERMINATED BY '\t']
   [[OPTIONALLY] ENCLOSED BY '']
   [ESCAPED BY '\\' ]
]
          [LINES
   [STARTING BY '']
   [TERMINATED BY '\n']
]
[IGNORE número LINES]
[(nome_columna,...)]
```

La mayoría de las opciones ya están vistas en el Tema 4. Solo añadir que cuando se quiera ignorar las restricciones clave foránea durante la operación de carga, se puede ejecutar el comando **SET FOREIGN_KEY_CHECKS=0** antes de ejecutar LOAD DATA.
Es importante saber que para poder ejecutar una sentencia LOAD DATA es necesario ser administrador de la base de datos o tener concedido el privilegio FILE.

Mysqlimport

Mysqlimport proporciona una interfaz de línea de comandos para el comando LOAD DATA INFILE. De hecho, varias de las opciones de mysqlimport corresponden directamente con cláusulas LOAD DATA INFILE.

La sintaxis de *mysqlimport* es la siguiente:

```
shell> mysqlimport [opcións] nombre_de_base_de_datos fichero_de_texto1 [fichero_de_texto2 ...]
```

De cada nombre de archivo de texto especificado en la línea de comando, *mysqlimport* elimina cualquier extensión, y usa el resultado para determinar el nombre de la tabla para importar el contenido del archivo. Por ejemplo, archivos denominados *patient.txt*, patient.text y *patient* se importarán todos a la tabla llamada *patient.*
mysqlimport admite las siguientes opciones:

`--fields-terminated-by=...`, **`--fields-enclosed-by=...`**, **`--fields-optionally-enclosed-by=...`**, **`--fields-escaped-by=...`**, **`--lines-terminated-by=...`**, **`--ignore,-i`** **`--replace,-r`** **`--ignorelines=n`**. Estas opciones tienen el mismo significado que las cláusulas correspondientes de LOAD DATA INFILE.

`--help`, **`-?.`** Muestra un mensaje de ayuda.

`--columns=lista_de_columnas`, **`-c lista_de_columnas`**. Esta opción admite una lista de nombres de columnas separados por comas. La orden de dos nombres de columna indica como emparejar las columnas de los ficheros de datos con las columnas de la tabla.

`--compress`, **`-C.`** Comprime toda la información enviada entre el cliente y el servidor, si ambos soportan compresión.

`--debug[=opciones_de_depuracion]`, **`-#[opciones_de_depuracion]`**. Escribe un log de depuración.

`--delete`, **`-D`**. Vacia la tabla antes de importar el ficheiro de texto.

`--force`, **`-f`**. Ignora errores. Por ejemplo, cuando una tabla para un fichero de texto no existe, sigue procesando el resto de ficheros. Sin --force, mysqlimport finaliza si la tabla no existe.

`--host=nombre_de_equipo`, **`-h nombre_de_equipo`**. Importa datos al servidor MySQL en el equipo dado. El equipo por defecto es localhost.

`--local`, **`-L`**. Lee los ficheiros de entrada localmente del equipo cliente.

--lock-tables, -l. Bloquea todas las tablas para escritura antes de procesar cualquier fichero de texto. Esto asegura que todas las tablas estén sincronizadas en el servidor.

--password[=contrasinal], -p[contrasinal]. La contraseña a usar cuando se conecta al servidor. Cuando se usa la opción en su forma corta (-p), no puede haber un espacio entre la opción y la contraseña. Cuando se omite el valor de contraseña a continuación de **--password** o **-p** en la línea de comandos, aparece un prompt para que lo introduzca.

--port=número_de_puerto, -P número_de_puerto. El puerto TCP/IP para usar en la conexión.

--protocol={TCP | SOCKET | PIPE | MEMORY}. El protocolo de conexión a usar.

--user=nombre_de_usuario, -u nombre_de_usuario. El nombre de usuario MySQL a usar cuando se conecta con el servidor.

--verbose, -v. Modo explícito. Muestra más información sobre lo que hace el programa.

--version, -V. Muestra información de versión.

Un ejemplo de uso de mysqlimport podría ser:

```
shell> mysqlimport --local test imptest.txt
```

ÍNDICE