**MEMORIA PRACTICA 1**

Pedro Sánchez de la Muela

Jorge Fernández García

Esquema de la base de datos:

actor(**actor\_id**, first\_name, last\_name, last\_update)

address(**address\_id**, address, address2, district, city\_id-->city.city\_id, postal\_code, phone, last\_update)

category(**category\_id**, name, last\_update)

city(**city\_id**, city, country\_id--> country.country\_id, last\_update)

country(**country\_id**, country, last\_update)

customers(**customers\_id**, store\_id, first\_name, last\_name, email, address\_id--> address.address\_id, activebool, create\_date, last\_update, active)

film(**film\_id**, title, description, release\_year, language\_id-->language.language\_id, rental\_duration, rental\_rate, length, replacement\_cost, rating, last\_update, special\_features, fulltext)

film\_actor(**actor\_id**-->actor.actor\_id, **film\_id** --> film.film\_id, last\_update)

film\_category(**film\_id**--> film.film\_id, **category\_id**--> category.category\_id, last\_update)

inventory(**inventory\_id**, film\_id--> film.film\_id, store\_id, last\_update)

language(**language\_id**, name, last\_update)

payment(**payment\_id**, customers\_id--> customers.customers\_id, staff\_id--> staff.staff\_id, rental\_id--> rental.rental\_id, ammount, payment\_date)

rental(**rental\_id**, rental\_date, inventory\_id--> inventory.inventory\_id, customer\_id -->customer.customer\_id, return\_date, staff\_id--> staff.staff\_id, last\_update)

staff(**staff\_id**, first\_name, last\_name, address\_id-->address.address\_id, email, store\_id, active, username, password, last\_update, picture)

store(**store\_id**, manager\_staff\_id->staff.staff\_id, address\_id-->address.address\_id, last\_update)

Ejercicio 4: Rediseño de la base de datos:

Para solucionar el problema basta con crear una tabla intermedia entre staff y store donde se almacene la id del trabajador la tienda donde trabaja o ha trabajado y las fechas de entrada y salida de la tienda, además de la columna last\_update. De esta forma si deja de trabajar en una tienda y entra en otra, se escribe la fecha en la que dejó la tienda y se añade una nueva entrada en la tabla con los datos actualizados (tienda, entrada, salida). En el caso de que el empleado siga trabajando en una tienda se puede indicar por la fecha 0001-01-01. El diagrama actualizado se adjunta en la próxima página.

El código sql para modificar la base de datos es el siguiente:

ALTER TABLE public.staff DROP COLUMN store\_id CASCADE;

CREATE TABLE staff\_in\_store(

staff\_id INTEGER NOT NULL,

store\_id INTEGER NOT NULL,

fecha\_entrada DATE NOT NULL,

fecha\_salida DATE NOT NULL,

last\_update timestamp without time zone DEFAULT now(),

FOREIGN KEY (staff\_id) REFERENCES staff(staff\_id ),

FOREIGN KEY (store\_id) REFERENCES store(store\_id),

PRIMARY KEY (staff\_id , store\_id));

CONSULTAS:

CONSULTA 1:

SELECT *Count* (\*),

**Extract** (year FROM rental\_date)

FROM rental

GROUP BY **Extract** (year FROM rental\_date)

ORDER BY 1;

La consulta debe devolver el número de películas alquiladas cada año.

Ya que la fecha de alquiler se encuentra en rental, es suficiente contar las filas de rental y agruparlas según la fecha (año).

CONSULTA 2:

DROP VIEW IF EXISTS rents;

CREATE VIEW rents

AS

(SELECT **COUNT**(\*) peliculas,

customer.customer\_id

FROM customer,

rental

WHERE customer.customer\_id = rental.customer\_id

GROUP BY customer.customer\_id);

SELECT peliculas,

customer.customer\_id,

customer.first\_name,

customer.last\_name

FROM rents

natural JOIN customer

WHERE peliculas = (SELECT **Max**(peliculas)

FROM rents);

Para esta consulta es necesario crear una vista donde se almacena el número de películas que ha alquilado cada cliente. Luego basta con seleccionar el máximo de la columna películas (suma de las películas alquiladas) de la vista rents para obtener el máximo. De esta forma, en caso de empate la consulta imprime a todos los que hayan alquilado el máximo.

CONSULTA 3:

SELECT DISTINCT city\_id,

city.city

FROM city

WHERE city\_id IN (SELECT city\_id

FROM address

WHERE address\_id IN (SELECT address\_id

FROM customer

WHERE customer\_id IN (SELECT

customer\_id

FROM rental

WHERE

inventory\_id IN (SELECT

inventory\_id

FROM

inventory

WHERE

film\_id IN (SELECT film\_id

FROM film\_actor,

actor

WHERE

film\_actor.actor\_id =

actor.actor\_id

AND first\_name='Bob' AND

last\_name='Fawcett')))))

ORDER BY city.city;

Para esta consulta es necesario recorrer muchas tablas. Como nos piden las ciudades donde se han alquilado películas de Bob Fawcett. Analizándolo de dentro a fuera, comenzamos obteniendo los id de las películas en las que aparece el actor, luego los inventory\_id de las películas (pueden haber varios por película), desde rental seleccionamos los customer\_id que han alquilado los inventory\_id, a continuación obtenemos las direcciones de los clientes que hayan efectuado el alquiler y por último obtenemos las ciudades.

Otra posible manera habría sido enlazando las tablas mediante joins hasta llegar a actor y establecer la condición, pero esta nos pareció más gráfica y entendible.

CONSULTA 4:

DROP VIEW IF EXISTS pelis\_mismo\_idioma;

CREATE VIEW pelis\_mismo\_idioma

AS

SELECT **COUNT**(\*) AS num\_pelis,

language\_id

FROM film

GROUP BY language\_id;

SELECT LANGUAGE.name

FROM LANGUAGE

WHERE language\_id IN (SELECT language\_id

FROM pelis\_mismo\_idioma

WHERE num\_pelis = (SELECT **Max**(num\_pelis)

FROM pelis\_mismo\_idioma));

Para obtener el lenguaje en el que están la mayor parte de las películas hace falta crear una vista donde se seleccionan el número de películas que hay en cada lenguaje, agrupandolas por language\_id. Por ultimo se selecciona el language.name donde su language\_id esté en una subconsulta donde se escoge el language\_id de máximo numero de peliculas en un mismo idioma.

CONSULTA 5:

DROP VIEW IF EXISTS num\_pelis;

CREATE VIEW num\_pelis

AS

(SELECT language\_id,

**Sum**(rents\_por\_peli) rents\_por\_lang

FROM (SELECT **Sum**(rents) rents\_por\_peli,

film\_id

FROM (SELECT **COUNT**(rental\_id) rents,

inventory.inventory\_id,

film\_id

FROM rental,

inventory

WHERE rental.inventory\_id = inventory.inventory\_id

GROUP BY inventory.inventory\_id

ORDER BY film\_id) T

GROUP BY film\_id

ORDER BY film\_id) Q

natural JOIN film

GROUP BY language\_id);

SELECT LANGUAGE.name

FROM LANGUAGE

natural JOIN num\_pelis

WHERE rents\_por\_lang = (SELECT **Max**(rents\_por\_lang)

FROM num\_pelis);

Para esta consulta es necesario seleccionar el lenguaje en el que se producen un mayor número de alquileres. Para ello creamos una vista en la que, de dentro a fuera, seleccionamos la suma de alquileres agrupados por inventory\_id, luego sumamos los inventory\_id que pertenezcan a la misma película y por último sumamos las películas alquiladas que tengan el mismo language\_id.

Una vez tenemos el número de alquileres por idioma hay que seleccionar el máximo tal y como se ha hecho en otras consultas anteriormente.

CONSULTA 6:

DROP VIEW IF EXISTS rents CASCADE;

CREATE VIEW rents

AS

(SELECT **COUNT**(\*) peliculas,

customer.customer\_id

FROM customer,

rental

WHERE customer.customer\_id = rental.customer\_id

GROUP BY customer.customer\_id);

DROP VIEW IF EXISTS info\_cat;

CREATE VIEW info\_cat

AS

(SELECT **SUM**(suma\_alquiler) suma\_category,

category\_id

FROM (SELECT **COUNT**(rental\_id) suma\_alquiler,

inventory.inventory\_id,

film\_id

FROM rental,

inventory

WHERE rental.inventory\_id = inventory.inventory\_id

AND rental.customer\_id = (SELECT customer.customer\_id

FROM rents

natural JOIN customer

WHERE peliculas = (SELECT **Max**(

peliculas)

FROM rents))

GROUP BY inventory.inventory\_id

ORDER BY film\_id) T

natural JOIN film\_category,

inventory,

rental

WHERE T.film\_id = inventory.film\_id

AND inventory.inventory\_id = rental.inventory\_id

AND rental.customer\_id = (SELECT customer.customer\_id

FROM rents

natural JOIN customer

WHERE peliculas = (SELECT **Max**(peliculas)

FROM rents))

GROUP BY category\_id

ORDER BY category\_id);

SELECT category\_id,

category.name

FROM info\_cat

natural JOIN category

WHERE suma\_category = (SELECT **Max**(suma\_category)

FROM info\_cat);

Para esta consulta se puede reutilizar partes de la consulta anterior.

Con la vista rents seleccionamos el número de alquileres por cliente.

En la vista info\_cat se seleccionan el número de pelis de cada categoría que ha alquilado el cliente que más ha comprado. Primero se escogen el número de veces que se ha alquilado cada inventory\_id, como sabemos la película asociada a cada inventory, mediante la función SUM calculamos el número de alquileres agrupados por category\_id. De esta forma, como se usa el inventory\_id para calcular la suma, si alquila varias veces la misma película pero con distintos inventory\_id, se contarán varias veces. Si, por otro lado, alquila varias veces el mismo inventory\_id, con la función SUM se contarían varias veces.

Después de esto solo habría que seleccionar el máximo de la suma\_category y imprimir su id y nombre.