# DOCKER Y BD

8 DE ENERO DE 2026

ALEJANDRO SAINZ SAINZ
CLOUD COMPUTING

# 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.

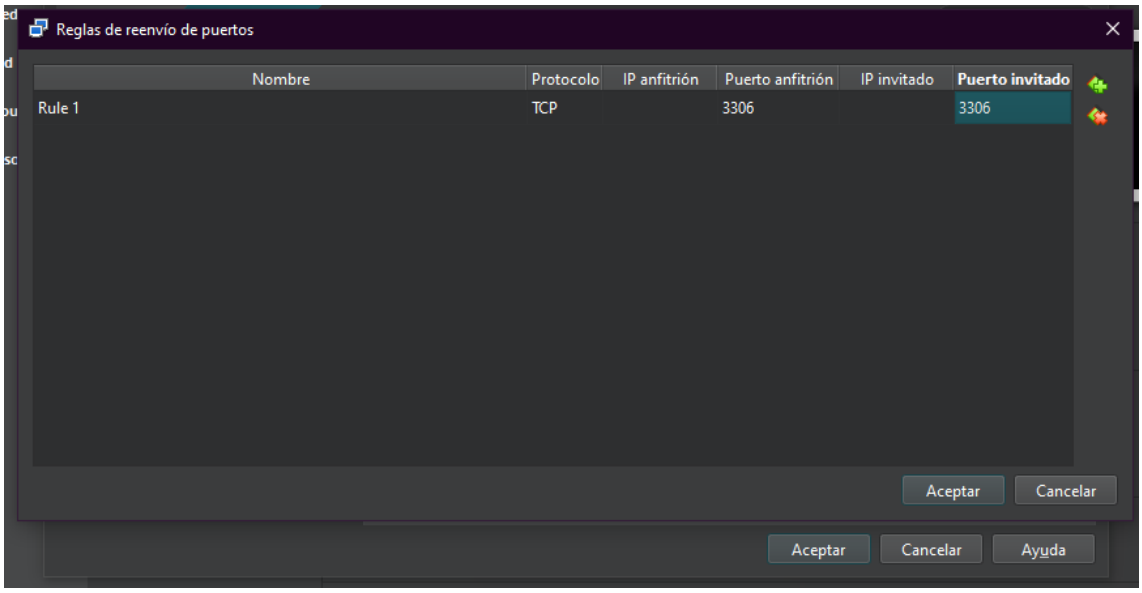


*Ilustración 1 Port Forwarding*

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.

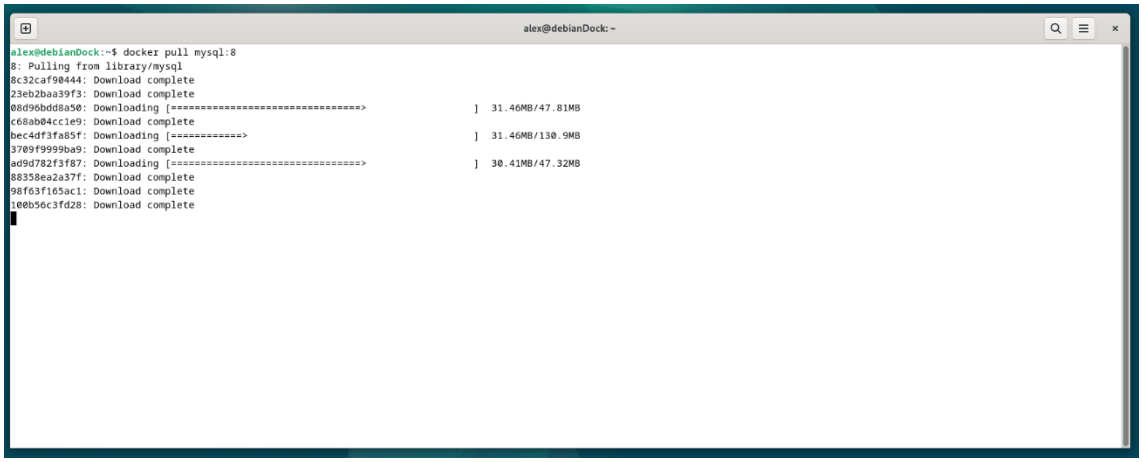


*Ilustración 2 Docker pull*

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.


*Ilustración 3 Descargado*

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

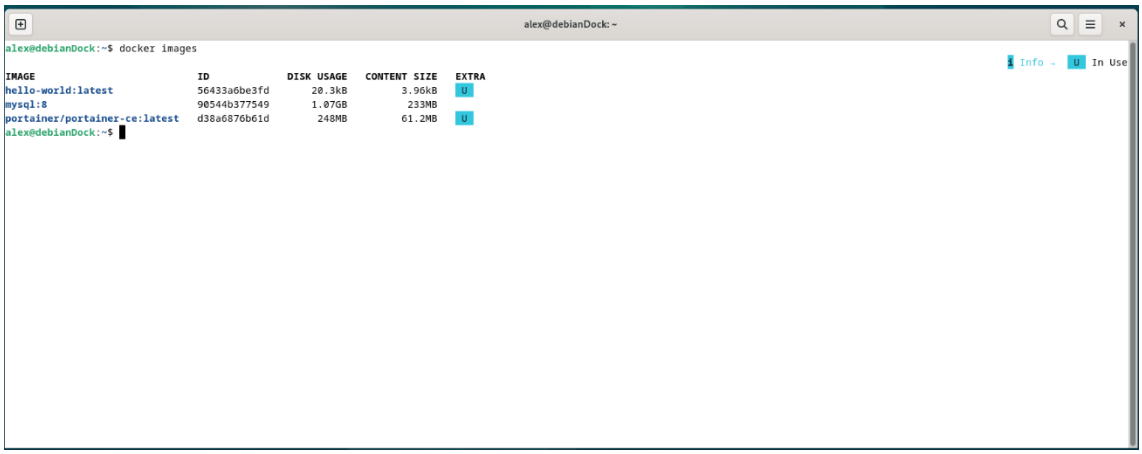


*Ilustración 4 Docker images*

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
48d5276748d2634a7e3a0b904ddcfff7b5247905087866670ae54d4a5df93140b
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.

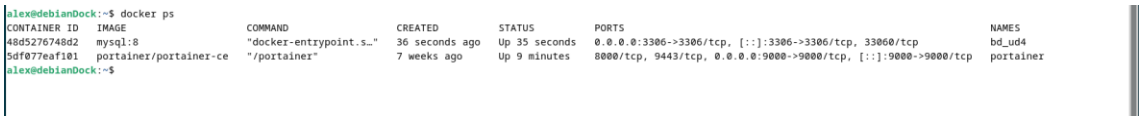


*Ilustración 6 Docker ps*

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

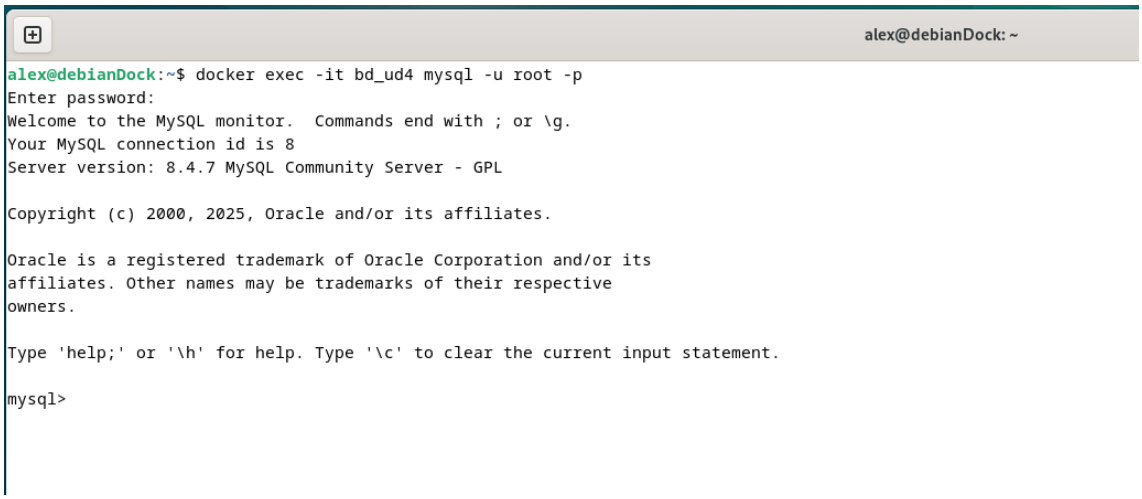


*Ilustración 7 Abrir la BD en la terminal*

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.

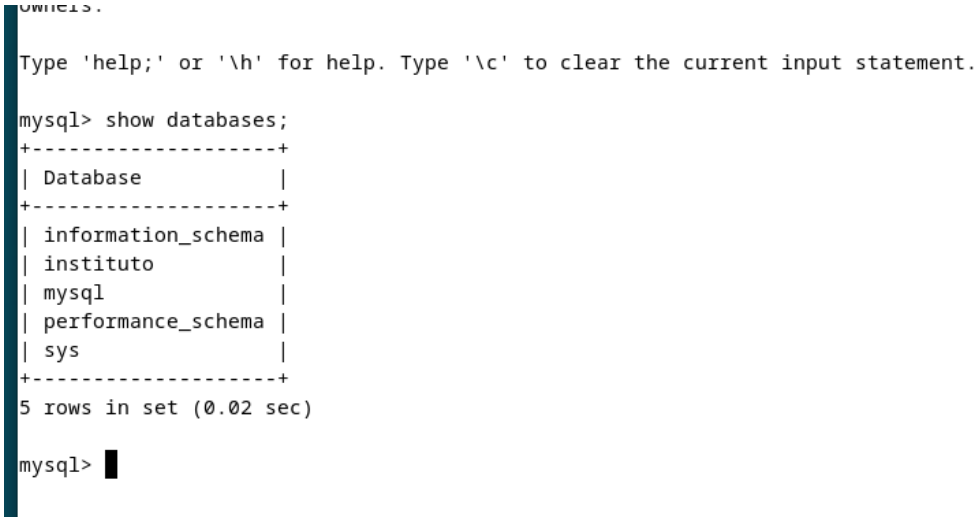


*Ilustración 8 Ya casi se me habían olvidado estos comandos*

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.

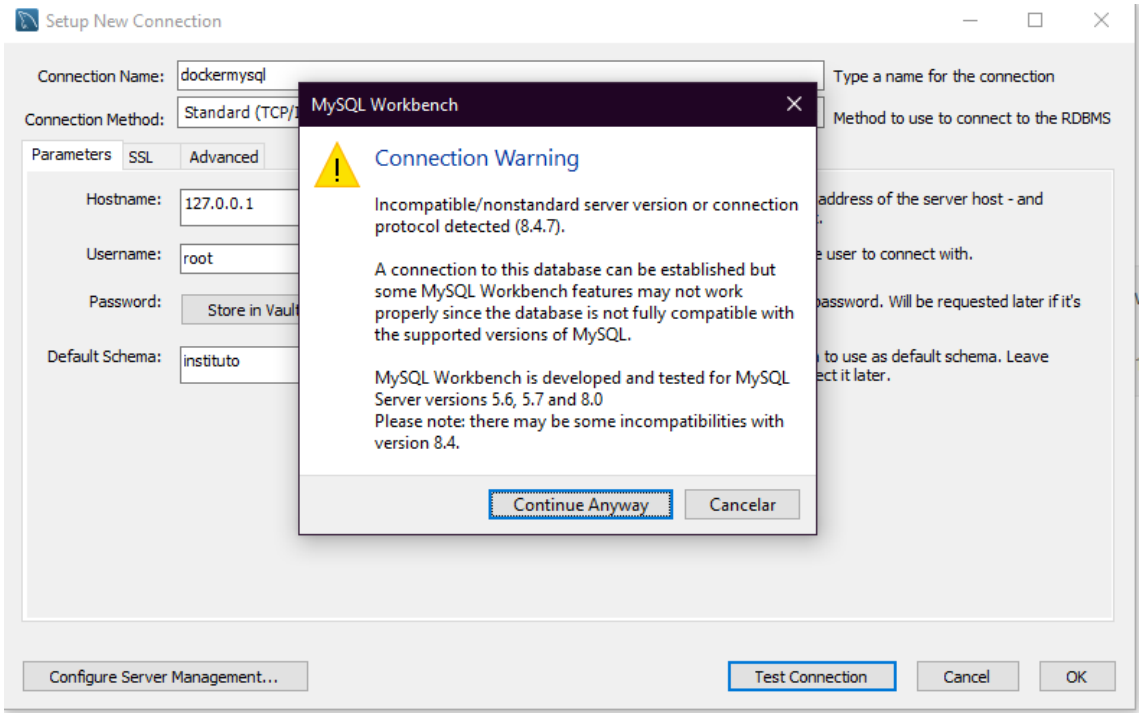


*Ilustración 9 Prueba de conexión después de los cambios.*

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.

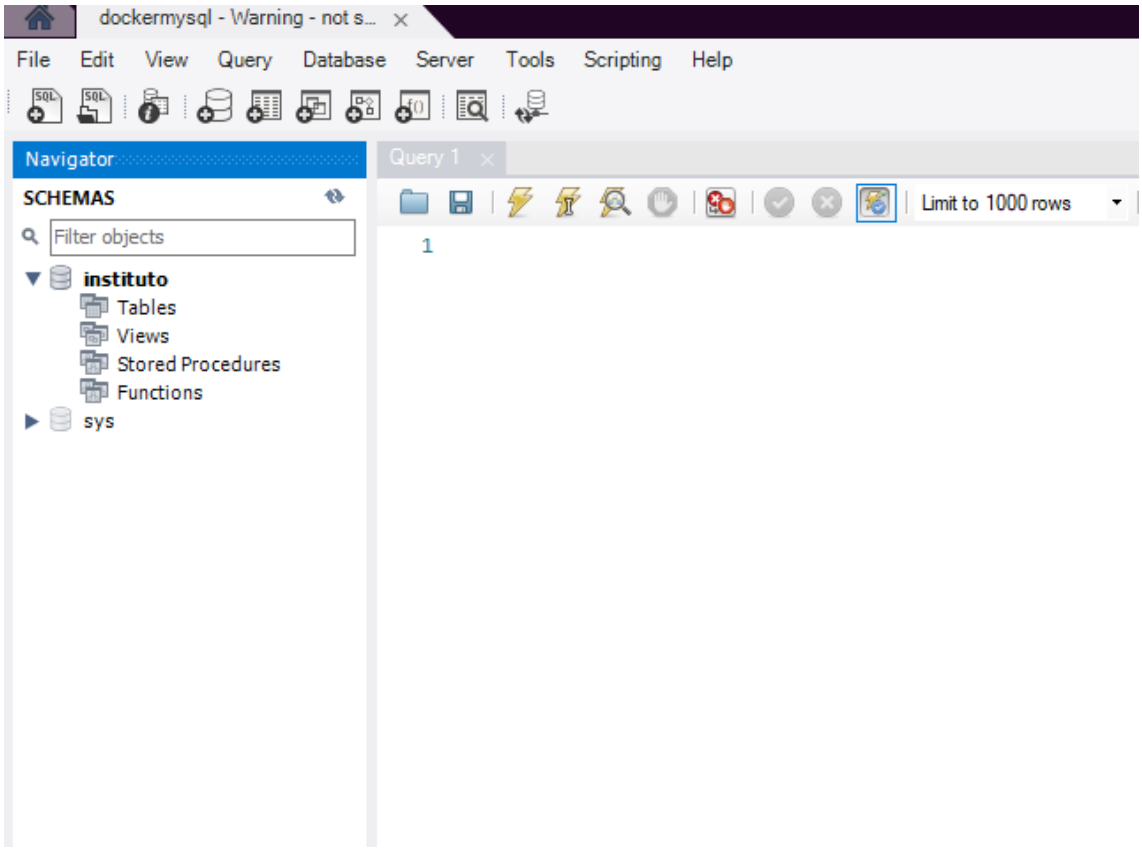


*Ilustración 10 Conexión finalizada*

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.

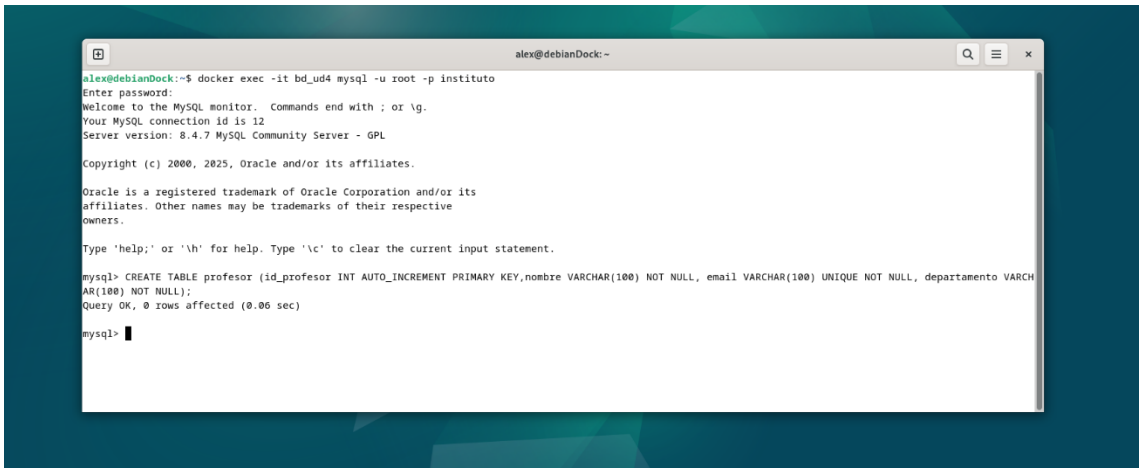


*Ilustración 11 Creación de la primera tabla*

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.

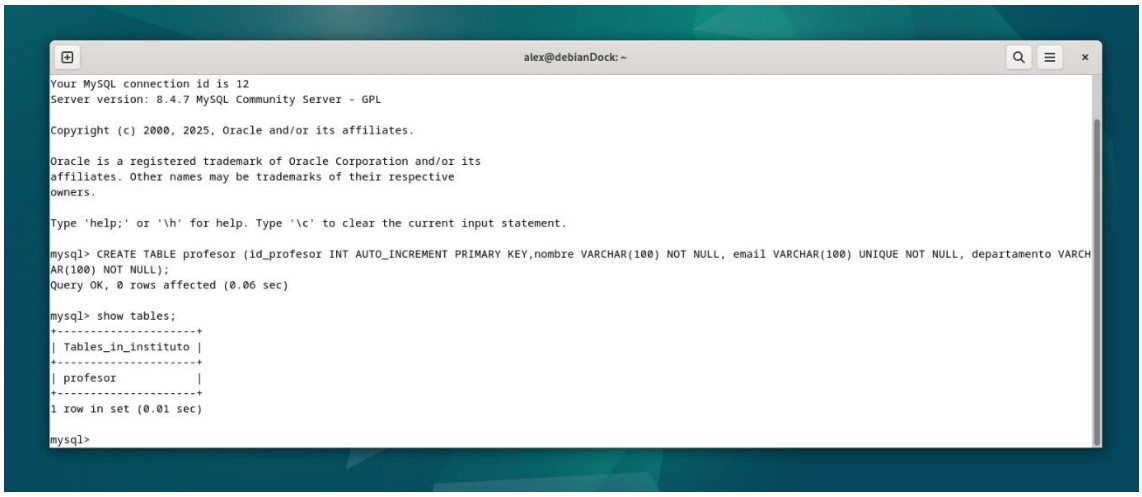


*Ilustración 12 Compruebo que se ha creado la tabla*

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 ;.

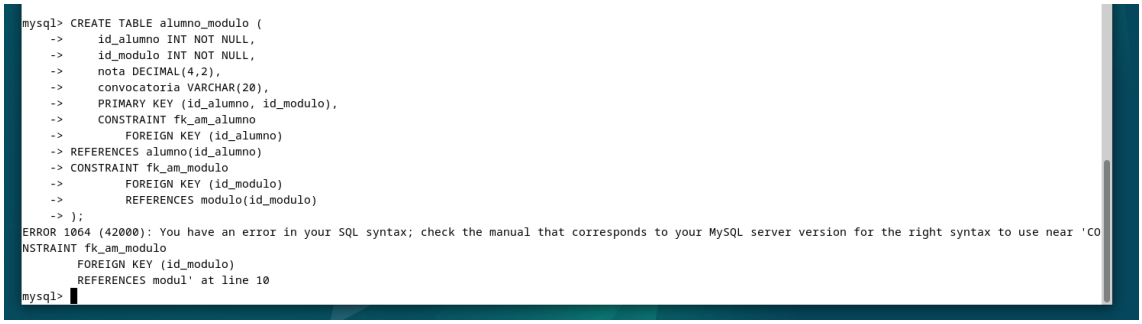


*Ilustración 13 Como ya dije, los errores no son iguales*

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.

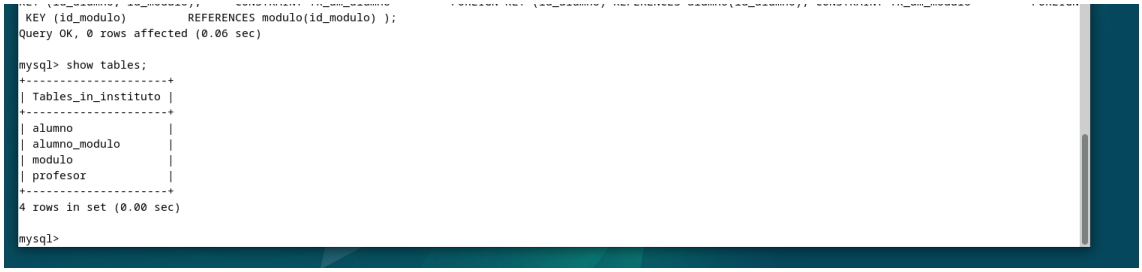


*Ilustración 14 Comprobando las tablas*

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

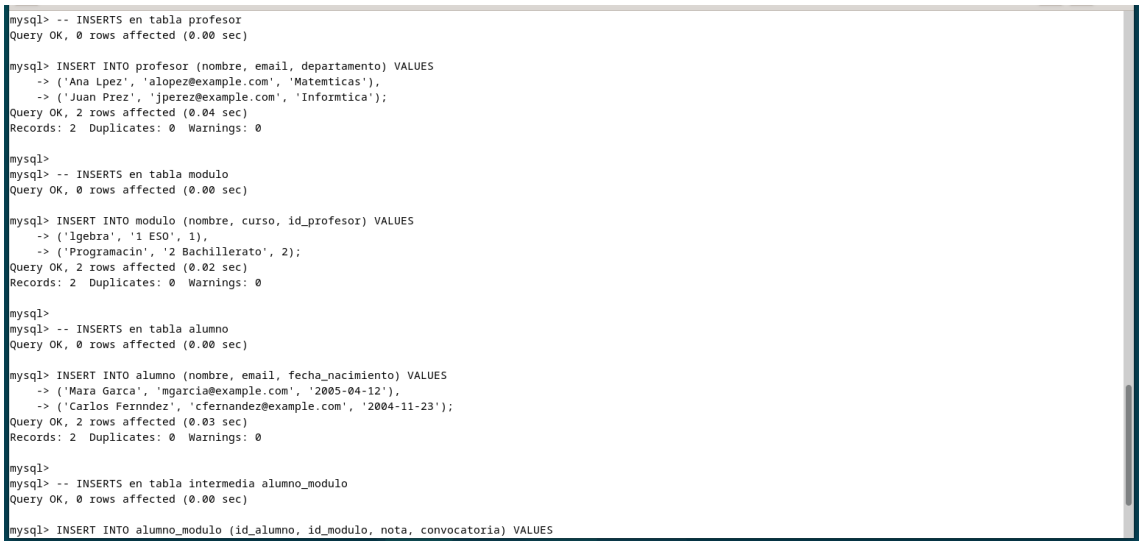


*Ilustración 15 Inserts*

Lo que también descubrí, que no estaba seguro de si se podía, es que puedo añadir lineas 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”.

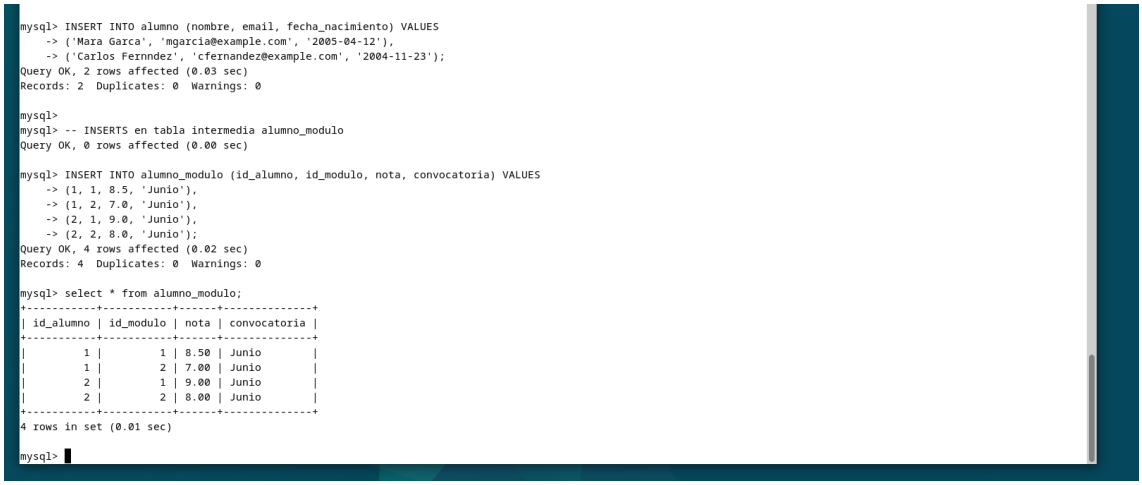


*Ilustración 16 Una consulta de prueba*

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.

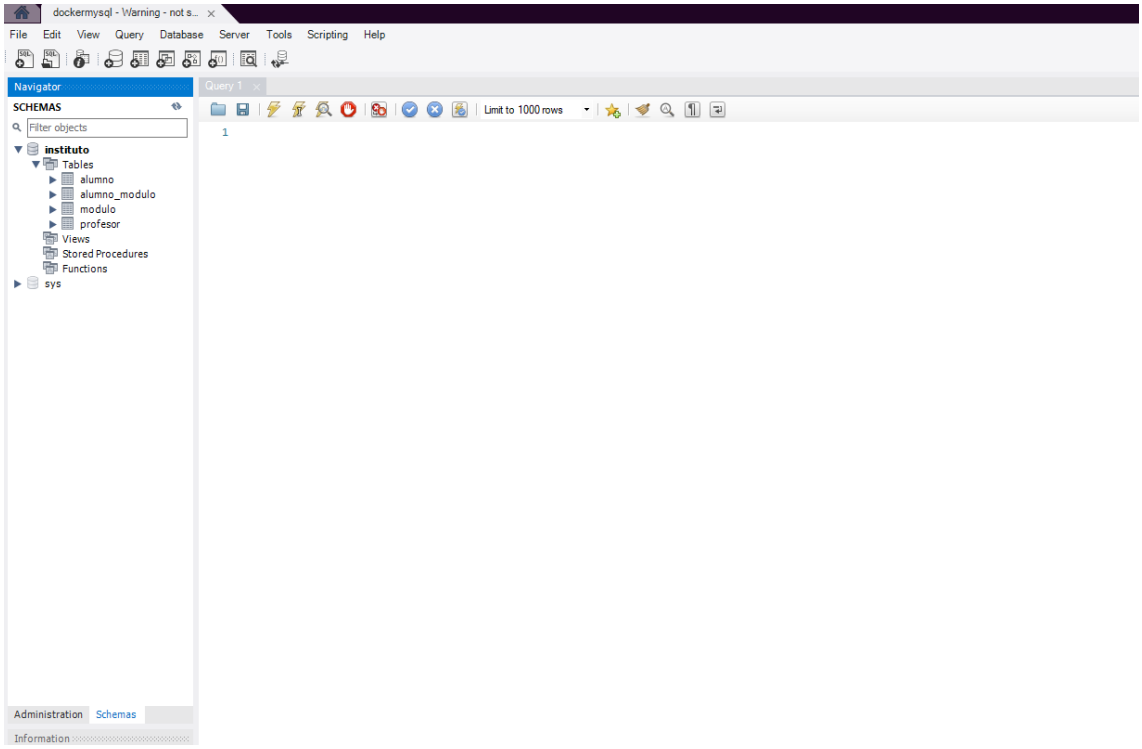


*Ilustración 18 Ahora a workbench*

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.

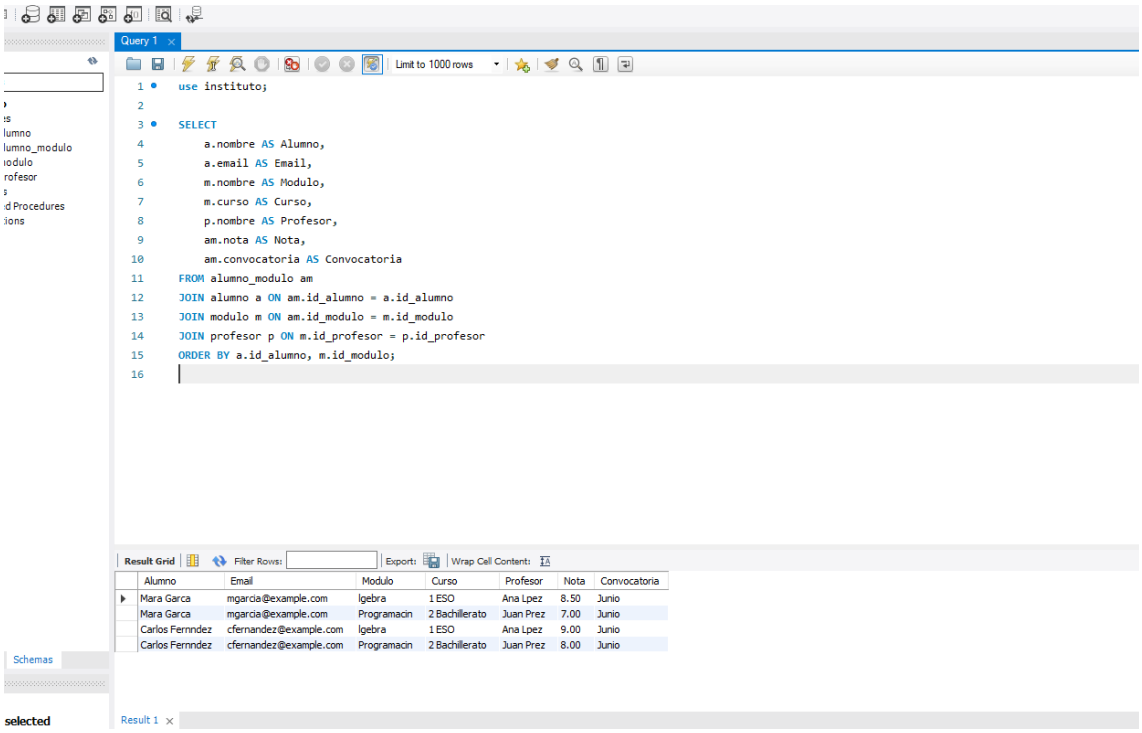


*Ilustración 19 Aquí por lo menos es a color*

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.

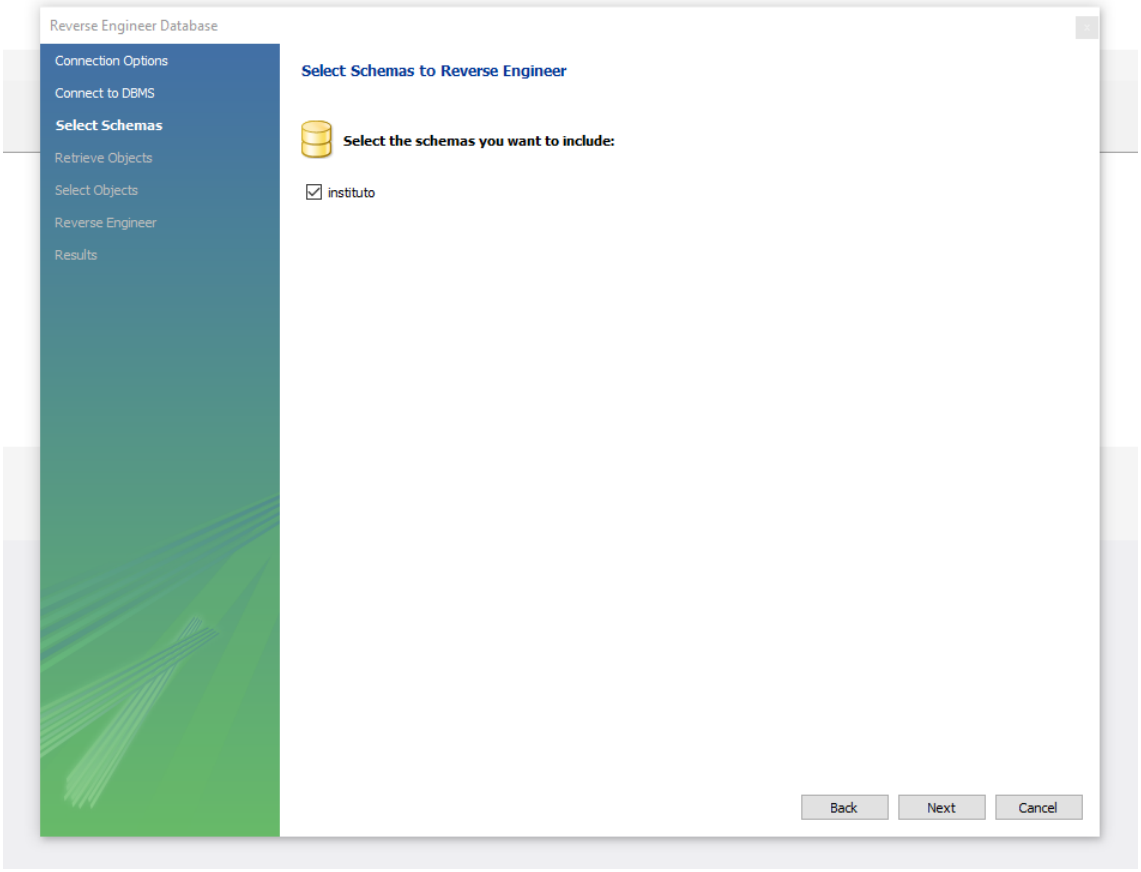


*Ilustración 20 Reverse Engineering*

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.

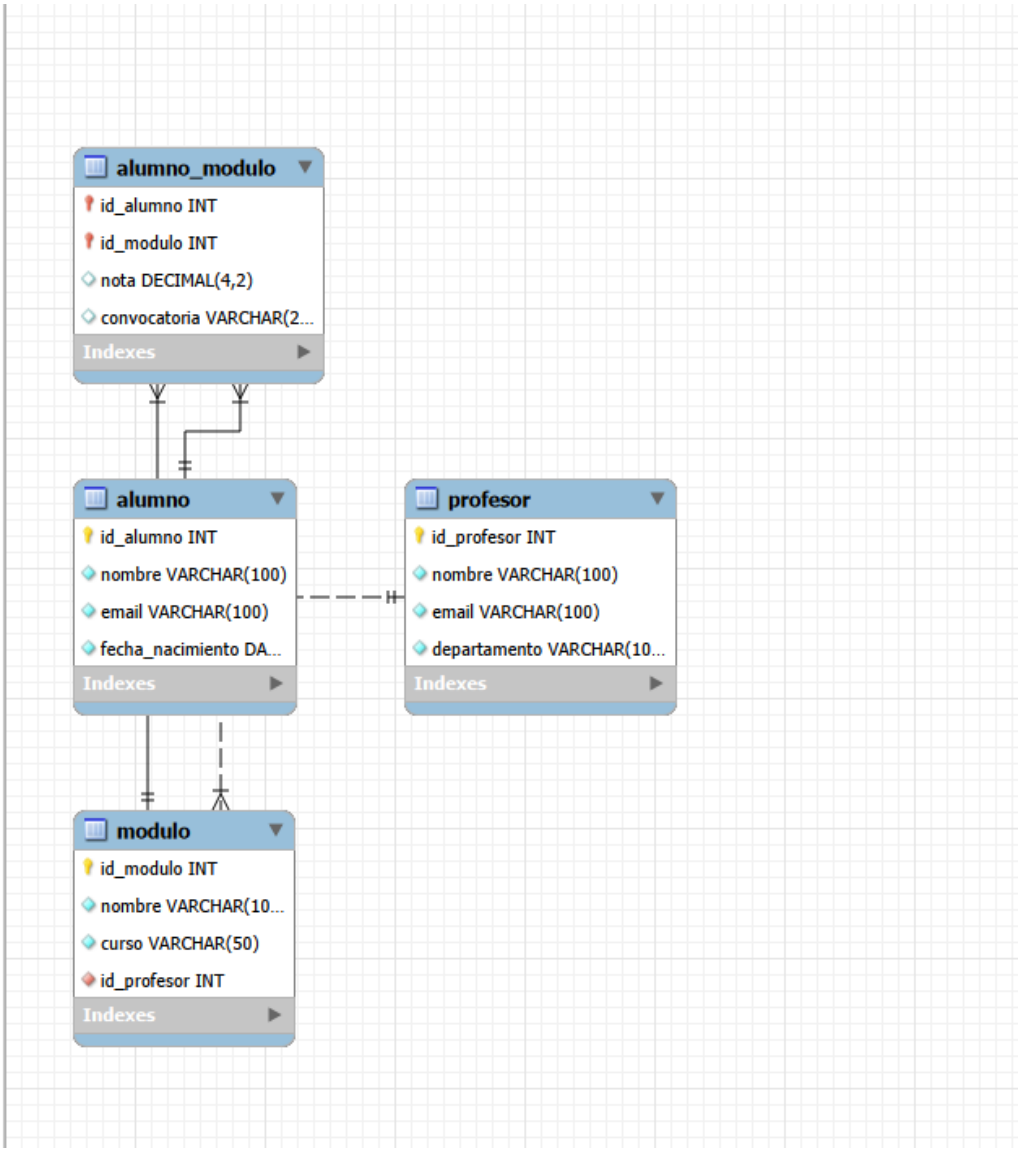


*Ilustración 21 Resultado final*

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 creemos el contenedor o lo arranquemos, se cargarán los datos desde allí. Eso sí, esta parte no la he probado.