En este laboratorio, seguiremos utilizando la base de datos **sakila**, disponible en MySQL. Esta base de datos está contenida en un **schema** del mismo nombre.

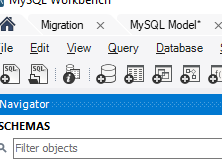
Trabajaremos sobre un **schema** distinto, para crear, modificar y borrar objetos en él, sin intervenir los objetos del schema **sakila**. Por ejemplo, crearemos **vistas** (consultas almacenadas) en el **schema info209**.

Tener 2 schemas separados permite distinguir entre nuestro código y el que trae MySQL. Esta forma de trabajar ayuda bastante cuando utilizamos sistemas desarrollados por terceros, ya que facilita distinguir nuestro desarrollo y también facilita la carga de las versiones futuras del software de terceros.

Si, por el contrario, escribimos código dentro del schema sakila (o del software de terceros), después será más difícil actualizar el software.

1. **Schemas.**

Para crear un schema, podemos utilizar un wizard o la línea de comando.



En este caso, usaremos la línea de comando:

create schema info209;

Después de crear el schema, con botón derecho sobre él, puedo seleccionar la opción “Set as Default Schema”, para crear los objetos ahí. Con la opción “Refresh All”, puedo actualizar la lista de objetos.

En esta sesión, usaremos el schema “**info209**” (o el nombre que ustedes quieran darle), para crear consultas almacenadas (vistas) que trabajen sobre el **schema sakila**.

1. **Usuarios.**

Para acceder a la información contenida en un schema, debo conectarme a un usuario de la base de datos.

Es usual que nos conectemos con el usuario root, pero esto tiene riesgos potenciales de seguridad. Por ejemplo, un desarrollador podría borrar por error objetos de un schema al cual no debiera tener acceso.

Para crear un usuario y asignarle una password, utilizo el siguiente comando:

create user '**juansalazar**'@'%' identified by '**jpsf**';/\*creen un usuario y password con su nombre \*/

Esta tarea debo hacerla conectado a un usuario con privilegios de administrador (ejm root).

A continuación, crearé una conexión a mi nuevo usuario en MySQL Workbench

|  |  |
| --- | --- |
| Dibujo en blanco y negro  El contenido generado por IA puede ser incorrecto. |  |

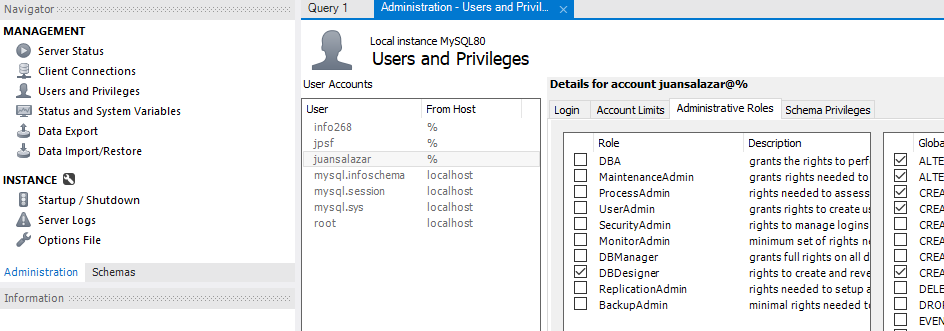
Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**Permisos.**

Aun cuando la base de datos MySQL no tiene definidos formalmente los roles, como un conjunto de privilegios, MySQL Workbench sí define las combinaciones más típicas, lo que facilita nuestro trabajo.

El rol de “DBDesigner” da al usuario privilegios para crear y modificar objetos. Los permisos debo otorgarlos desde una cuenta de **administrador** (root).



Además, para crear vistas, deberemos darle **permisos de consulta** (select) a tablas del schema sakila, a nuestro nuevo usuario.

La instrucción GRANT nos permite dar permisos de acceso a objetos de un schema.

Ejemplo (debe dar permisos de acceso a ambas tablas):

**GRANT SELECT ON** sakila.film **TO** juansalazar;

**GRANT SELECT ON** sakila.language **TO** juansalazar;

Además, como definí que el schema por default para juansalazar es **info209**, le debo dar privilegio de **select** de los objetos de ese schema (desde una cuenta de administrador).

**GRANT SELECT ON** info209.\* **TO** juansalazar;

Ahora me puedo conectar al usuario juansalazar

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

1. **Consultas**

Veamos ahora unas consultas sencillas, conectados a nuestro usuario (juansalazar, en mi caso):

1. Select \* from sakila.film;
2. Select \* from sakila.actor;   Al ejecutar esta consulta, verá que da error, porque su usuario no tiene **permiso de select** sobre la tabla **actor**. Debe darlo, **desde root**, con la instrucción GRANT.
3. Una consulta que entrega información de las tablas film y language:

SELECT f.title, f.release\_year, l.name, f.length

FROM sakila.film f, sakila.language l

WHERE f.language\_id = l.language\_id

ORDER BY f.title

1. Agregue a la consulta anterior la cantidad de actores que están registrados para cada película.

**Hint:** Debe darle permiso de SELECT a la tabla film\_actor, al usuario donde está realizando las consultas (en este caso, juansalazar).

SELECT

f.title,

f.release\_year,

l.name AS language,

f.length,

COUNT(fa.actor\_id) AS actor\_count

FROM sakila.film f

JOIN sakila.language l ON f.language\_id = l.language\_id

JOIN sakila.film\_actor fa ON f.film\_id = fa.film\_id

GROUP BY f.film\_id, f.title, f.release\_year, l.name, f.length

ORDER BY f.title;

1. Dado que utilizaremos el schema **info209** para crear vistas que involucran a todas las tablas del schema sakila, por simplicidad daremos permiso de SELECT de todas las tablas de **sakila** al usuario (**juansalazar**).

GRANT SELECT ON sakila.\* to juansalazar;

Esto debe hacerlo desde el usuario **root**.

1. **Vistas (consultas almacenadas)**

Una vista es una consulta que, para todos los efectos, es entendida por el álgebra relacional como una RELACIÓN; es decir, tiene propiedades similares a las de una tabla. Puedo hacer consultas sobre ella, usarla como fuente de datos en un join, etc.

Veamos algunos ejemplos:

1. CREATE VIEW v\_films\_largos

AS

SELECT \*

FROM sakila.film

WHERE length >90;

A continuación, realice la consulta

SELECT \* FROM v\_films\_largos;

Veamos la diferencia:

|  |  |
| --- | --- |
| Select count(\*) from sakila.film | Select count(\*) from v\_films\_largos |