

Módulo 5: Práctica INSERT-SELECT-DELETE-UPDATE

CONSULTAS INSERT-SELECT-DELETE-UPDATE – CONSULTAS WHERE, OPERADORES LÓGICOS Y RELACIONALES-COPIA DE SEGURIDAD DE UNA BASE DE DATOS

HERRAMIENTAS USADAS: MYSQL Y WORKBENCH

Representación visual de las tablas

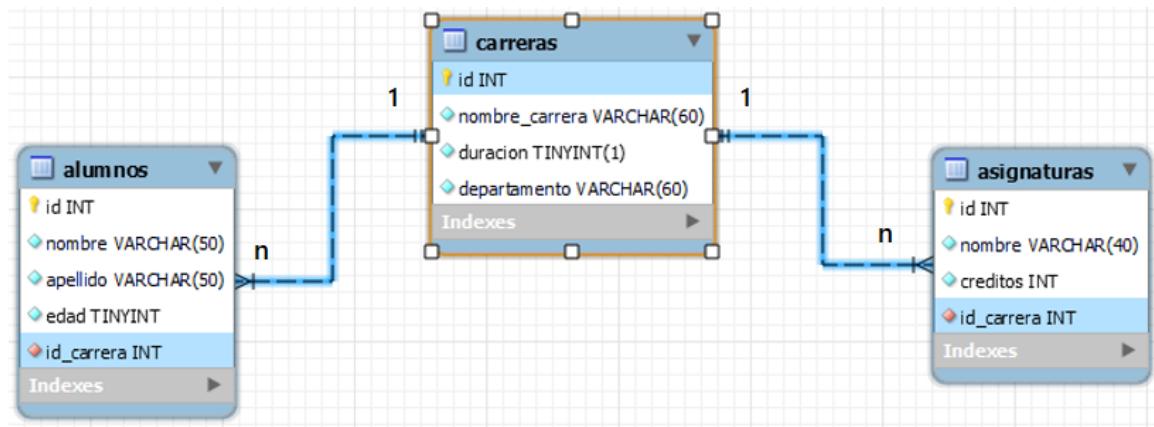


Table: **alumnos**

Columns:

id	int PK
nombre	varchar(50)
apellido	varchar(50)
edad	tinyint
id_carrera	int

Table: **asignaturas**

Columns:

id	int PK
nombre	varchar(40)
creditos	int
id_carrera	int

Table: **carreras**

Columns:

id	int PK
nombre_carrera	varchar(60)
duracion	tinyint(1)
departamento	varchar(60)

Tipos de datos usados:

TINYINT

Entero pequeño, rango de valores (de 0 a 255). Ocupa 1 byte de almacenamiento.

INT

Entero estándar, rango de valores (de 0 a 4294967295). Ocupa 4 bytes de almacenamiento.

VARCHAR(n)

Almacena cadenas de longitud variable con un máximo de n caracteres.

El valor de n puede ser de 1 a 65,535. Si se requiere un texto más largo, se pueden usar tipos de datos TEXT.



INSERCIÓN DE REGISTROS DE UNA TABLA (INSERT)- MOSTRAR REGISTROS (SELECT)

INSERT

SELECT

TABLA CARRERAS

Insertamos usando la forma más clásica:

Código

```
INSERT INTO tabla (columna1, columna2, columna3) VALUES (valor1, valor2, valor3);
```

Ejemplo1:

Vamos a insertar en la tabla carreras, una nueva carrera llamada 'CIENCIA DE DATOS' con 5 años de duración perteneciente al departamento de 'SISTEMAS'

Código:

```
use gestionacademica;  
  
INSERT INTO carreras (id,nombre_carrera,duracion,departamento) VALUES  
(1,'CIENCIA DE DATOS', '5', 'SISTEMAS');
```

Salida:

```
SELECT * FROM carreras;
```

	id	nombre_carrera	duracion	departamento
▶	1	CIENCIA DE DATOS	5	SISTEMAS

(Siempre al comienzo de escribir código sql se usa la sentencia **use** que nos dice que vamos a usar la base de datos que definamos)

Ejemplo2:

Vamos a insertar en la tabla carreras, 2 carreras, 'ANALISIS DE DATOS' y 'DISEÑO WEB' , en los registros con id=2 , id=3 .

Código:

```
use gestionacademica;  
  
INSERT INTO carreras (id,nombre_carrera,duracion, departamento) VALUES  
(2,'ANALISIS DE DATOS', '3', 'SISTEMAS');
```



```
INSERT INTO carreras (id,nombre_carrera,duracion, departamento) VALUES
(3,'DISEÑO WEB', '3', 'ARTES Y DISEÑO');
```

Salida:

```
SELECT * FROM carreras; .
```

	id	nombre_carrera	duracion	departamento
▶	1	CIENCIA DE DATOS	5	SISTEMAS
	2	ANALISIS DE DATOS	3	SISTEMAS
*	3	DISEÑO WEB	3	ARTES Y DISEÑO
*	NULL	NULL	NULL	NULL

TABLA ALUMNOS

Insertamos usando la siguiente forma:

Código

```
INSERT INTO tabla SET columna1 = valor1, columna2 = valor2;
```

INSERT ... SET: Similar a `INSERT ... VALUES`, pero permite asignar valores a las columnas usando la sintaxis `SET`. Es útil cuando se desea insertar datos de forma más explícita, especialmente si se están modificando algunas columnas.

Ejemplo 1:

Vamos a insertar un alumno con nombre y apellido Juan Soto, con una edad de 25 años y que se inscribirá en la carrera con id=1 (CIENCIA DE DATOS)

```
INSERT INTO alumnos SET id = 101, nombre="Juan", apellido="Soto", edad = 25,
id_carrera = 1;
```

Salida:

```
SELECT * FROM alumnos;
```

id	nombre	apellido	edad	id_carrera
101	Juan	Soto	25	1

El id asignado es 101

Ejemplo 2:

Vamos a insertar 3 alumnos Gabriela, Diego y Mariela con los apellidos, edades y carreras correspondientes siguiendo la misma sintaxis que el primer ejemplo



Código:

```
INSERT INTO alumnos SET id = 102, nombre = "Gabriela", apellido = "Ruiz", edad = 35 , id_carrera = 2;

INSERT INTO alumnos SET id = 103, nombre = "Diego", apellido = "Lopez", edad = 29 , id_carrera = 3;

INSERT INTO alumnos SET id = 104, nombre = "Mariela", apellido = "Puertas", edad = 19 , id_carrera = 1;

SELECT * FROM alumnos;
```

id	nombre	apellido	edad	id_carrera
101	Juan	Soto	25	1
102	Gabriela	Ruiz	35	2
103	Diego	Lopez	29	3
104	Mariela	Puertas	19	1

Gabriela esta inscripta en la carrera 2 que corresponde a ANALISIS DE DATOS, Diego a la carrera DISEÑO WEB y Mariela a la carrera CIENCIA DE DATOS

TABLA ASIGNATURAS

Insertamos datos visualmente. Insertamos usando las opciones que nos da Workbench

The screenshot illustrates the three steps of inserting data into the 'asignaturas' table:

- 1 SELECCIONAMOS**: The 'asignaturas' table is selected in the Navigator under the 'gestionacademica' schema.
- 2- INSERTAMOS VISUALMENTE**: The data is inserted into the 'Result Grid' table. The grid shows the following data:

id	nombre	creditos	id_carrera
1	MATEMÁTICA I	8	1
2	PROGRAMACION I	12	1
3	BASE DE DATOS	10	1
4	ESTRUCTURA DE DATOS	8	1
5	INTELIGENCIA ARTIFICIAL I	6	1

The 'Apply' button at the bottom right of the grid is highlighted with a red circle.



1 • `SELECT * FROM asignaturas;`

	id	nombre	creditos	id_carrera
▶	1	MATEMÁTICA I	8	1
	2	PROGRAMACION I	12	1
	3	BASE DE DATOS	10	1
	4	ESTRUCTURA DE DATOS	8	1
*	5	INTELIGENCIA ARTIFICIAL I	6	1
	NULL	NULL	NULL	NULL

MODIFICACIÓN DE REGISTROS DE UNA TABLA (UPDATE)

UPDATE

Para modificar uno o varios datos de uno o varios registros utilizamos "update" (actualizar).

Podemos modificar algunos registros, para ello debemos establecer condiciones de selección con "where".

La consulta UPDATE en MySQL se utiliza para modificar los datos existentes en una tabla.

La sintaxis básica es:

```
UPDATE tabla SET columnal = valor1, columna2 = valor2, ... WHERE  
condición;
```

El WHERE es la condición que le asignamos a la consulta, es opcional, pero es importante saber que, si se omite, se actualizarán todas las filas de la tabla, por eso es recomendable su uso y su importancia en las consultas.

Ejemplo1:

1. Actualizar una sola columna:

Código

```
UPDATE alumnos SET nombre='Juan Jose' WHERE id=101;
```

Este ejemplo actualiza el nombre a Juan Jose para el alumno con id = 101.



Ejemplo2:

2. Actualizar múltiples columnas:

Código



```
UPDATE asignaturas SET nombre = 'BASE DE DATOS I', creditos = 12 WHERE id = 3;
```

Este ejemplo actualiza el nombre a BASE DE DATOS I y créditos a 12 para la asignatura con id =3.

La cláusula WHERE es crucial para especificar qué filas se verán afectadas por la actualización. Sin la cláusula WHERE, se modificarán todas las filas de la tabla. Es importante usarla con cuidado para evitar modificar datos no deseados. Además, se pueden utilizar funciones y subconsultas para realizar actualizaciones más complejas.

ELIMINACIÓN DE REGISTROS DE UNA TABLA

DELETE

La instrucción DELETE en MySQL se usa para eliminar filas de una tabla. Se puede usar con una cláusula WHERE para eliminar filas específicas o sin ella para eliminar todas las filas de la tabla.

Sintaxis básica:

Código



```
DELETE FROM nombre_tabla  
WHERE condición;
```

- **nombre_tabla:** Es el nombre de la tabla de la que se eliminarán las filas.
- **condición:** Es una expresión que especifica qué filas se eliminarán. Si se omite, se eliminarán todas las filas de la tabla.

Ejemplo1:

1. Eliminar una fila específica:

Código



```
DELETE FROM alumnos WHERE id = 103;
```



Este ejemplo elimina la fila de la tabla alumnos donde el ID es igual a 103

Ejemplo2:

2. Eliminar varias filas que coincidan con una condición:

Código



```
DELETE FROM asignaturas WHERE id_carrera = 1 AND creditos = 8;
```

Este ejemplo elimina todas las filas de la tabla asignaturas donde la id_carrera es 1 y el créditos es 8.

id	nombre	creditos	id_carrera
1	MATEMÁTICA I	8	1
2	PROGRAMACION I	12	1
3	BASE DE DATOS	10	1
4	ESTRUCTURA DE DATOS	8	1
5	INTELIGENCIA ARTIFICIAL I	6	1

se eliminarían esos 2 registros

Ejemplo 3:

3 Eliminar todas las filas de una tabla:

Código



```
DELETE FROM alumnos;
```

Este ejemplo elimina todas las filas de la tabla alumnos. ¡Cuidado con este! Asegúrate de que realmente quieras eliminar todo antes de ejecutarlo.

SIEMPRE ES RECOMENDABLE CREAR UNA COPIA DE SEGURIDAD EN TABLAS Y BASE DE DATOS

AL FINAL DE ESTE DOCUMENTO TE EXPLICARÉ COMO HACERLO

OPERADORES DE
COMPARACIÓN

OPERADORES

Los operadores relacionales que veremos son los siguientes:

=	igual
<>	distinto
>	mayor
<	menor
>=	mayor o igual
<=	menor o igual
!=	distinto a



Operador <>

En MySQL, el operador `<>` se utiliza para representar la comparación de "distinto de", es decir, para verificar si dos valores son diferentes

Ejemplo 1

Supongamos que tenemos la tabla alumnos completada de la siguiente manera, Y queremos traer todos los registros menos el del alumno Diego

id	nombre	apellido	edad	id_carrera
101	Juan Jose	Soto	25	1
102	Gabriela	Ruiz	35	2
103	Diego	Lopez	29	3
104	Mariela	Puertas	19	1
NULL	NULL	NULL	NULL	NULL

Podemos seleccionar los registros cuyo nombre sea distinto de 'Diego', para ello usamos la condición:

```
SELECT nombre, apellido, edad FROM alumnos WHERE nombre <> 'Diego';
```

nombre	apellido	edad
Juan Jose	Soto	25
Gabriela	Ruiz	35
Mariela	Puertas	19

Ejemplo 2

Queremos listar solo las carreras que no tengan 5 años de duración. La sentencia sería la siguiente:

```
SELECT * FROM carreras WHERE duracion <> 5;
```

id	nombre_carrera	duracion	departamento
1	CIENCIA DE DATOS	5	SISTEMAS
2	ANALISIS DE DATOS	3	SISTEMAS
3	DISEÑO WEB	3	ARTES Y DISEÑO
NULL	NULL	NULL	NULL

➡

id	nombre_carrera	duracion	departamento
2	ANALISIS DE DATOS	3	SISTEMAS
3	DISEÑO WEB	3	ARTES Y DISEÑO

Operador =

En MySQL, el operador `=` se utiliza para la comparación de igualdad. Se usa para verificar si dos valores son iguales

Ejemplo 1:

Listamos de la tabla carreras solo las carreras que tenga 5 años de duración



```
SELECT * FROM carreras WHERE duracion = 5;
```

	id	nombre_carrera	duracion	departamento
▶	1	CIENCIA DE DATOS	5	SISTEMAS

Ejemplo 2:

Listamos de la tablas asignaturas las asignaturas que tienen créditos por valor de 12

```
SELECT * FROM asignaturas WHERE creditos = 12;
```

id	nombre	creditos	id_carrera
2	PROGRAMACION I	12	1
3	BASE DE DATOS I	12	1

Operador > , Operador >= . Operador mayor , Operador mayor que

Ejemplo 1:

Podemos comparar valores numéricos. Por ejemplo, queremos mostrar los registros cuya edad sean mayores a 25 en la tabla alumnos

```
SELECT nombre, apellido, edad FROM ALUMNOS where edad>25;
```

	nombre	apellido	edad
▶	Gabriela	Ruiz	35
	Diego	Lopez	29

Ejemplo 2 :

Mostrar los registros cuya edad sean mayores o iguales a 25 en la tabla alumnos. Vemos que incluye a los alumnos que tienen 25 o más años

```
SELECT nombre, apellido, edad FROM ALUMNOS where edad>=25;
```

	nombre	apellido	edad
▶	Juan Jose	Soto	25
	Gabriela	Ruiz	35
	Diego	Lopez	29

Operador < , Operador <= . Operador menor , Operador menor que

Ejemplo 1:

Este ejemplo listamos las asignaturas que tienen un crédito menor a 8

```
SELECT nombre, creditos FROM asignaturas WHERE creditos < 8;
```

	nombre	creditos
▶	INTELIGENCIA ARTIFICIAL I	6

Ejemplo 2 :

Este ejemplo listamos las asignaturas que tienen un crédito menor o igual a 8. Vemos que también incluye al valor 8

```
SELECT nombre, creditos FROM asignaturas WHERE creditos<=8;  
SELECT * FROM asignaturas;
```

▶	MATEMÁTICA I	8
	ESTRUCTURA DE DATOS	8
	INTELIGENCIA ARTIFICIAL I	6

Operador !=

En MySQL, el operador != se utiliza para verificar si dos valores son diferentes, devolviendo TRUE si lo son y FALSE si no lo son. También se puede usar el operador <> para el mismo propósito, ya que ambos son equivalentes.

Ejemplo 1:

En este ejemplo mostramos de la tabla asignaturas , las asignaturas con un crédito distinto a 8

```
SELECT * FROM asignaturas WHERE creditos != 8;
```

	id	nombre	creditos	id_carrera
▶	2	PROGRAMACION I	12	1
	3	BASE DE DATOS I	12	1
	5	INTELIGENCIA ARTIFICIAL I	6	1



WHERE

OTROS EJEMPLOS DEL USO DE LA CLÁUSULA WHERE

Aquí tienes 7 ejemplos del uso de la cláusula WHERE aplicados a la base de datos gestionacademica, utilizando diferentes condiciones relevantes para las tablas alumnos, carreras y asignaturas. También se utilizan operadores lógicos **AND**, **OR**.

Los operadores lógicos **AND** y **OR** se utilizan para combinar condiciones en una cláusula **WHERE** o en sentencias **SELECT**, **UPDATE** o **DELETE**.

AND

El operador AND devuelve TRUE si todas las condiciones conectadas por él son verdaderas, mientras que OR devuelve TRUE si al menos una de las condiciones conectadas por él es verdadera.

OR

1. Seleccionar todos los alumnos de una carrera específica

```
SELECT * FROM alumnos WHERE id_carrera = 3;
```

id	nombre	apellido	edad	id_carrera
103	Diego	Lopez	29	3

- ◆ Muestra todos los alumnos que están inscriptos en la carrera con id = 3.

2. Buscar asignaturas que tengan más de 5 créditos

```
SELECT * FROM asignaturas WHERE creditos > 5;
```

id	nombre	creditos	id_carrera
1	MATEMÁTICA I	8	1
2	PROGRAMACION I	12	1
3	BASE DE DATOS I	12	1
4	ESTRUCTURA DE DATOS	8	1
5	INTELIGENCIA ARTIFICIAL I	6	1

- ◆ Filtra y muestra las asignaturas que otorgan más de 5 créditos.

3. Obtener alumnos que tengan menos de 21 años

```
SELECT * FROM alumnos WHERE edad < 21;
```



id	nombre	apellido	edad	id_carrera
104	Mariela	Puertas	19	1

- ◆ Lista los estudiantes jóvenes, menores de 21 años.
-

4. Mostrar carreras del departamento 'SISTEMAS'

```
SELECT * FROM carreras WHERE departamento = 'SISTEMAS';
```

id	nombre_carrera	duracion	departamento
1	CIENCIA DE DATOS	5	SISTEMAS
2	ANALISIS DE DATOS	3	SISTEMAS

- ◆ Muestra todas las carreras que pertenecen al departamento de SISTEMAS.
-

5. Buscar asignaturas de la carrera de ID 1 y con al menos 4 créditos

```
SELECT * FROM asignaturas WHERE id_carrera = 1 AND creditos >= 4;
```

id	nombre	creditos	id_carrera
1	MATEMÁTICA I	8	1
2	PROGRAMACION I	12	1
3	BASE DE DATOS I	12	1
4	ESTRUCTURA DE DATOS	8	1

- ◆ Asignaturas de una carrera específica que tienen 8 créditos o más.
-

6. Obtener alumnos cuyo apellido sea 'Lopez'

```
SELECT * FROM alumnos WHERE apellido = 'Lopez';
```

id	nombre	apellido	edad	id_carrera
103	Diego	Lopez	29	3

- ◆ Devuelve todos los alumnos con apellido 'Lopez'.
-

7. Seleccionar carreras con duración igual a 5 años o del departamento 'ARTES Y DISEÑO'

```
SELECT * FROM carreras WHERE duracion = 5 OR departamento = 'ARTES Y DISEÑO';
```

id	nombre_carrera	duracion	departamento
1	CIENCIA DE DATOS	5	SISTEMAS
3	DISEÑO WEB	3	ARTES Y DISEÑO

- ◆ Devuelve carreras que duran 5 años o que pertenecen al departamento indicado.

COMO CREAR COPIA DE SEGURIDAD EN WORKBENCH

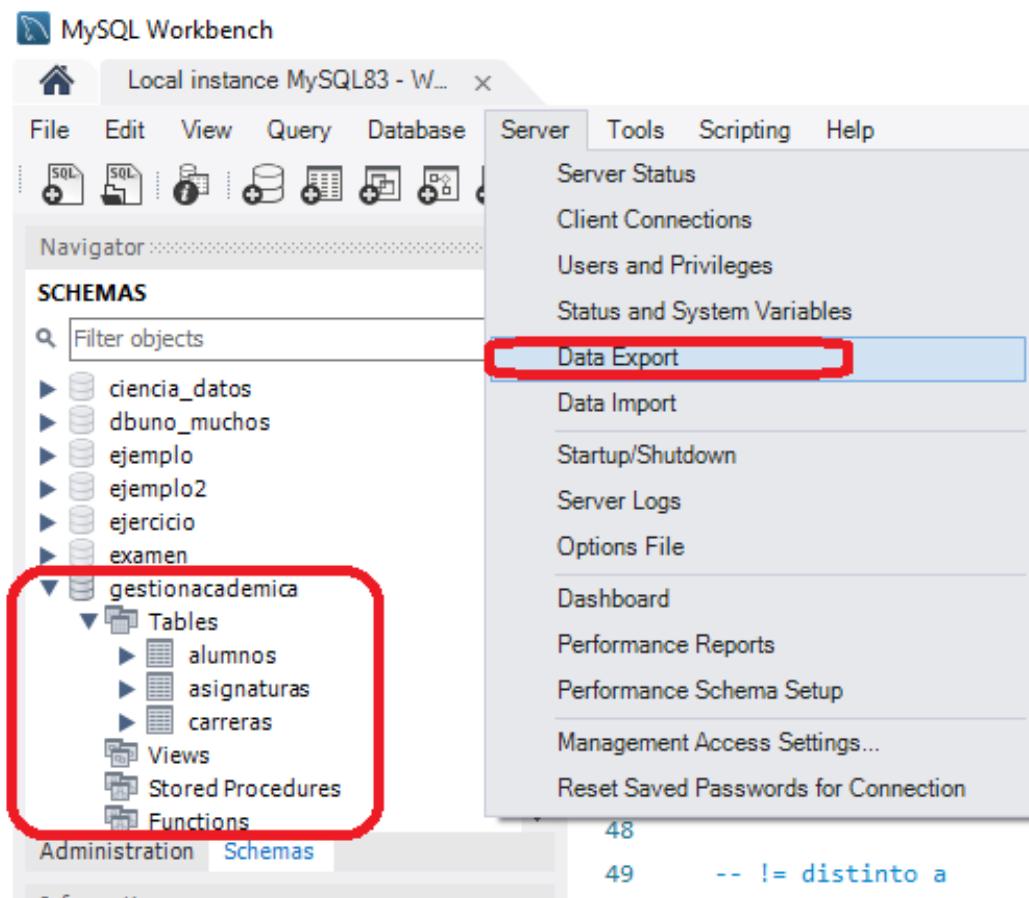
Para hacer una copia de seguridad de una tabla en MySQL Workbench, primero debes conectarte a tu base de datos, luego seleccionar la opción "Data Export" en el menú "Server". Selecciona la base de datos y las tablas que deseas respaldar, elige la opción de exportación y ejecuta el proceso. Finalmente, verifica la copia de seguridad.

PASOS

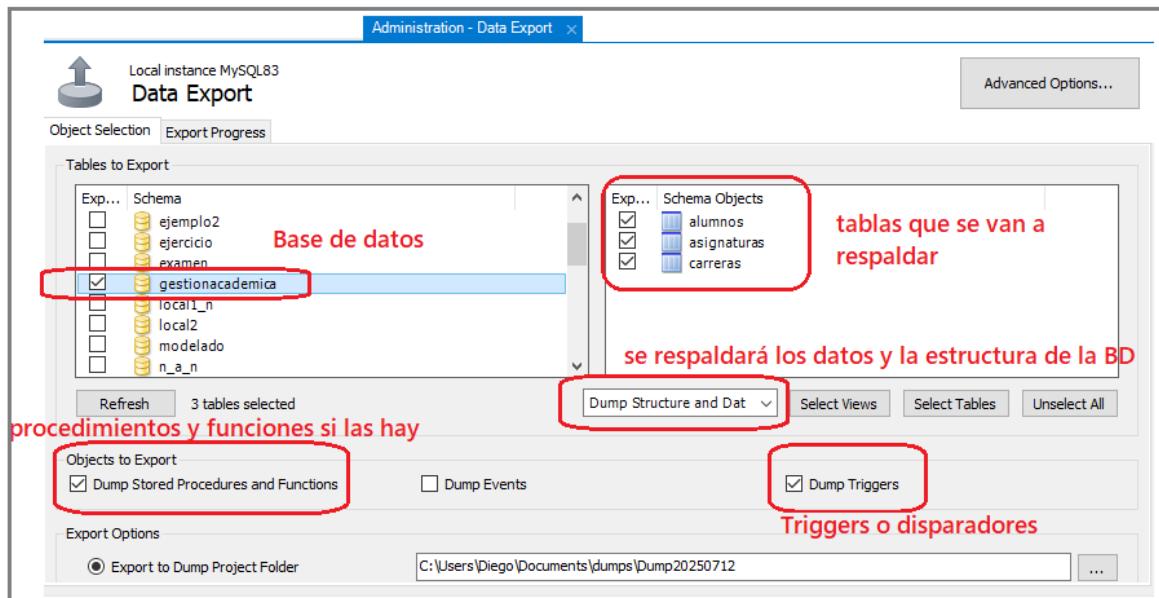
EXPORTAR BASE DE DATOS

1. Ir a Server->DataExport

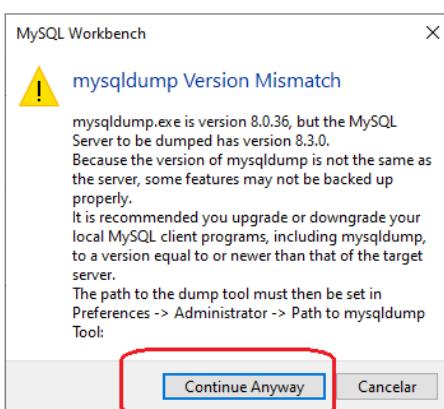
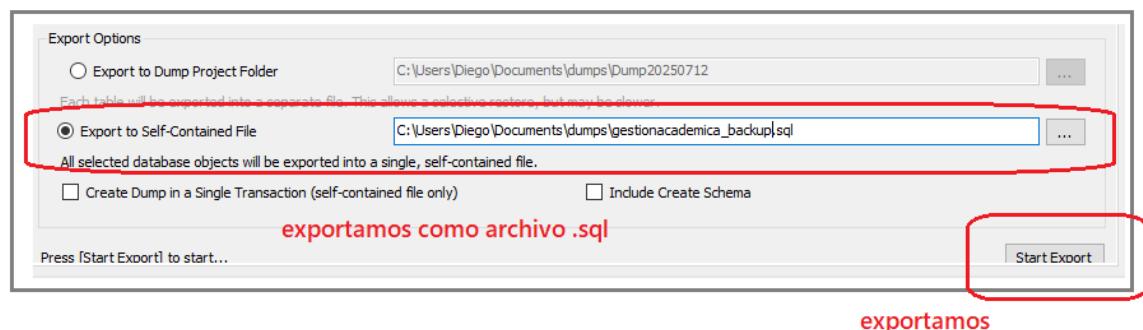
BASE DE
DATOS



2. Seleccionamos las siguientes opciones



3. Seleccionamos Start Export para exportar la base de datos



me dice que hay una
diferencia de versiones
pero no afecta a la
exportación.
Continuamos de todos
modos



4. Proceso completado exitosamente

The screenshot shows the MySQL Data Export interface. At the top, it says "Local instance MySQL83" and "Data Export". Below that, there are tabs for "Object Selection" and "Export Progress". The "Export Progress" tab is active, showing a green progress bar labeled "Export Completed". Underneath, the status message says "Status: 3 of 3 exported.". A log window displays the command-line output of the mysqldump process:

```
09:43:30 Dumping gestionacademica (all tables)
Running: mysqldump.exe --defaults-file="C:\Users\Diego\AppData\Local\Temp\tmpu6ptp8rz.cnf" --host=localhost --port=3306 --default-character-set=utf8 --
user=root --protocol=tcp --routines="gestionacademica"
09:43:31 Export of C:\Users\Diego\Documents\dumps\gestionacademica_backup.sql has finished
```

At the bottom right of the interface are "Stop" and "Export Again" buttons.

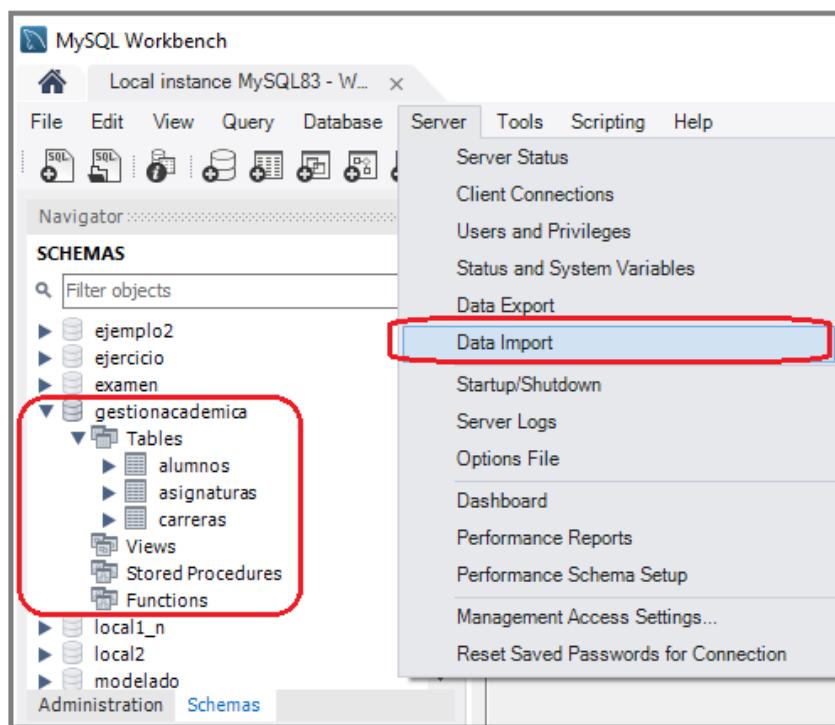
The screenshot shows a Windows File Explorer window. The path is "Docu... > dumps >". The contents of the "dumps" folder are listed, including "Dump20250612" (highlighted) and "gestionacademica_backup.sql". The left sidebar shows standard folder icons for Escritorio, Descargas, Documentos, Imágenes, and PROGRAMACI II.

IMPORTAR BASE DE DATOS

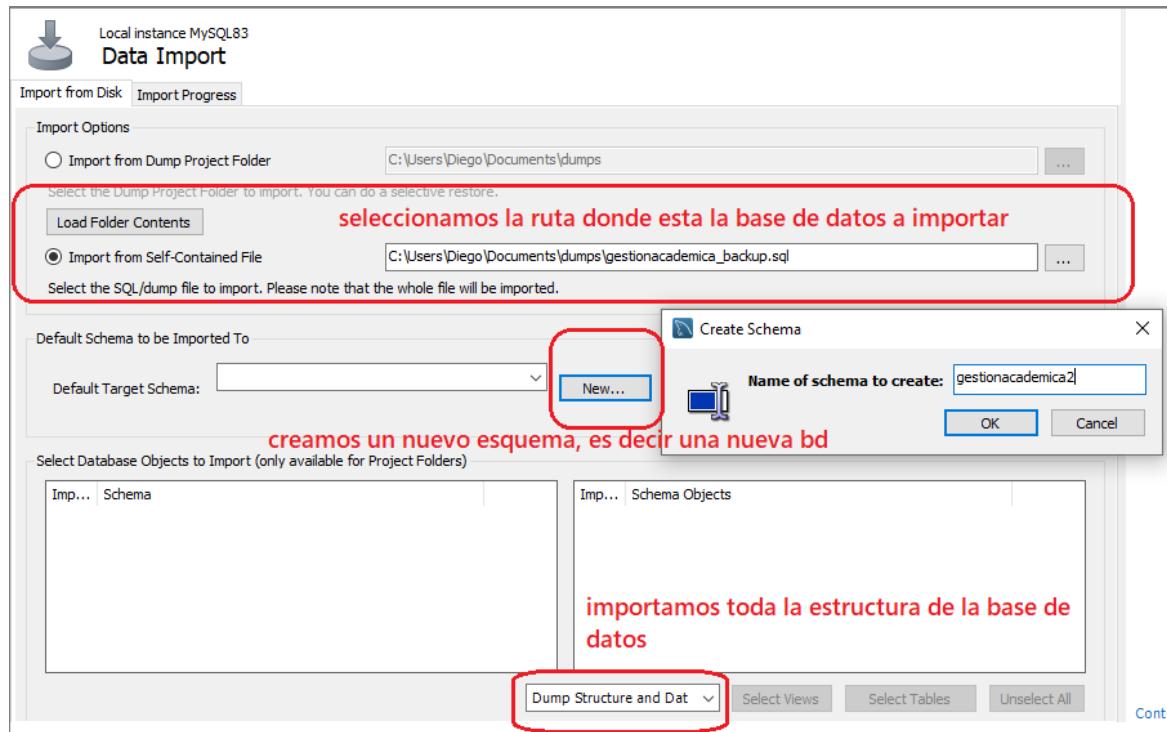
1. ir a Server->DataExport
 - a. Una vez exportada la base de datos la sentencia DROP para eliminar la base de datos "gestionacademica". (DROP DATABASE gestionacademica).
 - b. Esto lo hacemos para practicar la recuperación de la base de datos que terminamos de hacer la copia y posterior eliminación.



**BASE DE
DATOS A
IMPORTAR**



2- Seleccionamos las siguientes opciones



3- Importamos



Local instance MySQL83
Data Import

Import from Disk Import Progress

Import Completed

Status:
1 of 1 imported.

Log:

```
Creating schema gestionacademica2
09:59:38 Restoring C:\Users\Diego\Documents\dumps\gestionacademica_backup.sql
Running: mysql.exe --defaults-file="C:\Users\Diego\AppData\Local\Temp\tmpmbigo4a.cnf" --protocol=tcp --host=localhost --user=root --port=3306 --default-character-set=utf8 --comments --database=gestionacademica2 < "C:\Users\Diego\Documents\dumps\gestionacademica_backup.sql"
09:59:39 Import of C:\Users\Diego\Documents\dumps\gestionacademica_backup.sql has finished
```

Output: Stop Import Again

4- Por último, refrescamos y vemos la base de datos importada

The screenshot shows the MySQL Workbench Navigator interface. The left pane displays the 'SCHEMAS' tree. A blue box highlights the 'gestionacademica2' schema, which is expanded to show its contents: 'Tables' (alumnos, asignaturas, carreras), 'Views', 'Stored Procedures', and 'Functions'. A black box highlights the 'Tables' section of the schema. To the right of the interface, two annotations are present: 'resfrescamos el navegador de esquemas' (refresh the schema navigator) and 'base de datos importada' (imported database).

Realización:

La práctica consiste en la realización de los ejercicios y ejemplos planteados en el texto empleando Workbench. Por cada ejercicio debe obtener la evidencia como una copia de pantalla del resultado obtenido, acompañado del script que lo produjo.

Entrega:

Deberán entregar:

Un documento preferencia en PDF conformado con los títulos identificatorios de cada ítem realizado.

Criterios de Evaluación:

Se evaluará la correcta sintaxis y resultados y la correcta aplicación de los procedimientos realizados.