Práctica #6. Funciones y Triggers

Luis Eduardo Galindo Amaya (1274895)

2023-10-22

1. Sumario

2. Video 1

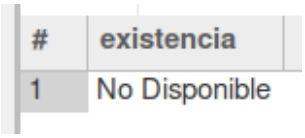
2.1. Funciones

```
DELIMITER %
CREATE FUNCTION fn_existencia_pelicula_tienda (
       p_film_id smallint,
       p_store_id tinyint
    )
    RETURNS varchar (
        13) deterministic
BEGIN
DECLARE
    existencia varchar(13);
    SELECT
        (
            CASE WHEN count(*) > 0 THEN
                'Disponible'
            ELSE
                'No Disponible'
            END) INTO existencia
    FROM
        inventory
    WHERE
        film_id = p_film_id
        AND store_id = p_store_id;
    RETURN existencia;
END %

-- No Disponible
select fn_existencia_pelicula_tienda(839,1) existencia;

-- Disponible
select fn_existencia_pelicula_tienda(839,2) existencia;
```

2.2. Salida del comando



#	existencia
1	No Disponible

Figura 2.2.1: Salida de pelicula 839 en la tienda 1

Result Grid Filter Rows:

#	existencia
1	Disponible

Figura 2.2.2: Salida de pelicula 839 en la tienda 2

3. Video 2

3.1. Trigger before insert

```
DELIMITER %
CREATE TRIGGER before_asignaractivocliente_insert
    BEFORE INSERT ON customer
    FOR EACH ROW
BEGIN
    SET new.active = 1;

END
%

-- Insercion de prueba
INSERT INTO customer (store_id, first_name, last_name, address_id)
    VALUES (1, 'sukey', 'nakasima', 1);

-- Verificar salida
SELECT * FROM customer
ORDER BY customer_id
DESC LIMIT 1;
```

3.2. Salida del comando

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

#	customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
1	601	1	sukey	nakasima	NULL	1	1	2023-10-23 01:03:47	2023-10-23 01:03:47

Figura 3.2.1: El valor de active es igual a 1

3.3. Trigger after insert

```
-- buscar la película de rocky
SELECT * FROM film
WHERE title LIKE "%rock%";

-- agregar la película al inventario
DELIMITER %
CREATE TRIGGER after_agregarinventariopelicula_insert
    AFTER INSERT ON film
    FOR EACH ROW
BEGIN
    INSERT INTO inventory (film_id, store_id)
VALUES
    (new.film_id, 1),
    (new.film_id, 2);
END
%

-- Informacion del film 1
SELECT * FROM film
ORDER BY film_id DESC
LIMIT 1;

-- informacion del inventario
SELECT * FROM inventory
ORDER BY inventory_id DESC
LIMIT 2;
```

3.4. Salida del comando

#	film_id	title	descriptio	release_year	language_id	original_language_i	rental_duratio	rental_rate	length	replacement_cos	rating	spe
1	1001	Rocky V	NULL	NULL	1	1	3	4.99	NULL	19.99	G	NULL

Figura 3.4.1: Informacion del film 1

#	inventory_id	film_id	store_id	last_update
1	4583	1001	2	2023-10-21 18:55:26
2	4582	1001	1	2023-10-21 18:55:26

Figura 3.4.2: informacion del inventario

4. Video 3

4.1. Trigger before update

```
UPDATE staff SET PASSWORD = 'admin'
WHERE staff_id = 1;

DELIMITER $$
CREATE TRIGGER before_verificarcambiospassword_update
    BEFORE UPDATE ON staff FOR EACH ROW
BEGIN
    IF EXISTS
    (
       SELECT 1
       FROM staff
       WHERE staff_id = OLD.staff_id
       AND OLD.password = NEW.password
    )
    THEN
        signal sqlstate '45000'
        SET message_text = 'El password es igual al anterior... Cambialo!';
    END IF;
END
$$
```

4.2. Salida del comando

Action Output

	#	Time	Action	Message
✔	1	01:48:27	use sakila	0 row(s) affected
✖	2	01:48:27	update staff set password='admin' where staff_id=1	Error Code: 1644. El password es igual al anterior... Cambialo!

Figura 4.2.1: Salida

4.3. Trigger after update

```
DELIMITER $$
CREATE TRIGGER after_actualizaramountpayment_update
    AFTER UPDATE ON film
    FOR EACH ROW UPDATE payment AS p
    INNER JOIN rental AS r ON p.rental_id = r.rental_id
    INNER JOIN inventory AS inv ON r.inventory_id = inv.inventory_id
    SET amount = new.rental_rate
WHERE
    inv.film_id = old.film_id;
END
$$

-- actualixar el rate
UPDATE film SET rental_rate = 5.00
WHERE film_id = 80;


-- Mostrar pagos
SELECT p.*
FROM rental r
    INNER JOIN inventory inv
    ON r.inventory_id = inv.inventory_id

    INNER JOIN payment p
    ON p.rental_id = r.rental_id
WHERE inv.film_id = 80;
```

4.4. Salida del comando

Result Grid Filter Rows: Export: Wrap Cell Content:

#	payment_id	customer_id	staff_id	rental_id	amount	payment_date	last_update
1	6545	242	1	4426	5.00	2005-07-07 22:28:32	2023-10-22 14:12:03
2	10514	388	2	9840	5.00	2005-07-31 12:23:18	2023-10-22 14:12:03
3	13376	496	1	13182	5.00	2006-02-14 15:16:03	2023-10-22 14:12:03
4	1059	38	1	11344	5.00	2005-08-02 17:13:26	2023-10-22 14:12:03
5	10827	400	2	11530	5.00	2005-08-17 00:29:00	2023-10-22 14:12:03
6	3735	138	1	8424	5.00	2005-07-29 07:06:03	2023-10-22 14:12:03
7	13460	499	1	14858	5.00	2005-08-22 02:46:18	2023-10-22 14:12:03
8	3504	130	1	1	5.00	2005-05-24 22:53:30	2023-10-22 14:12:03
9	8828	327	2	1577	5.00	2005-06-16 04:03:28	2023-10-22 14:12:03

Result 39

Figura 4.4.1: Salida de mostrar pagos

5. Video 4

5.1. Trigger before delete

```
CREATE TABLE IF NOT EXISTS logeliminarcliente (
    fecha_eliminar timestamp DEFAULT CURRENT_TIMESTAMP,
    usuario_elimino varchar(100),
    customer_id smallint,
    first_name varchar(45),
    last_name varchar(45),
    store_id tinyint,
    active tinyint
);

DELIMITER $$
CREATE TRIGGER before_eliminarcliente_delete
    BEFORE DELETE ON customer
    FOR EACH ROW
BEGIN
    DECLARE usuario varchar(100);
SELECT
    current_user() INTO usuario;

INSERT INTO logeliminarcliente
       (usuario_elimino, customer_id, first_name, last_name, store_id, active)
VALUES
    (usuario, old.customer_id, old.first_name, old.last_name, old.store_id,
old.active);
END $$

-- Prueba
DELETE FROM customer
WHERE customer_id = 600;

SELECT * FROM customer
WHERE customer_id = 600;

SELECT * FROM logeliminarcliente;
```

5.2. Salida del comando

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

#	fecha_eliminar	usuario_elimino	customer_i	first_name	last_name	store_id	active
1	2023-10-22 15:18:34	root@localhost	600	sukey	nakasima	1	1

Figura 5.2.1: Salida de la prueba

6. Trigger after delete

Para probar el trigger pense en mostrar un mensaje si se elimina un registro de la tabla `logeliminarcliente` y usar el mismo codigo para hacer las pruebas:

```
DELIMITER $$
CREATE TRIGGER delete_logeliminarcliente
    AFTER DELETE ON logeliminarcliente
    FOR EACH ROW
BEGIN
    signal sqlstate '45000'
    SET message_text = 'Se ha eliminado los logs';

END;
$$

-- mostrar logs
SELECT * FROM logeliminarcliente;

-- Eliminar el log
DELETE FROM logeliminarcliente
WHERE customer_id = 600;
```

6.1. Salida del comando

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

#	fecha_eliminar	usuario_elimino	customer_id	first_name	last_name	store_id	active
1	2023-10-22 15:18:34	root@localhost	600	sukey	nakasima	1	1

Figura 6.1.1: Usuarios eliminados previamente

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

#	fecha_eliminar	usuario_elimino	customer_id	first_name	last_name	store_id	active
1	2023-10-22 15:18:34	root@localhost	600	sukey	nakasima	1	1

Figura 6.1.2: mostar mesajes que se eliminaron logs