**Base de datos 1**

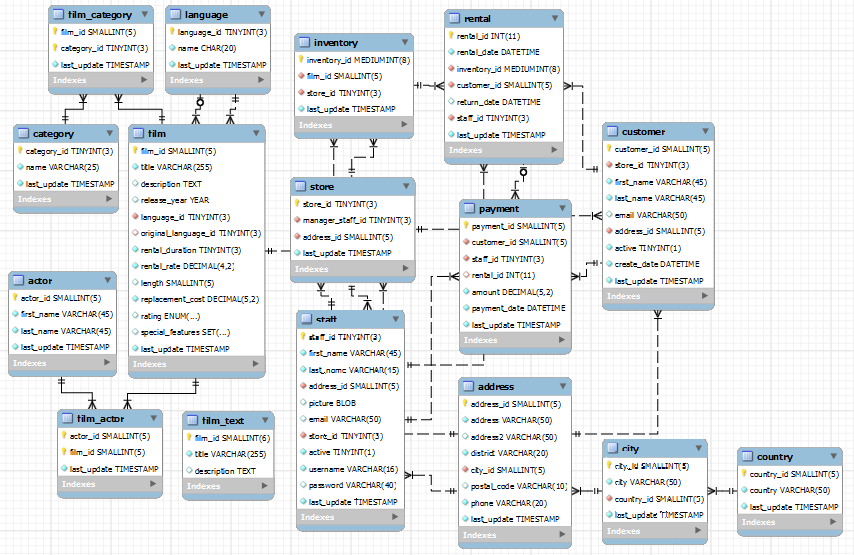
Alumno: Santiago Vietto

Docente: Federico Luis Garofalo

DNI: 42654882

Institución: UCC

Año: 2021



**Ejercicio 3: Gestión de un Video Club**

**Actividades:**

1)\_ Obtener una lista completa de las películas que no se encuentren en stock.

\_ Solución con left join:

**SELECT** \*

**FROM** film

**LEFT** **JOIN** inventory **ON** inventory.film\_id = film.film\_id

**WHERE** inventory.film\_id **IS** NULL

\_ Para saber si la película está registrada pero no existe en inventario, esto implica que no tenemos ninguna tupla en la tabla de inventario que apunte a una película, tenemos que aplicar un RIGHT o LIEFT JOIN. Necesitamos traer las películas que no tienen inventario.

\_ Solución alternativa con right join, es lo mismo que el anterior solo que invertimos lo que está en el from con lo que está en el join:

**SELECT** \*

**FROM** inventory

**RIGHT JOIN** film **ON** inventory.film\_id = film.film\_id

**WHERE** inventory.film\_id **IS** NULL

2)\_ A partir de una única consulta responder: ¿Existen películas que no estén vinculadas con ningún actor?

**SELECT** \*

**FROM** film

**LEFT JOIN** film\_actor **ON** film\_actor.film\_id = film.film\_id

**WHERE** film\_actor.film\_id **IS** NULL

\_ Para unir los actores con las películas usamos la tabla pasarela. Todas las películas que tengan actores vinculados las vamos a encontrar si o si en la tabla pasarela film\_actor, para saber que películas no tienen actores registrados necesitamos solamente saber de qué existe en la tabla film pero no en la tabla pasarela. Si quisiéramos usar un right join, en el WHERE va film\_actor y en el RIGHT JOIN va film, es decir invertimos el orden.

3)\_ Obtener nombre y apellido de aquellos clientes que hayan alquilado la misma película al menos en dos ocasiones. El resultado de la consulta debe incluir además el título de la película y la cantidad de veces que fue alquilada por dicho cliente.

**SELECT** customer.first\_name, customer.last\_name, film.title, **COUNT** (film.film\_id) **AS** cant

**FROM** customer

**INNER JOIN** rental **ON** customer.customer\_id = rental.customer\_id

**INNER JOIN** inventory **ON** rental.inventory\_id = inventory.inventory\_id

**INNER JOIN** film **ON** inventory.film\_id = film.film\_id

**GROUP BY** film.film\_id, customer.customer\_id

**HAVING** cant >= 2

**ORDER BY** cant **DESC**

\_ Leyendo la consigna vemos que las entidades son customer y film, pero el camino para llegar de uno a otro incluye rental e inventory, entonces en orden de recorrido seria customer, rental, inventory y film. Hacemos dos agrupaciones, porque tenemos que verificar que sea el mismo cliente y la misma película para saber la cantidad de veces que cada cliente alquilo cada película. Si agrupamos solamente por customer, nos va a dar la cantidad de alquileres que hizo cada customer, y si además agrupamos por film vamos a tener grupos donde coincide que sean rentas de cada película. Entonces si agrupamos por film\_id y hacemos un COUNT nos quedamos con la cantidad de veces que se alquiló esa película. Por último el HAVING que muestra el resultado siempre y cuando la cantidad sea mayor a dos. Por la regla de MySQL Server customer.first\_name, customer.last\_name, film.title pueden y deben estar incluidos en el GROUP BY.

4)\_ Obtener todos los datos de las películas cuyo stock sea de exactamente 2 ejemplares.

**SELECT** film.\*, **COUNT** (film.film\_id) **AS** stock

**FROM** film

**INNER JOIN** inventory **ON** film.film\_id = inventory.film\_id

**GROUP BY** film.film\_id

**HAVING** stock = 2

\_ En este caso para saber cuántas copias o cuanto stock hay de una película tal, tenemos que unir la tabla film con la tabla inventory y agrupar por film. Traemos todos los datos de las películas, y contamos cuantas veces se repite un ID, por ende en el HAVING ponemos como condición que solo muestre aquellas películas que tengan dos ejemplares en el inventario. A modo de resolución alternativa, podemos hacerlo cambiando lo que hay en el FROM por lo que hay en el INNER JOIN , contando y agrupando por inventory.film\_id.

5)\_ Obtener una tabla a través de la cual se pueda responder cuántos alquileres fueron realizados en cada sucursal del videoclub y cuál es el total recaudado en cada de ellas. Presentar los datos solicitados como “cant\_alquileres” y “ganancia” respectivamente.

**SELECT** store.sotre\_id, **COUNT** (rental.rental\_id) **AS** ‘cant\_alquileres’, **SUM** (payment.amount) **AS** ‘ganancia’

**FROM** payment

**INNER JOIN** rental **ON** payment .rental\_id = rental.rental\_id

**INNER JOIN** staff **ON** rental.staff\_id = staff.staff\_id

**INNER JOIN** store **ON** staff.store\_id = store.store\_id

**GROUP BY** store.store\_id

\_ Determinamos las entidades y vemos que del payment podemos llegar a la renta, porque tenemos el pago de una renta, de la renta tenemos el staff que es la persona que efectuó la renta, y a su vez ese staff trabaja en un store. Este GROUP BY es la cantidad de alquileres por sucursal. A modo de solución alternativa, podemos poner en el FROM rental y en un INNER JOIN ponemos payment donde relacionamos.

6)\_ Obtener mediante una única consulta todos los actores que hayan actuado en más de 3 películas de la categoría ‘Comedy’. Mostrar los actores que hayan actuado en más películas de este género primero.

**SELECT** actor.\* **COUNT** (film\_actor.film\_id) **AS** cant

**FROM** actor

**INNER JOIN** film\_actor **ON** actor.actor\_id = film\_actor.actor\_id

**INNER JOIN** film\_category **ON** film\_actor.film\_id = film\_category.film\_id

**INNER JOIN** category **ON** film\_category.category\_id = category.category\_id

**WHERE** category.name **LIKE** “COMEDY”

**GROUP BY** actor.actor\_id

**HAVING** cant > 3

**ORDER BY** cant **DESC**

\_ En este caso cuando leemos a una única consulta nos referimos a no hacer subconsultas. Identificamos las entidades y son actor, film\_actor, film, film\_category, y category, pero de la tabla pasarela film\_actor podemos hacer un salto entregándole el film\_id a la tabla pasarela film\_category, pero no hay problema. Además tenemos que agrupar por el ID del actor que actuó en todas las películas de la categoría Comedy. Mostramos en orden de mayor a menor porque nos pide que mostremos los actores que más actuaron primero.

7)\_ Obtener el título de las 10 películas que más ganancia hayan generado a la empresa y la ganancia total acumulada por alquileres de cada una de ellas. Ordenar los resultados de mayor a menor.

**SELECT** film.title, **SUM** (payment.amount) **AS** “Ganancia\_total”

**FROM** film

**INNER JOIN** inventory **ON** film.film\_id = inventory.film\_id

**INNER JOIN** rental **ON** inventory.inventory\_id = rental.inventory\_id

**INNER JOIN** payment **ON** rental.rental\_id = payment.rental\_id

**GROUP BY** film.film\_id, film.title

**ORDER BY** **SUM** Ganancia\_total **DESC**

**LIMIT** 10

\_ Primero determinamos las entidades, podemos ver que necesitamos las películas y el monto de cada alquiler, por ende el recorrido seria de film, pasando por inventory, luego rental, y por último payment. Ordenamos por título en este caso, ya que es lo que nos pide nada más. Sumamos los amount de cada película, y los ordenamos de mayor a menor, para solo mostrar los 10 que más monto tienen. A modo de solución alternativa podemos empezar el recorrido al revés por payment hasta film.

8)\_ Obtener el título de las películas cuya ganancia promedio sea menor al promedio de ganancia de todas las películas.

**SELECT** film.title, **AVG**( payment.amount ) **AS** ‘Ganancia\_promedio’

**FROM** film

**INNER JOIN** inventory **ON** film.film\_id = inventory.film\_id

**INNER JOIN** rental **ON** inventory.inventory\_id = rental.inventory\_id

**INNER JOIN** payment **ON** rental.rental\_id = payment.rental\_id

**GROUP BY** film.film\_id, film.title

**HAVING** Ganancia\_promedio < **ALL** (

**SELECT** **AVG** ( payment.amount )

**FROM** payment

**INNER** **JOIN** rental **ON** payment.rental\_id = rental.rental\_id

**INNER** **JOIN** inventory **ON** rental.inventory\_id = inventory.inventory\_id

**GROUP** **BY** inventory.film\_id )

\_ Notamos que las entidades son film y payment, por lo que el recorrido es de film, pasando por inventory, luego rental, y por último payment. Mediante el HAVING y una subconsulta comparamos que el promedio del amount Ganancia\_promedio sea menor al promedio de ganancia de todas las películas que es lo que se muestra en la subconsulta, donde agrupamos por ID de cada película en el inventario. A modo de solución alternativa podemos empezar el recorrido al revés por payment hasta film, y en la sub consulta de inventory a payment.