

# Construcción de software y toma de decisiones

## TC2005B

**Dr. Esteban Castillo Juarez**

ITESM, Campus Santa Fe



esteban.castillojz@tec.mx

# Agenda

- Uso de la base de datos “sakila”
- Creación de tablas

# Uso de la base de datos “sakila”

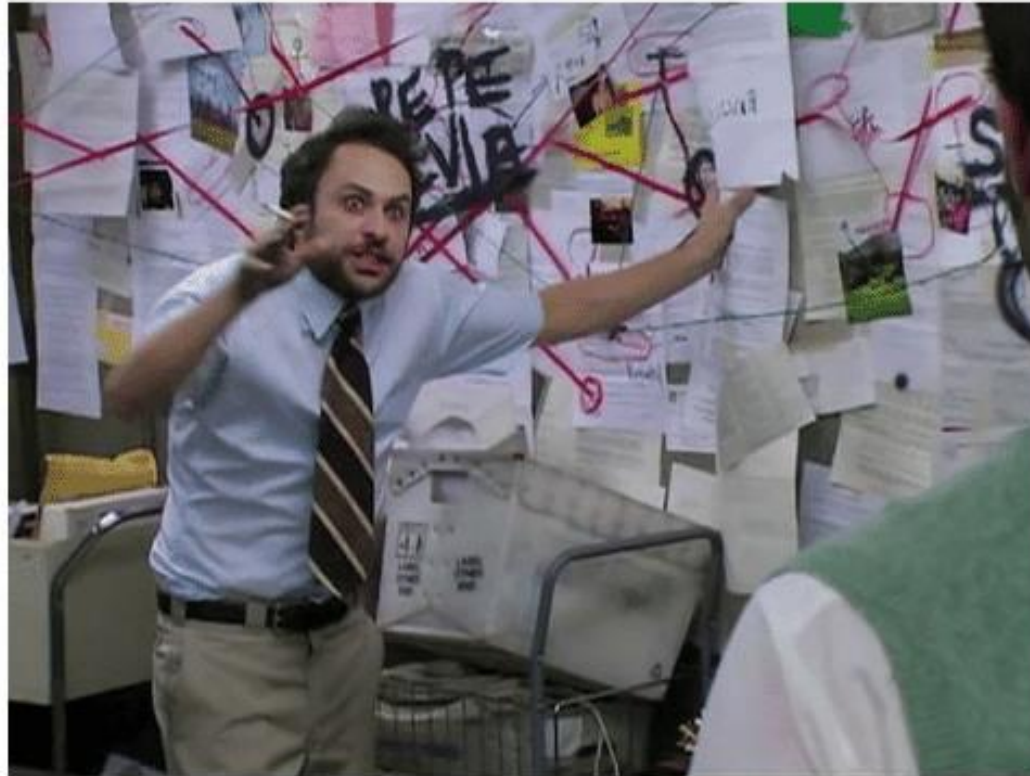
- Para poder entender la creación, modificación y uso de una base de datos usaremos un esquema preconfigurado.
- La base de datos Sakila, cuyo desarrollo comenzó en 2005\*, esta basada en el modelado de un extinto videoclub de películas.

Nota: Para saber mas acerca de esta base de datos puede leer el siguiente artículo: [“Three Approaches to MySQL Applications on Dell PowerEdge Servers”](#).



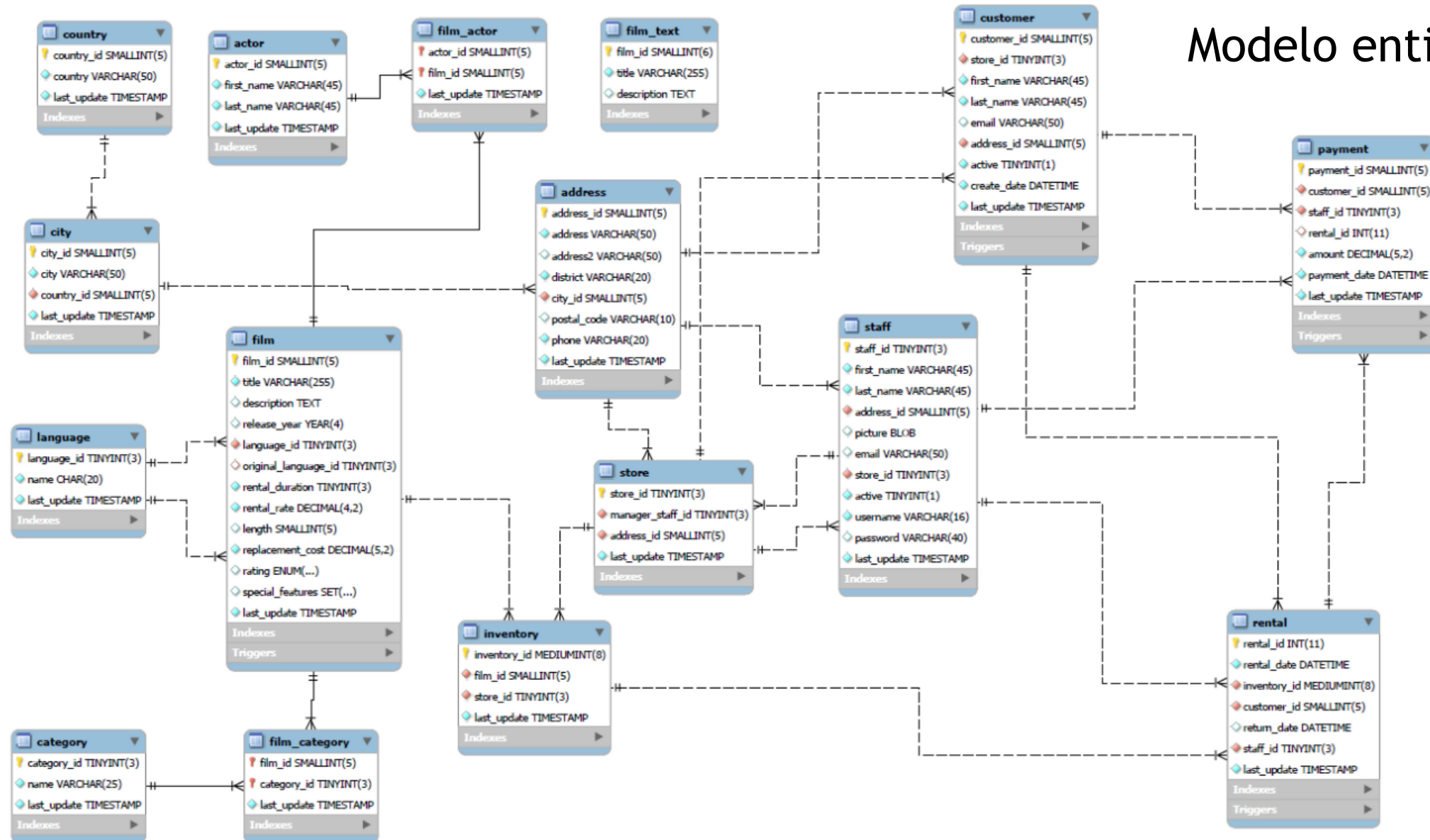
# Uso de la base de datos “sakila”

When people ask me to explain my database design



Its all related guys

# Uso de la base de datos “sakila”



Modelo entidad-relación

# Uso de la base de datos “sakila”

- En términos prácticos, MySQL proporciona dos scripts para importar esta base de datos.
- El primero tiene toda la lógica de negocio de la creación de tablas y restricciones.
- Mientras que el segundo “inyecta” o añade información a la base de datos.
- Ambos archivos están disponibles en la siguiente [liga](#) de GitHub del curso.



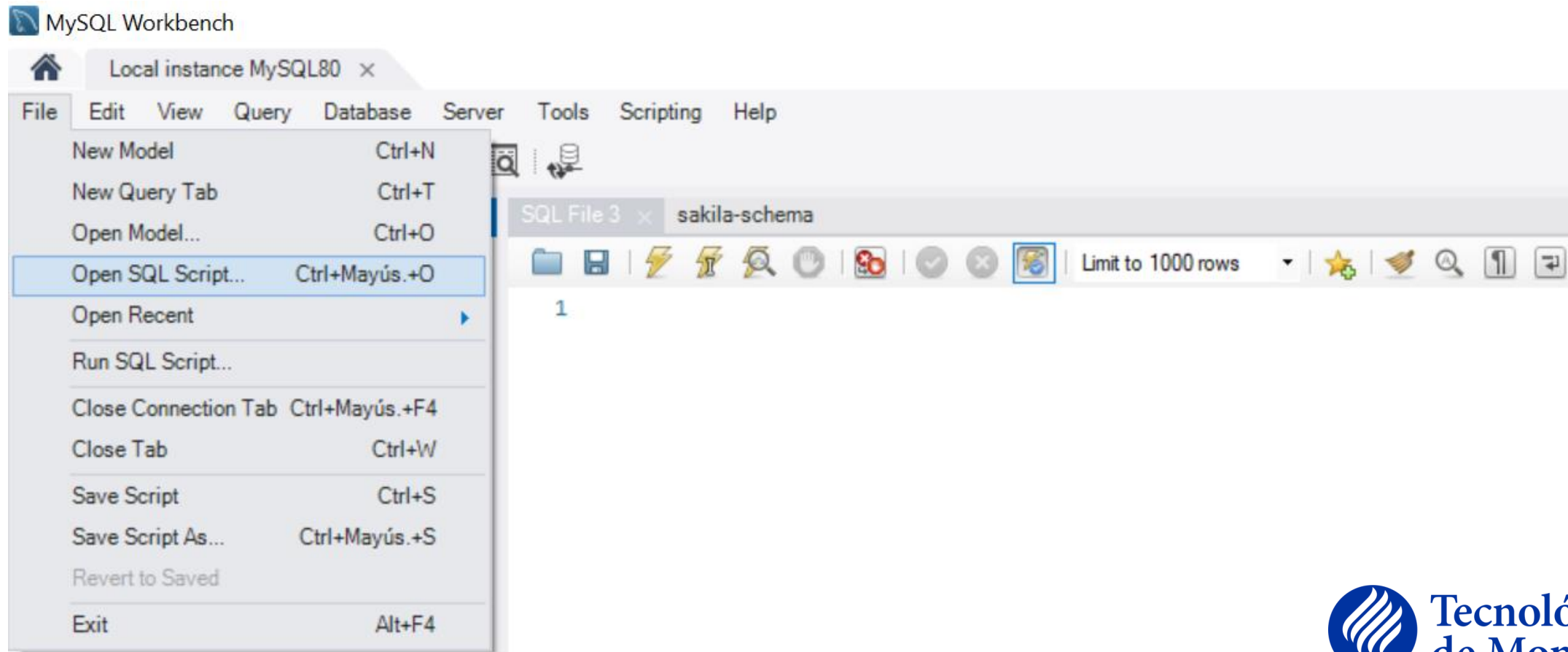
# Uso de la base de datos “sakila”

`wait-until DB_IS_READY`



# Uso de la base de datos “sakila”

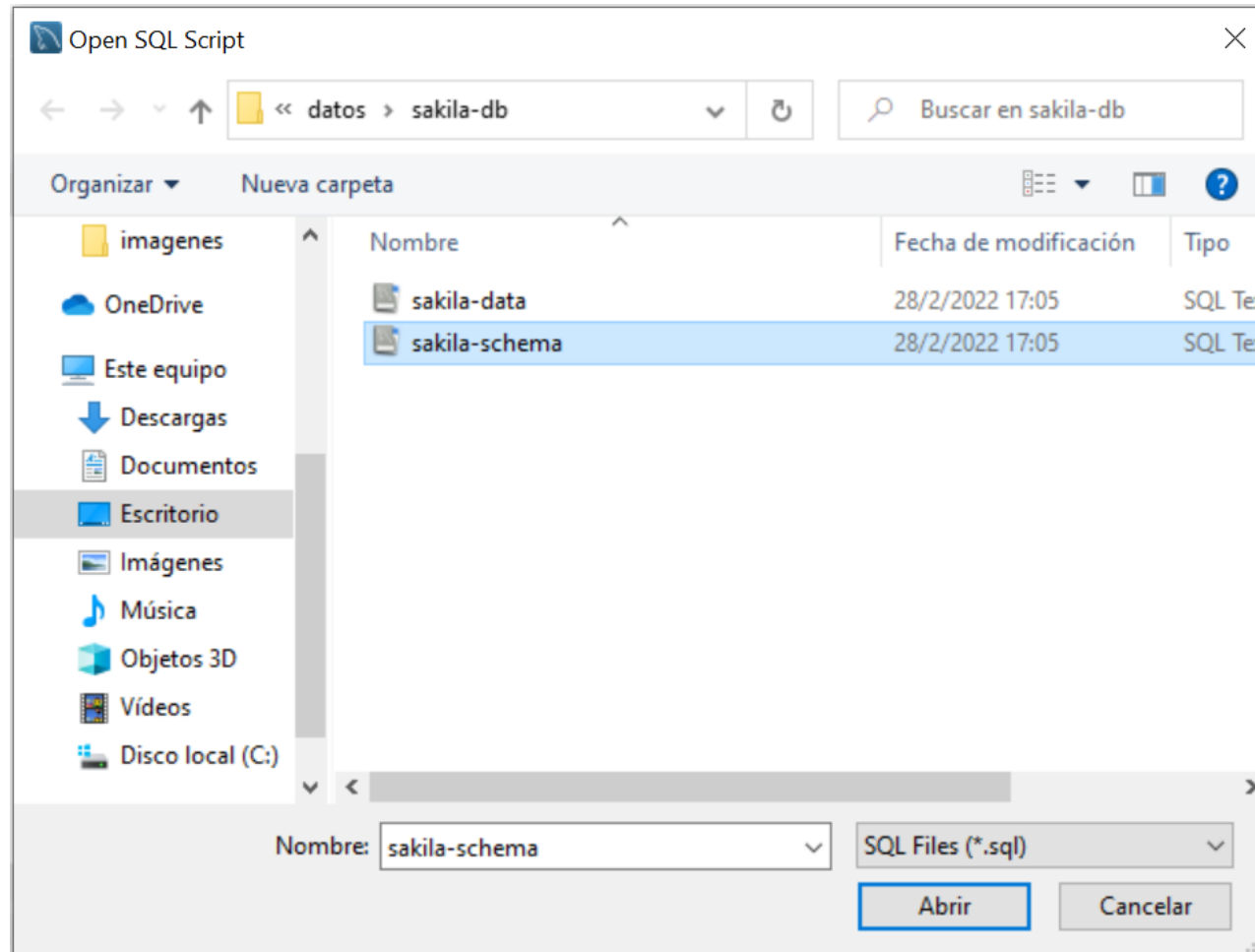
Dentro de MySQL Workbench se puede añadir el esquema sakila con todas sus tablas en la instancia local de la base de datos.





# Uso de la base de datos “sakila”

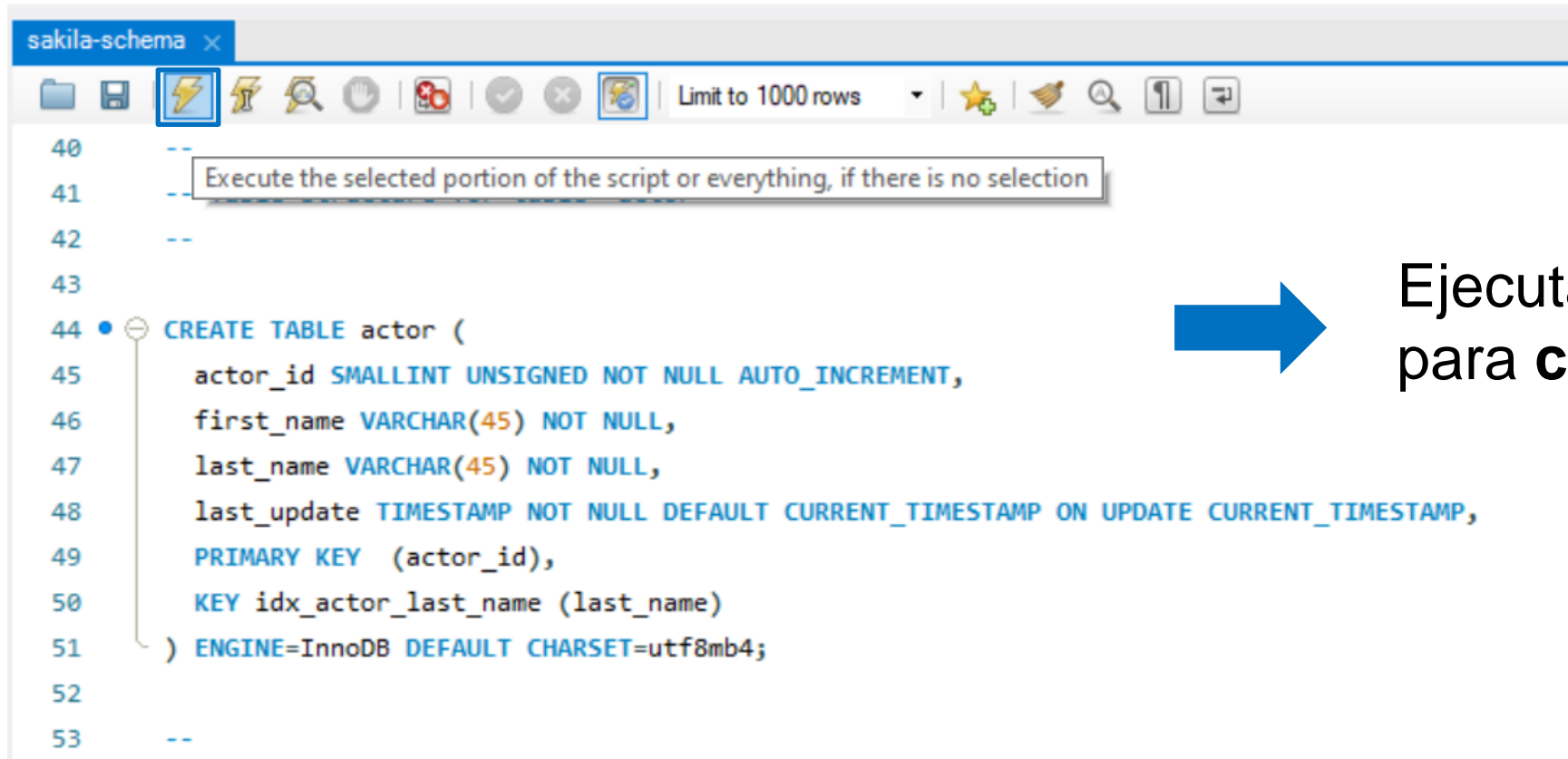
Dentro de MySQL Workbench se puede añadir el esquema sakila con todas sus tablas en la instancia local de la base de datos.



**Creación** de tablas y relaciones en el esquema.

# Uso de la base de datos “sakila”

Dentro de MySQL Workbench se puede añadir el esquema sakila con todas sus tablas en la instancia local de la base de datos.



The screenshot shows the MySQL Workbench interface with a window titled "sakila-schema". The toolbar includes icons for file operations, execution (lightning bolt), and other database functions. A tooltip is visible over the execution icon, stating: "Execute the selected portion of the script or everything, if there is no selection". The SQL editor displays the following script:

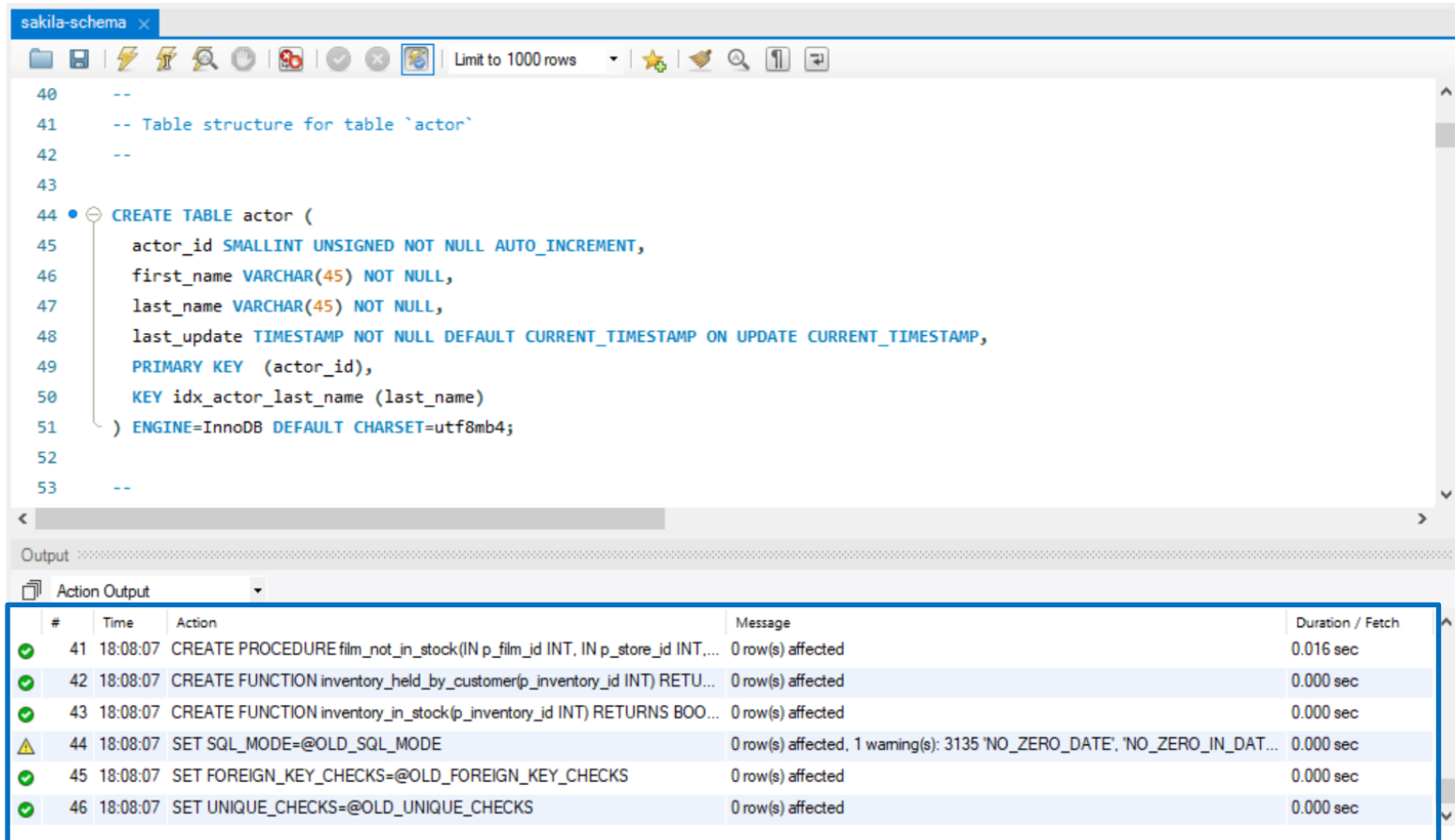
```
40  --
41  -- Execute the selected portion of the script or everything, if there is no selection
42  --
43
44  ● CREATE TABLE actor (
45      actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
46      first_name VARCHAR(45) NOT NULL,
47      last_name VARCHAR(45) NOT NULL,
48      last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
49      PRIMARY KEY (actor_id),
50      KEY idx_actor_last_name (last_name)
51  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
52
53  --
```



Ejecutamos todo el script para **crear** las tablas.

# Uso de la base de datos “sakila”

Dentro de MySQL Workbench se puede añadir el esquema sakila con todas sus tablas en la instancia local de la base de datos.



The screenshot shows the MySQL Workbench interface with the 'sakila-schema' window open. The main editor displays the SQL script for creating the 'actor' table. Below the editor, the 'Output' tab is selected, showing a list of actions and their results. A blue arrow points from the text 'La salida no muestra errores y las tablas se han creado.' to the output window.

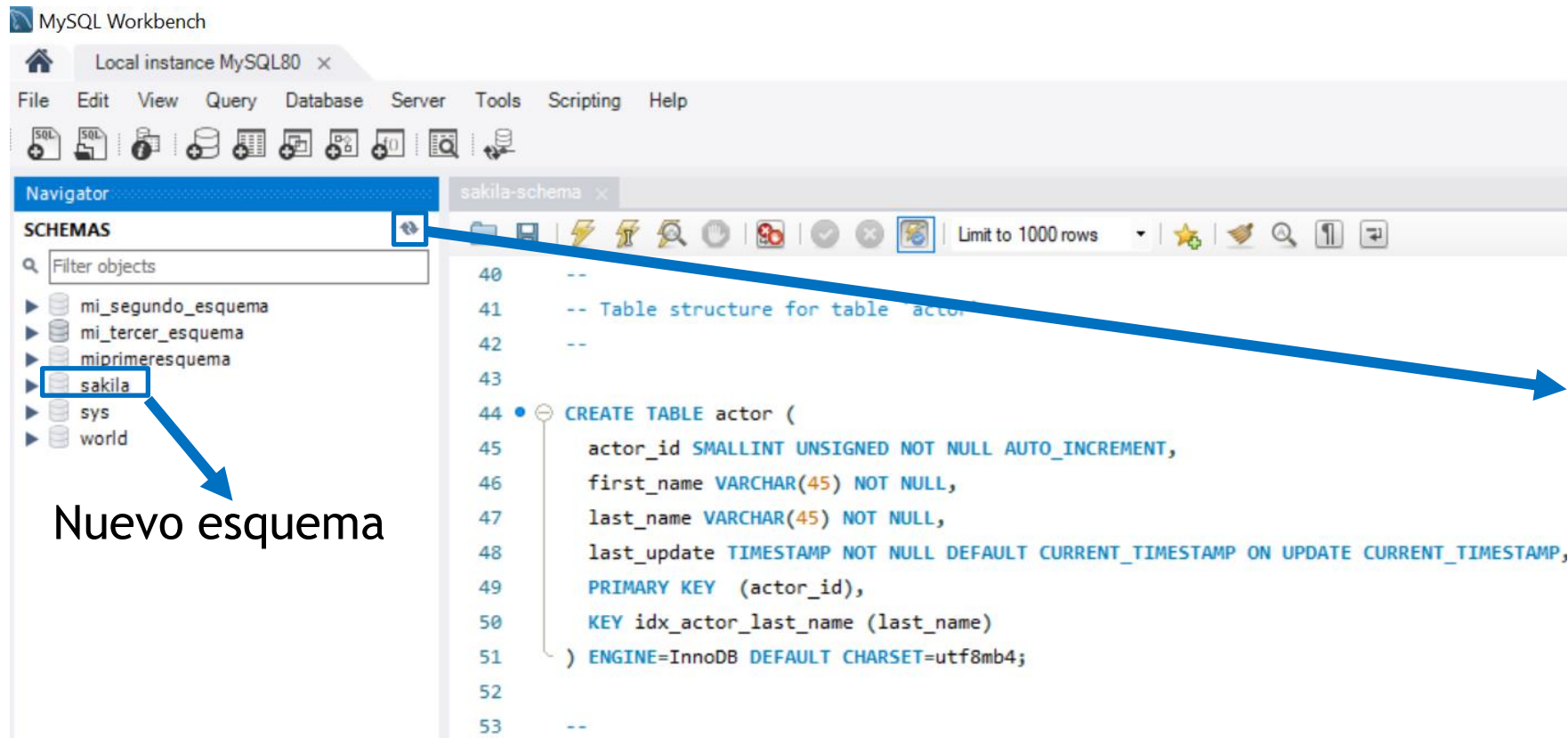
```
--
-- Table structure for table `actor`
--
CREATE TABLE actor (
  actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(45) NOT NULL,
  last_name VARCHAR(45) NOT NULL,
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (actor_id),
  KEY idx_actor_last_name (last_name)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
```

#	Time	Action	Message	Duration / Fetch
✓ 41	18:08:07	CREATE PROCEDURE film_not_in_stock(IN p_film_id INT, IN p_store_id INT,...	0 row(s) affected	0.016 sec
✓ 42	18:08:07	CREATE FUNCTION inventory_held_by_customer(p_inventory_id INT) RETU...	0 row(s) affected	0.000 sec
✓ 43	18:08:07	CREATE FUNCTION inventory_in_stock(p_inventory_id INT) RETURNS BOO...	0 row(s) affected	0.000 sec
⚠ 44	18:08:07	SET SQL_MODE=@OLD_SQL_MODE	0 row(s) affected, 1 warning(s): 3135 'NO_ZERO_DATE', 'NO_ZERO_IN_DAT...	0.000 sec
✓ 45	18:08:07	SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS	0 row(s) affected	0.000 sec
✓ 46	18:08:07	SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS	0 row(s) affected	0.000 sec

La salida no muestra errores y las **tablas** se han creado.

# Uso de la base de datos “sakila”

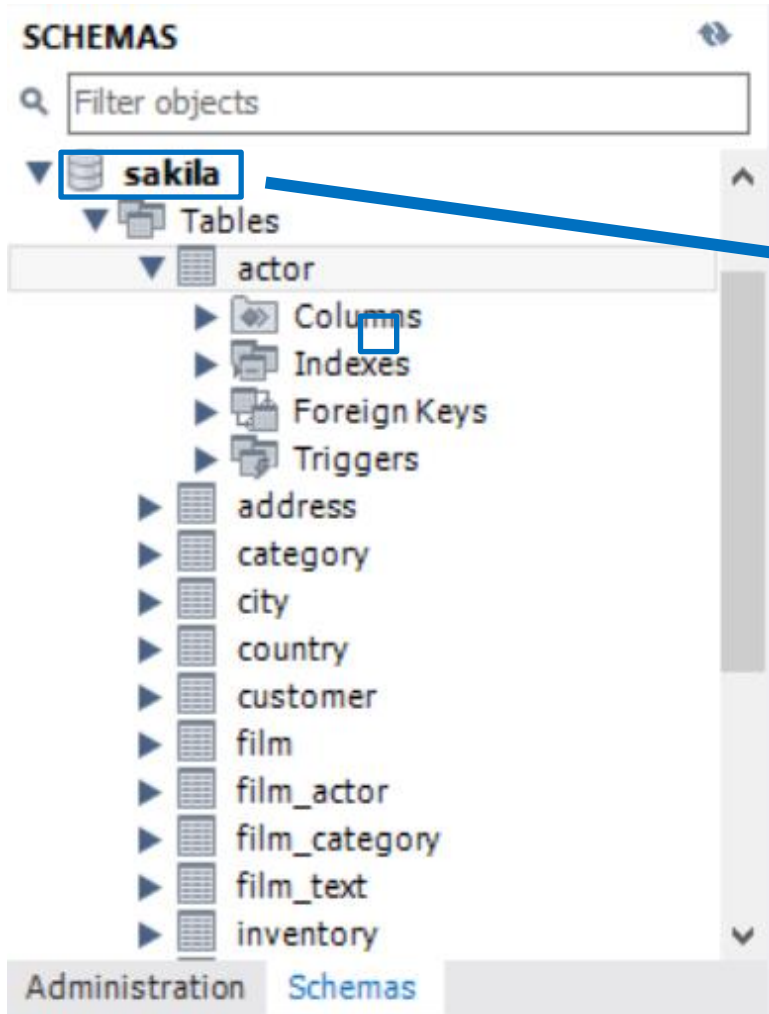
Dentro de MySQL Workbench se puede añadir el esquema sakila con todas sus tablas en la instancia local de la base de datos.



Si no se refleja el esquema, podemos actualizarlo manualmente.

# Uso de la base de datos “sakila”

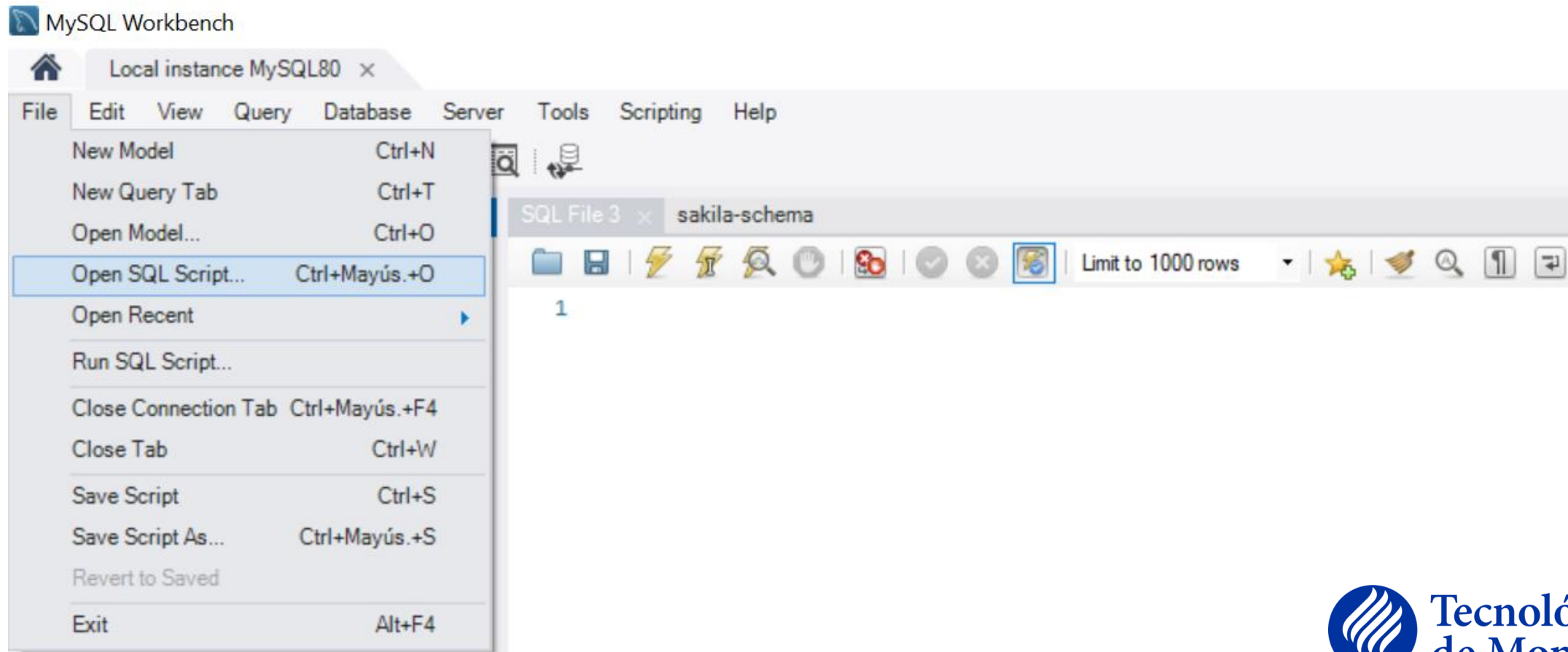
Dentro de MySQL Workbench se puede añadir el esquema sakila con todas sus tablas en la instancia local de la base de datos.



Podemos inspeccionar todas las tablas creadas así como las columnas y restricciones impuestas sobre estas.

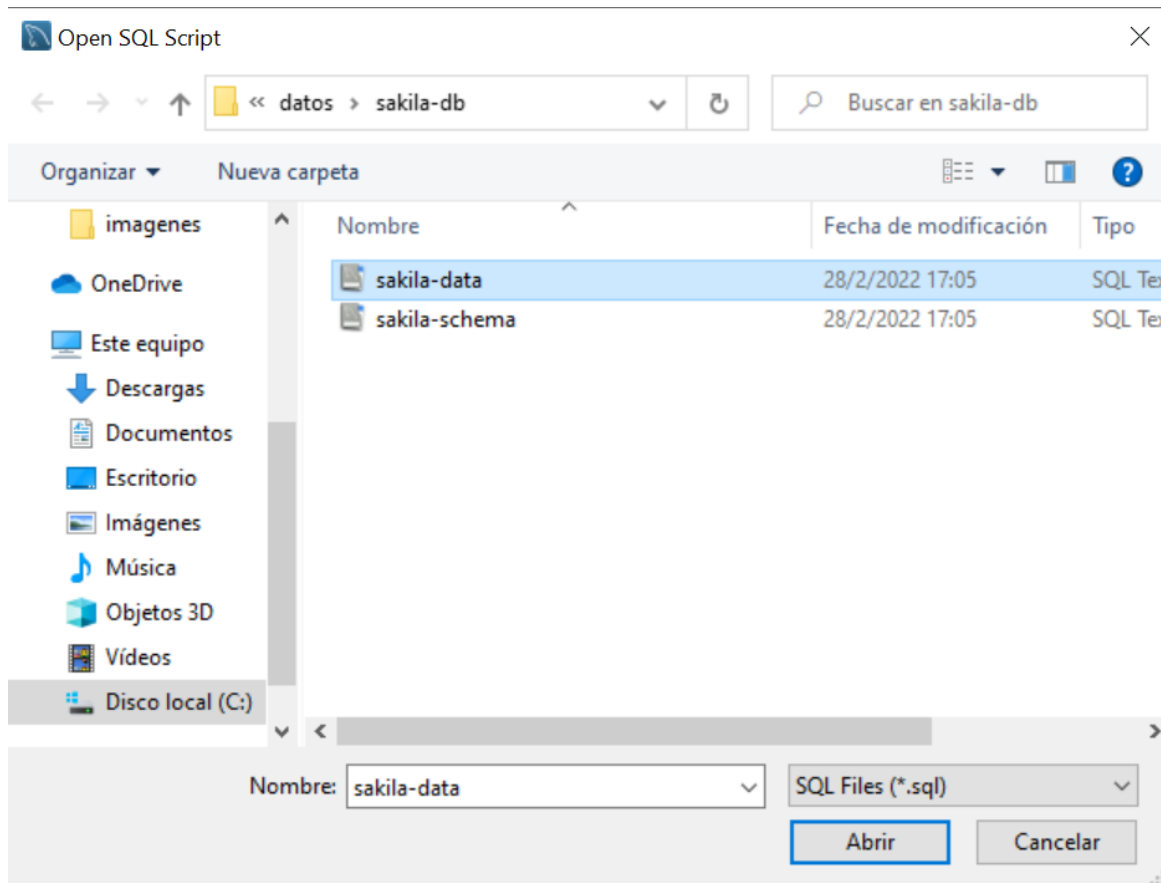
# Uso de la base de datos “sakila”

Dentro de MySQL Workbench se puede insertar filas en el esquema sakila de la base de datos.



# Uso de la base de datos “sakila”

Dentro de MySQL Workbench se puede insertar filas en el esquema sakila de la base de datos.

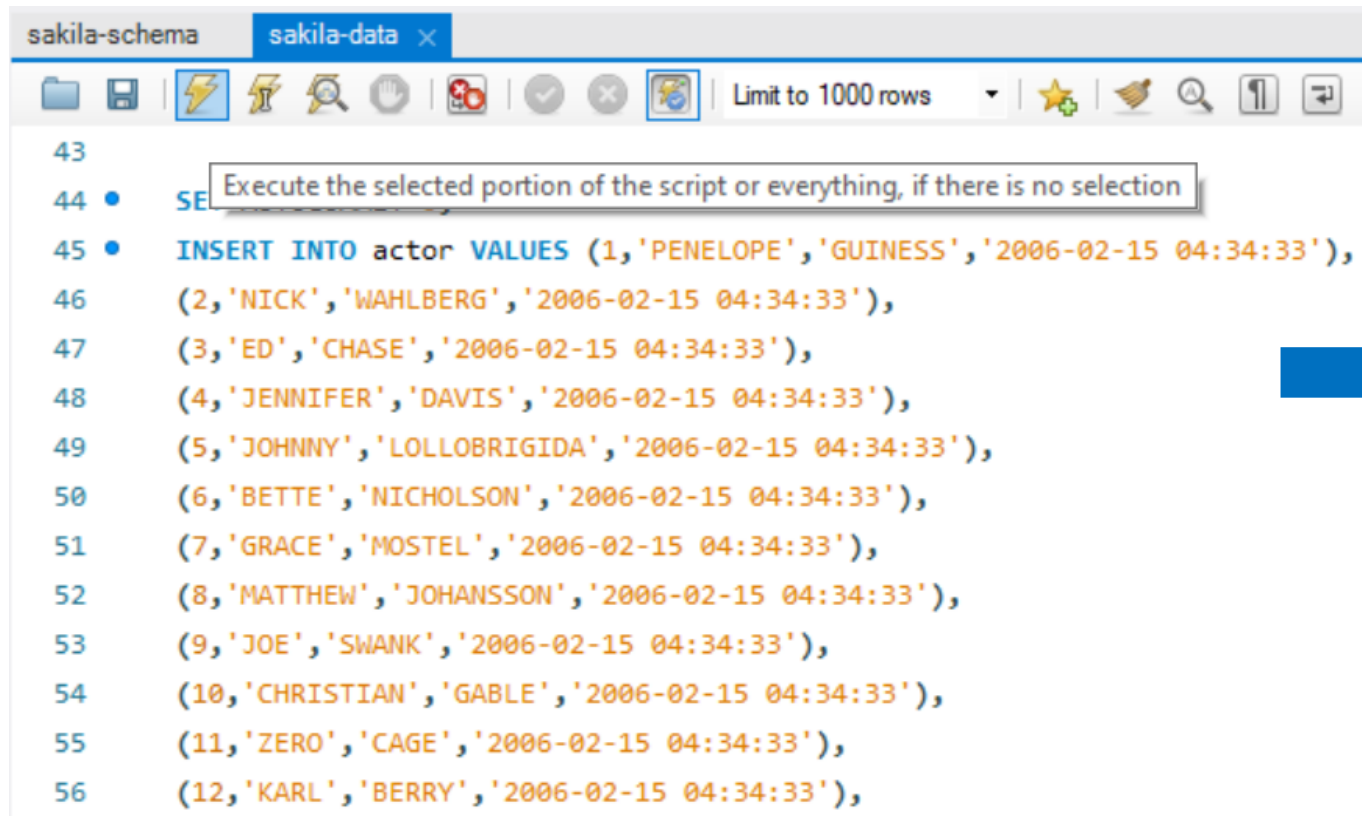


**Insertamos** las filas de las distintas tablas, mas adelante veremos el código SQL apropiado.



# Uso de la base de datos “sakila”

Dentro de MySQL Workbench se puede insertar filas en el esquema sakila de la base de datos.



The screenshot shows the MySQL Workbench interface with two tabs: 'sakila-schema' and 'sakila-data'. The 'sakila-data' tab is active, displaying a script editor. The script contains an SQL INSERT statement for the 'actor' table. A tooltip is visible over the 'Execute' button (a lightning bolt icon) in the toolbar, stating 'Execute the selected portion of the script or everything, if there is no selection'. The script lines are numbered 43 to 56.

```
43
44 • SE
45 • INSERT INTO actor VALUES (1, 'PENELOPE', 'GUINNESS', '2006-02-15 04:34:33'),
46   (2, 'NICK', 'WAHLBERG', '2006-02-15 04:34:33'),
47   (3, 'ED', 'CHASE', '2006-02-15 04:34:33'),
48   (4, 'JENNIFER', 'DAVIS', '2006-02-15 04:34:33'),
49   (5, 'JOHNNY', 'LOLLOBRIGIDA', '2006-02-15 04:34:33'),
50   (6, 'BETTE', 'NICHOLSON', '2006-02-15 04:34:33'),
51   (7, 'GRACE', 'MOSTEL', '2006-02-15 04:34:33'),
52   (8, 'MATTHEW', 'JOHANSSON', '2006-02-15 04:34:33'),
53   (9, 'JOE', 'SWANK', '2006-02-15 04:34:33'),
54   (10, 'CHRISTIAN', 'GABLE', '2006-02-15 04:34:33'),
55   (11, 'ZERO', 'CAGE', '2006-02-15 04:34:33'),
56   (12, 'KARL', 'BERRY', '2006-02-15 04:34:33'),
```

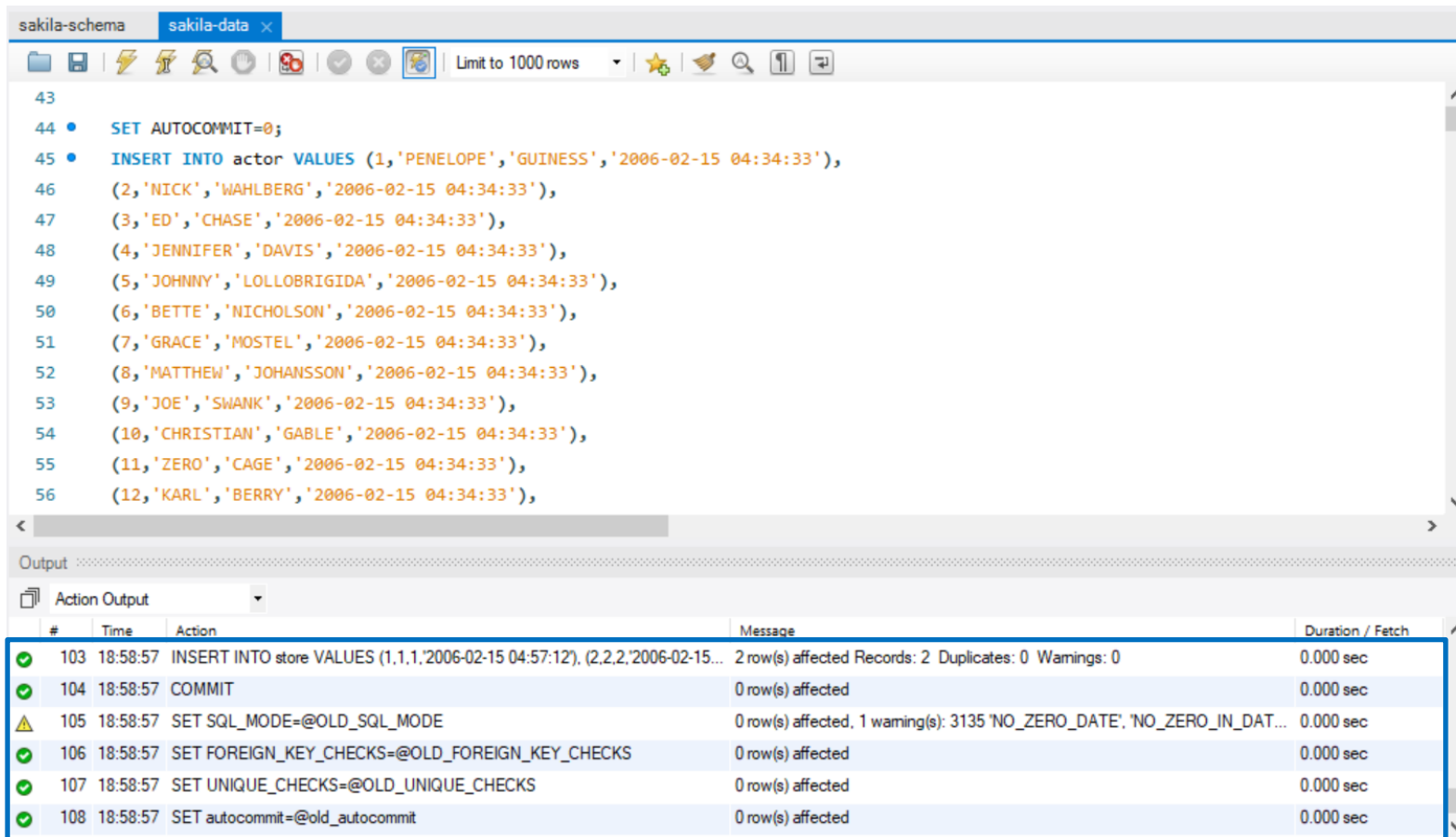


Ejecutamos todo el script para **insertar** filas en las tablas.



# Uso de la base de datos “sakila”

Dentro de MySQL Workbench se puede insertar filas en el esquema sakila de la base de datos.

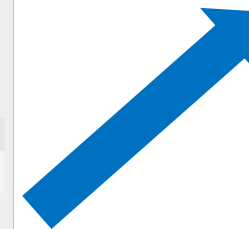


The screenshot shows the MySQL Workbench interface with the 'sakila-schema' and 'sakila-data' tabs. The SQL editor contains a script to insert 12 rows into the 'actor' table. The output pane shows the execution results, including the successful insertion of 2 rows and the setting of various SQL modes.

```
43
44 • SET AUTOCOMMIT=0;
45 • INSERT INTO actor VALUES (1,'PENELOPE','GUINNESS','2006-02-15 04:34:33'),
46 (2,'NICK','WAHLBERG','2006-02-15 04:34:33'),
47 (3,'ED','CHASE','2006-02-15 04:34:33'),
48 (4,'JENNIFER','DAVIS','2006-02-15 04:34:33'),
49 (5,'JOHNNY','LOLLOBRIGIDA','2006-02-15 04:34:33'),
50 (6,'BETTE','NICHOLSON','2006-02-15 04:34:33'),
51 (7,'GRACE','MOSTEL','2006-02-15 04:34:33'),
52 (8,'MATTHEW','JOHANSSON','2006-02-15 04:34:33'),
53 (9,'JOE','SWANK','2006-02-15 04:34:33'),
54 (10,'CHRISTIAN','GABLE','2006-02-15 04:34:33'),
55 (11,'ZERO','CAGE','2006-02-15 04:34:33'),
56 (12,'KARL','BERRY','2006-02-15 04:34:33'),
```

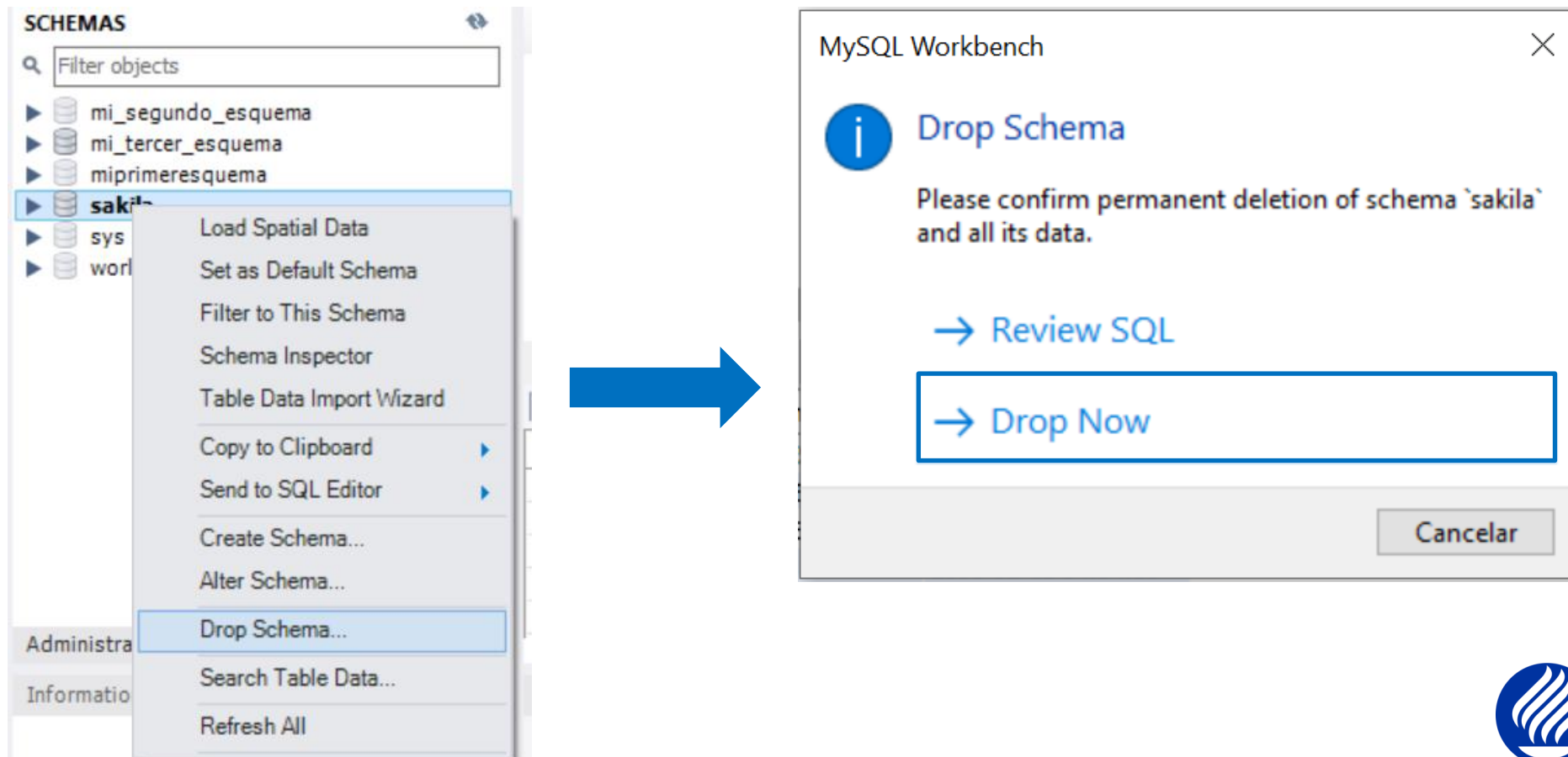
#	Time	Action	Message	Duration / Fetch
✓ 103	18:58:57	INSERT INTO store VALUES (1,1,1,'2006-02-15 04:57:12'), (2,2,2,'2006-02-15...	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
✓ 104	18:58:57	COMMIT	0 row(s) affected	0.000 sec
⚠ 105	18:58:57	SET SQL_MODE=@OLD_SQL_MODE	0 row(s) affected, 1 warning(s): 3135 'NO_ZERO_DATE', 'NO_ZERO_IN_DAT...	0.000 sec
✓ 106	18:58:57	SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS	0 row(s) affected	0.000 sec
✓ 107	18:58:57	SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS	0 row(s) affected	0.000 sec
✓ 108	18:58:57	SET autocommit=@old_autocommit	0 row(s) affected	0.000 sec

La salida no muestra errores y las **filas** se han creado.



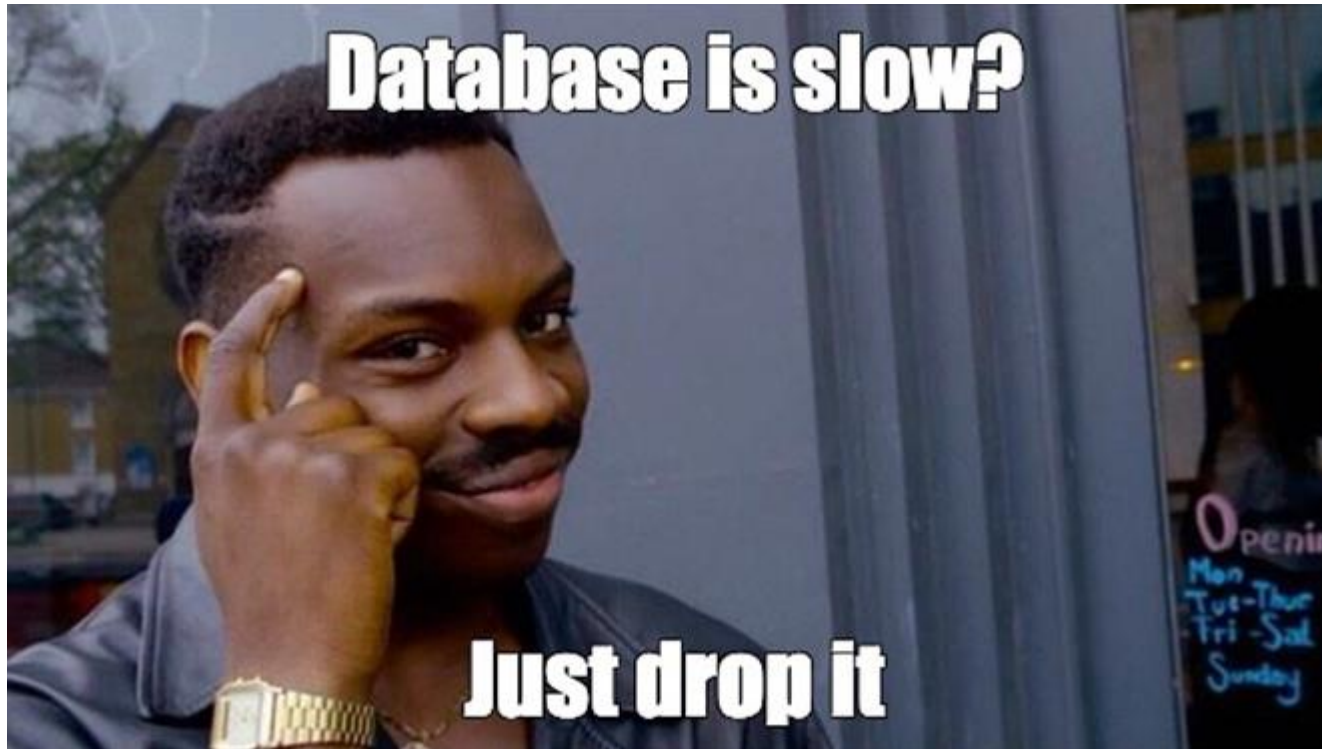
# Uso de la base de datos “sakila”

Si lo deseamos podemos eliminar el esquema sakila de la instancia local de MySQL Workbench.



# Uso de la base de datos “sakila”

Si los deseamos podemos eliminar el esquema sakila de la instancia local de MySQL Workbench.



# Creación de tablas

Dentro del esquema sakila, tomemos la tabla **actor**, el cual participa en una película (La película esta disponible para renta en el videoclub).

```
CREATE TABLE actor (  
  actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  first_name VARCHAR(45) NOT NULL,  
  last_name VARCHAR(45) NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id),  
  KEY idx_actor_last_name (last_name)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

# Creación de tablas

Dentro del esquema sakila, tomemos la tabla **actor**, el cual participa en una película (La película esta disponible para renta en el videoclub).

```
CREATE TABLE actor (  
  actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  first_name VARCHAR(45) NOT NULL,  
  last_name VARCHAR(45) NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id),  
  KEY idx_actor_last_name (last_name)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

- **Nombre de la tabla** con elementos/restricciones separados por comas.
- Todas las columnas de la tabla están encerradas entre paréntesis.

# Creación de tablas

```
CREATE TABLE actor (  
    actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
    first_name VARCHAR(45) NOT NULL,  
    last_name VARCHAR(45) NOT NULL,  
    last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
    PRIMARY KEY (actor_id),  
    KEY idx_actor_last_name (last_name)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1. Creación de columnas en la tabla con sus restricciones.
2. Creación de llaves para crear la relación entre tablas.
3. Definición de un motor de almacenamiento y codificación para la tabla.

# Creación de tablas

```
CREATE TABLE actor (  
  actor_id SMALLINT UNSIGNED NOT NULL AUTO INCREMENT,  
  first_name VARCHAR(45) NOT NULL,  
  last_name VARCHAR(45) NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id),  
  KEY idx_actor_last_name (last_name)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1. Nombre de la columna.
2. Tipo de dato de la columna (primera restricción de integridad).
3. Tipo de dato sin signo (segunda restricción de integridad).
4. No se permiten valores nulos en esa columna (tercera restricción de integridad).
5. Permite generar automáticamente un número único cuando se inserta una nueva fila en una tabla.



# Creación de tablas

```
CREATE TABLE actor (  
  actor_id SMALLINT UNSIGNED NOT NULL AUTO INCREMENT,  
  first_name VARCHAR(45) NOT NULL,  
  last_name VARCHAR(45) NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id),  
  KEY idx_actor_last_name (last_name)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1. Nombre de la columna.
2. Tipo de dato de la columna (primera restricción de integridad).
3. Tipo de dato sin signo (segunda restricción de integridad).
4. No se permiten valores nulos en esa columna (tercera restricción de integridad).
5. Permite generar automáticamente un número único cuando se inserta una nueva fila en una tabla.
6. La fecha tiene la estampa de tiempo actual para su valor predeterminado y se actualiza automáticamente.



# Creación de tablas

## Tipos de datos **genéricos** en MySQL

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 2147483647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LOBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

# Creación de tablas

Tipos de datos **específicos** para números enteros en MySQL

Numeric Types	Description
<b>TINYINT</b>	A very small integer
<b>SMALLINT</b>	A small integer
<b>MEDIUMINT</b>	A medium-sized integer
<b>INT</b>	A standard integer
<b>BIGINT</b>	A large integer
<b>DECIMAL</b>	A fixed-point number
<b>FLOAT</b>	A single-precision floating point number
<b>DOUBLE</b>	A double-precision floating point number
<b>BIT</b>	A bit field

# Creación de tablas

## Tipos de datos **específicos** para cadenas de caracteres en MySQL

String Types	Description
<b>CHAR</b>	A fixed-length nonbinary (character) string
<b>VARCHAR</b>	A variable-length non-binary string
<b>BINARY</b>	A fixed-length binary string
<b>VARBINARY</b>	A variable-length binary string
<b>TINYBLOB</b>	A very small BLOB (binary large object)
<b>BLOB</b>	A small BLOB
<b>MEDIUMBLOB</b>	A medium-sized BLOB
<b>LOBLOB</b>	A large BLOB
<b>TINYTEXT</b>	A very small non-binary string
<b>TEXT</b>	A small non-binary string
<b>MEDIUMTEXT</b>	A medium-sized non-binary string
<b>LONGTEXT</b>	A large non-binary string
<b>ENUM</b>	An enumeration; each column value may be assigned one enumeration member
<b>SET</b>	A set; each column value may be assigned zero or more <b>SET</b> members

# Creación de tablas

Tipos de datos **específicos** para **estampas de tiempo** (timestamp) en MySQL

Date and Time Types	Description
<b>DATE</b>	A date value in <b>CCYY-MM-DD</b> format
<b>TIME</b>	A time value in <b>hh:mm:ss</b> format
<b>DATETIME</b>	A date and time value in <b>CCYY-MM-DD hh:mm:ss</b> format
<b>TIMESTAMP</b>	A timestamp value in <b>CCYY-MM-DD hh:mm:ss</b> format
<b>YEAR</b>	A year value in <b>CCYY</b> or <b>YY</b> format

# Creación de tablas

```
CREATE TABLE actor (  
  actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  first_name VARCHAR(45) NOT NULL,  
  last_name VARCHAR(45) NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id),  
  KEY idx_actor_last_name (last_name)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1. El motor de almacenamiento (storage-engine) es quien almacenará, manejará y recuperará información de una tabla en particular.
2. Los motores más conocidos son MyISAM e InnoDB.

# Creación de tablas

```
CREATE TABLE actor (  
  actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  first_name VARCHAR(45) NOT NULL,  
  last_name VARCHAR(45) NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id),  
  KEY idx_actor_last_name (last_name)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1. Se crea una llave primaria para asegurar que no existan filas repetidas.
2. Se crea un índice (llave especial para acelerar búsquedas sobre columnas específicas en MySQL, NO TODO SON LLAVES PRIMARIAS Y SECUNDARIAS).

# Creación de tablas

```
CREATE TABLE actor (  
  actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  first_name VARCHAR(45) NOT NULL,  
  last_name VARCHAR(45) NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id),  
  KEY idx_actor_last_name (last_name)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

3. Si necesitamos transacciones (bloque de instrucciones SQL), llaves foráneas y bloqueo de información, tendremos que escoger InnoDB. Por el contrario, escogeremos MyISAM en aquellos casos en los que predominen las consultas SELECT a la base de datos (sin muchas inserciones, actualizaciones, etc.).

# Creación de tablas

```
CREATE TABLE actor (  
  actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  first_name VARCHAR(45) NOT NULL,  
  last_name VARCHAR(45) NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id),  
  KEY idx_actor_last_name (last_name)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

4. InnoDB dota a MySQL de un motor de almacenamiento transaccional con capacidades de commit, rollback y recuperación de fallos.
5. InnoDB realiza bloqueos a nivel de fila y también proporciona funciones de lectura consistente para múltiples usuarios simultáneos.



# Creación de tablas

```
CREATE TABLE actor (  
  actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  first_name VARCHAR(45) NOT NULL,  
  last_name VARCHAR(45) NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id),  
  KEY idx_actor_last_name (last_name)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

6. MyISAM es el motor por defecto. Para crear una tabla InnoDB se debe especificar la opción ENGINE = InnoDB o TYPE = InnoDB en la sentencia SQL de creación de tabla.
7. MyISAM proporciona mayor velocidad en general a la hora de recuperar datos.

# Creación de tablas

```
CREATE TABLE actor (  
  actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  first_name VARCHAR(45) NOT NULL,  
  last_name VARCHAR(45) NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id),  
  KEY idx_actor_last_name (last_name)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

8. MyISAM es recomendable para aplicaciones en las que dominan las sentencias SELECT ante los INSERT /UPDATE.
9. MyISAM tiene una ausencia de características de atomicidad ya que no tiene que hacer comprobaciones de la integridad referencial, ni bloquear las tablas para realizar las operaciones, esto nos lleva a una mayor velocidad.

# Creación de tablas

```
CREATE TABLE actor (  
  actor_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  first_name VARCHAR(45) NOT NULL,  
  last_name VARCHAR(45) NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id),  
  KEY idx_actor_last_name (last_name)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

1. MySQL incluye compatibilidad con distintos tipos de codificaciones de caracteres que le permiten almacenar datos usando una variedad de información textual.
2. El conjunto de caracteres predeterminados de MySQL es latin1 y latin1\_swedish\_ci, pero puede especificar conjuntos de caracteres distintos para la base de datos, tabla, columna y cadenas de caracteres específicas.

# Creación de tablas

Dentro del esquema sakila, tomemos la tabla actor de película, la cual empareja una película específica con un actor específico:

```
CREATE TABLE film_actor (  
  actor_id SMALLINT UNSIGNED NOT NULL,  
  film_id SMALLINT UNSIGNED NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id, film_id),  
  KEY idx_fk_film_id (`film_id`),  
  CONSTRAINT fk_film_actor_actor FOREIGN KEY (actor_id) REFERENCES actor (actor_id) ON DELETE RESTRICT ON UPDATE CASCADE,  
  CONSTRAINT fk_film_actor_film FOREIGN KEY (film_id) REFERENCES film (film_id) ON DELETE RESTRICT ON UPDATE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```



¿A que se refiere esta instrucción?

# Creación de tablas

Dentro del esquema sakila, tomemos la tabla actor de película, la cual empareja una película específica con un actor específico:

```
CREATE TABLE film_actor (  
  actor_id SMALLINT UNSIGNED NOT NULL,  
  film_id SMALLINT UNSIGNED NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id, film_id),  
  KEY idx_fk_film_id (`film_id`),  
  CONSTRAINT fk_film_actor_actor FOREIGN KEY (actor_id) REFERENCES actor (actor_id) ON DELETE RESTRICT ON UPDATE CASCADE,  
  CONSTRAINT fk_film_actor_film FOREIGN KEY (film_id) REFERENCES film (film_id) ON DELETE RESTRICT ON UPDATE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```



Básicamente es a la creación de una llave secundaria o foránea, la cual cree una relación entre la tabla actor y actor de película.

# Creación de tablas

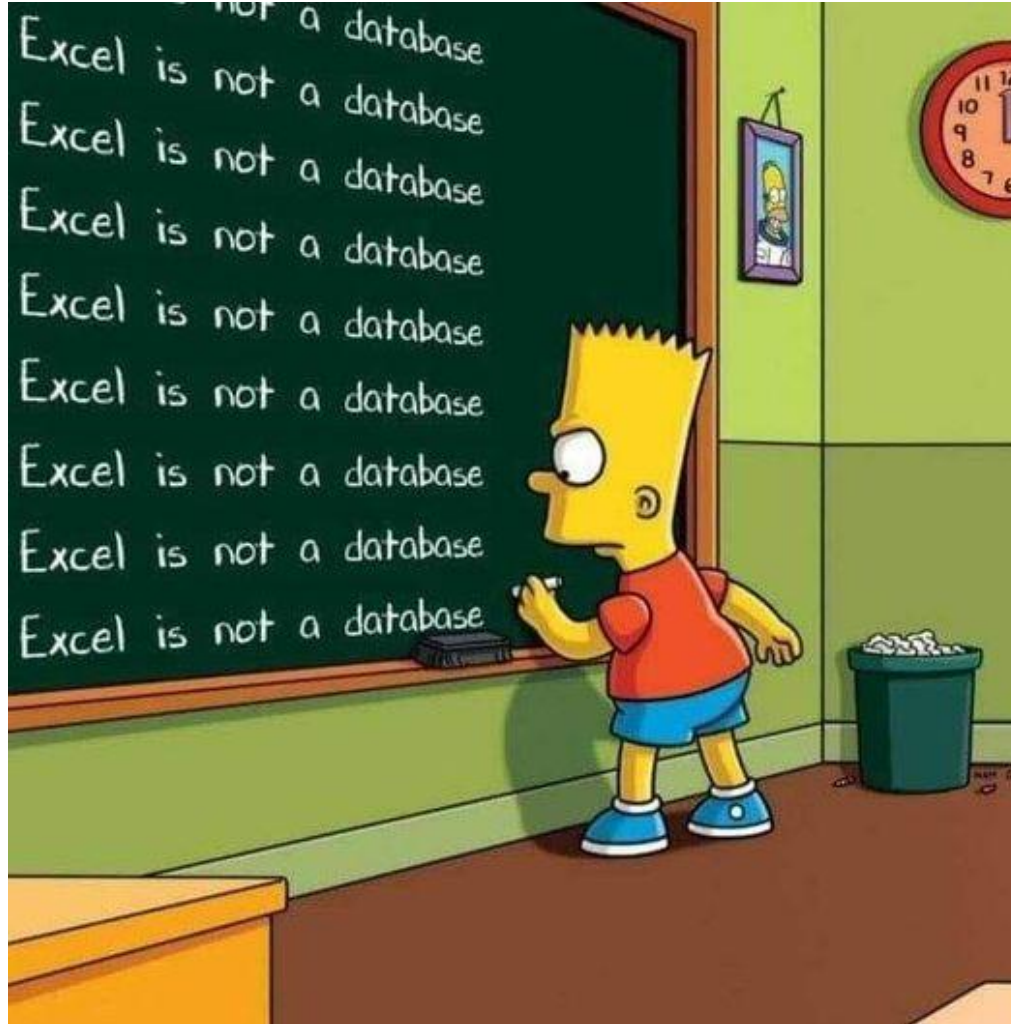
Dentro del esquema sakila, tomemos la tabla actor de película, la cual empareja una película específica con un actor específico:

```
CREATE TABLE film_actor (  
  actor_id SMALLINT UNSIGNED NOT NULL,  
  film_id SMALLINT UNSIGNED NOT NULL,  
  last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (actor_id, film_id),  
  KEY idx_fk_film_id (`film_id`),  
  CONSTRAINT fk_film_actor_actor FOREIGN KEY (actor_id) REFERENCES actor (actor_id) ON DELETE RESTRICT ON UPDATE CASCADE,  
  CONSTRAINT fk_film_actor_film FOREIGN KEY (film_id) REFERENCES film (film_id) ON DELETE RESTRICT ON UPDATE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```



La instrucción se refiere que si se elimina/actualiza un id de alguna de las dos tablas, este cambio se vera reflejada en las tablas participantes.

# Creación de tablas



# Referencias

- Sommerville, I., Software Engineering, 10th Edition, Pearson, 2016, IN, 1292096144, 9781292096148.
- Connolly Thomas M, Database systems : a practical approach to design, implementation and management, 5thed., London : Addison-Wesley, 2010, 9780321523068.
- Perez, C., MySQL para windows y Linux, España, Alfaomega, 2004.
- <https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html>



**Gracias!**  
**Preguntas...**



**Dr. Esteban Castillo Juarez**

**Google academics:**

<https://scholar.google.com/citations?user=JfZpVO8AAAAJ&hl=en>

<https://dblp.uni-trier.de/pers/hd/c/Castillo:Esteban>