Disparadores y Vistas en SQL

Samuel Martín Morales

November 7, 2023

Contents

Chapter 1

Introducción

Para esta cuarta práctica de la asignatura *Administración y Diseño de Bases de Datos* se solicita el empleo de una base de datos que debe de ser restaurada de manera previa a la implementación de una serie de ejercicios que son demandados haciendo uso de dicha base de datos.

En este caso, la base de datos a emplear se denomina como ***alquilerdvd.tar*** y se encuentra disponible en el campus virtual de la asignatura. Pero, puede ser descargada desde el siguiente enlace de GitHub.

Dicha base de datos se encuentra en formato *.tar* por lo que, para poder restaurarla, se debe de emplear el siguiente comando:

```
$ pg_restore -U postgres -d alquilerdvd alquilerdvd.tar
```

Es decir, el comando anterior restaura la base de datos *alquilerdvd* haciendo uso del fichero *alquilerdvd.tar* y empleando el usuario *postgres*.

Una vez restaurada la base de datos, se puede proceder a la realización de los distintos ejercicios.

Chapter 2

Resultados

2.1 Ejercicio 1

Para el primer ejercicio de la práctica, se deben de identificar las distintas tablas, vistas y secuencias que tiene la base de datos que ha sido restaurada.

Tras la carga de la base de datos a partir del fichero con extensión *.tar*, se puede observar que la base de datos *alquilerdvd* cuenta con un total de 15 tablas, 4 vistas y 15 secuencias. Para poder visualizar todos estos datos comentados sobre la base de datos, se hace uso de la terminal interactiva de *PostgreSQL*, es decir, de *psql*, y, una vez dentro de la base de datos se ejecutan los siguientes comandos para obtener los distintos valores obsevados anteriormente:

```
# \dt  -- Muestra las tablas de la base de datos
# \dv  -- Muestra las vistas de la base de datos
# \ds  -- Muestra las secuencias de la base de datos
```

2.2 Ejercicio 2

Tras la identificación de las distintas tablas más importantes de la base de datos junto con sus atributos y relaciones entre las distintas tablas, se procede a la implementación de distintas consultas que permitan obtener aquella información que es solicitada en el enunciado del ejercicio.

2.2.1 Ventas totales

Para obtener las ventas totales por categoría de películas ordendas de manera descendente, se emplea la siguiente consulta:

```
SELECT COUNT(*) AS total_rent, category.name AS category_name
FROM rental
INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN film ON inventory.film_id = film.film_id
INNER JOIN film_category ON film.film_id = film_category.film_id
INNER JOIN category ON film_category.category_id = category.category_id
GROUP BY category_name
ORDER BY total_rent DESC;
```

2.2.2 Ventas totales por tienda

Para obtener las ventas totales por tienda donde se refleja la ciudad, el país y el encargado, se emplea la siguiente consulta:

```
SELECT COUNT(*) AS total_rent , store.store_id AS store_id, city.city || ', ' || country.country AS cityu_
FROM rental
INNER JOIN inventory on rental.inventory_id = inventory.inventory_id
INNER JOIN store ON inventory.store_id = store.store_id
INNER JOIN staff ON store.manager_staff_id = staff.staff_id
INNER JOIN address ON store.address_id = address.address_id
INNER JOIN city ON address.city_id = city.city_id
INNER JOIN country ON city.country_id = country.country_id
GROUP BY store.store_id, manager_staff_first_name, manager_staff_last_name, city, country
ORDER BY total_rent DESC;
```

Nota: para la consulta anterior se ha empleado una concatenación de cadenas de caracteres para poder obtener la ciudad y el país en una misma columna, para ello, se ha hecho uso de la doble barra vertical que permite establecer la concatenación por pares de elementos, y para el ejemplo de consulta observado anteriormente, se hace uso del separador *","* para realizar esta operación.

2.2.3 Lista de películas

Para obtener una lista de películas junto con sus actores, se emplea la siguiente consulta:

```
SELECT film.film_id, title, description, category.name AS category_name,rental_rate, length, rating ,actor
FROM film
INNER JOIN film_actor ON film.film_id = film_actor.film_id
INNER JOIN actor ON film_actor.actor_id = actor.actor_id
INNER JOIN film_category ON film.film_id = film_category.film_id
INNER JOIN category ON film_category.category_id = category.category_id
ORDER BY film_id;
```

2.2.4 información de los actores

Para obtener información de los distintos actores junto con sus películas existentes en la base de datos, se implementa la siguiente consulta:

```
SELECT actor.actor_id, actor.first_name, actor.last_name,film.title || ' : ' || film.description || ' : '
FROM actor
INNER JOIN film_actor ON actor.actor_id = film_actor.actor_id
INNER JOIN film ON film_actor.film_id = film.film_id
INNER JOIN film_category ON film.film_id = film_category.film_id
INNER JOIN category ON film_category.category_id = category.category_id
GROUP BY actor.actor_id, actor.first_name, actor.last_name, films_made
ORDER BY actor.actor_id;
```

2.3 Ejercicio 3

Implementación de todas las vistas a partir de las consultas realizadas en el ejercicio anterior.

2.3.1 Vista 1: Ventas totales

Para la implementación de la primera vista, se emplea la siguiente consulta:

```
CREATE VIEW total_rent_per_category AS
SELECT COUNT(*) AS total_rent, category.name AS category_name
FROM rental
INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN film ON inventory.film_id = film.film_id
```

```
INNER JOIN film_category ON film.film_id = film_category.film_id
INNER JOIN category ON film_category.category_id = category.category_id
GROUP BY category_name
ORDER BY total_rent DESC;
```

2.3.2 Vista 2: Ventas totales por tienda

Para la implementación de la segunda vista, se emplea la siguiente consulta:

```
CREATE VIEW total_rent_per_store AS
SELECT COUNT(*) AS total_rent , store.store_id AS store_id, city.city || ', ' || country.country AS cityu_
FROM rental
INNER JOIN inventory on rental.inventory_id = inventory.inventory_id
INNER JOIN store ON inventory.store_id = store.store_id
INNER JOIN staff ON store.manager_staff_id = staff.staff_id
INNER JOIN address ON store.address_id = address.address_id
INNER JOIN city ON address.city_id = city.city_id
INNER JOIN country ON city.country_id = country.country_id
GROUP BY store.store_id, manager_staff_first_name, manager_staff_last_name, city, country
ORDER BY total_rent DESC;
```

2.3.3 Vista 3: Lista de películas

Para la implementación de la tercera vista, se emplea la siguiente consulta:

```
CREATE VIEW films_list AS
SELECT film.film_id, title, description, category.name AS category_name,rental_rate, length, rating ,actor
FROM film
INNER JOIN film_actor ON film.film_id = film_actor.film_id
INNER JOIN actor ON film_actor.actor_id = actor.actor_id
INNER JOIN film_category ON film.film_id = film_category.film_id
INNER JOIN category ON film_category.category_id = category.category_id
ORDER BY film_id;
```

2.3.4 Vista 4: Información de los actores

Para la implementación de la cuarta vista, se emplea la siguiente consulta:

```
CREATE VIEW actor_list AS
SELECT actor.actor_id, actor.first_name, actor.last_name,film.title || ' : ' || film.description || ' : '
FROM actor
INNER JOIN film_actor ON actor.actor_id = film_actor.actor_id
INNER JOIN film ON film_actor.film_id = film.film_id
INNER JOIN film_category ON film.film_id = film_category.film_id
INNER JOIN category ON film_category.category_id = category.category_id
GROUP BY actor.actor_id, actor.first_name, actor.last_name, films_made
ORDER BY actor.actor_id;
```

2.4 Ejercicio 4

2.5 Ejercicio 5

2.6 Ejercicio 6

Chapter 3

Conclusiones

Example....

Chapter 4

Bibliografía

Example....