

# Construcción de software y toma de decisiones

## TC2005B

**Dr. Esteban Castillo Juarez**

ITESM, Campus Santa Fe

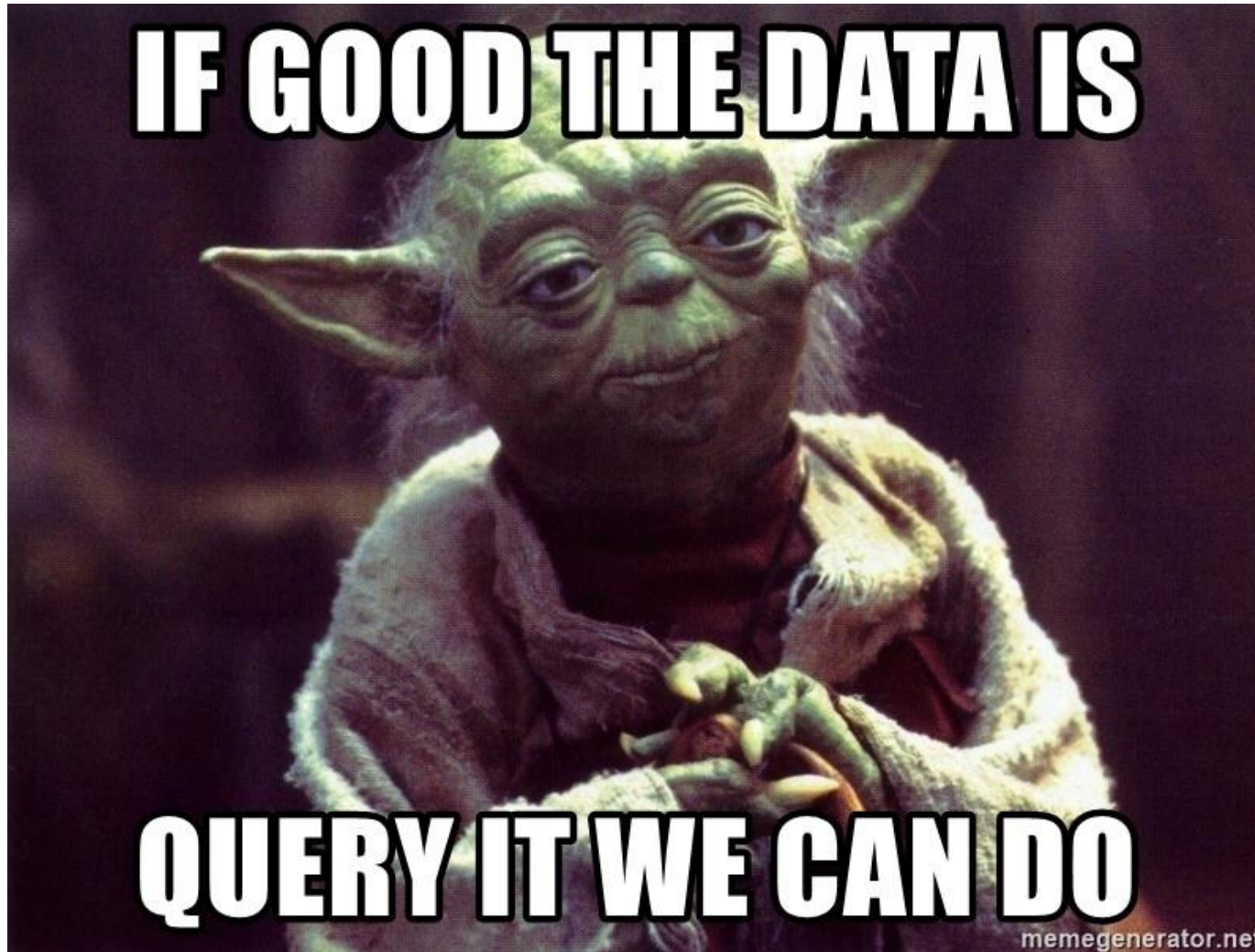


esteban.castillojz@tec.mx

# Agenda

- La clausula INSERT
- La clausula DELETE
- La clausula DELETE con WHERE, ORDER BY y LIMIT
- La clausula UPDATE
- La clausula UPDATE con WHERE, ORDER BY y LIMIT

# La clausula INSERT



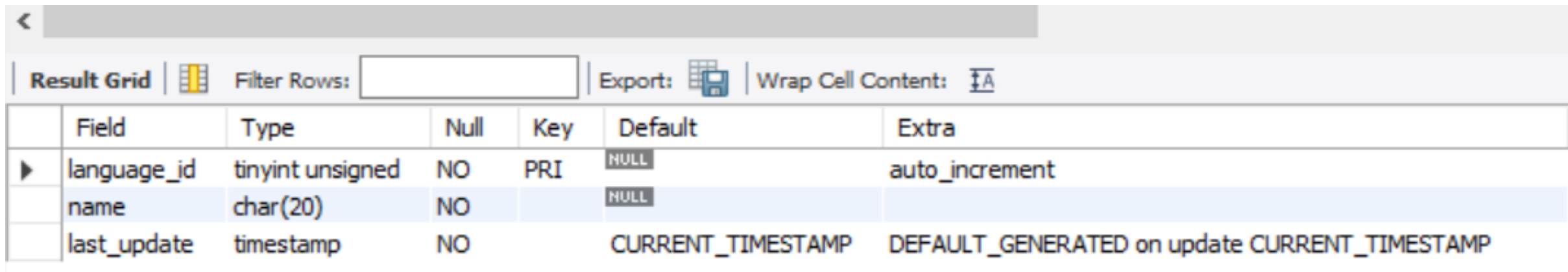
# La cláusula INSERT

- La palabra reservada INSERT se usa para agregar nuevos datos a una tabla existente.
- La inserción de datos generalmente ocurre en dos situaciones: cuando realiza una carga masiva de datos (por medio de script, generalmente) y cuando agrega datos ad-hoc a medida que usa la base de datos.
- Tomando en cuenta lo anterior, comencemos con la tarea básica de insertar una nueva fila en una tabla de la base de datos sakila (la tabla de idioma, language).

# La cláusula INSERT

- Para insertar una nueva fila/registro en la tabla language, necesitamos recordar la estructura de la tabla, para esto usamos la instrucción:

SHOW COLUMNS FROM sakila.language; (SQL)



The screenshot shows a database client interface with a toolbar at the top containing a back arrow, 'Result Grid' button, 'Filter Rows' input, 'Export' button, and 'Wrap Cell Content' button. Below the toolbar is a table with 7 columns: Field, Type, Null, Key, Default, and Extra. The table contains three rows of data for the sakila.language table.

|   | Field       | Type             | Null | Key | Default           | Extra   |
|---|-------------|------------------|------|-----|-------------------|---|
| ▶ | language_id | tinyint unsigned | NO   | PRI | NULL              | auto_increment                                |
|   | name        | char(20)         | NO   |     | NULL              |   |
|   | last_update | timestamp        | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED on update CURRENT_TIMESTAMP |

# La cláusula INSERT

- La anterior instrucción resalta que la columna `language_id` se genera automáticamente, y la columna `last_update` se actualiza cada vez que ocurre una operación de actualización.
- Ahora, agreguemos una nueva fila para el idioma portugués.
- Hay dos maneras de hacer esto. la más común es dejar que MySQL complete el valor predeterminado para `language_id`, de la siguiente manera:

```
INSERT INTO sakila.language VALUES (NULL, 'Portuguese', NOW()); (SQL)
```

Función de MySQL que nos da la estampa de tiempo actual.

# La cláusula INSERT

- Si ejecutamos una consulta SELECT en la tabla language, veremos que MySQL insertó la fila:

```
SELECT * FROM language;(SQL)
```

|   | language_id | name       | last_update         |
|---|-------------|------------|---------------------|
| ▶ | 1           | English    | 2006-02-15 05:02:19 |
|   | 2           | Italian    | 2006-02-15 05:02:19 |
|   | 3           | Japanese   | 2006-02-15 05:02:19 |
|   | 4           | Mandarin   | 2006-02-15 05:02:19 |
|   | 5           | French     | 2006-02-15 05:02:19 |
|   | 6           | German     | 2006-02-15 05:02:19 |
|   | 7           | Portuguese | 2022-03-16 15:21:05 |
| • | NULL        | NULL       | NULL                |

→ Nueva fila añadida a la tabla language

# La cláusula INSERT

- La segunda opción es insertar el valor de la columna language\_id manualmente. Ahora que ya tenemos siete idiomas, debemos usar 8 para el siguiente valor de language\_id. Podemos verificarlo con la instrucción:

```
SELECT MAX(language_id) FROM language; (SQL)
```



Obtiene el valor máximo que esta presente en una columna específica

|             |                  |  |                                   |         |                    |
|-------------|------------------|--|-----------------------------------|---------|--------------------|
| Result Grid |                  |  | Filter Rows: <input type="text"/> | Export: | Wrap Cell Content: |
|             | MAX(language_id) |  |                                   |         |                    |
| ▶           | 7                |  |                                   |         |                    |



# La cláusula INSERT

- La segunda opción es insertar el valor de la columna `language_id` manualmente. Ahora que ya tenemos siete idiomas, debemos usar 8 para el siguiente valor de `language_id`. Podemos verificarlo con la instrucción:

```
SELECT MAX(language_id) FROM language; (SQL)
```

Son operaciones análogas, pero la función `MAX()` es mucho más simple.

```
SELECT language_id FROM language ORDER BY language_id DESC LIMIT 1;
```

# La cláusula INSERT

- Tomando en cuenta lo anterior, estamos listos para insertar y mostrar la fila de manera manual, usando las siguientes instrucciones:

```
INSERT INTO sakila.language VALUES (8, 'Russian', '2020-09-26 10:35:00'); (SQL)  
SELECT * FROM sakila.language;
```

|   | language_id | name       | last_update         |
|---|-------------|------------|---------------------|
| ▶ | 1           | English    | 2006-02-15 05:02:19 |
|   | 2           | Italian    | 2006-02-15 05:02:19 |
|   | 3           | Japanese   | 2006-02-15 05:02:19 |
|   | 4           | Mandarin   | 2006-02-15 05:02:19 |
|   | 5           | French     | 2006-02-15 05:02:19 |
|   | 6           | German     | 2006-02-15 05:02:19 |
|   | 7           | Portuguese | 2022-03-16 15:21:05 |
|   | 8           | Russian    | 2020-09-26 10:35:00 |
| * | NULL        | NULL       | NULL                |

→ Nueva fila añadida a la tabla language

# La cláusula INSERT

- La instrucción INSERT detecta duplicados en la llave principal y detiene cualquier instrucción tan pronto encuentra valores repetidos. Por ejemplo, supongamos que intentamos insertar otra fila con el mismo language\_id:

```
INSERT INTO sakila.language VALUES (8, 'Arabic', '2020-09-26 10:35:00'); (SQL)
```

| Output        |          |   |  |  |
|---------------|----------|---|--|--|
| Action Output |          |   |  |  |
| #             | Time     | Action  | Message  |  |
| 1             | 17:04:09 | INSERT INTO sakila.language VALUES (8, 'Arabic', '2020-09-26 10:35:00') | Error Code: 1062. Duplicate entry '8' for key 'language.PRIMARY' |  |

# La cláusula INSERT

- También es posible insertar múltiples filas a la vez utilizando la instrucción INSERT:

```
INSERT INTO sakila.language VALUES (NULL, 'Spanish', NOW()),(NULL, 'Hebrew', NOW());  
SELECT * FROM sakila.language; (SQL)
```

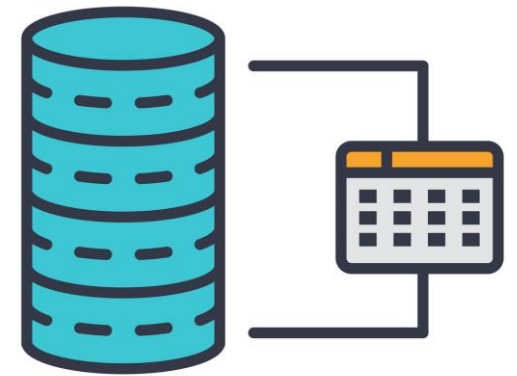
| language_id | name       | last_update         |
|-------------|------------|---------------------|
| 1           | English    | 2006-02-15 05:02:19 |
| 2           | Italian    | 2006-02-15 05:02:19 |
| 3           | Japanese   | 2006-02-15 05:02:19 |
| 4           | Mandarin   | 2006-02-15 05:02:19 |
| 5           | French     | 2006-02-15 05:02:19 |
| 6           | German     | 2006-02-15 05:02:19 |
| 7           | Portuguese | 2022-03-16 15:21:05 |
| 8           | Russian    | 2020-09-26 10:35:00 |
| 9           | Spanish    | 2022-03-20 17:07:01 |
| 10          | Hebrew     | 2022-03-20 17:07:01 |

→ Nueva fila añadida a la tabla language

→ Nueva fila añadida a la tabla language

# La cláusula INSERT

- Hay algunas ventajas en la sintaxis de la cláusula VALUES que hemos estado usando: funciona tanto para inserciones individuales como masivas.
- La sintaxis muestra un mensaje de error si se olvida proporcionar valores para todas las columnas y no se tiene que escribir en nombre de las columnas.
- Sin embargo, también tiene algunas desventajas: debe recordar el orden de las columnas, debe proporcionar un valor para cada columna y la sintaxis está estrechamente relacionada con la estructura de la tabla subyacente.



VectorStock®

VectorStock.com/35629622

# La cláusula INSERT

- Suponga que sabe que la tabla de actores (actors) tiene cuatro columnas y recuerda sus nombres, pero ha olvidado su orden. Puede insertar una fila utilizando el siguiente **enfoque alternativo**:

```
INSERT INTO sakila.actor (actor_id, first_name, last_name,  
last_update) VALUES (NULL, 'Vinicius', 'Grippa', NOW()); (SQL)
```

- El nombre de las columnas se incluyen entre paréntesis después del nombre de la tabla y los valores almacenados en esas columnas se enumeran entre paréntesis después de la palabra clave VALUES.



# La cláusula INSERT

- La sintaxis alternativa permitirle insertar valores solo para algunas columnas. Para entender cómo esto podría ser útil, exploremos la tabla de ciudades (city):

DESC city; (SQL)

|   | Field       | Type              | Null | Key | Default           | Extra   |
|---|-------------|-------------------|------|-----|-------------------|---|
| ► | city_id     | smallint unsigned | NO   | PRI | NULL              | auto_increment                                |
|   | city        | varchar(50)       | NO   |     | NULL              |   |
|   | country_id  | smallint unsigned | NO   | MUL | NULL              |   |
|   | last_update | timestamp         | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED on update CURRENT_TIMESTAMP |

# La cláusula INSERT

- Nótese que la columna `last_update` tiene un valor predeterminado de `CURRENT_TIMESTAMP`.
- Lo anterior significa que si no inserta un valor para la columna `last_update`, MySQL insertará la fecha y la hora actuales de forma predeterminada. Esto es exactamente lo que queremos: cuando almacenamos un registro, no queremos molestarnos en verificar la fecha y la hora y escribirlo.
- Intentemos insertar una fila incompleta:

```
INSERT INTO sakila.city (city, country_id) VALUES ('Bebedouro', 19); (SQL)
```



# La cláusula INSERT

- No establecimos un valor para la columna `city_id`, por lo que MySQL lo establece por defecto con el siguiente valor disponible (debido a la propiedad `auto_increment`), y `last_update` almacena la fecha y la hora actuales.
- Se puede comprobar esto con una consulta:

```
SELECT * FROM sakila.city where city like 'Bebedouro'; (SQL)
```

|   | city_id | city      | country_id | last_update         |
|---|---------|-----------|------------|---------------------|
| ▶ | 601     | Bebedouro | 19         | 2022-03-20 17:29:50 |
|   | 602     | Bebedouro | 19         | 2022-03-20 17:48:00 |
| • | NULL    | NULL      | NULL       | NULL                |

# La cláusula INSERT

- También se puede utilizar este enfoque para la inserción masiva, de la siguiente manera:

```
INSERT INTO sakila.city (city, country_id) VALUES ('Sao Carlos',19),  
          ('Araraquara',19),  
          ('Ribeirao Preto',19);  
SELECT * FROM sakila.city (SQL)
```

- Además de tener que recordar y escribir los nombres de las columnas, una desventaja de este enfoque es que se puede omitir accidentalmente los valores de las columnas. MySQL establecerá las columnas omitidas en los valores predeterminados.

# La cláusula INSERT

- Todas las columnas de una tabla MySQL tienen un valor predeterminado NULL, a menos que se asigne explícitamente otro valor predeterminado cuando se crea o modifica la tabla.
- Cuando necesite usar valores predeterminados para las columnas de la tabla, es posible que desee usar la palabra clave DEFAULT. El siguiente ejemplo agrega una fila a la tabla de países (country) usando DEFAULT:

DESC sakila.country; (SQL)

|   | Field       | Type              | Null | Key | Default           | Extra   |
|---|-------------|-------------------|------|-----|-------------------|---|
| ► | country_id  | smallint unsigned | NO   | PRI | NULL              | auto_increment                                |
|   | country     | varchar(50)       | NO   |     | NULL              |   |
|   | last_update | timestamp         | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED on update CURRENT_TIMESTAMP |

# La cláusula INSERT

```
INSERT INTO sakila.country VALUES (NULL, 'Uruguay', DEFAULT); (SQL)
```

- La palabra clave DEFAULT le dice a MySQL que use el valor predeterminado para esa columna, por lo que la fecha y la hora actuales se insertan en nuestro ejemplo.
- Hay otra sintaxis alternativa para INSERT. En este enfoque, enumera los nombres y valores de las columnas juntos, por lo que no tiene que asignar mentalmente la lista de valores a la lista anterior de columnas. Aquí hay un ejemplo que agrega una nueva fila a la tabla de países:

```
INSERT INTO sakila.country SET country_id=NULL, country='Bahamas',  
last_update=NOW(); (SQL)
```

# La cláusula INSERT

```
INSERT INTO sakila.country SET country_id=NULL, country='Bahamas',  
last_update=NOW(); (SQL)
```

- La sintaxis requiere que muestre un nombre de tabla, la palabra reservada SET y luego pares de columna-valor, separados por comas.
- Las columnas para las que no se proporcionan valores se establecen en sus valores predeterminados.
- Nuevamente, las desventajas son que puede omitir accidentalmente los valores de las columnas y que debe recordar y escribir los nombres de las columnas. Una desventaja adicional significativa es que no puede usar este método para la inserción masiva.

# La cláusula DELETE

- La palabra reservada DELETE se usa para eliminar una o más filas de una tabla.
- El uso más simple de DELETE es eliminar todas las filas de una tabla.
- Suponga que desea vaciar la tabla de rentas (rent). Puedes hacerlo con la siguiente instrucción:

SET SQL\_SAFE\_UPDATES = 0; —————> Eliminar filas sin formato seguro

DELETE FROM renta1; (SQL) —————> Eliminar todas las filas de una tabla

# La cláusula DELETE

`select * from sakila.rental; (SQL)` **Antes**

|   | rental_id | rental_date         | inventory_id | customer_id | return_date         | staff_id | last_update         |
|---|-----------|---------------------|--------------|-------------|---------------------|----------|---------------------|
| ▶ | 1         | 2005-05-24 22:53:30 | 367          | 130         | 2005-05-26 22:04:30 | 1        | 2006-02-15 21:30:53 |
|   | 2         | 2005-05-24 22:54:33 | 1525         | 459         | 2005-05-28 19:40:33 | 1        | 2006-02-15 21:30:53 |
|   | 3         | 2005-05-24 23:03:39 | 1711         | 408         | 2005-06-01 22:12:39 | 1        | 2006-02-15 21:30:53 |
|   | 4         | 2005-05-24 23:04:41 | 2452         | 333         | 2005-06-03 01:43:41 | 2        | 2006-02-15 21:30:53 |
|   | 5         | 2005-05-24 23:05:21 | 2079         | 222         | 2005-06-02 04:33:21 | 1        | 2006-02-15 21:30:53 |
|   | 6         | 2005-05-24 23:08:07 | 2792         | 549         | 2005-05-27 01:32:07 | 1        | 2006-02-15 21:30:53 |
|   | 7         | 2005-05-24 23:11:53 | 3995         | 269         | 2005-05-29 20:34:53 | 2        | 2006-02-15 21:30:53 |
|   | 8         | 2005-05-24 23:31:46 | 2346         | 239         | 2005-05-27 23:33:46 | 2        | 2006-02-15 21:30:53 |

```
SET SQL_SAFE_UPDATES = 0;  
DELETE FROM sakila.rental;
```

`select * from sakila.rental; (SQL)` **Después**

|   | rental_id | rental_date | inventory_id | customer_id | return_date | staff_id | last_update |
|---|-----------|-------------|--------------|-------------|-------------|----------|-------------|
| • | NULL      | NULL        | NULL         | NULL        | NULL        | NULL     | NULL        |

# La cláusula DELETE

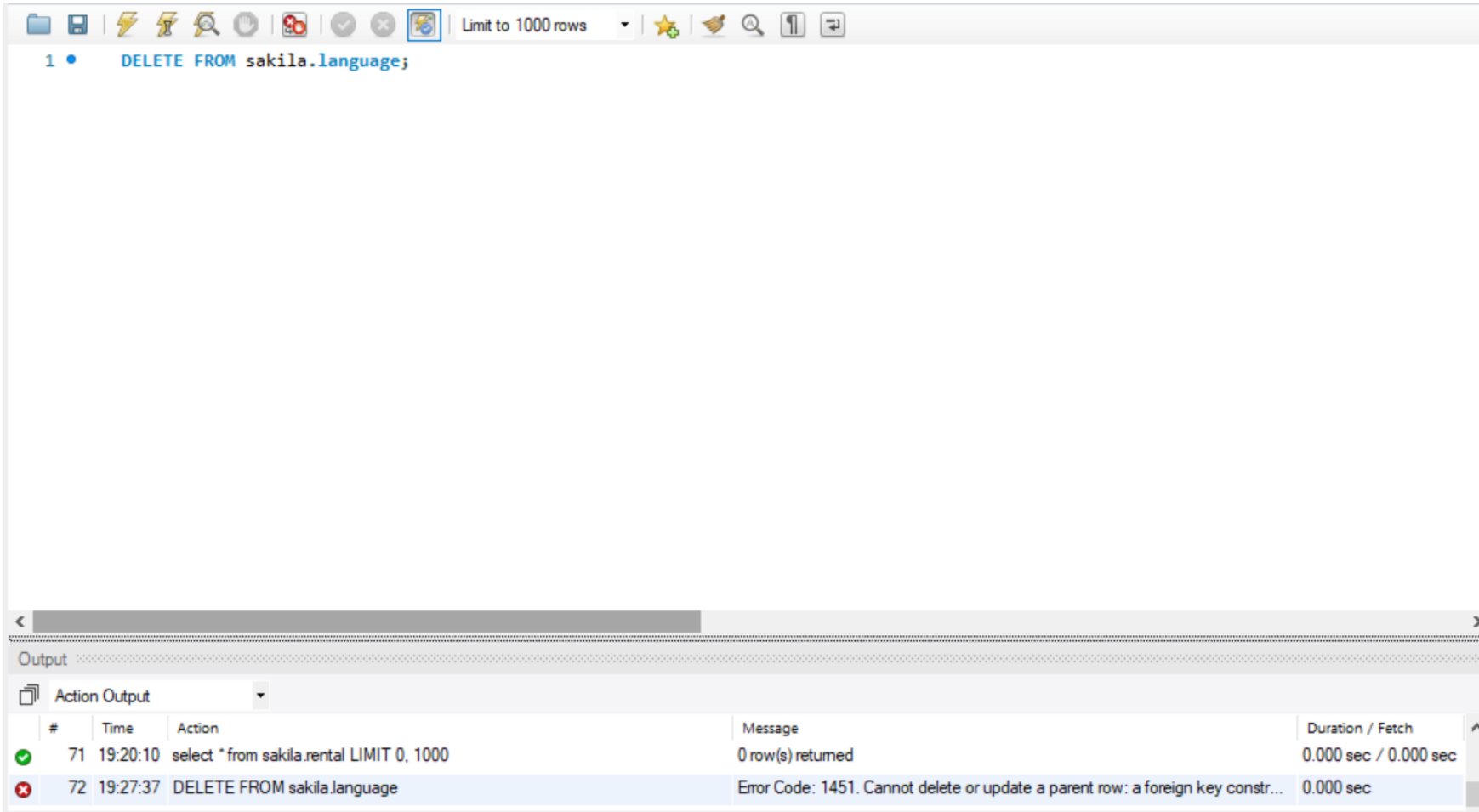
- La sintaxis de la palabra reservada DELETE no incluye nombres de columnas, ya que se usa para eliminar filas completas y no solo valores de una fila.
- Tenga en cuenta que la operación DELETE no elimina la tabla en sí.
- Tenga en cuenta que si la tabla tiene una relación con otra tabla, la eliminación puede fallar debido a la restricción de la clave secundaria o foránea:



DELETE FROM language; (SQL)



# La cláusula DELETE



The screenshot shows a SQL IDE interface. The main editor area contains a single SQL statement: `DELETE FROM sakila.language;`. The toolbar at the top includes icons for file operations, execution, and a dropdown menu set to "Limit to 1000 rows". Below the editor is an "Output" pane with a tab labeled "Action Output". It displays a log of two actions:

| #  | Time     | Action                                    | Message   | Duration / Fetch      |
|----|----------|---|---|-----------------------|
| 71 | 19:20:10 | select * from sakila.rental LIMIT 0, 1000 | 0 row(s) returned   | 0.000 sec / 0.000 sec |
| 72 | 19:27:37 | DELETE FROM sakila.language               | Error Code: 1451. Cannot delete or update a parent row: a foreign key constr... | 0.000 sec             |

# Clausula DELETE con WHERE, ORDER BY y LIMIT

- Para eliminar una o más filas, pero no todas las filas de una tabla, utilice una cláusula WHERE.
- Esto funciona de la misma manera que lo hace para SELECT.
- Por ejemplo, suponga que desea eliminar todas las filas de la tabla de alquiler (rental) con un rental\_id inferior a 10. Puede hacerlo con:

```
DELETE FROM sakila.rental WHERE rental_id < 10; (SQL)
```



# Clausula DELETE con WHERE, ORDER BY y LIMIT

```
SELECT * FROM sakila.rental (SQL)
```

|   | rental_id | rental_date         | inventory_id | customer_id | return_date         | staff_id | last_update         |
|---|-----------|---------------------|--------------|-------------|---------------------|----------|---------------------|
| ▶ | 1         | 2005-05-24 22:53:30 | 367          | 130         | 2005-05-26 22:04:30 | 1        | 2006-02-15 21:30:53 |
|   | 2         | 2005-05-24 22:54:33 | 1525         | 459         | 2005-05-28 19:40:33 | 1        | 2006-02-15 21:30:53 |
|   | 3         | 2005-05-24 23:03:39 | 1711         | 408         | 2005-06-01 22:12:39 | 1        | 2006-02-15 21:30:53 |
|   | 4         | 2005-05-24 23:04:41 | 2452         | 333         | 2005-06-03 01:43:41 | 2        | 2006-02-15 21:30:53 |
|   | 5         | 2005-05-24 23:05:21 | 2079         | 222         | 2005-06-02 04:33:21 | 1        | 2006-02-15 21:30:53 |
|   | 6         | 2005-05-24 23:08:07 | 2792         | 549         | 2005-05-27 01:32:07 | 1        | 2006-02-15 21:30:53 |
|   | 7         | 2005-05-24 23:11:53 | 3995         | 269         | 2005-05-29 20:34:53 | 2        | 2006-02-15 21:30:53 |
|   | 8         | 2005-05-24 23:31:46 | 2346         | 239         | 2005-05-27 23:33:46 | 2        | 2006-02-15 21:30:53 |
|   | 9         | 2005-05-25 00:00:40 | 2580         | 126         | 2005-05-28 00:22:40 | 1        | 2006-02-15 21:30:53 |

```
DELETE FROM sakila.rental WHERE rental_id < 10;
```

```
SELECT * FROM sakila.rental (SQL)
```

|   | rental_id | rental_date         | inventory_id | customer_id | return_date         | staff_id | last_update         |
|---|-----------|---------------------|--------------|-------------|---------------------|----------|---------------------|
| ▶ | 10        | 2005-05-25 00:02:21 | 1824         | 399         | 2005-05-31 22:44:21 | 2        | 2006-02-15 21:30:53 |
|   | 11        | 2005-05-25 00:09:02 | 4443         | 142         | 2005-06-02 20:56:02 | 2        | 2006-02-15 21:30:53 |
|   | 12        | 2005-05-25 00:19:27 | 1584         | 261         | 2005-05-30 05:44:27 | 2        | 2006-02-15 21:30:53 |
|   | 13        | 2005-05-25 00:22:55 | 2294         | 334         | 2005-05-30 04:28:55 | 1        | 2006-02-15 21:30:53 |
|   | 14        | 2005-05-25 00:31:15 | 2701         | 446         | 2005-05-26 02:56:15 | 1        | 2006-02-15 21:30:53 |
|   | 15        | 2005-05-25 00:39:22 | 3049         | 319         | 2005-06-03 03:30:22 | 1        | 2006-02-15 21:30:53 |
|   | 16        | 2005-05-25 00:43:11 | 389          | 316         | 2005-05-26 04:42:11 | 2        | 2006-02-15 21:30:53 |
|   | 17        | 2005-05-25 01:06:36 | 830          | 575         | 2005-05-27 00:43:36 | 1        | 2006-02-15 21:30:53 |
|   | 18        | 2005-05-25 01:10:47 | 3376         | 19          | 2005-05-31 06:35:47 | 2        | 2006-02-15 21:30:53 |

En el resultado se eliminan las nueve filas que coinciden con el criterio.

# Clausula DELETE con WHERE, ORDER BY y LIMIT

- Puede utilizar las palabras reservadas ORDER BY y LIMIT con DELETE.
- Por lo general, hace esto cuando desea limitar el número de filas eliminadas. Por ejemplo:

```
DELETE FROM sakila.payment ORDER BY customer_id LIMIT 10000; (SQL)
```



# Clausula DELETE con WHERE, ORDER BY y LIMIT

```
SELECT * FROM sakila.payment; (SQL)
```

|   | payment_id | customer_id | staff_id | rental_id | amount | payment_date        | last_update         |
|---|------------|-------------|----------|-----------|--------|---------------------|---------------------|
| ▶ | 1          | 1           | 1        | 76        | 2.99   | 2005-05-25 11:30:37 | 2006-02-15 22:12:30 |
|   | 2          | 1           | 1        | 573       | 0.99   | 2005-05-28 10:35:23 | 2006-02-15 22:12:30 |
|   | 3          | 1           | 1        | 1185      | 5.99   | 2005-06-15 00:54:12 | 2006-02-15 22:12:30 |
|   | 4          | 1           | 2        | 1422      | 0.99   | 2005-06-15 18:02:53 | 2006-02-15 22:12:30 |
|   | 5          | 1           | 2        | 1476      | 9.99   | 2005-06-15 21:08:46 | 2006-02-15 22:12:30 |
|   | 6          | 1           | 1        | 1725      | 4.99   | 2005-06-16 15:18:57 | 2006-02-15 22:12:30 |
|   | 7          | 1           | 1        | 2308      | 4.99   | 2005-06-18 08:41:48 | 2006-02-15 22:12:30 |
|   | 8          | 1           | 2        | 2363      | 0.99   | 2005-06-18 13:33:59 | 2006-02-15 22:12:30 |
|   | 9          | 1           | 1        | 3284      | 3.99   | 2005-06-21 06:24:45 | 2006-02-15 22:12:30 |

```
DELETE FROM sakila.payment ORDER BY customer_id LIMIT 10000;  
SELECT * FROM sakila.payment; (SQL)
```

|   | payment_id | customer_id | staff_id | rental_id | amount | payment_date        | last_update         |
|---|------------|-------------|----------|-----------|--------|---------------------|---------------------|
| ▶ | 10001      | 370         | 2        | 4400      | 7.99   | 2005-07-07 21:22:26 | 2006-02-15 22:17:07 |
|   | 10002      | 370         | 2        | 6714      | 0.99   | 2005-07-12 13:29:06 | 2006-02-15 22:17:07 |
|   | 10003      | 370         | 1        | 6968      | 0.99   | 2005-07-27 00:16:45 | 2006-02-15 22:17:07 |
|   | 10004      | 370         | 2        | 7152      | 7.99   | 2005-07-27 07:15:01 | 2006-02-15 22:17:07 |
|   | 10005      | 370         | 1        | 7226      | 6.99   | 2005-07-27 09:47:53 | 2006-02-15 22:17:07 |
|   | 10006      | 370         | 2        | 7797      | 0.99   | 2005-07-28 07:41:07 | 2006-02-15 22:17:07 |
|   | 10007      | 370         | 2        | 8258      | 0.99   | 2005-07-29 01:03:42 | 2006-02-15 22:17:07 |
|   | 10008      | 370         | 2        | 10095     | 0.99   | 2005-07-31 20:38:35 | 2006-02-15 22:17:07 |
|   | 10009      | 370         | 1        | 10336     | 4.99   | 2005-08-01 04:59:53 | 2006-02-15 22:17:07 |

Se eliminan 10000 filas tomando  
como criterio al atributo  
customer\_id

# Clausula DELETE con WHERE, ORDER BY y LIMIT

- Si se desea eliminar todas las filas de una tabla, existe un método más rápido que eliminarlas con la clausula DELETE.
- Cuando se usa la clausula TRUNCATE TABLE, MySQL descarta la tabla de manera mas rápida al eliminar las estructuras de la tabla y luego volver a crearlas.
- Cuando hay muchas filas en una tabla, TRUNCATE es mucho más rápido.



# Clausula DELETE con WHERE, ORDER BY y LIMIT

- Si se desea eliminar todos los datos en la tabla de pagos (payment), se puede ejecutar esto:

```
SELECT * FROM sakila.payment; (SQL)
```

|   | payment_id | customer_id | staff_id | rental_id | amount | payment_date        | last_update         |
|---|------------|-------------|----------|-----------|--------|---------------------|---------------------|
| ► | 10001      | 370         | 2        | 4400      | 7.99   | 2005-07-07 21:22:26 | 2006-02-15 22:17:07 |
|   | 10002      | 370         | 2        | 6714      | 0.99   | 2005-07-12 13:29:06 | 2006-02-15 22:17:07 |
|   | 10003      | 370         | 1        | 6968      | 0.99   | 2005-07-27 00:16:45 | 2006-02-15 22:17:07 |
|   | 10004      | 370         | 2        | 7152      | 7.99   | 2005-07-27 07:15:01 | 2006-02-15 22:17:07 |
|   | 10005      | 370         | 1        | 7226      | 6.99   | 2005-07-27 09:47:53 | 2006-02-15 22:17:07 |
|   | 10006      | 370         | 2        | 7797      | 0.99   | 2005-07-28 07:41:07 | 2006-02-15 22:17:07 |
|   | 10007      | 370         | 2        | 8258      | 0.99   | 2005-07-29 01:03:42 | 2006-02-15 22:17:07 |
|   | 10008      | 370         | 2        | 10095     | 0.99   | 2005-07-31 20:38:35 | 2006-02-15 22:17:07 |
|   | 10009      | 370         | 1        | 10336     | 4.99   | 2005-08-01 04:59:53 | 2006-02-15 22:17:07 |

```
TRUNCATE TABLE sakila.payment;
```

```
SELECT * FROM sakila.payment;
```

|   | payment_id | customer_id | staff_id | rental_id | amount | payment_date | last_update |
|---|------------|-------------|----------|-----------|--------|--------------|-------------|
| * | NULL       | NULL        | NULL     | NULL      | NULL   | NULL         | NULL        |



# Clausula DELETE con WHERE, ORDER BY y LIMIT

La instrucción TRUNCATE TABLE difiere de DELETE en muchos aspectos, pero vale la pena mencionar algunos:

- Las operaciones TRUNCATE eliminan y vuelven a crear la tabla, lo que es mucho más rápido que eliminar filas una por una, especialmente en tablas grandes.
- Las operaciones TRUNCATE provocan una confirmación implícita, por lo que no puede revertirlas.
- No puede realizar operaciones TRUNCATE si la sesión tiene un bloqueo de tabla activo (uso de llaves).

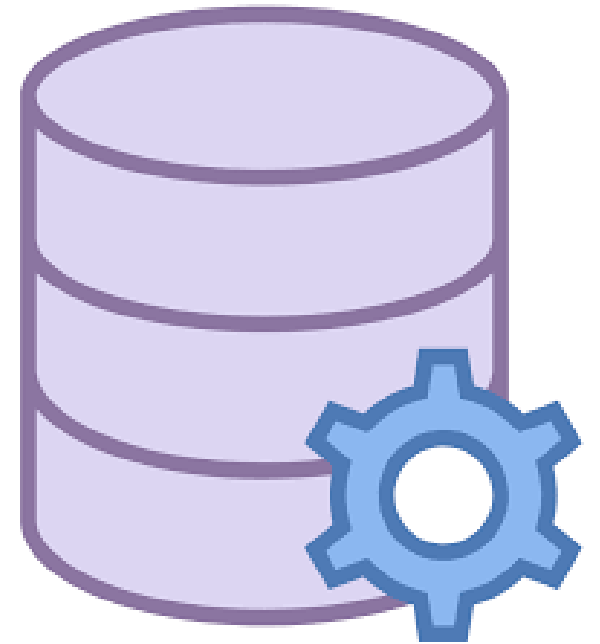




# La cláusula UPDATE

- La instrucción UPDATE se utiliza para cambiar datos.
- El uso más simple de la instrucción UPDATE es cambiar todas las filas de una tabla.
- Suponga que necesita actualizar la columna de monto (amount) de la tabla de pagos (payment) agregando un 10% para todos los pagos. Podrías hacer esto ejecutando:

```
UPDATE sakila.payment SET amount=amount*1.1; (SQL)
```



# La clausula UPDATE

```
SELECT * FROM sakila.payment; (SQL)
```

|   | payment_id | customer_id | staff_id | rental_id | amount | payment_date        | last_update         |
|---|------------|-------------|----------|-----------|--------|---------------------|---------------------|
| ► | 1          | 1           | 1        | 76        | 2.99   | 2005-05-25 11:30:37 | 2006-02-15 22:12:30 |
|   | 2          | 1           | 1        | 573       | 0.99   | 2005-05-28 10:35:23 | 2006-02-15 22:12:30 |
|   | 3          | 1           | 1        | 1185      | 5.99   | 2005-06-15 00:54:12 | 2006-02-15 22:12:30 |
|   | 4          | 1           | 2        | 1422      | 0.99   | 2005-06-15 18:02:53 | 2006-02-15 22:12:30 |
|   | 5          | 1           | 2        | 1476      | 9.99   | 2005-06-15 21:08:46 | 2006-02-15 22:12:30 |
|   | 6          | 1           | 1        | 1725      | 4.99   | 2005-06-16 15:18:57 | 2006-02-15 22:12:30 |
|   | 7          | 1           | 1        | 2308      | 4.99   | 2005-06-18 08:41:48 | 2006-02-15 22:12:30 |
|   | 8          | 1           | 2        | 2363      | 0.99   | 2005-06-18 13:33:59 | 2006-02-15 22:12:30 |
|   | 9          | 1           | 1        | 3284      | 3.99   | 2005-06-21 06:24:45 | 2006-02-15 22:12:30 |

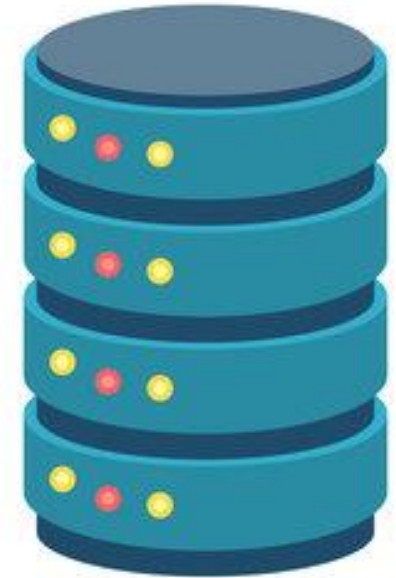
```
UPDATE sakila.payment SET amount=amount*1.1;
```

```
SELECT * FROM sakila.payment; (SQL)
```

|   | payment_id | customer_id | staff_id | rental_id | amount | payment_date        | last_update         |
|---|------------|-------------|----------|-----------|--------|---------------------|---------------------|
| ► | 1          | 1           | 1        | 76        | 3.29   | 2005-05-25 11:30:37 | 2022-03-20 21:24:33 |
|   | 2          | 1           | 1        | 573       | 1.09   | 2005-05-28 10:35:23 | 2022-03-20 21:24:33 |
|   | 3          | 1           | 1        | 1185      | 6.59   | 2005-06-15 00:54:12 | 2022-03-20 21:24:33 |
|   | 4          | 1           | 2        | 1422      | 1.09   | 2005-06-15 18:02:53 | 2022-03-20 21:24:33 |
|   | 5          | 1           | 2        | 1476      | 10.99  | 2005-06-15 21:08:46 | 2022-03-20 21:24:33 |
|   | 6          | 1           | 1        | 1725      | 5.49   | 2005-06-16 15:18:57 | 2022-03-20 21:24:33 |
|   | 7          | 1           | 1        | 2308      | 5.49   | 2005-06-18 08:41:48 | 2022-03-20 21:24:33 |
|   | 8          | 1           | 2        | 2363      | 1.09   | 2005-06-18 13:33:59 | 2022-03-20 21:24:33 |
|   | 9          | 1           | 1        | 3284      | 4.39   | 2005-06-21 06:24:45 | 2022-03-20 21:24:33 |

# La cláusula UPDATE con WHERE, ORDER BY y LIMIT

- A menudo, no desean cambiar todas las filas de una tabla.
- En su lugar, se puede actualizar una o más filas que coincidan con una condición.
- Al igual que con SELECT y DELETE, la cláusula WHERE se usa para la tarea. Además, de la misma manera que con DELETE, puede usar ORDER BY y LIMIT juntos para controlar cuántas filas se actualizan de una lista ordenada.



# La cláusula UPDATE con WHERE, ORDER BY y LIMIT

- Probemos un ejemplo que modifica una fila en una tabla.
- Supongamos que la actriz Penélope Guinness ha cambiado de apellido. Para actualizarlo en la tabla de actores (actors) de la base de datos, debe ejecutar:

```
UPDATE sakila.actor SET last_name= UPPER('cruz') WHERE first_name LIKE 'PENELOPE' AND last_name LIKE 'GUINNESS'; (SQL)
```



# La cláusula UPDATE con WHERE, ORDER BY y LIMIT

```
select * from sakila.actor; (SQL)
```

|   | actor_id | first_name | last_name    | last_update         |
|---|----------|------------|--------------|---------------------|
| ▶ | 1        | PENELOPE   | GUINNESS     | 2006-02-15 04:34:33 |
|   | 2        | NICK       | WAHLBERG     | 2006-02-15 04:34:33 |
|   | 3        | ED         | CHASE        | 2006-02-15 04:34:33 |
|   | 4        | JENNIFER   | DAVIS        | 2006-02-15 04:34:33 |
|   | 5        | JOHNNY     | LOLLOBRIGIDA | 2006-02-15 04:34:33 |
|   | 6        | BETTE      | NICHOLSON    | 2006-02-15 04:34:33 |
|   | 7        | GRACE      | MOSTEL       | 2006-02-15 04:34:33 |
|   | 8        | MATTHEW    | JOHANSSON    | 2006-02-15 04:34:33 |
|   | 9        | JOE        | SWANK        | 2006-02-15 04:34:33 |

Como era de esperar, MySQL realizó la coincidencia de una fila y cambió esa misma.

```
UPDATE sakila.actor SET last_name= UPPER('cruz') WHERE first_name LIKE 'PENELOPE'  
AND last_name LIKE 'GUINNESS';select * from sakila.actor; (SQL)
```

|   | actor_id | first_name | last_name    | last_update         |
|---|----------|------------|--------------|---------------------|
| ▶ | 1        | PENELOPE   | CRUZ         | 2022-03-20 21:38:20 |
|   | 2        | NICK       | WAHLBERG     | 2006-02-15 04:34:33 |
|   | 3        | ED         | CHASE        | 2006-02-15 04:34:33 |
|   | 4        | JENNIFER   | DAVIS        | 2006-02-15 04:34:33 |
|   | 5        | JOHNNY     | LOLLOBRIGIDA | 2006-02-15 04:34:33 |
|   | 6        | BETTE      | NICHOLSON    | 2006-02-15 04:34:33 |
|   | 7        | GRACE      | MOSTEL       | 2006-02-15 04:34:33 |
|   | 8        | MATTHEW    | JOHANSSON    | 2006-02-15 04:34:33 |
|   | 9        | JOE        | SWANK        | 2006-02-15 04:34:33 |

# La cláusula UPDATE con WHERE, ORDER BY y LIMIT

- Para controlar cuántas actualizaciones ocurren, puede usar la combinación de ORDER BY y LIMIT:

Select \* from sakila.payment; (SQL)

|   | payment_id | customer_id | staff_id | rental_id | amount | payment_date        | last_update         |
|---|------------|-------------|----------|-----------|--------|---------------------|---------------------|
| ► | 1          | 1           | 1        | 76        | 3.29   | 2005-05-25 11:30:37 | 2022-03-20 21:24:33 |
|   | 2          | 1           | 1        | 573       | 1.09   | 2005-05-28 10:35:23 | 2022-03-20 21:24:33 |
|   | 3          | 1           | 1        | 1185      | 6.59   | 2005-06-15 00:54:12 | 2022-03-20 21:24:33 |
|   | 4          | 1           | 2        | 1422      | 1.09   | 2005-06-15 18:02:53 | 2022-03-20 21:24:33 |
|   | 5          | 1           | 2        | 1476      | 10.99  | 2005-06-15 21:08:46 | 2022-03-20 21:24:33 |

Select \* from sakila.payment

UPDATE sakila.payment SET last\_update=NOW() LIMIT 10; (SQL)

|   | payment_id | customer_id | staff_id | rental_id | amount | payment_date        | last_update         |
|---|------------|-------------|----------|-----------|--------|---------------------|---------------------|
| ► | 1          | 1           | 1        | 76        | 3.29   | 2005-05-25 11:30:37 | 2022-03-20 21:44:33 |
|   | 2          | 1           | 1        | 573       | 1.09   | 2005-05-28 10:35:23 | 2022-03-20 21:44:33 |
|   | 3          | 1           | 1        | 1185      | 6.59   | 2005-06-15 00:54:12 | 2022-03-20 21:44:33 |
|   | 4          | 1           | 2        | 1422      | 1.09   | 2005-06-15 18:02:53 | 2022-03-20 21:44:33 |
|   | 5          | 1           | 2        | 1476      | 10.99  | 2005-06-15 21:08:46 | 2022-03-20 21:44:33 |

# La cláusula UPDATE con WHERE, ORDER BY y LIMIT

- Al igual que con DELETE, se puede hacer esta operación en pequeños fragmentos o modificar solo algunas filas.
- En el ejemplo anterior se puede ver que 10 filas coincidieron y fueron cambiadas.
- El ejemplo anterior también ilustra un aspecto importante de las actualizaciones. Como se ha visto, las actualizaciones tienen dos fases: una fase de coincidencia, donde se encuentran las filas que coinciden con la cláusula WHERE, y una fase de modificación, donde se actualizan las filas que necesitan cambios.

# La cláusula UPDATE con WHERE, ORDER BY y LIMIT

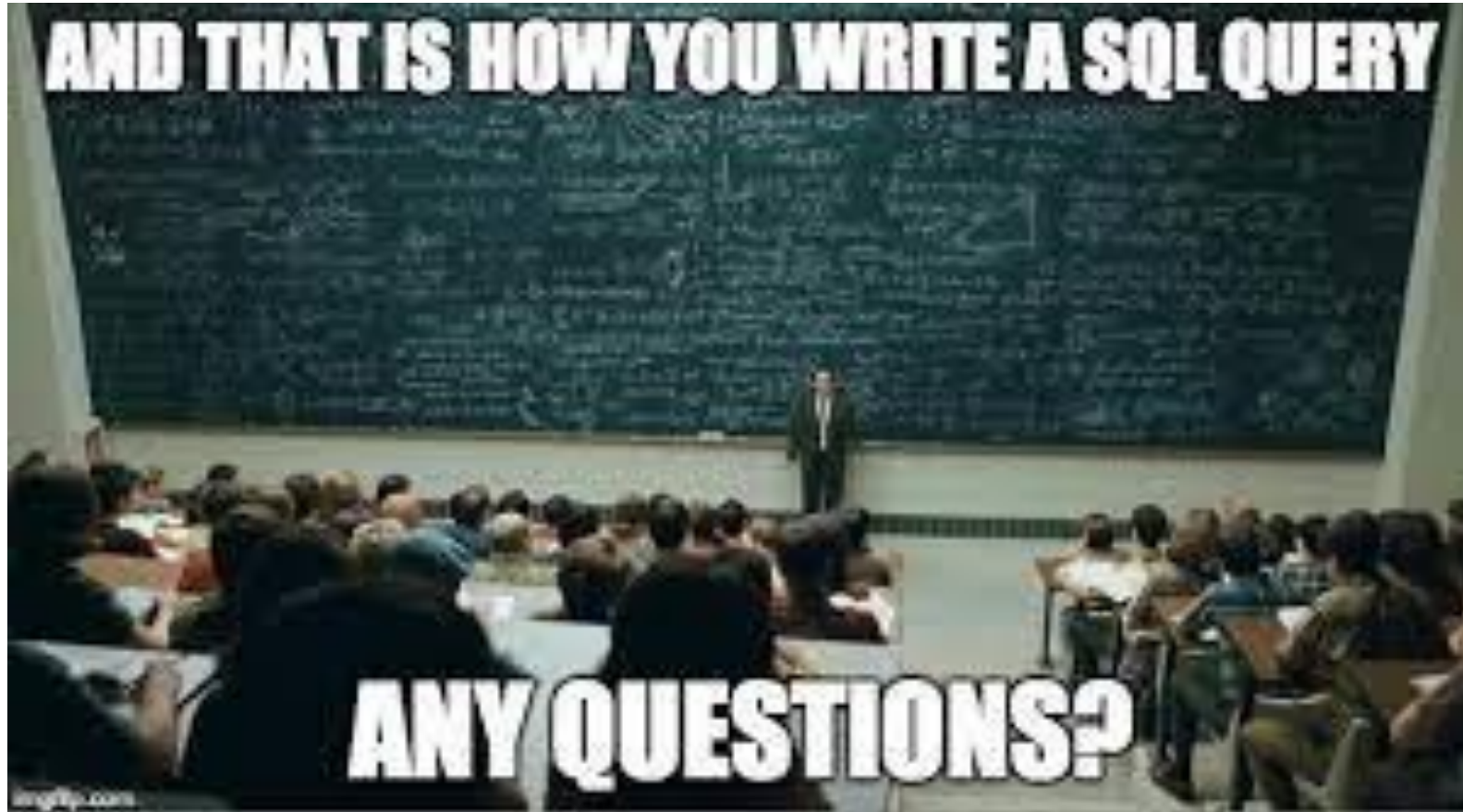
**BE SMART**



**USE CRUD**



# La cláusula UPDATE con WHERE, ORDER BY y LIMIT



# 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>