



DOCKER Y BD



8 DE ENERO DE 2026
ALEJANDRO SAINZ SAINZ
CLOUD COMPUTING

1EXPLICACIÓN DE LA PRÁCTICA.....	4
2COMENZANDO	4
3PREGUNTAS ADICIONALES	13

Ilustración 1 Port Forwarding	4
Ilustración 2 Docker pull.....	5
Ilustración 3 Descargado.....	5
Ilustración 4 Docker images	6
Ilustración 5 Creación del contenedor	6
Ilustración 6 Docker ps	7
Ilustración 7 Abrir la BD en la terminal	7
Ilustración 8 Ya casi se me habían olvidado estos comandos	7
Ilustración 9 Prueba de conexión después de los cambios.....	8
Ilustración 10 Conexión finalizada	9
Ilustración 11 Creación de la primera tabla.....	9
Ilustración 12 Compruebo que se ha creado la tabla.....	10
Ilustración 13 Como ya dije, los errores no son iguales	10
Ilustración 14 Comprobando las tablas	10
Ilustración 15 Inserts.....	11
Ilustración 16 Una consulta de prueba.....	11
Ilustración 17 Join múltiple.....	12
Ilustración 18 Ahora a workbench	12
Ilustración 19 Aquí por lo menos es a color.....	13
Ilustración 20 Reverse Engineering.....	14
Ilustración 21 Resultado final	15

1 EXPLICACIÓN DE LA PRÁCTICA

Desde el principio. Para mi esta práctica se ha complicado algo. No se si para los demás, dado que ellos parece que pueden usar bien Docker y WSL 2 en clase, habrán tenido menos problemas, pero a mi me han saltado unos cuantos, que iré explicando poco a poco. Algunos por el hecho de hacer yo estos ejercicios con la máquina virtual de debian y otros por la forma de hacer yo la cosa y calentarme un poco. Lo de siempre. Al final, mirando documentación en Docker docs y otras veces con ayuda del amigo invisible cuando ya no imaginaba porque las cosas no funcionaban, ha salido todo. Iré comentando las cosas que encontré yo por el camino.

2 COMENZANDO

Lo primero que debemos de hacer es preparar todo para poder crear un contenedor de mysql en Docker. Estuve pensando en probar un contenedor de postgre pero, como ya de java y springboot, me he acostumbrado a preparar la conectividad con BD de mysql. Ese es el motivo por el que elegí principalmente mysql. El otro motivo es que, al usarlo ya por defecto para clase, como ya tenemos el Workbench preparado pensé que sería más sencillo.

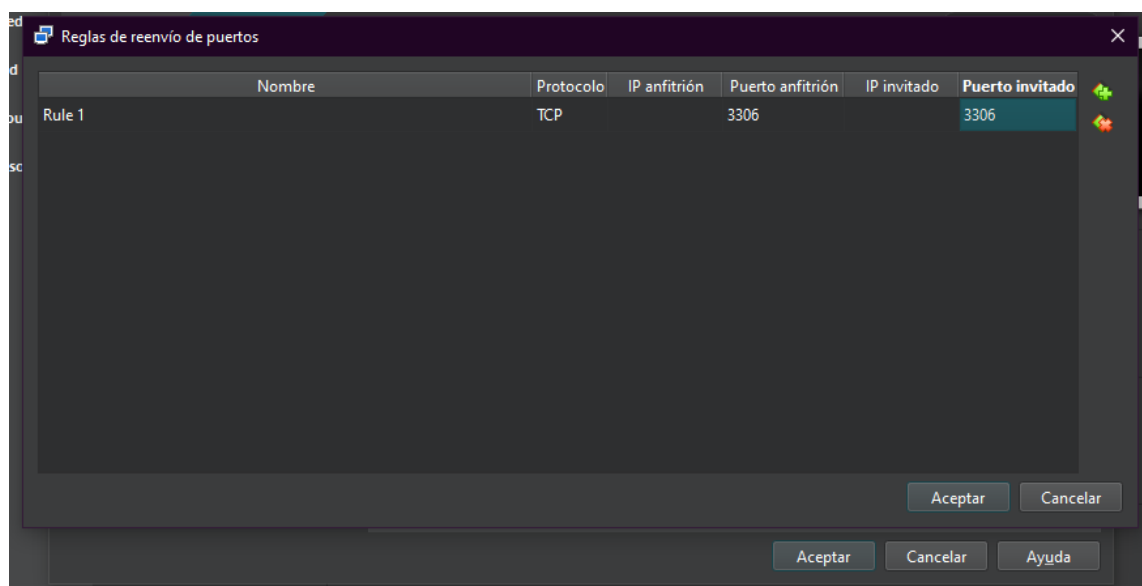
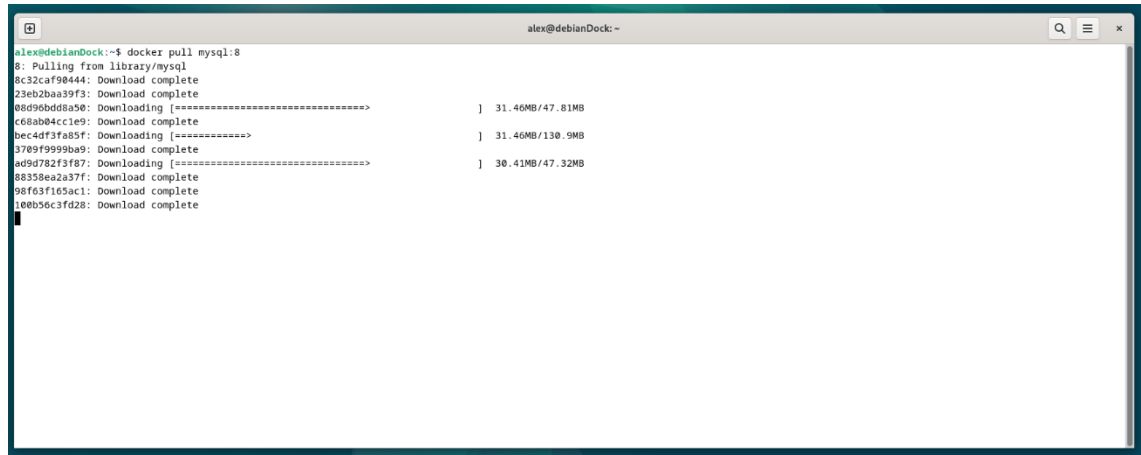


Ilustración 1 Port Forwarding

Lo primero es esto. Yo no sé si los demás habrán tenido que hacerlo así. Como yo lo hago en una MV, si dejo la conexión simplemente como NAT(que esto me dio casi todos los problemas de conectividad) la máquina tiene salida a internet, pero no se puede acceder a ella ni al contenedor de forma sencilla. Aunque esta captura la tenga aquí la primera, esto no es lo

primero que hice y de ahí quebraderos de cabeza para ver la solución. Primer contacto con agentes de internet.

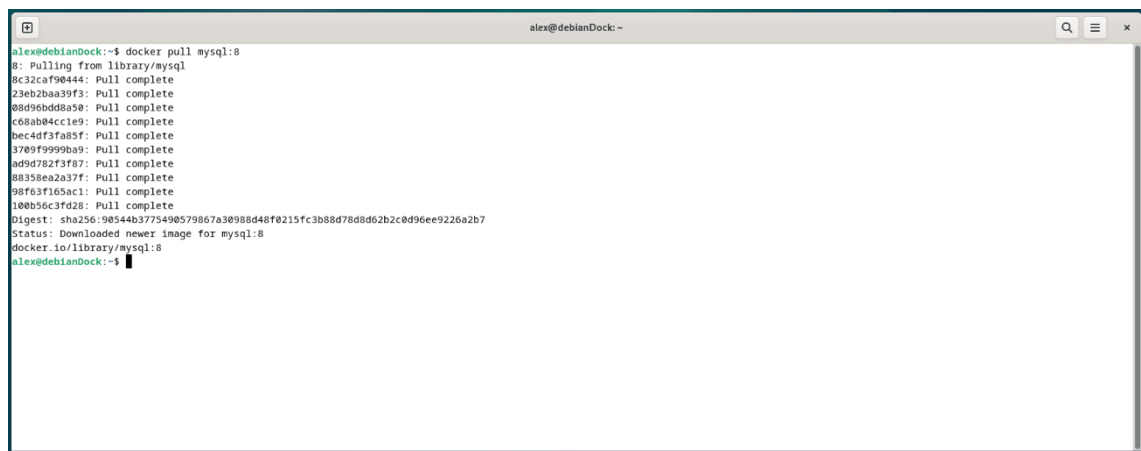
Ahora ya voy contando lo que sería el flujo normal que fui haciendo de primeras. Pero había que aclarar esto para entender algunos de mis problemas.



```
alex@debianDock:~$ docker pull mysql:8
8: Pulling from library/mysql
8c32caf90444: Download complete
23eb2baa39f3: Download complete
08d96bdd8a50: Downloading [=====>] 31.46MB/47.81MB
c68ab04cc1e9: Download complete
bec4df3fa85f: Downloading [=====>] 31.46MB/130.9MB
3789f999ba9: Download complete
ad9d782f3f87: Downloading [=====>] 30.41MB/47.32MB
88358ea2a37f: Download complete
98f63f165ac1: Download complete
100b56c3fd28: Download complete
```

Ilustración 2 Docker pull

Con este comando, que yo no había visto todavía, ya que siempre creamos el contenedor del tirón, lo que se hace es descargar la imagen que necesitamos. Como vemos arriba, típico procedimiento de la terminal de Linux.



```
alex@debianDock:~$ docker pull mysql:8
8: Pulling from library/mysql
8c32caf90444: Pull complete
23eb2baa39f3: Pull complete
08d96bdd8a50: Pull complete
c68ab04cc1e9: Pull complete
bec4df3fa85f: Pull complete
3789f999ba9: Pull complete
ad9d782f3f87: Pull complete
88358ea2a37f: Pull complete
98f63f165ac1: Pull complete
100b56c3fd28: Pull complete
Digest: sha256:98544b3775490579867a30988d48f0215fc3b88d78d8d62b2c0d96ee9226a2b7
Status: Downloaded newer image for mysql:8
docker.io/library/mysql:8
alex@debianDock:~$
```

Ilustración 3 Descargado

Tras un ratito, no mucho, queda descargada la imagen.

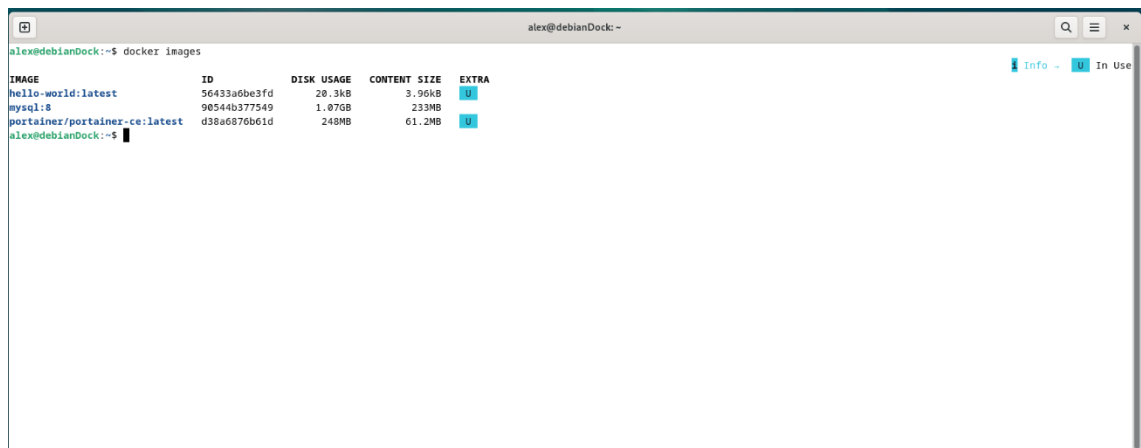


Ilustración 4 Docker images

Con este otro comando, Docker images, podemos ver las imágenes que tenemos descargadas en nuestro equipo. No sé qué significa la columna extra. Aunque he creado otros contenedores de nginx, esa imagen no aparece porque no la he descargado nunca.

```
alex@debianDock:~$ docker run -d --name bd_ud4 -e MYSQL_ROOT_PASSWORD=alex -e MYSQL_DATABASE=instituto -p 3306:3306 mysql:8
48d5276748d2634a7e3a0b904ddcfff7b524790508786670ae54d4a5df93140b
alex@debianDock:~$
```

Ilustración 5 Creación del contenedor

Esto es prácticamente copia de lo que ya hemos hecho anteriormente con otros contenedores, aunque hay una serie de cambios.

Docker run -d para crear el contenedor. -d para que se ejecute en segundo plano.

--name bd_ud4 para darle un nombre a nuestro contenedor.

Parámetro -e. Este parámetro sirve para pasar variables y valores al contenedor en su creación. Los ejemplos que tenemos son para pasarle el valor de la password del root y para darle un nombre a la BD que vamos a crear.

-p y valores. Con menos p exponemos puertos al exterior, en este caso el 3306 que es el que usa mysql.

Finalmente, mysql:8 para indicar a partir de que imagen se va a crear el contenedor.

```
alex@debianDock:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
48d5276748d2   mysql:8       "docker-entrypoint.s..." 36 seconds ago Up 35 seconds 0.0.0.0:3306->3306/tcp, [::]:3306->3306/tcp, 33060/tcp   bd_ud4
5df077eaf101   portainer/portainer-ce "/portainer"           7 weeks ago   Up 9 minutes  8000/tcp, 9443/tcp, 0.0.0.0:9000->9000/tcp, [::]:9000->9000/tcp   portainer
alex@debianDock:~$
```

Ilustración 6 Docker ps

Con Docker ps compruebo que el contenedor está creado y levantado.

```
alex@debianDock:~$ docker exec -it bd_ud4 mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.4.7 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Ilustración 7 Abrir la BD en la terminal

Después busqué el comando para conectarme a la BD desde la terminal. Primera línea de la imagen. Te pide el password y te da la bienvenida. Ahora hay que comprobar que todo está como debiese.

```
OWNERS.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database                |
+-----+
| information_schema      |
| instituto               |
| mysql                  |
| performance_schema     |
| sys                    |
+-----+
5 rows in set (0.02 sec)

mysql>
```

Ilustración 8 Ya casi se me habían olvidado estos comandos

Con show databases mostramos las BD que tenemos creadas actualmente. Muy importante no olvidar el ; al final, que si no da fallo y no ejecuta nada.

Ahora, como dio problemas como dije al principio, voy a explicar la segunda parte de la conectividad.

Creo una conexión normal en Workbench. El problema es el siguiente. Aunque yo había creado la regla de port forwarding en la máquina, todavía surge otro pequeño escollo. Por defecto las conexiones de Workbench, al tener instalado el server en nuestro ordenador, las hace hacia la ip 127.0.0.1 y al puerto 3306. El problema que surgió aquí, fue que el contenedor también estaba levantado con el puerto 3306 expuesto. Así que workbench no diferenciaba y se conectaba al servidor de la máquina física.

Soluciones que se me ocurrieron, volver a crear el contenedor. Pero con otro puerto distinto no va bien o puede dar fallos. Solución final, cambiar el puerto de la regla de portforwarding del guest a 3307 y poner ese mismo puerto en la conexión de workbench.

Yo en lo de redes no estoy muy suelto, así que, para que quiero más.

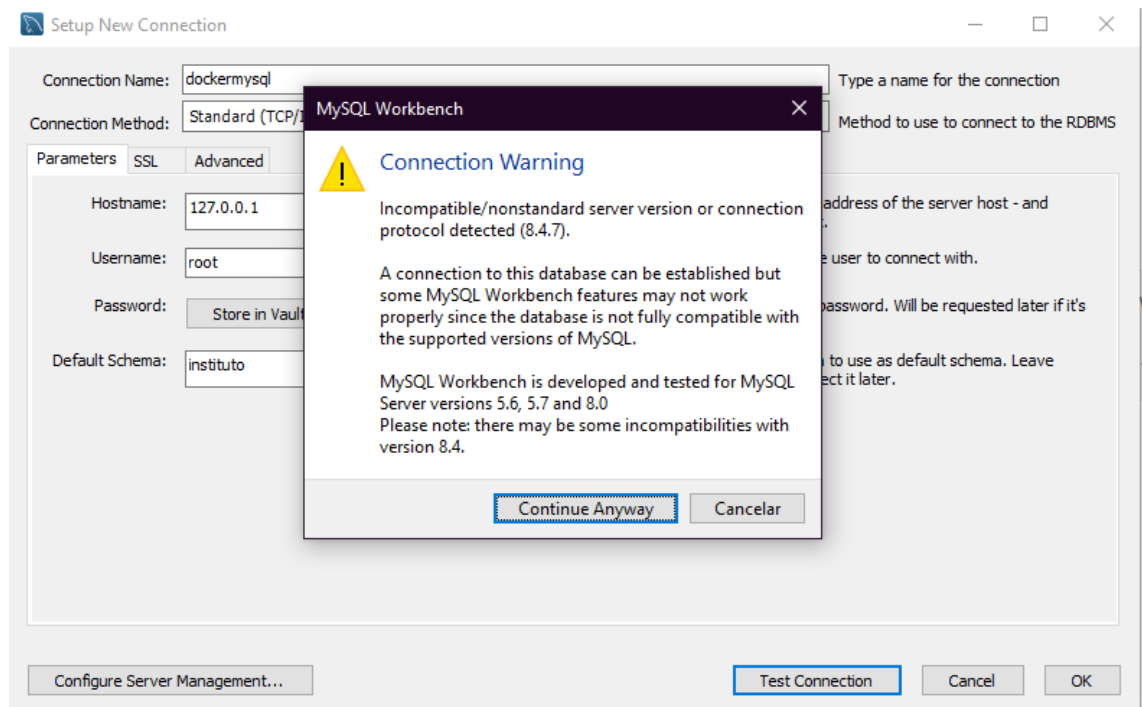


Ilustración 9 Prueba de conexión después de los cambios.

Una vez cambias los puertos, ya funciona la prueba de conexión. Salta este warning. Busqué el motivo. Se debe a que la versión de Docker no tiene los mismo plugins ni elementos con los que viene workbench y algunas cosas pueden no funcionar. Pero en principio, lo que vamos a hacer nosotros, no sufre de ninguno de estos problemas.

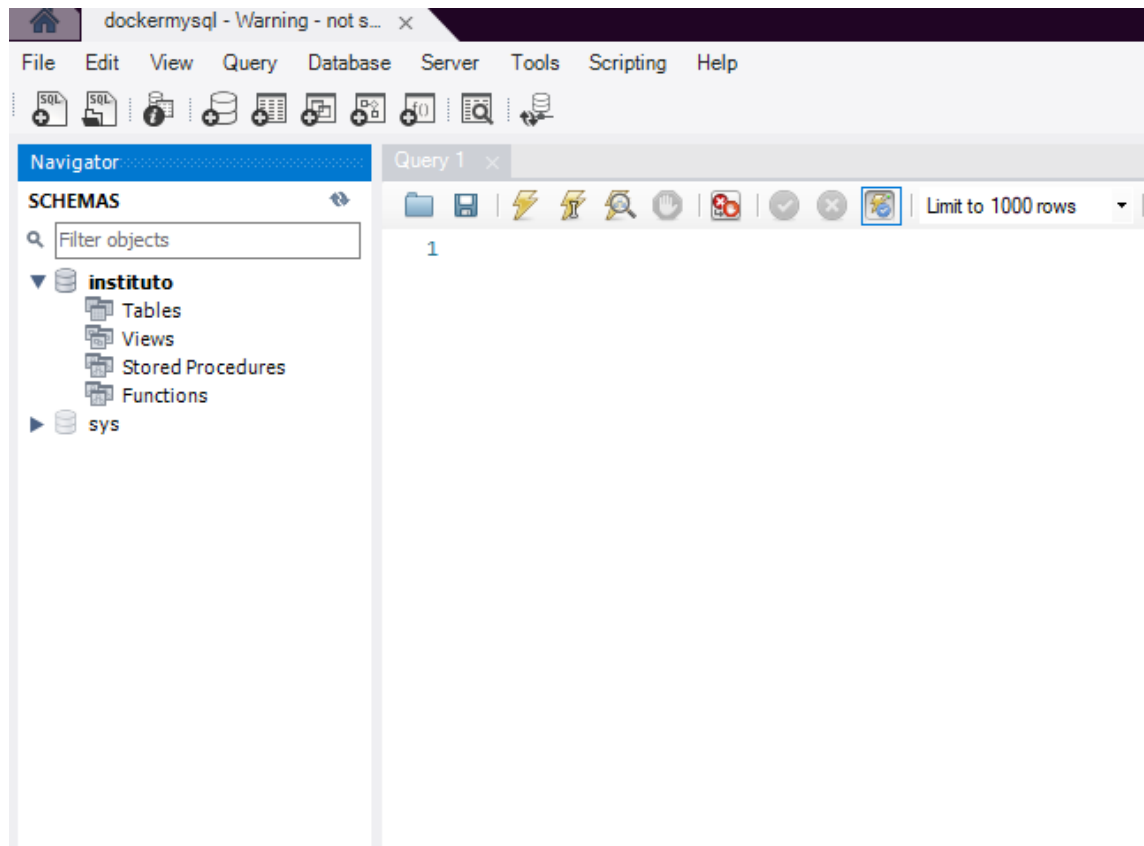


Ilustración 10 Conexión finalizada

Después de terminar la prueba tratamos de conectarnos con todas las de la ley. Nos salta el mismo warning que en la prueba, pero se conecta, y vemos la BD de la que disponemos. Todavía con todo vacío, es decir, sin tablas.

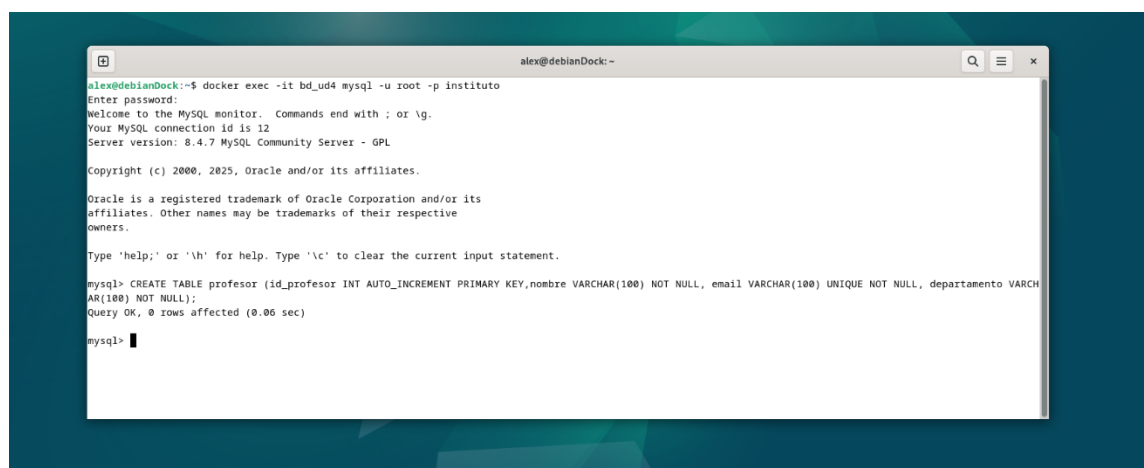


Ilustración 11 Creación de la primera tabla

Como yo soy yo, y no lo puedo evitar, en vez de crear las tablas en el workbench decidí crearlas desde la terminal. Con el lenguaje de siempre, pero yo pensé que tendría que escribirlo todo seguido. Eso sí, aquí los fallos no son iguales que en workbench. No hay colores que te indiquen nada, no hay marcas de error y, me equivocaba pensando que no podía incluir saltos de línea para el comando. En la siguiente captura lo explico.

```

alex@debianDock: ~
Your MySQL connection id is 12
Server version: 8.4.7 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE TABLE profesor (id_profesor INT AUTO_INCREMENT PRIMARY KEY,nombre VARCHAR(100) NOT NULL, email VARCHAR(100) UNIQUE NOT NULL, departamento VARCH
AR(100) NOT NULL);
Query OK, 0 rows affected (0.06 sec)

mysql> show tables;
+-----+
| Tables_in_instituto |
+-----+
| profesor            |
+-----+
1 row in set (0.01 sec)

mysql>

```

Ilustración 12 Compruebo que se ha creado la tabla

Después de comprobar que la tabla se había creado, busqué el tema de como escribir los comandos, para ver si podía ser algo más legible. Y si, se puede usar el salto de línea hasta el momento en que incluyas el carácter ;.

```

mysql> CREATE TABLE alumno_modulo (
-> id_alumno INT NOT NULL,
-> id_modulo INT NOT NULL,
-> nota DECIMAL(4,2),
-> convocatoria VARCHAR(20),
-> PRIMARY KEY (id_alumno, id_modulo),
-> CONSTRAINT fk_am_alumno
-> FOREIGN KEY (id_alumno)
-> REFERENCES alumno(id_alumno)
-> CONSTRAINT fk_am_modulo
-> FOREIGN KEY (id_modulo)
-> REFERENCES modulo(id_modulo)
-> );
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'CONSTRAINT fk_am_modulo
FOREIGN KEY (id_modulo)
REFERENCES modul' at line 10
mysql>

```

Ilustración 13 Como ya dije, los errores no son iguales

Como vemos en la captura, cuando uso un salto de línea, si no he usado “;”, comienza la línea con un símbolo de flecha. Y como vemos también la imagen, te indica los fallos, pero yo hasta que no ejecuto el comando no tengo ningún indicativo de cual es. En este caso es que me faltaba una “;” después de references alumno(id_alumno). Lo bueno, que si pulso la flecha arriba en la terminal de Linux me vuelve a escribir el último comando. Añado la “;” y listo.

```

mysql> CREATE TABLE alumno_modulo (
KEY (id_modulo) REFERENCES modulo(id_modulo) );
Query OK, 0 rows affected (0.06 sec)

mysql> show tables;
+-----+
| Tables_in_instituto |
+-----+
| alumno              |
| alumno_modulo       |
| modulo              |
| profesor            |
+-----+
4 rows in set (0.00 sec)

mysql>

```

Ilustración 14 Comprobando las tablas

Una vez he insertado todos los comandos, uso show tables para comprobar que de verdad se han creado todas las tablas. Arriba el resultado.

```
mysql> -- INSERTS en tabla profesor
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO profesor (nombre, email, departamento) VALUES
-> ('Ana Lopez', 'alopez@example.com', 'Matemáticas'),
-> ('Juan Perez', 'jperez@example.com', 'Informática');
Query OK, 2 rows affected (0.04 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql>
mysql> -- INSERTS en tabla modulo
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO modulo (nombre, curso, id_profesor) VALUES
-> ('Álgebra', '1 ESO', 1),
-> ('Programación', '2 Bachillerato', 2);
Query OK, 2 rows affected (0.02 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql>
mysql> -- INSERTS en tabla alumno
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO alumno (nombre, email, fecha_nacimiento) VALUES
-> ('Mara Garcia', 'mgarcia@example.com', '2005-04-12'),
-> ('Carlos Fernandez', 'cfernandez@example.com', '2004-11-23');
Query OK, 2 rows affected (0.03 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql>
mysql> -- INSERTS en tabla intermedia alumno_modulo
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO alumno_modulo (id_alumno, id_modulo, nota, convocatoria) VALUES
```

Ilustración 15 Inserts

Lo que también descubrí, que no estaba seguro de si se podía, es que puedo añadir líneas de comentarios también por la terminal. Así por lo menos se puede ver en la captura donde hago Inserts. Otra cosa que descubrí, supongo que por la configuración del lenguaje es que hay algún problema con los acentos, por eso queda “lgebra” en vez de “Álgebra”.

```
mysql> INSERT INTO alumno (nombre, email, fecha_nacimiento) VALUES
-> ('Mara Garcia', 'mgarcia@example.com', '2005-04-12'),
-> ('Carlos Fernandez', 'cfernandez@example.com', '2004-11-23');
Query OK, 2 rows affected (0.03 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql>
mysql> -- INSERTS en tabla intermedia alumno_modulo
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO alumno_modulo (id_alumno, id_modulo, nota, convocatoria) VALUES
-> (1, 1, 8.5, 'Junio'),
-> (1, 2, 7.0, 'Junio'),
-> (2, 1, 9.0, 'Junio'),
-> (2, 2, 8.0, 'Junio');
Query OK, 4 rows affected (0.02 sec)
Records: 4 Duplicates: 0 Warnings: 0

mysql> select * from alumno_modulo;
+-----+
| id_alumno | id_modulo | nota | convocatoria |
+-----+
| 1 | 1 | 8.50 | Junio |
| 1 | 2 | 7.00 | Junio |
| 2 | 1 | 9.00 | Junio |
| 2 | 2 | 8.00 | Junio |
+-----+
4 rows in set (0.01 sec)

mysql>
```

Ilustración 16 Una consulta de prueba

Ejecuto un `SELECT * FROM alumno_modulo` para comprobar que todo ha ido bien. Vemos aquí el resultado.

Ahora voy a ejecutar una consulta con joins para poder así ver todos los datos representados en pantalla.

```
mysql> SELECT
->   a.nombre AS Alumno,
->   a.email AS Email,
->   m.nombre AS Modulo,
->   m.curso AS Curso,
->   p.nombre AS Profesor,
->   am.nota AS Nota,
->   am.convocatoria AS Convocatoria
-> FROM alumno_modulo am
-> JOIN alumno a ON am.id_alumno = a.id_alumno
-> JOIN modulo m ON am.id_modulo = m.id_modulo
-> JOIN profesor p ON m.id_profesor = p.id_profesor
-> ORDER BY a.id_alumno, m.id_modulo;
```

Alumno	Email	Modulo	Curso	Profesor	Nota	Convocatoria
Mara Garca	mgarcia@example.com	lgebra	1 ESO	Ana Lpez	8.50	Junio
Mara Garca	mgarcia@example.com	Programacin	2 Bachillerato	Juan Prez	7.00	Junio
Carlos Fernndez	cfernandez@example.com	lgebra	1 ESO	Ana Lpez	9.00	Junio
Carlos Fernndez	cfernandez@example.com	Programacin	2 Bachillerato	Juan Prez	8.00	Junio

```
4 rows in set (0.00 sec)

mysql> █
```

Ilustración 17 Join múltiple

Como vemos en la imagen, ejecuto una consulta un poco más elaborada, pero sencilla. No tiene campos agregados ni nada por el estilo. La hago para que se tengan que mostrar datos de todas las tablas y comprobar que las PK y las FK están como debe de ser.

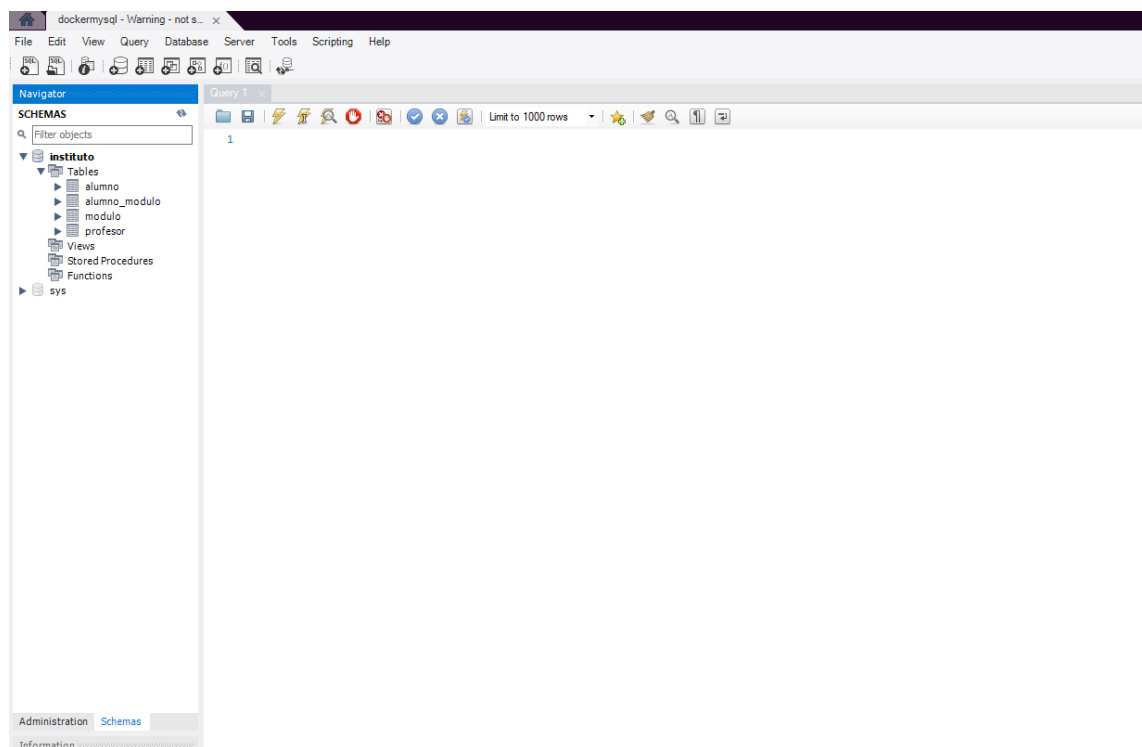


Ilustración 18 Ahora a workbench

Vuelvo a abrir la conexión desde workbench. Al entrar esta vez ya vemos las tablas que tiene nuestra BD. Ahora voy a hacer la misma consulta con joins pero aquí.

The screenshot shows a SQL query editor with a query named 'Query 1'. The query is as follows:

```

1 • use instituto;
2
3 • SELECT
4     a.nombre AS Alumno,
5     a.email AS Email,
6     m.nombre AS Modulo,
7     m.curso AS Curso,
8     p.nombre AS Profesor,
9     am.nota AS Nota,
10    am.convocatoria AS Convocatoria
11 FROM alumno_modulo am
12 JOIN alumno a ON am.id_alumno = a.id_alumno
13 JOIN modulo m ON am.id_modulo = m.id_modulo
14 JOIN profesor p ON m.id_profesor = p.id_profesor
15 ORDER BY a.id_alumno, m.id_modulo;
16

```

The result grid shows the following data:

Alumno	Email	Modulo	Curso	Profesor	Nota	Convocatoria
Mara Garca	mgarcia@example.com	lgebra	1 ESO	Ana Lpez	8.50	Junio
Mara Garca	mgarcia@example.com	Programacin	2 Bachillerato	Juan Prez	7.00	Junio
Carlos Fernndez	cfernandez@example.com	lgebra	1 ESO	Ana Lpez	9.00	Junio
Carlos Fernndez	cfernandez@example.com	Programacin	2 Bachillerato	Juan Prez	8.00	Junio

Ilustración 19 Aquí por lo menos es a color

Muy importante y que no se olvide, antes de la consulta, escribir use instituto. Si no, fallo. Como podemos ver el resultado es el mismo, pero la claridad de la presentación es otra totalmente distinta.

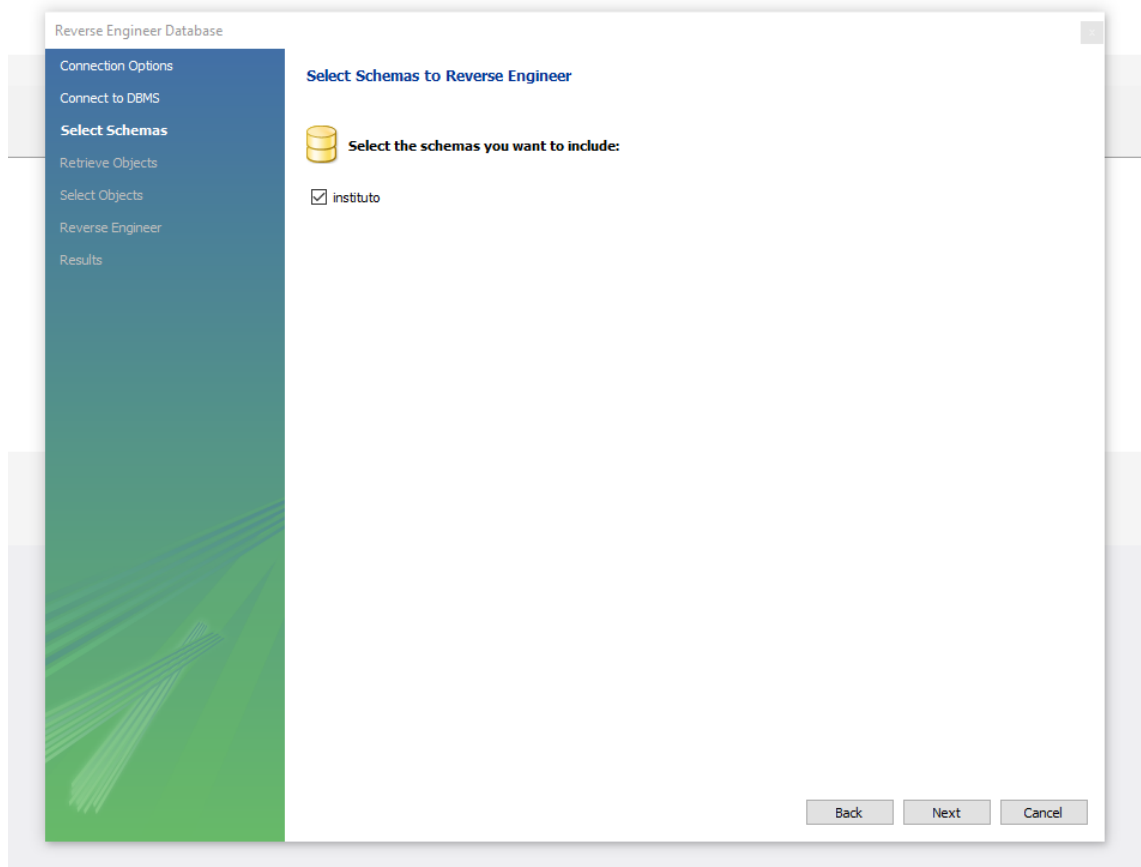


Ilustración 20 Reverse Engineering

Se me olvidó hacer una captura de la parte anterior, pero, lo que pasó fue que, si yo uso los parámetros por defecto, con el puerto 3306, solo me deja hacer RE de los schemas que tengo en mi máquina física. Una vez cambio el puerto al 3307 ya aparece la BD instituto, y solo esa BD, pues es la única que existe en el contenedor.

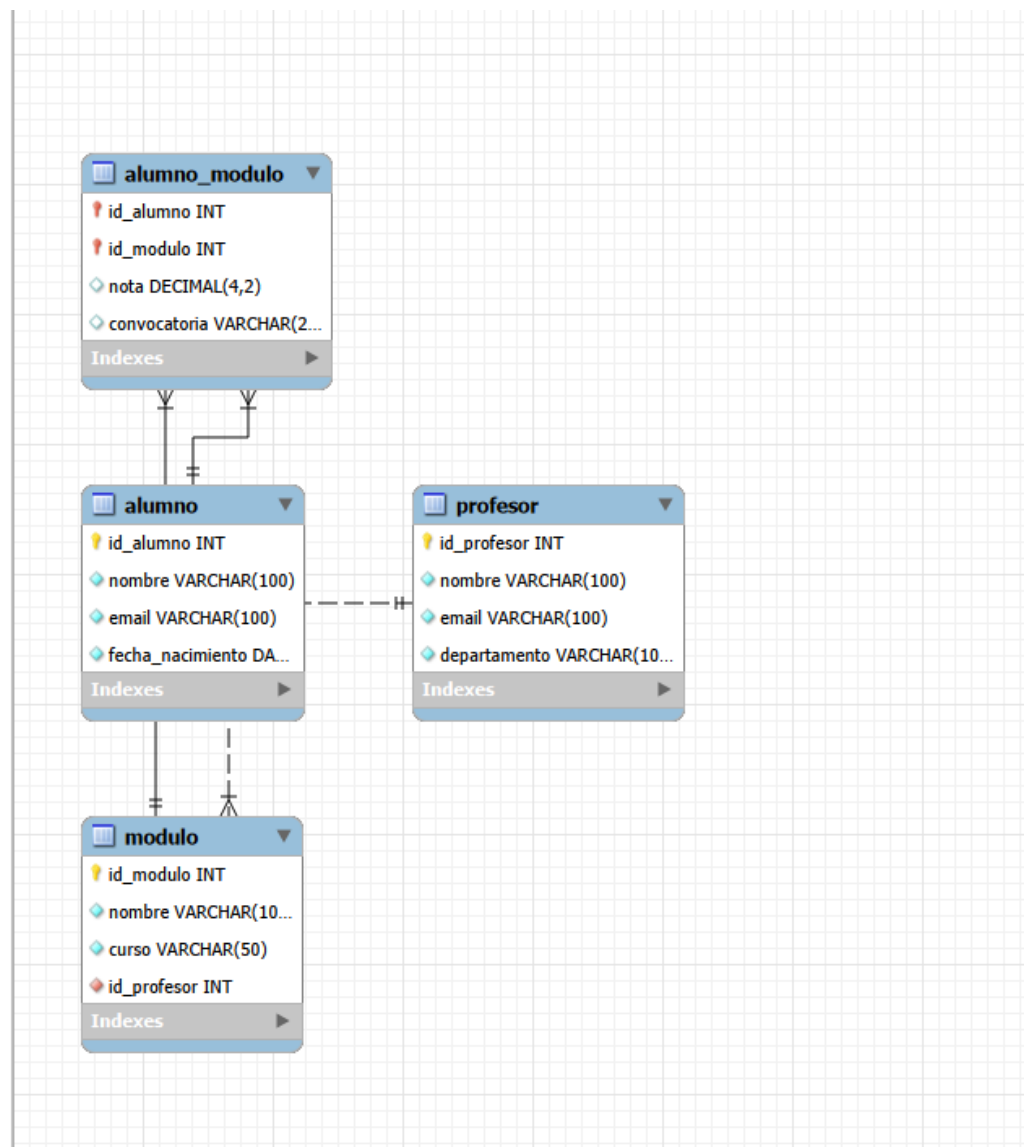


Ilustración 21 Resultado final

Aquí ya, el resultado final de la actividad.

3 DETALLES ADICIONALES

¿Tiene persistencia nuestro contenedor?

Respuesta corta, no. Para ello nos vamos al siguiente enlace.

https://docs.docker.com/get-started/docker-concepts/running-containers/persisting-container-data/?utm_source=chatgpt.com

Dentro de la documentación de Docker ya hay una parte donde nos habla de la persistencia. Nos dice que esta existe mientras el contenedor está activo, si no lo he entendido mal. Para evitar esto debemos de crear volúmenes, como hacíamos con el contenedor de nginx. Así, cada vez que creamos el contenedor o lo arranquemos, se cargarán los datos desde allí. Eso sí, esta parte no la he probado.