**Memoria Práctica 2.**

Martín Sánchez y César Ramírez. Grupo 1201.

1. **dvdreq**

Este programa permitirá al usuario realizar una serie de consultas sobre la basa de datos dvdrental2.

-dvdreq customer.

El usuario podrá consultar toda la información los clientes registrados en la base de datos.

Esta consulta en SQL sería la siguiente:

*SELECT customer.customer\_id, customer.first\_name,*

*customer.last\_name, customer.create\_date, address.address, address.address2,*

*country.country, city.city, address.district, address.postal\_code*

*FROM customer, address, city, country*

*WHERE customer.address\_id = address.address\_id and*

*address.city\_id = city.city\_id and*

*city.country\_id = country.country\_id and*

*customer.first\_name = ? or*

*customer.last\_name = ?*

Como se pude observar, esta consulta devuelve toda la información sobre el cliente registrado que coincida con el nombre y/o apellido introducido por el usuario. Las “?” serán luego sustituidas por el nombre y apellido del cliente o los clientes que el usuario busca en la base de datos.

En esta consulta accedemos a las tablas de customer para obtener el Id, el nombre, el apellido y su fecha de registro, y a las tablas address, city y country para obtener la dirección completa del cliente en cuestión.

-dvdreq film

El usuario podrá acceder a toda la información sobre las peliculas registradas en la base de datos que coincidan total o parcialmente con el título introducido por el mismo.

Esta consulta en SQL sería:

*SELECT film.film\_id, film.title, film.release\_year,*

*film.length, film.description, language.name*

*FROM film, language*

*WHERE language.language\_id = film.language\_id and*

*film.title LIKE ?*

En esta consulta accedemos a la tabla film para obtener su Id, título, año de estreno, duración y su descripción, y a la tabla language para obtener el idioma de la película.

La “?” será sustituida por el título de la película introducido por el usuario.

Gracias a LIKE podemos devolver todas las peliculas que coincidan parcial o totalmente con el título introducido por el usuario.

-dvreq rent

Gracias a esta consulta el usuario podrá obtener la información sobre las películas alquiladas por un cliente en concreto entre dos fechas introducidas.

En SQL la consulta se realizaría de la siguiente manera:

*SELECT rental.rental\_id, rental.rental\_date, film.film\_id, film.title,*

*staff.staff\_id, staff.first\_name, staff.last\_name, store.store\_id, payment.amount*

*FROM customer, rental, inventory, film, staff, store, payment*

*WHERE customer.customer\_id = rental.customer\_id and*

*rental.inventory\_id = inventory.inventory\_id and*

*inventory.film\_id = film.film\_id and*

*rental.staff\_id = staff.staff\_id and*

*staff.store\_id = store.store\_id and*

*rental.rental\_date >= '%s' and*

*rental.rental\_date <= '%s' and*

*payment.rental\_id = rental.rental\_id and*

*customer.customer\_id = %s*

*ORDER BY rental.rental\_date*

En esta consulta el usuario tendrá que introducir 3 cosas. La primera será el Id del cliente del que se quiere sacar la información, y luego un intervalo de tiempo dado por dos fechas.

Esta consulta imprimirá una lista con muchísima información acerca de los alquileres realizados por el cliente entre esas dos fechas. La lista estará ordenada por la fecha en que se realizó el alquiler.

-dvdreq recommend

Esta consulta imprime una lista con las 3 películas más alquiladas, que pertenezcan a la categoría más alquilada por el cliente, y que este todavía no las haya alquilado.

Esta consulta es bastante compleja, y en SQL se realizaría así:

*CREATE VIEW CountCategory AS*

*SELECT COUNT(\*) AS C, category.category\_id*

*FROM film, film\_category, category, customer, rental, inventory*

*WHERE customer.customer\_id = rental.customer\_id and*

*inventory.inventory\_id = rental.inventory\_id and*

*inventory.film\_id = film.film\_id and*

*film\_category.film\_id = film.film\_id and*

*category.category\_id = film\_category.category\_id and*

*customer.customer\_id = %s*

*GROUP BY category.category\_id*

En esta VIEW obtenemos el número de películas de cada categoría alquiladas por el cliente en cuestión.

*CREATE VIEW MaxCategory AS*

*SELECT category\_id*

*FROM CountCategory*

*WHERE C = (SELECT MAX(C) FROM CountCategory)*

En esta VIEW seleccionamos la categoría más repetida entre las películas alquiladas por el cliente.

*CREATE VIEW NotRentedByCust AS*

*SELECT film.film\_id*

*FROM film, film\_category, MaxCategory*

*WHERE film.film\_id = film\_category.film\_id and*

*film\_category.category\_id = MaxCategory.category\_id and*

*film.film\_id NOT IN (SELECT film.film\_id*

*FROM rental, inventory, film, customer*

*WHERE customer.customer\_id=*

*rental.customer\_id and*

*inventory.inventory\_id = rental.inventory\_id and*

*inventory.film\_id = film.film\_id and*

*customer.customer\_id = %s)*

En esta VIEW localizamos las películas que pertenecen a la categoría más alquilada por el cliente, pero que este aún no ha alquilado.

*CREATE VIEW MostRentedFilms AS*

*SELECT COUNT(\*) AS D, NotRentedByCust.film\_id*

*FROM customer, rental, inventory, NotRentedByCust, MaxCategory,*

*film\_category*

*WHERE customer.customer\_id = rental.customer\_id and*

*inventory.inventory\_id = rental.inventory\_id and*

*inventory.film\_id = NotRentedByCust.film\_id and*

*MaxCategory.category\_id = film\_category.category\_id and*

*NotRentedByCust.film\_id = film\_category.film\_id*

*GROUP BY NotRentedByCust.film\_id*

En esta VIEW contamos el número de veces que las películas pertenecientes a la categoría favorita del cliente han sido alquiladas en general, pero que no lo han sido por el cliente introducido.

*SELECT film.film\_id, film.title*

*FROM film, MostRentedFilms*

*WHERE film.film\_id = MostRentedFilms.film\_id*

*ORDER BY D DESC LIMIT 3*

*DROP VIEW MostRentedFilms*

*DROP VIEW NotRentedByCust*

*DROP VIEW MaxCategory*

*DROP VIEW CountCategory*

Aquí, finalmente, seleccionamos las tres películas más alquiladas que pertenecen a la categoría favorita del cliente introducido, pero que este aún no ha alquilado. Y también eliminaríamos las listas creadas para realizar esta consulta.

1. **dvdrent**

Este programa permite la gestión de un alquiler.

-dvdrent new

Dará de alta una película**,** incluye tanto el alquiler como el pago.

Esta consulta en SQL sería de la siguiente forma:

*SELECT customer.customer\_id*

*FROM customer*

*WHERE customer.customer\_id = %s*

Primeramente chequearíamos si el customer introducido se encuentra en la base de datos.

*SELECT film.film\_id, inventory.inventory\_id*

*FROM film, rental, inventory*

*WHERE film.film\_id = %s and*

*film.film\_id = inventory.film\_id and*

*rental.inventory\_id = inventory.inventory\_id*

Aquí chequeamos si la película que se quiere alquilar existe en la base de datos y si está disponible para ser alquilada.

*SELECT staff.staff\_id*

*FROM store, staff*

*WHERE staff.staff\_id = %s and*

*store.store\_id = %s and*

*staff.store\_id = store.store\_id*

Aquí comprobamos que el staff al que se le pretende alquilar trabaje en la misma tienda donde se pretende alquilar la película.

*SELECT store.store\_id*

*FROM store, customer*

*WHERE store.store\_id = %s and*

*customer.customer\_id = %s and*

*customer.store\_id = store.store\_id*

Aquí comprobamos que la tienda en la que está dado de alta el cliente coincida con la tienda donde pretende alquilar la película.

*INSERT INTO rental (rental\_id, rental\_date, inventory\_id,*

*customer\_id, return\_date, staff\_id, last\_update)*

*VALUES (DEFAULT, GETDATE(), %s, %s, TO\_DATE('20/12/2019',*

*'DD/MM/YYYY'), %s, DEFAULT)*

Una vez chequeado todo lo anterior se registra en la tabla rental un nuevo alquiler de película.

*SELECT rental.rental\_id*

*FROM rental*

*ORDER BY rental.rental\_id DESC*

Cada alquiler lleva adsociado un pago, luego también es necesario introducir el pago asociados al alquiler que acabamos de hacer en la tabla payment.

Para ello necesitamos el rental\_id, que es generado automáticamente por la base de datos al registrar un nuevo alquiler. Lo que hacemos en esta consulta es obtener el rental\_id del último alquiler que se ha realizado.

*INSERT INTO payment (payment\_id, customer\_id, staff\_id,*

*rental\_id, amount, payment\_date)*

*VALUES (DEFAULT, %s, %s, %s, %d, GETDATE())*

Por último introducimos a la tabla payment toda la información sobre el pago del alquiler que acabamos de realizar.

-dvdrent remove

Da de baja un alquiler de una película. Esto requerirá eliminar el alquiler y el pago asociado a él.

*SELECT rental.rental\_id*

*FROM rental ,customer, inventory*

*WHERE rental.rental\_id = %s and*

*rental.inventory\_id = inventory.inventory\_id and*

*inventory.film\_id = film.film\_id*

Esta primera consulta si el alquiler que deseamos eliminar se ha hecho alguna vez o si no se ha hecho nunca.

*DELETE FROM rental*

*WHERE rental.rental\_id = %s*

Una vez chequeado que el alquiler se ha hecho, lo eliminamos de la tabla ren tabla.

*DELETE FROM payment*

*WHERE payment.rental\_id = %s*

Finalmente eliminamos el pago asociado al alquiler que acabamos de eliminar.

1. **dvdfilm**

Este programa permitirá borrar una película de la base de datos.

La consulta en SQL sería la siguiente:

*SELECT film.film\_id*

*FROM film*

*WHERE film.film\_id = %s*

En esta consulta chequeamos sin la película está en la base de datos.

*DELETE FROM film\_actor*

*WHERE film\_actor.film\_id = %*

*DELETE FROM film\_category*

*WHERE film\_category.film\_id = %s*

*DELETE FROM film*

*WHERE film.film\_id = %s*

*DELETE FROM inventory*

*WHERE inventory.film\_id = %s*

Aquí lo que hacemos es eliminar la película de todas las tablas en las que se encontraba, de esta forma ya no estará en la basse de datos.